计算机科学
COMPUTER SCIENCE

TAMP:面向区域覆盖的层次化多机器人任务分配方法

安浩嘉, 史殿习, 李林, 孙亦璇, 杨绍武, 陈旭灿

引用本文

安浩嘉, 史殿习, 李林, 孙亦璇, 杨绍武, 陈旭灿. TAMP:面向区域覆盖的层次化多机器人任务分配方法[J]. 计算机科学, 2023, 50(9): 269-277.

AN Haojia, SHI Dianxi, LI lin, SUN Yixuan, YANG Shaowu, CHEN Xucan. TAMP:A Hierarchical Multi-robot Task Assignment Method for Area Coverage [J]. Computer Science, 2023, 50(9): 269-277.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220800094

TAMP:面向区域覆盖的层次化多机器人任务分配方法

安浩嘉[1] 史殿习[1,2,3] 李 林[1] 孙亦璇[1] 杨绍武[1] 陈旭灿[1,2]
1 国防科技大学计算机学院 长沙 410073
2 军事科学院国防科技创新研究院 北京 100166
3 天津(滨海)人工智能创新中心 天津 300457
(ahj20@alumni.nudt.edu.cn)

摘 要 作为诸多移动机器人应用的基础,完全覆盖旨在为机器人规划出一条访问目标区域所有点且耗时最短的无碰撞路径。此类覆盖应用中,利用多台机器人协同覆盖可以有效缩短覆盖时间并提升系统的鲁棒性,同时也增加了算法设计复杂度和机器人协同管理难度。因此,文中研究了已知环境下的多机器人覆盖问题,该问题已被证明是一个NP难题。文中提出了一种启发式的基于多层次图划分的多机器人任务分配方法(Multi-robot Task Assignment Based on Multi-level Graph Partitioning,TAMP),该方法包含一种粗化任务分配算法和一种精细任务分配算法。粗化任务分配算法采用分层粗化的方法,通过图的极大匹配实现了节点融合以降低图的规模,并基于均匀种子的图增长方式获取了一个接近均衡的初始任务分配结果,提高算法效率;精细任务分配算法在粗化任务分配算法的基础上,提出了一种基于边界节点交换的Lazy&Lock策略,用于实现任务细分,提高求解精度。文中在不同规模的随机图和真实世界的治安巡逻场景下进行了仿真验证。仿真结果表明,相比经典的任务分配方法,TAMP方法将可求解的最大计算规模从千级扩大到百万级,小规模图(3000以内)的计算速度加快了20倍,距离最优解偏差均优于经典方法;能够在60s内解决大规模图(3000~1000000)的任务分配问题,同时将距离最优解偏差控制在0.3%以内。

关键词:多机器人系统;区域覆盖;任务分配;多层次图划分;最小最大平衡连通q分割
中图法分类号 TP391

TAMP:A Hierarchical Multi-robot Task Assignment Method for Area Coverage

AN Haojia[1],SHI Dianxi[1,2,3],LI lin[1],SUN Yixuan[1],YANG Shaowu[1] and CHEN Xucan[1,2]
1 School of Computer Science,National University of Defense Technology,Changsha 410073,China
2 National Innovation Institute of Defense Technology,Academy of Military Sciences,Beijing 100166,China
3 Tianjin Artificial Intelligence Innovation Center,Tianjin 300457,China

Abstract As the foundation of many mobile robot applications,complete coverage aims to plan a collision-free path for robot to visit all points in the target area quickly. Using multiple robots for cooperative coverage can significantly reduce coverage time and improve system robustness. However,it increases the algorithm's complexity and makes cooperative robot management more challenging. Therefore,the multi-robot coverage problem in a given environment is studied in this paper,which has been proven to be an NP problem. This work proposes a heuristic multi-robot task assignment based on multi-level graph partitioning(TAMP) method,which consists of a coarse task assignment algorithm and a fine task assignment algorithm. The coarse task assignment algorithm reduces the size of the graph by the strategies of multi-level coarsening and graph maximal matching and then obtains a roughly balanced task assignment by the graph growth strategy. The fine task assignment algorithm proposes a Lazy&Lock strategy to achieve task subdivision,which improves the solution accuracy. Simulations validate the performance of the TAMP approach under different scales of random graphs and real-world policing patrol scenarios. Compared to the conventional task assignment method,TAMP expands the maximum computational scale from thousands to millions. For small-scale graphs(within 3000),TAMP accelerates computation time by 20 times and outperforms the conventional method in terms of the deviation from the optimal solution for different small-scale graphs. For large-scale graphs(3000~1 million),TAMP can solve the task assignment problem in 60s while keeping the deviation of the optimal solution within 0.3%.

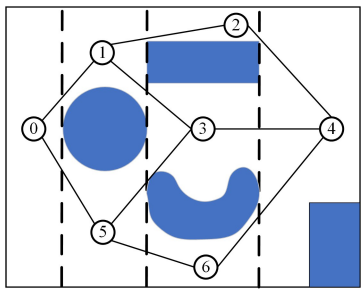
Keywords Multi-robot system,Area coverage,Task assignment,Multi-level graph partitioning,Min-max balanced connected q-partition problem

到稿日期:2022-08-10 返修日期:2022-12-03
基金项目:国家自然科学基金(91948303)
This work was supported by the National Natural Science Foundation of China(91948303).
通信作者:史殿习(dxshi@nudt.edu.cn)

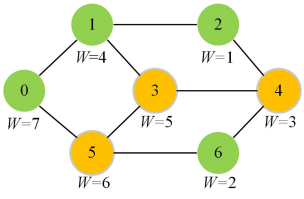
1 引言

完全覆盖问题(Complete Coverage Problem, CCP)是移动机器人领域的一个基础问题,其研究成果被广泛应用在地形绘制[1]、智能农业[2]、摄影测量[3]、抢险救灾[4]等场景中。CCP旨在规划一条覆盖目标区域所有点的路径,该路径可以在避免障碍物碰撞的同时,最小化覆盖时间[5]。当前CCP的相关工作大多聚焦于解决单机器人的覆盖问题,然而在如战区排雷、灾难救援、海底搜救的覆盖场景中,单台机器人受制于有限的能量载荷和计算资源,难以快速高效地完成覆盖。此时,利用多台机器人进行协同覆盖,可以有效提高区域覆盖任务的效率和鲁棒性。然而多机器人协同覆盖问题已被证明是一个NP难题[6],亟需设计一种简单高效的多机器人协同覆盖方法。

