doi:10.11835/j.issn.1000-582X.2022.208

解决图着色问题的膜进化算法

郭 平,郭 宾
(重庆大学 计算机学院,重庆 400044)

摘要:图着色问题是图论中比较热门的NP难问题之一。针对该问题,有许多启发式求解算法,但都存在求解的质量不高,计算时间较长等问题。近些年提出的膜进化算法,在处理NP难问题中展现出了独特的优势。基于膜进化算法框架,提出了解决图着色问题的膜进化算法,把图着色问题和膜结合,设计了复制、融合、分裂、溶解、融合分裂、禁忌搜索6种膜进化算子。这些算子在演变的过程中使膜和膜结构发生进化,从而找到更优解,最后求得解决方案。在DIMACS的40个挑战数据集上面进行了实验,与3个最新的图着色算法比较的结果表明:在保证解的质量的情况下,文中提出的膜进化算法能有效降低求解的时间,其中有58%的实例占优。

关键词:图论;组合优化;NP难问题;图着色问题;膜进化算法
中图分类号:TP301.6 **文献标志码:**A **文章编号:**1000-582X(2023)07-023-13

A membrane evolutionary algorithm for solving graph coloring problem

GUO Ping, *GUO Bin*
(College of Computer Science, Chongqing University, Chongqing 400044, P. R. China)

Abstract: The graph coloring problem is one of the popular NP-hard problems in graph theory. Various heuristic algorithms have been proposed to solve this problems; however, they often suffer from poor quality and long computation time. In recent years, the membrane evolutionary algorithm has shown unique advantages in dealing with NP-hard problems. Based on the membrane evolutionary algorithm framework, this study proposed a membrane evolutionary algorithm to solve the graph coloring problem. Six membrane evolutionary operators, namely, copy, fusion, division, cytolysis, fusion-division, and tabucol were designed to facilitate the evolution of the membranes and membrane structures, leading to the discovery of more optimal solutions. Experiments were conducted on 40 challenging datasets from DIMACS, and the results were compared with three latest algorithms. The resalts show the proposed algorithm effectively reduces the computation time while maintaining the solution quality, outperforming the other algorithms in 58% of the instances.

Keywords: graph theory; combinatorial optimization; NP-hard; graph coloring problem; membrane evolutionary algorithm

收稿日期:2022-02-02 **网络出版日期:**2022-05-19
基金项目:重庆市自然科学基金资助项目(cstc2019jcyj-msxmX0622)。
Supported by Natural Science Foundation of Chongqing (cstc2019jcyj-msxmX0622).
作者简介:郭平(1963—),男,教授,主要从事生物计算、进化计算和人工智能研究,(E-mail)guoping@cqu.edu.cn。

图着色问题是一个经典的组合优化问题，它的应用十分广泛，例如：Wifi信号的分配问题[1]，车辆网络中的资源分配问题[2]，资源约束问题中的调度[3]等。同时，图着色问题和最大团问题、最大独立集问题都存在关联，所以研究图着色问题在理论和实践方面都有很大的意义。作为NP难问题，可行的求解图着色问题的算法主要是启发式算法。经典算法包括混合进化算法[4]及其变体[5-6]。近些年涌现了一些新的方法来解决图着色问题，比如通过利用邻接矩阵的对角线分配颜色的算法[7]。将组合优化问题转换为泛函优化问题来解决[8]。随着膜计算的兴起，利用膜结构与遗传算法的结合来解决图着色问题[9]。针对量子计算设备的优缺点，改进量子优化算法来解决图着色问题[10]。用2个策略改进的二元蜻蜓算法来解决图着色问题[11]，离散自私群优化算法[12]，基于概率学习的局部搜索算法[13]，基于种群的梯度下降权重学习算法[14]等来解决图着色问题。

仿生群智能算法用于求解NP难问题一直都是比较热门的话题。膜计算是自然计算中一个快速发展的分支，近几年提出的膜进化算法，已经用来解决了很多NP难问题。例如：郭平等利用膜进化算法来解决最小顶点覆盖问题[15]、最大团问题[16]、TSP问题[17]、双目标关键节点检测问题[18]和容量受限聚类问题[19]。

针对图着色问题，结合进化算子和局部搜索的混合进化算法一直是研究的热点。这些算法中，由于引入了较多的个体，在管理多样性时存在很大的挑战。在文献[5]中，介绍了2种算法来解决图着色问题：一种是加了精心设计的多样性管理方法，另一种没有加多样性管理办法，前者的结果更优。膜进化算法是进化算法中的一种，它根据问题来设计膜对象，膜进化算子和膜结构。利用膜进化算法中并行进化机制和膜进化算子之间的协同机制，有可能获得更高质量的候选解并且缩短求解的时间。所以文中将膜进化算法应用到解决图着色问题中。实验结果表明，这个方法能够得到一个良好的结果。

1 相关工作

1.1 图着色问题和膜进化算法

图着色问题是给定一个无向图 $G = (V, E)$，其中 V 是顶点集合，E 是边集合。给每个顶点分配一个颜色数，使得有边相连的顶点拥有不同的颜色数。如果边 e 连接的2个顶点 v_1 和 v_2 的颜色数相同，称边 e 为冲突边，顶点 v_1 和 v_2 为冲突顶点。一个解决方案是给所有的顶点分配一个不大于 K 的颜色数并且无冲突。求得最小的整数 K，称之为色数。

膜进化算法是根据膜进化的过程提出的仿生智能算法[15]，膜进化算子抽象于膜进化过程中的行为，只要能在细胞活动中找到依据的行为都可以抽象成膜进化算子。膜进化算法最主要的特点是这些算子可以单独使用或者根据问题的需求把不同的算子进行组合使用。在文献[15]中，针对最小顶点覆盖问题，设计出融合、分裂、选择、溶解算子来解决。在文献[17]中，设计了针对旅行商问题的膜进化算子，并将选择和溶解算子进行组合使用。文献[15]给出了膜进化算法的基本框架。

1.2 图着色问题的典型算法

图着色问题是经典的NP难问题，研究者已经提出了许多的求解算法。近年来，性能优秀的算法包括PLSCOL(improving probability learning based local search for graph coloring)、TensCol(population-based gradient descent weight learning for graph coloring problems)和HEAD(variations on memetic algorithms for graph coloring problems)。

