软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(10):2305−2316 [doi: 10.3724/SP.J.1001.2011.03969]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

采用染色划分改进的 RLS 算法及性能分析*

张 雁[1,2+], 焦方正[1], 卢昕玮[3], 黄永宣[1]

[1](西安交通大学 系统工程研究所,陕西 西安 710049)

[2](陕西电力信通公司 运行管理部,陕西 西安 710048)

[3](长安大学 经济与管理学院,陕西 西安 710064)

Improved RLS Algorithm with Color-Partition and Performance Analysis

ZHANG Yan[1,2+], JIAO Fang-Zheng[1], LU Xin-Wei[3], HUANG Yong-Xuan[1]

[1](Systems Engineering Institute, Xi'an Jiaotong University, Xi'an 710049, China)

[2](Department of Operation and Management, Shaanxi Power Information and Communication Corporation, Xi'an 710048, China)

[3](School of Economics and Management, Chang'an University, Xi'an 710064, China)

+ Corresponding author: E-mail: nowed@stu.xjtu.edu.cn, http://www.xjtu.edu.cn

Zhang Y, Jiao FZ, Lu XW, Huang YX. Improved RLS algorithm with color-partition and performance analysis. *Journal of Software*, 2011,22(10):2305−2316. http://www.jos.org.cn/1000-9825/3969.htm

Abstract: By exploiting the special structure in the solution space of the maximum clique problem (MCP), an improved RLS method, the RLS-II method, is has been created where the neighborhood-moving direction of the local search is guided by the multivariate constraints constructed by the dying partitioning. Therefore, the probability of the local search approaching the optimal solution is increased. Using the absorbing Markov chain theory as a reference, the mathematical models of the RLS and RLS-II algorithms in solving maximum clique problem are constructed and the absorbing time of the two algorithms is analyzed. Moreover, the analytical results are experimentally demonstrated on 77 Benchmark instances. Both the theoretical analysis and simulation results show that the dying partitioning filter can effectively improve the performance of the RLS algorithm. Furthermore, the longer average length of the dying group, the higher probability and amplitude of the performance improvement.
Key words: local search algorithm; maximum clique problem; coloring; absorbing Markov chain

摘 要: 利用最大团问题解空间特殊的结构特征,提出一种基于染色划分构建高维约束指导局部搜索移动方向的改进 RLS 算法——RLS-II 算法,该算法提高了局部搜索向最优解靠近的概率.基于吸收态 Markov 链理论,建立了 RLS 和 RLS-II 算法求解最大团问题的数学模型,分析了两种算法的吸收时间,并在 77 个标准测试算例上对分析结果进行了实验验证.理论分析及实验结果都表明,染色划分过滤确实能够有效改善 RLS 算法的性能,且平均染色组长度越大,性能改进的概率和幅度就越大.
关键词: 局部搜索算法;最大团问题;染色;吸收态马尔可夫链
中图法分类号: TP18 **文献标识码**: A

最大团问题在许多领域都有重要的应用,是最早从理论上获得证明的 NP-hard 问题之一[1].由于组合爆炸的

* 收稿时间: 2010-03-13; 修改时间: 2010-07-28; 定稿时间: 2010-11-03

存在,寻找最优解的精确算法所需的计算时间会随着顶点和边密度的增加而呈指数型地增长[2,3],无法用于大型例子的求解,所以近年来大部分研究都集中于启发式算法[4].

在众多启发式算法中,Batti 的自反应局部搜索算法 RLS 一经提出就广受关注[5],并被认为是求解最大团问题最成功的算法之一[6].继 RLS 之后提出的求解最大团问题的算法虽然很多,但其中性能最好的几个都采用了 RLS 算法的基本结构,如 Hansen 的变邻域搜索算法 VNS[7]、Katayama 的变深度局部搜索算法 KLS[8]、Pullan 和 Hoos 的动态局部搜索算法 DLS-MC[9]、Wayne Pullan 的阶段局部搜索算法 PLS[10]以及 Guturu 的 IEA-PTS 算法等[6].2009 年,研究最大团问题求解策略的算法 RDLS 依然是主要凭借了 RLS 算法的高效局部搜索机制而获得了好的结果[11].

在 RLS 的各种改进算法中,较常用的思路就是引入有关问题结构的信息引导搜索趋向优化.到目前为止,引入的导向信息基本上都集中于图的"度"上.pls 算法使用静态度作为引导信息[10],KLS 算法使用动态度作为引导信息[8],VNS 使用局部度作为引导信息[7].虽然从概率上分析,度较高的点入选最大团的概率更大,但这并不具有绝对的意义.在很多结构的图中,最大团是由那些度较小的顶点构成的,所以,用"度"信息定位最优解的正确性无法得到保证.

本文利用最大团问题解空间特殊的结构特征,提出了一种基于染色划分构建高维约束指导局部搜索的改进型 RLS 算法——RLS-II 算法.新算法通过优选局部搜索中最有希望的邻域,减少了局部搜索在无效区域中的停留时间,增加了局部搜索向最优解靠近的概率.在 77 个最大团问题 Benchmark 算例上的实验结果以及基于吸收态 Markov 链模型的理论分析均表明,所提出的染色划分过滤技术确实能够有效改善 RLS 算法的性能.

本文第 1 节介绍原 RLS 算法的基本思路.第 2 节在对最大团问题结构进行分析的基础上引入染色划分过滤技术,并给出基于染色划分过滤的 RLS-II 算法的流程.第 3 节是理论分析:首先分析染色划分过滤技术使局部搜索趋向最优解的概率提高的原因和条件;然后建立局部搜索过程的 Markov 链模型,给出 RLS 和 RLS-II 两种算法吸收时间的计算公式;并通过比较两种算法的吸收时间,分析影响算法性能改善效果的因素.第 4 节在 77 个最大团问题 Benchmark 算例上对 RLS-II 算法的性能进行测试,测试结果表明了染色划分过滤机制的有效性,并验证出理论分析的相关结论.最后一节是结论.

1 RLS 算法相关知识

设无向图 $G=(V,E)$,V 为 G 的顶点集合,E 为 G 的边集合.设顶点子集合 $K\subseteq V$,记图 $G(K)=(K,E\cap K\times K)$为由 K 推导出的图 G 的生成子图.若图 $G(K)$的顶点两两相连,则 K 为图 G 的一个团.在图 G 的所有团中,包含顶点最多的一个团就是图 G 的最大团,记为 C.

RLS 算法采用单个体迭代式的搜索,先随机生成一个团 K,然后通过各种操作不断地对团 K 进行改进,直到找到目标最大团.RLS 算法中的操作包括添加 Add、替换 Swap、跃迁 Leap 和重启 Restar 等.这些操作在禁忌策略的指导下相互配合,共同完成对最大团的全局搜索.有关 RLS 算法的细节请参阅文献[5].