本文假定目标区域已知,且采用已知分解方法如Morse方法[5]完成分解。分解得到的多个子单元被抽象成一个连通图,其中连通图的顶点代表分解得到的一个子单元,其权重代表该单元的面积,连通图的边代表单元间的共同边界。图1(a)给出了一个区域分解后得到的连通图示例。一方面,为了避免出现冗余的覆盖,完全覆盖要求每个子区域只能分配给一台机器人。另一方面,为了避免机器人之间发生碰撞风险,完全覆盖还要求分配给单台机器人的任务之间是连通且非交叉的。连通性保证了覆盖的路径是实际可行的,同一个任务不被障碍物分割开来,而非交叉则保证了覆盖的路径是非重复和高效的。任务分配结束后,单个机器人分别调用已有的单机器人覆盖方法[6],规划出覆盖其任务的覆盖路径。图1(b)和图1(c)分别给出了连通任务和非连通任务的划分示例。

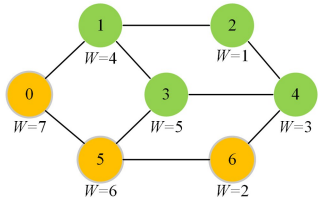


(a)环境分解图



$\omega_{max}=14$

(b)不连通任务分配示例



$\omega_{max}=15$

(c)连通任务分配示例

图1 区域覆盖任务分配示意图

Fig. 1 Diagram of area coverage task assignment

上述多机器人任务分配问题通常被建模成一个最小最大平衡连通q分割问题(Min-max Balanced Connected q-partition Problem, BCP_q),该问题已经被证明是一个NP难的问题[6]。BCP_q的相关工作可分为精确求解方法和近似求解方法两类。常见的精确求解方法有混合整数线性规划(Mixed Integer Linear Program, MILP)[7]、流模型方法[8-9]等。此类方法通常计算精度高,但是求解时间较长,因此仅适用于规模较小的覆盖场景。规模较大的场景通常采用启发式方法进行近似求解,此类近似求解方法可以分为基于树划分和基于图划分两个子类[8]。基于树划分的方法预先生成一棵树,然后利用遗传算法[10-11]或者构造生成树的方法[12]将树划分成若干棵均衡的子树,每棵子树对应一台机器人的覆盖任务。基于树划分的方法仅保留与树相关边的信息,因此求解速度较快;但是因为包含边的信息太少,此类方法通常对树很敏感,不同的树可能会产生差异很大的分配结果。基于图划分的方法需要考虑连通图中所有顶点和边的信息来进行划分,典型的基于图划分的方法有基于聚类的方法[13-14]和拍卖算法[6]等。此类方法通过制定启发式策略,大多考虑局部信息进行求解,通常会陷入局部最优,求解精度不高。本文中的大规模场景指多机器人区域覆盖应用中,目标区域抽象图顶点数量在3000到100万之间的分配任务;小规模场景指图顶点数量不大于3000的分配任务。

针对大规模场景的任务分配问题,本文提出了一种基于多层次图划分的多机器人任务分配方法(TAMP),该方法包含一种粗化任务分配算法和一种精细任务分配算法。粗化任务分配算法通过降低图的规模,来获取一个尽量均衡的初始任务分配结果;精细任务分配算法在粗化任务分配的基础上,对细化节点进行任务重分配,最终得到一个近似最优均衡的任务分配结果。本文在不同规模的随机网格图以及真实世界实例上进行了仿真实验验证。总的来说,本文的创新如下:

1)本文提出了一种启发式的粗化任务分配算法,该算法采用分层粗化的方式,基于图的极大匹配实现了对连通图的顶点融合,降低了图的规模,并基于均匀种子的图增长方式得到了一个尽量均衡的初始任务分配结果,从而降低了计算复杂度,提高了计算效率。

2)本文提出了一种启发式的精细任务分配算法,在基于粗化任务分配算法的基础上细化任务分配。提出了一个基于边界节点交换的Lazy&Lock策略,通过增益预排序、有效性验证等方式对细化节点进行重划分,实现了任务的近似最优均衡分配,提高了求解的精度。

3)本文在不同规模的随机图和真实世界的治安巡逻地图场景下进行了仿真验证。仿真结果表明,相比经典的任务分配方法,TAMP方法将可求解的最大任务规模从千级扩大到百万级,小规模图的计算速度加快了20倍,距离最优解偏差均优于经典方法;能够在60 s内解决大规模图的任务分配问题,同时将距离最优解偏差控制在0.3%以内。

本文第2章描述了机器人区域覆盖及任务分配的相关工作;第3章描述了多机器人区域覆盖任务分配的问题定义;第4章详细描述了TAMP方法的原理,包括粗化任务分配算法以及精细任务分配算法;第5章是关于仿真实验的验证;最后总结全文并展望未来。

2 相关工作

经典的区域覆盖工作包括环境表示、任务分配、路径规划

3 个步骤[15]，本文主要关注区域覆盖中任务分配这一重要环节。根据环境表示方法，区域分解方法可以分为精确分解和近似分解。精确分解能够实现对环境的完整表示[5]，经典的精确分解方法有 Morse 分解、布兰切特分解[6]等。近似分解将环境用统一的网格表示，覆盖的完整性依赖于网格的分辨率，典型的近似分解有波前算法[16]、STC 算法[17]等，其特点是易于表示和计算，但存在扩展性不足的问题，当目标区域变大时，网格数量急剧增加会导致算法的计算成本增高。

多机器人任务分配问题已被证明是一个 NP 难题[6]。该问题通常可以建模成一个BCP_q问题，旨在满足分区内部节点连通的条件下，使得各分区权重均衡。当输入图为树时，BCP_q问题可以看作是将树划分为 q-树森林，同时最小最大化该 q-树森林权重的问题，这种问题能够在 $O(|V|)$时间内解决[18]。然而，在现实中，很多BCP_q问题并不能被简化成一个森林的 q-树划分问题。BCP_q相关工作可分为精确求解方法和近似求解方法两类。

关于BCP_q问题的精确求解算法，文献[7]针对BCP_2问题提出了一种利用 MILP 进行精确求解的算法，该算法能够在 2 h 内解决 70 个顶点的 2-划分问题；文献[8]通过构建一个网络流量模型，首次提出针对 $q>2$ 的 MilpFlow 算法，在 $q=2$ 的情况下，MilpFlow 能够解决的顶点规模达到 170，在 $q\leqslant 5$ 的情况下，其能够解决的顶点规模达到 70；文献[9]同样基于流提出了具有多项式数量的变量和约束的精确算法，相比 MilpFlow 算法，其将BCP_2问题的规模增加了 400 倍，完成任务的速度快了 5 倍。

