中国科学: 信息科学 2022 年 第 52 卷 第 6 期: 960–970
SCIENTIA SINICA Informationis
理论计算机科学专题 · 论文

最小最大圈覆盖问题的精确算法

袁森, 陈开奇, 李江坤, 张鹏*

山东大学软件学院, 济南 250101
* 通信作者. E-mail: algzhang@sdu.edu.cn

收稿日期: 2021–12–31; 修回日期: 2022–03–24; 接受日期: 2022–04–15; 网络出版日期: 2022–06–08

国家自然科学基金 (批准号: 61972228, 61672323)、山东省自然科学基金重大基础研究 (批准号: ZR2021ZD15) 和山东省自然科学基金 (批准号: ZR2019MF072) 资助项目

摘要 最小最大圈覆盖问题是旅行售货商问题的推广, 该问题在无线传感器网络和无人机救灾等领域有着广泛的应用. 目前关于最小最大圈覆盖问题的研究主要集中在近似算法方面, 而缺少精确算法方面的结果. 本文根据最小最大圈覆盖问题的组合特征, 对该问题设计了基于动态规划策略的首个精确算法, 时间复杂度为 $O^*(3^n)$. 本文将对最小最大圈覆盖问题的求解分为两个阶段, 第一个阶段是对问题的输入进行预处理, 第二个阶段是在预处理的基础上求问题的最优解. 有趣的是, 两个阶段的方法都是基于动态规划策略设计的, 这是本文处理最小最大圈覆盖问题的一个主要的特色. 本文所证明的求到最优解的时间 $O^*(3^n)$, 显著优于基于暴力搜索策略的枚举算法的时间.

关键词 最小最大圈覆盖问题, 动态规划, 分支定界, 精确算法

1 引言

1.1 问题的定义和来源

最小最大圈覆盖问题 (min-max cycle cover problem, MMCCP) 是运筹学和组合优化领域中的一个基本的问题, 是著名的旅行售货商问题 (traveling salesman problem, 简称为 TSP 问题) [1,2] 的推广. 该问题如定义 1 所示.

定义1 (最小最大圈覆盖问题) 实例: 完全图 $G = (V, E)$, 其中 V 是顶点集合, E 是边集合. 边 (v_i, v_j) 上有非负权重 (也称为长度) $w(v_i, v_j)$. 边上的权重满足三角不等式. 问题的实例中还包括一个正整数 k.

目标: 找不超过 k 个圈, 覆盖图 G 上的所有顶点, 使得这些圈中的最大圈的长度最小. 圈的长度定义为该圈上所有边的权重之和. 最大圈是指长度最大的圈.

引用格式: 袁森, 陈开奇, 李江坤, 等. 最小最大圈覆盖问题的精确算法. 中国科学: 信息科学, 2022, 52: 960–970, doi: 10.1360/SSI-2021-0444
Yuan S, Chen K Q, Li J K, et al. Exact algorithms for the min-max cycle cover problem (in Chinese). Sci Sin Inform, 2022, 52: 960–970, doi: 10.1360/SSI-2021-0444

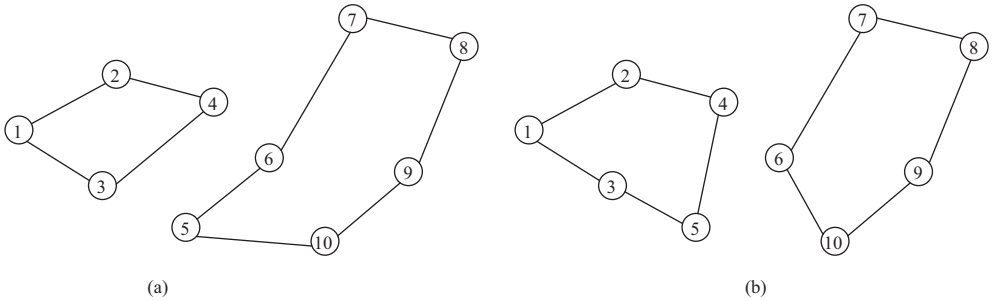

图 1 不平衡的 (a) 和平衡的 (b) 数据收集路径示例

Figure 1 An example of the unbalanced (a) and balanced (b) data gathering route

最小最大圈覆盖问题是组合优化问题的典型代表, 其应用广泛, 应用领域包括物流配送 [3]、车辆路径规划 [4]、无人机灾难救援 [5,6]、无线传感器网络 [7,8] 等. 例如, 在无线传感器网络中, 传感器之间距离较远, 无法直接相互通信, 因此如何从传感器收集数据就成为一个基本的问题. 解决该问题的一个有效的方法是, 调度多个移动接收器来收集来自传感器的数据. 每一个移动接收器收集完数据后都需要回到它出发的位置, 因此移动接收器的路径是一个圈. 由于移动接收器的工作时间是受到它所携带的电能的约束的, 我们希望平衡不同数据收集路径的长度, 以便尽快收集传感数据. 例如, 图 1(a) 和 (b) 表示了在由 10 个无线传感器组成的网络上, 有两种不同的数据收集路径. 图上的边表示移动接收器在数据收集途中的移动路径. 显然, 收集数据的完成时间取决于最大圈的长度. 图 1(a) 是不平衡的数据收集路径的例子, 其收集完成时间长; 图 1(b) 是平衡的数据收集路径的例子, 其收集完成时间短. 因此, 在无线传感器网络的数据收集问题中, 我们希望数据收集路径的最大长度最短, 这恰好就得到了最小最大圈覆盖问题.