RLS 算法成功地定义了适合于最大团问题的邻域结构和移动策略,但缺少对问题本身结构信息的挖掘和利用.在诸多 RLS 改进算法中,虽然引入了基于问题结构的"度"信息引导局部搜索,但由于"度"将高维结构信息压缩成一维数值表示,蕴含的信息量大为减少,难以实现对局部搜索正确而有效的引导[12].本文利用最大团问题的结构特点,通过染色划分的方法构建高维约束信息,帮助指导局部搜索的方向.高维约束强大的表示能力保留了有关图结构的关键性信息,使局部搜索沿着更有希望的方向进行,提高了 RLS 算法的性能.

2 基于染色划分过滤的改进 RLS 算法

2.1 最大团问题分析

对图 $G(V,E)$,若有顶点子集合 $S\subseteq V$,使得生成子图 $G(S)$中没有一条边全是孤立的顶点,则 S 为图 G 的一个独立集.记$\mathcal{S}$为图 G 的所有独立集构成的集合,则$\mathcal{S}=\{S_1,S_2,\ldots,S_m\}$,其中,$m=|\mathcal{S}|$,每个 S_i 都是图 G 的独立集.最大团问

题可以表示如下:

$$\begin{aligned} &Z=\max\sum_{j=1}^{n}x_j\\ &\text{s.t. } AX\leqslant 1\\ &\quad x_j\in\{0,1\},\ j=1,2,...,n \end{aligned} \tag{1}$$

其中,$n=|V|$为图 G 的顶点数,$X=\{x_1,x_2,\ldots,x_n\}$为解向量.如果 $x_j=\begin{cases}1, & \text{如果顶点}v_j\in\text{团}K\\ 0, & \text{否则}\end{cases}$,$A$ 为$\mathbb{S}$与顶点集 V 之间的隶属关系矩阵,A 为 $m\times n$ 维,即 $A_{ij}=\begin{cases}1, & \text{如果}v_j\in S_i\\ 0, & \text{否则}\end{cases}$.

定义 1. 记$\mathbb{S}$为图 G 的所有独立集构成的集合,若有 $S_1,S_2,\ldots,S_r\in\mathbb{S}$,使得 $\bigcup_{i=1}^{r}S_i=V$,且$\forall i,j\in\{1,2,\ldots,r\}$,有 $S_i\cap S_j=\varnothing$,则称 $IC=\{S_i\}_{i=1}^{r}$ 为图 G 的一个染色划分.

2.2 基于染色划分的高维约束构建

定理 1. 对图 G 的染色划分 $IC=\{S_i\}_{i=1}^{r}$ 和团 K,记 $V_{OUT}(K,IC)=\bigcup_{S_i\cap K=\varnothing}S_i$. 若 K 不是图 G 的唯一最大团 C,则图 G 的最大团 C 中至少有一点 $v\in V_{OUT}$.

证明:对图 G 的任意一个独立集 $S\in\mathbb{S}$和团 K,S 中的顶点最多只能有一个出现在团 K 中,所以对任意的染色划分 IC,每种染色 S_i 也最多只能有一个顶点出现在团 K 中,即$|K\cap S_i|\leqslant 1$.记 $V_{IN}(K,IC)=\bigcup_{S_i\cap K\neq\varnothing}S_i$, 则 V_{IN} 可划分为 $|K|$个独立集 S_i,则 C 中最多有$|K|$个点$\in V_{IN}$,即$|V_{IN}\cap C|\leqslant|K|$.因此,$C$ 中至少有$|C|-|K|$个点$\in V\backslash V_{IN}=V_{OUT}$.因为 $K\neq C$ 且 C 唯一,所以$|C|-|K|\geqslant 1$,即 C 中至少有一个点 $v\in V_{OUT}$. □

根据上面的定理,可以用染色划分 $IC=\{S_i\}_{i=1}^{r}$ 构建 $V_{OUT}(K,IC)$,作为选择搜索方向的依据.由于 IC 为$\mathbb{S}$的一个子集,所以,这就相当于从高维约束式 $AX\leqslant 1$ 中挑选出一部分来构成高维约束子集,引导局部搜索.

对当前团 K,不同的染色划分 IC 对应不同的 $V_{OUT}(K,IC)$.通过 IC 的不同选择,可以将搜索引导到不同的方向.为了使 V_{OUT} 能够为当前团 K 向最优团 C 移动提供更强的指导信息,V_{OUT} 中所含顶点元素的个数应该尽可能地少.对任意的当前团 K,其约束构建问题的数学表示为

$$\begin{aligned} &\min|V_{OUT}(K,IC)|\\ &\text{s.t. } IC=\{S_i\}_{i=1}^{r}\text{构成图}G\text{的一个染色划分} \end{aligned} \tag{2}$$

众所周知,染色问题是NP-hard的问题[13].问题(2)的难度不亚于染色问题本身,求解最优解相当困难.因为对于当前团 K,任何染色划分 IC 提供的 V_{OUT} 都具有一定的指导意义,且局部搜索具有随机性,所以无需花太大代价去求问题(2)的精确解,只要采用启发式算法求得问题(2)的一个满意解即可.

可以采用贪婪策略[13]对问题(2)进行求解,算法的计算复杂度为 $O(|V|^2)$.如果对每个不同的 K 都重新进行一次染色划分,所增加的计算开销是难以接受的.所以,算法在求解效果和计算代价之间选一折衷方案,每隔一段时间后才进行一次染色划分,令 *color_interval* 代表此染色间隔.

2.3 过滤局部搜索

RLS 算法的局部搜索中,定义了 3 种邻域移动操作[5]:

- 对当前团 K,定义“可加点集合”PA 为:在不破坏 K 的团性质的前提下,可直接加入当前团 K 中的所有顶点的集合.定义“添加”操作 $Add(K,v)=K\cup\{v\}$,其中,$v\in PA$;
- 定义“可替换点集合”OM 为:与当前团 K 中除一点外的所有顶点相连的顶点的集合.记这个 K 中唯一与 v 不相连的顶点为 u,则团中的 u 可以用 v 替换,并且 K 的团性质不变.定义“替换”操作 $Swap(K,v,u)=K\cup\{v\}\backslash\{u\}$,其中,$u\in K,v\in OM$.

- 定义“跃迁”操作 $Leap(K,v)$为:将任一点 $v\in V\backslash K$ 加入当前团 K,同时从 K 中移去所有与 v 不相连的顶点.

在 RLS 算法中,这 3 种邻域移动操作相互配合,有组织地完成局部搜索,相关细节请参阅文献[5].

