Directed File Transfer Scheduling

Weizhen Mao
Department of Computer Science
The College of William and Mary
Williamsburg, Virginia 23187-8795
wm@cs.wm.edu

Abstract

The file transfer scheduling problem was introduced and studied by Coffman, Garey, Johnson, and LaPaugh. We extend their model to directed networks and study the complexity of the problem under different conditions. We discover that even though the general problem is still NP-complete, adding directions in the file transfer graph makes the problem easy in the sense that the problem in some special cases is solvable in polynomial time while its undirected counterpart was proved NP-complete.

1 Introduction

The problem of scheduling file transfers in a network so as to minimize the makespan of the schedule was first introduced by Coffman *et al.* [2]. In their model, an instance of the problem consists of a weighted undirected multigraph $G = (V, E)$, called a file transfer graph. The vertices of G correspond to the nodes in the network, and the edges correspond to the files to be transferred between the nodes. Each vertex v in V is labeled with a positive integer $p(v)$, which denotes the maximum number of file transfers that v can engage in simultaneously. Each edge e in E is labeled with a positive real $t(e)$, which denotes the amount of time needed to transfer that file. Under the assumptions that each file is transferred directly between the vertices that are its endpoints, i.e. forwarding is not allowed, and that once the transfer of a file begins, it continues until it completes, i.e. preemption is not allowed, we are asked to schedule the file transfers in G so that the makespan, which is the time interval between the beginning of the first transfer and the completion of the last transfer, is minimized. This problem is called File Transfer Scheduling (FTS).

In this paper, we extend their model to directed file transfer graphs. Let $G = (V, A)$ be a digraph with multi-arcs allowed. For each vertex v in V, $s(v)$ represents the maximum number of file transfers that v can engage in as a sender, and $r(v)$ represents the maximum number of file transfers that v can engage in as a receiver. For each arc a in A, denoted by (u, v) if it goes from vertex u to vertex v, $t(a)$ represents the amount of time needed to transfer the file from the source u to the destination v. We also assume that forwarding and preemption are not allowed. The Directed File Transfer Scheduling ($DFTS$) is to find a schedule of the file transfers such that the sending and receiving constraints are respected and the makespan is minimized.

We organize this paper as follows. In Section 2, we will present a polynomial-time algorithm for the problem when all file transfers require the equal amount of time. In Section 3, we will study the case when the file transfer time is arbitrary. And finally, we make the conclusion in Section 4. Without confusion, we shall also use FTS and $DFTS$ to denote the corresponding decision problems.

2 Equal file transfer time

In this section, we assume that all files in the file transfer graph require the same amount of time to be transferred from their sources to their destinations. Without loss of generality, we assume $t(a) = 1, \forall a \in A$. We define the following graph coloring problem: Given a digraph $G = (V, A)$, with each vertex v being weighted by two posi-

tive integers $s(v)$ and $r(v)$. We are asked to use the minimum number of colors to color the arcs in G such that for any vertex v, at most $s(v)$ outgoing arcs have the same color and at most $r(v)$ incoming arcs have the same color. It is obvious that there is a direct reduction from $DFTS$ with equal file transfer time to this problem, which indicates that if there is a polynomial-time algorithm for the graph coloring problem, then there is a polynomial-time algorithm for $DFTS$ with equal file transfer time.

Lemma

Given a digraph $G = (V, A)$. For any $v \in V$, let $in(v)$ be the number of arcs that end at v, and $out(v)$ be the number of arcs that start at v. Let $D(G) = \max_{v \in V}\{\lceil in(v)/r(v) \rceil, \lceil out(v)/s(v) \rceil\}$. Then the minimum number of colors needed to color G is $D(G)$.

Proof Mao [4] has proved the lemma for $s(v) = r(v) = 1$. We are going to generalize the proof to arbitrary $s(v)$ and $r(v)$. It is clear that in order to color G we have to use at least $D(G)$ colors. To prove the lemma, we need to show that $D(G)$ colors are enough, i.e. G is $D(G)$-colorable. We prove this by induction on the number of arcs. When $|A| = 1$, $D(G) = 1$. G is obviously 1-colorable. Assume that G is $D(G)$-colorable when $|A| = n - 1$.

