

带权最大团问题的两种高效局部搜索算法

—— [Two Efficient Local Search Algorithms for Maximum Weight Clique Problem](#) 中文翻译版

带权最大团问题的两种高效局部搜索算法

摘要

介绍

准备

CC(Configuration Checking)回顾

SCC (Strong Configuration Checking)

LSCC算法

根据标准基准对LSCC进行评估

改进大规模图的LSCC

BMS启发式算法和LSCC+BMS算法

在大规模图上的实验

SCC和BMS的效果

结论

致谢

引用

摘要

带权最大团问题 (MWCP) 是最大团问题的一个重要推广, 它有着广泛的应用。本文介绍了两种启发式算法, 并且提出了两种局部搜索算法以解决MWCP问题。首先, 我们提出了一种称为强配置检查(SCC)的启发式算法, 它是最近一种称为配置检查(CC)的用于减少局部搜索中的循环的强大策略的变体。基于SCC策略, 我们提出了一个名为LSCC的局部搜索算法。此外, 为了提高在大规模的图上的性能, 我们采用了一种低复杂度的启发式方法, 称为最佳多选(Best from Multiple Selection, BMS), 以快速有效地选择交换顶点对。BMS启发式算法用于改进LSCC, 从而得到LSCC+BMS算法。实验表明, 在标准基准DIMACS与BHOSLIB以及现实世界的各种大规模图上, 我们提出的这一算法性能超过了已经是最高水准的局部搜索算法MN/TS以及它的改进版本MN/TS+BMS。

介绍

给定一个无向图 $G = (V, E)$, G 的一个团 C 是 V 的子集, 使得 C 中的每一对顶点相互相邻。最大团问题(MCP) 在于找到顶点数最大的团。MCP的一个重要推广是带权最大团问题 (MWCP), 其中每个顶点都有一个非负整数作为点权, 目标是找到一个总点权最大的团。显然, 如果每个顶点具有相同的权重, MWCP将退化为MCP。MWCP已广泛应用于许多领域, 从理论计算机科学到有价值的应用 (Ballard和Brown 1982; Balasndaram和Butenko 2006; Gomez Ravetti和Moscato 2008) 。

众所周知, MCP的决策版本是Karp最著名的21个NP完全组合问题之一 (Karp 1972) 。MCP和MWCP都已被证明是NP-hard问题, 最先进的近似算法只能达到 $O(n(\log \log n)^2 / (\log n)^3)$ 的近似比率 (Feige 2004) 。因此, 在合理的时间, 为了找到一个“好”的团, 人们付出了巨大的努力, 这是很常见的。到目前为止, MCP和MWCP主要有两种算法, 即精确算法和启发式算法。

人们提出了许多精确的算法来解决MCP和MWCP。经典的分支限界算法是MCQ (Tomita and Seki 2003)，它使用启发式顶点顺序进行独立集划分。MCQ算法通过动态计算顶点的度数得到进一步改进，从而得到MaxCliqueDyn算法 (Konc和Janezic 2007)。最近，另一种范式将MCP编码到MaxSAT中，然后应用MaxSAT来改进上界 (Li and Quan 2010; Li, Fang, and Xu 2013)。对于MWCP，(Babel 1994)提出了一种早期的分支限界算法。一种基于纠错码改进的分支限界算法在 (Osterg ard 2001) 中被提出。(Yamaguchi and Masuda 2008) 根据从原始图构造的有向无环图中的最长路径计算上界。最近，Fang et. al 提出一种基于MaxSAT的MWCP算法，应用Top-k failed literal检测来改进上界 (Fang et al. 2014)。

虽然精确算法可以保证其解的最优性，但它们可能无法解决大规模的困难实例。对于求解大型实例，一种流行的方法是局部搜索，它可以在合理的时间内找到近似解。MCP有许多局部搜索算法 (Singh and Gupta 2006; Pullan and Hoos 2006; Pullan 2006; Guturu and Dantu 2008; Wu and Hao 2013; Benlic and Hao 2013)。在这些算法中，DLS (Pullan and Hoos 2006) 是一个里程碑算法，它采用在搜索过程中动态调整的顶点惩罚。DLS进一步改进为叫做分阶段局部搜索 (PLS) 的两阶段算法 (Pullan 2006)。在 (Wu and Hao 2013) 中，提出了基于k-固定惩罚策略的禁忌搜索算法。(Benlic and Hao 2013)介绍了解决MCP的突破性局部搜索方法。此外，MCP与最小顶点覆盖 (MinVC) 和最大独立集 (MaxIS) 问题密切相关，解决MinVC和MaxIS算法可以直接用来解决MCP问题。

与MCP相比，关于MWCP的启发式方法相对较少。原因可能是，从算法设计的角度来看，MWCP更复杂。在 (Bomze, Pelillo, and Stix 2000) 中，开发了一种基于动力学原理的平行分布式启发式方法来逼近MWCP，并在数学生物学的各个分支中进行研究。Busygin (Busygin 2006) 提出了一种新的快速启发式方法，使用非线性编程表述MWCP。Pullan将相位局部搜索 (PLS) 算法扩展到MWCP (Pullan 2008)。根据文献记载，目前最好的MN/NT (Wu, Hao, and Glover 2012)，这是一种多邻域局部搜索算法，其主要特征包括一个组合邻域和一个专门的tabu机制。

在本文中，我们为MWCP提出了两种局部搜索算法。首先，我们提出了一个新的启发式算法，它是配置检查 (CC) 策略的一个变种。CC是最近提出的一种在局部搜索过程中避免循环问题的机制，并且已经成功地应用于一些NP-hard问题，如MinVC (Cai, Su, and Sattar 2011)，SAT (Cai and Su 2012; 2013; Andre, Djamel, and Donia 2014)，以及MaxSAT (Luo et al. 2015)。我们遵循这一研究思路，尝试应用CC策略来解决MWCP。然而，直接应用CC策略并不能带来成功的算法。因为在MWCP的背景下，CC的禁止强度通常太弱。在MWCP的背景下。我们提出一种新的策略，叫做 Strong CC策略 (简称 SCC)，它比CC更严格，并能减少更多不必要的搜索区域。基于 SCC，我们开发了一种叫做LSCC (带 SCC的局部(Local)搜索) 的局部搜索算法。将LSCC与最先进的局部搜索算法MN/TS的比较实验表明，它在标准基准DIMACS (Johnson和 Trick 1996) 和BHOSLIB (Xu等人, 2005) 上表现得更好。