本文提出的过滤局部搜索就是在“替换”$Swap(K,v,u)$和“跃迁”$Leap(K,v)$操作中,利用染色划分约束的指导,优先考虑 V_{OUT} 中的点.图 1 给出了点过滤的示意图,其中,每个黑点代表图 G 的一个顶点,每条直线代表图 G 的一条边.如图 1 所示,5 个独立集 S_1,S_2,S_3,S_4,S_5 构成了顶点集合 V 的染色划分.K 为当前团中所包含的顶点,PA 为团 K 的可加点集合,OM 为团 K 的可替换点集合.由于 S_2 和 S_3 不包含当前团 K 中的点,$V_{OUT}=S_2\cup S_3$.

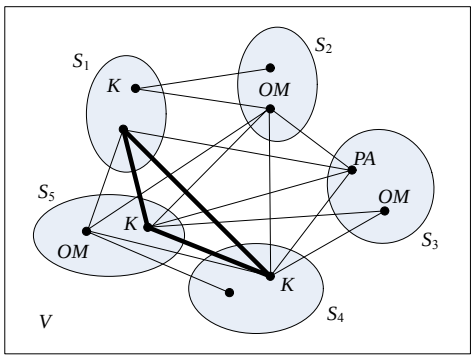


Fig.1 Schematic graph of the vertex refining

图 1 点过滤示意图

在 RLS 的局部搜索中,随机地从 OM 中选择一点进行替换操作,从 $V\backslash K$ 中选择一点进行跃迁操作.而在本文提出的过滤局部搜索中,利用染色约束 IC 对邻域移动方向进行过滤优选,随机地从 $OM\cap V_{OUT}$ 中选择一点进行替换操作,从 V_{OUT} 中选择一点进行跃迁操作.染色约束 IC 代表着一种对解空间构成结构的猜测,也更明确地限制了局部搜索的方向.过滤局部搜索的操作步骤如下,其中,$Tabu$ 是禁忌策略下暂时禁止选择的顶点集合[7]:

IF $PA\backslash Tabu\neq\varnothing$,则随机选择一个顶点 $v\in PA\backslash Tabu$,$Add(K,v_i)$,$Tabu=Tabu\cup\{v\}$;

ELSE IF $OM\backslash Tabu\neq\varnothing$,则随机选择一个顶点 $v\in(OM\backslash Tabu)\cap V_{OUT}$,$Swap(K,v,u)$,$Tabu=Tabu\cup\{v\}$;

END

过滤局部搜索策略将问题特定知识融合到 RLS 算法的动态局部搜索技术中,利用最大团问题结构上的特征,从邻域空间中优选出更有希望的移动方向,从而加快全局优化搜索的进程.

2.4 基于染色划分过滤的RLS-II算法流程

采用过滤局部搜索对 RLS 算法进行改进,得到基于染色划分过滤的 RLS-II 算法,其流程如下:

Step 1:设定染色间隔 $color\_interval$ 和局部搜索深度 $random\_interval$ 的值[7];

Step 2:染色计数变量 $color\_num=0$;

Step 3:如果 $\mathrm{mod}(color\_num,color\_interval)==0$,则对图 $G(V,E)$进行染色划分;

Step 4:随机产生一个团 K;

Step 5:局部搜索深度计数变量 $random\_num=0$;

Step 6:对团 K 进行过滤局部搜索;

Step 7:检验停止准则,若满足,则算法停止;

Step 8:$random\_num=random\_num+1$;

Step 9:如果 $\mathrm{mod}(random\_num,random\_interval)\neq0$,则进行跃迁操作 $Leap(K,v)$,并返回 Step 6;

如果 $\mathrm{mod}(random\_num,random\_interval)==0$,则 $color\_num=color\_num+1$,并返回 Step 3;

终止准则:当找到目标最优解 C,或者基本搜索次数超过了预设的最大允许值 Max_Steps 时算法结束.

3 RLS-II 算法的改进分析

3.1 RLS-II算法性能改善分析

RLS 算法虽然简单,但其性能却远远好于其他一些复杂的算法.究其原因,主要与最大团问题的结构有关:首先,最大团问题解空间的规模是图的顶点数目的指数级.全局优化算法虽然能够收集大量的函数点信息,但相对于组合爆炸的解空间大小来说依然是微不足道的.即使是付出高昂的时间代价,其性能也不一定比局部搜索算法好多少;其二,最大团问题的约束条件都是二元结构的边约束,无高维约束意味着邻域的强连通性,局部搜索可以以很大的概率从当前位置转移到其他任何解空间区域,较少遇到搜索陷入局部最优解而停滞的问题;第三,最大团问题作为最困难的组合优化问题之一,解空间包含大量平坦区域,而数目稀少的最优解常无规律地矗立于某个平坦区域中,即使由最优解周围的函数值信息,也难以推测出最优解是否存在的判断.这种大海捞针式的解空间结构使得建立在“积木块假设”和“相似最优原则”基础上的演化算法[14]难以发挥优势.爬山式的随机局部搜索算法反而是最简单、最可行的方法.

随机局部搜索算法求解最大团问题虽然效果出众,但进行数学分析比较困难.随机局部搜索算法作为一种不要求保证与最优解的偏离程度的启发式算法,到目前为止还没有找到合适的数学分析工具.而最大团问题本身的结构也很复杂,具体邻域结构的表示和分析都比较困难.局部搜索算法中经常使用的禁忌搜索策略使得算法不仅与当前状态有关,还与过去的决策选择有关,更增加了分析的难度.本文为分析方便,对问题做出以下简化和假设:

a) 不考虑禁忌策略的影响;

b) 只考虑替换操作,并假设从任一当前解 K 出发,都可以通过有限次的替换操作到达最优解 C.由于在 RLS 和 RLS-II 算法中大量进行的是替换操作[5],因此可以认为这种假设是合理的;

c) 假设图的结构是均匀的;

d) 假设最优解存在且唯一.

本文不求对所提算法的性能做出严格的数学分析,只尝试给出一个启发式的解释,说明染色划分过滤在改善局部搜索性能方面的作用.

3.1.1 向最优解靠近的概率

采用 RLS 算法的替换操作 $Swap(K,v,u)$,从 OM 中任选一点,该点属于 C 的概率为 $P_H(\text{RLS})=|C\cap OM|/|OM|$.$P_H(\text{RLS})$为替换操作下局部搜索向最优解靠近的概率.记 D 为图的密度[13],即图中任意两顶点间有边相连的概率,则$|OM|\approx(n-|K|)\cdot D^{|K|-1}\cdot(1-D)$.

又因为 $C\cap K\notin OM$,而 $C\backslash K$ 中的点入选 OM 的概率为 $D^{|K\backslash C|-1}\cdot(1-D)$,所以$|C\cap OM|\approx|C\backslash K|\cdot D^{|K\backslash C|-1}\cdot(1-D)$,

$$P_H(\text{RLS})\approx\frac{|C\setminus K|}{n-|K|}\cdot D^{-|K\cap C|} \tag{3}$$

记图 G 的染色划分为 $IC=\{S_1,S_2,\ldots,S_r\}$,其中,$l_i=|S_i|$为各组长度,$r$ 为总的染色分组数目,则平均染色长度为

$$\overline{l}=\frac{1}{r}\sum_{i=1}^{r}l_i=\frac{n}{|IC|}.$$

令 $I(v)$代表 v 在 IC 中所在的染色分组,即 $I(v)=i$,如果 $v\in S_i$,则 $I(C)=\{i|S_i\cap C\neq\varnothing\}$,$I(K)=\{i|S_i\cap K\neq\varnothing\}$.

