Scheduling Loosely Connected Task Graphs

Abhiram G. Ranade

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Powai, Mumbai 400076
Email: ranade@cse.iitb.ac.in

Abstract

We present a polynomial time algorithm for precedence constrained scheduling problems in which the task graph can be partitioned into large disjoint parts by removing edges with high *float*, where the float of an edge is defined as the difference between the length of the longest path in the graph and the length of the longest path containing the edge. Our algorithm guarantees schedules within a factor 1.875 of the optimal independent of the number of processors. The best known factor for this problem and in general is $2 - \frac{2}{p}$, where p is the number of processors, due to Coffman-Graham. Our algorithm is unusual and considerably different from that of Coffman-Graham and other algorithms in the literature.

Key words: precedence constrained scheduling, approximation algorithm

1 Introduction

Precedence constrained scheduling is a classical NP-complete problem. It arises in several applications such as project management and parallel computing.

In this paper we consider the simplest variant. In this we are given a directed acyclic graph G, a number p of processors, and a deadline d which is an integer. The vertices in the graph represent unit time tasks, and the arcs indicate precedence: if there is an arc from u to v, then task u must be executed before task v. The problem is to decide if the given set of tasks can be executed in d time steps using p processors, with each processor being able to execute only one task during any time step. More formally, a length d *schedule* for the graph is an assignment of an integer (time slot) $T(v)$ to each vertex v such that (i) $1 \leq T(v) \leq d$, (ii) if u is a predecessor of v in the graph then $T(u) < T(v)$, and (iii) For all i we have $|\{v : T(v) = i\}| \leq p$.

In this paper, we give approximation algorithms for some special cases of the problem. We believe that these special cases are likely to be common in practice. Further, our algorithmic ideas are substantially different from those in the literature.

1.1 Previous work

The problem of determining whether length d schedules exist is known to be NP-complete [7]. Approximation algorithms (those that find schedules which are no longer than a factor α of the optimal schedule length) have also been studied. $2 - \frac{1}{p}$ approximation is easy – any algorithm that will not idle processors while work is available will achieve this. A remarkable algorithm, due to Coffman and Graham [3] was shown to give $2 - \frac{2}{p}$ approximation by Lam and Sethi [6] [1] .

It is also known [7] that better than $\frac{4}{3}$ factor approximation is not possible unless P=NP. However, getting a $2 - \epsilon$ factor approximation algorithms for any nonzero constant ϵ has remained a longstanding open problem.

Two special cases have also been studied in the literature. The first is the case in which the graph G is a rooted tree, for which Hu [5] gave a polynomial time algorithm for computing an optimal schedule. The second special case studied is that of $p = 2$. For this Fujii, Kasami and Ninomiya [4] gave an algorithm that constructed optimal schedules.

Remarkably enough, the Coffman-Graham [3] algorithm subsumes both the special case results! In fact, Coffman and Graham originally proposed their algorithm to solve the 2 processor case; and since it is in fact a refinement of Hu's algorithm, it also optimally solves the case when G is a rooted tree. It is fair to say that even 30 years after it was proposed, the Coffman-Graham algorithm remains the last word on this problem. Neither has its approximation factor been improved, nor have better algorithms appeared even for special cases!

On the other hand, it is clear that to get a $2 - \epsilon$ approximation, an approach quite different from that of Coffman-Graham/Hu will be needed. These algorithms are *level-by-level* algorithms, i.e., they assign to each vertex a *level* which is simply the length of the longest path originating at the vertex. Vertices in higher levels are given higher priority for scheduling purposes [2] . It has been shown that there exist graphs for which no level-by-level algorithm will give better than $2 - \frac{2}{\sqrt{p}}$ factor approximation [2].

[1] Also see [1] which corrects an error in [6].

[2] Discussed at length later.

So several natural questions arise. What algorithmic ideas can get us to a $2-\epsilon$ factor approximation?[3] Are there other interesting classes of graphs for which optimal or approximately optimal schedules can be constructed in polynomial time?

1.2 Main Result

Our main result is a polynomial time algorithm that gives schedules that are within a factor 1.875 of the optimal provided the graph to be scheduled can be partitioned into large disjoint parts by removing edges with *high float*, defined shortly. We feel that such graphs are likely to arise in practice. For example, a company may be handling several independent projects over which it needs to allocate manpower. In this case, the task graph representing the work needed to be done by the company will simply be a disjoint union of several small task graphs (i.e., there is no need to even remove any edges.).