关于BCP_q问题的近似求解算法，Chen 等[19]提出了分别针对BCP_2和BCP_3问题的 5/4 近似和 3/2 近似算法；Chen 等[20]将问题推广至 $q\geqslant 3$ 的情况下，BCP_q问题存在 $q/2$ 近似比的算法，并特别地针对 $q=4$ 作出了局部改进和优化，提出并证明了一种针对BCP_4问题的近似比为 24/13 的算法。除了这些近似求解算法，许多启发式算法被提出并取得了不错的实际效果。文献[10]使用遗传算法解决了节点规模不超过 300 的BCP_2问题。Zhou 等[8]利用前驱数组和邻接表对树进行遗传算子编码，提出了基于树分割和树进化的 STED 算法，该算法解决了以多机协同区域覆盖问题为应用背景的任务分配问题，能对节点规模不大于 3 000 个节点的BCP_q问题实现 12-划分。文献[11]提出了一种带有时间限制的最优机器人数目任务分配算法 Mofint(Multi-objective GA with forest individual containing non-intersecting trees)。该算法通过代数法和生成树方法，估计出机器人数目的上下界，之后将问题转化为给定机器人数目的单目标优化问题，再利用遗传算法进行求解。在机器人数目固定时，Mofint 算法将已存在的最优近似比从 1.5～2 倍优化到 1.1～1.5 倍，在机器人数目不固定时，能够在任务分配时得到更少的机器人数目，Mofint 解决的问题规模能够达到 400 个。

3 问题定义

本文假定目标覆盖区域边界和障碍物已知，利用已知分解方法将覆盖区域抽象为 n 个顶点的权重图，每个顶点表示一个子任务，顶点权重表示划分区域的面积。有 q 个具有无线通信能力的同构机器人，通信条件以及机器人的续航能力理想。多机任务分配问题的目标就是将这 n 个区域分配给 q 个机器人以完成覆盖，并且最小化覆盖时间。

多机任务分配问题可以建模为一个最小最大平衡连通 q 分割(BCP_q)问题，该问题的形式化定义如下：设 $G=(V,E,W)$为 n 个顶点的连通图，V 为顶点集合，E 为边集合，权重函数 $W:V\mapsto\mathbb{N}^+$，$q\geqslant 2$ 且为正整数。目标是在满足一定约束的条件下，寻找图 G 的 q 个顶点分割 $\{V_i\}_{i=0}^{q-1}$，使得式(1)得到最优解：

$$\text{Target min max } \omega(V_i), i\in\{0,\cdots,q-1\} \tag{1}$$

$$\text{s.t. } V=V_0\cup V_1\cdots\cup V_{q-1} \tag{2}$$

$$V_i\cap V_j=\varnothing,\forall i,j\in\{0,\cdots,q-1\},i\neq j \tag{3}$$

$$V_i \text{ is connected} \tag{4}$$

其中，式(1)表示要优化的全局目标函数，$\omega(V_i)$表示V_i中所有顶点权重的总和；式(2)保证了覆盖的完整性；式(3)表示任务分配的非交叉性，进而保证了覆盖的路径是非重复和高效的；式(4)表示划分得到的任务需要满足连通性，从而保证了覆盖的路径是实际可行的。

4 TAMP 方法

针对大规模场景下的多机协同区域覆盖任务分配问题，本文提出了一种启发式的多机器人任务分配方法——TAMP，此方法包含粗化任务分配和精细任务分配两种子算法。TAMP 方法的框架图如图 2 所示。该方法的输入为目标区域经过分解后得到的抽象节点权重图。粗化任务分配方法通过分层粗化和初始划分得到初始任务划分结果；精细任务分配算法将该结果作为输入，通过增益排序、节点移动、更新细化图等对任务进行细化分配，最终得到多个机器人覆盖目标区域的均衡任务分配结果。

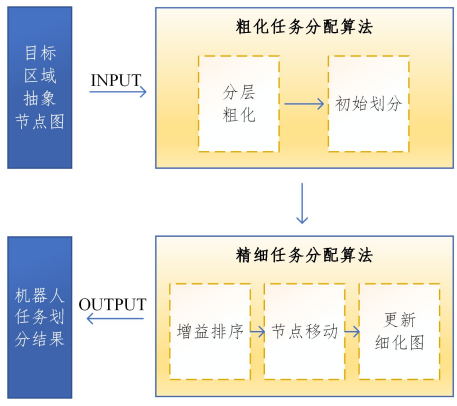


图 2 TAMP 方法的框架图

Fig. 2 Framework of TAMP approach

4.1 粗化任务分配算法

多机器人任务分配问题已被证明是一个 NP 难题[6]，该问题的复杂度取决于边和顶点的数量。经典的基于图划分的多机器人任务分配方法不对顶点和边进行处理，在规模较大的场景下直接计算会导致计算时间较长。而基于树划分的方法会对图进行剪枝，通过启发式的方法简化了一部分边。由于树划分舍弃了大量边的信息，因此影响了后续的求解精度。粗化任务分配算法受多层次图划分算法启发，通过压缩顶点

和边把暂时不关注的顶点和边隐藏，从而获得一个简化的图。与基于树划分的方法相比，粗化任务分配算法并不会抛弃边，而是生成一个能够反映原始图顶点和边特征的粗化图，因此可以在简化图的同时确保图信息的完整度。图 3 为粗化任务分配算法示意图，分为粗化阶段和初始划分阶段。示意图最左侧是一个边界和障碍物已知的目标区域，通过网格分解方法完成了区域分解[5]，每个子区域用蓝色的点表示，点的大小代表子区域面积，相邻的子区域用边相连。将目标区域抽象为原始图 G_0，并将其作为 *TAMP* 方法的输入，经过多轮循环的粗化阶段后，得到顶点规模最小的粗化图 G_t，再经过初始划分阶段得到 G_t 的一个尽量均衡的划分结果。由于目标区域抽象点过多，全部展示存在困难，因此目标区域和抽象图 G_0 中点和点之间并不一一对应，点和点之间的边不再专门画出，抽象过程只做示意。

4.1.1 分层粗化

分层粗化阶段对原始图逐层进行极大匹配，并对图的节点进行融合，以降低连通图的规模，最终生成一个顶点规模小于阈值 T 的粗化连通图。考虑到极大匹配简单高效，分层粗化算法采用极大匹配[21]进行节点融合，匹配到的两个节点融合为一个粗化节点，其权重为两个节点权重之和。为了保证连通性，粗化节点之间的边将被保留。同时，为了降低图的复杂度，粗化节点内部子节点之间的边将被隐藏。在粗化图构建过程中，粗化节点和边被重新编号，算法为此保存了新的粗化节点和对应细化节点之间的映射关系，以便在下个阶段的细化算法中用于查询。