而采用 RLS-II 算法的替换操作 $Swap(K,v,u)$,从 $V_{OUT}\cap OM$ 中任选一点,该点属于 C 的概率为

$$P_H(\text{RLS-II})=\frac{|C\cap V_{OUT}\cap OM|}{|V_{OUT}\cap OM|}.$$

因为$|V_{OUT}|\approx n-|K|\cdot\overline{l}=n-|K|\cdot\frac{n}{|IC|}$,所以,

$$|V_{OUT}\cap OM|\approx|V_{OUT}|\cdot D^{|K|-1}\cdot(1-D)\approx\left(n-|K|\cdot\frac{n}{|IC|}\right)\cdot D^{|K|-1}\cdot(1-D).$$

因为 $C\cap V_{OUT}$ 中的点入选 OM 的概率为 $D^{|K\backslash C|-1}\cdot(1-D)$,有

$$|C\cap V_{OUT}\cap OM|\approx|C\cap V_{OUT}|\cdot D^{|K\setminus C|-1}\cdot(1-D).$$

又因为$|C\cap V_{OUT}|=|C|-|I(C)\cap I(K)|=|C|-(|C\cap K|+|I(C\setminus K)\cap I(K)|)$.为了求出$|I(C\setminus K)\cap I(K)|$,可以将问题转换为下面的求组合数的问题:图 2 中,○代表 K 中元素,×代表 C 中元素,每个矩形框代表一个染色分组 S_i,每个染色分组具有不同的长度 l_i,则$|I(C\setminus K)\cap I(K)|$就相当于将$|C\setminus K|$个球无重复地放入$|IC|-|K\cap C|$个盒中,则其中$|K\setminus C|$个盒子能够得到的球的个数.由此有 $|I(C\setminus K)\cap I(K)|\approx\frac{|K\setminus C|}{|IC|-|K\cap C|}\cdot|C\setminus K|$ 及

$$P_H(\text{RLS-II})\approx\frac{|C\setminus K|\cdot|IC|}{n\cdot(|IC|-|K\cap C|)}\cdot D^{-|K\cap C|} \tag{4}$$

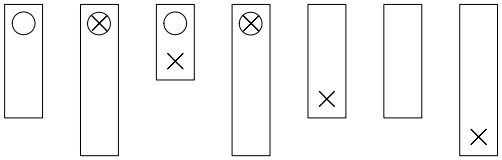


Fig.2 Schematic graph of color partition

图 2 染色划分示意图

3.1.2 P_H(RLS-II)≥P_H(RLS)的条件

$$\begin{aligned}P_H(\text{RLS-II})-P_H(\text{RLS})&\approx\frac{|C\setminus K|\cdot|IC|}{n\cdot(|IC|-|K\cap C|)}\cdot D^{-|K\cap C|}-\frac{|C\setminus K|}{n-|K|}\cdot D^{-|K\cap C|}\\&=\frac{|IC|\cdot(n-|K|)-n\cdot(|IC|-|K\cap C|)}{n\cdot(|IC|-|K\cap C|)\cdot(n-|K|)}\cdot|C\setminus K|\cdot D^{-|K\cap C|}\\&=\frac{|C\setminus K|\cdot D^{-|K\cap C|}}{(|IC|-|K\cap C|)\cdot(n-|K|)}\cdot\left(|K\cap C|-\frac{|K|}{\overline{l}}\right)\end{aligned} \tag{5}$$

由于$|C\setminus K|)\cdot D^{-|K\cap C|}\geq 0$,$|IC|\geq|K\cap C|$,$n\geq|K|$,$|C|\geq|K|$,所以,当满足 $|K\cap C|\geq\frac{|C|}{\overline{l}}$ 时,有 P_H(RLS-II)≥P_H(RLS).

又因为$|K\cap C|=|C|-d$,所以当满足下述条件(6)时,有 P_H(RLS-II)≥P_H(RLS):

$$d/|C|\leqslant 1-1/\overline{l} \tag{6}$$

由于搜索空间为 $d\in[0,|C|]$,即 $d/|C|\in[0,1]$,所以在 $d/|C|\in[0,1-1/\overline{l}]$ 的区间里,P_H(RLS-II)≥P_H(RLS);只在 $d/|C|\in(1-1/\overline{l},1]$ 的区间里,有 P_H(RLS-II)<P_H(RLS).在 Frb 和 DIMACS 共 120 个最大团问题标准测试图例中[15],$\overline{l}\in[1.78,58.82]$,$\overline{l}$ 的均值为 12.46,方差为 9.41;$1-1/\overline{l}\in[0.44,0.98]$,$1-1/\overline{l}$ 的平均值为 0.85,方差为 0.13.所以可以认为,在大部分情况下,染色划分过滤提高了局部搜索趋向最优解的概率;且在相同的 n,$|C|$,$|K|$,$|K\cap C|$下,当 $\overline{l}$ 越大时,P_H(RLS-II)−P_H(RLS)越大,提高越明显.

3.2 RLS与RLS-II算法吸收时间的比较分析

3.2.1 吸收态 Markov 链模型

沿用第 3.1 节的简化和假设,建立局部搜索算法求解最大团问题的吸收态 Markov 链模型.记$\zeta=\{K|K$ 是图 G 的团$\}$为解空间.定义 $d(K)=|C\setminus K|$为当前解 K 与最优解 C 之间的距离,也即从 K 移动到 C 最少所需进行的替换操作步数.那么,$D_i=\{K\in\zeta|d(K)=i\}$,$i=0,1,\ldots,|C|$为解空间ζ的一个划分,记为 $D=\{D_0,D_1,\ldots,D_{|C|}\}$.记$\{X_t;t=0,1,2,\ldots\}$为替换操作下当前团 K 在状态空间 D 上的随机游动序列,则满足第 3.1 节假设的 X_t 是一个齐次吸收态 Markov 链[16],其吸收态为 $D_0=C$,一步状态转移概率有以下几种情况:

a) 当 $X_t\in D_i$,$i=0$ 时,$P(X_{t+1}\in D_j\mid X_t\in D_i)=\begin{cases}1, & j=0\\0, & j\neq 0\end{cases}$;

b) 当 $X_t\in D_i,i=|C|$时, $P(X_{t+1}\in D_j \mid X_t\in D_i)=\begin{cases}P(v\notin C\mid K\in D_i), & j=i\\ P(v\in C\mid K\in D_i), & j=i-1\\ 0, & j=i+1\end{cases}$;

c) 当 $X_t\in D_i,1\leqslant i<|C|$时, $P(X_{t+1}\in D_j \mid X_t\in D_i)=\begin{cases}P(v\notin C\mid K\in D_i)\cdot P(u\notin C\mid v\notin C,K\in D_i), & j=i\\ P(v\in C\mid K\in D_i), & j=i-1\\ P(v\notin C\mid K\in D_i)\cdot P(u\in C\mid v\notin C,K\in D_i), & j=i+1\end{cases}$.

