Technical Memorandum Number 753

Maximizing Project Cash Availability

by

Joseph G. Szmerekovsky and George Vairaktarakis

October 2001

Department of Operations
Weatherhead School of Management
Case Western Reserve University
10900 Euclid Avenue
Cleveland, Ohio 44106-7235

Maximizing Project Cash Availability

Joseph G. Szmerekovsky and George Vairaktarakis*

Submitted to *Naval Research Logistics*, October 2001.

Abstract

Consider a project during the life cycle of which there are cash pay-outs and in-flows. To better meet his financial commitments, the project owner would like to meet all deadlines without risking running out of cash. We show that the cash availability objective is similar to the total weighted flowtime used to measure work-in-progress performance in the scheduling and inventory control literatures. In this article we provide several specialized solution methods for the problem of minimizing total weighted flowtime in an arbitrary acyclic project network, subject to activity release times and due-dates, where the activity weights may be positive or negative and represent cash in- and out-flows. We describe the structure of an optimal solution and provide several efficient algorithms and their complexity based on mincost and maxflow formulations.

*Weatherhead School of Management, Dept. of Operations, Case Western Reserve University, 10900 Euclid Avenue, Cleveland, OH 44106-7235

1 Motivation and Literature Review

Consider the cash in- and out-flows associated with a project. Multiply each cash flow by the amount of time from the time it is committed until project completion. This product is measured in money-time. The measure proposed in this article is the sum of these products over all cash flows associated with a project. This sum is called *cash availability*. The associated optimization problem is to identify a schedule that maximizes cash availability.

The notion of cash availability as a means for evaluating projects is discussed in Goldratt, 1997. The author suggests cash availability as an alternative measure for financial evaluation of projects. Today the two most popular methods for evaluating projects are the *Payback period* and the *Net Present Value* (NPV).

Unlike manufactured products, projects may not require storing materials at sites controlled by the project owner. This is for instance the case with construction projects. Such projects often involve hundreds of subcontractors. Partial payments are made to subcontractors according to bilateral agreements. Such payments are only recovered by the project owner upon project completion, much the same way that work-in-process costs are captured by manufacturers upon selling finished goods. Evidently, cash availability is the project equivalent of work-in-progress (WIP) inventory costs in production. A lean manufacturer wants to minimize the investment in WIP in the same way that a project owner wants to minimize the average cash amount needed throughout the lifetime of the project. WIP costs are measured using total weighted flowtime. This measure is a sum of n products, one for every job. Associated with each job i is the product of the inventory value w_i committed to initiate (or complete) the job, multiplied by the length of time t_i for which this amount is committed. Following the usual assumption that an order of n jobs starts processing once all materials are available for the order, i.e., at time 0, the length of time t_i is the completion time C_i of job i. Hence, the product $w_i C_i$ measures the WIP cost associated with job i. The sum of these costs is known as *total weighted flowtime*. Minimizing this measure results to minimizing WIP costs as measured in money-time units. This measure was first suggested in Smith, 1956 as a way to minimize WIP costs in single machine scheduling.

To see that cash availability in projects is analogous to WIP costs in production, replace jobs by project activities, inventory costs w_i by the corresponding activity cash flows, and the completion time C_i of a job by the elapsed time $D - C_i$ between the completion of the project and the completion of

the corresponding activity i. Then, minimizing the total weighted flowtime $\sum_i w_i C_i$ of a set of jobs is equivalent to maximizing the cash availability $\sum_i w_i(D - C_i)$ in a project. When considering WIP costs all w_i's reflect out-flows. In the case of projects however, the w_i's are unconstrained in sign. Minimizing $\sum_i w_i C_i$ in projects when the w_i's are unconstrained in sign is a different problem not previously considered in the literature. Motivated by these observations, in this article we consider the project scheduling problem of maximizing the cash availability of a project subject to precedence constraints for activities, or equivalently, minimizing the total weighted flowtime (MWF).

To the best of our knowledge problem MWF first appeared as a relaxation of a more general project scheduling problem in Vairaktarakis and Syam, 2001. In that paper standard linear programming is used to solve MWF. In this article, infinite capacity is assumed for executing the various project activities. In contrast, Vairaktarakis and Syam, 2001 assume the presence of machine resources and that each activity can be processed by a subset of the machines. Moreover, the duration of each activity is the same for all eligible machines. Other problems with the weighted flowtime objective have been studied extensively in the machine scheduling literature (see Brucker, 1998 and/or Pinedo, 1996). Motivated by an application in the postal service, Horn, 1972 provides an efficient optimal algorithm for minimizing weighted flowtime under treelike precedence constraints on a single machine. Compared to the problem solved by Horn, MWF is unconstrained in machine resources (i.e. at least one machine per activity) and allows for arbitrary precedence constraints. All other examples in the machine scheduling literature that we are aware of consider only positive or only negative weights for activities.

As discussed in the previous section, NPV methods are relevant to our work. Such methods consider cash flows that are not restricted in sign. In the deterministic unconstrained *maximum NPV* problem (maxNPV), a cash in- or out-flow occurs at the completion of each activity and the objective is to maximize the net present value of the project relative to precedence constraints for the activities. Problem maxNPV was introduced in Russell, 1970 using the activity-on-arc (AoA) formulation. He showed that when the objective of maxNPV is approximated by a linear function through use of a Taylor series expansion, the dual of the resulting problem has the structure of a minimum cost flow problem. This lead to a successive approximation algorithm which produces a sequence of points guaranteed to converge to a local optimum of maxNPV. In Grinold, 1972 the structure of the dual was further exploited to develop a simplex based algorithm in which an extreme point corresponds to a tree of critical arcs in the precedence network. This tree helps to determine the project duration/cost

trade-off. Using this critical tree structure Elmaghraby and Herroelen, 1990 develop a simple recursive algorithm for maxNPV. The algorithm works by first scheduling all activities as early as possible. This results to a critical tree as identified by Grinold, 1972. The algorithm then searches the critical tree for a subtree of jobs that can be delayed to improve the objective function. This results to a new critical tree on which the search is repeated, recursively, until no improving subtree is found. Elmaghraby and Herroelen, 1990 propose an algorithm to solve maxNPV. Sepil, 1994 shows that the algorithm of Elmaghraby and Herroelen will not always provide the optimal solution to maxNPV. In a follow up paper, Herroelen and Gallens, 1993 present a revised version of the recursive algorithm and demonstrate through computational results that the new algorithm provides the optimal solution to maxNPV, however, no formal proof of correctness is given.