在此对本文经常使用的两个符号作一说明. (1) 给一个图 $G=(V,E)$, 总是用 n 来表示图 G 上的顶点数目. (2) $O^*(f(n))$ 表示忽略 n 的任意多项式的渐近符号. 本文使用的其他符号在该符号被首次使用时进行说明.

1.2 相关工作

众所周知旅行售货商问题是 NP 困难问题. 作为旅行售货商问题的推广, MMCCP 问题自然也是 NP 困难的. 目前关于 MMCCP 问题的研究主要集中在近似算法方向. Xu 等 [9] 证明, 若最小最大树覆盖问题可近似到 ρ, 则最小最大圈覆盖问题可近似到 2ρ. 由于最小最大树覆盖问题存在 3- 近似算法 [10], 这表明最小最大圈覆盖问题可近似到 6. Jorati [11] 在 2013 年和 Xu 等 [12] 在 2015 年独立地得到了最小最大圈覆盖问题的 $(16/3+\epsilon)$- 近似算法, 在此 $\epsilon>0$ 是一个任意小的常数. 文献 [11,12] 使用了不同的技术. 2016 年, Yu 等 [13] 将最小最大圈覆盖问题的近似比改进到 5, 关于该问题这是目前已知的最好的近似比.

MMCCP 问题还有许多变化的版本, 如有根的 MMCCP 问题, Xu 等 [12] 在 2015 年设计了有根的 MMCCP 问题的 $(7+\epsilon)$- 近似算法, 其中 $\epsilon>0$ 是一个任意小的常数. Yu 和 Liu [14] 在 2019 年在 Xu 等 [12] 算法的基础上进行改进, 设计了该问题的 6- 近似算法.

TSP 问题是经典的、著名的组合优化问题, 关于该问题有大量的算法方面的研究结果, 本文仅做简要介绍. 对于一般的 TSP 问题, 可以使用动态规划的方法在 $O^*(2^n)$ 时间内找到最优解 [1,2]. 这是目前已知的 TSP 问题求到最优解的最快的方法. 近似算法方面的研究主要集中在度量空间中的 TSP 问题 (即边上的长度满足三角不等式的 TSP 问题) 上. 长久以来, 度量空间 TSP 问题已知的近似比

是 3/2 [15]. 最近 (2021 年), 这一问题的近似比 (在期望意义上) 被改进到 $3/2-\epsilon$, 其中 $\epsilon \approx 10^{-36}$ 是一个很小的常数 [16]. 这是度量空间 TSP 问题的一个重大突破. 在求解 TSP 问题的启发式算法方面, 有遗传算法 [17]、蚁群算法 [18, 19]、粒子群算法 [20] 等. 这些算法通常能够以较快的速度得到一个解, 但无法保证求出的解是最优的.

1.3 本文的结果

本文对最小最大圈覆盖问题给出了首个精确算法, 时间复杂度为 $O^*(3^n)$. 算法分为两个阶段. 首先, 在第一阶段使用动态规划技术对问题的输入进行预处理, 得到每个顶点子集上的最优 TSP 回路 (即 TSP 问题的最优解, 这是经过该子集内每个顶点一次且仅一次的长度最短的圈). 然后, 在第二阶段再使用动态规划技术计算出最小最大圈覆盖问题的最优解. 在两个阶段都使用动态规划技术对问题进行求解, 是本文算法技术的主要特色和创新点. 另外, 本文还给出了求解 MMCCP 问题的简单暴力搜索算法 (即简单枚举算法), 证明其时间复杂度为 $O^*(k^n n!)$. 因此, 本文所给出的基于动态规划技术求解 MMCCP 问题的精确算法, 其时间复杂度比简单暴力搜索算法有显著的改进. 最后, 本文还对 MMCCP 问题设计了分支定界算法. 分支定界算法亦能求到 MMCCP 问题的最优解. 本文在 MMCCP 分支定界算法上的贡献是算法所使用的分支规则和剪枝策略[1)].

本文余下的部分作如下安排. 首先, 在第 2 节介绍 MMCCP 问题的简单枚举算法. 然后在第 3 节介绍使用动态规划技术对 MMCCP 问题输入的预处理. 第 4 节给出了 MMCCP 问题的分支定界算法, 第 5 节给出了 MMCCP 问题的动态规划算法, 这是本文的主要结果. 最后, 第 6 节总结全文.

2 MMCCP 问题的简单枚举算法

MMCCP 问题是一个典型的组合问题, 其最简单的求解策略就是枚举所有可能的解, 通过比较, 找出最优解. 本节给出 MMCCP 问题的一种暴力搜索枚举算法, 并分析其时间复杂度. 介绍这一算法的目的是与本文的主要结果 (即 MMCCP 的动态规划算法) 进行对比.

首先, 介绍 MMCCP 问题最优解的一个性质. 在 MMCCP 问题的定义中 (定义 1), 问题的解将顶点集 V 划分成 $\leqslant k$ 个子集. 问题的解是不超过 k 个圈的集合. 解的值是解中最大圈的长度, 简称为 “解值”.