PLSCOL[13]是一种基于概率学习的局部搜索算法，该算法包括3个阶段：基于概率矩阵的起始着色阶段、启发式着色改进阶段和基于学习的概率更新阶段。该方法维护一个动态更新的概率矩阵，这个矩阵中的每个顶点会有一个指定属于每个颜色组的概率。为了避免对称解带来的困难，使用了种群匹配过程来识别初始解和其改进解之间的关系，在优化解的阶段采用流行的禁忌搜索算法。

TensCol[14]是一种基于种群的权重概率学习算法，它把求解搜索表述为一个连续权值张量优化过程。首先，使用权重矩阵表示候选颜色，对矩阵中每个元素对应于一个顶点接受特定颜色的学习倾向，然后使用梯度下降的方法，设计最小化全局损失函数引导梯度下降来改进解决方案。

HEAD[5]是一种混合进化算法，该算法结合交叉算子和禁忌搜索算法。禁忌搜索算法是一种局部搜索算法，交叉算子提供了一种良好的全局搜索能力。先利用交叉算子生成2个不同的解，再用禁忌搜索来提升这2个解的质量，最后用这2个提升的解去替换原来的2个解。如此往复，解的质量得到不断提升，并在固定的

进化代数引入精英来管理种群的多样性。

2　膜进化算法 MEA_GCP

根据膜进化算法框架,讨论提出的解决图着色问题的膜进化算法 MEA_GCP(membrane evolutionary algorithm for solving graph coloring problem)。

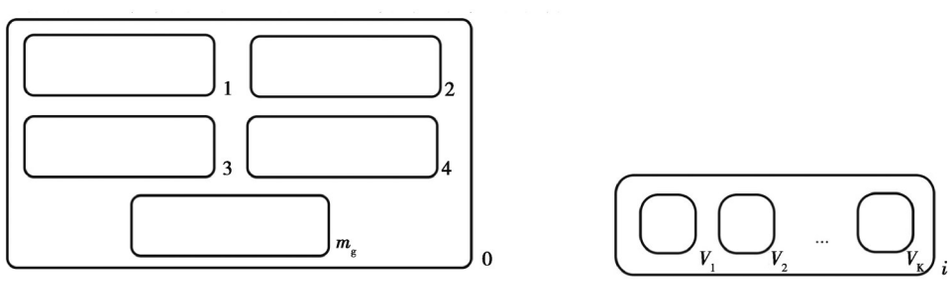
2.1　初始膜结构

为了方便描述,采用以下符号。

M:膜结构,它包含 4 个膜个体和 1 个全局最优膜 m_g。0 代表皮肤膜,如图 1(a)所示。

m_i (i =1,2,3,4):代表一个膜个体,如图 1(b)所示。为了简化,在图中用阿拉伯数字来表示膜个体,一个膜个体构成一个解决方案,它由 K 个子膜构成。

V_i ($1 \leqslant i \leqslant K$):代表一个子膜,存放一个解决方案中颜色为 i 的所有顶点。每个顶点只能有一种颜色,所以 $V_i \cap V_j = \varnothing\ (i \neq j)$, $V = V_1 \cup V_2 \cup \cdots \cup V_K$。$V_i$ 中的对象(物质)表示为 X_i^j, i 表示膜 i, j 表示颜色 j。特别,当只在一个膜中讨论问题时,省略对象的下标。



(a)初始膜结构 M　　(b)膜 i 的膜结构

图 1　解决图着色问题的膜进化算法的膜结构

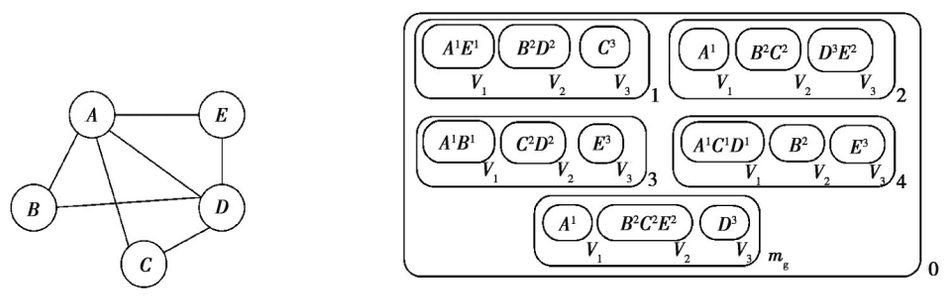
Fig. 1　Membrane structure of MEA_GCP

基于图 1(b)所示的膜结构,采用式(1)来评价膜的优劣,称为适应度函数。显然,只有当 $f_{\text{it}}(m) = 0$ 时表示该膜是一个解决方案, $f_{\text{it}}(m) \neq 0$ 时代表该膜是一个候选解决方案。当 $f_{\text{it}}(m_1) < f_{\text{it}}(m_2)$ 代表 m_1 的适应度函数更小,即膜 m_1 比 m_2 更好。

$$f_{\text{it}}(m) = \sum_{V_i \in V} \sum_{u,v \in V_i} \delta_{uv}, \tag{1}$$

式中, $\delta_{uv} = \begin{cases} 1, \langle u, v \rangle \in E, \\ 0, \text{otherwise}. \end{cases}$

图 2 给出了一个例子。图 2(a)为图 G,顶点为 $\{A,B,C,D,E\}$,色数 K 为 3。图 2(b)是一个膜结构。膜 1、2、3、4 分别为图 G 的候选解决方案。m_1 中包含有 V_1、V_2、V_3 三个子膜,顶点的颜色分别为:A 和 E 为颜色 1, B 和 D 为颜色 2, C 为颜色 3。按照式(1),可以求出 m_1 的适应度函数值为 2, m_2 的适应度函数值为 1。所以 m_2 比 m_1 要好。m_g 的适应度函数为 0,它是图 G 的一个解决方案。