记 $a_i=P(X_{t+1}\in D_{i-1}|X_t\in D_i)$,$b_i=P(X_{t+1}\in D_{i+1}|X_t\in D_i)$,则$\{X_t;t=0,1,\ldots\}$在状态空间 D 上的转移矩阵为

$$\begin{pmatrix}1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0\\ a_1 & 1-a_1-b_1 & b_1 & 0 & \cdots & 0 & 0 & 0\\ 0 & a_2 & 1-a_2-b_2 & b_2 & \cdots & 0 & 0 & 0\\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots\\ 0 & 0 & 0 & 0 & \cdots & a_{|C|-1} & 1-a_{|C|-1}-b_{|C|-1} & b_{|C|-1}\\ 0 & 0 & 0 & 0 & \cdots & 0 & a_{|C|} & 1-a_{|C|}\end{pmatrix}.$$

根据文献[17]中定理 1 的推论 2,从任一初始点 $X_0\in D_i$ 出发,首次到达吸收态 D_0 的平均期望时间(即找到最优解 C 所需的基本替换操作步数)为

$$\begin{cases}m_0=0\\ m_i=m_{i-1}+\dfrac{1}{a_i}+\sum\limits_{j=0}^{|C|-(i+1)}\dfrac{1}{a_{j+i+1}}\prod\limits_{h=0}^{j}\dfrac{b_{i+h}}{a_{i+h}},\ i=1,\ldots,|C|-1\\ m_{|C|}=m_{|C|-1}+1/a_{|C|}\end{cases}\tag{7}$$

记 $m=[m_i]_{i\in D}$,则$\|m\|_\infty=\max\{m_i\}$可以被视为算法在最差情况下的计算复杂性[17].

3.2.2 两种算法的吸收时间分析

最大团问题属于 long path 类的问题[16],在结构上具有多峰密布且各峰值都很相近的特征.根据鞅理论,一幅图的 clique 数高度集中于期望值附近[18],为简化计算,后面都假设$|K|\approx|C|$.在替换操作 $Swap(K,v,u)$中,对任意的 $v\notin C$,每一点 $u\in K$ 与相连的概率都是相同的.因此,对于 RLS 和 RLS-II 算法,都有

$$P(u\in C\mid v\notin C,K\in D_i)\approx\frac{|C\cap K|}{|K|}\approx\frac{|C\cap K|}{|C|}\tag{8}$$

因为$|C|=|C\cap K|+|C\setminus K|$,所以$|C\cap K|=|C|-i$.对 RLS 算法的替换操作,有

$$P_{\mathrm{RLS}}(v\in C\mid K\in D_i)=P_H(\mathrm{RLS})\approx\frac{|C\setminus K|}{n-|K|}\cdot D^{-|K\cap C|}\approx\frac{|C\setminus K|}{n-|C|}\cdot D^{-|K\cap C|}=\frac{i\cdot D^{i-C}}{n-|C|}\tag{9}$$

对 RLS-II 算法的替换操作,有

$$P_{\text{RLS-II}}(v\in C\mid K\in D_i)=P_H(\text{RLS-II})\approx\frac{|C\setminus K|\cdot|IC|}{n\cdot(|IC|-|K\cap C|)}\cdot D^{-|K\cap C|}=\frac{|IC|}{n}\cdot\frac{i\cdot D^{i-C}}{|IC|-|C|+i}\tag{10}$$

又因为

$$P(v\notin C|K\in D_i)=1-P(v\in C|K\in D_i),P(u\notin C|v\notin C,K\in D_i)=1-P(u\in C|v\notin C,K\in D_i)\tag{11}$$

将公式(8)~公式(11)代入公式(7)中,就可以求出从任一初始状态 $K\in D_i$ 出发,首次游动到最优解 C 所需的替换操作步数的期望值 m_i.记 τ_{RLS} 为 RLS 算法的 $\max\{m_i\}$值,$\tau_{\text{RLS-II}}$ 为 RLS-II 算法的 $\max\{m_i\}$值.定义算法的理论改善率 I_1 为 $I_1=(\tau_{\mathrm{RLS}}-\tau_{\text{RLS-II}})/\tau_{\mathrm{RLS}}$.在 77 个国际公认的最有代表性的最大团问题测试算例(下一节给出详细叙述)上进行计算,得到如图 3 所示的理论改善率 I_1 与平均染色组长度 $\overline{l}$ 之间的关系图.0 轴上方的每个点对应 RLS-II 算法好于 RLS 算法的一个图例.由图 3 中可以看出,在 77 个例子中,有 57 个例子,RLS-II 算法相较于 RLS 算法都有改进,改进率达到 74%.且平均染色组长度越大,改进的可能性和幅度也越大.

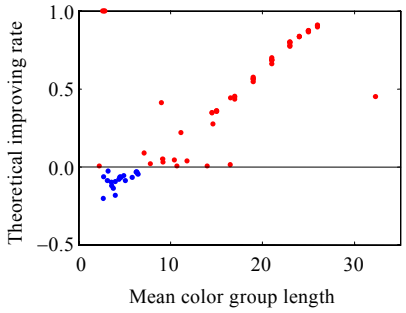


Fig.3 Relations between mean color group length and theoretical improving rate

图 3 染色组平均长度与理论改善率的关系

4 实验分析

本文全部实验在 cygwin 平台上运行,采用 g++,-O2 编译,机器主频为 2.66GHz,内存为 1.98GB.参数选择为 Max_Steps=400,000,000,*color_interval*=200,*random_interval*=2000.

4.1 Frb测试数据上的实验结果

最大团问题的 Benchmark 图是从以下网址下载到的:http://www.nlsde.uaa.edu.cn/~kexu/benchmar-ks/graph-benchmarks.htm.这里共有 40 个例子,是目前国际公认最难的一组最大团测试问题[19].

表 1 中给出了 RLS-II 算法与 pls 和 COVER 算法在不同 Frb 例子上的性能比较.pls 算法为目前性能最好的 RLS 改进算法[10],COVER 为目前性能最好的求解该组问题的算法[6].