引理1 不失一般性, 可假设 MMCCP 问题的最优解将顶点集 V 划分成了恰好 k 个子集.

证明 若问题实例中子集个数 k 大于等于顶点个数 n, 则问题的最优解是平凡的, 每个顶点自成一个子集即可, 此时解值为 0. 因此接下来我们可以假设 $k<n$.

令 $\mathcal{C}^*$ 是问题的一个最优解. 假设 $\mathcal{C}^*$ 将顶点集 V 划分成了 $k'<k$ 个子集 $V_1, V_2, \ldots, V_{k'}$. 由问题的定义, 最优解 $\mathcal{C}^*$ 对每个子集使用一个圈进行覆盖. 因此, $\mathcal{C}^*$ 中含有 k' 个圈 $C_1, C_2, \ldots, C_{k'}$. 由于 $k'<k<n$, 则 $V_1, V_2, \ldots, V_{k'}$ 中必有一个子集含有 $\geqslant 2$ 个顶点, 假设这个子集为 V_i $(1 \leqslant i \leqslant k')$, $\mathcal{C}^*$ 中覆盖 V_i 的圈为 C_i. 在 V_i 中任取一个顶点 v, 将 V_i 分成两个子集 $V_i' = \{v\}$ 和 $V_i'' = V \setminus V_i'$. 由于 $|V_i'| = 1$, 可以使用一个长度为 0 的圈覆盖 V_i' (即 V_i' 中的顶点 v 自成一个圈 C_i'). 在圈 C_i 上去掉顶点 v, 余下的顶点使用抄近路的方法再连成一个圈, 则得到覆盖 V_i'' 的一个圈 C_i''. 如图 2 所示.

这样, 就得到了一个新的解 $\mathcal{C}'$, 所包含的圈的个数为 $k'+1$. 由三角不等式, C_i'' 的长度不超过 C_i 的长度, 因此 $\mathcal{C}'$ 的解值 (即最大圈的长度) 不超过 $\mathcal{C}^*$ 的解值, 即 $\mathcal{C}'$ 也是最优解. 重复应用上述论证, 可

1) 说明: “分支定界” (branch-and-bound) 还有写作 “分枝定界” 的. 本文在叙述分支定界这一算法时, 使用了抽象意义的 “分支”. 但在叙述算法中的剪枝策略时, 仍然使用了更加具体的 “剪枝” 的写法.

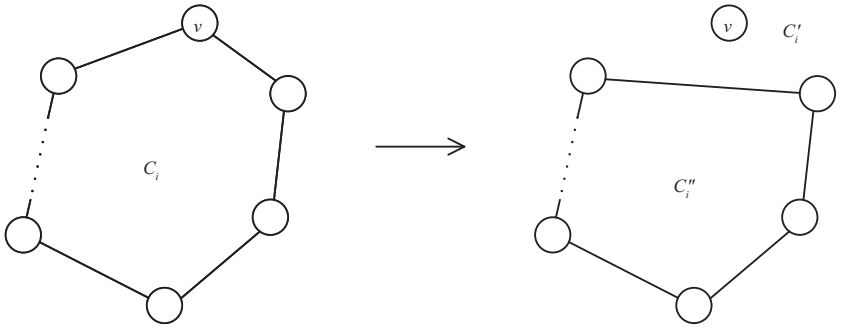

图 2 将圈 C_i 分解为两个圈 C_i' 和 C_i''

Figure 2 Decompose cycle C_i into two cycles C_i' and C_i''

得到一个将顶点集 V 划分成了恰好 k 个子集的最优解.

由引理 1, 我们只需要考虑将顶点集 V 划分成 k 个子集的划分. 为简单起见, 称这种划分为 k- 划分. MMCCP 问题的简单枚举算法的思想是首先枚举出所有可能的 k- 划分, 然后对每一个 k- 划分, 找 k 个圈覆盖这个划分中 k 个子集中的所有顶点, 使最大圈的长度最小. 这样, 每个 k- 划分就对应一个解. 所有的 k- 划分中解值最小的, 显然是 MMCCP 问题的最优解.

定理1 通过简单枚举的方法, 可在 $O^*(k^n n!)$ 时间内求到 MMCCP 问题的最优解.

证明 在一个 k- 划分中, 要把 n 个顶点放入 k 个集合中. 由于每一个顶点有 k 种选择, 且一共有 n 个顶点, 因此 k- 划分的总个数不超过 k^n 个. 然后, 对一个 k- 划分, 计算产生它的解所需要的时间. 假设这个划分为 $\mathcal{P} = \{V_1, V_2, \ldots, V_k\}$, 子集 V_i 的大小为 n_i. 在子集 V_i 上, 要找一个顶点的排列, 使其构成的圈的长度最短. 这样的排列共有 $n_i!$ 种. k- 划分 $\mathcal{P}$ 的所有子集的顶点排列的总的数目即为 $n_1!+ n_2!+ \cdots + n_k! = O(n!)$. 因此, MMCCP 问题可通过简单枚举的方法在 $O^*(k^n n!)$ 时间内求到最优解.

在子集 V_i 上求长度最短的圈覆盖其全部顶点, 就是求解 V_i 上的 TSP 问题. 使用旅行售货商问题的动态规划算法 [1,2], 可在 $O^*(2^{n_i})$ 时间内求到最优解. 因此, 若在每个子集上使用 TSP 的动态规划算法, 则 MMCCP 问题可在 $O^*(k^n 2^n)$ 时间内求到最优解. 这是对定理 1 中简单枚举算法的一个改进.