More recent algorithms for solving maxNPV have used an activity-on-node (AoN) formulation. Icmeli and Erenguc, 1996 adapted Grinold's work to the AoN formulation. This new algorithm was then applied within a branch-and-bound procedure to solve a resource constrained version of maxNPV. In a related paper, Demelumeester *et al.*, 1996 adapted the recursive procedure of Herroelen and Gallens, 1993 using an AoN formulation. Computational experimentation has shown this recursive procedure to be the most promising to date.

For an overview of literature relating to maxNPV and related problems see Herroelen *et al.*, 1997. Herroelen *et al.*, 1997 describe motivations behind the maxNPV formulation and related contractual issues. The authors classify the models in the literature as either deterministic or stochastic and as either unconstrained or resource constrained. A number of papers in each classification scheme is discussed. The survey also includes references to problems related to the maxNPV model, such as the time/cost tradeoff problem and the payment scheduling problem. In this classification MWF would be considered a deterministic unconstrained problem. MWF is a restricted version of maxNPV in that there is no discounting of cash flows. However it extends other maxNPV models in that it allows for time dependent cash flows. The notion of time dependent cash flows for maxNPV was first suggested in Elmaghraby and Herroelen, 1990 though no analysis on the proposed model has yet appeared in the literature.

The rest of this article is organized as follows. In Section 2 we formulate the minimum weighted flowtime project scheduling problem MWF. In Section 3 we describe an efficient algorithm for solving the MWF, prove its optimality and compute its complexity. The algorithm described in Section 3 requires the partitioning of activities into two sets. This subproblem is referred to as the *bipartition*

subproblem (BP). In Section 3.1 subproblem BP is formulated and shown to be solvable as a maximum flow problem. In Section 4 a specialized version of the algorithm described in Section 3 is presented. This new algorithm uses sensitivity analysis to improve the performance of the original algorithm. As a final solution procedure Section 5 discusses the solution of the dual of MWF as a minimum-cost-flow problem MCF. Concluding remarks are offered in Section 6.

2 Problem Formulation and Solution Procedure

The following notation is used throughout the paper:

$\mathcal{N}$**:** the project network

n**:** the number of activities in $\mathcal{N}$

a_i**:** the i-th activity to be scheduled; $1 \leq i \leq n$

A**:** the arc set of $\mathcal{N}$; $A \subset \{1, \ldots, n\} \times \{1, \ldots, n\}$. If $(i, j) \in A$ then a_i is a predecessor of a_j and a_j is a successor of a_i

$S(a_i)$**:** the set of successors of a_i including a_i

$P(a_i)$**:** the set of predecessors of a_i including a_i

p_i**:** the processing time of activity a_i

d_i**:** the due-date of activity a_i

r_i**:** the release-date of activity a_i

C_i**:** the completion time of activity a_i

w_i**:** the cash flow paid or received upon completion of activity a_i

$EF_i(C_1, \ldots, C_{i-1}, C_{i+1}, \ldots, C_n)$: the early finish time of activity a_i for given completion times $C_1, \ldots, C_{i-1}, C_{i+1}, \ldots, C_n$ for activities $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n$, respectively.

$LF_i(C_1, \ldots, C_{i-1}, C_{i+1}, \ldots, C_n)$: the late finish time of activity a_i for given completion times $C_1, \ldots, C_{i-1}, C_{i+1}, \ldots, C_n$ for activities $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n$, respectively.

Problem MWF can now be written as follows:

$$(MWF) \qquad \min \quad \sum_{i=1}^{n} w_i C_i \tag{1}$$

$$\mathbf{s.t.} \quad C_j - C_i \geq p_j, \quad \forall\ (i,j) \in A \tag{2}$$

$$C_i \geq r_i + p_i \quad i = 1, \ldots, n \tag{3}$$

$$C_i \leq d_i \quad i = 1, \ldots, n \tag{4}$$

As discussed earlier objective (1) captures cash availability. Constraints (2) preserve the precedence constraints while (3) prevent activities from starting before their release dates. Constraints (4) insure that every job is completed by its due date.

MWF is solved by solving a series of bipartition subproblems BP defined as follows.

Let x_i=1 if $a_i \in L$ and 0 otherwise.

$$(BP) \qquad \max \quad \sum_{a_i \in V} w_i x_i \tag{5}$$

$$\mathbf{s.t.} \quad x_i - x_j \geq 0, \quad \forall\ (i,j) \in \mathcal{A} \tag{6}$$

$$x_i \geq 0, \qquad a_i \in V \tag{7}$$

$$x_i \leq 1, \qquad a_i \in V. \tag{8}$$

The set of constraints (6) corresponds to the activity precedence constraints and (7) and (8) are upper and lower bounds on the binary variables. Since the set of constraints (6) is totally unimodular and (7), (8) are upper and lower bounds, integrality constraints are satisfied on extreme point solutions.

Solving BP is equivalent to selecting a subset L of activities in V so as to maximize the combined weight of the activities in that subset. This bipartition is performed under the restriction that an activity cannot be selected unless all of its predecessors are selected.