此外，为了提高在大规模图上的性能，我们应用了一个低复杂度的启发式算法，称为Best from Multiple Selection (BMS)，以快速有效地选择交换顶点对。最近的一项工作 (Cai 2015) 提出了一种简单而快速的局部搜索算法，称为FastVC，用于解决大规模图中的MinVC，该算法基于两个低复杂度启发式算法。受BMS在FastVC中的成功启发 (Cai 2015)，我们也使用了BMS启发式算法，它近似于最佳贪婪交换启发式(best-greedy swap heuristic) (Wu, Hao, and Glover 2012)，并且具有较低的复杂度。我们还用SCC策略加强BMS启发式。利用BMS启发式，我们改进了LSCC，得到的算法被称为LSCC+BMS，同时也改进了MN/TS，得到MN/TS+BMS。实验表明，LSCC+BMS在广泛的大规模图上的表现优于MN/NT及其改进版MN/TS+BMS。(Rossi and Ahmed 2015)。我们还进行了实验来分析这两种启发式方法的有效性。

在下一节中，我们将介绍一些必要的背景知识。然后，我们提出了MWCP的SCC策略，给出了LSCC算法，并进行了相关实验。然后，利用BMS启发式算法对大规模图上的LSCC进行了改进，得到了LSCC+BMS算法，并对大规模图进行了实验。最后，我们得出结论并概述了未来的工作。

准备

给定一个无向图 $G=(V,E)$ ，其中 $V=\{v_1, v_2, \dots, v_n\}$ 是顶点的集合， $E=\{e_1, e_2, \dots, e_m\}$ 是边的集合。在图 G 中，每条边都是一个由 V 中两个元素组成的集合。对于一条边 $e=\{v,u\}$ ，我们说顶点 u 和 v 是改变的端点，并且称 u 与 v 相邻。一个团 C 是 V 的一个子集，其中每一对顶点都是相邻的。MCP问题就是要找到一个拥有最多顶点的团。当每个顶点 v_i 都有一个正整数的权重时，MCP被扩展为MWCP，MWCP问题要求找到一个总权重最大的团。给定一个加权函数 $w: V \rightarrow \mathbb{Z}^+$ ，一个团 C 的权重是 $w(C)=\sum_{v \in C} w(v)$ 。一个顶点 v 的邻域 $N(v)=\{u \in V \mid (v,u) \in E\}$ 。对于一个顶点 v 来说，它的age被定义为自它最后一次改变状态（被选择或不被选择）以来的步数。

通常，MWCP的局部搜索算法（类似在MCP中）保持一个当前的团 C ，并通过三个运算符反复修改它：Add、Drop和Swap。操作符 "Add" 是指将一个顶点添加到团 C 中，条件是该顶点与 C 中的所有顶点相邻。操作符 "Drop"是指从 C 中删除一个顶点。操作符"Swap"将一个顶点 $u \in C$ 与另一个顶点 $v \notin C$ 交换，条件是 v 与 C 中除 u 外的每个顶点都相邻。通常，只有当Add和 "好的"Swap操作不可能时，才考虑操作Drop。

CC(Configuration Checking)回顾

重访搜索空间的同一部分被称为循环问题(cycling problem)，这是局部搜索中的一个严重问题。最近，Cai等人提出了一种叫做配置检查（CC）的策略（Cai, Su, and Sattar 2011），它利用问题的结构来减少局部搜索的循环。CC策略已经成功地用于组合优化问题的局部搜索算法，如MinVC（Cai, Su, and Sattar 2011）和Set Covering（Wang et al. 2015），以及约束满足问题，如Satisfiability（Cai and Su 2013; Andre, Djamel, and Donia 2014）和Maximum Satisfiability（Luo et al. 2015）。

粗略地说，对于那些任务是找到一个最佳元素集的组合问题，CC的思想可以描述如下。对于一个元素（如一个顶点），如果它的配置(configuration)与上次从候选集(candidate set)中移除时相同，那么它就被禁止重新加入候选集。通常，一个顶点的配置是指其相邻顶点的状态。CC策略通常用一个名为confChange的布尔数组来实现，其中 $\text{confChange}(v)=1$ 表示允许将 v 添加到候选解中， $\text{confChange}(v)=0$ 表示禁止将 v 添加到候选解中。

可以很容易地设计出MWCP的直接CC策略。在开始时，每个顶点 v 的 $\text{confChange}(v)$ 被初始化为1，因为每个顶点最初都被允许被选择。在搜索过程中，当一个顶点 v 被Add到当前的团中时， $\text{confChange}(v)$ 对每个顶点 $v' \in N(v)$ 都被设置为1。当一个顶点 v 从当前的团中Drop时， $\text{confChange}(v)$ 被设置为0，而 $\text{confChange}(v)$ 对每个顶点 $v' \in N(v)$ 被设置为1。对于Swap步骤，当一个顶点 v 从当前团中移除，顶点 u 被添加到团中，那么 $\text{confChange}(v)$ 被设置为0；对于 $v' \in N(v) \cup N(u)$ ， $\text{confChange}(v')$ 被设置为1。

SCC (Strong Configuration Checking)

在这一节中，我们讨论CC策略应用于MWCP问题时的缺点，并且提出CC策略用以解决MWCP和MCP问题时的一种变体，即SCC。

我们观察到，在具有 "Add"、"Drop" 和 "Swap"三个操作符的局部搜索算法中，CC策略会因为允许添加太多的顶点而误导搜索。根据CC，相关顶点的confChange值会随着每个操作的进行而更新。然而，一些直觉分析表明，并不总是建议在每次操作后将相邻顶点的confChange值设置为1。