3 MMCCP 问题实例的预处理

本节介绍使用动态规划技术对 MMCCP 问题实例进行预处理的方法. 本节的目的是为第 4 节的分支定界算法和第 5 节的动态规划算法做准备.

最小最大圈覆盖问题的解用不超过 k 个圈覆盖所有的顶点, 因此, 在一个顶点子集 S 上求覆盖这个子集中所有顶点的最短的圈, 是解决最小最大圈覆盖问题必须解决的一个基本步骤. 这实际上是要在图 $G[S]$ 上求解 TSP 问题. 在这里, $G[S]$ 指图 G 在顶点子集 S 上的导出子图, 即 $G[S]$ 上包含了 S 中所有的顶点, 以及 G 上所有两个端点都在 S 中的边.

在有 n 个顶点的图 G 上求解 TSP 问题, 有著名的众所周知的动态规划算法 [1,2], 其时间复杂度为 $O^*(2^n)$. 在最小最大圈覆盖问题中, 我们需要在每一个顶点子集上解 TSP 问题, 以获得在这个顶点子集上的长度最短的圈. 由于一共有 2^n 个可能的顶点子集, 这样初看上去, 如果把 TSP 的动态规划算法当作黑盒调用, 要求出所有顶点子集上的最优 TSP 回路, 就需要花费 $O^*(2^n \cdot 2^n) = O^*(4^n)$ 时间. 但实际上, 本文证明可以在更快的 $O^*(2^n)$ 时间内解决此问题. 这需要深入到 TSP 问题的动态规划算

表 1　计算所有包含 v_1 的顶点子集上的最优 TSP 回路所使用的动态规划表 sp

Table 1　Dynamic program table sp used to compute the optimal TSP tours for all the subsets containing v_1

	v_2	v_3	v_4	…	v_{n-1}	v_n
$\emptyset$	c_{21}	c_{31}	c_{41}	…	$c_{n-1,1}$	c_{n1}
$\{v_2\}$	×	f	f	f	f	f
$\{v_3\}$	f	×	f	f	f	f
⋮						
$\{v_n\}$	f	f	f	f	f	×
$\{v_2, v_3\}$	×	×	f	f	f	f
$\{v_2, v_4\}$	×	f	×	f	f	f
⋮						
$\{v_{n-1}, v_n\}$	f	f	f	f	×	×
⋮						
$\{v_2, v_3, \ldots, v_{n-1}\}$	×	×	×	×	×	f
$\{v_2, v_3, \ldots, v_{n-2}, v_n\}$	×	×	×	×	f	×
⋮						
$\{v_3, \ldots, v_{n-2}, v_n\}$	f	×	×	×	×	×

法内部, 即 “白盒” 的方法. 下面介绍这一方法.

引理2　可在 $O^*(2^n)$ 时间内计算出顶点集 V 的所有非空子集上的最优 TSP 回路.

证明　假设图 G 上所有的顶点编号为 $v_1, v_2, \ldots, v_n$. 首先计算出所有包含 v_1 的顶点子集上的最优 TSP 回路. 然后, 在图上去掉 v_1, 计算出所有包含 v_2 的顶点子集上的最优 TSP 回路. 然后, 在图上继续去掉 v_2, 计算出所有包含 v_3 的顶点子集上的最优 TSP 回路. 依次类推, 直到图上只剩下一个顶点 v_n, 单独一个顶点 v_n 上的最优 TSP 回路就是 v_n 本身, 长度为 0.

下面考虑所有包含 v_1 的顶点子集上的最优 TSP 回路的计算方法. 这实际上要用到 TSP 问题的动态规划算法. 构造一个动态规划表格 sp, sp 的每一行表示一个顶点子集 S (不包含 v_1), sp 的每一列表示一个顶点 (不含 v_1). 任取一个顶点 $v_i \neq v_1$. 假设顶点子集 S 不包含 v_i (从而, S 既不包含 v_1, 也不包含 v_i), 则 $\mathrm{sp}[S, v_i]$ 的含义是从 v_1 出发, 经过 S 中的顶点一次且仅一次, 最后到达 v_i 的最短路径 (这是动态规划表格叫作 sp (shortest path) 的原因) 的长度. 若顶点子集 S 包含 v_i, 则 $\mathrm{sp}[S, v_i]$ 无定义 (或定义为 ∞). 这解释了动态规划表的构成, 如表 1 所示.

为便于阅读, 将 $\mathrm{sp}[S, v_i]$ 记为 $\mathrm{sp}[v_1, S, v_i]$, 以突出表达从 v_1 出发的含义. 注意, 这只是一个符号记法, 并不改变动态规划表的结构. $\mathrm{sp}[v_1, S, v_i]$ ($v_1 \notin S$, $v_i \notin S$, 且 $v_1 \neq v_i$) 的递推公式为

$$\mathrm{sp}[v_1, S, v_i] = \begin{cases} \min_{v_j \in S} \left\{ \mathrm{sp}[v_1, S \setminus \{v_j\}, v_j] + w(v_j, v_i) \right\}, & S \neq \emptyset, \\ w(v_1, v_i), & S = \emptyset. \end{cases} \tag{1}$$

这个公式很容易理解. 当 S 为空集时, $\mathrm{sp}[v_1, S, v_i]$ 就是 v_1 和 v_i 之间的边上的长度 $w(v_1, v_i)$. 当 S 不为空集时, $\mathrm{sp}[v_1, S, v_i]$ 是取遍所有的 $v_j \in S$, $\mathrm{sp}[v_1, S \setminus \{v_j\}, v_j] + w(v_j, v_i)$ 的最小值, 即从 v_1 出发, 经过 $S \setminus \{v_j\}$ 中的顶点一次且仅一次, 然后到达 v_j, 再到达 v_i 的最短路的长度.