3 An Algorithm based on Maxflow Subproblems

The algorithm O-MWF presented in this section optimally solves MWF by solving BP subproblems. The start and finish times of at least one job from each subset are then fixed and the two subsets are updated to include the remaining unscheduled jobs. Once the start and finish times of all jobs are determined, the algorithm terminates with the optimal solution. The bipartition subproblem and

the algorithm for solving it are described in subsection 3.1. Next we formally state the algorithm and discuss its complexity.

In the following algorithm set V denotes the set of jobs on which the bipartition subproblem needs to be applied. When $V = \emptyset$ the algorithm terminates. The variables D_i represent upper bounds on the finish time of the activities a_i for $i = 1, 2, \ldots, n$ respectively and take on initial values $D_i = d_i$, for $i = 1 \ldots, n$. These bounds are updated iteratively by O-MWF so that the start and finish times of some activities are determined while preserving the precedence constraints of all activities. The variables $R_1, \ldots, R_n$ are lower bounds on the start times of the activities a_i for $1 \leq i \leq n$ respectively. Given vectors $\vec{R} = \{R_i\}_{i=1}^n$ and $\vec{D} = \{D_i\}_{i=1}^n$, we denote by $CPM(\vec{R}, \vec{D})$ the schedule produced by the critical path method when the i-th activity is scheduled to start no earlier than R_i and complete no later than D_i. Similarly, we use the early-start and late-start $CPM(\vec{R}, \vec{D})$ schedules.

Algorithm O-MWF

Input : Network $\mathcal{N}$ of activities $V = \{a_1, \ldots, a_n\}$, $\vec{R} = \{R_i\}_{i=1}^n$ and $\vec{D} = \{D_i\}_{i=1}^n$.

Output : Optimal activity completion times C_i, $i = 1, \ldots n$, for MWF.

Repeat

[1] Find L that solves subproblem BP.

[2] **If** $L \neq \emptyset$ **then**

Find the early-start $CPM(\vec{R}, \vec{D})$ schedule. **If** infeasible **then Stop** (MWF is infeasible)

Let $a_h \in L$ be the activity with the largest start time.

Set $R_h = C_h - p_h$, $D_h = C_h$ and $V = V/\{a_h\}$

[3] **If** $R \neq \emptyset$ **then**

Find the late-start $CPM(\vec{R}, \vec{D})$ schedule. **If** infeasible **then Stop** (MWF is infeasible)

Let $a_h \in R$ be the activity with the smallest completion time.

Set $R_h = C_h - p_h$, $D_h = C_h$ and $V = V/\{a_h\}$

Until $V = \emptyset$

Later we show that when the algorithm terminates the values R_i and D_i are optimal start and completion times, respectively, for activity a_i. At each iteration the set V is the subset of activities for which start and completion times are not yet optimized. At each iteration the bipartition subproblem on V is solved. This results to a partitioning of V into sets L and R. In steps [2] and [3] the completion

time of at least one activity is fixed. All activities with fixed completion times are removed from V and the algorithm repeats on the reduced set V with updated release-dates and due-dates for all activities. Since at least one activity is scheduled in each iteration, the algorithm terminates in no more than n iterations. In addition, if in some iteration of [2] or [3] the due- and release-dates are not satisfied, the algorithm should terminate concluding that the original problem is infeasible. Otherwise a feasible solution has been found. Therefore, we need only show that when the algorithm terminates, $D_1, \ldots, D_n$ are the optimal activity completion times. The following lemmata will be helpful in establishing the optimality of O-MWF.

Lemma 1 *If $C^* = \{C_i^*\}_{i=1}^n$ is optimal for MWF, then $C_i^* = EF_i(C_1^*, \ldots, C_{i-1}^*, C_{i+1}^*, \ldots, C_n^*)$ for all a_i with $w_i > 0$ and $C_i^* = LF_i(C_1^*, \ldots, C_{i-1}^*, C_{i+1}^*, \ldots, C_n^*)$ for all a_i with $w_i < 0$.*

Proof. If $w_i > 0$ and $C_i^* > EF_i(C_1^*, \ldots, C_{i-1}^*, C_{i+1}^*, \ldots, C_n^*)$ then a_i can be started earlier by $C_i^* - EF_i(C_1^*, \ldots, C_{i-1}^*, C_{i+1}^*, \ldots, C_n^*) > 0$ units without effecting $C_1^*, \ldots, C_{i-1}^*, C_{i+1}^*, \ldots, C_n^*$. This would result in a decrease in the objective of $w_i(C_i^* - EF_i(C_1^*, \ldots, C_{i-1}^*, C_{i+1}^*, \ldots, C_n^*)) > 0$, contradicting the optimality of C^*. Hence, we must have $C_i^* = EF_i(C_1^*, \ldots, C_{i-1}^*, C_{i+1}^*, \ldots, C_n^*)$. Similarly we must have $C_i^* = LF_i(C_1^*, \ldots, C_{i-1}^*, C_{i+1}^*, \ldots, C_n^*)$ when $w_i < 0$. This completes the proof of the lemma. □

Lemma 2 *Let $C^* = \{C_i^*\}_{i=1}^n$ be an optimal solution for MWF and L^* be the associated solution for BP applied to $\{a_1, \ldots, a_n\}$. Then, for every $a_i \in L^*$ with $w_i < 0$, there exists $a_j \in S(a_i) \cap L^*$ with $w_j > 0$.*

Proof. For contradiction, suppose that for all $a_j \in S(a_i)$ we have $a_j \in R^* = \{a_i\}_{i=1}^n / L^*$ or $w_j \leq 0$. Then, consider a new solution $L' = L^*/S(a_i)$ for BP. For L' we have $\sum_{a_k \in L'} w_k = \sum_{a_k \in L^*} w_k - \sum_{a_k \in S(a_i) \cap L^*} w_k$. By assumption, for all $a_k \in S(a_i) \cap L^*$ we have $w_k \leq 0$. In addition we have $a_i \in S(a_i) \cap L^*$ and $w_i < 0$. Thus $\sum_{a_k \in S(a_i) \cap L^*} w_k < 0$ and $\sum_{a_k \in L'} w_k = \sum_{a_k \in L^*} w_k - \sum_{a_k \in S(a_i) \cap L^*} w_k > \sum_{a_k \in L^*} w_k$, violating the optimality of L^*. This completes the proof of the lemma. □

We now establish the correctness of O-MWF.