(a)图 G　　(b)解决图 G 的初始膜结构

图 2　初始膜结构的一个例子

Fig. 2　An example of an initial membrane structure

2.2 MEA_GCP算法

基于图1(a)的初始膜结构,解决图着色问题的膜进化算法见算法1,其膜进化算子包括:复制、融合、分裂、溶解、融合分裂和禁忌搜索算子(这些算子将在2.4节中详细讨论)。在算法1中,输入的参数为图G,色数K,禁忌搜索的迭代次数MaxIter,种群的大小P(文中设计的融合分裂算子需要两两组合,种群的数量应为偶数。在进化过程中,膜个体进行并行优化,需要多核处理器,结合实验环境,最终选定为4个),$\max_{gen}$为最大进化代数。

在算法1中,根据图1(a)的初始膜结构初始化膜种群,并初始化参数(第1行到2行)。第3行是算法的结束条件,没有达到最大的进化代数并且没有找到解就一直执行第4到13行。融合分裂操作需要2个膜,所以将4个膜随机分为2组(第4行)。因为要对每组膜进行2次融合分裂操作,所以要先将每组膜进行1次复制操作(第5行)。2组膜分别进行2次融合分裂操作之后会形成4个膜(第6到7行)。把这4个膜经过禁忌搜索进行局部提升之后,再将这4个膜全部替换原来的4个膜(第9行),同时更新全局最优的膜m_g(第10到11行)。再把进化代数加1,这就完成了一次完整的进化过程。进化结束时,输出最优膜m_g。

算法1:MEA_GCP//解决图着色问题的膜进化算法

输入:$G=(V,E)$, K, MaxIter, $P=4$, $\max_{gen}$;
输出:最优膜m_g;
M = PopulationInitialization (G,K,P); // 2.3节
gen = 0, $m_g = m_1$, M.fusion(m_g); // fusion操作见图5(b)
while(gen < $\max_{gen}$ && $f_{it}(m_g)>0$){
把4个膜随机分成2个组g_1, g_2;// 图3(b)
复制g_i获得g_i',i = 1,2;//把每个组里的膜分别进行1次复制操作,得到4组膜,2.4.1节和图3(c)
m_1' = fusion_division(g_1, 0), m_2' = fusion_division(g_1', 1);// fusion_division见2.4.5节
m_3' = fusion_division(g_2, 0), m_4' = fusion_division(g_2', 1);
for $1 \leqslant i \leqslant P$ do
m_i =Tabucol(G, m_i', MaxIter); //2.4.6节
if ($f_{it}(m_g)>f_{it}(m_i)$)
$m_g = m_i$;
end for
gen++;
}
end while;
返回 m_g;

解决图着色问题的膜进化算法的膜结构变化如图3所示。

由图3可以看出,在解决图着色问题中,膜结构在进化的过程中会变化。初始膜结构为图3(a),把4个膜随机分为2组之后,膜结构变成图3(b)所示。再把每1组的膜经过复制算子复制1次,得到图3(c)的膜结构。在禁忌搜索的过程中,膜结构如图3(d)所示。

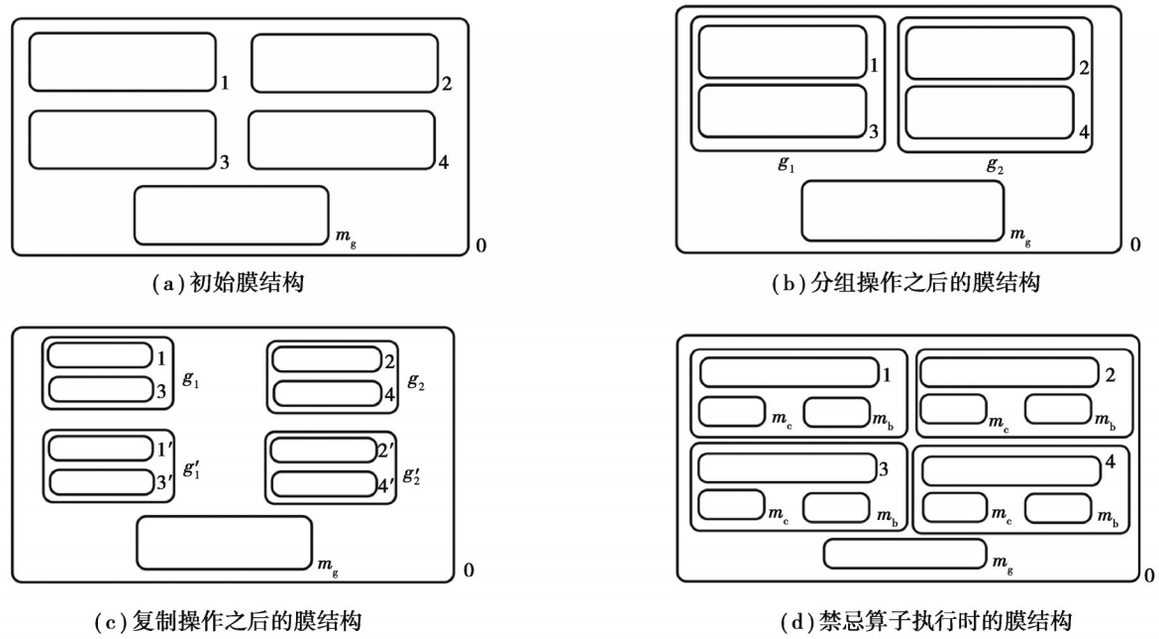
在种群的一代进化中,膜结构的变化过程可以总结为:图3(a)→图3(b)→图3(c)→图3(a)→图3(d)→图3(a)。

膜进化框架和MEA_GCP相比较,它们有3个主要区别。

1)膜结构方面的区别。对于MEA_GCP,膜主要有4种结构,膜种群为4个,在进化过程中,膜结构根据需求在一直变化。对于MEA的膜结构,给出了初始膜结构,膜种群的大小是根据不同的问题设定。

2)膜进化算子的区别:MEA_GCP的膜进化算子,分别为:复制算子、融合算子、分裂算子、溶解算子、融

合分裂算子、禁忌搜索算子。MEA的进化算子有:融合算子、分裂算子、选择算子、溶解算子。



(a)初始膜结构　(b)分组操作之后的膜结构

(c)复制操作之后的膜结构　(d)禁忌算子执行时的膜结构

图3　解决图着色问题的膜结构变化图

Fig. 3　Membrane structure change diagram of MEA_GCP

3)适应度函数的区别:MEA_GCP的适应度函数是针对图着色问题的一个具体衡量指标。MEA中的适应度函数也用来衡量一个膜的优劣,没有针对具体问题,是一个比较抽象的概念。