在表 1 中, 若顶点集 S 对应的行、顶点 v_i 对应的列处的符号为 "f", 表示需要计算 $\mathrm{sp}[v_1, S, v_i]$ 的值. 若该位置的符号为 "$\times$", 表示不需要计算 $\mathrm{sp}[v_1, S, v_i]$ 的值. 不需要计算是因为在 "$\times$" 位置处, 有 $v_i \in S$, 不符合 $\mathrm{sp}[v_1, S, v_i]$ 的定义.

对于表 1 所示的动态规划表 sp, 从上到下逐行、每行从左到右逐列计算每个需要计算的单元格即可. 由对 TSP 动态规划算法的分析可知, 完成表 1 的计算所花费的时间为 $O((n-1)(n-2)2^{n-3})$. 这是一个经典的分析, 具体细节可见参考文献 [1,2] 或运筹学、组合优化方面的著作. 这个分析在本证明中就省略了.

有了这些计算, 就能求出任何一个包含 v_1 的顶点子集 T 上的最优 TSP 回路. 令 $S = T \setminus \{v_1\}$, 这就是取得最小值

$$\min_{v_j \in S} \{\mathrm{sp}[v_1, S \setminus \{v_j\}, v_j] + w(v_j, v_1)\} \tag{2}$$

的顶点子集 T 中的顶点的 "走法" (即顶点的序列). 式 (2) 表明这个走法从顶点 v_1 出发, 访问 $S \setminus \{v_j\}$ (对某个 v_j) 中的所有顶点, 然后到达 v_j, 再到达 v_1, 因此是访问了 T 中每个顶点至少一次的一个圈. 由于这个圈是最短的, 因此它恰好访问了 T 中每个顶点一次且仅一次. 通过记录动态规划表的计算过程, 可找到长度为式 (2) 中最小值的圈, 即 T 上的最优 TSP 回路.

注意到在式 (2) 中, 有 $|S| \leqslant n-1$, 因此 $|S \setminus \{v_j\}| \leqslant n-2$. 这解释了为什么表 1 中的动态规划表 sp 的行对应的集合的大小是从 0 到 $n-2$.

现在考虑计算出所有包含 v_1 的顶点子集上的最优 TSP 回路所花费的时间. 任取这样的一个子集 T, 其上最优 TSP 回路的长度记为 $\mathrm{tsp}^*(T)$. 则 $\mathrm{tsp}^*(T)$ 的值即为 (2) 的值. 假设动态规划表 sp 已经计算完毕, 则计算式 (2) 需要花费 $|T|-1 \leqslant n-1$ 次基本操作 (加法和比较). 由于一共有 2^{n-1} 个不同的 T, 对所有的这样的子集完成 (2) 的计算所花费的时间为 $O((n-1)2^{n-1})$. 因此, 计算出所有包含 v_1 的顶点子集上的最优 TSP 回路所花费的总的时间 (即动态规划表 sp 的计算时间和所有子集的式 (2) 计算时间之和) 为

$$O\left((n-1)(n-2)2^{n-3} + (n-1)2^{n-1}\right) = O^*(2^n).$$

这里一个关键的观测是, 为计算包含 v_1 的不同的顶点子集对应的式 (2), 只需要使用同一个动态规划表 sp 即可. 换言之, 若有两个不同的顶点子集 T_1 和 T_2, 满足 $v_1 \in T_1$, $v_1 \in T_2$, 则计算 $\mathrm{tsp}^*(T_1)$ 和 $\mathrm{tsp}^*(T_2)$ 都使用到动态规划表 sp, 并不需要为 T_1 和 T_2 准备两份不同的动态规划表. 正是这个观测保证了本引理所证明的时间 (即 $O^*(2^n)$) 比 $O^*(4^n)$ 更快.

同样地, 在图上去掉顶点 v_1 之后, 图上还有 $n-1$ 个顶点. 计算出所有包含 v_2 的顶点子集 (显然不包含 v_1) 上的最优 TSP 回路所花费的时间为 $O^*(2^{n-1})$. 依次类推, 直到图上只剩下一个顶点, 就不需要进行计算了. 这整个计算过程所花费的时间为

$$O^*(2^2) + O^*(2^3) + \cdots + O^*(2^n) = O^*(2 \cdot 2^n) = O^*(2^n).$$

设 $S \subseteq V$ 是一个非空顶点子集. S 上的最优 TSP 回路记为 $\mathrm{TSP}^*(S)$, 其长度记为 $\mathrm{tsp}^*(S) = w(\mathrm{TSP}^*(S))$, 正如引理 2 的证明中使用的符号那样. 这里 $w(\mathrm{TSP}^*(S))$ 表示 $\mathrm{TSP}^*(S)$ 上各边的长度之和.