Theorem 1 *The activity schedule, $\{C_i^*\}_{i=1}^n$, produced by O-MWF optimally solves problem MWF.*

Proof. We prove the optimality of the activity completion time determined by O-MWF in a single arbitrary iteration. Applying the same argument iteratively yields the result. We distinguish the following two cases.

Case 1: *Activity a_h is scheduled in line [2] of O-MWF.*

Let S^* be an optimal schedule associated with the completion times $\{C_i^*\}_{i=1}^n$ and without loss of generality assume that S^* is maximal in that it has the largest number of activity completion times in common with the schedule obtained by O-MWF.

Recall that in line [2], we set $C_h = D_h$ by scheduling a_h as early as possible. Then, by choice of the early-start schedule we have $C_h = D_h \leq C_h^*$. If $C_h^* = D_h$, then $C_h = D_h$ is optimal. Otherwise $C_h^* > D_h$. Then, in S^*, some predecessors of a_h have not completed prior to time $D_h - p_h$. Since $C_h^* > D_h$, it is possible to complete a_h at time D_h by shifting earlier by $\epsilon = C_h^* - D_h > 0$ time units some of the activities in $P(a_h)$. Let K be the set of predecessors of a_h with completion time equal to the start time of a_h and $S(K) = \cup_{a_i \in K} S(a_i) \cap L$. Hence, for every activity $a_i \in S(K)$, we must have $C_i^* \geq D_i + \epsilon$. Therefore, in S^* it is possible to start sooner every $a_i \in S(K)$ by ϵ time units. The associated change in the objective function will be $\epsilon \sum_{a_j \in S(K)} w_j$. If $\epsilon \sum_{a_j \in S(K)} w_j < 0$ we have $\sum_{a_j \in S(K)} w_j < 0$ and hence $\sum_{a_j \in L} w_j = \sum_{a_i \in L/S(K)} w_i + \sum_{a_j \in S(K)} w_j < \sum_{a_i \in L/S(K)} w_i$, thus violating the optimality of S^*. On the other hand, if $\sum_{a_i \in L/S(K)} w_i = 0$, shifting the activities in $S(K)$ earlier produces a schedule that has one additional completion time in common (i.e., C_h) with the schedule produced by O-MWF, violating the maximality of S^*. We conclude that $C_h = C_h^* = D_h$.

Case 2: *Activity a_h is scheduled in line [3] of O-MWF.*

This case is symmetric to Case 1. Note that, for given network $\mathcal{N}$ and weights w_i for the activities, MWF is equivalent to the instance where $\mathcal{N}$ is replaced by the reverse network $\mathcal{N}_R$ (where all predecessor/successor relationships are reversed) and the weights w_i are replaced by $-w_i$. Hence, scheduling an activity $a_h \in R$ of $\mathcal{N}$ is equivalent to scheduling $a_h \in L$ of $\mathcal{N}_R$.

The above arguments show that, after a fixed number r of iterations of O-MWF the completion time of a_h is optimal. Let H be the subset of activities with fixed completion times after r iterations of O-MWF. Then we have shown that there is an optimal solution to the original instance of MWF which is also an optimal solution to the following program:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} w_i C_i \\ \textbf{s.t.} \quad & (2) - (4) \\ & C_i = D_i \quad \forall\ a_i \in H. \end{aligned} \tag{9}$$

If we perform the substitution suggested by constraints (9) we will obtain a new instance of MWF with

updated release-dates and due-dates for activities not in H. Let I and I' be the original and updated instances after the first iteration of O-MWF. The $(r+1)$st iteration of O-MWF on I is identical to iteration 1 on I'. Hence, applying our arguments recursively establishes the correctness of O-MWF at every iteration of the algorithm. This completes the proof of the theorem. □

To determine the complexity of O-MWF note that, in every iteration we fix the completion time of at least one activity. Therefore the algorithm solves no more then n BP subproblems and no more than n CPM subproblems. In the next section we show that the BP subproblem can be solved as a maximum flow problem. Simple algorithms exist for solving the maximum flow problem in $\mathcal{O}(n^3)$ time (see Ahuja *et.al.*, 1993). Hence, O-MWF requires $\mathcal{O}(n^4)$ time.

3.1 The Bipartition Problem

In line [1] of algorithm O-MWF we are required to solve BP. We will find an optimal extreme point for BP by solving its dual program:

$$
\begin{array}{lrcll}
(DMWF) & \min & \sum\limits_{a_i \in V} v_i & & \\
\textbf{s.t.} & \sum\limits_{(k,j)\in\mathcal{A}} u_{kj} - \sum\limits_{(i,k)\in\mathcal{A}} u_{ik} + y_k + v_k & = & w_k, & a_k \in V \\
& u_{ij} & \leq & 0, & \forall\ (i,j) \in \mathcal{A} \\
& y_i & \leq & 0, & a_i \in V \\
& v_i & \geq & 0, & a_i \in V.
\end{array}
\tag{10}
$$

Here the variables u_{ij} correspond to the set of constraints (6), the variables y_k to the set of constraints (7) and the variables v_k to the set of constraints (8). We will convert this dual formulation into a maximum flow problem in 2 steps. First we show that DMWF can be written as a minimum cost flow problem with special structure. Then, we show that this special structure allows us to solve the minimum cost flow problem as a maximum flow problem. Indeed, let us substitute $-u_{ij}$ for u_{ij} and