We need some definitions to state our result precisely. In this paper, we will use **length** of a (directed) path to mean the number of vertices on it. The **height** of a directed acyclic graph (DAG) denotes the length of the longest path in it. The **float** of an edge (vertex) is the difference between the height of the DAG and the length of the longest path containing the edge (vertex). The notion of vertex floats is standard in project management literature. An n vertex DAG G is said to be **α-loosely connected** if on removing all edges of float at least α it splits into disjoint DAGs G_1, G_2 both having at least $\frac{n}{4}$ vertices.

Theorem 1 *Let $G = (V, E)$ be an n vertex DAG of height H, and p the number of processors. Let G be $\frac{1}{8}\min\left(H, \frac{n}{p}\right)$-loosely connected. Then there exists a polynomial time algorithm to schedule G that generates a schedule of length no larger than 1.875 times the optimal.*

The algorithm is *not* level-by-level. This is to be expected, since the family of $\frac{1}{8}\min\left(H, \frac{n}{p}\right)$-loosely connected graphs includes the instances which are shown difficult to schedule using level-by-level algorithms in [2]. Instead, our algorithm works by either (i) overlapping regions of one part in which there is substantial parallel work with regions of other parts which are mostly sequential, or (ii) by showing that if such overlapping is not possible, then even the optimal schedule will have to be long. We feel that this general structure will be inevitable for devising an approximation algorithm for the general case. Finally, we feel that our result may also lead to a divide and conquer algorithm

[3] While Fujii-Kasami-Ninomiya's algorithm is quite different from that of Hu/Coffman-Graham, it appears that the central idea in it (based on finding matchings) cannot be extended for $p > 2$.

for graphs in which a suitable separator structure can be found.

We also consider another class of graphs in this paper, those in which many vertices have high float. We show that such graphs can be scheduled compactly by level-by-level algorithms (e.g. Hu's).

Theorem 2 *Let G be a DAG with n vertices in which at least $n\epsilon$ vertices have float at least $n\epsilon/p$. Then there exists a linear time algorithm that will give a schedule within factor $2 - \epsilon$ of the optimal.*

High float graphs and the above result are of interest because of two reasons. First, in real problems, it is likely that many vertices will have a large float (every vertex need not be on a critical path, or nearly critical path). Second, high float graphs are important to us because they arise in proving Theorem 1 as well.

1.3 Overview

Section 3 considers graphs with high float. Theorem 2 is a corollary of a theorem proved there. Section 4 states our main observation that enables us to prove lower bounds on schedule length. Section 5 considers the problem of scheduling loosely connected graphs and proves Theorem 1. We begin with some preliminaries.

2 Preliminaries

If g is a subgraph or a set of vertices, and T a schedule, then will use $T(g)$ to denote the time interval over which the vertices in g are scheduled. Likewise $T^{-1}(i)$ and $T^{-1}(i,j)$ will mean the set of vertices scheduled at time i and during the interval $[i,j]$ respectively, for $1 \leq i \leq j \leq d$, where d is the length of T. The phrase "ith slot of T" will mean $T^{-1}(i)$. We will say that "the ith slot of T has holes" iff $|T^{-1}(i)| < p$.

Unconstrained schedules: We use this term to mean optimal schedules in which the number of processors is unbounded. We study such schedules because they serve as starting points in the construction of schedules with a bounded numbers of processor. Note that the length of unconstrained schedules is the same as the height of the graph. The height is also a lower bound on the length of schedules with a bounded number of processors. Two specific unconstrained schedules are of interest. The **eager** schedule E is an unconstrained schedule in which each task is scheduled at the earliest possible time.

The **lazy** schedule L is an unconstrained schedule in which each task is scheduled at the latest possible time. Our algorithms use $E(v)$ and $L(v)$, so we note that $E(v)$ and $L(v)$ can be computed in linear time.

We will use $F(v)$ to denote the float of a vertex. Note that $F(v) = L(v) - E(v)$.

