文章编号:0253-2778(2019)10-0851-10

基于 Spark 的多策略蚁群算法求解最大团问题

顾军华[1,2],王守彬[1,2],武君艳[1,2],张素琪[3]

(1.河北工业大学 人工智能与数据科学学院,天津 300401;2.河北省大数据计算重点实验室,天津 300401;
3.天津商业大学 信息工程学院,天津 300134)

摘要:社会网络分析目前是数据挖掘领域的研究热点之一,凝聚子群是测量社会网络结构的重要指标,而最大团结构是社会网络中最紧密的凝聚子群,最大团问题的研究也成为社会网络分析的一个重要角度.随着大数据的发展,图中节点的丰富性和边结构的复杂性对求解最大团问题提出了更高的要求.为此提出了一种基于 Spark 的多策略蚁群算法求解最大团的算法.首先,该算法利用多条件选点策略扩大搜索空间,增加可行解的多样性,避免了陷入局部最优解;然后,采取一个局部搜索策略来提高该算法的精度和收敛速度;最后,在 Spark 分布式平台上并行地实现了该算法,验证了算法的并行性,证明该算法提高了算法处理大规模社区网络的执行效率.

关键词:最大团;Spark;蚁群算法;蚂蚁选点策略;局部改善

中图分类号:TP391 **文献标识码**:A doi:10.3969/j.issn.0253-2778.2019.10.011

引用格式:顾军华,王守彬,武君艳,等. 基于 Spark 的多策略蚁群算法求解最大团问题[J]. 中国科学技术大学学报,2019,49(10):851-860.

GU Junhua, WANG Shoubin, WU Junyan, et al. Multi-strategy ant colony algorithm for solving the maximum clique problem based on Spark[J]. Journal of University of Science and Technology of China, 2019,49(10):851-860.

Multi-strategy ant colony algorithm for solving the maximum clique problem based on Spark

GU Junhua[1,2], WANG Shoubin[1,2], WU Junyan[1,2], ZHANG suqi[3]

(1. *School of Artificial Intelligence, Hebei University of Technology, Tianjin* 300401, *China*;
2. *Hebei Province Key Laboratory of Big Data Computing, Tianjin* 300401, *China*;
3. *School of Information Engineering, Tianjin University of Commerce, Tianjin* 300134, *China*)

Abstract: Social network analysis has become one of the hotspots in data mining research. Aggregating subgroups are important indicators for measuring the structure of social networks. The maximum clique structure is the most compact condensed subgroup in social networks. The study of the maximum clique problem has also become an important angle of social network analysis. With the development of big data, the massiveness of the nodes and the complexity of the edge structure in the graph put forward a higher requirement for solving the maximum clique problem. Therefore, a multi-strategy ant colony algorithm is proposed for solving the maximum clique problem first, the algorithm uses a multi-conditional selection strategy to expand the search space, increases the diversity of feasible solutions, and avoids falling into the local optimal solution. At the same time, a local search strategy is adopted to improve the accuracy and

收稿日期:2018-05-28;**修回日期**:2018-09-18

基金项目:国家自然科学基金(61802282);天津市自然科学基金(19JCTPJC54200);河北省创新能力提升计划(199676146H)资助.

作者简介:顾军华,男,1966 年生,博士/教授. 研究方向:智能信息处理. E-mail: jhgu@hebut.edu.cn

通讯作者:张素琪,女,博士/讲师. E-mail: zhangsuqie@163.com

convergence speed of the algorithm; Then, the algorithm is implemented in parallel on the Spark distributed platform, which verifies the parallelism of the algorithm and improves the efficiency of the algorithm in handling large-scale community networks.

Key words: maximum clique; Spark; ant colony algorithm; ants choosing strategy; local improvement strategy

0 引言

社会网络是社会行动者及其关系的集合,代表了典型的社会交往关系.在社会网络中,节点代表社会行动者,边代表社会行动者之间的关系[1].社会网络分析的一项重要任务就是揭示社会网络中的子结构,其中凝聚子群[2]是一种典型的社会网络结构,完全连接的最大团结构则是紧密度最高的凝聚子群,因此寻找社会网络中的最大团成为社会网络分析的研究热点之一[3].最大团问题也被广泛地应用到各种实际应用中,如信息检索[4],实验设计[5],约束优化方法[6],社区发现[7]等.大数据时代的社会网络关系更复杂,近年来最大团问题的求解精度和速度逐渐成为研究的重点[8,9].

蚁群算法是一种启发式搜索算法,目前已经在任务调度[10]、参数估计[11]和路径规划[12]等方面表现出了良好的性能. Fenet 等[13]首先将蚁群算法应用于最大团问题,然后研究了信息素释放在节点和边上的区别[14],实验结果表明,这两种方法都适用于搜索最大团. Xu 等[15]提出动态改变参数的方法,并将模拟退火与蚁群算法相结合求解最大团问题,大大增加了解的准确性,但收敛速度有待进一步改善.王会颖等[16]对蚂蚁选点策略进行了改进,采用轮盘赌与概率最大原则相结合的方法选取候选节点,虽然增加了可行解的多样性,但是降低了算法的收敛速度. Liu 等[17]对信息素的更新策略进行改进,提出了一种全局信息素更新和局部信息素更新相结合的方式,获得了更高的求解精度和更快的收敛速度. Soleimani 等[18]提出与粒子群算法结合的蚁群算法,该算法对信息素的更新过程进行了改进,然而改进后的算法仍有可能陷入局部最优.上述方法在不同的方面改善了蚁群算法求解最大团的效果,但是求解精度和收敛速度仍需要进一步提高.

随着互联网的快速发展,社会网络数据剧增,蚁群算法本身的时间复杂度较高,这使得在传统的单片机上使用蚁群算法处理大规模的数据集非常耗时,而且时间效率又是挖掘社会网络信息的重要因素,难以满足当下的需求.为了解决时间效率问题,有学者将蚁群算法运用到分布式平台上并行实现.夏卫雷等[19]实现了基于 MapReduce 的蚁群算法,增强了蚁群算法求解大规模数据的能力;王诏远等[20]等探讨了蚁群算法的三种并行方式以及这 3 种并行方式结合 MapReduce 编程框架的可行性和实用场景;王诏远等[21]等实现了基于 Spark 的蚁群优化算法,相比基于 MapReduced 蚁群优化算法在执行速度上有了大幅的提升;Dong 等[22]提出了在 Spark 平台上结合蚁群算法和遗传算法解决旅行商问题.

