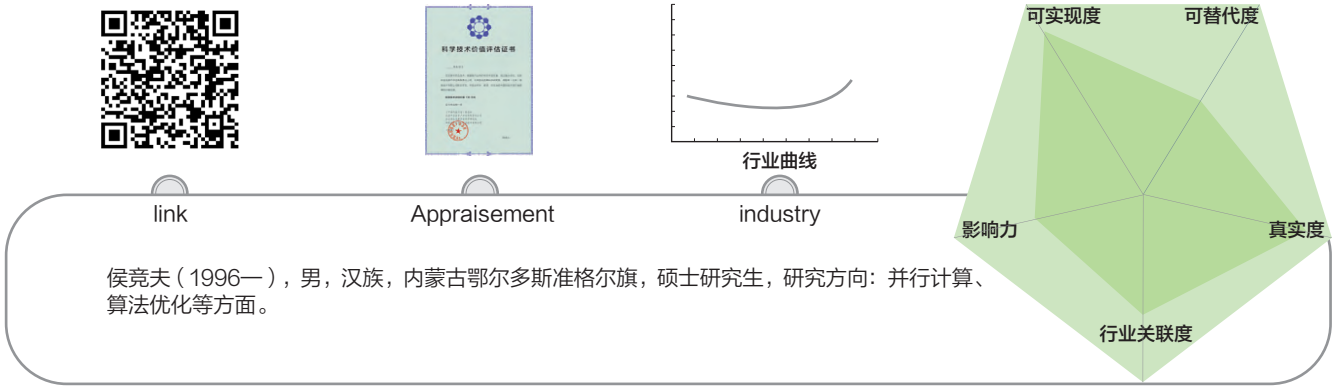
DOI：10.3969/j.issn.1001- 8972.2022.11.034



侯竞夫（1996—），男，汉族，内蒙古鄂尔多斯准格尔旗，硕士研究生，研究方向：并行计算、算法优化等方面。

基于 OpenMP 的 Dijkstra 算法并行优化研究

侯竞夫
University of Nottingham

1959 年，计算机科学家 E.W.Dijkstra 提出了 Dijkstra 算法。Dijkstra 算法是一种解决单源最短路径问题的贪心算法，其作用主要表现在解决有向图中的最短路径问题方面。但随着科学技术的发展，Dijkstra 算法的应用领域得到了极大的扩展，“最短路径”也被赋予了全新的含义，不再单单指空间中的最短距离，同样也可以用来研究各种时间、经济、能量消耗问题。但是，随着研究问题的复杂化以及所研究数据规模的逐渐扩大，传统的串行 Dijkstra 算法计算量大、时间复杂度较高的问题也逐渐体现出来，影响着 Dijkstra 算法在诸多问题中的表现。而并行化计算不仅在计算大规模数据时表现优异，也非常契合 Dijkstra 算法的算法思想。

1.Dijkstra 算法的基本特点以及优化方式

1.1 Dijkstra 算法的基本算法思想

传统串行 Dijkstra 算法采用广度优先搜索思想，它的主要特点是选定起始点后，一个点一个点地求取最短距离，通过邻点逐步扩展，不断更新，直至求出起始点到目标点的最短距离后才停止。

假定赋权图 G（V，E，W），V、E、W 分别为图的顶点集、图的边集和图的边权集。假定起点为 u0，设集合 $S \in V$，初始时集合 S 中仅包含起点 u0。对于任意一个属于顶点集 V-S 的顶点 u1，若已知起点 u0 至该顶点 u1 的最短距离，则将该顶点 u1 放入集合 S 中，并记录起点 u0 到该顶点 u1 的最短距离。

在集合 V-S 中找到距离起点 u0 最近的顶点，记为顶点 q，将顶点 q 加入集合 S 的同时将顶点 q 从集合 V-S 中移除。然后将顶点 q 视为新的起点，重复之前的步骤，直至目标点被加入集合 S 中。

我们可以推出 3 个基本概念：

（1）将 L 定义为起点 u0 到另一个顶点 u1 的最短路径权值，若顶点 u1 在第 r 步获得了标号 L，则称顶点 u1 在第 r 步获得永久性标号，r ≥ 0。