Given an unconstrained schedule T, we define unconstrained schedules T' and T'' respectively obtained by **floating** or **sinking** a vertex u as follows. We set $T'(v) = L(v)$ if $v = u$ or v is a successor of u, otherwise $T'(v) = T(v)$. We set $T''(v) = E(v)$ if $v = u$ or v is a predecessor of u, otherwise $T''(v) = T(v)$. Clearly, T' and T'' as defined are also unconstrained schedules.

2.1 Greedy Scheduling Algorithms:

Every scheduling algorithm that does not idle processors while work is available can be described using the following generic structure:

(1) Assign a priority to each vertex. How to do this is considered shortly.
(2) Pick the smallest priority vertex v all whose predecessors (if any) have already been scheduled.
(3) Schedule v in the earliest possible time slot, consistent with the previously scheduled vertices.
(4) Repeat from step 2 until all vertices have been scheduled.

We will call such algorithms *greedy*, and the schedules they generate *greedy schedules.*[4] We later need the following property of greedy schedules:

Lemma 1 *For any greedy schedule T and any vertex v, at most $E(v) - 1$ slots with holes precede slot $T(v)$.*

Proof: We use induction on $E(v)$. Clearly, if $E(v) = 1$ then v has no predecessors. Hence there can be no slot with holes before slot $T(v)$; otherwise v would have been scheduled in such a slot. Thus the base case is proved.

Suppose the lemma is true for vertices u with $E(u) \leq l$. Consider any vertex v with $E(v) = l + 1$. Since $E(v) > 1$, v must have predecessors. Of all the predecessors of v, let u be one that is scheduled latest. Clearly $E(u) < E(v)$, and thus by the induction hypothesis, we know that the number of slots with holes preceding slot $T(u)$ in which u is scheduled must be at most $E(u) - 1$. Slot $T(u)$ can itself have holes. However, there can be no holes in slots $T(u)+1$ through $T(v) - 1$ since otherwise v could have been scheduled there. Thus the

[4] It should be clear that there is no reason to consider non-greedy schedules in practice. However, non-greedy schedules, such as the lazy (unconstrained) schedule described above can be indirectly useful.

number of slots with holes preceding slot $T(v)$ is at most $E(u)-1+1=E(u)$. But $E(u) \leq E(v)-1$. $\square$

Corollary 1 *Any greedy schedule for a graph G with height H and n vertices has length at most $\frac{n}{p}+H(1-\frac{1}{p}) \leq \frac{n}{p}+H$.*

Since the schedule length must be at least $\max(H, \frac{n}{p})$, the above provides the well known $2-\frac{1}{p}$ approximation. The proof is obvious and so omitted.

Sharper results can be obtained by assigning priorities carefully. Hu's algorithm [5] uses $L(v)$ as the priority (increasing order of $L(v)$ is the same as the order of decreasing length of path originating at v). If several vertices have same $L(v)$ in step 2 above, Hu's algorithm selects any one. Coffman-Graham algorithm is the same as Hu's except that the ties in step 2 are broken by using a clever, more refined priority assignment – and this gives the stronger results.

3 High Float Graphs

Theorem 3 (High Float Theorem) *Suppose a task graph G with height H and n vertices has at least n_f vertices with float at least f. The size of the schedule generated using Hu's algorithm is at most*

$$H+\frac{n}{p} \;-\; \min\left(f,\ \frac{n_f}{p}\right)$$

Proof: We use a *delay sequence* argument. A vertex with float at least f is said to be a *high float* vertex, and one with float less than f a *low float* vertex.

Let the length of the schedule be d, and let v_d denote any vertex in the last slot. We investigate why v_d was not scheduled earlier.

In general, given any v_i in slot $i>1$ there could be two reasons why v_i could not be scheduled in slot $i-1$:

(1) A predecessor v of v_i was scheduled in slot $i-1$. We colour v green and set $v_{i-1}=v$. Note that $L(v_i)>L(v_{i-1})$.

(2) No predecessor was in slot $i-1$, but the slot is full, and contains vertices v such that $L(v) \leq L(v_i)$ (remember that the algorithm schedules in increasing order of $L(v)$). Now there are two cases to be considered:
   (a) None of the vertices in slot $i-1$ has high float. In this case from these vertices we choose any vertex v, colour it red and set $v_{i-1}=v$.