目前,在 Spark 平台上使用蚁群算法求解社会网络中的最大团问题还未见报道,因此本文首先针对蚁群算法求解最大团时出现的容易陷入局部最优解的问题提出了多条件选点策略;然后针对收敛速度慢的问题提出了一种局部搜索策略;最后针对在处理大规模数据集时的时间消耗问题,在 Spark 平台上并行地实现了该算法,提高了算法的执行效率.

1 最大团问题

给定一个无向图 $G=(V,E)$,其中 $V=\{v_1,v_2,\cdots,v_n\}$ 是节点集合,E 是边集合并且满足 $E\subseteq V\times V$.用无序对 (v_i,v_j) 表示图 G 的边,则 $v_i,v_j\in V$,$(v_i,v_j)\in E$.

定义 1.1 将 $U=(V',E')$ 称为无向图 $G=(V,E)$ 的完全子图,当且仅当对任意给定的无向图 $G=(V,E)$,若 $V'\in V$,$E'\in E$,且对 $\forall u,v\in V'$,有 $(u,v)\in E'$.

定义 1.2 将无向图 G 的完全子图 U 称为 G 的一个团,当且仅当在图 G 中没有更大的完全子图包含子图 U.

定义 1.3 将包含顶点数最多的团称为无向图 G 的最大团 C.最大团问题就是求解图 G 的最大团 C.

2 蚁群算法求解最大团

蚁群算法模型的基本思路是生物学上蚁群在寻找食物时对信息素的感知来寻找最优解.采用蚁群算法求解最大团问题主要包括 3 个任务:可行解构

造、蚂蚁选点策略及信息素更新策略.

2.1　可行解构造

设 n 只蚂蚁,一次迭代过程中每只蚂蚁均需构建一个团结构,设第 k 只蚂蚁本次迭代构建的团 $C_k=\{v_{k1},v_{k2},\cdots,v_{ki},\cdots,v_{km}\}$.其中,v_{ki} 表示第 k 只蚂蚁在第 i 次选择的节点,m 表示团的大小.首先,第 k 只蚂蚁随机地选择一个节点 $v_{k1}\in V$ 作为团的第一个节点,同时得到加点候选集合 addCandidates.加点候选集 addCandidates 中的节点需要满足与当前团 C_k 中的每一个节点都相连接且不包含在当前团 C_k 中.然后,根据 3.2 蚂蚁选点策略,从加点候选集 addCandidates 中选择节点 v_{k2} 添加到团结构 C_k 中,并更新 addCandidates.这样不断向 C_k 添加节点和更新 addCandidates 直到 addCandidates 为空,第 k 只蚂蚁构造团 C_k 的过程结束,输出最终的 C_k .团结构 $C_1,C_2,\cdots,C_k,\cdots,C_n$ 中节点个数最大的团是本次迭代中的最优解,称为最优团.最后,根据本次迭代得到的最优团进行信息素更新(2.3 节),直到达到最大迭代次数时算法终止.

2.2　蚂蚁选点策略

蚂蚁选点策略,也称蚂蚁转移策略.蚂蚁的搜索过程也就是不断向当前团中添加节点的过程,蚂蚁根据候选节点的概率从加点候选集 addCandidates 中选择候选节点 v_i ,候选节点的概率公式为

$$p(v_i)=\frac{[\tau_c(v_i)]^{\alpha}}{\sum\limits_{v_j\in \text{addCandidates}}[\tau_c(v_j)]^{\alpha}} \tag{1}$$

式中,$p(v_i)$ 表示候选节点 v_i 被选择的概率;α 表示信息素权重因子;$\tau_c(v_i)$ 表示节点 v_i 与当前解 C 中每个节点相连的边上的信息素之和,即 $\tau_c(v_i)=\sum\limits_{v_j\in C}\tau_{ij}$,τ_{ij} 表示边 (v_i,v_j) 上的信息素.

2.3　信息素更新

蚁群算法求解最大团时,信息素可以释放在节点上或者边上,与信息素在节点上相比,信息素在边上能突出候选节点与团的连接关系,提高蚂蚁判断的准确性,因此本文选择将信息素释放在边上.每一次迭代后,所有蚂蚁完成团构造,首先对图中每个边上的信息素进行衰减,再根据本次迭代的最优团对图中相应边上的信息素进行增加,引导后续蚂蚁选择较好的点,向好的方向转移.信息素更新公式为

$$\tau_{ij}(t+1)=(1-\rho)\times\tau_{ij}(t)+V\tau_{ij} \tag{2}$$

$$V\tau_{ij}=\begin{cases}1/(1+|C_{\text{best}}|-|C_{\text{new}}|),(v_i,v_j)\in C_{\text{new}}\\0,\text{其他}\end{cases} \tag{3}$$

式中,t 表示迭代次数,$\rho(0<\rho<1)$ 表示信息素挥发率,$|C_{\text{best}}|$ 表示历次迭代得到的最优团的大小,$|C_{\text{new}}|$ 表示本次迭代得到的最优团的大小.

3　多策略蚁群算法求解最大团

本文提出的多策略蚁群算法主要从两方面进行改进:①根据分条件选点策略进行蚂蚁转移,主要目的是增强蚂蚁对解空间的搜索能力,提高解的多样性,避免过早陷入局部最优;②采用局部改善策略,提高算法的求解精度和收敛速度,解决由于多样性引起的收敛速度变慢的问题.

3.1　分条件选点策略

基本蚁群算法在蚂蚁转移策略中选择信息素强的节点来构造团,因此被选中的节点的信息素越来越强,节点被选择的概率也越来越大,导致算法后期容易陷入局部最优.针对这个问题,本文提出一种新的分条件选点策略,灵活交替应用轮盘赌选点策略和补充选点策略.轮盘赌选点策略的出发点是信息素越强的节点被选择的概率越大,从而保证求解过程的收敛.补充选点策略增加在可行解中出现次数较少节点被选择的概率来扩大搜索空间、增加解的多样性.