对于Add操作来说，团被一个顶点所扩展，因此允许被选中的顶点的邻居通过将其confChange值设置为1而被添加是非常合理的。事实上，这些顶点非常被鼓励加入团中。

Drop操作表明算法遇到了一个局部最优，并通过从团中移除一个顶点来回滚。在这种情况下，我们认为不应该鼓励被删除顶点的相邻顶点被加入到团中。

Swap操作通常作为一种多样化的形式，通过引导搜索切换到另一个靠近当前团的团。由于我们不确定Swap步骤是否将搜索引向更好的团，在我们的算法中，我们采取了一种保守的策略--不鼓励被交换顶点的更多相邻顶点，而是只鼓励那些confChange值已经为1的顶点。

基于上述考虑，我们将CC修改为一个更具限制性的版本，称为强配置检查（SCC）。这种启发式方法由以下四条规则规定：

- **SCC-InitialRule.** 在搜索算法的开始，对每个顶点 v ，将 $\text{confChange}(v)$ 设置为1.
- **SCC-AddRule.** 当 v 被加入当前团时，对于每个 $v' \in N(v)$ ，将 $\text{confChange}(v')$ 设置为1.
- **SCC-DropRule.** 当 v 从当前团移除时，将 $\text{confChange}(v)$ 设置为0.
- **SCC-SwapRule.** 当 u 从当前团移除、 v 加入当前团时，将 $\text{confChange}(u)$ 设置为0.

简而言之，SCC只允许 v 被添加到当前团，仅当在 v 上次被移除后， v 的一些邻居被添加到当前团中，而CC则允许 v 被添加到当前团当 v 的一些邻居被添加或移除。CC策略通常与加权技术配合得很好，所以在我们的算法中缺少加权技术可能是原始CC策略失败的一个原因。我们还注意到，在SAT中存在一个叫做有希望的变量(promising variable)的概念 (Li and Huang 2005)，如果一个变量的分数因为其相邻变量的翻转而变成正数，那么它就可以被翻转。这个概念在某种意义上与包括SCC在内的CC策略相似。

LSCC算法

基于SCC启发式算法，我们开发了一种名为LSCC (Local search with SCC) 的本地搜索算法。LSCC与三个运算符 "Add"、"Swap" 和 "Drop" 一起工作。我们分别为Add和Swap操作维护一个集合。用 C 来表示当前的团，这两个集合的定义如下。Drop操作的集合是就是 C 。

$$\text{AddSet} = \{v | v \notin C, v \in N(u) \text{ for } \forall u \in C\}$$

$$\text{SwapSet} = \{(u, v) | u \in C, v \notin C, v \in N(y) \text{ for } \forall y \in C \setminus \{u\}\}$$

我们用 Δ_{add} 、 Δ_{drop} 和 Δ_{swap} 分别表示操作Add、Drop和Swap对 $w(C)$ 值的改变。很明显，我们可以根据以下公式来计算它们。

- 对于一个顶点 $v \in \text{AddSet}$ ， $\Delta_{\text{add}}(v) = w(v)$
- 对于一个顶点 $u \in C$ ， $\Delta_{\text{drop}}(u) = -w(u)$
- 对于一个点对 $(u, v) \in \text{SwapSet}$ ， $\Delta_{\text{swap}}(u, v) = w(v) - w(u)$

在我们的算法中，操作的顶点在上下文中是明确的，因此被省略了。

LSCC的伪代码如下：

Algorithm 1: LSCC ($G, cutoff$)

Input: graph $G = (V, E, w)$, the *cutoff* time

Output: A maximum weight clique C of G

```
1  $C^* := \emptyset$ 
2 while elapsed time < cutoff do
3   initialize confChange;
4    $C := \text{InitGreedyConstruction}()$ ;
5    $C_{localbest} := C$ ;
6   for  $step = 0; step < L; step++$  do
7      $v :=$  a vertex in AddSet with the biggest  $\Delta_{add}$  and
       $confChange(v) = 1$ , breaking ties in favor of the
      oldest one;
8      $(u, u') :=$  a vertex pair in SwapSet such that
       $confChange(u') = 1$  with the biggest  $\Delta_{swap}$ ,
      breaking ties in favor of the oldest one;
9     if AddSet  $\neq \emptyset$  then
10       $C := (\Delta_{add} > \Delta_{swap})? (C \cup \{v\}):$ 
       $(C \setminus \{u\} \cup \{u'\})$ ;
11    else
12       $x :=$  a vertex in  $C$  with the biggest  $\Delta_{drop}$ ,
      breaking ties in favor of the oldest one;
13       $C := (\Delta_{drop} > \Delta_{swap})? (C \setminus \{x\}):$ 
       $(C \setminus \{u\} \cup \{u'\})$ ;
14    update confChange according to SCC rules;
15    if  $w(C) > w(C_{localbest})$  then  $C_{localbest} := C$ ;
16  if  $w(C_{localbest}) > w(C^*)$  then  $C^* := C_{localbest}$ ;
17 return  $C^*$ ;
```

在开始时，LSCC将发现的最优团 C^* 初始化为一个空集。有一个外循环（第2-16行）和一个内循环（第6-15行）。在每个内循环中（ $step < L$ ），LSCC搜索一个局部最优的团，表示为 $C_{localbest}$ 。在每个内循环之后，如果 $w(C_{localbest})$ 大于 $w(C^*)$ ， C^* 就被 $C_{localbest}$ 更新（第16行）。最后，当算法达到一个时间限制时，LSCC返回 C^* 。

在每个内循环之前，LSCC通过迭代选择一个与 C 中所有顶点相邻的顶点，贪婪地构建一个初始候选解 C ，直到不存在这样的顶点为止，并随机打破联系（第4行）。贪婪的初始化过程非常简单，对于大规模的图来说仍然有效。同时，由于采用了随机打破联系的机制，该程序能够在不同的回合中找到多样化的初始解决方案。然后， $C_{localbest}$ 被初始化为 C （第5行）。