Suppose $|A| = n$. Define G' to be G with one arc removed. Without loss of generality, we assume the arc is $(1, 2)$. Therefore, $G' = (V, A')$, where $A' = A - \{(1, 2)\}$. Because $|A'| = n - 1$, G' is $D(G')$-colorable by inductive hypothesis. After we color G' using $D(G')$ colors, $(1, 2)$ is the only uncolored arc in G. We will study the following cases and show that there is always a way to color $(1, 2)$ such that only $D(G)$ colors are used for G.

Case 1. If $D(G) = D(G') + 1$, then we can color $(1, 2)$ with a new color. Therefore, G is $D(G)$-colorable.

Case 2. If $D(G) = D(G')$, we need to prove that $(1, 2)$ can be colored without using a new color. Let C be the set of colors used in G'. Let C_{2k} be the set of colors used $r(2k)$ times for the arcs coming in to vertex $2k$ and C_{2k+1} be the set of colors used $s(2k + 1)$ times for the arcs going out from vertex $2k + 1$. Consider the following coloring procedure.

Step i. $C_1 \neq C$ and $C_2 \neq C$, otherwise case 1 happens. If $(C - C_1) \cap (C - C_2) \neq \phi$, then there exists some color in C, but not in C_1 and C_2, which can be used to color $(1, 2)$. Therefore G is $D(G)$-colorable. If $(C - C_1) \cap (C - C_2) = \phi$, then there exists a color α, not in C_1 (since $C_1 \neq C$), but in C_2 (otherwise $(C - C_1) \cap (C - C_2) \neq \phi$). Let $(3, 2)$ be the arc which comes in to vertex 2 and is colored with α. Uncolor $(3, 2)$, and color $(1, 2)$ with α. The situation is now as depicted in Figure 1. We need to modify C_1 such that it includes α if α is used $s(1)$ times at vertex 1, and modify C_3 such that it excludes α. It is clear that C_2 is unchanged.

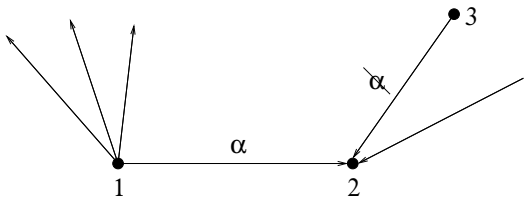

Figure 1

Step ii. $C_2 \neq C$ (since C_2 is unchanged in step i) and $C_3 \neq C$ (since $\alpha \notin C_3$). If $(C - C_2) \cap (C - C_3) \neq \phi$, then there exists some color in C, but not in C_2 and C_3, which can be used to color $(3, 2)$. Therefore, G is $D(G)$-colorable. If $(C - C_2) \cap (C - C_3) = \phi$, then there exists a color β, not in C_2 (since $C_2 \neq C$), but in C_3 (otherwise $(C - C_2) \cap (C - C_3) \neq \phi$), and $\beta \neq \alpha$ (since $\beta \in C_3$, $\alpha \notin C_3$). Let $(3, 4)$ be the arc which goes out from vertex 3 and is colored with β. Uncolor $(3, 4)$, and color $(3, 2)$ with β. See Figure 2 for this situation. Before we go to next step, modify C_2 such that it includes β if β is used $r(2)$ times at vertex 2, and modify C_4 such that it excludes β. C_3 is unchanged in this step.

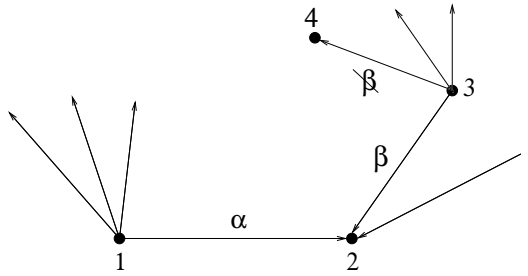

Figure 2

Step iii. Similar to step i, if $(C - C_3) \cap (C - C_4) \neq \phi$, then G is $D(G)$-colorable. If $(C - C_3) \cap (C - C_4) = \phi$, then $\alpha \notin C_3$, but $\alpha \in C_4$. Let $(5, 4)$ be the arc which comes in to vertex 4 and is colored with α. Uncolor $(5, 4)$, and color $(3, 4)$ with α. Finally, modify C_3 and C_5.

Step iv. Similar to step ii, if $(C - C_4) \cap (C - C_5) \neq \phi$, then G is $D(G)$-colorable. If $(C - C_4) \cap (C - C_5) =$

ϕ, then $\beta \notin C_4$, but $\beta \in C_5$. Let $(5,6)$ be the arc which goes out from vertex 5 and is colored with β. Uncolor $(5,6)$, and color $(5,4)$ with β. Finally, modify C_4 and C_6.