补充选点策略中,首先,建立节点历史次数表(HT)以记录每个节点被选中的次数,称为 HT 数;然后,在加点候选集中选择 HT 数最小的节点作为蚂蚁下一个选择的节点(如果加点候选集中 HT 数最小的节点不止一个,就随机选中一个).这样,增加了出现次数较少的节点被选择的概率,提高了蚂蚁选择的多样性,避免陷入局部最优解.通过正整数 q_0 和 q 来控制分条件选点策略中轮盘赌选择策略和补充选点策略的执行次数.每次的迭代中执行 q 次补充选点策略,执行 q_0-q 次轮盘赌选点策略。t 表示当前迭代次数,两个策略采用以下方法控制:

(Ⅰ)当 $0<t\%q_0<(q_0-q)$ 时,采用轮盘赌选点策略进行选点.

(Ⅱ)当 $t\%q_0\geqslant(q_0-q)$ 时,采用补充选点策略,按照节点 HT,对每个候选节点的 HT 数进行排序,选择 HT 数最小的节点作为蚂蚁选择的下一个节点.

当所有蚂蚁均得到团之后,对每一个团进行局

部改善操作(3.2 节),最后更新节点 HT.

3.2 局部改善策略

分条件选点策略增加了解的多样性,但是降低了算法的收敛速度.为了克服这个缺点,本文提出了局部改善策略,即在已构造的团周围进一步搜索得到更优的团结构.通过引入局部改善策略提高算法的收敛速度,同时也能提高算法的求解精度.

局部改善策略具体为:选择团外度数较大且相连的两个节点 (v_1,v_2) $((v_i,v_j)\in V$ 且 $(v_i,v_j)\in E)$.若这两个节点和团中除 v_3 之外的所有节点均相连,则用 (v_1,v_2) 替换 v_3 从而得到新的团结构;然后从团外节点中选择与新团相连的所有节点添加到团结构中,使团得到进一步扩展.局部改善策略应用在每只蚂蚁利用选点策略和可行解构造方法构造出团之后,信息素更新之前进行.具体过程如下:

步骤 1 选择团 C 中的节点 v_i ($i=1,2,3,\cdots,m$, m 表示团 C 的大小),针对 v_i 建立换点候选集 exchangeCandidates,exchangeCandidates 中的节点满足条件:$\forall u_i$ ($u_i\in$ exchangeCandidates) 与 v_i 不相连,但与 $C-\{v_i\}$ 中每一个节点均相连.

步骤 2 选择换点候选集 exchangeCandidates 中所有节点构成的子图中度数最大的且相连的两个节点 u_j,u_k ,用 u_j,u_k 置换节点 v_i ,更新团 C.

步骤 3 调整 exchangeCandidates,删除与 u_j 和 u_k 不相连的节点,得到加点候选集 addCandidates.

步骤 4 选择 addCandidates 构成的子图中度数最大的节点 v_j 加入团 C.调整 addCandidates,删除与 v_j 不相连的节点.

重复步骤 4 直到 addCandidates 为空,得到局部改善完成的团 C.

3.3 多策略蚁群算法过程

增加分条件选点策略和局部改善策略的蚁群算法的主要步骤如下:

输入:最大迭代次数 cycle,蚂蚁个数 n ,信息素权重因子 α 、信息素挥发系数 ρ.

输出:最大团 C_{best}.

步骤 1 参数初始化.节点历史次数表 HT $=\{0\}$ 和最大团 $|C_{\text{best}}|=\varnothing$,当前迭代 $t=0$;

步骤 2 按照如下步骤依次构造团 $C_1,C_2,\cdots,C_k,\cdots,C_n$, C_k 为第 k 只蚂蚁对应的团,每只蚂蚁构造团的过程相同,下面用 C 表示.

(a)随机选择一个节点 $v_i\in V$.作为团的第一个节点, $C\leftarrow\{v_i\}$.

(b)构造节点 v_i 的加点候选集, addCandidates $\leftarrow\{v_j/(v_j,v_i)\in E\}$.

(c)加点候选集 addCandidates 不为空时,若 $0<t\%q_0<(q_0-q)$,根据概率公式(1)计算每一个候选节点的概率,根据轮盘赌策略选择下一个节点 v_k , $C\leftarrow C\cup\{v_k\}$;若 $t\%q_0\geqslant(q_0-q)$,根据补充选点策略选择的下一个节点 v_k , $C\leftarrow C\cup\{v_k\}$.

(d)调整加点候选集, addCandidates $\leftarrow$ addCandidates $\cap\{v_j/(v_j,v_i)\in E\}$.

(e)若加点候选集 addCandidates 不为空则返回步骤(c);否则执行步骤(f).

(f)得到团 C .

(g)对团 C 采用局部改善策略,得到改善后的团 C .

步骤 3 根据所有蚂蚁得到的 $C_1,C_2,\cdots,C_k,\cdots,C_n$ 更新节点 HT.

步骤 4 选取 $C_1,C_2,\cdots,C_k,\cdots,C_n$ 中最大的团为本此迭代的最优 $|C_{\text{new}}|$. 如果 $|C_{\text{best}}|<|C_{\text{new}}|$,则 $|C_{\text{best}}|=|C_{\text{new}}|$,否则 C_{best} 不变.根据信息素的更新规则和公式(2)、(3)对信息素进行更新.

步骤 5 判断 t 是否达到设置的最大迭代次数 cycle.若是则输出最大团 C_{best} ;否则, $t=t+1$,返回到步骤 2.