图 3 为分层粗化的过程示例图。在对原始图 G_0 进行极大匹配后，得到了第一层粗化图 G_1。如此逐层压缩，直到粗化图顶点数量小于方法预设的阈值 T 时，粗化过程结束，得到顶点规模最小的目标粗化图 G_t。对于任意的 q 划分任务，算法可以将图的顶点数量直接压缩至 q 个。算法未采用该方式的原因是，首先过多的粗化循环次数会导致粗化效率降低；其次过度的粗化会产生局部极大或者极小的节点，造成初始划分任务的不均衡。与基于树划分的多机任务分配方法相比，粗化任务分配算法在粗化阶段会保留粗细节点之间的映射关系，为后续的精细任务分配方法提供更多图的信息。同时，顶点匹配与融合的方式降低了图的规模，为后续的快速求解提供了保证。

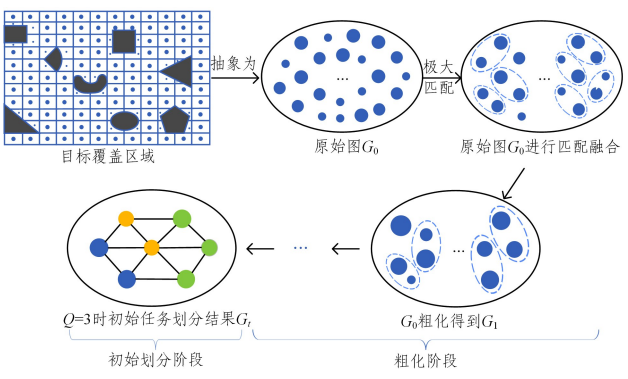


图 3 粗化任务分配算法示意图(电子版为彩图)

Fig. 3 An example of the coarse task assignment algorithm

4.1.2 初始划分

初始划分基于均匀种子的图增长方法，对分层粗化阶段得到的粗化图 G_t 进行划分，得到一个初始任务分配结果。假设需要将粗化图 G_t 划分为 $q(q\geqslant 2, q\in N)$ 个部分。算法从一个顶点开始，以广度优先搜索的方式标记其相邻顶点，并生长出一片区域，直到该区域的权重大于或者等于总权重的 $1/q$，这种方式被称为图增长方式[22]。但是图增长方式的质量对初始顶点比较敏感，容易产生不均衡的结果。为了避免场景中出现个别划分区域过大或者过小的极端情况，间隔一定的步长均匀选择节点序列中的节点作为初始划分的种子，以这些种子为图增长的起点，这种策略被称为均匀种子图增长策略。当所有的种子节点都被选择并搜索完毕后，将未被标记的节点就近划分到相应的分区内，同时更新相应分区的节点集合与权重。最后对边界节点进行标记，在下个阶段任务细化分配时，只对边界节点进行交换和移动，从而提高算法的效率。至此，得到了任务的初始划分：V 的 q 个初始划分 $\{V_i\}_{i=0}^{q-1}$。

粗化任务分配伪代码如算法 1 所示。算法 1 的输入为原始图 G_0 以及划分数量 q，输出为粗化图 G_t 以及初始划分 $\{V_i\}_{i=0}^{q-1}$。算法 1 中的第 1 行表示输入图 G；第 2－10 行是分层粗化，当节点规模大于设定的阈值时，循环迭代粗化步骤；第 4－7 行表示对图进行极大匹配，实现节点融合、更新权重；第 8－9 行表示匹配完毕后对边进行更新，并创建新的粗化图；第 11－16 行表示初始任务划分；第 12 行表示产生均匀种子序列；第 13－14 行表示遍历种子序列，进行图增长初始划分，直到当前划分区域总权重大于等于理想权重；第 15 行表示将未划分节点分配给邻近的区域；第 16 行表示标记边界节点。粗化任务分配算法的复杂度为 $n\log_2 n$。

算法 1 粗化任务分配算法

输入：(G_0, q)

输出：$(G_t, \{V_i\}_{i=0}^{q-1})$

```
1. G← Loadgraph from file(graph file)
2. /* 分层粗化 */
3. While G's nodes number >T do
4.    For v in G's nodes do
5.       v_match=random_match(v)
6.       coarse_node=merge(v_match,v)
7.    End for
8.    Traverse_Update(G's Edges)
9.    G←Construct Coarser G with new nodes,edges
10. End while
11. /* 初始划分 */
12. V_seed=generate_seeds(G_t,policy=uniform)
13. For v_i in V_seed do
14.    V_i←Start BFS from v_i until ω(V_i)≥ω_ideal
15. {V_i}_{i=0}^{q-1}←Allocate unpartitioned nodes
16. V_boarder←Mark G's boundary nodes
```

4.2 精细任务分配算法

粗化任务分配算法输出的是一个初始的任务分配结果，精细任务分配算法在粗化任务分配算法的基础上，利用一个基于边界节点交换的 Lock&Lazy 节点交换策略，对初始任务

分配进行优化，从而最终获得一个近似最优的任务分配结果。精细任务分配算法是通过多层细化实现的，直到恢复至原始图G_0，算法结束。每一层的细化过程都包括增益排序、节点移动和更新细化图3个阶段。精细任务分配算法的过程如图4中蓝色梯形虚线框内的内容所示。细化任务分配算法将上个阶段得到的粗化图G_t作为输入。红色椭圆框内的内容是G_t细化后得到的细化图G_{t-1}，蓝色椭圆框内的内容是G_{t-1}更新细化信息后得到的下一个循环的输入G_{t-2}，依此方法循环迭代，直至恢复到原始图G_0，此时的G_0已经完成任务的均衡划分，在图中的绿色椭圆框中通过不同的颜色标记出属于不同任务划分的节点，G_0中点的颜色对应于图4中左侧原始目标覆盖区域任务的分配结果。

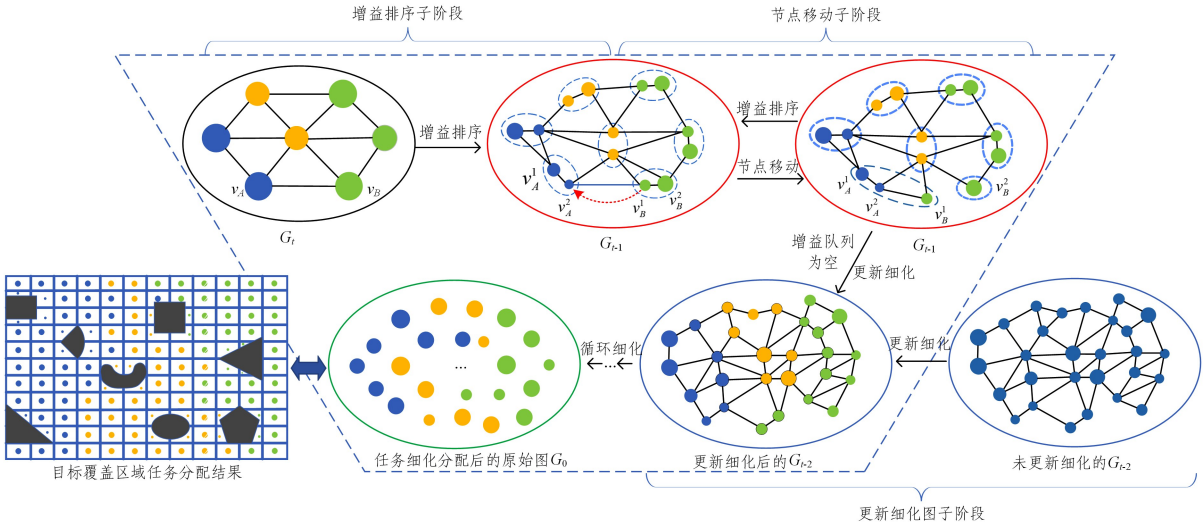


图4 $q=3$时的精细任务分配算法示意图(电子版为彩图)