Repeat the procedure described in step iii and step iv and in each step always uncolor the arc that has never been uncolored before, until either G is $D(G)$-colorable due to the fact that the two endpoints of the uncolored arc have a common usable color or all the available arcs have been uncolored before. We claim that in the second case G is still $D(G)$-colorable.

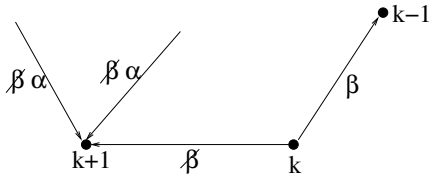

Figure 3

Without loss of generality, we assume as shown in Figure 3 that we come to a point to color $(k, k+1)$ which was colored with β and was later uncolored because β was given to $(k, k-1)$. Define A_α to be the set of the arcs that end at $k+1$ and are colored with α. According to the coloring procedure, we intend to uncolor one arc in A_α, and color $(k, k+1)$ with α. Since all arcs in A_α have been uncolored before, the old color of them must be β. It is clear that the old color of $(k, k+1)$ is also β. So we have $r(k+1) \geq |A_\alpha|+1$. Therefore, $\alpha \notin C_{k+1}$, and $\alpha \notin C_k$. We can color $(k, k+1)$ with α. This completes the proof. □

Theorem 1

$DFTS$ with equal file transfer time is solvable in time $O(|A|^2)$.

Proof First, let us consider the corresponding graph coloring problem. Define $G_1 = (V, A_1)$, where $A_1 = \{(a,b)\}$ for some (a,b) in A, and $G_i = (V, A_i)$, where $A_i = A_{i-1} \cup \{(a,b)\}$ for some (a,b) in A but not in A_{i-1}. Obviously, $G_{|A|} = G$. Using the method described in the proof of the lemma, we can color G_1, G_2, ..., $G_{|A|}$ in the order given. Assume $T(n)$ is the time required to color a graph with n arcs. We have $T(1) = 1$, and $T(n) = T(n-1) + n$. So $T(n) = O(n^2)$. Therefore, any digraph $G = (V, A)$ can be colored in time $O(|A|^2)$.

In $DFTS$, if G is the file transfer graph, then the minimum makespan is $D(G)$, and the schedule can be constructed in time $O(|A|^2)$. □

According to Coffman *et al.* [2] and later Choi *et al.* [1], FTS with equal file transfer time is NP-complete even though it becomes polynomial when the file transfer graph is a bipartite graph, tree, path or cycle. From the above discussion, we find that $DFTS$ with equal file transfer time is solvable in polynomial-time for arbitrary directed file transfer graphs. Intuitively, this is because that the direction constraint imposed on the graph and the relaxed constraint of dual vertex capacities eliminate many of the possible cases that need to be considered in the exhaustive search of the optimal schedule, which as a result changes the problem from NP-complete to polynomial.

3 Arbitrary file transfer time

Let us consider $DFTS$ with arbitrary file transfer time. We use the approach similar to that in [2] to discuss the complexity of the problem for different graph structures, such as cycle, path, tree, bipartite graph, and general graph.

Theorem 2

$DFTS$ with arbitrary file transfer time is solvable in time $O(|A|)$ if the file transfer graph is a directed cycle, and either $s(v) = r(v) = 1$ for any $v \in V$ or no multi-arcs are present.

Proof Let $V = \{1, 2, \ldots, n\}$ and $A = \{(1,2), (2,3), \ldots, (n-1,n), (n,1)\}$. If $s(v) = r(v) = 1$ for any $v \in V$, and multi-arcs may present in G, the files between any two adjacent vertices in the cycle have to be transferred sequentially, and files not sharing sources and destinations can be transferred simultaneously. For $(u,v) \in A$, let $T(u,v)$ be the sum of the transfer time of files from u to v. Then, the minimum makespan is $\max_{(u,v)\in A}\{T(u,v)\}$. If $s(v)$ and $r(v)$ are arbitrary positive integers, and multi-arcs are not allowed, then the minimum makespan is $\max_{a\in A}\{t(a)\}$. In both cases, the schedule can be easily constructed in time $O(|A|)$. □

Corollary 2.1

$DFTS$ with arbitrary file transfer time is solvable in time $O(|A|)$ if the file transfer graph is a directed path, and either $s(v) = r(v) = 1$ for any $v \in V$ or no multi-arcs are present.