4 基于 Spark 的多策略蚁群算法的实现

4.1 Spark

Spark 是基于内存的分布式并行计算平台,它拥有 Hadoop 平台和 MapReduce 框架的全部优点,不同的是 Spark 运算的中间结果能存储在内存中,而不再需要读写 HDFS,提高了并行计算的速度,因此 Spark 更适合进行数据挖掘与机器学习等需要迭代处理算法的实现[23].Spark 集群启动时包括一个 Master 节点和若干个 Worker 节点,其中 Master 节点主要负责集群资源的管理,Worker 节点主要负责数据的计算.Spark 的工作流程如图 1 所示,当在 Master 节点使用 Spark-submit 命令提交作业时,首先在本地客户端启动一个 Driver 进程;Driver 进程会根据设置的参数向 Master 节点申请相应的集群资源,主要有 Worker 节点个数、每个 Worker 节点

上 Executor 的内存和 CPU 数量；Master 节点与 Worker 节点进行通信，通知 Worker 节点启动 Executor 并向 Driver 进程注册；Driver 进程与 Worker 节点连接起来，将需要执行的任务分配给集群中的各 Worker 节点，Worker 节点按照任务分配从读取数据并计算，Driver 进程对各个 Worker 节点处理完的结果进行收集和汇总.

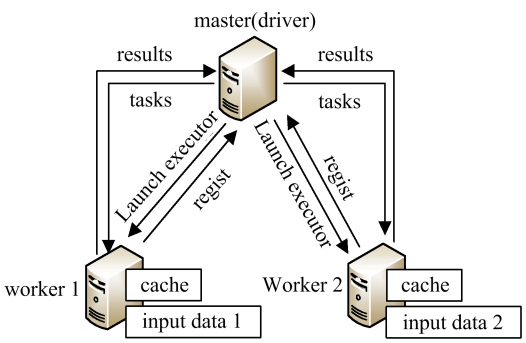


图 1　Spark 工作流程图

Fig. 1　Spark workflow diagram

4.2　多策略蚁群算法的并行化实现流程

多策略蚁群算法求解最大团需要反复迭代求解，每次迭代求解后都要对全局的信息素进行更新. 基于 Spark 的多策略蚁群算法并行化实现流程如图 2 所示，图 2 中的每个 RDD 分片都代表一只蚂蚁使用多策略方法求解最大团的过程. 蚁群并行求解完之后找出本次迭代的最大团，用于全局信息素的更新，然后进行下一次迭代找最大团，直到迭代次数满足终止条件.

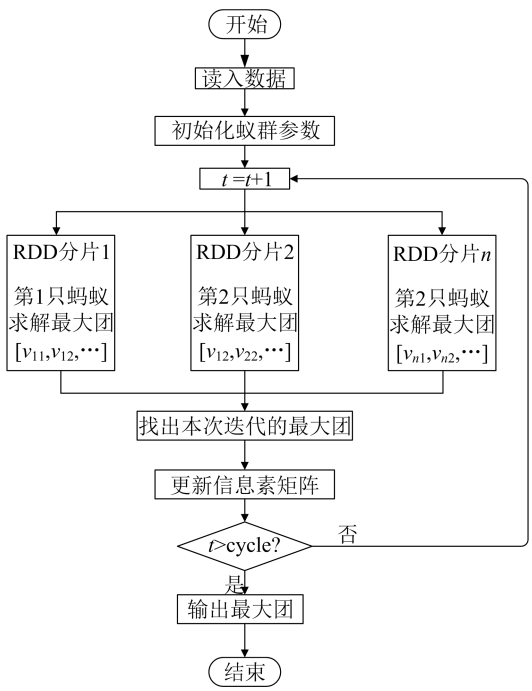


图 2　蚁群算法并行化实现流程图

Fig. 2　The parallel training algorithm of ACO

Spark 平台的优势在于基于内存的计算，RDD (resilient distributed dataset)是在 Spark 平台中对分布式内存的一种抽象，主要是由 HDFS 上的分布式文件创建得出或是从其他 RDD 转换而来[24]. RDD 可被分为多个分区并分布在集群中的不同节点上，从而让 RDD 中的数据可以被并行地操作. 蚁群算法并行化实现所使用的 RDD 算子和数据状态的并行化转换过程如图 3 所示. 具体的并行转换过程如下：

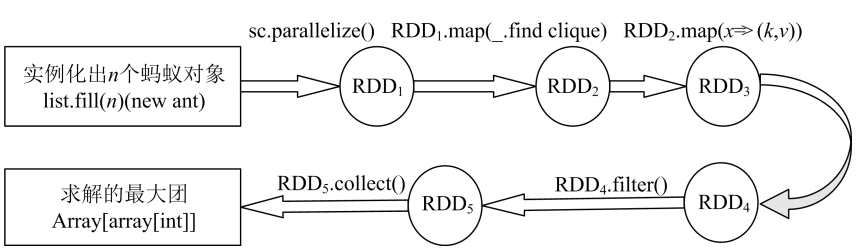


图 3　数据状态并行化转换图

Fig. 3　Data state parallel conversion diagram

Step1 首先定义蚂蚁求解最大团的类 ant，针对 n 只蚂蚁把该类实例化出 n 个对象，存放在列表 List(n)(new ant)中；

Step2 使用 parallelize 算子将列表 List(n)(new ant)创建为 RDD，记为 RDD_1；

Step3 使用 map 算子，调用类 ant 里的 findClique 方法实现多策略蚁群算法求解最大团，结果以 Array[Int]形式返回，返回值记为 RDD_2；

Step4 使用 map 算子，将 RDD_2 转换为(k，v)的形式，记为 RDD_3，其中 k 为每个解中的节点个数，v 为每只蚂蚁求得的解；

Step5 使用 sortByKey 算子对 RDD_3 进行降序排序，排序结果记为 RDD_4；

Step6 使用 filter 算子从 RDD_4 里过滤出排序

靠前的一个或多个的解，记为 RDD_5；

Step7 使用 collect 算子将 RDD_5 以二维数值的形式返回，即为本次并行求解的最大团.