(b) Slot $i-1$ contains at least one high float vertex. In this case we select one of these as v_{i-1}.

In this way we can identify vertices v_{d-1}, v_{d-2} and so on. We stop when we reach a vertex v_m that has high float, or if $m = 1$.

It is easy to see that $H \geq L(v_d) \geq L(v_{d-1}) \geq \ldots \geq L(v_m)$. Further, if v_i is green, then $L(v_{i+1}) > L(v_i)$. Thus the total number of green vertices can be at most $L(v_d) - L(v_m) \leq H - L(v_m)$. Now there are two cases depending upon how the construction terminated.

Suppose v_m is a high float vertex. Thus $L(v_m) - E(v_m) = F(v_m) \geq f$, and by Lemma 1 there are at most $E(v_m) - 1$ slots with holes in slots 1 through $m-1$. Now consider slots in the range m through $d-1$. Slot i can have holes only if v_i is green. Thus the number of slots with holes is at most $H - L(v_m)$. In addition, slot d may have holes. Thus the total number of slots with holes is at most $E(v_m) - 1 + H - L(v_m) + 1 \leq H - f$. Thus there are at least $d - (H - f)$ full slots, i.e., $n \geq p(d - H + f)$. Thus

$$d \leq \frac{n}{p} + H - f \tag{1}$$

Otherwise v_m is not a high float vertex, and $m = 1$. Of the vertices $v_d, \ldots, v_1$ we know that v_d could be colourless, and at most $H - L(v_1)$ vertices green. Thus there are more than $d - 1 - H + L(v_m) = d - 1 - H + 1 = d - H$ red vertices. But if v_i is red, then we know that slot i contains p low float vertices. Thus there are at least $p(d - H)$ low float vertices. But the number of low float vertices is at most $n - n_f$. Hence:

$$d \leq \frac{n - n_f}{p} + H \tag{2}$$

From (1) and (2) we get $d \leq \frac{n}{p} + H - \min(\frac{n_f}{p}, f)$. $\square$

Theorem 2 follows from this easily and the proof is omitted.

4 Incompressible Graphs

Definition 1 *Graph G is said to be (n', h) incompressible if there exist integers $s + e = h$ such that each of at least n' vertices of G has a path of length at least e preceding it and a path of length s ahead of it.*

Theorem 4 *Any schedule of a (n', h) incompressible graph has length at least $h + \frac{n'}{p}$.*

Proof: The length of the interval in which the n' vertices get scheduled must be at least $\frac{n'}{p}$. But this interval must be preceded and succeeded by at least e and s slots respectively with $s + e = h$. $\square$

5 Loosely connected Graphs

We begin by outlining the algorithm for loosely connected graphs, and eventually prove Theorem 1.

Throughout this section G, n, p, H are as defined in Theorem 1. We use δ to denote $\frac{1}{8}\min\left(H, \frac{n}{p}\right)$. G_1, G_2 are as per the definition of δ loosely connected graphs, i.e., G splits into G_1, G_2 when edges of float at least δ are removed from G, with $|G_1|, |G_2| \geq \frac{n}{4} \geq 2\delta p$.

Algorithm Outline: We first determine if the given graph G is incompressible. If G is incompressible, then we use any greedy schedule for G. Otherwise, we will show that it has a *compressibility certificate* in a sense to be defined shortly. This certificate enables us to partition G into two subgraphs G' and G'' such that (i) Both subgraphs have a large number of vertices with high float, and (ii) There are no edges from G'' to G'. We then generate a good schedule for G simply by concatenating the schedules for G' and G'' – these schedules are in turn obtained using the High float theorem (Theorem 3).

Definition 2 *Suppose T is an unconstrained schedule for G and x, y are integers such that $y - x = 1 + \delta$. Suppose further that at least δp vertices of G_1 are scheduled at or before x and at least δp vertices of G_2 are scheduled at or after y, or vice versa. More precisely, either*

$$|T^{-1}(1, x) \cap G_1| \geq \delta p \text{ and } |T^{-1}(y, H) \cap G_2| \geq \delta p,$$

or

$$|T^{-1}(1, x) \cap G_2| \geq \delta p \text{ and } |T^{-1}(y, H) \cap G_1| \geq \delta p.$$

Then we will say that (T, x, y) is a compressibility certificate.