膜进化算法是一个基础的框架,解决图着色问题的膜进化算法是在它的基础上针对图着色问题设计的算法。针对不同的问题,也要在该框架上面进行设计。

2.3　种群初始化

在膜种群的初始化过程中,每个膜的初始化原则是保证在较短的时间得到一个适应度函数比较小的初始解。算法2给出了膜种群初始化的伪代码。

算法2:PopulationInitialization //初始化膜种群

输入:$G = (V, E)$, K, P;

输出:the membrane population M;

```
创建一个空的皮肤膜;
for i = 1 to P do
  m_i = Random_init(G, K);   //算法 3
  M.fusion(m_i);   //图 5(b)
end for
返回 M;
```

Random_init(算法3)完成种群的随机初始化。先根据色数创建K个空的子膜(1到4行)。对于每个顶点随机分配一个1到K的颜色数(第6行),再对应颜色数把顶点分配到不同的子膜中,形成一个膜个体(第7到10行)。

算法 3：Random_init //随机初始化

输入：$G = (V, E)$, K;

输出：m;

```
for(1≤i≤K)
    创建一个空的子膜 V_i;
    m.fusion(V_i);    //图 5(c)
end for
for v∈V do
    C[v] = random(1, K); //C[v]代表 v 的颜色，随机给 v 赋予一个 1 到 K 的颜色数
    if(C[v] == i)
        V_i←{v};
end for
返回 m;
```

2.4 进化算子

结合一个顶点为{A,B,C,D,E,F,G,H,I,J}，色数为 3 的图对进化算子做详细的介绍。

2.4.1 复制算子

复制算子就是将一个膜复制成为 2 个膜，复制之后，这 2 个膜中的子膜也应该是相同的，如图 4 所示。

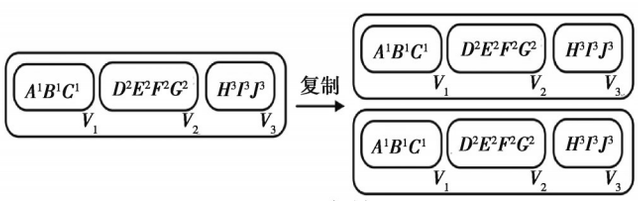


图 4 细胞复制示例

Fig. 4 Copy operator example

2.4.2 融合算子

融合算子有 3 种操作：1）将 2 个或者 2 个以上的膜合并形成一个膜的过程。如图 5(a)，将 2 个膜里面的子膜合并在同一个膜中(为了对融合后的子膜方便描述，把融合后的子膜记为 V_{ij}，其中 i 代表原膜 i，j 代表子膜 j。如 V_{21} 代表原膜 2 中的子膜 1，V_{12} 代表原膜 1 中的子膜 2)。2）把一个膜融合到皮肤膜中，如图 5(b)所示。3）子膜和膜的融合，如图 5(c)所示。

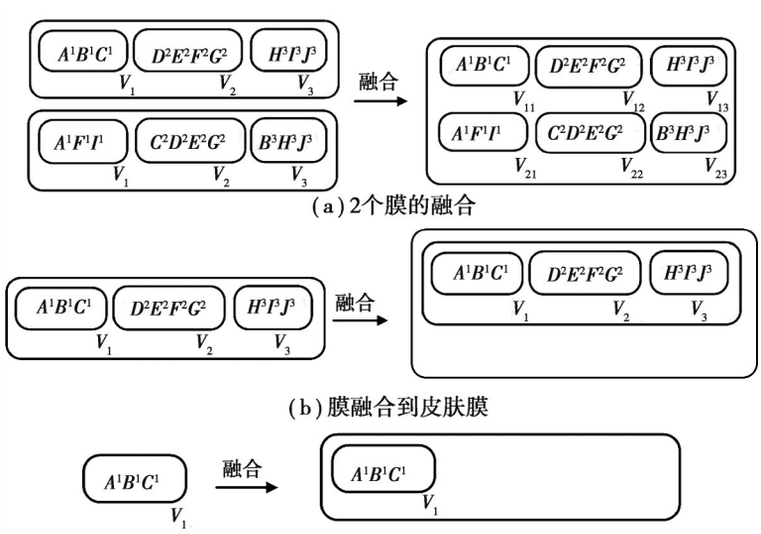


图 5 细胞融合示例

Fig. 5 Fusion operator example

2.4.3　分裂算子

分裂算子是把膜中顶点按照一定的规则分裂在不同膜中。分裂过程中,子膜中顶点的颜色不改变,如图6所示。

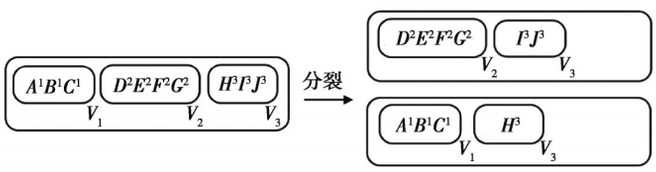


图6　细胞分裂示例

Fig. 6　Division operator example

2.4.4　溶解算子

在进化的过程中,当一个膜表达的解决方案已经远离最优解时,需要将该膜内的对象以及膜都删除,这时就需要进行溶解。溶解算子的作用在于删除膜以及膜内的对象。

在MEA_GCP中,不单独使用溶解算子,而是将它与其他算子一起使用。例如,在融合分裂算子结束后,会得到2个膜,其中1个膜是比较差的,所以要将这个膜溶解掉,就用到溶解算子。

2.4.5　融合分裂算子

在膜进化算法中,算子不仅可以单独使用,而且可以组合使用。在算法4中,就是将融合与分裂算子结合起来使用实现融合分裂功能。

算法4:fusion_division //融合分裂算子

输入:m_1, m_2, f; //膜1、膜2、标志f(0 or 1)

输出:m_1'; //新的膜1

　把m_1和m_2通过融合算子融合成m_t; //2.4.2节

　创建一个空膜m_1';

　for $1 \leqslant L \leqslant K$ do

　　if $(L+f)$% 2 is odd, then A=1,else A=2;