5 实验结果分析

5.1 实验环境

本文实验在 Spark2.0.0 和 Hadoop2.4.1 集群上实现，集群环境包含 6 个节点，其中 1 个 Master 节点，5 个 Worker 节点，节点处在同一个局域网内，操作系统为 CentOs6.5，CPU 为 E5-2620 v4，核心频率 2.10 GHz，Master 节点内存为 96 GB，Worker 节点内存 32 GB。本文使用美国离散数学及理论计算机科学中心提供的 DIMACS 基准图例数据集作为实验数据，验证该算法的有效性和并行性。

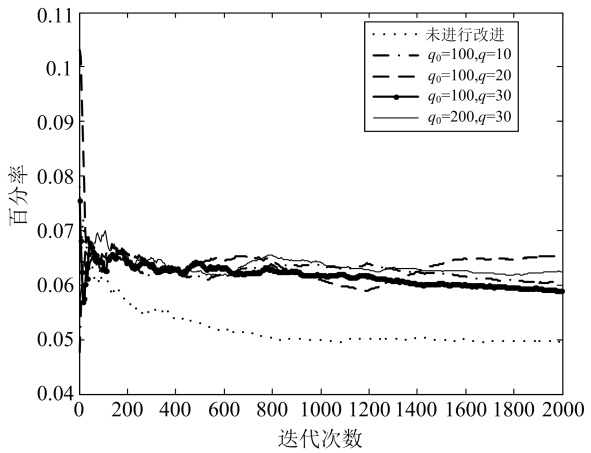
5.2 多条件选点策略的有效性验证

为了验证多条件选点策略的有效性，采用 4 组不同的 q_0 和 q 值在 DIMACS 基准数据集的 3 个典型图例 Brock200-2、C250-9 和 DSJC500-5 上进行实验，每次实验重复进行 50 次，取其平均值，实验结果如图 4 所示.

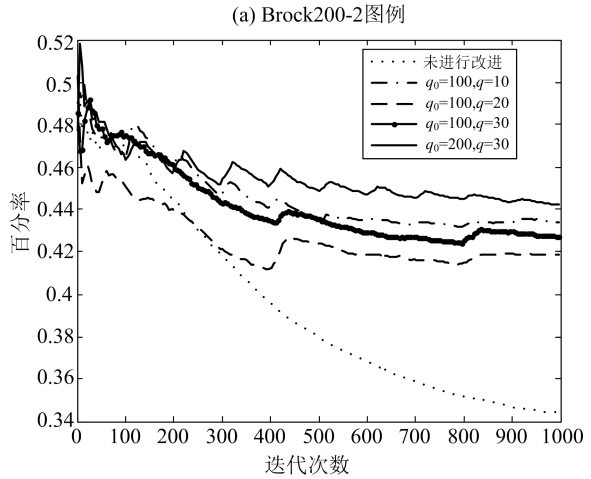
图 4 中，横坐标是迭代次数，纵坐标是本次迭代中最大团中节点(已知最大团中的节点)累计出现在团(本次迭代得到的所有团)中次数占图中所有节点累计出现在团中次数的百分率，最大团中的节点出现的百分率越大表明正确节点被搜索到的概率越大，最终得到最大团的概率也就越大. 图 4 五条曲线中，标记为'未进行改进'的曲线表示未使用分条件选点策略的蚁群算法的实验结果，其他 4 条曲线分别表示在不同参数配置下的实验结果. 它们的参数分别是 $q_0=100, q=10$、$q_0=100, q=20$、$q_0=100, q=30$ 和 $q_0=200, q=30$. 图 4 表明，随着迭代次数的增加，最大团中节点被选择的概率逐渐减少，未使用分条件选点策略的蚁群算法的曲线尤为明显，因为在迭代后期整个算法趋于收敛，团中节点的多样性降低，往往陷入局部最优. 使用分条件选点策略的蚁群算法结果明显更好，尤其在迭代后期，百分比远远高于未进行改进的蚁群算法的结果，在保持多样性较好的迭代过程中更有利于得到最优解，避免陷入局部最优，因此采用分条件选点策略更有利于得到最大团. 实验结果也验证了合理的 q 与 q_0 的比例应该在 1∶10 与 1∶5 之间.

5.3 局部改善策略的有效性验证

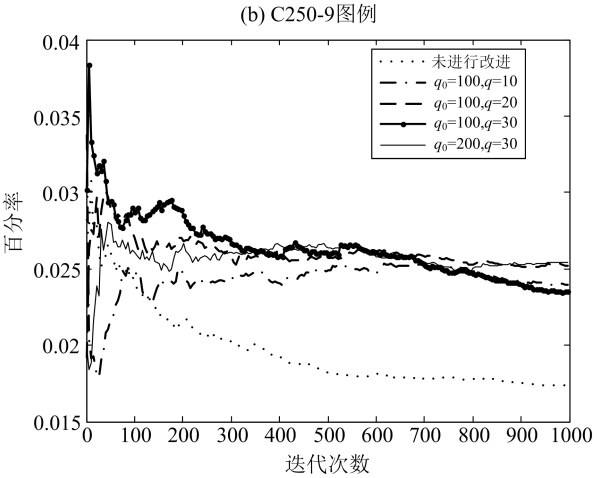
为了验证局部改善策略，在 DIMACS 基准数据集中的 C250-9，C500-9 和 Brock800-4 图例上进行测试. 相应参数为 $q_0=100$，$q=20$，$n=30$，$\alpha=2$，$\rho=0.005$. 每次实验重复进行 50 次，取其平均值，结果如图 5 所示. 横坐标表示迭代次数，纵坐标表示本次迭代中，通过局部改善策略后得到改善的蚂蚁的数量(每个蚂蚁对应一个团结构，团的大小有所增加