在每个内循环中，LSCC选择一个操作者来修改当前的团 C 。它首先选择一个顶点 $v \in \text{AddSet}$ ，使得其 Δ_{add} 最大且 $confChange(v)=1$ （第7行），并选择一个交换对 $(u, u') \in \text{SwapSet}$ ，使得 $confChange(u')=1$ ，具有最大的 Δ_{swap} （第8行）。两个联系都是通过优先选择最古老的一个来打破。如果可能有一个Add操作，LSCC对 Δ_{add} 和 Δ_{swap} 进行比较，并选择效益较大的操作来执行（第9-10行）。相反，如果AddSet是空的，这

意味着不可能有添加操作，那么LSCC会执行Swap或Drop操作。它挑选了一个顶点 $x \in C$ ，它具有最大的 Δ_{drop} （即最小的权重）（第12行），然后比较 Δ_{swap} 和 Δ_{drop} ，并选择具有较大效益的操作来执行（第13行）。

每次操作后， confChange 的值根据相应的SCC规则进行更新（第14行），如果 $w(C)$ 大于 $w(C_{\text{localbest}})$ ， $C_{\text{localbest}}$ 就被 C 更新（第15行）。

根据标准基准对LSCC进行评估

我们进行了广泛的实验来评估MWCP的LSCC算法在两个标准基准上的性能，包括DIMACS和BHOSLIB。DIMACS基准来自于第二届DIMACS实施挑战赛（Johnson and Trick, 1996），包括来自真实应用的问题和随机生成的图。BHOSLIB实例是根据相变区的RB模型随机生成的（Xu等人，2005）。这些实例最初是未加权的，为了获得相应的MWCP实例，我们使用了与（Pullan 2008; Wu, Hao, and Glover 2012）中相同的方法。对于第 i 个顶点 v_i ， $w(v_i) = (i \bmod 200) + 1$ 。

作为比较，我们选择MN/TS（Wu, Hao, and Glover 2012）来代表解决MWCP的最先进算法。MN/TS是开源的，用C++实现。我们的算法LSCC也是用C++实现的。这两种算法都是由g++ 4.6.2的-O2选项编译的。对于搜索深度 L ，MN/TS和LSCC为所有实例设置了 $L=4000$ 。MN/TS采用tabu启发式，tabu tenure TL 被设置为7，如（Wu, Hao, and Glover 2012）。

为了证明SCC启发式的有效性，我们还将LSCC与它的变种LCC（Local search with CC）进行了比较，后者利用原始的CC策略（如前面所介绍的）而不是SCC。

所有的实验都在Ubuntu Linux上运行，CPU为3.1GHZ，内存为8GB。对于每个实例，每个算法用不同的随机种子进行100次独立运行，每次运行在达到给定的时间限制（1000秒）时终止。对于每个实例， w_{max} 是找到的最大的权重，而 w_{avg} 是100次运行的平均权重。我们还报告了LSCC和MN/TS发现团的最大和平均权重值之间的差异 δ_{max} 和 δ_{avg} 。

Instance	MN/TS	LCC	LSCC	$\delta_{\text{max}}(\delta_{\text{avg}})$
	$w_{\text{max}}(w_{\text{avg}})$	$w_{\text{max}}(w_{\text{avg}})$	$w_{\text{max}}(w_{\text{avg}})$	
C2000.9	10999 (10948.5)	10267(9948)	10999 (10922.6)	0(-25.9)
p_hat1500-3	10321(10314.4)	10321(10130.1)	10321(10321)	0(6.6)
MANN_a27	12281(12270.6)	12275(12268.8)	12283(12283)	2(12.4)
MANN_a45	34192(34167)	34183(34175.9)	34254(34242.1)	62(75.1)
MANN_a81	111128(111074.6)	111135(111084.8)	111135(111118.1)	7(10.2)
frb56-25-1	5916(5815.6)	5669(5588.1)	5916(5825.7)	0(10.1)
frb56-25-2	5872(5790.8)	5589(5550.7)	5886(5813.7)	14(22.9)
frb56-25-3	5859(5780.4)	5689(5545.7)	5859(5777.6)	0(-2.8)
frb56-25-4	5892(5818.9)	5712(5311.7)	5892(5821.1)	0(2.2)
frb56-25-5	5839(5750.9)	5597(5536.9)	5839(5754.2)	0(3.3)
frb59-26-1	6591(6516)	6318(6108.9)	6591(6538.3)	0(22.3)
frb59-26-2	6645(6542.8)	6320(6190.1)	6645(6546.9)	0(4.1)
frb59-26-3	6608(6579.5)	6178(6105.5)	6608(6505.7)	0(-73.8)
frb59-26-4	6592(6463.7)	6246(6076.2)	6592(6488.6)	0(24.9)
frb59-26-5	6584(6491)	6269(6100.2)	6584(6512.6)	0(21.6)

表1: MN/TS、LCC和LSCC在DIMACS和BHOSLIB基准上的实验结果。DIMACS实例中，MN/TS和LSCC能很快找到相同质量的团的情况没有报告。正的 δ_{max} 或 δ_{avg} 表示LSCC比MN/TS找到更好质量的团。

对DIMACS的实验结果见表1。大多数DIMACS实例非常简单，以至于MN/TS和LSCC很快就能找到相同质量的团，因此没有报告。结果显示，在DIMACS实例上，LSCC比MN/TS和LCC找到了更好的质量的团。特别是，LSCC在MANN a27、MANN a45和MANN a81上获得了新的最佳解决方案。LSCC在MANN领域的表现一直很出色。对于p hat1500-3，LSCC是唯一能在100%的运行实例中持续找到大小为10321的团的算法。最后，我们注意到LSCC成功地找到了所有DIMACS实例的最佳已知解，这表明它的鲁棒性。

BHOSLIB实例的结果也显示在表1中。为了专注于困难的实例，我们只介绍了两组最大的实例，它们比其他小的实例要困难得多。结果表明，LSCC在这些实例中的表现优于MN/TS。此外，LSCC提高了一个实例frb56-25-2的最大clique。对于两种算法都能找到相同质量的最大权重团的实例，除了frb56-25-3和frb59-26-3，LSCC找到的团的平均权重大于MN/TS。最后，LSCC和LCC的比较也证实了SCC启发式的有效性。