　　选择m_t中顶点最多的子膜V_{Aj} $(1 \leqslant j \leqslant K)$,把该子膜中的所有顶点组合成子膜$V_L$融合到$m_1$'中,然后把$m_t$中与子膜$V_{Aj}$中相同的顶点分裂到$m_2$'中;

　　L++;

　end for

将m_t中还未分裂的顶点对应分裂到m_1'和m_2'中(V_{1j}分裂到m_1',V_{2j}分裂到m_2'中);

将m_2'溶解;

返回m_1';

在图7中,给出了融合分裂算子的具体分裂过程,这是解决图着色问题的关键。算法4中,第1行将2个膜融合为1个膜m_t。第4到5行,算法会间隔选择顶点最多的子膜融合到m_1'中。如图7(a),首先,选择子膜V_{12}(D,E,F,G)融合到m_1'中,再把m_t中与子膜V_{12}相同的顶点分裂到m_2'中;然后,选择子膜V_{23}(B,H,J)融合到m_1'中,接着,把m_t中与子膜V_{23}相同的顶点分裂到m_2'中,如图7(b)。直到循环结束后,把还没有分裂的顶点分裂到m_1'和m_2'中,如图7(d),把V_{13}分裂到m_1'的V_3中,把V_{21}分裂到m_2'的V_1中。最后,把m_2'溶解,就得到了m_1'。

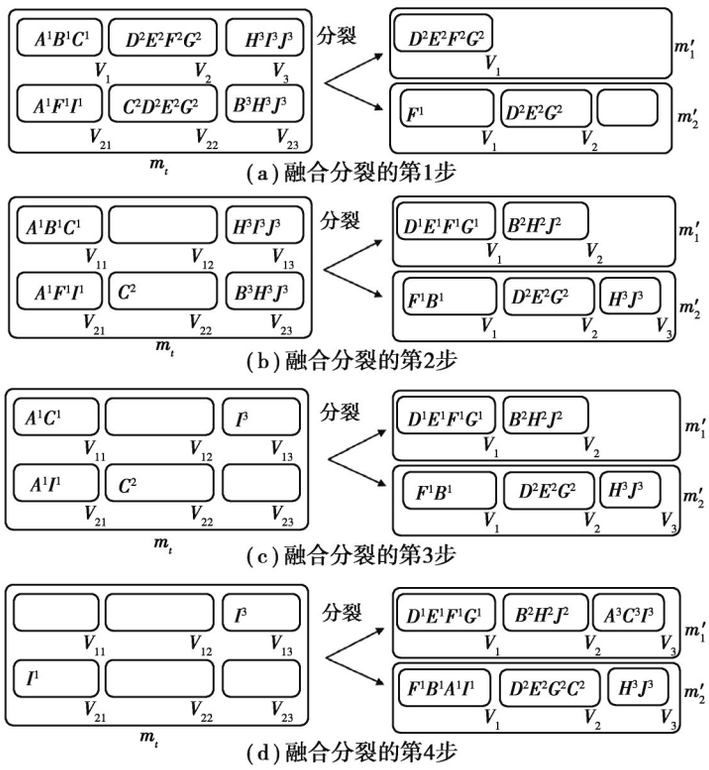


图 7 融合分裂过程的一个例子

Fig. 7 An example of the fusion-division operator

2.4.6 禁忌搜索算子

如果单独只用进化算法去解决图着色问题,其结果比较差。比较有效的做法是把局部搜索算法与进化算法相结合组成混合进化算法:进化算法的算子用来提供种群的多样性,提供一个更广的搜索空间;局部搜索用来做局部提升,能在一个基础上找到一个更好的解。文中用到的局部搜索算法是禁忌搜索。

禁忌搜索[20]自 1987 年提出以来得到了广泛的应用,算法 5 给出了文中的禁忌搜索算子的具体步骤。文中提出的禁忌搜索引入了标记膜 m_c,标记膜的膜结构如图 8(a)所示,膜内对象为顶点,顶点右上角的数字代表顶点移动到的子膜的标记,右下角的 t 代表在 t 次迭代内不能做同样的移动操作。图 G,膜 m 和最大迭代的次数 MaxIter 作为算法的输入。迭代的过程中首先将输入的膜 m 作为当前最优的膜,并重置迭代次数,设置一个空的标记膜,把膜内顶点的右上下标设置为 0(第 1 到 2 行)。第 3 行是终止条件。在第 4 行,尝试把顶点 v 从 V_i 移动到 V_j 使适应度函数变小。如果标记膜中该 v 的 t 为 0,则代表该移动是被允许的(第 5 行)。在膜 m 中,就执行该移动。在标记膜 m_c 中,把这个移动记录下来(第 6 行),并且更新最优膜(第 7 行)。如果该移动不被允许,那就将该移动的禁忌次数减 1(第 9 行)。在第 12 行,达到迭代次数后,输出最优的膜。

图 8 标记膜示例

Fig. 8 Mark membrane example

在禁忌搜索当中,参数 t 叫禁忌期。它的大小直接影响了对邻域的搜索效果。禁忌期较长可以探索大的搜索空间,但是如果设置得太长则不能起到禁忌作用;如果禁忌期较短,搜索将被限制在一个较小范围,不易获得更优的解。文献[21]中采用式(2)以半随机的方式动态生成禁忌期有很强的鲁棒性。

$$t=\mathrm{random}(A)+\partial * f_{\mathrm{it}}(m), \tag{2}$$

式中:A 的区间是[0,9];∂ = 0.6;$f_{\mathrm{it}}(m)$代表当前膜的适应度函数。在文中,也采用这样的设置方式。在加入标

记膜之后，膜结构会发生变化，具体见图3(d)所示。

算法5：Tabucol //禁忌搜索算子

输入：G, m, MaxIter;