4 分支定界算法

分支定界算法本质上仍是一种枚举算法, 只不过是一种 “智能的” 枚举算法, 即在枚举的过程中, 若能够判断在某个子空间中的搜索不可能找到比当前解更好的解, 则直接摒弃对这个子空间的搜索, 这称为 “剪枝”. 对于 MMCCP 问题, 可按照分支定界算法的思想设计一个算法, 如算法 1 (即算法 branch) 所示.

首先对算法中的符号作简单说明. 假设 $\mathcal{C}$ 是一个圈覆盖. 在算法 branch 中, $\mathrm{val}(\mathcal{C})$ 表示圈覆盖 $\mathcal{C}$ 的解值, 即 $\mathcal{C}$ 中最大圈的长度. 若 $\mathcal{C}$ 不是一个有效的圈覆盖 (比如 $\mathcal{C}$ 为空集), 则 $\mathrm{val}(\mathcal{C})$ 的值为 ∞. 设 $S \subset V$ 是一个非空顶点集. 在算法 branch 中, $\mathrm{TSP}^*(S)$ 表示 S 上的最优 TSP 回路, $w(\mathrm{TSP}^*(S))$ 表示圈 $\mathrm{TSP}^*(S)$ 的长度 (圈的边上的权重之和). $\mathrm{TSP}^*(S)$ 在第 3 节中已经计算出了, 因此可以在算法 1 中直接使用. 换言之, 在执行算法 1 之前首先需要执行第 3 节中的预处理算法 (引理 2).

在算法 1 中, $\mathcal{C}_{\mathrm{OPT}}$ 表示当前已经找到的最好的解. $\mathcal{C}_{\mathrm{OPT}}$ 初始化为空 (算法 1 第 7 步). 当算法完成对整个解空间的搜索后, $\mathcal{C}_{\mathrm{OPT}}$ 就是问题的最优解了, 即算法 1 第 17 步返回的解.

算法 1 The branch-and-bound algorithm branch(V, k) for MMCCP

Input: A vertex set V, and a positive integer k;
Output: An optimal solution to MMCCP;

```
 1: if V = ∅ then
 2:     Return ∅;
 3: end if
 4: if k = 1 then
 5:     Return TSP*(V);
 6: end if
 7: C_OPT ← ∅;
 8: for each S ⊆ V, S ≠ ∅ do
 9:     if w(TSP*(S)) ⩾ val(C_OPT) then
10:         Continue;
11:     end if
12:     C' ← branch(V \ S, k − 1);
13:     if val(C') < val(C_OPT) then
14:         C_OPT ← {TSP*(S)} ∪ C';
15:     end if
16: end for
17: Return C_OPT;
```

最小最大圈覆盖问题的规模由两个参数确定: 顶点个数 n 和圈的数目 k. 我们不妨把问题的规模记为 (n, k). 算法 1 是一个递归算法, 它对顶点集 V 的解空间进行搜索时, 将 V 划分为两个子集 S 和 $V \setminus S$, 并枚举所有这样的划分 (算法 1 第 8~16 步). 这是算法 1 的分支规则. 在子集 S 上求最优 TSP 回路, 这样就用掉了 k 个圈中的一个圈. 在子集 $V \setminus S$ 上解允许圈数为 $k-1$ 的 MMCCP 问题. 设子集 S 包含的顶点为 s 个, 则在 $V \setminus S$ 上的 MMCCP 问题的规模为 $(n-s, k-1)$. 这样经过一次分支, 就把 MMCCP 问题的规模由 (n, k) 降为了 $(n-s, k-1)$.

在分支的过程中, 假设当前分支为 $(S, V \setminus S)$, 若判断到子集 S 上的最优 TSP 回路的长度大于当前已经找到的最好的解的解值 (算法 1 第 9 步), 则直接搜索下一个分支, 而放弃对当前分支 $(S, V \setminus S)$ 的解空间的搜索 (算法 1 第 10 步). 这是算法 1 所使用的剪枝规则.

表 2 MMCCP 问题第二阶段的动态规划算法所使用的动态规划表 dp

Table 2 Dynamic program table dp used in the second stage dynamic programming algorithm for MMCCP

	1	2	$\cdots$	k
$\emptyset$	$\mathrm{dp}[\emptyset,1]$	$\mathrm{dp}[\emptyset,2]$	$\cdots$	$\mathrm{dp}[\emptyset,k]$
$\{v_1\}$	$\mathrm{dp}[\{v_1\},1]$	$\mathrm{dp}[\{v_1\},2]$	$\cdots$	$\mathrm{dp}[\{v_1\},k]$
$\vdots$				
V	$\mathrm{dp}[V,1]$	$\mathrm{dp}[V,2]$	$\cdots$	$\mathrm{dp}[V,k]$

在最坏情形下, 算法 1 所枚举到的解的解值恰好一个比一个小, 这样算法 1 就会枚举所有可能的解. 因此, 算法 1 的最坏时间复杂度仍然是很高的. 但分支定界算法是一种实际可取的启发式方法, 其实际运行效果往往远比理论上的最坏时间复杂度好得多, 这是因为 "所枚举到的解的解值恰好一个比一个小" 这种事件几乎不可能发生. 在分支定界算法的枚举过程中, 许多的解可能被剪枝掉了. 越是较早发现了解值小的解, 被剪枝的解就越多, 算法的实际运行效果就越好.