改进大规模图的LSCC

尽管LSCC在标准基准上表现得相当好，但它在大规模图上却不那么有效。在这一节中，我们采用了一种叫做“最佳多选”（Best from Multiple Selection, BMS）的启发式算法来改进LSCC，从而形成了一种叫做LSCC+BMS的改进算法。我们通过在广泛的大规模图上的实验，展示了LSCC+BMS的效率和基础启发式算法的有效性。

BMS启发式算法和LSCC+BMS算法

在LSCC中，我们使用最佳选择启发式从SwapSet中选择效益最好的交换顶点对（w.r.t. Δ_{swap} ）来进行Swap。有了合适的标准，这种启发式可以引导搜索走向最有希望的区域，因此在局部搜索算法中普遍采用（Wu, Hao, and Glover 2012; Cai等人, 2013）。这种最佳选择的启发式方法适用于大多数情况，但在大规模图中效果不佳，因为大规模图中的SwapSet通常非常大，寻找最佳配对不仅浪费大量时间，而且不能保证此举是对解决方案质量的最佳选择。

基于上述考虑，我们采用了一种快速有效的启发式方法，叫做最佳多选(BMS)，可以花费很少的时间从SwapSet中选择质量较高的顶点对。BMS启发式在顶点对的质量和复杂度之间取得了良好的平衡。对BMS启发式的正式描述见算法2。

Algorithm 2: the BMS heuristic

```

1  pick a random vertex pair  $(v, v') \in \text{SwapSet}$  with
    $\text{confChange}(v')=1$ ;
2   $\Delta_{\text{swap}}^* := \Delta_{\text{swap}}(v, v')$ ;
3  for  $i = 0; i < k; i++$  do
4      pick a random vertex pair  $(u, u') \in \text{SwapSet}$  with
        $\text{confChange}(u')=1$ ;
5      if  $(\Delta_{\text{swap}}(u, u') > \Delta_{\text{swap}}^*) \vee ((\Delta_{\text{swap}}(u, u') =$ 
         $\Delta_{\text{swap}}^* \wedge \text{age}(u') < \text{age}(v')))$  then
6           $(v, v') := (u, u')$ ;
7           $\Delta_{\text{swap}}^* := \Delta_{\text{swap}}(u, u')$ ;
8  return  $(v, v')$ ;
```

基本上，BMS启发式随机选择 k 个交换对 (v, v') ，然后返回最佳交换对，即最大化 Δ_{swap} 的值，其中 k 是一个参数。加速BMS的一个技巧是当 $|\text{SwapSet}| < k$ 时选择最佳配对。此外，我们使用SCC策略来帮助BMS排除一些不合理的顶点配对。

我们算法中的BMS启发式和（Cai 2015）中的原始BMS启发式有两个区别。首先，FastVC中的BMS启发式是用来选择要放弃的顶点，而我们算法中的BMS是用来选择交换顶点对。其次，更重要的是，我们在BMS启发式中结合了配置检查技术(CC)，以修剪一些 "没有希望的 "候选点对，而FastVC中的BMS没有任何机制来排除没有希望的候选点对。

我们使用BMS启发式来改进LSCC算法，只需用BMS启发式取代选择交换顶点对的最佳选择启发式（即算法1的第8行）。因此，所产生的算法被称为LSCC+BMS。

在大规模图上的实验

我们在网络数据存储库在线（Rossi and Ahmed 2015）的真实世界的大规模图上评估LSCC+BMS，这些图最近被用于测试局部搜索方法和并行算法的性能（Rosin 2014；Rossi等人2014；Cai 2015）。由于篇幅原因，我们没有报告少于1000个顶点的图的结果，对于这些图，两种算法都能快速找到相同质量的解决方案。

需要注意的是，MN/TS在许多大规模图中都不能找到一个团，这主要是由于其昂贵的内存数据结构和高复杂度的启发式方法。为了进行更有趣的比较，我们通过更好的数据结构和BMS启发式来改进MN/TS，使其也能很好地处理大规模图。最终的算法被称为MN/TS+BMS。对于LSCC+BMS和MN/TS+BMS中的BMS启发式，根据一些初步实验，我们将 k 参数设置为100。

实验设置与上一节相同。在本实验中， δ_{max} 和 δ_{avg} 表示LSCC+BMS和MN/TS+BMS找到的团的最大和平均权重值的差异。另外，有相当一部分实例，LSCC+BMS和MN/TS+BMS在所有运行中都能找到相同质量的团，即 $\delta_{\text{max}}(\delta_{\text{avg}}) = 0(0)$ 。对于这些实例，我们报告另一个统计数字 δ_{time} ，它代表LSCC+BMS和MNTS之间的运行时间差。对于MN/TS未能在规定时间内找到团的情况，MN/TS一栏被标记为 "不适用 (n/a)"。

表2总结了大规模图上的结果，其中 δ_{max} 或 δ_{avg} 为正表示LSCC+BMS比MN/TS+BMS找到了质量更好的团。MN/TS基本上比其他两种算法差，我们重点讨论MN/TS+BMS和LSCC+BMS的比较。总的来说，LSCC+BMS在这些大规模图上比MN/TS+BMS找到更好的解决方案。特别是，我们观察到LSCC+BMS在17个图上找到了MN/TS+BMS无法达到的团，而在另外20个图上，它们都能找到相同质量的团，LSCC+BMS以更好的平均解决方案质量做到这一点。对于剩下的49个实例，这两种算法都能找到相同质量的解决方案。对于这49个实例中的40个，LSCC+BMS比MN/TS+BMS快。在这49个实例中，LSCC+BMS的平均运行时间只有MN/TS+BMS的一半。