（2）假定 P 为已获得永久性标号的顶点的集合，对于任意顶点 $u1 \in P$，$u1$ 已获得永久性标号。设集合 T=V/P，则 T 中的元素为在第 r 步尚未获得永久性标号的顶点，T 称为临时性标号集合。

（3）将 l 定义为起点 u0 到顶点 u1 的最短路径权值上限。若顶点 u1 在第 r 步可以达到 l，则称 u1 在第 r 步获得了临时性标号，r ≥ 0。若起点 u0 与顶点 u1 无法通过集合 P 可达，则 l 为无穷。

1.2 Dijkstra 算法的优化方向分析

结合对有关文献进行调查可以发现，在选择永久性标号的过程中，如果在第 r 步时，有多个顶点已经获得最小临时性标号，那么应该同时赋予这些顶点永久性标号。这一操作不仅可以明显减少对临时性标号集合的遍历次数，达到缩短计算时间，提升计算效率的目的；同时也是并行化的理想对象。在进行优化的过程中，可以从这个角度进行算法优化，达到提升计算效率的目的。同时，结合 Dijkstra 算法的特点可以推断出，在 Dijkstra 算法标号的过程中，并行化 Dijkstra 算法的计算准确性和传统 Dijkstra 算法的准确性并无明显差异。所以在该并行化优化方法的运用过程中，因为对准确性并无太大影响，所以可以尽可能地减少永久性标号选择次数，进而缩短计算时间，提升算法运行效率。

1.3 并行多标号的 Dijkstra 算法

在进行程序计算研究的过程中，随着研究问题演变得越来越复杂，所研究内容的计算量越来越大，使用常规串行计算的方式完成计算已经很难满足当前的实际计算需求，往往用时较长，无法在较为理想的时间内得到运算结果。

在进行 Dijkstra 算法并行优化的过程中，可以充分运用

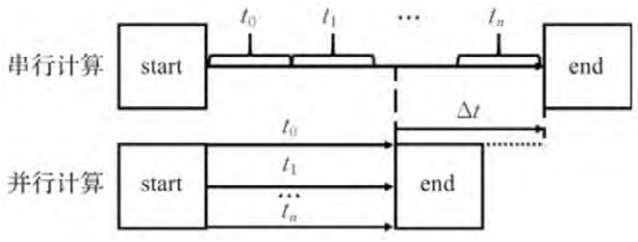


图 1 并行计算与串行计算特点示意图

分治法，即面对复杂问题时，并不直接求解，而是将该复杂问题等效转化为多个等价的较简单的子问题。而这些子问题则可以通过常规的方式进行运算与解决。在 Dijkstra 算法并行化优化的过程中，可以使用分治法，将原有问题转化为多个子问题求解，同时使用多个 CPU 线程分别处理不同的子问题，达到提升运算效率的目的。

图 1 简要地展示了串行算法和并行算法的主要差异：

图中 t_0 至 t_n 分别表示原问题分解后所得的子问题，图中的 Δt 则表示并行计算可以节约的时间。而且在实际运算的过程中，因为并行计算可以同时调用多个 CPU 线程进行计算，CPU 利用率更高，计算消耗时间更少，从而达到对整体处理性能进行提升的目的。

结合实际可以发现，若多个顶点同时拥有最短路径权值，在赋予永久性标号时，还需要改变其邻点，这增加了算法的计算量。而且这一过程也会使修改临时性标号变得更为复杂。但是在修改多个获得永久性标号的顶点的邻点以及临时性标号时，因为各步骤之间并不存在递推关系或依赖性，所以同样可以通过并行计算的方式进行优化，提高计算效率，极大地节省计算时间。按照并行计算的方式展开计算，可以同时展开多方面运算，在相同的时间下得到更多的计算结果，且对应的准确性较高，可以达到节约时间成本的目的。且在并行计算的过程中，可以在计算的同时进行并行对比，便于在最短的时间内获得最优解。

