Participant's Commentary: Finding Makespans Is NP-Complete

Clifford A. McCarthy
Dept. of Mathematics
University of Illinois
Urbana, IL 61801
mccarthy@math.uiuc.edu

We consider a special case of the general optimization problem of finding the makespan for an arbitrary network—the case in which the graph of the network is a tree. We show that this more limited problem is polynomial-time reducible to an NP-complete problem, the partition problem.

Let G be a tree representing a network, with weights $T_1, T_2, \ldots, T_k$ assigned to its k edges. Interpret each edge-weight as the time required for a file transfer between the two nodes. Require both that no node can be involved in more than one transfer at a time and that transfers be *atomic* (they cannot be interrupted and resumed later).

If we can solve the *decision problem*:

Can all the transfers be performed in time n or less?

then we can also solve the *optimization problem*:

What is the minimum time to perform all of the transfers?

Starting with $n = 1$, we simply solve the decision problem for each successive value of n until we get a "yes" answer. We are guaranteed to reach a "yes" answer, since any graph can complete its transfers in $\sum T_i$ units of time by doing them one after another.

Theorem: *The decision problem is NP-complete.*

Proof: The proof involves two parts [Manber 1989, 341–357]. We must show that

- the decision problem is in the class NP, meaning that we can check in polynomial time whether a proposed solution is in fact a valid solution; and
- some NP-complete problem is polynomial-time reducible to the decision problem, meaning that we can convert (in polynomial time) any instance

of the known NP-complete problem to an instance of the decision problem, such that the answer for the decision problem is positive if and only if the answer for the NP-complete problem is positive.

The first part of the proof is easy. Given an integer n and a proposed schedule for any graph, we can test in polynomial time whether it is a valid schedule requiring no more than time n, by checking each node to see if it is ever involved in two transfers, and checking if any transfers happen after time n. Hence, the graph scheduling problem (the decision version) is in the class NP.

For the second part, we use as our known NP-complete problem the *partition problem* [Garey and Johnson 1979, 60–62]:

Given integers $a_1, a_2, \ldots, a_m$, is there a partition of these integers into sets A and B so that

$$\sum_{a_i \in A} a_i = \sum_{a_i \in B} a_i?$$

We first exhibit how to convert an instance of the partition problem into an instance of the decision problem. Let an instance of the partition problem be given, with integers $a_1, a_2, \ldots, a_m$. If $\sum a_i$ is odd, then there cannot be a partition into two sets with equal sum. So suppose $\sum a_i$ is even, with $\sum a_i = 2n$. Construct a tree with the structure and edge weights shown in **Figure 1**. Given an instance of the partition problem, we can certainly produce a description of the corresponding tree in polynomial time.

Finally, we show that *a partition exists if and only if the transfers on the tree can be performed in* $(2n+1)$ *units of time.* In other words, the partition problem is polynomial-time reducible to the decision problem, so the decision problem is NP-complete.

($\Longleftarrow$) Suppose that the transfers indicated in **Figure 1** can be performed in $(2n+1)$ units of time. This implies that the two nodes with edge weights n and $(n+1)$ spend the entire time engaged in transfers. Each can either perform the n-unit transfer followed immediately by the $(n+1)$, or the $(n+1)$ followed by the n.

If they both perform $(n+1)$-unit transfers first, then the two n-unit transfers will not be possible at the same time, because they are incident upon a common node. Hence, at least $(n+1)+n+n = 3n+1$ units of time are required, which contradicts our supposition that the transfers can be performed in $(2n+1)$ units of time. Similarly, if both n-unit transfers are performed first, the $(n+1)$-unit transfers will not be possible at the same time, and $n+(n+1)+(n+1) = 3n+2$ units of time will be necessary.

Hence, one of the $(n+1)$'s must be first, and the other must be last. This means that the only time that the 1-unit transfer can be performed is in the unit of free time between the two n's. During that transfer, the node connecting the a_i's is occupied (from time n to time $(n+1)$, assuming that the clock starts at time 0). Since $\sum a_i = 2n$, this node must also be

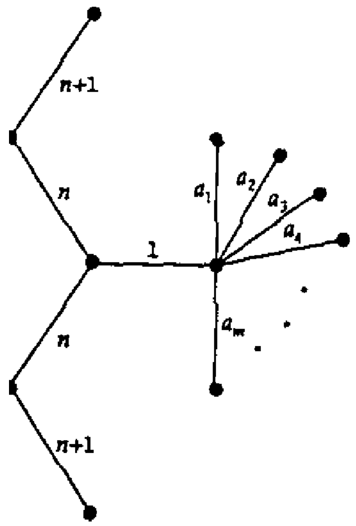

Figure 1. The tree representing the network of transfers.

continuously engaged in transfers. So the durations of the transfers that it performs before handling the 1-unit transfer must sum to n, and similarly for those performed after the 1-unit transfer.

Let A and B be the sets of the durations for the transfers handled in the two halves. Then A and B constitute a partition of $a_1, a_2, \ldots, a_m$, and

$$\sum_{a_i \in A} a_i = n = \sum_{a_i \in B} a_i.$$

If the transfers in the tree can be performed in $2n+1$ units of time, there is a partition of $a_1, a_2, \ldots, a_m$ such that $\sum_{a_i \in A} a_i = \sum_{a_i \in B} b_i$.

($\Longrightarrow$) Now suppose that there is such a partition into sets A and B. As before, we can run the two $(n+1)$-unit transfers, the two n-unit transfers, and the 1-unit transfer in a total of $(2n+1)$ units of time, if the 1-unit transfer is run at the halfway point. The node joining the a_i's is then available for n units of time before this transfer and for n units of time after this transfer. We can handle the transfers corresponding to the elements of A in the first n-unit interval, and those corresponding to B in the last n-unit interval. So, if there is a partition of $a_1, a_2, \ldots, a_m$ into A and B such that $\sum_{a_i \in A} a_i = \sum_{a_i \in B} b_i$, then the transfers in the corresponding tree can be performed in $2n+1$ units of time.

Hence, the desired partition exists if and only if the corresponding tree can be executed in $(2n+1)$ units of time. $\square$

References

Garey, M.R. and D.S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco, CA: W.H. Freeman.

Manber, Udi. 1989. *Introduction to Algorithms: A Creative Approach.* Reading, MA: Addison-Wesley.

About the Author

Clifford McCarthy completed a mathematics B.S. in 1994 at Harvey Mudd College and is continuing in mathematics as a graduate student at the University of Illinois. His team's entry in the MCM, with fellow students Brian Diggs and Andrew M. Ross (advisor: David Bosley), was judged Meritorious. The proof in this commentary is his own; it constituted an appendix to their entry.