$-y_i$ for y_i. This yields

$$\begin{array}{lll} \min & \sum\limits_{a_k \in V} v_k & \\ \textbf{s.t.} \quad (\sum\limits_{(i,k)\in\mathcal{A}} u_{ik} + v_k) - (\sum\limits_{(k,j)\in\mathcal{A}} u_{kj} + y_k) & = & w_k, \quad a_k \in V \\ u_{ij} & \geq & 0, \quad \forall\ (i,j) \in \mathcal{A} \\ y_i & \geq & 0, \quad a_i \in V \\ v_i & \geq & 0, \quad a_i \in V \end{array} \tag{11}$$

Now, by adding a dummy node a_0, substituting $v_k = u_{0k}$ and $y_k = u_{k0}$, setting $w_0 = -\sum\limits_{i=1}^{n} w_i$ and updating $\mathcal{A}$ to $\mathcal{A}'$ with arcs from node a_0 to every other node and from every other node to node a_0, we get

$$(MC) \qquad \min \quad \sum_{a_k \in V} u_{0k} \tag{12}$$

$$\textbf{s.t.} \quad \sum_{(i,k)\in\mathcal{A}'} u_{ik} - \sum_{(k,j)\in\mathcal{A}'} u_{kj} = w_k, \quad a_k \in V \tag{13}$$

$$u_{ij} \geq 0, \qquad \forall\ (i,j) \in \mathcal{A}' \tag{14}$$

which is an uncapacitated minimum cost flow problem. The table below gives the cost for each edge in the network and the demand at each node.

Node	Demand	Arc	Cost
a_0	$w_0 = -\sum\limits_{i=1}^{n} w_i$	(a_0, a_i)	1
		(a_i, a_0)	0
$a_i \in V$	w_i	$(a_i, a_j) \in \mathcal{N}$	0

Example: Consider the 8-activity instance with weights $w_1 = -5, w_2 = -2, w_3 = -15, w_4 = 8, w_5 = 12, w_6 = 10, w_7 = -10$ and $w_8 = 13$, processing times $p_1 = 3, p_2 = 8, p_3 = 5, p_4 = 6, p_5 = 7, p_6 = 5, p_7 = 11$ and $p_8 = 14$, release-dates $r_1 = 0, r_2 = 0, r_3 = 0, r_4 = 5, r_5 = 5, r_6 = 10, r_7 = 0$ and $r_8 = 10$, and due-dates $d_1 = 40, d_2 = 40, d_3 = 40, d_4 = 40, d_5 = 20, d_6 = 40, d_7 = 20$ and $d_8 = 40$. Figure 1 depicts network $\mathcal{N}$ for this example. The bold line in the figure indicates an optimal solution of BP for this example problem.

INSERT FIGURE 1 HERE

Figure 2 shows the mincost flow network for the example of Figure 1.

INSERT FIGURE 2 HERE

To solve problem MC as a maximum flow problem observe that every arc has cost 0 except for those out of node a_0 which have cost 1. This problem is similar to Phase 1 of solving minimum cost flow problems. This Phase 1 problem can be solved as a maximum flow problem (see Ahuja *et. al.*, 1993) as follows. Consider $s-t$ network $\mathcal{N}'$ with edge set

$$B \quad = \quad \{(i,j) \in \mathcal{A}' : i \neq 0\} \bigcup \{(s,j) : w_j \leq 0\} \bigcup \{(i,t) : w_i > 0\},$$

capacity $-w_j$ on the arcs (s, a_j), and capacity w_i on arcs (a_i, t). The maximum flow problem for $\mathcal{N}'$ can be written as

$$(MF) \qquad \max \qquad z \tag{15}$$

$$\textbf{s.t.} \quad \sum_{j:(i,k)\in\mathcal{B}} u_{ik} - \sum_{i:(k,j)\in\mathcal{B}} u_{kj} = \begin{cases} -z, & \text{if } k = s \\ z, & \text{if } k = t \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

$$u_{ij} \geq 0, \qquad \forall\ (i,j) \in \mathcal{B} \tag{17}$$

$$u_{sj} \leq -w_j, \qquad \forall\ a_j\ : w_j \leq 0 \tag{18}$$

$$u_{it} \leq w_i, \qquad \forall\ a_i\ : w_i > 0. \tag{19}$$

The table below describes the network $\mathcal{N}'$ associated with this maxflow problem.

Edge	Capacity
(s, a_i)	$-w_i$
(a_i, t)	w_i
(a_i, a_j)	∞

Example: The maxflow network $\mathcal{N}'$ corresponding to our example is shown in Figure 3.

INSERT FIGURE 3 HERE

In the following theorem we prove that solving the minimum cost flow problem MC is equivalent to solving problem MF.

Theorem 2 *u' is an optimal solution for MF iff u^* is an optimal solution for MC where,*

$$u^*_{ij} \quad = \quad \begin{cases} u'_{ij} & \text{if } i \neq 0 \text{ and } j \neq 0, \\ w_j - u'_{jt} & \text{if } i = 0 \text{ and } w_j > 0, \\ u_{ij} + (-w_i - u'_{si}) & \text{if } j = 0 \text{ and } w_i \leq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{20}$$

Moreover, if z^*, z' are the optimal values for problems MC, MF respectively, $z^* = \sum_{w_j>0} w_j - z'$.

Proof. Let u' be optimal for MF and u^* be as in (20). We first show that u^* is a feasible solution for MC. To do this we must show that u^* is nonnegative and that it satisfies the flow balance constraints (13). If $u^*_{ij} = 0$ or $u^*_{ij} = u'_{ij}$ then u^*_{ij} is nonnegative due to (17). If $u^*_{ij} = w_j - u'_{jt}$ then $u^*_{ij} \geq 0$ due to (19). Otherwise $u^*_{ij} = u'_{ij} + (-w_i - u'_{si})$ and $u^*_{ij} \geq 0$ due to (17) and (18). In all cases it is guaranteed that u^* will be nonnegative. To see that u^* satisfies the flow balance constraints (13) at each node k we distinguish two cases.