输出：m_b;

```
Set m_b = m and iter = 0; //m_b代表迭代过程的最优膜
设置一个空的标记膜 m_c;//如图8(b)所示
while iter != MaxIter do
  尝试把 m 中的 v 从 V_i 移动到 V_j 中;//v 代表 m 中的任意一个顶点
  if(m_c 中 v 的 t 为 0)
    在 m 中，把 v 从 V_i 移动到 V_j 中。在 m_c 中，把 v 的移动标记设置为 j，并按照公式2重置 t;
    if f_it(m) < f_it(m_b), then set m_b=m;
  else
    在 m_c 中，把 v 的 t 减 1;
  iter ++;
end while
返回 m_b;
```

3　对比实验

在这一节中，介绍所选用的基准实例和实验的环境，参数设置和对比实验。为了验证MEA_GCP的有效性，将和现在已知最好的和最新的算法作对比。

3.1　实例和实验环境

本次选用的基准实例主要是第二次DIMACS竞赛中的40个困难实例，这些实例在近年的研究中被广泛使用。

MEA_GCP算法的实现语言是C++，实验环境是windows10，处理器是：AMD Ryzen5 4600H六核，主频3GHz。

为了体现实验对比的客观性，下面将对比实验的实验环境也列出来做参考。

1）PLSCOL[13]：基于概率学习的局部搜索算法，用的语言是C++，在Intel Xeon E5-2760 2.7 GHz的处理器上实现，结束时间为5 h。

2）TensCol[14]：是基于种群的权重概率学习。用的语言是Python，在GPU设备上运行，利用英伟达RTX2080Ti显卡和12GB的内存。

3）HEAD[5]：最新的混合进化算法。用的语言是C++，并行运行在4核3.1GHz的Intel Xeon处理器。

在文中的实验对比情况中，比较2个算法哪个较优是采用先比较色数，再比较成功率，再比较求解时间的顺序评价。因为更新一个图的色数是非常困难的，所以先比较色数，其次是成功率，成功率代表着该算法是否比较稳定，鲁棒性是否比较强。同时，求解时间的长短也是比较重要的一个指标。

3.2　与其他算法的对比

关于图着色问题的色数，已经很多年没有出现过更新，说明想要取得更好的结果的难度是巨大的。文中选择与HEAD[5]、PLSCOL[13]和TensCol[14]算法进行对比，它们是近几年来求解成功率和求解效率都是优秀的算法。需要说明的是，对于对比算法，笔者没有采用文中实现的这些算法的结果，而是直接使用了相关文献中的结果。原因在于笔者实现这些算法所获得的结果中的大多数没有参考文献中的结果优秀。

表1是MEA_GCP和3个算法的对比结果，其中最好的结果用粗体表示，‘-’表示在相应的文献中没有给出对应实例的计算结果。第1列为实例的名称，第2列K*代表已知的最小色数，后面4大列分别为4种算法的结果。k_{best}代表用该算法出的最优色数，S_R代表成功率(成功次数/总共运行的次数)，$t(s)$为算法的平均时间，单位为s。

表1 MEA_GCP、PLSCOL、TensCol、HEAD实验结果对比

Table 1 Experimental results of MEA_GCP, PLSCOL, TensCol and HEAD

实例名	K*	PLSCOL			TensCol			HEAD			MEA_GCP		
		k_{best}	S_R	t/s	k_{best}	S_R	t/s	k_{best}	S_R	t/s	k_{best}	S_R	t/s
DSJC125.1	5	5	10/10	<60	5	10/10	40	5	20/20	<0.6	**5**	**10/10**	**0.02**
DSJC125.5	17	17	10/10	<60	17	10/10	68	17	20/20	<0.6	**17**	**10/10**	**0.11**
DSJC125.9	44	44	10/10	<60	44	10/10	22	44	20/20	<0.6	**44**	**10/10**	**0.07**
DSJC250.1	8	8	10/10	<60	8	10/10	95	8	20/20	<0.6	**8**	**10/10**	**0.12**
DSJC250.5	28	28	10/10	4	28	10/10	199	**28**	**20/20**	**0.6**	28	10/10	1.80
DSJC250.9	72	72	10/10	<60	72	10/10	87	**72**	**20/20**	**1.2**	**72**	**10/10**	**1.20**
DSJC500.1	12	12	7/10	43	12	10/10	1 098	12	20/20	6	**12**	**10/10**	**4.80**
DSJC500.5	47	48	3/10	1786	48	5/10	7 807	**47**	**2/ 10 000**	**48**	48	10/10	11.40
DSJC500.9	126	**126**	**10/10**	**747**	126	6/10	18 433	126	13/20	72	126	3/10	20.40
DSJC1000.1	20	20	1/10	3 694	20	10/10	10 225	20	20/20	12	**20**	**10/10**	**9.00**
DSJC1000.5	82	87	10/10	1 419	84	9/10	32 495	**82**	**3/20**	**2 880**	83	10/10	215.40
DSJC1000.9	222	223	5/10	12 094	224	6/10	58 084	**222**	**2/20**	**5 160**	223	8/10	637.80
DSJR500.1	12	12	10/10	<60	12	10/10	7	-	-	-	**12**	**10/10**	**0.18**
DSJR500.1c	85	85	10/10	386	**85**	**10/10**	**298**	85	1/20	12	85	3/10	0.60
DSJR500.5	122	126	8/10	1 860	**122**	**10/10**	**4 310**	-	-	-	122	9/10	106.20
flat300_26_0	26	26	10/10	195	26	10/10	176	-	-	-	**26**	**10/10**	**0.60**
flat300_28_0	28	**30**	**10/10**	**233**	31	10/10	586	31	20/20	1.2	31	10/10	1.20
flat1000_76_0	81	86	1/10	5 301	83	3/10	34 349	**81**	**3/20**	**3600**	82	8/10	409.80
flat1000_50_0	50	-	-	-	-	-	-	50	20/20	18	**50**	**10/10**	**15.60**
flat1000_60_0	60	-	-	-	-	-	-	60	20/20	30	**60**	**10/10**	**27.00**
latin_square_10	97	99	8/10	2 005	**98**	**10/10**	**28 925**	-	-	-	100	6/10	1 240.00
le450_5a	5	-	-	-	-	-	-	5	20/20	<0.6	**5**	**10/10**	**0.19**
le450_5b	5	-	-	-	-	-	-	5	20/20	<0.6	**5**	**10/10**	**0.24**
le450_5c	5	-	-	-	-	-	-	5	20/20	<0.6	**5**	**10/10**	**0.20**
le450_5d	5	-	-	-	-	-	-	5	20/20	<0.6	**5**	**10/10**	**0.19**
le450_15a	15	15	10/10	<60	15	10/10	333	15	20/20	<0.6	**15**	**10/10**	**0.27**
le450_15b	15	15	10/10	<60	15	10/10	333	15	20/20	<0.6	**15**	**10/10**	**0.23**
le450_15c	15	15	7/10	1 718	**15**	**10/10**	**507**	15	3/20	1.2	15	4/10	1.20
le450_15d	15	15	3/10	2 499	**15**	**10/10**	**301**	15	1/20	1.8	15	1/10	1.20
le450_25a	25	25	10/10	<60	25	10/10	87	25	20/20	<0.6	**25**	**10/10**	**0.19**
le450_25b	25	25	10/10	<60	25	10/10	12	25	20/20	<0.6	**25**	**10/10**	**0.19**
le450_25c	25	25	10/10	1 296	25	10/10	19 680	25	20/20	1800	**25**	**10/10**	**792.00**