Let y be largest such that at least $2\delta p$ vertices are scheduled at or after y in the lazy schedule for G, i.e., y is largest such that

$$\left|L^{-1}(y,H)\right| \geq 2\delta p$$

Let S be the unconstrained schedule obtained from L by sinking all vertices in $L^{-1}(1,y-1)$. Let x be smallest such that at least $2\delta p$ vertices are scheduled at or before x in S, i.e., x is smallest such that

$$\left|S^{-1}(1,x)\right| \geq 2\delta p$$

Lemma 2 *If $y-x \leq 2\delta$ then the graph is $(n-4\delta p, H-2\delta-1)$ incompressible. Otherwise the graph has a compressibility certificate.*

Proof: Suppose $y-x \leq 2\delta$. As per the definition of x and y we know that $|S^{-1}(1,x-1)| < 2\delta p$ and $|S^{-1}(y+1,H)| < 2\delta p$. But $|S^{-1}(1,H)| = n$. Hence $S^{-1}(x,y) > n-4\delta p$. Further by construction each vertex in $S^{-1}(x,y)$ has a path of length at least $H-y$ ahead of it and a path of length at least $x-1$ preceding it. But $H-y+x-1 \geq H-2\delta-1$. Thus G is $(n-4\delta p, H-2\delta-1)$ incompressible.

Otherwise $y-x \geq 2\delta+1$. For $i=1,2$ (see Figure 1) let $X_i = S^{-1}(1,x) \cap G_i$, and let $Y_i = S^{-1}(y,H) \cap G_i$. We know that $|X_1|+|X_2| \geq 2\delta p$, and $|Y_1|+|Y_2| \geq 2\delta p$. If $|X_1|,|Y_2| \geq \delta p$ or $|X_2|,|Y_1| \geq \delta p$, then $(S,x,x+\delta+1)$ is the required certificate. Otherwise let $|X_1|,|Y_1| \geq \delta p$ without loss of generality. Let $X' = S^{-1}(1,y-\delta-1) \cap G_2$. Let $Y' = S^{-1}(x+\delta+1,H) \cap G_2$. But we assumed that $y-\delta-1 \geq x+\delta$, and hence the intervals $[1,y-\delta-1]$ and $[x+\delta+1,H]$ together cover the interval $[1,H]$. Thus $|X'|+|Y'| \geq |G_2| \geq 2\delta p$. Thus either (a) $|X'| \geq \delta p$ and $(S,y-\delta-1,y)$ is the required certificate, or (b) $|Y'| \geq \delta p$ and $(S,x,x+\delta+1)$ is the required certificate. $\square$

It should be clear that deciding compressibility as above and constructing the certificate if necessary can be done in linear time.

5.2 Algorithm for the Compressible case

Suppose we have a certificate (T,x,y) for compressibility of G.

Let T' be the unconstrained schedule obtained from T by sinking vertices $T^{-1}(1,y-1)$. Let T'' be the unconstrained schedule obtained from T' by floating vertices $T'^{-1}(x+1,H)$.