5 动态规划算法

分支定界算法 branch (算法 1) 之所以耗费大量时间, 是因为在递归调用时做了大量重复计算, 即有大量的小规模的 MMCCP 实例被重复计算了很多次. 由此, 很容易想到使用动态规划的方法, 把计算过的实例的解存储起来, 在需要这个解时, 只需要查表即可, 就不必重复计算了.

首先介绍动态规划表的构造. 对顶点集 V 中的顶点编号为 $v_1, v_2, \ldots, v_n$. 任取 V 的一个子集 S, 定义 S 的标签为它所包含的顶点的编号从小到大排列的序列. 例如, 若 $S = \{v_1, v_2, v_6\}$, 则 S 的标签为 $(1, 2, 6)$. 有了标签的概念, 就可以把 V 的所有子集定一个序了, 比如按照标签从小到大排列, 这样的序称为 V 的子集的字典序.

MMCCP 的动态规划法所使用的动态规划表记为 dp, 如表 2 所示. 动态规划表 dp 的行对应按照字典序排列的 V 的子集. 动态规划表 dp 有 k 个列, 分别对应从 1 到 k 这 k 个整数.

在表 2 中, 动态规划表 dp 的第 S 行 (即集合 S 对应的行) 第 i 列为 $\mathrm{dp}[S,i]$, 其含义为用最多 i 个圈去覆盖 S 中所有的顶点, 这样的最小最大圈覆盖的最优解值. $\mathrm{dp}[S,i]$ 的递推公式为

$$\mathrm{dp}[S,i] = \begin{cases} \min_{T\subseteq S}\{\max\{\mathrm{dp}[S\setminus T, i-1], \mathrm{tsp}^*(T)\}\}, & S\neq\emptyset, i\geqslant 1, \\ \infty, & S\neq\emptyset, i=0, \\ 0, & S=\emptyset. \end{cases} \tag{3}$$

在式 (3) 中, $\mathrm{tsp}^*(T)$ 表示顶点集 T 上的最优 TSP 回路的长度. 该值已经在 MMCCP 问题实例的预处理阶段计算出, 参见第 3 节.

最小最大圈覆盖问题的动态规划算法, 就是对动态规划表 dp 从上到下逐行、从左到右逐列计算 dp 的每个单元格. 该动态规划算法如算法 2 所示. 这是 MMCCP 问题的第二阶段的动态规划算法. (第一阶段的动态规划算法是对 MMCCP 问题的实例进行预处理, 参见第 3 节及引理 2). 在算法 2 中, $\mathrm{tsp}^*(S)$ 表示顶点集 S 上的最优 TSP 回路的长度. 该值已经在 MMCCP 问题实例的预处理阶段计算出.

通过反向追踪 $\mathrm{dp}[V,k]$ 的计算过程, 可求出对 V 的划分. 这个划分的每一部分的最优 TSP 回路合在一起, 就是最小最大圈覆盖问题的最优解.

算法 2 The dynamic programming algorithm dp-calc(V, k) for MMCCP

Input: A vertex set V, and a positive integer k;
Output: The optimal value for MMCCP;
1: $\forall 1 \leqslant i \leqslant k$, $\mathrm{dp}[\emptyset, i] \leftarrow 0$;
2: **for** each $S \subseteq V$ in lexicographic order **do**
3: $\quad \mathrm{dp}[S, 1] \leftarrow \mathrm{tsp}^*(S)$;
4: $\quad$ **for** $i \leftarrow 2$ **to** k **do**
5: $\quad\quad$ Compute dp[S, i] by (3);
6: $\quad$ **end for**
7: **end for**
8: **Return** $\mathrm{dp}[V, k]$;

下面对算法 2 的时间复杂度进行分析.

引理3 算法 2 的时间复杂度为 $O^*(3^n)$.

证明 动态规划表 dp 的每一行对应一个子集, 大小为 j 的子集共有 $\binom{n}{j}$ 个. 任取一个大小为 j 的顶点集 S, 对于 S 所对应的 dp 中的行, 有 k 个单元格需要计算. 按照递推公式 (3) 计算每一个单元格 $\mathrm{dp}[S, i]$, 需要花费的时间为 $O(2^{|S|}) = O(2^j)$ (因为要枚举顶点集 S 的所有子集 T). 因此, 计算整个动态规划表 dp 所花费的时间为

$$\sum_{j=0}^{n} \binom{n}{j} \cdot k \cdot 2^j = k \sum_{j=0}^{n} \binom{n}{j} \cdot 2^j = k3^n,$$

其中最后一个等号成立是由于二项式定理 (在 $(x+y)^n$ 的展开式中令 $x=2$, $y=1$ 即得). 这表明算法 2 总的时间复杂度为 $O^*(3^n)$.

至此, 就可以证明本文的主要定理, 即定理 2.

定理2 最小最大圈覆盖问题可在 $O^*(3^n)$ 时间内求到最优解.

证明 MMCCP 问题完整的算法由第一阶段的预处理动态规划算法 (参见引理 2) 和第二阶段的动态规划算法 (即算法 2) 两部分构成. 引理 2 表明第一阶段的动态规划算法的时间复杂度为 $O^*(2^n)$. 引理 3 表明第二阶段的动态规划算法的时间复杂度为 $O^*(3^n)$. 因此, 求到最小最大圈覆盖问题的最优解所花费的总的时间为 $O^*(3^n)$.