(a) Brock200-2图例



(b) C250-9图例

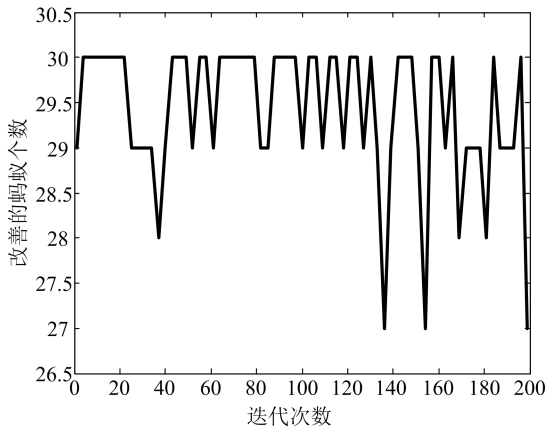


(c) DSJC500-5图例

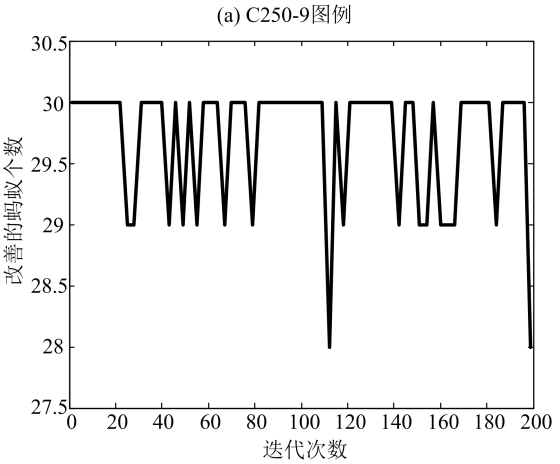
图 4 最大团中节点出现次数的百分比

Fig. 4 Percentage of occurrences of nodes in the largest group

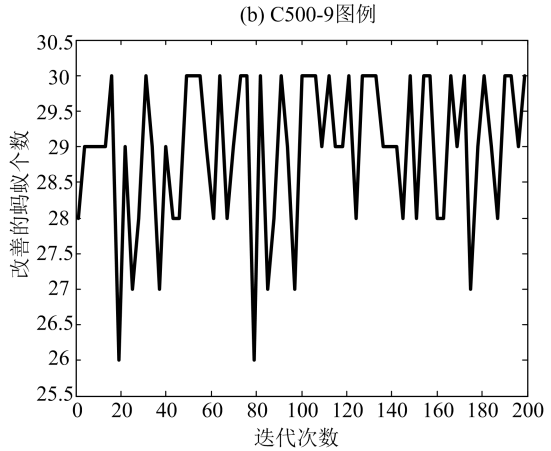
表明该蚂蚁得到改善). 蚂蚁总个数为 30,图 5(a),(b)和(c)均有 28 到 30 个蚂蚁得到改善,即每次迭代均有超过 96.7%的蚂蚁得到改善,说明局部改善的作用明显.



(a) C250-9图例



(b) C500-9图例

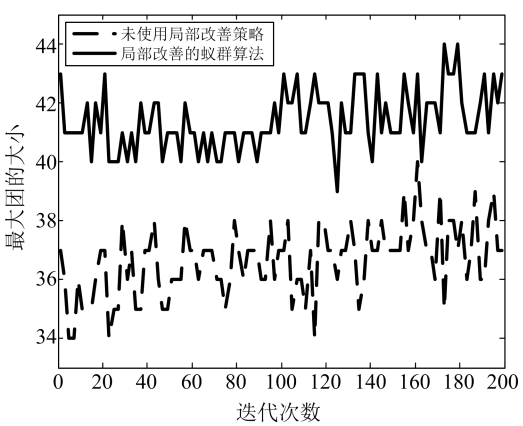


(c) Brock800-4图例

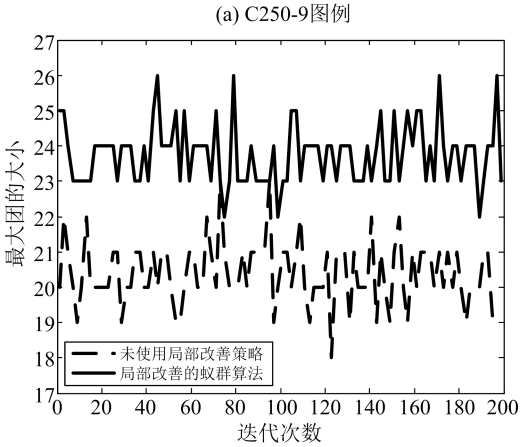
图 5　得到改善的蚂蚁数量

Fig. 5　The number of improved ants

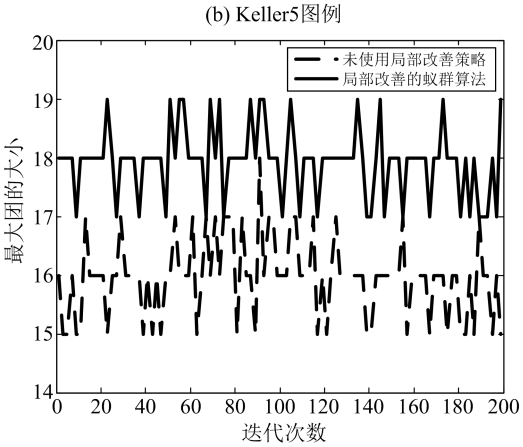
为了进一步展示局部改善的效果,我们在 C250-9、Keller5 和 Brock 800-4 图例上将未使用局部改善策略的蚁群算法和增加局部改善策略后的蚁群算法进行了对比,每次实验重复进行 50 次,取其平均值,实验结果如图 6 所示. 横坐标表示迭代次数,纵坐标表示每次迭代得到的局部最优团的大小. 虚线表示未使用局部改善策略的蚁群算法得到的结果,实线表示使用局部改善后得到的结果. 从图 6 可以明显看到,在整个算法迭代运行期间实线均高于虚线,进一步说明增加局部改善后算法的求解精度得到了提高,更易收敛到全局最优,减少了迭代次数,从而提高了收敛速度.



(a) C250-9图例



(b) Keller5图例



(c) Brock 800-4图例

图 6　最大团的大小

Fig. 6　The size of maximum cliques

5.4 算法的求解精度比较

与目前求解精度最高的 Serge Fenet 的改进方法 Edge-AC+LS 进行比较以验证多策略蚁群算法(multi-strategy)的求解精度. 为保证算法具有可比性,本文采用的运行次数、最大迭代次数、蚂蚁个数、α 、ρ 以及信息素的范围与 Edge-AC+LS 的参数设置相同,参数设置见表 1. 表 2 给出了算法 Edge-AC+LS 和多策略蚁群算法在 DIMACS 基准图例上分别运行 50 次求得的最优解(Max)和平均解(Avg),BR 表示目前已知最大团的大小,其中算法 Edge-AC+LS 的结果来源于文献[13]. 为保持一致,平均解都保存到 0.1. 相比 Edge-AC+LS,在最优解 Max 上有 3 个图例得到改善;在平均解上有 12 个图例得到改善. 多策略蚁群算法在 32 个图例上的最优解均能得到已知最大团,多策略蚁群算法在 32 个图例当中有 24 个图例最优解和平均解均能得到已知最大团,其中对 Brock400_2 求得了个数为 29 的已知最大团,是目前蚁群算法中求解最好的结果. 结果表明,通过分条件选点策略和局部改善策略,提高了蚂蚁搜索空间的能力、增加了算法的多样性,很大程度地提高了算法的求解精度.