Table 1 Performance comparison of different algorithms on Frb Benchmarks

表 1 标准测试数据集 Frb 上各算法性能比较

Instance	$\omega(G)$	pls			RLS-II			COVER		
		Success	Average clique size	CPU (s)	Success	Average clique size	CPU (s)	Success	Average clique size	CPU (s)
Frb30-15-1	30	100	30	0.06	100	30	0.01*	100	30	0.17
Frb30-15-2	30	100	30	0.07	100	30	0.02*	100	30	0.22
Frb30-15-3	30	100	30	0.61	100	30	0.20*	100	30	1.16
Frb30-15-4	30	100	30	0.04	100	30	0.01*	100	30	0.15
Frb30-15-5	30	100	30	0.29	100	30	0.07*	100	30	0.38
Frb35-17-1	35	100	35	3.61	100	35	0.77*	100	35	2.33
Frb35-17-2	35	100	35	1.07	100	35	0.32*	100	35	2.68
Frb35-17-3	35	100	35	0.23	100	35	0.05*	100	35	0.59
Frb35-17-4	35	100	35	4.93	100	35	1.20*	100	35	2.87
Frb35-17-5	35	100	35	0.68	100	35	0.15*	100	35	1.17
Frb40-19-1	40	100	40	2.89	100	40	0.25*	100	40	1.38
Frb40-19-2	40	100	40	46.83	100	40	6.87*	100	40	38.43
Frb40-19-3	40	100	40	4.19	100	40	1.07*	100	40	8.74
Frb40-19-4	40	100	40	19.57	100	40	4.70*	100	40	22.61
Frb40-19-5	40	100	40	85.19	100	40	24.07*	96	39.96	200.51
Frb45-21-1	45	100	45	32.52	100	45	5.47*	100	45	23.914
Frb45-21-2	45	100	45	66.64	100	45	15.98*	100	45	69.26
Frb45-21-3	45	100	45	333.77	100	45	62.18*	99	44.99	202.5
Frb45-21-4	45	100	45	47.79	100	45	10.26*	100	45	38.23
Frb45-21-5	45	100	45	88.14	100	45	22.12*	99	44.99	215.87
Frb50-23-1	50	72	49.72	753.06	100	50*	196.68	89	49.89	338.45
Frb50-23-2	50	45	49.45	827.58	97	49.97*	547.55	30	49.30	431.77
Frb50-23-3	50	16	49.16	720.78	66	49.66*	935.53	24	49.24	608.17
Frb50-23-4	50	100	50	88.31	100	50	12.57*	100	50	62.59
Frb50-23-5	50	99	49.99	309.48	100	50*	38.15	98	49.98	294.3
Frb53-24-1	53	6	52.06	1 130.83	32	52.32*	1 162.83	9	52.09	666.15

Table 1 Performance comparison of different algorithms on Frb benchmarks (continue)

表 1 标准测试数据集 Frb 上各算法性能比较(续)

Instance	$\omega(G)$	pls			RLS-II			COVER		
		Success	Average clique size	CPU (s)	Success	Average clique size	CPU (s)	Success	Average clique size	CPU (s)
Frb53-24-2	53	23	52.23	1 010.53	84	52.84*	840.82	34	52.34	414.23
Frb53-24-3	53	66	52.66	777.09	100	53*	190.56	91	52.91	323.91
Frb53-24-4	53	46	52.46	929.40	98	52.98*	593.40	24	52.24	387.66
Frb53-24-5	53	85	52.85	638.28	100	53*	218.01	84	52.84	428.61
Frb56-25-1	56	12	55.1	772.92	57	55.57*	1 112.15	15	55.15	550.95
Frb56-25-2	56	6	54.93	1 058.10	40	55.40*	1 087.49	12	55.12	765.23
Frb56-25-3	56	8	55.08	906.32	80	55.80*	1 091.74	76	55.76	506.08
Frb56-25-4	56	68	55.66,	805.41	100	56*	295.18	84	55.84	460.82
Frb56-25-5	56	81	55.81	717.35	100	56*	120.47	98	55.98	237.8
Frb59-26-1	59	0	57.85	–	6	58.06	668.04	11	58.11*	827.59
Frb59-26-2	59	0	57.63	–	9	58.09*	809.44	6	58.06	453.89
Frb59-26-3	59	6	57.77	1 169.60	71	58.71*	978.29	12	58.12	831.79
Frb59-26-4	59	5	57.71	1 241.64	47	58.47*	1 269.51	1	58.01	1928.5
Frb59-26-5	59	78	58.77	723.51	100	59*	112.65	89	58.89	609.18

仿照文献[20]的性能评估方法,“成功次数”是算法运行 100 次成功找到最优解的次数;“平均计算时间”是所有成功运行时计算时间的平均值,单位为秒;“团均值”是算法在 100 次运行中所找到的最大团的平均大小.pls 算法和 COVER 算法的“时间”都是在作者机器上运行的结果,源程序分别为作者 Pullan 和 Richter 所提供.

表 1 按照“团均值→平均计算时间”的顺序对 3 种算法进行了比较,*代表性能最好.由表 1 可见:在 40 个 Frb 例子中,有 39 个例子里 RLS 算法的性能都超过了目前最好的 COVER 算法;在全部 40 个例子上,RLS-II 算法的性能都优于 pls 算法.

COVER 算法是将最大团问题转换为顶点覆盖问题后再求解,从求解思路上看,不属于 RLS 算法系列;而 pls 算法是目前性能最好的 RLS 算法.这证明了在 Frb 例子上,所提出的基于染色划分构建高维约束指导局部搜索的策略能够有效地提高原 RLS 算法的性能.而增加了染色划分过滤机制的 RLS-II 算法,在 Frb 系列例子上已达到了目前最大团问题求解算法的最好水平.

4.2 DIMACS测试数据上的实验结果

DIMACS 系列最大团测试数据集是迄今为止使用最多的最大团问题 Benchmark 数据集[6,14].

表 2 中给出了 RLS-II 算法与原始 RLS 算法[5]、DLS 算法[9]和 IEA-PTS 算法[6]在 37 个典型 DIMACS 例子上的性能比较.DLS 是目前该组例子上性能最好的算法[6],IEA-PTS 是最新的算法[6].

由表 2 可见,除了 brock800_2 和 C4000.5 之外,在所有例子上,算法的性能都超过或接近目前最好的 DLS 算法.在较简单的例子上,4 种算法都能轻易地找到目标最优解.但由于染色划分带来了额外的计算开销,算法的计算时间比 DLS 算法要长一些,但依然好于其他两种算法;且相对于简单例子 100s 以内的计算时间而言,这应该不能算是明显的缺点.而在大型困难例子上,例如 C2000.9,MANN_a81 和 keller6,RLS-II 算法的优势非常明显,总能找到更好的解,使结果获得质的提高,这充分证明了所提出的基于染色划分构建高维约束指导局部搜索的策略是有效的.

需要特别指出的是,RLS-II 算法的参数设定非常简单(在所有图例上均使用同一组参数值).而 DLS 算法虽然计算时间相对较小,但算法性能受惩罚延迟参数的影响很大[9],而该参数的设定又没有统一的规律,全靠实验选择,因此确定过程相当费时.