续表 1

实例名	K^*	PLSCOL			TensCol			HEAD			MEA_GCP		
		k_{best}	S_R	t/s	k_{best}	S_R	t/s	k_{best}	S_R	t/s	k_{best}	S_R	t/s
le450_25d	25	**25**	**10/10**	**1 704**	25	10/10	9 549	25	20/20	5400	25	8/10	4 753.20
R125.1	5	5	10/10	<60	5	10/10	0	-	-	-	**5**	**10/10**	**0.02**
R125.5	36	36	10/10	<60	**36**	**10/10**	**6**	-	-	-	36	7/10	0.60
R250.1	8	8	10/10	<60	**8**	**10/10**	**0**	-	-	-	8	10/10	0.07
R250.5	65	66	10/10	705	**65**	**10/10**	**33**	65	1/20	780	66	7/10	627.00
R1000.1	20	20	10/10	<60	20	10/10	15	-	-	-	**20**	**10/10**	**0.20**
R1000.1c	98	98	10/10	256	98	10/10	4 707	98	3/20	12	**98**	**10/10**	**27.00**
R1000.5	234	254	4/10	7 818	**234**	**2/10**	**23 692**	245	20/20	14640	245	3/10	14 400.00

从表 1 可以看出,MEA_GCP 更具有优势,具体来说:

1)在最优解方面,PLSCOL 在 34 个实例中的获得了 1 个比其他 3 个算法更好的解,TensCol 在 34 个实例中获得了 2 个比其他 3 个算法更好的解,HEAD 在 32 个实例中获得了 4 个比其他 3 个算法更好的解,MEA_GCP 在 40 个实例中有 31 个实例与其他 3 个算法的最优解相同。

2)在求解成功率方面,PLSCOL 在 34 个实例中的获得了 1 个比其他 3 个算法高的成功率,TensCol 在 34 个实例中获得了 4 个比其他 3 个算法更高的成功率,HEAD 在 32 个实例中获得了 1 个比其他 3 个算法更高的成功率,MEA_GCP 的 40 个实例中有 3 个实例比其他 3 个算法的成功率高。

3)在时间消耗方面,PLSCOL 在 34 个实例中的获得了 1 个比其他 3 个算法更少的时间,TensCol 在 34 个实例中获得了 2 个比其他 3 个算法更少的时间,HEAD 在 32 个实例中获得了 2 个比其他 3 个算法更少的时间,MEA_GCP 的 40 个实例中有 33 个实例比其他 3 个算法的时间少。

总体来说,MEA_GCP 在保证解的质量相当的情况下有效降低了求解的时间,其中有 58% 的实例具有更少的求解时间。这主要源于进化机制不同与计算策略不同 2 个方面。

1)进化机制不同。MEA_GCP 中把 4 个膜分成两两一组,然后分别对每个组的膜进行融合、分裂操作。经历融合、分裂之后得到的膜通过禁忌搜索算子并行提升之后再去替换原来的 4 个膜,再进行分组操作,继续进化。HEAD 利用交叉算子和禁忌搜索的结合,整个种群是 2 个解在并行进化,2 个解利用交叉算子生成不同的个体,再用禁忌搜索进行提升后替换原来的 2 个解。PLSCOL 是改进的禁忌搜索算法,整个进化过程主要是围绕着 1 个解进行计算,解的改进过程可以概括为:初始解→局部最优解→全局最优解。TensCol 是基于种群的梯度下降权值学习方法,种群的大小为 200,把图着色问题转化为连续权值张量优化问题,建立每个个体和种群的关系,每个个体又在并行进化,一次进化结束,所有的解会通过全局损失函数整合最好的解。对于 HEAD、MEA_GCP 的种群个体数较多,在进行融合分裂时,把 2 个膜分为两两一组,每次都会有 3 种不同的选择,而 HEAD 只有 2 个个体,没有不同的选择,所以 MEA_GCP 个体的多样性会更好,求解的效率更高。PLSCOL 求解只针对 1 个个体进行改进,本质是一种局部搜索算法。MEA_GCP 有 4 个个体,个体之间会互换信息,它的全局搜索能力更好,所以用的时间较少。TensCol 的种群大小为 200,这 200 个个体并行经历一次进化后,都会去计算各个解之间的关系,然后整合关系去寻找最优解。而 MEA_GCP 只需要比较每个膜的适应度函数就能判断哪个膜更优,所以 MEA_GCP 有更高的效率。

2)计算策略不同。HEAD 是混合进化算法,其中采用交叉算子来提供全局搜索能力。MEA_GCP 是膜进化算法,其中的融合分裂算子在设计时用到了交叉算子的贪心策略,但是融合分裂算子的最后一步做了改进。交叉算子在最后一步是采用随机着色的方式,这会使解产生较多的冲突。而融合分裂算子在最后分裂时,顶点的颜色和原来的膜相同,比随机着色的产生的冲突要少。PLSCOL 算法主要是围绕着一个概率矩阵对禁忌搜索算法进行改进来解决图着色问题。概率矩阵贯穿着色的整个过程,从起始着色到着色改进,再到概率更新,还需要计算解之间的关系。这些步骤都需要大量的时间,并且计算时间和图的规模成正比。再加