表 1 参数设置

Tab. 1 Parameter settings

times	cycles	n	q_0	q	α	ρ	τ_{min}	τ_{max}
50	5 000	30	100	20	2	0.005	0.01	6

表 2 32 个基准图例上的实验结果

Tab. 2 Test results on 32 benchmarks

Graph	Edge-AC+LS		multi-strategy		BR
	Max	Avg	Max	Avg	
C125-9	34	34.0	34	34.0	34
C250-9	44	44.0	44	44.0	44
C500-9	57	55.9	57	55.9	57
C1000-9	68	66.2	68	66.3	68
C2000-5	16	15.3	16	16.0	16
DSJC500-5	13	13.0	13	13.0	13
DSJC1000-5	15	14.3	15	15.0	15
Brock200_2	12	12.0	12	12.0	12
Brock200_4	17	16.8	17	17.0	17
Brock400_2	25	24.8	29	28.8	29
Brock400_4	33	27.1	33	33.0	33

续表 2

Graph	Edge-AC+LS		multi-strategy		BR
	Max	Avg	Max	Avg	
Brock800_2	24	20.1	24	21.6	24
Brock800_4	26	20.0	26	23.3	26
Gen200_p0.9_44	44	44.0	44	44.0	44
Gen200_p0.9_55	55	55.0	55	55.0	55
Gen400_p0.9_55	53	52.2	55	53.1	55
Gen400_p0.9_65	65	65.0	65	65.0	65
Gen400_p0.9_75	75	75.0	75	75.0	75
Hamming8_4	16	16.0	16	16.0	16
Hamming10_4	40	39.3	40	40.0	40
Keller4	11	11.0	11	11.0	11
Keller5	27	27.0	27	27.0	27
Keller6	57	55.1	59	57.6	59
p_hat300_1	8	8.0	8	8.0	8
p_hat300_2	25	25.0	25	25.0	25
p_hat300_3	36	36.0	36	36.0	36
p_hat700_1	11	11.0	11	11.0	11
p_hat700_2	44	44.0	44	44.0	44
p_hat700_3	62	62.0	62	62.0	62
p_hat1500_1	12	11.1	12	11.9	12
p_hat1500_2	65	65.0	65	65.0	65
p_hat1500_3	94	94.0	94	94.0	94

5.5 算法并行性的验证

该实验选取了个 DIMACS 基准数据集中最大团节点个数最多的 3 个图例作为实验数据,实验图例的描述见表 3.

表 3 实验图例的描述

Tab. 3 Description of experiment instance

	graph	nodes	edges	clique
I	MANN_a81	3 321	5 506 380	1 100
II	MANN_a45	1 035	533 115	345
III	MANN_a27	378	70 551	126

实验通过使用不同的图例,统计 Spark 平台在不同节点数目下迭代训练所需的时间,计算加速比来验证算法的并行性。加速比的计算如公式为

$$S_p = \frac{T_1}{T_p} \tag{4}$$

式中，T_1 表示使用 1 个节点时任务执行的时间，T_p 表示使用 P 个节点时任务执行的时间，S_p 代表加速比，表示在并行化之后的效率提升情况。当 $S_p = P$ 时为线性加速比，越接近线性加速比，说明算法的并行性越好。

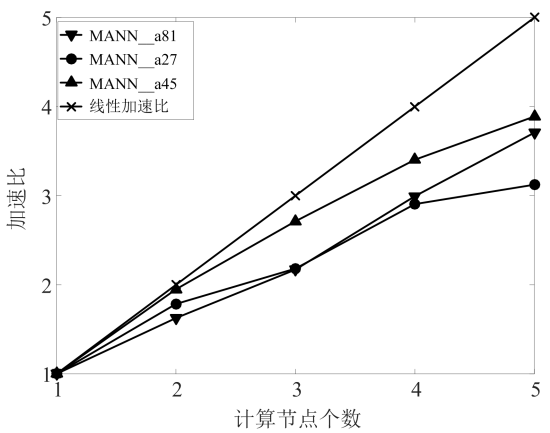


图 7　加速比示意图

Fig. 7　Speedup

实验结果如图 7 所示，其中横坐标为计算节点个数，纵坐标为加速比值。图 7 四条线分别为线性加速比、MANN_a45 图例的加速比、MANN_a81 图例的加速比以及 MANN_a27 图例的加速比。从图 7 可知，3 个图例的加速比值与计算节点数目都近似呈线性关系，其中节点数较多的两个图例的结果更为明显。这表明基于 Spark 的多策略蚁群算法在求解最大团问题时有较好的并行性。

6　结论

本文提出了一种基于 Spark 的多策略蚁群算法求解最大团的问题. 算法的改进主要包括 3 个方面：①提出一种分条件选点策略，蚂蚁采用补充选点策略和轮盘赌策略选择节点，这样可以提高蚂蚁对解空间的搜索能力，增加解的多样性，避免陷入局部最优；②引入局部改善策略，提高算法的求解精度和收敛速度. ③将改进后的算法部署到分布式平台 Spark 上并行实现，验证了算法的并行性，提高了算法求解大规模社区网络的执行效率.

实验结果证明，改进后的蚁群算法对 DIMACS 标准图例的所有图例都有效，在提高求解精度的同时也加快了收敛速度，其中对 Brock400_2 求得了个数为 29 的已知最大团，这是目前蚂蚁算法求解最好的结果，在 Spark 分布式平台上实现了该算法的并行化，验证了该算法在处理大规模社区网络的情况下有较好的并行性.