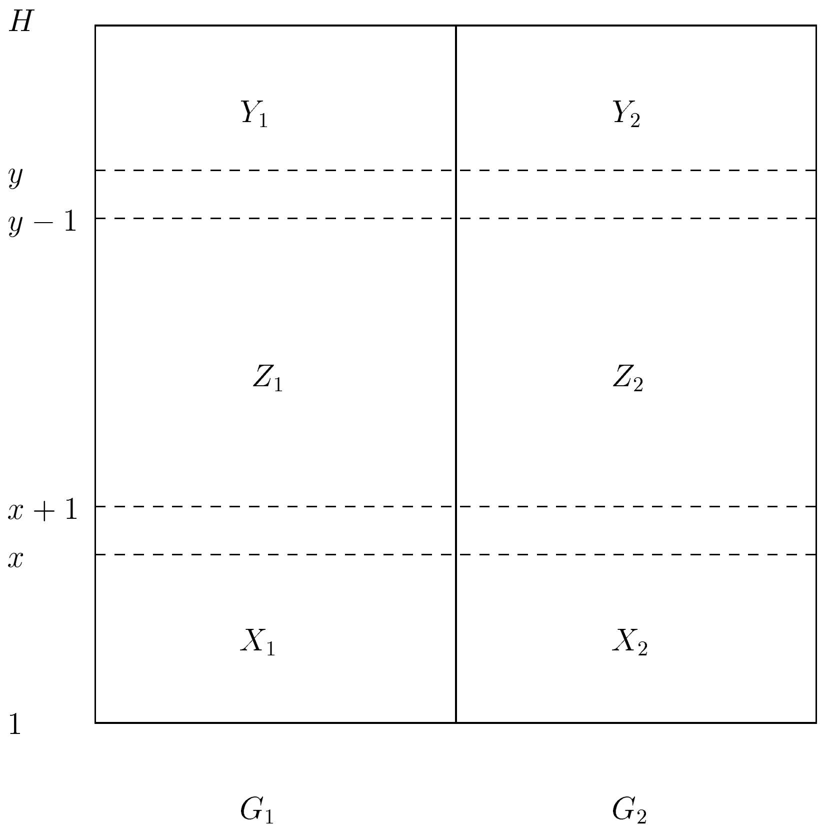

Fig. 1. Schematic definition of X_i, Z_i, Y_i (include bordering levels)

Lemma 3 *(T'', x, y) is a certificate of compressibility for G. Further, every vertex in $T''^{-1}(x+1, y-1)$ has a path of length at least x preceding it and a path of length $H - y + 1$ ahead of it.*

Proof: Clearly $T'^{-1}(1, x) \supseteq T^{-1}(1, x)$, while $T'^{-1}(y, H) = T^{-1}(y, H)$. Thus (T', x, y) is also a certificate. Further, for any $v \in T'^{-1}(x+1, y-1)$, we know that $T'(v) = E(v)$, guaranteeing that there is a path of length at least $E(v) - 1 \geq x$ preceding v. Continuing in this manner we can prove that (T'', x, y) is a certificate, and the second requirement on paths. $\square$

Let $X_i = T''^{-1}(1, x) \cap G_i$, $Z_i = T''^{-1}(x+1, y-1) \cap G_i$ and $Y_i = T''^{-1}(y, H) \cap G_i$. Note that these definitions of the names X_i, Y_i are different from the ones in Lemma 2. However, the new definitions are similar in spirit, and we can still use Figure 1 for reference. Since (T'', x, y) is a certificate, we know that either $|X_1|, |Y_2| \geq \delta p$, or $|X_2|, |Y_1| \geq \delta p$. Assume the former without loss of generality.

Let G' be the graph induced by vertices in X_1, X_2, Z_2. Let G'' be the graph induced by vertices in Y_1, Y_2, Z_1. Let H', H'' be the heights of G', G''.

Lemma 4 *The graph G' has the following important properties: (i) $H' \leq y-1$, and (ii) considering the problem of scheduling G' alone, every vertex $v \in X_1$*

has float at least $H' - y + \delta + 1$.

Proof: Since T'' includes an unconstrained schedule for G' of length at most $y-1$, clearly $H' \leq y-1$. Consider any vertex $v \in X_1$. Let P denote the longest path through v with $H'(v)$ its length. There are two cases:

(1) Suppose P only contains vertices in X_1 or X_2. Since all vertices in X_1, X_2 are scheduled in the interval $[1, x]$ it follows that $H'(v) \leq x = y - \delta - 1$.

(2) Suppose P ends at a vertex in Z_2. By Lemma 3, we know that P can be extended in G by at least $H-y+1$ vertices. This extended path has length at least $H'(v) + H - y + 1$. But this extended path passes through G_1 as well as G_2. Hence its length can be at most $H - \frac{1}{8}\min\left(H, \frac{n}{p}\right) \leq H - \delta$. Thus $H'(v) + H - y + 1 \leq H - \delta$, that is, $H'(v) \leq y - 1 - \delta$.

Hence the float of v is $H' - H'(v) \geq H' - y + \delta + 1$. $\square$

Lemma 5 *The graph G'' has the following important properties: (i) $H'' \leq H - x$, and (ii) considering the problem of scheduling G'' alone, every vertex $v \in Y_2$ has float at least $H'' - H + x + \delta$.*

Proof: Similar to the proof of Lemma 4 and therefore omitted. $\square$

Lemma 6 *There is no directed edge from G'' to G'.*

Proof: Suppose such an edge (v_1, v_2) exists. Since T'' is a valid schedule it follows that $v_1 \in Z_1$ and $v_2 \in Z_2$.