Fig. 4 Diagram of fine task assignment algorithm when $q=3$

4.2.1 增益排序

由于精细任务分配需要对非完美均衡的初始任务进行再分配，因此需要一个指标来衡量节点重分配的优先级。这里引入了节点移动增益(见式(5))，增益越大，节点移动的优先级就越高。在图4所示的增益排序子阶段，算法根据粗化阶段保存的映射关系将粗化图细化，释放隐藏的节点和边(蓝色椭圆形虚线框中为释放的细化节点)，并根据节点移动增益大小对节点进行排序，生成一个增益从大到小排序的队列。按照增益的顺序进行节点移动，能够快速改善图的均衡程度，并提高求解精度。节点移动增益的定义如下：

$$g_{v\to v_{\mathrm{nei}}}=(\omega(V_i)-\omega(v))*(\omega(V_{\mathrm{nei}})+\omega(v))-\omega(V_i)*\omega(V_{\mathrm{nei}}) \tag{5}$$

其中，v为当前节点，V_i表示当前节点所属区域，v_{nei}表示相邻节点，其所属区域为V_{nei}，节点v移动到v_{nei}所在区域所带来的增益为$g_{v\to v_{\mathrm{nei}}}$。当增益$g_{v\to v_{\mathrm{nei}}}>0$时，说明当前的移动能够改善当前的划分均衡性，$g_{v\to v_{\mathrm{nei}}}$越大，说明当前的移动能够带来越大的增益；当$g_{v\to v_{\mathrm{nei}}}<0$时，说明当前的移动使得划分的均衡性更加糟糕。

由图4可知，G_t中的v_A和v_B两个粗化节点分别释放出了v_A^1，v_A^2和v_B^1，v_B^2这4个细化节点。通过增益排序发现，v_B^1节点移动到v_A^2节点所在的划分区域V_{dst}能够带来最大的均衡增益$g_{v_B^1\to v_A^2}$，因此将$(v_B^1,V_{\mathrm{dst}}):g_{v_B^1\to v_A^2}$作为键值对存储在增益队列中，以便在节点移动阶段进行查询操作。

4.2.2 节点移动

在经过增益排序子阶段后，算法根据队列中增益的顺序进行节点的交换和移动，从而实现任务的精细划分。在精细任务分配循环开始阶段，粗化图的规模较小，节点权重偏大，通过节点移动能够迅速实现任务的快速均衡。当细化阶段接近恢复至G_0时，粗化图已经恢复到接近原始图的粒度，图中节点的权重已经变小，此时再根据增益进行节点移动，能够进一步实现更加均衡的任务分配结果，从而提高计算求解的精度。

在节点移动之前，还需要针对增益队列中的节点进行连通性判断，如果源分区移除当前节点后仍然保持连通，则此移动就是连通有效的。在满足连通有效性的前提下，节点才能执行移动操作。另外，在单次队列遍历过程中，每个划分的任务分区只参与一次移动，如果前面移动的有效节点涉及某个任务分区，则该分区会在当前遍历中被锁定，不参与之后的节点移动。这种先通过预排序、预交换，再经过有效判断后才真正执行交换，一个区域在单次循环中最多参与一次移动的操作被称为Lazy&Lock策略。该策略的优点是通过队列排序、有效性验证以及区域约束大大减少了可交换的节点数量，在提高计算效率的同时，保证了计算求解的准确性。

如图4所示，在节点移动阶段通过对上个阶段构建的增益队列进行遍历，得到增益最大的移动是v_B^1到v_A^2，在通过连通性判断后执行节点移动操作，并且锁定v_B^1和v_A^2所在的绿色和蓝色分区。由于当前任务的$Q=3$，因此队列中后续相关节点已经无法进行移动。接下来回到增益排序子阶段构建新的增益队列，直至增益队列为空，进入更新细化图阶段。

4.2.3 更新细化图

由于只有在完成了粗化任务分配后，最下层的粗化图才包含任务划分信息以及边界节点信息，因此上层的细化图是不包含粗化图中任务分配信息的。这些信息在细化过程中

需要从粗化图同步到相邻的上一层细化图中，以便细化图能利用这些信息进行细化操作。需要同步的更新信息包括更新后的任务划分信息、每个分区的权重以及新的边界标记信息等。

精细任务分配算法循环重复上述3个子阶段，通过分层细化将图细化恢复至原始图，图的均衡因子在此过程中不断地被优化，直到找到最优解或者均衡因子在收敛阈值 ε 以内的近似最优解，算法结束。

整个算法的伪代码如算法2所示。算法的输入为粗化任务分配算法得到的初始划分图 G_t 以及初始划分方案 $\{V_i\}_{i=0}^{q-1}$，输出是细化分配后的任务分配方案 $\{V_i\}_{i=0}^{q-1}$ 以及最大集合权重 $\max\omega(V_i)$。第1行将输入 G_t 赋值给 G；第2行进入分层细化循环，直至恢复到原始图 G_0；第3—9行是增益排序子阶段，首先遍历边界节点，计算其与相邻节点的移动增益并将其添加至增益队列，根据序列是否为空来决定是否进入更新细化图子阶段，若不为空则对增益队列进行排序；第10—19行是节点移动子阶段，第11—18行按照增益队列中增益从大到小的顺序对可移动节点进行连通性和有效性验证，待通过验证后执行节点移动操作，并且更新边界节点，锁定当前参与移动的源分区和目的分区；第19行在节点移动结束后重新回到增益排序子阶段；第20—23行是更新细化图子阶段，第21行将更新信息同步到上一层细化图中，第22行中若当前平衡因子小于收敛阈值，则算法结束，第23行将更新信息后的细化图传至下个循环。