参考文献(References)

[1] 邵娜娜. 蚁群算法求解最大团问题研究与应用[D]. 河北工业大学，2015.
SHAO Nana. Research and application of ant colony algorithm for the maximum clique problem[D]. Hebei university of technology, 2015.

[2] WU D, SCHAEFER D, ROSEN D W. Cloud-based design and manufacturing systems: A social network analysis[C]// International Conference on Engineering Design. Seoul, Korea: ACM, 2013: 1-10.

[3] FADIGAS I S, PEREIRA H B B. A network approach based on cliques[J]. Physica A: Statistical Mechanics & Its Applications, 2013, 392(10): 2576-2587.

[4] ARULSELVAN A, BAOURAKIS G, BOGINSKI V, et al. Analysis of food industry market using network approaches[J]. British Food Journal, 2013, 110(9): 916-928.

[5] WANG P, LIN H T, WANG T S. An improved ant colony system algorithm for solving the IP traceback problem[J]. Information Sciences, 2016, 326(1): 172-187.

[6] XU B, MIN H. Solving minimum constraint removal (MCR) problem using a social-force-model-based ant colony algorithm[J]. Applied Soft Computing, 2016, 43(6): 553-560.

[7] 徐永成，陈崚. 基于蚁群优化的二分网络社区挖掘[J]. 计算机科学与探索，2014，8(3)：296-304.
XU Yongcheng, CHEN Ling. Community detection on bipartite networks based on ant colony optimization [J]. Journal of Frontiers of Computer Science & Technology, 2014, 8(3): 296-304.

[8] WU Q, HAO J K. A review on algorithms for maximum clique problems[J]. European Journal of Operational Research, 2015, 242(3): 693-709.

[9] 支志兵，宁爱兵，陈吉珍，等. 最大团问题的加权分治算法[J]. 计算机工程与应用，2016，52(2)：50-53.
ZHI Zhibing, NING Aibing, CHEN Jizhen, et al. Measure and conquer approach for maximum clique problem[J]. Computer Engineering and Applications, 2016, 52(2): 50-53.

[10] ARI A A A, DAMAKOA I, TITOUNA C, et al. Efficient and scalable ACO-based task scheduling for green cloud computing environment [C]// IEEE International Conference on Smart Cloud. New York, USA: IEEE, 2017: 66-71.

[11] SITARZ P, POWAŁKA B. Modal parameters estimation using ant colony optimisation algorithm[J].

Mechanical Systems and Signal Processing, 2016, 76(8): 531-554.

[12] 王辉,王景良,朱龙彪,等.基于改进蚁群算法的泊车系统路径规划[J]. 控制工程,2018, 25(2): 53-58.
WANG Hui, WANG Jingliang, ZHU Longbiao, et al. Path planning of parking system based on improved ant colony algorithm[J]. Control Engineering of China, 2018, 25(2): 53-58.

[13] FENET S, SOLNON C. Searching for maximum cliques with ant colony optimization[J]. Applications of Evolutionary Computing Proceedings of Evoworkshops Evocop Evoiasp Evostim, 2003, 2611(4): 236-245.

[14] SOLNON C, FENET S. A study of ACO capabilities for solving the maximum clique [J]. Journal of Heuristics, 2006, 12(3): 155-180.

[15] XU X, MA J, LEI J. An improved ant colony optimization for the maximum clique problem[C]// Third International Conference on Natural Computation ICNC. Haikou, China: IEEE, 2007: 766-770.

[16] 王会颖, 耿家礼. 基于蚁群算法求解最大团问题[J]. 计算机应用与软件, 2010, 27(10): 107-113.
WANG Huiying, GENG Jiali. Solving maximum clique problem on ant colony optimization [J]. Computer Application and Software, 2010, 27(10): 107-113.

[17] LIU Lei, WANG Shaoqiang. An improved ant colony optimization algorithm using local pheromone and global pheromone updating rule[C]// International Conference on Intelligent Transportation, Big Data & Smart City. Changsha, China: IEEE Computer Society, 2016: 63-67.

[18] SOLEIMANI-POURI M, REZVANIAN A, MEYBODI M R. Finding a maximum clique using ant colony optimization and particle swarm optimization in social networks [C]// Proceedings of the 2012 international conference on advances in social networks analysis and mining ASONAM. Washington, USA: IEEE Computer Society, 2012: 58-61.

[19] 夏卫雷, 王立松. 基于 MapReduce 的并行蚁群算法研究与实现[J]. 电子科技,2013, 26(2): 146-149.
XIA Weilei, WANG Lisong. Research on and Implementation of parallel ant colony algorithm based on MapReduce[J]. Electronic Science and Technology, 2013, 26(2): 146-149.

[20] 王诏远, 李天瑞, 易修文. 基于 MapReduce 的蚁群优化算法实现方法[J]. 计算机科学, 2014, 41(7): 261-265.
WANG Zhaoyuan, LI Tianrui, YI Xiuwen. Approach for development of ant colony optimization based on MapReduce[J]. Computer Science, 2014, 41(7): 261-265.

[21] 王诏远, 王宏杰, 邢焕来, 等. 基于 Spark 的蚁群优化算法 [J]. 计算机应用, 2015, 35(10): 2777-2780, 2797.
WANG Zhaoyuan, WANG Hongjie, XING Huanlai, et al. Ant colony optimization algorithm based on Spark[J]. Journal of Computer Applications, 2015, 35(10): 2777-2780, 2797.

[22] DONG G, FU X, LI H, et al. Cooperative ant colony-genetic algorithm based on Spark[J]. Computers & Electrical Engineering, 2017, 60(C): 66-75.

[23] LIU Tiantian, FANG Zhiyi, ZHAO Chen, et al. Parallelization of a series of extreme learning machine algorithms based on Spark[C]// 15th International Conference on Computer and Information Science. Okayama, Japan: IEEE, 2016: 1-5.

[24] JING W, HUO S, MIAO Q, et al. A model of parallel mosaicking for massive remote sensing images based on Spark[J]. IEEE Access, 2017, 5(99): 18229-18237.