Case 1: Activity a_k has weight $w_k > 0$. We need to show $\sum_{(i,k)\in\mathcal{A}'} u^*_{ik} - \sum_{(k,j)\in\mathcal{A}'} u^*_{kj} = w_k$. Indeed,

$$\begin{aligned}
\sum_{i:(i,k)\in B} u'_{ik} - \sum_{j:(k,j)\in B} u'_{kj} &= 0 && \text{(flow balance constraints (16))}\\
\sum_{i:(i,k)\in B} u'_{ik} - \sum_{j:(k,j)\in B, j\neq t} u'_{kj} &= u'_{kt}\\
\sum_{i:(i,k)\in B} u^*_{ik} - \sum_{j:(k,j)\in B, j\neq t} u^*_{kj} + u^*_{0k} &= u'_{kt} + u^*_{0k} && \text{(from equality (20))}\\
\sum_{i:(i,k)\in\mathcal{A}'} u^*_{ik} - \sum_{j:(k,j)\mathcal{A}'} u^*_{kj} &= w_k && \text{(by (20): } u^*_{0k} = w_k - u'_{kt} \text{ when } i = 0 \text{ and } w_k > 0)
\end{aligned}$$

Case 2: Activity a_k has weight $w_k \leq 0$. To show the converse, observe that

$$\begin{aligned}
\sum_{i:(i,k)\in B} u'_{ik} - \sum_{j:(k,j)\in B} u'_{kj} &= 0 && \text{(flow balance constraints (16))}\\
\sum_{i:(i,k)\in B, i\neq s} u'_{ik} - \sum_{j:(k,j)\in B} u'_{kj} &= -u'_{sk}\\
\sum_{i:(i,k)\in B, i\neq s} u^*_{ik} - \sum_{j:(k,j)\in B, j\neq 0} u^*_{kj} - (u'_{k0} - w_k - u'_{sk}) &= -u'_{sk} + w_k + u'_{sk} && \text{(by adding } w_k + u'_{sk} \text{ to both sides)}\\
\sum_{i:(i,k)\in\mathcal{A}'} u^*_{ik} - \sum_{j:(k,j)\mathcal{A}'} u^*_{kj} &= w_k && \text{(by (20): } u^*_{0k} = w_k - u'_{kt} \text{ when}\\
& && j = 0 \text{ and } w_k \leq 0)
\end{aligned}$$

Since u^* is feasible for MC it can be used to obtain an upperbound on the optimal objective value of MC. Indeed $z^* \leq \sum_{a_j\in V} u^*_{0j} = \sum_{w_j>0}(w_j - u'_{jt}) = \sum_{w_j>0} w_j - \sum_{w_j>0} u'_{jt} = \sum_{w_j>0} w_j - z'$, where z' represents the optimal objective function value of MF. The last step in the inequality follows because when $k = t$ the total flow leaving t is 0, i.e. $\sum u'_{tj} = 0$, and hence $\sum_{(i,t)\in B} u'_{it} - \sum_{(t,j)\in B} u'_{tj} = \sum_{(i,t)\in B} u'_{it} = z$.

Similarly, let v be optimal for MC. We will use an equality similar to (21) to revise v so as to satisfy the property

$$v'_{0k} \leq \begin{cases} w_k & \text{if } w_k > 0\\ -w_k & \text{if } w_k \leq 0. \end{cases} \tag{21}$$

If v already satisfies (21) for $k = 1, 2, \ldots, n$ set $v' = v$. Otherwise, let k be a node at which (21) is not satisfied. Without loss of generality suppose that (21) is satisfied for every predecessor of k. Consider

the case where $w_k > 0$ (and hence $v'_{0k} > w_k$). Since the flow along $(0,k)$ exceeds the demand at k there must be an arc with positive flow f from k to some node j. So, by reducing the flow along $(0,k)$ and (k,j) and increasing the flow along $(0,j)$ we maintain flow balance and do not change the objective function value of the flow. In this fashion we can always reduce the flow into a node k so that it becomes w_k. The case where $w_k \leq 0$ is quite analogous and a similar argument shows that the flow out of node k can be changed to $-w_k$. If after adjusting the flow at node k, all nodes satisfy the property, then set $v' = v$. Otherwise select a new activity k and repeat the process. Because at each iteration we never change the flow at a predecessor of k, no node can repeat and in at most n iterations we obtain v' with the desired property (21).

Using v' let v^* be determined by:

$$v^*_{ij} = \begin{cases} v'_{ij}, & \text{if } i \neq s \text{ and } j \neq t \\ w_i - v'_{0i}, & \text{if } j = t \text{ and } w_i > 0 \\ -w_j - v'_{j0}, & \text{if } i = s \text{ and } w_j \leq 0 \\ 0, & \text{otherwise} \end{cases} \tag{22}$$

In order to show that v^* is feasible for MF we must show that it satisfies (16)-(19). Inequalities (17)-(19) are satisfied by the nonnegativity of v^* and property (21). It remains to show that flow balance is maintained at every node. This is directly analogous to the if part of the theorem.

Since v^* is feasible for MF we obtain the bound $z' \geq \sum_{a_j \in V} v^*_{jt} = \sum_{w_j>0} (w_j - v'_{0j}) = \sum_{w_j>0} w_j - \sum_{w_j>0} v'_{0j} = \sum_{w_j>0} w_j - z^*$. Inequality $z' \geq \sum_{w_j>0} w_j - z^*$ together with $z^* \leq \sum_{w_j>0} w_j - z'$ obtained earlier yields $z^* = \sum_{w_j>0} w_j - z'$. It follows that u^* is optimal for MC. Similarly, we have seen that v^* achieves value $\sum_{w_j>0} w_j - z^*$ and hence v^* is optimal for MF. This completes the proof of the theorem. □

Once we have an optimal solution u' to MF we can use (21) to obtain an optimal solution to MC and thereby a solution to the dual of BP (our original problem) in $\mathcal{O}$(n) time. Finally, notice that as long as we obtain an integer feasible solution to the maximum flow problem we will obtain integer feasible solutions to all the problems (including BP) because all weights and processing times are integer.