综合来看,在 Frb Benchmark 测试数据库上,染色划分过滤机制的改进效果明显地更加突出,这与 Frb 系列图例的结构有关.Frb 系列图例的平均染色组长度都较大,最小的是 14.52,平均为 21.16;而在 DIMACS Benchmark 测试数据库上,平均染色组长度都较小,最小的是 2.27,平均为 7.19.这再次验证了染色划分过滤机制的改进效果与平均染色组长度的大小有关.

Table 2 Performance comparison of different algorithms on DIMACS Benchmarks

表 2 标准测试数据集 DIMACS 上各算法性能比较

Instance	RLS		RLS-II		DLS		IEA-PTS	
	CPU (s)	CLQavg (CLQmax)	CPU (s)	CLQavg (CLQmax)	CPU (s)	CLQavg (CLQmax)	CPU (s)	CLQavg (CLQmax)
C125.9	0.004	34	0.001	34	0.000 1*	34	0.000 4	34
C250.9	0.029	44	0.01	44	0.000 9*	44	0.015 6	44
C500.9	3.124	57	0.08*	57	0.127 2	57	0.987 2	56.18 (57)
C1000.9	41.660	68	0.98*	68	4.44	68	4.475 8	66.93 (68)
C2000.9	823.358	77.57 (78)	178.13	78.11 (80)*	(193.224)	77.93 (78)	52.252 8	76.4 (79)
DSJC500.5	0.194	13	0.05	13	0.013 8*	13	0.086 2	13
DSJC1000.5	6.453	15	0.96	15	0.799*	15	3.548 2	15
C2000.5	9.976	16	4.58	16	0.969 7*	16	2.936 6	16
C4000.5	2 183.089	18	477.12	18	181.233 9*	18	276.184	17.66 (18)
MANN_a27	3.116	126	0.02*	126	0.047 6	126	0.566 3	126
MANN_a45	398.770	343.60 (345)	91.54	344.16 (345)*	(51.960 2)	344 (344)	25.005 4	343.97 (345)
MANN_a81	2 830.820	1 098 (1 098)	314.49	1 098.5 (1 100)*	(264.009 4)	1 097.96 (109 8)	629.529	1 097.01 (109 9)
brock200_2	9.605	12	0.04	12	0.024 2*	12	0.098 2	12
brock200_4	19.491	17	0.08	17	0.046 8*	17	0.184 7	17
brock400_2	42.091	26.06 (29)	0.86	29	0.477 4*	29	2.870 2	27.52 (29)
brock400_4	108.638	32.42 (33)	0.08	33	0.067 3*	33	2.227 9	33
brock800_2	4.739	21	78.65	23.97 (24)	15.733 5	24*	2.733 2	21.06 (24)
brock800_4	6.696	21	31.13	26	8.880 7*	26	5.505 9	21.4 (26)
gen200_p0.9_44	0.037	44	0.001*	44	0.001	44	0.005 2	44
gen200_p0.9_55	0.016	55	0.001	55	0.000 3*	55	0.001 4	55
gen400_p0.9_55	1.204	55	0.05	55	0.026 8*	55	5.277 3	54.8 (55)
gen400_p0.9_65	0.050	65	0.02	65	0.001*	65	0.016 7	65
gen400_p0.9_75	0.051	75	0.02	75	0.000 5*	75	0.008 2	75
Hamming8-4	0.003	16	0.001	16	<*Q**	16	0.000 4	16
hamming10-4	0.078	40	0.19	40	0.008 9*	40	0.058	40
keller4	0.002	11	0.001	11	<*Q**	11	0.000 2	11
keller5	0.171	27	0.09	27	0.020 1	27	0.019 8*	27
keller6	189.814	59	82.35*	59	170.482 9	59	120.325	57.06 (59)
p_hat300-1	0.018	8	0.001	8	0.000 7*	8	0.001 9	8
p_hat300-2	0.006	25	0.001	25	0.000 2*	25	0.001 1	25
p_hat300-3	0.021	36	0.01	36	0.000 7*	36	0.005 7	36
p_Hat700-1	0.186	11	0.05	11	0.019 4*	11	0.045	11
p_hat700-2	0.028	44	0.07	44	0.001*	44	0.000 91	44
p_hat700-3	0.035	62	0.13	62	0.001 5*	62	0.026 7	62
p_hat1500-1	30.274	12	3.05	12	2.706 4*	12	9.647 6	12
p_hat1500-2	0.158	65	0.77	65	0.006 1*	65	0.133 4	65
p_hat1500-3	0.192	94	1.54	94	0.010 3*	94	0.430 8	94

4.3 性能改善的实验分析

图 4 给出了在 frb35-17-1 图例上,不同的局部搜索深度 *random_interval* 之下,RLS 算法和 RLS-II 算法找到最大团所需的基本搜索次数的比较.图 4 中的每个点都是算法运行 100 次得到的平均结果.可见,RLS-II 算法的性能明显优于 RLS 算法.这说明,基于染色划分引导局部搜索的策略确实能够有效提高 RLS 算法在最大团问题上的全局优化性能.

记去除染色划分过滤后的 RLS-II 算法为 RLS0 算法.定义实际改善率 I_2 为 RLS-II 算法相对 RLS0 算法找到相同最优解时基本搜索步数的减少百分比,即

$$I_2=[step(\text{RLS0})-step(\text{RLS-II})]/step(\text{RLS0}).$$

图 5 给出了染色组平均长度 $\overline{I}$ 与实际改善率 I_2 之间的关系图(RLS0 和 RLS-II 算法在 37 个 DIMACS 和 40 个 Frb 例子上的运行结果,*step* 为 100 次运行中成功找到最大团时的基本操作步数的平均值).对照图 3 可以看出:在染色组平均长度较大时,实验结果与理论分析结果虽然存在差异,但大致趋势上具有一致性;而在染色组平均长度较小时,实验结果与理论分析结果存在一定的出入.这一方面是因为理论分析建立在 4 个假设和简化的 Markov 链模型基础上;另一方面,当 $\overline{I}$ 较大时,$\frac{|C|}{|IC|}=\frac{|C|}{n}\cdot\overline{I}$ 较大,即$|C|$更接近于$|IC|$,导向信息较强,染色划分过滤机制对算法性能的影响也就较强,理论分析与实验结果有较强的可比性;而当 $\overline{I}$ 较小时,导向信息较弱,染

色划分过滤机制对算法性能的影响不明显,算法性能更多地受随机性的影响,可比性相对较小.