But we know that v_1 has a path preceding it of length at least x, and v_2 has a path ahead of it of length at least $H-y+1$. Thus the edge (v_1, v_2) is contained in a path of length at least $x+2+H-y+1$. But the length of any path passing through both G_1 and G_2 is at most $H - \frac{1}{8}\min\left(H, \frac{n}{p}\right) \leq H - \delta = H - y + x + 1$. Thus we have a contradiction. $\square$

5.3 Schedule

Since there are no edges from G'' to G', a schedule for G can be constructed by concatenating the schedules for G' and G'' in that order. G' and G'' are scheduled keeping in mind their high float components X_1 and Y_2 respectively.

Lemma 7 *There exists a schedule for G of length at most $H + \frac{|G|}{p} - \delta$.*

Proof: We schedule G' using Theorem 3. By Lemma 4 we know G' contains at least $|X_1| \geq \delta p$ vertices of high float. Thus the schedule length is at most:

$$\frac{|G'|}{p} + H' - \min\left(H' - y + \delta + 1, \frac{\delta p}{p}\right) = \frac{|G'|}{p} + y - \delta - 1$$

We schedule G'' using Theorem 3. By Lemma 4 we know G'' contains at least $|Y_2| \geq \delta p$ vertices of high float. Thus the schedule length is at most:

$$\frac{|G''|}{p} + H'' - \min\left(H'' - H + x + \delta, \frac{\delta p}{p}\right) = \frac{|G''|}{p} + H - x - \delta$$

Thus the overall schedule length is

$$\frac{|G'|}{p} + y - \delta - 1 + \frac{|G''|}{p} + H - x - \delta = \frac{|G|}{p} + H - \delta$$

since $y - x = 1 + \delta$. $\square$

5.4 Proof of Theorem 1

There are two cases depending upon whether the graph is incompressible.

If the graph is $(n - 4\delta p, H - 2\delta - 1)$ incompressible, then we have a schedule of length $H + \frac{n}{p}$ and by Theorem 4 the lower bound is $H + \frac{n}{p} - 6\delta - 1$. Thus the ratio is

$$\frac{H + \frac{n}{p}}{H + \frac{n}{p} - 6\delta - 1} \leq \frac{H + \frac{n}{p}}{H + \frac{n}{p} - 7\delta} \tag{3}$$

Otherwise we can find a compressibility certificate and produce a schedule of length $H + \frac{n}{p} - \delta$. The lower bound in this case is $\max\left(H, \frac{n}{p}\right)$. The ratio is thus:

$$\frac{H + \frac{n}{p} - \delta}{\max\left(H, \frac{n}{p}\right)} \tag{4}$$

For $\delta = \frac{1}{8}\min\left(H, \frac{n}{p}\right)$ both the above ratios are at most 1.875.

Acknowledgements: I am grateful to Raghavendra Udupa for comments on an earlier draft. I would also like to thank an anonymous referee for careful reading and suggestions.

References

[1] B. Braschi and D. Trystram. A new insight into the Coffman-Graham algorithm. *SIAM Journal of Computing*, pages 662–669, June 1994.

[2] M. Charikar. Approximation algorithms for problems in combinatorial optimization, 1995. B. Tech. Project Report, Department of Computer Science and Engineering, Indian Institute of Technology, Bombay.

[3] E. Coffman and R. Graham. Optimal scheduling for two-processor systems. *Acta Informatica*, 1:200–213, 1972.

[4] M. Fujii, T. Kasami, and K. Ninomiya. Optimal sequence of two equivalent processors. *SIAM J. Appl. Math.*, 17(3):784–789, 1969.

[5] T. Hu. Parallel sequencing and assembly line problems. *Operations Research*, 9(6):61–68, November 1961.

[6] S. Lam and R. Sethi. Worst-case analysis of two scheduling algorithms. *SIAM Journal of Computing*, 6:518–536, 1977.

[7] J. Lenstra and A. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Operations Research*, 26:22–35, 1978.