在对 Dijkstra 算法进行并行化优化时，资源争夺的情况时有发生。多个线程可能同时访问某个公共资源或者数据，造成数据混乱、结果错误等诸多情况，进而影响程序运行的效率及准确性，即所谓的非线程安全。在实际进行程序运算的过程中，为避免出现非线程安全的情况，需要采取有效的方式，确保公共资源中仅存在有一个线程。可以使用 OpenMP 自带的 critical 指令实现枷锁效果，使同一时间只允许一个线程操作指定代码；或者通过使用锁来实现互斥的效果；还可以使用原子操作来避免资源争夺的情况发生。三种方法的耗时比比例大约为 7：3.5：1，虽然原子操作速度最快、耗时最短，但原子操作所受限制比前两者更多，且可能出现自旋锁、死锁等问题。

2. 时间复杂度分析

因为并行算法的效率和实验设备的处理器个数和性能存在关联，较难计算其复杂度。所以可以使用时间复杂度来评估算法效率。本文通过比较理想情况下两种算法的时间复杂度，来展示并行多标号 Dijkstra 算法的优越性。

对于给定的赋权正则树，传统串行 Dijkstra 算法的时间复杂度为 $O(n^2)$，可使用优先队列的方法将时间复杂度降为 $O(n\times logn|m\times logn)$；或可采用斐波那契数列储存永久性标号，将时间复杂度进一步降为 $O(n\times logn|m)$。其中 n 表示顶点数，m 表示边数。

若实验设备可支持 k 条线程同时工作，则并行多标号 Dijkstra 算法的时间复杂度为 $O(h(n/k+logn))$。h 为所使用的赋权正则树的深度。

$$O(n^2) > O(n*logn+m*logn) > O(n*logn+m) > O(h(n/k+logn))$$

所以，多标号的并行 Dijkstra 算法可以有效降低算法时间复杂度。其原理在于并行多标号的 Dijkstra 算法减少了中间步骤，从而缩短了算法运行时间。

该算法的时间复杂程度在属于建立在理想状态下的算法，且在实际运算的过程中具备较高的准确性。结合实际运用可以发现，按照多标号的 Dijkstra 算法进行运算，在运算的过程中线程的有效切换需要耗费对应的时间成本，且在同一个操作平台下，将对应的时间成本定义为 T。则可以得出，在理想状态下，并行多标号的 Dijkstra 算法对用时间复杂度为 $O(h(n/k+logn))$，对应的现实状态下的时间复杂度和理想状态下的时间复杂度比值则为 $(h(n/k+lgn)+T)/(h(n/k+lgn)$。在该比值下，若 n 相对较小，则可以从理论上推断出，在 T 存在的情况下，此值所对应的结果大于 1。在优化提速和线程开销相互抵消的情况下，可以促使运算的效率进一步得到提升，缩短单次运算所运用的时间。当 n 区域无穷大时，则 $(h(n/k+lgn)+T)/(h(n/k+lgn)=1$，即可以发现时间复杂度和对应的算法速率表现为正相关的关系。

3. 空间复杂度分析

受当时的计算机硬件水平的限制，传统 Dijkstra 算法通常建立在时间换空间的算法基础上，即牺牲一定的时间效率来优化储存空间，其空间复杂度为 $O(n)$。而储存空间的大小和所使用的图的顶点个数及其出度密切相关。若所使用的图的顶点数为 n，边数为 m。则顶点的平均出边条数，即出度为 e=n/m，而顶点的平均出度可以体现出图的连通程度。一般情况下，e ∈［2，5］。若当前获得永久性标记的顶点数为 p，则下一步需要修改临时性标记的顶点数约为 p~$4p$。若采用链表结构储存，实际运用中并行多标号的 Dijkstra 的总存储空间通常可以控制在 100k 以内。可优化的潜力不大，而且随着计算机硬件技术的发展进步，空间储存问题现在已经不再是考虑的主要问题。

4. 总结

传统 Dijkstra 算法的建立在很大的程度上解决了对最短路径的探索问题，而并行多标号的 Dijkstra 算法则进一步提升了该方面算法的综合效率，促使最短路径可以按照更加高效快捷的方式选取，且具备较高的准确性。同时，基于并行计算的多标号 Dijkstra 算法的社会运用前景极为广泛，在各个行业均有较高的使用价值，对于推动行业的发展存在着重要的作用。按照在并行算法 Dijkstra 算法的作用下，可以进一步提升该算法在实际运用过程中的效率，在保障运算准确性的基础上，缩短运算时间，可以充分保障运算的效果。