Proof If $s(v) = r(v) = 1$ for any $v \in V$, the minimum makespan is obviously $\max_{(u,v)\in A}\{T(u,v)\}$. If $s(v)$ and $r(v)$ are arbitrary positive integers, and

multi-arcs are not allowed, the minimum makespan is $\max_{a \in A}\{t(a)\}$. □

Theorem 3

DFTS with arbitrary file transfer time is NP-complete if the file transfer graph is a directed cycle with multi-arcs and arbitrary $s(v)$ and $r(v)$.

Proof Consider the NP-complete Multiprocessor Scheduling (MS) [3], in which we are given n independent tasks with processing time $p_1, \ldots, p_n$, respectively, and are asked to schedule the tasks on $m(\geq 2)$ identical processors so that the makespan of the schedule is minimized.

We can define a polynomial transformation from MS to $DFTS$ as follows. Let $V = \{1, 2, \ldots, k\}$ and $(i, i{+}1) \in A$ with $t(i, i{+}1) = 1$ for $i = 1, 2, \ldots, k{-}1$. From k to 1, add an arc for each task in MS and label the arc with the processing time of the task. Let $r(1) = m$ and $s(k) = m$. Files $(1, 2), \ldots, (k - 1, k)$ can be transferred simultaneously, and at any time m of the n files from k to 1 can be transferred. It is not hard to see that MS has a schedule with makespan B or less if and only if $DFTS$ of the graph in Figure 4 has a schedule with makespan B or less.

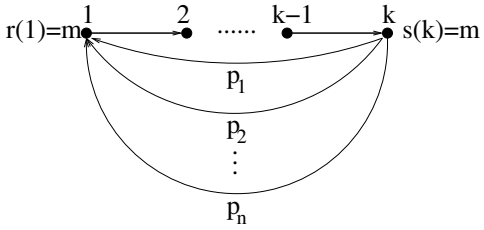

Figure 4

Since $DFTS$ is in NP and there is a polynomial transformation from an NP-complete problem to it, $DFTS$ in this special case is NP-complete. □

Corollary 3.1

DFTS with arbitrary file transfer time is NP-complete if the file transfer graph is a directed path with multi-arcs and arbitrary $s(v)$ and $r(v)$.

Proof Similar to the proof of Theorem 3 except that the file transfer graph is the graph in Figure 4 with the arcs $(1, 2), (2, 3), \ldots, (k - 1, k)$ being removed. □

Theorem 4

DFTS with arbitrary file transfer time is solvable in time $O(|A|)$ if the file transfer graph is a directed tree and $s(v) = r(v) = 1$ for any $v \in V$.

Proof Without loss of generality, assume that the directions of arcs are from parents to children. Files sharing the source have to be transferred sequentially, and files not sharing the source can be transferred simultaneously. Let $T'(v)$ be the sum of the transfer time of files to be transferred from v, then the minimum makespan is $\max_{v \in V}\{T'(v)\}$. □

Theorem 5

DFTS is NP-complete if the file transfer graph is a directed tree with arbitrary $s(v)$ and $r(v)$.

Proof Consider the transformation from MS. Assume that n tasks have to be scheduled on $m(\geq 2)$ processors. Define $G = (\{1, \ldots, n+1\}, \{(n+1, i) : i = 1, \ldots, n\})$, where $t(n + 1, i) = p_i$ for $i = 1, \ldots, n$, and let $s(n + 1) = m$. Since at any time only m of n files can be transferred out from $n+1$, MS has a schedule with makespan B or less if and only if $DFTS$ of the graph in Figure 5 has a schedule with makespan B or less. □

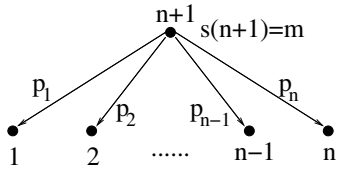

Figure 5

Theorem 6

DFTS with arbitrary file transfer time is NP-complete if the file transfer graph is a directed bipartite graph.

Proof According to Coffman *et al.* [2], FTS of a bipartite graph with single ports ($p(v) = 1, \forall v$), single edges, and arbitrary file transfer time is NP-complete. Given a bipartite graph $G = (V_1 \cup V_2, E)$, where each edge has one vertex in V_1 and one vertex in V_2, consider the corresponding directed bipartite graph G' by adding direction from u to v if $(u, v) \in E$, and $u \in V_1$, $v \in V_2$. Assume that in G' $s(v) = r(v) = 1$ for any $v \in V_1 \cup V_2$. We notice that in the schedule for G' when file (u, v) is being transferred no file from source u or to destination v can be transferred, which happens to be the exactly same case in the schedule for G. Therefore, G has a schedule with makespan B or less if and only if G' has a schedule with makespan B or less. □

Corollary 6.1

DFTS with arbitrary file transfer time is NP-complete.