Instance	MN/TS w_{max} (w_{avg})	MN/TS+BMS w_{max} (w_{avg})	LSCC+BMS w_{max} (w_{avg})	δ_{max} (δ_{avg})	δ_{time}
bio-dmela	805(805)	805(805)	805(805)	0(0)	0
bio-yeast	629(629)	629(629)	629(629)	0(0)	0.42
ca-AstroPh	5338(5338)	5338(5338)	5338(5338)	0(0)	46.14
ca-citeSeer	n/a	8838(8838)	8838(8838)	0(0)	133.22
ca-coauthors-dblp	n/a	37884(28196)	37884(34622)	0(6426)	
ca-CondMat	n/a	2887(2887)	2887(2887)	0(0)	24.1
ca-CSphd	489(489)	489(489)	489(489)	0(0)	0.1
ca-dblp-2010	n/a	7575(7087.9)	7575(7479.8)	0(391.9)	
ca-dblp-2012	n/a	14108(10197)	14108(14108)	0(3911)	
ca-Erdos992	958(958)	958(958)	958(958)	0(0)	0.19
ca-GrQc	4279(4279)	4279(4279)	4279(4279)	0(0)	0.27
ca-HepPh	24533(24533)	24533(24533)	24533(24533)	0(0)	0.59
ca-hollywood-09	n/a	222720(121846)	222720(211311)	0(89465)	
ca-MathSciNet	n/a	2792(2374.3)	2792(2543)	0(168.7)	
ia-email-EU	n/a	1350(1350)	1350(1350)	0(0)	-0.36
ia-email-univ	1473(1473)	1473(1473)	1473(1473)	0(0)	0.05
ia-enron-large	n/a	2490(2490)	2490(2490)	0(0)	-2.54
ia-fb-messages	791(791)	791(791)	791(791)	0(0)	0.01
ia-reality	374(374)	374(374)	374(374)	0(0)	0.56
ia-wiki-Talk	n/a	1884(1884)	1884(1884)	0(0)	-1.19
inf-power	888(888)	888(888)	888(888)	0(0)	0.51
inf-roadNet-CA	n/a	597(594.5)	752(613.4)	155(18.9)	
inf-roadNet-PA	n/a	597(596.5)	599(599)	2(2.5)	
rec-amazon	n/a	942(942)	942(942)	0(0)	7.63
sc-lldoor	n/a	4018(3836.1)	4018(3936.4)	63(100.3)	
sc-msdoor	n/a	4088(3959.4)	4088(4043.9)	0(84.5)	
sc-nasasrb	n/a	4548(4441)	4548(4548)	0(107)	
sc-pkustk11	n/a	5091(4769.8)	5298(5298)	207(528.2)	
sc-pkustk13	n/a	5853(5565.4)	5928(5874.6)	75(309.2)	
sc-pwtk	n/a	4548(4372)	4620(4603.2)	72(231.2)	
sc-shipsec1	n/a	3255(3100.4)	3540(3381.5)	285(281.1)	
sc-shipsec5	n/a	4500(4338.8)	4524(4445.4)	24(106.6)	
soc-BlogCatalog	n/a	4803(4803)	4803(4803)	0(0)	35.90
soc-brightkite	n/a	3672(3653.8)	3672(3655.7)	0(1.9)	
soc-buzznet	n/a	2981(2981)	2981(2981)	0(0)	22.86
soc-delicious	n/a	1547(1523.3)	1547(1543.5)	0(20.2)	
soc-digg	n/a	4675(4675)	5303(4800.6)	628(125.6)	
soc-douban	n/a	1682(1682)	1682(1682)	0(0)	19.35
soc-epinions	n/a	1657(1657)	1657(1657)	0(0)	27.48
soc-flickr	n/a	7050(6998.1)	7083(7083)	33(84.9)	
soc-flixster	n/a	3805(3036.4)	3805(3500.9)	0(464.5)	
soc-FourSquare	n/a	3064(3043.6)	3064(3053.6)	0(10)	
soc-gowalla	n/a	2335(2209)	2335(2291.8)	0(82.8)	
soc-lastfm	n/a	1773(1773)	1773(1773)	0(0)	65.84
soc-livejournal	n/a	2521(2050.4)	3120(2327.7)	599(277.3)	
soc-LiveMocha	n/a	1784(1784)	1784(1784)	0(0)	-4.97
soc-pokec	n/a	2341(1984.3)	3191(2075.3)	850(91)	
soc-slashdot	n/a	2811(2811)	2811(2811)	0(0)	-21.24
soc-twitter-follows	n/a	808(785.1)	808(808)	0(22.9)	
soc-youtube	n/a	1961(1961)	1961(1961)	0(0)	-18.63
soc-youtube-snap	n/a	1787(1787)	1787(1787)	0(0)	-51.79
socfb-A-anon	n/a	2576(2096.5)	2777(2196.4)	201(99.9)	
socfb-B-anon	n/a	2513(1986.9)	2537(2071.3)	24(84.4)	
socfb-Berkeley13	n/a	4906(4906)	4906(4906)	0(0)	7.24
socfb-CMU	4141(4141)	4141(4141)	4141(4141)	0(0)	1.11
socfb-Duke14	3694(3694)	3694(3694)	3694(3694)	0(0)	12.25
socfb-Indiana	n/a	5412(5412)	5412(5412)	0(0)	29.67
socfb-MIT	3658(3658)	3658(3658)	3658(3658)	0(0)	0.74
socfb-OR	n/a	3523(3523)	3523(3523)	0(0)	120.2
socfb-Penn94	n/a	4738(4709.3)	4738(4738)	0(28.7)	
socfb-Stanford3	5769(5769)	5769(5769)	5769(5769)	0(0)	10.52
socfb-Texas84	n/a	5546(5524.5)	5546(5546)	0(21.5)	
socfb-UCLA	n/a	5595(5595)	5595(5595)	0(0)	26.42
socfb-UConn	5733(5733)	5733(5733)	5733(5733)	0(0)	2.24
socfb-UCSB37	5669(5669)	5669(5669)	5669(5669)	0(0)	46.66
socfb-UF	n/a	6043(6021)	6043(6043)	0(22)	
socfb-Ullinois	n/a	5730(5721.6)	5730(5730)	0(8.4)	
socfb-Wisconsin87	n/a	4239(4239)	4239(4239)	0(0)	27.29
tech-as-caida2007	n/a	1869(1869)	1869(1869)	0(0)	-0.41
tech-as-skitter	n/a	5527(4387)	5703(5271.8)	176(884.8)	
tech-internet-as	n/a	1692(1692)	1692(1692)	0(0)	-0.38
tech-p2p-gnutella	n/a	703(675.8)	703(703)	0(27.2)	
tech-RL-caida	n/a	1861(1861)	1861(1861)	0(0)	26.25
tech-routers-rl	1460	1460(1460)	1460(1460)	0(0)	0.12
tech-WHOIS	6154(6154)	6154(6154)	6154(6154)	0(0)	19.60
web-arabic-2005	n/a	10558(10529)	10558(10558)	0(29)	
web-BerkStan	3249(3249)	3249(3249)	3249(3249)	0(0)	2.2
web-edu	2077(2077)	2077(2077)	2077(2077)	0(0)	5.06
web-google	1749(1749)	1749(1749)	1749(1749)	0(0)	0.1
web-indochina-2004	6997(6997)	6997(6997)	6997(6997)	0(0)	8.58
web-it-2004	n/a	43842(36402)	45477(45313)	1635(8911)	
web-sk-2005	n/a	11925(10775)	11925(11925)	0(1150)	
web-spam	2503(2503)	2503(2503)	2503(2503)	0(0)	3.4
web-uk-2005	n/a	54850(54850)	54850(54850)	0(0)	467.52
web-webbase-2001	3574(3574)	3574(3574)	3574(3574)	0(0)	110.69
web-wikipedia2009	n/a	1997(1582.3)	3455(2451)	1458(868.7)	