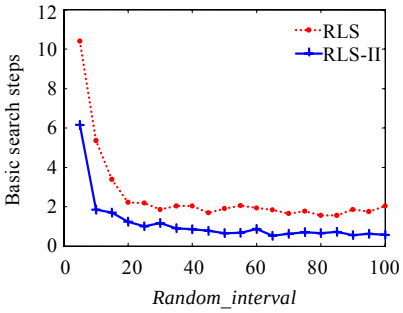


Fig.4 Influence of *random_interval* parameter on algorithm performance

图 4 局部搜索深度参数对算法性能的影响

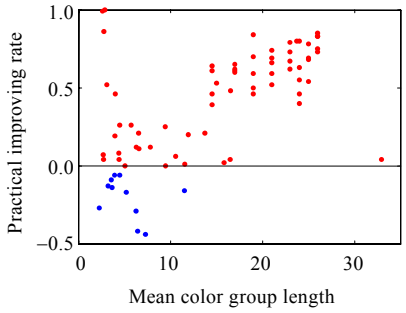


Fig.5 Relations between mean color group length and practical improving rate

图 5 染色组平均长度与实际改善率的关系

5 结 论

本文利用最大团问题解空间特殊的结构特征,提出了一种新的基于染色划分构建高维约束指导局部搜索的 RLS-II 算法.该算法融合了 RLS 局部搜索算法的优点和最大团问题的特定结构信息,使单步移动沿着更有希望的邻域方向进行,减少了无效搜索,提高了算法的性能.基于吸收态 Markov 链的理论分析以及在 DIMACS 和 Frb 标准测试数据集上的仿真结果均表明,所提出的基于染色划分构建高维约束改进局部搜索性能的算法是有效的.

References:

[1] Xie ZQ. Biviariate entropy function method and homotopy method for maximum clique problem [MS. Thesis]. Dalian: Dalian University of Technology, 2006 (in Chinese with English abstract).

[2] Li KL, Zhou X, Zou ST. Improved volume molecular solutions for the maximum clique problem on DNA-based supercomputing. Chinese Journal of Computers, 2008,31(12):2173−2181 (in Chinese with English abstract).

[3] Lu Q, Bai ZH, Xia XY. Leader-Based parallel cross entropy algorithm for maximum clique problem. Journal of Software, 2008, 19(11):2899−2907 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/2899.htm [doi: 10.3724/SP.J.1001.2008.02899]

[4] Bomze IM, Budinich M, Pardalos PM, Pelillo M. The Maximum Clique Problem. Berlin: Springer-Verlag, 1999. 1−74.

[5] Battiti R, Protasi M. Reactive local search for the maximum clique problem. Algorithmica, 2001,29(4):610−637. [doi: 10.1007/s004530010074]

[6] Guturu P, Dantu R. An impatient evolutionary algorithm with probabilistic Tabu search for unified solution of some NP-hard problems in graph and set theory via clique finding. IEEE Trans. on Systems Man and Cybernetics, Part B—Cybernetics, 2008, 38(3):645−666. [doi: 10.1109/TSMCB.2008.915645]

[7] Hansen P, Mladenović N, Urošević D. Variable neighborhood search for the maximum clique. Discrete Applied Mathematics, 2004, 145(1):117−125. [doi: 10.1016/j.dam.2003.09.012]

[8] Katayama K, Hamamoto A, Narihisa H. An effective local search for the maximum clique problem. Information Processing Letters, 2005,95(5):503−511. [doi: 10.1016/j.ipl.2005.05.010]

[9] Pullan W, Hoos HH. Dynamic local search for the maximum clique problem. Journal of Artificial Intelligence Research, 2006,25(1): 159−185.

[10] Pullan W. Phased local search for the maximum clique problem. Journal of Combinatorial Optimization, 2006,12(3):303−323. [doi: 10.1007/s10878-006-9635-y]

[11] Battiti R, Mascia F. Reactive and dynamic local search for max-clique: Engineering effective building blocks. Computers and Operations Research, 2010,37(3):534−542. [doi: 10.1016/j.cor.2009.02.013]

[12] He J, Yao X, Li J. A comparative study of three evolutionary algorithms incorporating different amounts of domain knowledge for node covering problem. IEEE Trans. on Systems Man and Cybernetics, Part C—Applications and Reviews, 2005,35(2):266−271. [doi: 10.1109/TSMCC.2004.841903]

[13] Diestel R. Graph Theory. 2nd ed., Berlin: Springer-Verlag, 2005. 111−136.

[14] Zhang QF, Sun JY, Tsang E. An evolutionary algorithm with guided mutation for the maximum clique problem. IEEE Trans. on Evolutionary Computation, 2005,9(2):192−200. [doi: 10.1109/TEVC.2004.840835]

[15] Richter S, Helmert M, Gretton C. A stochastic local search approach to vertex cover. Lecture Notes in Computer Science, 2007, 4667:412−426. [doi: 10.1007/978-3-540-74565-5_31]

[16] He J, Yao X. Towards an analytic framework for analysing the computation time of evolutionary algorithms. Artificial Intelligence, 2003,145(1-2):59−97. [doi: 10.1016/S0004-3702(02)00381-8]

[17] Zhou YR, He J, Nie Q. A comparative runtime analysis of heuristic algorithms for satisfiability problems. Artificial Intelligence, 2009,173(2):240−257. [doi: 10.1016/j.artint.2008.11.002]

[18] Motwani R, Raghavan P. Randomized Algorithms. London: Cambridge University Press, 1995. 83−96.

[19] Xu K, Boussemart F, Hemery F, Lecoutre C. Random constraint satisfaction: Easy generation of hard (satisfiable) instances. Artificial Intelligence, 2007,171(8-9):514−534. [doi: 10.1016/j.artint.2007.04.001]

[20] Grosso A, Locatelli M, Pullan W. Simple ingredients leading to very efficient heuristics for the maximum clique problem. Journal of Heuristics, 2008,14(6):587−612. [doi: 10.1007/s10732-007-9055-x]

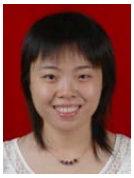
附中文参考文献:

[1] 谢震乔.最大团问题的二元熵函数法及同伦方法[硕士学位论文].大连:大连理工大学,2006.

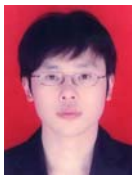
[2] 李肯立,周旭,邹舒婷.一种改进的最大团问题 DNA 计算机算法.计算机学报,2008,31(12):2173−2181.

[3] 吕强,柏战华,夏晓燕.一种求解最大团问题的并行交叉熵算法.软件学报,2008,19(11):2899−2907. http://www.jos.org.cn/1000-9825/19/2899.htm [doi: 10.3724/SP.J.1001.2008.02899]

张雁(1979－),女,河南许昌人,博士生,主要研究领域为演化计算,启发式算法,组合优化,图论.



卢昕玮(1979－),女,博士,讲师,主要研究领域为计算数学,运筹与最优化.



焦方正(1984－),男,博士生,主要研究领域为航天器故障诊断.

黄永宣(1939－),男,教授,博士生导师,主要研究领域为复杂系统建模与控制,航天器测控技术研究.