6 结语

本文对最小最大圈覆盖问题给出了基于动态规划策略的首个非平凡精确算法, 时间复杂度为 $O^*(3^n)$. 这一算法比简单暴力搜索算法有了显著的改进. 另外, 本文还对 MMCCP 问题设计了分支定界算法, 给出了具体的分支规则和剪枝策略.

接下来的研究工作可从两个方面进行考虑. 首先是本文所提出的两阶段动态规划策略, 可以推广到其他版本的圈覆盖问题以及树覆盖问题中. 这将为若干问题给出精确求解算法. 其次是可以考虑设计 MMCCP 问题的更快的精确算法, 即尝试改进时间复杂度 $O^*(3^n)$.

参考文献

1 Bellman R. Dynamic programming treatment of the travelling salesman problem. J ACM, 1962, 9: 61–63
2 Held M, Karp R. A dynamic programming approach to sequencing problems. J SIAM, 1962, 10: 196–210
3 Novellani S. Models and algorithms for the optimization of real-world routing and logistics problems. 4OR-Q J Oper Res, 2016, 14: 331–332
4 Mousaei A, Taghaddos H, Tak A N, et al. Optimized mobile crane path planning in discretized polar space. J Constr Eng Manage, 2021, 147: 04021036
5 Baker C, Ramchurn S, Teacy L, et al. Planning search and rescue missions for UAV teams. In: Proceedings of the 22nd European Conference on Artificial Intelligence, Hague, 2017. 1777–1782
6 Erdelj M, Natalizio E, Chowdhury K R, et al. Help from the sky: leveraging UAVs for disaster management. IEEE Pervasive Comput, 2017, 16: 24–32
7 Xu W Z, Liang W F, Jia X H, et al. Maximizing sensor lifetime with the minimal service cost of a mobile charger in wireless sensor networks. IEEE Trans Mobile Comput, 2018, 17: 2564–2577
8 Xu W Z, Liang W F, Jia X H, et al. Minimizing the maximum charging delay of multiple mobile chargers under the multi-node energy charging scheme. IEEE Trans Mobile Comput, 2021, 20: 1846–1861
9 Xu Z, Xu D S, Zhu W B. Approximation results for a min-max location-routing problem. Discrete Appl Math, 2012, 160: 306–320
10 Khani M R, Salavatipour M R. Improved approximation algorithms for the min-max tree cover and bounded tree cover problems. Algorithmica, 2014, 69: 443–460
11 Jorati A. Approximation algorithms for some min-max vehicle routing problems. Dissertation for Master Degree. Edmonton: University of Alberta, 2013
12 Xu W Z, Liang W F, Lin X L. Approximation algorithms for min-max cycle cover problems. IEEE Trans Comput, 2015, 64: 600–613
13 Yu W, Liu Z H. Improved approximation algorithms for some min-max and minimum cycle cover problems. Theor Comput Sci, 2016, 654: 45–58
14 Yu W, Liu Z H. Better approximability results for min-max tree/cycle/path cover problems. J Comb Optim, 2019, 37: 563–578
15 Christofides N. Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem. Report 388, School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, 1976
16 Karlin A, Klein N, Gharan S O. A (slightly) improved approximation algorithm for metric TSP. In: Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC), Virtual Event, 2021. 32–45
17 Zhang Z F, Ma J X, Cui X. Genetic algorithm with three-dimensional population dominance strategy for university course timetabling problem. Int J Grid High Perform Comput, 2021, 13: 56–69
18 García-Martínez C, Cordón O, Herrera F. A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. Eur J Oper Res, 2007, 180: 116–148
19 Puris A, Bello R, Herrera F. Analysis of the efficacy of a two-stage methodology for ant colony optimization: case of study with TSP and QAP. Expert Syst Appl, 2010, 37: 5443–5453
20 Shi X H, Liang Y C, Lee H P, et al. Particle swarm optimization-based algorithms for TSP and generalized TSP. Inf Process Lett, 2007, 103: 169–176

Exact algorithms for the min-max cycle cover problem

Sen YUAN, Kaiqi CHEN, Jiangkun LI & Peng ZHANG*

School of Software, Shandong University, Jinan 250101*, China*
* Corresponding author. E-mail: algzhang@sdu.edu.cn

Abstract The min-max cycle cover problem is a generalization of the traveling salesman problem. Wireless sensor networks, UAV disaster rescue, and other fields all leverage the challenge. While approximation solutions for the min-max cycle cover issue receive a lot of attention, research on exact algorithms is sparse. Based on the combinatoric characteristics of the problem, we design the first exact algorithm for the min-max cycle cover problem using the dynamic programming strategy. We prove that the time complexity of our algorithm is $O^*(3^n)$. Our method for the min-max cycle cover problem consists of two stages. The first stage involves preprocessing the problem's input, and the second stage involves finding the best solution to the problem based on the first stage's results. Interestingly, the two stages are both based on the dynamic programming strategy. This is the main feature of our algorithm. The time complexity $O^*(3^n)$ of our algorithm is significantly better than that of the enumeration algorithm using brute-force search.

Keywords min-max cycle cover, dynamic programming, branch and bound, exact algorithm