算法2 精细任务分配算法

输入：$(G_t,\{V_i\}_{i=0}^{q-1})$

输出：$(\{V_i\}_{i=0}^{q-1},\max\omega(V_i))$

1. $G\leftarrow G_t$
2. While $G!=G_0$ do
3. /* 增益排序子阶段 */
4. For v_b in $V_{boarder}$ do
5. $g_{max}^b, V_{dst}\leftarrow$ Calculate_max_gain(v_b)
6. Dic_{gain}.append({(v_b, V_{dst}):g_{max}^b})
7. End for
8. if Dic_{gain} is empty then go to line 21
9. Sort Dic_{gain} by g_{max}^b
10. /* 节点移动子阶段 */
11. For(v_b, V_{dst}), g in Dic_{gain} do
12. if(v_b, V_{dst})in locklist then continue
13. if $V_{src}-\{v_b\}$ is connected then
14. $V_{dst}\leftarrow V_{dst}\cup\{v_b\}$
15. $V_{src}\leftarrow V_{dst}-\{v_b\}$
16. Update G's $V_{boarder}$
17. Locklist.append(V_{dst} and V_{src})
18. End for
19. go to line 4
20. /* 更新细化图子阶段 */
21. Update G's info to finer G
22. if G's B $\leqslant\varepsilon$ then return
23. $G\leftarrow$ finer G
24. End while

5 仿真实验验证

5.1 实验设置与评价指标

5.1.1 实验设置

实验台式机配置为 IntelCorei7-8700 3.2 GHz 处理器，8GRAM 和 Ubuntu 1604 x64 操作系统，TAMP 算法采用 Python3.6 编写，对比的算法包括 Mofint 和 STED。

实验中使用的数据集除了 Matic[7] 以及 Zhou 等[8] 给出的小规模实例之外，还考虑了大规模实例以及真实世界的实例[9]，真实世界实例是针对4个城市治安巡逻问题的公共安全地图数据。地图需要被划分为 q 个犯罪率大致相同的连通区域，以平衡 q 个警察的工作量。这个问题本质上就是一个多机器人覆盖的任务分配问题，符合本文的场景需求。每个实例图运行3次取平均值作为实验结果。数据集包括如下3类：

1)小规模随机网格图，节点规模 $N\in\{50,400,3000\}$，每种规模包含3张不同的实例图，顶点权重在[1,20]之间。

2)大规模随机网格图，节点规模 $N\in\{10^4,10^5,10^6\}$，每种规模包含3张不同的实例图，顶点权重在[1,20]之间。

3)真实世界实例，包括美国4个城市部分区域的公共安全地图数据，节点规模为[400,1500]，节点权重在[0,15000]之间。

实验中粗化阈值 T 被设置为划分任务数量 Q 的3倍，大量实验结果证明，这样的设置往往能够获取更快和更精确的实验结果；在大规模图的实验验证中引入参数——收敛阈值 ε，当细化阶段的均衡因子 $B\leqslant\varepsilon$ 时，算法收敛并结束。大规模图的总权重一般较大，ε 的选择对算法的完成时间有着关键影响，若 ε 过大则会造成计算精度下降，若 ε 过小则会造成计算时间成本增加。

5.1.2 评价指标

对于 BCP_q 问题，经典的方法[7-9]一般使用计算时间、最小最大权重、距离最优解间距、是否找到最优解和实现最优解的次数作为评价指标。本实验沿用了计算时间 T 和距离最优解间距两个指标，并增加了一个计算规模的指标。为方便表示，我们将距离最优解间距转换为另一种表达方式并将其定义为均衡因子 B(见式(6))。均衡因子表示实际计算值和理想值的距离偏差，均衡因子越趋近0，说明划分结果越均衡；计算时间表示从图输入到划分完毕的时间，时间越短，说明算法的效率越高，覆盖的计算成本越低；计算规模包括节点规模 N 和划分数量 Q，节点规模是抽象图的节点数量，划分数量是要划分的目标分区数，计算规模越大，说明越能适用于更大规模的场景，算法的扩展性越强。

衡量图划分结果的一个重要指标为均衡因子，符号记作 B，其具体定义如下：

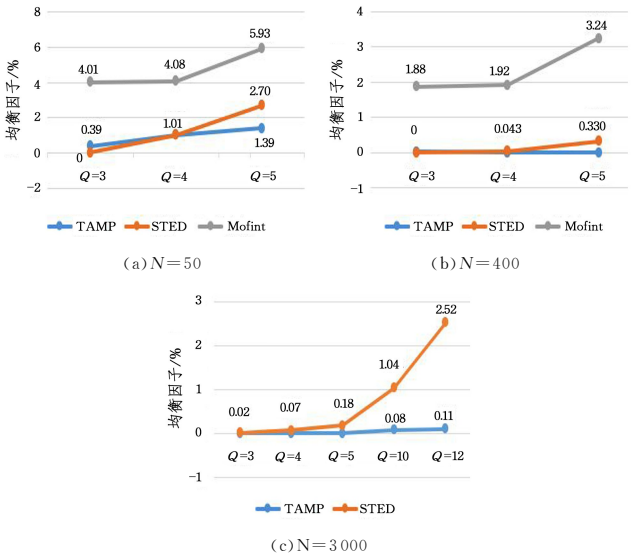
$$B=\frac{\omega_{\max}}{\omega_{\text{ideal}}}-1,B\geqslant 0 \tag{6}$$

其中，最大划分权重 $\omega_{\max}=\max\omega(V_i)$，理想划分权重 $\omega_{\text{ideal}}=\lceil\omega_V/q\rceil$。均衡因子 B 代表当前任务划分结果的好坏，是衡量划分算法的一个重要指标，B 越接近0，表示当前的划分越均衡，在实际划分中可能永远得不到一个 $\omega_{\max}=\omega_{\text{ideal}}$ 的划分。

5.2 小规模图的对比

在小规模图的实验中，我们对比了 STED 和 Mofint 两种算法，目的是对比相同可处理规模的条件下，TAMP 算法在时间和计算精度上的优越性。由于 Mofint 算法的可解决问题上限为 $N=400$，STED 算法的可解决问题规模上限 $N=3000$，因此我们设计了包括顶点规模为 $N=50$，$N=400$，$N=3000$ 这 3 种数量级的小规模图仿真实验，同时设置划分任务数量 $Q=3$，$Q=4$ 以及 $Q=5$。需要说明的是，当 $N=3000$ 时，实验额外增加了 $Q=10$ 和 $Q=12$ 的划分，这是因为 STED 算法在原实验中的顶点规模上限为 $N=3000$，划分任务数量上限为 $Q=12$，且在该参数下能得到不大于 3% 的距离最优解偏差[8]。因此，通过设置相同的顶点规模和划分数量等参数进行对比，实验会更加合理有效。