To summarize, Table 1 and Table 2 show the complexity results for FTS [2] and $DFTS$, respec-

tively, when the file transfer time is arbitrary. To keep the tables small, we use the following abbreviations: G =general graph or digraph, B =bipartite graph or digraph, T =tree, P =path, E =even-length cycle, O =odd-length cycle, and finally, "Single" means that the file transfer graph has only single edges or arcs, and "Multiple" means that multi-edges or multi-arcs are allowed.

	$p(v)=1$		Arbitrary $p(v)$	
	Single	Multiple	Single	Multiple
G	NPC	NPC	NPC	NPC
B	NPC	NPC	NPC	NPC
T	NPC	NPC	NPC	NPC
P	$O(\|E\|)$	$O(\|E\|)$	$O(\|E\|)$	NPC
E	$O(\|E\|)$	$O(\|E\|)$	$O(\|E\|)$	NPC
O	$O(\|E\|)$	NPC	$O(\|E\|)$	NPC

Table 1

	$s(v)=r(v)=1$		Arbitrary $s(v), r(v)$	
	Singles	Multiple	Single	Multiple
G	NPC	NPC	NPC	NPC
B	NPC	NPC	NPC	NPC
T	$O(\|A\|)$	$O(\|A\|)$	NPC	NPC
P	$O(\|A\|)$	$O(\|A\|)$	$O(\|A\|)$	NPC
O	$O(\|A\|)$	$O(\|A\|)$	$O(\|A\|)$	NPC
E	$O(\|A\|)$	$O(\|A\|)$	$O(\|A\|)$	NPC

Table 2

4 Conclusion

In this paper, we have studied the complexity of the Directed File Transfer Scheduling under different conditions. We have proved that if all files have the same transfer time the problem can be reduced to a graph coloring problem which has a polynomial solution, and that if files have arbitrary transfer time the problem is NP-complete for general digraphs, bipartite digraphs, directed trees with arbitrary $s(v)$ and $r(v)$, directed paths or cycles with multi-arcs and arbitrary $s(v)$ and $r(v)$. In comparison with the results of undirected File Transfer Scheduling, we have found that adding directions to the file transfer graph sometimes makes the problem easy.

As its undirected counterpart, there are a wide variety of directions for future research in Directed File Transfer Scheduling. Coffman *et al.* [2] proved for FTS that the List Scheduling (LS) heuristic gives a schedule of makespan at most 3 times that of the optimal schedule and that Decreasing List Scheduling (DLS) heuristic gives a schedule of makespan at most 2.5 times that of the optimal schedule. Within our model, an interesting question we see is to analyze the worst-case performance bounds of LS and DLS when they are applied to $DFTS$ with arbitrary file transfer time for general digraphs and see whether they can achieve better bounds on $DFTS$ than on FTS. Whitehead [5] studied the complexity issue of FTS with forwarding. An important extension of our model is to replace the $s(v)$ and $r(v)$ constraints with a network graph which contains possible routes from sources to destinations via intermediate nodes. In this modified model, in order for a file in the file transfer graph to be transferred from u to v, we must find a path from u to v in the network graph for it.

Acknowledgement

I wish to thank Rahul Simha for bringing the original file transfer scheduling problem to my attention, Adam Rifkin for helping me with the LaTeX format, and the three anonymous reviewers for their valuable suggestions.

References

[1] H.A. Choi and S.L. Hakimi, *Scheduling File Transfers for Trees and Odd Cycles*, SIAM J. on Comput., 16 (1987), pp. 162–168.

[2] E.G. Coffman, Jr., M.R. Garey, D.S. Johnson and A.S. LaPaugh, *Scheduling File Transfer*, SIAM J. on Comput., 14 (1985), pp. 744–780.

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

[4] W. Mao, *Arc-Coloring of a Digraph is Polynomial*, Technical Report, Department of Computer Science, College of William and Mary, 1992.

[5] J. Whitehead, *The Complexity of File Transfer Scheduling With Forwarding*, SIAM J. on Comput., 19 (1990), pp. 222-245.