Table 2: Experiment results on the massive graphs.

SCC和BMS的效果

为了研究SCC和BMS启发式方法的效果，我们将LSCC+BMS与LSCC和LCC进行比较。请注意，LSCC使用了SCC而没有使用BMS，而LCC使用了原始CC策略。表3显示LSCC比LCC找到更好的解决方案，这说明了SCC在大规模图上的有效性。由于采用了BMS策略，LSCC+BMS在 w_{max} 和 w_{avg} 方面都比LSCC获得了更好的团。

Instance	LCC	LSCC	LSCC+BMS
	$w_{max}(w_{avg})$	$w_{max}(w_{avg})$	$w_{max}(w_{avg})$
ca-coauthors-dblp	31925(25484.4)	37884 (34425.8)	37884(34622.6)
ca-dblp-2010	7575 (6966.8)	7575 (7439.8)	7575(7479.8)
ca-hollywood-2009	222720 (90209.1)	222720 (199902.8)	222720(211311.4)
ca-MathSciNet	2611(1991)	2611(2393.1)	2792(2543)
inf-roadNet-CA	594(574.9)	597(597)	752(613.4)
inf-roadNet-PA	597(579.3)	597(597)	599(599)
sc-lldoor	4060(3733.7)	4074(3922.5)	4081(3936.4)
sc-msdoor	3941(3749.9)	4074(4036.7)	4088(4043.9)
sc-pkustk11	5298 (4741.9)	5298 (5090.5)	5298(5298)
sc-pkustk13	5853(5662.8)	5928 (5864.1)	5928(5874.6)
sc-shipsec1	3540 (3116.8)	3540 (3373.2)	3540(3381.5)
sc-shipsec5	4440(4041.8)	4500(4444.8)	4524(4445.4)
socfb-B-anon	1907(1521.7)	2470(1993.2)	2537(2071.3)
soc-delicious	1466(1446.5)	1547 (1542.8)	1547(1543.5)
soc-digg	4429(4240.3)	4675(4675)	5303(4800.6)
soc-flickr	6717(6138.1)	7083 (7058.1)	7083(7083)
soc-flixster	3311(2184.3)	3805 (3162.3)	3805(3500.9)
soc-FourSquare	3038(2982.5)	3064 (3024.7)	3064(3053.6)
soc-lastfm	1695(1599.9)	1773 (1769.4)	1773(1773)
soc-pokec	1960(1619.3)	3191 (2020.2)	3191(2075.3)
soc-youtube-snap	1787 (1571.9)	1787 (1744.2)	1787(1787)
tech-as-skitter	5506(4302.4)	5703 (5258.2)	5703(5271.8)
web-arabic-2005	10445(10445)	10558 (10546.7)	10558(10558)
web-wikipedia2009	1879(1087.3)	1997(1378.9)	3455(2451)

Table 3: Comparing LCC, LSCC and LSCC+BMS on typical massive graphs

结论

我们为最大带权团问题（MWCP）开发了两种局部搜索算法。我们首先提出了一种配置检查（CC）策略的变体，称为强配置检查（SCC），它被用于开发一种名为LSCC的局部搜索算法。在标准基准上的实验表明它比目前最好的MWCP局部搜索算法，即MN/TS算法更有优势。

我们通过应用一种划算的启发式方法来选择交换顶点对，即多选最佳（BMS），进一步改进了LSCC，得到了LSCC+BMS算法，用于大规模图。我们还使用BMS来改进MN/TS算法。在大规模图上的实验结果表明，BMS启发式明显提高了算法在大规模图上的性能，而且LSCC+BMS明显比MN/TS+BMS的性能好。我们还进行了大量的实验来分析SCC和BMS启发式算法的有效性。

在未来，我们计划在MWCP和MCP的背景下进一步研究CC的变体，并利用顶点的其他属性，如subscore（Cai和Su 2013），以改善算法。对于大规模图来说，设计低复杂度的启发式算法来改进MWCP的局部搜索算法中的Add和Drop操作是非常有趣的。

致谢

这项工作得到了中国国家973计划2014CB340301、国家自然科学基金委 (61370156、61502464、61503074) 和大学新世纪优秀人才计划 (NCET-13-0724) 的部分支持。我们要感谢匿名审稿人提出的有益意见。

引用

Acknowledgments

This work was supported in part by China National 973 program 2014CB340301, NSFC under Grant Nos. (61370156, 61502464, 61503074) and Program for New Century Excellent Talents in University (NCET-13-0724). We would like to thank the anonymous referees for their helpful comments.

References

André, A.; Djamal, H.; and Donia, T. 2014. Improving configuration checking for satisfiable random k-SAT instances. In *Proceedings of International Symposium on Artificial Intelligence and Mathematics, ISAIM 2014*.