图 5 为小规模图均衡因子对比图，横坐标 Q 表示划分数量，纵坐标表示均衡因子 B（见式(6)），均衡因子越小，算法性能就越好。从图中能够看出，在 3 种小规模图的场景下，TAMP 算法均衡因子基本都小于 STED 算法和 Mofint 算法。随着划分数量 Q 的增加，3 种算法的平衡因子都不同程度地增加，TAMP 算法的平衡因子增加的趋势更加平缓，说明它对计算规模的扩展性更强，当 Q 较大时，TAMP 算法的优势更大。特别地，当 $N=3000$，$Q=12$ 时，STED 算法的均衡因子为 2.52%，符合 Zhou 等在其实验中得出的小于 3% 距离最优解偏差的结论[8]，TAMP 算法的距离最优解偏差为 0.11%，优于 STED 算法。



(a) $N=50$ (b) $N=400$

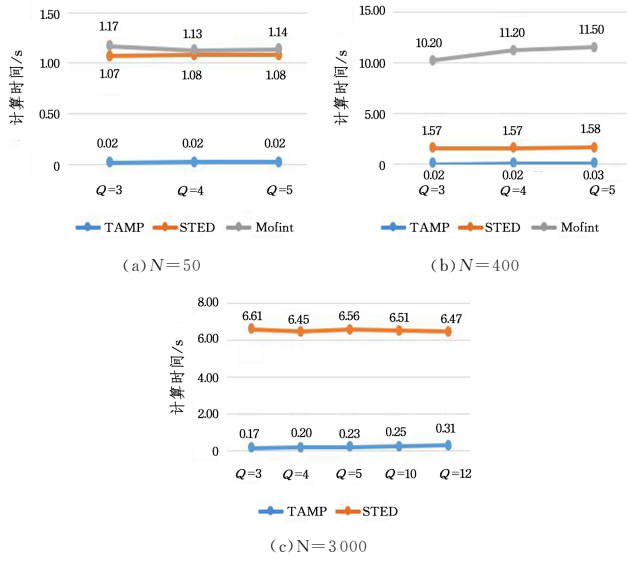
(c) $N=3000$

注：均衡因子越小，算法性能越好。

图 5 小规模图均衡因子对比图

Fig. 5 Comparison results of balance factors in small-scale graph

图 6 为小规模图计算时间对比图，横坐标 Q 表示划分数量，纵坐标为计算时间，时间越短，说明计算效率越高，计算性能越好。从图中能够看出，TAMP 算法的时间在 3 种规模图下都是最短的。对于同样规模和 Q 值的任务划分场景，TAMP 的速度比 STED 和 Mofint 至少加快了 20 倍（当 $N=3000$，$Q=12$ 时）。随着划分数量 Q 的增加，计算时间增加放缓，展现出了 TAMP 算法在小规模图上的良好扩展性。



(a) $N=50$ (b) $N=400$

(c) $N=3000$

注：时间越短，算法性能越好。

图 6 小规模图计算时间对比图

Fig. 6 Comparison results of computation time in small-scale graph

5.3 大规模图的对比

在大规模图的实验中，由于 STED 和 Mofint 算法不适用于大于 3000 个节点的最小最大平衡连通图划分任务，因此对 TAMP 算法的计算时间、均衡因子以及计算规模进行测试。如表 1 所列，第一列表示图的规模；第二列 Q 表示划分数量，包括 $Q=3$，$Q=4$，$Q=5$ 的小任务数量的划分，还包括 $Q=10$，$Q=20$ 的大任务数量的划分；第三列表示均衡因子 B；第四列表示计算时间，单位为 s。

表 1 大规模图的实验结果

Table 1 Experimental results in large-scale graph

Scale	Number of divisions(Q)	B	computation time/s
$N=10^4$	$Q=3$	0	1.30
	$Q=4$	0	1.50
	$Q=5$	0	1.90
	$Q=10$	$1*10^{-4}$	2.20
	$Q=20$	$2.61*10^{-3}$	2.35
$N=10^5$	$Q=3$	$0.5*10^{-}$	5.10
	$Q=4$	$1.3*10^{-4}$	5.10
	$Q=5$	$1.4*10^{-4}$	5.40
	$Q=10$	$2.8*10^{-4}$	5.45
	$Q=20$	$5.1*10^{-4}$	8.35
$N=10^6$	$Q=3$	$1.5*10^{-5}$	36.40
	$Q=4$	$3.5*10^{-5}$	35.80
	$Q=5$	$4*10^{-5}$	36.40
	$Q=10$	$1.5*10^{-4}$	50.10
	$Q=20$	$1.8*10^{-4}$	59.00

从表中可以看出，TAMP 算法在 3 种大规模场景下都能将均衡因子控制在 0.3% 以内，能够在规定收敛阈值的条件下，在 2.4 s 内完成 $N=10000$，8.4 s 内完成 $N=100000$，60 s 内完成 $N=1000000$ 以内的图划分任务。随着图规模、Q 值的增加，TAMP 算法的均衡因子缓慢增加，计算时间缓慢变长，说明扩展性很好；随着图规模的增大，均衡因子在逐渐变小，这是因为随着规模的增加，图的总权重随之增加。在大规模图的划分任务中，设置收敛阈值 $\varepsilon=0.02\%$，当均衡因子

小于收敛域值时，算法结束，以此来加快算法的收敛速度。

5.4 真实世界图的对比

除了不同规模的随机图，实验还考虑了现实世界中的真实覆盖应用。实验使用了包含4个城市在内的警察治安巡逻覆盖问题的地图数据进行仿真验证，这些地图被抽象成了图，并且首次在文献[9]中被当作解决BCP_q问题的测试实例。从表2中可以看出，在节点规模$N \leqslant 1\,368$的真实世界图上，TAMP算法能够在0.68s的时间内实现平衡因子小于5.4%的近似最优结果，在精度和时间上都要比STED算法优秀(表中加粗的数据表示性能更优)。