Example: For the example problem given above, the optimal solution u' to the maximum flow problem is $u'_{s1} = 5, u'_{s2} = 2, u'_{s3} = 13, u'_{s7} = 10, u'_{14} = 5, u'_{25} = 2, u'_{36} = 13, u'_{4t} = 5, u'_{5t} = 2, u'_{68} = 3, u'_{6t} = 10, u'_{78} = 12, u'_{8t} = 13$ and $u'_{ij} = 0$ for all other $(i,j) \in B$ (see Figure 3). Using Theorem 2

we obtain the optimal solution u^* for MC to be $u^*_{30} = 2, u^*_{14} = 5, u^*_{25} = 2, u^*_{36} = 13, u^*_{04} = 3, u^*_{68} = 3, u^*_{78} = 12, u^*_{05} = 10$ and $u^*_{ij} = 0$ for all other $(i,j) \in \mathcal{A}'$. The corresponding primal solution is then $x^*_1 = 1, x^*_2 = 1, x^*_4 = 1, x^*_5 = 1$ and $x^*_i = 0$ for all other $a_i \in V$. Recall that this is the optimal cut depicted in Figure 1. To verify optimality for the problem BP we need only check that the primal and dual objective values are the same for u^* and x^*. For the dual we have $u^*_{04} + u^*_{05}$=3+10=13 and for the primal we have $w_1 + w_2 + w_4 + w_5$=-5-2+8+12=13. Thus we have obtained the optimal solution to the instance of BP.

4 An Algorithm based on Sensitivity Analysis

In section 3 we showed that MWF can be solved in $\mathcal{O}(n^4)$ time by solving no more than n Bipartition problems BP. In this section we show that depending on the input data the worst case performance of O-MWF can be improved by using *sensitivity analysis* on the initial instance of BP to obtain solutions to subsequent instances of BP. The resulting algorithm is referred to as SMWF and its complexity is shown to be $\mathcal{O}(Wn^2+n^3)$ where $W = \sum_{i=1}^{n} |w_i|$. When W is significantly less than n^2, algorithm SMWF will outperform O-MWF. In SMWF a max-flow problem is solved to obtain the optimal solution to the initial instance of BP. Subsequent instances of BP are solved using its dual. The following definitions are used throughout this section.

H	:	A subset of the activities in $\{a_1, \ldots, a_n\}$ for which completion times have been fixed.
$\vec{v}$	:	An n-vector whose i-th component is v_i.
$\vec{w}$	:	An vector whose i-th component is the weight w_i of node i.
$\vec{w}^H$	:	The vector $\vec{w}$ whose i-th component is replaced with 0, for every $i \in H$.
$BP_H(V, \vec{v})$	:	The instance of BP corresponding to

$$(BP_H(V,\vec{v})) \qquad \max \quad \sum_{a_i \in V/H} v_i x_i \tag{23}$$

$$\textbf{s.t.} \quad x_i - x_j \geq 0, \quad \forall\ (i,j) \in \mathcal{A}\ with\ a_i, a_j \in V/H \tag{24}$$

$$1 \geq x_i \geq 0, \quad a_i \in V/H \tag{25}$$

Algorithm SMWF is based on the following observation. In each iteration of O-MWF we solve the problem $BP_\emptyset(V, \vec{w})$, fix the completion times of activities $a_h \in L$ and $a_k \in R$, update $V = V/\{a_h, a_k\}$ and repeat. In this scheme completion times of unscheduled activities are updated iteratively. An

equivalent scheme can be devised where the set H of scheduled activities is updated instead. Then, in each iteration, problem $BP_H(V, \vec{w})$ is solved, the completion times of activities a_h and a_k are fixed, and H is updated to $H \cup \{a_h, a_k\}$. Such an algorithm is described next.

Algorithm SMWF

Input : Network $\mathcal{N}$ of activities, $V = \{a_1, \ldots, a_n\}$, $H = \emptyset$, $\vec{R} = \{R_i\}_{i=1}^n$ and $\vec{D} = \{D_i\}_{i=1}^n$.

Output : Optimal schedule for MWF

Repeat

[1] Find L that solves $BP_H(V, \vec{w})$. Set $R = V/L \cup H$.

[2]$'$ Same as step [2] of O-MWF

[3]$'$ Same as step [3] of O-MWF

Until $H = \{a_1, \ldots, a_n\}$

As in O-MWF, algortihm SMWF either determines that the problem is infeasible or returns the optimal start times $\vec{R}$ and completion times $\vec{D}$. The following theorem establishes this result.

Theorem 3 *Algorithm SMWF solves MWF in $\mathcal{O}(Wn^2 + n^3)$ time.*

Proof. To establish our theorem we first need to show that: If S is optimal for $BP_\emptyset(V, \vec{w}^H)$ then S/H is optimal for $BP_H(V, \vec{w})$. Indeed, let S be an optimal solution for $BP_\emptyset(V, \vec{w}^H)$. For contradiction assume that S/H is not optimal for $BP_H(V, \vec{w})$ while L is optimal. Since L is feasible for $BP_H(V, \vec{w})$ we have $a_i \notin L$ for all $a_i \in H$. Also since S/H, L are subsets of V/H they are feasible for $BP_\emptyset(V, \vec{w}^H)$. Observe that,

$$\begin{aligned}
\sum_{i \in L} w_i^H &= \sum_{i \in L} w_i \qquad \text{(because } L/H = L\text{)} \\
&> \sum_{i \in S/H} w_i \quad \text{(because } L \text{ is optimal for } BP_H(V, \vec{w})\text{)} \\
&= \sum_{i \in S} w_i^H \qquad \text{(definition of } \vec{w}^H\text{)}
\end{aligned}$$

and hence $\sum_{i \in L} \vec{w}_i^H > \sum_{i \in S} \vec{w}_i^H$ which contradicts the optimality of S. Hence, solving $BP_H(V, \vec{w})$ is equivalent to solving $BP_\emptyset(V, \vec{w}^H)$. In any two consecutive iterations of S-MWF the BP subproblems differ only in that $w_h = w_k = 0$ for nodes h, k. Thus, once we have an optimal solution to $BP_\emptyset(V, \vec{w})$

we can solve subsequent subproblems by performing sensitivity analysis so as to drive the weights w_h and w_k down to 0, thus obtaining an optimal solution to the next subproblem. This observation together with Theorem 1 establish the correctness of SMWF.