上禁忌搜索的时间，PLSCOL的计算时间会更长。对于MEA_GCP，计算时间的99%都是由禁忌搜索产生，融合分裂所花费的时间非常少。所以MEA_GCP在大规模实例上的计算时间占较大优势。对于TensCol算法，它主要是把图着色问题转化为了连续权值张量的优化问题，该算法是解决图着色问题的一个通用框架。不仅仅针对文中提出的图着色问题，还可以解决ECP（equitable graph coloring problem）问题，它是在GCP问题上加上每种颜色的顶点个数相差不大于1的条件。1个方法解决2个问题，在计算全局损失函数时，针对GCP，会进行一些不必要的计算，所以消耗了大量的时间。

由于表1中各算法运行的实例数不同，表2给出了各算法获得最优解实例数的百分比。从表2可知，MEA_GCP获得的最优解比例远高于对比算法。

表2 实验结果百分比统计

Table 2 Percentage statistics of experimental results

算法名称	最优实例数/实例总数	最优实例比例/%
PLSCOL	3/34	9
TensCol	9/34	26
HEAD	6/32	19
MEA_GCP	**23/40**	**58**

4 结束语

膜进化算法是受膜进化过程启发的仿生智能算法，在膜进化演变的过程中寻找解。依据膜进化算法的框架，结合图着色问题，文中提出了MEA_GCP算法，设计并实现了该算法的复制、融合、分裂、融合分裂、溶解、禁忌搜索算子，并在DIMACS挑战数据集上与目前优秀的图着色问题求解算法进行了对比实验。MEA_GCP在78%的实例上与其他3个算法取得相同的结果，在70%的实例上获得了100%的求解成功率，在58%的实例上具有更少的求解时间。由此，使用膜进化算法解决图着色问题是有效的和可行的。

文中在设计MEA_GCP的进化算子的时候主要借鉴了交叉算子的设计思路。进一步的研究可以从以下两方面展开：更好地结合禁忌搜索与膜对象进化使对象更新效率更高；简化膜内对象的表示和引入新的进化策略（例如对象贡献度）使进化算法具有通用性。

参考文献

[1] Orden D, Marsa-Maestre I, Gimenez-Guzman J M, et al. Spectrum graph coloring to improve Wi-Fi channel assignment in a real-world scenario via edge contraction[J]. Discrete Applied Mathematics, 2019, 263: 234-243.

[2] Gao L, Hou Y, Tao X, et al. A graph-based resource sharing and admission control for vehicular networks[C]// 2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW). IEEE, 2019, 1-6.

[3] Hsiao H C, Chen C W, Wang J, et al. Architecture-aware memory access scheduling for high-throughput cascaded classifiers [C]// 2019 IEEE 22nd International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS). IEEE, 2019：1-4.

[4] Galinier P, Hao J K. Hybrid evolutionary algorithms for graph coloring[J]. Journal of Combinatorial Optimization, 1999, 3(4): 379-397.

[5] Moalic L, Gondran A. Variations on memetic algorithms for graph coloring problems[J]. Journal of Heuristics, 2018, 24(1): 1-24.

[6] Lü Z, Hao J K. A memetic algorithm for graph coloring[J]. European Journal of Operational Research, 2010, 203(1):241-250.

[7] Negi C, Shukla A N. An approach for solving the graph coloring problem using adjacency matrix[C]// 2019 8th International Conference System Modeling and Advancement in Research Trends (SMART). IEEE, 2019: 10-13.

[8] Crnkić A, Povh J, Jaćimović V, et al. Collective dynamics of phase-repulsive oscillators solves graph coloring problem[J]. Chaos, 2020, 30(3): 033128.

[9] Andreu-Guzmán J A, Valencia-Cabrera L. A novel solution for GCP based on an OLMS membrane algorithm with dynamic operators[J]. Journal of Membrane Computing, 2020, 2(1): 1-13.

［10］Tabi Z, El-Safty K H, Kallus Z, et al. Quantum optimization for the graph coloring problem with space-efficient embedding [C]// 2020 IEEE International Conference on Quantum Computing and Engineering (QCE). IEEE, 2020: 56-62.

［11］Baiche K, Meraihi Y, Hina M D, et al. Solving graph coloring problem using an enhanced binary dragonfly algorithm[J]. International journal of swarm intelligence research, 2019, 10(3): 23-45.

［12］Zhao R, Wang Y, Liu C, et al. Discrete selfish herd optimizer for solving graph coloring problem[J]. Applied Intelligence, 2020, 50(5): 1633-1656.

［13］Zhou Y, Duval B, Hao J K. Improving probability learning based local search for graph coloring[J]. Applied Soft Computing, 2018, 65: 542-553.

［14］Goudet O, Duval B, Hao J K. Population-based gradient descent weight learning for graph coloring problems[J]. Knowledge-Based Systems, 2021, 212:106581.

［15］Guo P, Quan C, Chen H. MEAMVC: A Membrane evolutionary algorithm for solving minimum vertex cover problem[J]. IEEE Access, 2019, 7: 60774-60784.

［16］Guo P, Wang X, Zeng Y, et al. MEAMCP: A membrane evolutionary algorithm for solving maximum clique problem[J]. IEEE Access, 2019, 7: 108360-108370.

［17］Guo P, Hou M, Ye L. MEATSP: a membrane evolutionary algorithm for solving TSP[J]. IEEE Access, 2020, 8:199081-199096.

［18］Xu Y C, Guo P. MEA-CNDP: a membrane evolutionary algorithm for solving biobjective critical node detection problem[J]. Computational Intelligence and Neuroscience, 2021, 2021: 1-20.

［19］Liu Y Y, Guo P, Zeng Y. MEACCP: a membrane evolutionary algorithm for capacitated clustering problem[J]. Information Sciences, 2022, 591: 319-343.

［20］Hertz A, Werra D. Using tabu search techniques for graph coloring[J]. Computing, 1987, 39(4): 345-351.

［21］Fleurent C, Ferland J A. Genetic and hybrid algorithms for graph coloring[J]. Annals of Operations Research, 1996, 63(3): 437-461.

（编辑　詹燕平）