表2 真实世界图的实验结果

Table 2 Experimental results in real world graph

Graph name	Graph information	Number of divisions(Q)	STED		TAMP(Ours)	
			B/%	T/s	B/%	T/s
Nyc_hellskitchen	$N=498$, $E=746$, $\omega_{total}=751\,857$	$Q=3$	0.025	2.61	**0.002**	**0.21**
		$Q=5$	0.122	2.22	**0.069**	**0.16**
		$Q=10$	2.391	2.29	**2.064**	**0.11**
		$Q=20$	8.494	2.28	**5.315**	**0.22**
Campinas_centro	$N=579$, $E=942$, $\omega_{total}=2\,152\,416$	$Q=3$	0.046	2.28	**0.036**	**0.10**
		$Q=5$	0.228	2.52	**0.190**	**0.08**
		$Q=10$	1.855	2.45	**0.443**	**0.13**
		$Q=20$	4.910	2.48	**4.500**	**0.17**
Chicago_lakeview	$N=1\,004$, $E=1\,563$, $\omega_{total}=416\,688$	$Q=3$	0.058	3.39	**0.012**	**0.42**
		$Q=5$	0.495	3.29	**0.116**	**0.49**
		$Q=10$	2.651	3.45	**0.825**	**0.42**
		$Q=20$	4.922	3.49	**4.312**	**0.61**
La_hollywood	$N=1\,368$, $E=2\,030$, $\omega_{total}=4\,102\,848$	$Q=3$	0.009	5.25	**0.006**	**0.68**
		$Q=5$	0.277	5.83	**0.036**	**0.32**
		$Q=10$	2.443	4.83	**0.555**	**0.58**
		$Q=20$	5.142	4.55	**3.401**	**0.44**

结束语 针对大规模场景下精确高效的多机任务分配问题，本文提出了一种启发式的多机器人任务分配方法TAMP。该方法首先采用分层粗化的方式，通过图的极大匹配实现节点融合以降低图的规模，并基于均匀种子的图增长方式快速获得一个尽量均衡的初始任务划分结果；接下来，TAMP提出了基于边界节点交换的Lazy&Lock策略，通过构造增益队列的方式对初始划分结果进行任务细分，进而获得接近最优的任务分配方案。本文分别在不同规模随机图以及真实世界实例上进行了仿真实验。实验结果表明，TAMP方法在计算规模、计算精度和计算效率上都优于经典的多机任务分配算法。

未来，我们将提出一些优化策略，来进一步提高算法的效率，使得TAMP能够在时间上得到进一步改善；其次，我们希望能够将算法扩展到不确定环境下的动态图中，使其更加符合实际的应用场景。

参 考 文 献

[1] SHEN Z, SONG J, MITTAL K, et al. CT-CPP: Coverage Path Planning for 3D Terrain Reconstruction Using Dynamic Coverage Trees[J]. IEEE Robotics and Automation Letters, 2022, 7(1): 135-142.

[2] CARBONE C, ALBANI D, MAGISTRI F, et al. Monitoring and mapping of crop fields with UAV swarms based on information gain[C]// International Symposium Distributed Autonomous Robotic Systems. Cham: Springer, 2021: 306-319.

[3] CABREIRA T M, DI FRANCO C, FERREIRA P R, et al. Energy-aware spiral coverage path planning for uav photogrammetric applications[J]. IEEE Robotics and Automation Letters, 2018, 3(4): 3662-3668.

[4] HAYAT S, YANMAZ E, BROWN T X, et al. Multi-objective UAV path planning for search and rescue[C]// IEEE International Conference on Robotics and Automation. IEEE, 2017: 5569-5574.

[5] GALCERAN E, CARRERAS M. A surveyon coverage path planning for robotics[J]. Robotics and Autonomous Systems, 2013, 61(12): 1258-1276.

[6] REKLEITIS I, NEW A P, RANKIN E S, et al. Efficient boustrophedon multi-robot coverage: an algorithmic approach[J]. Annals of Mathematics and Artificial Intelligence, 2008, 52(2): 109-142.

[7] MATIĆ D. A mixed integer linear programming model and variable neighborhood search for maximally balanced connected partition problem[J]. Applied Mathematics and Computation, 2014, 237: 85-97.

[8] ZHOU X, WANG H, DING B, et al. Balanced connected task allocations for multi-robot systems: An exact flow-based integer program and an approximate tree-based genetic algorithm[J]. Expert Systems with Applications, 2019, 116: 10-20.

[9] MIYAZAWA F K, MOURA P F S, OTA M J, et al. Partitioning a graph into balanced connected classes: Formulations, separation and experiments[J]. European Journal of Operational Research, 2021, 293(3): 826-836.

[10] KRATICA J. Solving the maximally balanced connected partition problem in graphs by usinggenetic algorithm[J]. Computing and Informatics, 2008, 27(3): 341-354.

[11] ZHOU X, WANG H, DING B. How many robots are enough: a multi-objective genetic algorithm for the single-objective time-limited complete coverage problem[C]// IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018: 2380-2387.

[12] AGMON N, HAZON N, KAMINKA G A. Constructing span-

ning trees for efficient multi-robot coverage[C]// Proceedings 2006 IEEE International Conference on Robotics and Automation. IEEE，2006：1698-1703.

[13] KARAPETYAN N，BENSON K，MCKINNEY C，et al. Efficient multi-robot coverage of a known environment[C]// 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE，2017：1846-1852.

[14] AZPÚRUA H，FREITAS G M，MACHARET D G，et al. Multi-robot coverage path planning using hexagonal segmentation for geophysical surveys[J]. Robotica，2018，36(8)：1144-1166.

[15] CHOSET H. Coverage for robotics—a survey of recent results [J]. Annals of Mathematics and Artificial Intelligence，2001，31(1)：113-126.

[16] SHIVASHANKAR V，JAIN R，KUTER U，et al. Real-time planning for covering an initially-unknown spatial environment [C]// Twenty-Fourth International FLAIRS Conference. 2011.

[17] HAZON N，KAMINKA G A. Redundancy，efficiency and robustness in multi-robot coverage[C]// Proceedings of the 2005 IEEE International Conference on Robotics and Automation. IEEE，2005：735-741.

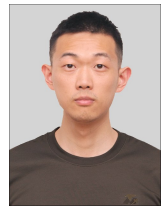
[18] FREDERICKSON G N，ZHOU S. Optimal parametric search for path and tree partitioning[J]. arXiv：1711. 00599，2017.

[19] CHEN G，CHEN Y，CHEN Z Z，et al. Approximation algorithms for the maximally balanced connected graph tripartition problem [J]. Journal of Combinatorial Optimization，2022，44(3)：1753-1773.

[20] CHEN Y，CHEN Z Z，LIN G，et al. Approximation algorithms for maximally balanced connected graph partition[J]. Algorithmica，2021，83(12)：3715-3740.

[21] WANG S H. Graph Theory，2nd ed[M]. Science Press，2009.

[22] KARYPIS G，KUMAR V. A fast and high quality multilevel scheme for partitioning irregular graphs[J]. SIAM Journal on scientific Computing，1998，20(1)：359-392.



AN Haojia，born in 1991，postgraduate. His main research interests include robotics and artificial intelligence.

SHI Dianxi，born in 1966，Ph.D，professor，Ph.D supervisor，is a member of China Computer Federation. His main research interests include distributed object middleware technology，adaptive software technology，artificial intelligence，and robot operating systems.

（责任编辑：喻藜）