Babel, L. 1994. A fast algorithm for the maximum weight clique problem. *Computing* 52(1):31–38.

Balasundaram, B., and Butenko, S. 2006. Graph domination, coloring and cliques in telecommunications. In *Handbook of Optimization in Telecommunications*. 865–890.

Ballard, D., and Brown, C. 1982. Computer vision. *New Jersey: Prentice Hall*.

Benlic, U., and Hao, J.-K. 2013. Breakout local search for maximum clique problems. *Computers & Operations Research* 40(1):192–206.

Bomze, I. M.; Pelillo, M.; and Stix, V. 2000. Approximating the maximum weight clique using replicator dynamics. *Neural Networks, IEEE Transactions on* 11(6):1228–1241.

Busygin, S. 2006. A new trust region technique for the maximum weight clique problem. *Discrete Applied Mathematics* 154(15):2080–2096.

Cai, S., and Su, K. 2012. Configuration checking with aspiration in local search for sat. In *Proceedings of AAAI 2012*, 334–340.

Cai, S., and Su, K. 2013. Local search for boolean satisfiability with configuration checking and subscore. *Artificial Intelligence* 204:75–98.

Cai, S.; Su, K.; Luo, C.; and Sattar, A. 2013. Numvc: An efficient local search algorithm for minimum vertex cover. *Journal of Artificial Intelligence Research* 687–716.

Cai, S.; Su, K.; and Sattar, A. 2011. Local search with edge weighting and configuration checking heuristics for minimum vertex cover. *Artificial Intelligence* 175(9):1672–1696.

Cai, S. 2015. Balance between complexity and quality: Local search for minimum vertex cover in massive graphs. In *Proceedings of IJCAI 2015*, 747–753.

Fang, Z.; Li, C.-M.; Qiao, K.; Feng, X.; and Xu, K. 2014. Solving maximum weight clique using maximum satisfiability reasoning. In *Proceedings of ECAI 2014*, volume 263, 303.

Feige, U. 2004. Approximating maximum clique by removing subgraphs. *SIAM Journal on Discrete Mathematics* 18(2):219–225.

Gomez Ravetti, M., and Moscato, P. 2008. Identification of a 5-protein biomarker molecular signature for predicting alzheimers disease. *PloS one* 3(9):e3111.

Guturu, P., and Dantu, R. 2008. An impatient evolutionary algorithm with probabilistic tabu search for unified solution of some np-hard problems in graph and set theory via clique finding. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 38(3):645–666.

Johnson, D. S., and Trick, M. A. 1996. *Cliques, coloring, and satisfiability: second DIMACS implementation challenge, October 11-13, 1993*, volume 26. American Mathematical Soc.

Karp, R. 1972. Reducibility among combinatorial problems. *Complexity of Computer Computations* 85–103.

Konc, J., and Janezic, D. 2007. An improved branch and bound algorithm for the maximum clique problem. *Communications in Mathematical and in Computer Chemistry* 58:569–590.

Li, C. M., and Huang, W. 2005. Diversification and de-terminism in local search for satisfiability. In *Proceedings of Theory and Applications of Satisfiability Testing, SAT 2005*, 158–172.

Li, C. M., and Quan, Z. 2010. An efficient branch-and-bound algorithm based on maxsat for the maximum clique problem. In *AAAI*, volume 10, 128–133.

Li, C.-M.; Fang, Z.; and Xu, K. 2013. Combining maxsat reasoning and incremental upper bound for the maximum clique problem. In *Proceedings of ICTAI 2013*, 939–946.

Luo, C.; Cai, S.; Wu, W.; Jie, Z.; and Su, K. 2015. CCLS: An efficient local search algorithm for weighted maximum satisfiability. *IEEE Trans. Computers* 64(7):1830–1843.

Östergård, P. R. 2001. A new algorithm for the maximum-weight clique problem. *Nordic Journal of Computing* 8(4):424–436.

Pullan, W., and Hoos, H. H. 2006. Dynamic local search for the maximum clique problem. *Journal of Artificial Intelligence Research* 159–185.

Pullan, W. 2006. Phased local search for the maximum clique problem. *Journal of Combinatorial Optimization* 12(3):303–323.

Pullan, W. 2008. Approximating the maximum vertex/edge weighted clique using local search. *Journal of Heuristics* 14(2):117–134.

Rosin, C. D. 2014. Unweighted stochastic local search can be effective for random csp benchmarks. *arXiv preprint arXiv:1411.7480*.

Rossi, R. A., and Ahmed, N. K. 2015. The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Rossi, R. A.; Gleich, D. F.; Gebremedhin, A. H.; and Patwary, M. M. A. 2014. Fast maximum clique algorithms for large graphs. In *Proceedings of the companion publication of the 23rd international conference on World wide web companion*, 365–366.

Singh, A., and Gupta, A. K. 2006. A hybrid heuristic for the maximum clique problem. *Journal of Heuristics* 12(1-2):5–22.

Tomita, E., and Seki, T. 2003. An efficient branch-and-bound algorithm for finding a maximum clique. In *Discrete mathematics and theoretical computer science*. 278–289.

Wang, Y.; Ouyang, D.; Zhang, L.; and Yin, M. 2015. A novel local search for unicost set covering problem using hyperedge configuration checking and weight diversity. *SCIENCE CHINA Information Sciences*.

Wu, Q., and Hao, J.-K. 2013. An adaptive multistart tabu search approach to solve the maximum clique problem. *Journal of Combinatorial Optimization* 26(1):86–108.

Wu, Q.; Hao, J.-K.; and Glover, F. 2012. Multi-neighborhood tabu search for the maximum weight clique problem. *Annals of Operations Research* 196(1):611–634.

Xu, K.; Boussemart, F.; Hemery, F.; Lecoutre, C.; et al. 2005. A simple model to generate hard satisfiable instances. In *Proceedings of IJCAI*, 337–342.

Yamaguchi, K., and Masuda, S. 2008. A new exact algorithm for the maximum weight clique problem. *ITC-CSCC: 2008* 317–320.