To show that the complexity of SMWF is $\mathcal{O}(Wn^2 + n^3)$ observe that, in the first iteration of SMWF the BP subproblem can be solved as a maxflow problem in $\mathcal{O}(n^3)$ time (as in O-MWF). To solve subsequent instances of BP in line [1] of O-MWF, sensitivity analysis is used to drive the w_i values to 0. In the maximum flow problem this corresponds to decreasing the upper bound on an arc from $|w_i|$ down to 0. In the worst case this reduces the maximum possible flow through the network by $|w_i|$. A flow augmenting path in the network (if one exists) can be found in $\mathcal{O}(n^2)$ time (see Ahuja *et. al.*, 1993), finding the new optimal solution takes no more than $\mathcal{O}(|w_i|n^2)$ time. Over all nodes, this results to a processing requirement of $\mathcal{O}(Wn^2)$. Hence, complexity of SMWF is $\mathcal{O}(Wn^2 + n^3)$. This completes the proof of the theorem. □

For problems with small values of W relative to n^2, algorithm SMWF should outperform O-MWF (which has complexity $\mathcal{O}(n^4)$).

5 Solving MWF as a Mincost Flow Problem

We have already shown that MWF can be solved in $\mathcal{O}(n^4)$ by solving BP and exploiting the structure of an optimal solution. In this section we show that the dual of MWF can be solved as a minimum cost flow problem. The results in this section are of similar nature to Lawler, 1976 where the network structure of dual problems is exploited for various applications. According to Ahuja *et. al.*, 1993 the best bound for the complexity of a minimum cost flow algorithm is $\mathcal{O}(\min\{nm \log (n^2/m) \log nC,$ $nm(\log\log U) \log nC,\ (m \log n)(m + n \log n)\})$. In our application n corresponds to the number of activities, m is the size of the set $\mathcal{A}$, C is the the maximum of $\max_i d_i$ and where $U = \sum_{i=1}^{n} |w_i|$ is an upper bound on arc capacities. Thus when C and U are not extremely large it may be advantageous to solve the dual of MWF as a minimum cost flow problem. Below we discuss a preprocessing procedure for reducing the number of arcs in the minimum cost flow network.

Using arguments similar to those for MC it can be shown that the dual of MWF is equivalent to

the problem MCF stated below.

$$
\begin{array}{lll}
(MCF)\ \min & \sum\limits_{(i,j)\in A} -p_j u_{ij} - \sum\limits_{i=1}^{n}(r_i+p_i)u_{i0} + \sum\limits_{j=1}^{n} d_j u_{0j} & \\
s.t. & \sum\limits_{(i,k)\in A'} u_{ik} - \sum\limits_{(k,j)\in A'} u_{kj} = w_k, & k = 0,\ldots,n \\
& u_{ij} \geq 0, & \forall\ (i,j) \in A'
\end{array}
\tag{26}
$$

In the above formulation $\mathcal{A}'$ is as in MC. The table below gives the cost for each edge in the network and the demand at each node.

Node	Demand	Arc	Cost
a_0	$w_0 = -\sum\limits_{i=1}^{n} w_i$	(a_0, a_j)	d_j
		(a_i, a_0)	$-(r_i+p_i)$
$a_i \in V$	w_i	$(a_i, a_j) \in \mathcal{N}$	$-p_j$

Next we present the preprocessing algorithm that reduces the size of the set $\mathcal{A}'$.

Preprocessing Procedure

Input : Arc set $\mathcal{A}''$=$\mathcal{A}'$, release-dates $\vec{R} = \{R_i\}_{i=1}^n$ and due-dates $\vec{D} = \{D_i\}_{i=1}^n$.

Output : Reduced arc set $\mathcal{A}''$.

Begin

[1] Find the early-start $CPM(\vec{R}, \vec{D})$ schedule.

For $i := 1$ **to** n **do If** $r_i + p_i \leq C_i$ **then** $\mathcal{A}''$=$\mathcal{A}''$-(0,i)

[2] Find the late-start $CPM(\vec{r}, \vec{d})$ schedule.

For $i := 1$ **to** n **do If** $d_i \geq C_i + p_i$ **then** $\mathcal{A}''$=$\mathcal{A}''$-(i,0)

End

Return : $\mathcal{A}''$.

The preprocessing procedure eliminates redundant arcs that are either into or out of node 0. If an arc $(0, i)$ is removed from $\mathcal{A}''$ then the early-start of activity a_i is not determined by the release-date of activity a_i. Similarly if an arc $(i, 0)$ is removed from $\mathcal{A}''$ then the late-start of activity a_i is not determined by the due-date of activity a_i.As will be shown in theorem 4 there exists an optimal solution to MCF that uses arcs in $\mathcal{A}''$ only. Hence, once the arc set $\mathcal{A}''$ has been identified it can be used in place of A' in the minimum cost flow problem to obtain an optimal solution to MCF. This

revised problem is referred to as MCFR and appears below.

$$
\begin{array}{lll}
(MCFR)\ \min & \sum\limits_{(i,j)\in A} -p_j v_{ij} - \sum\limits_{(i,0)\in A''} (r_i + p_i) v_{i0} + \sum\limits_{(0,j)\in A''} d_j v_{0j} & \\
s.t. & \sum\limits_{(i,k)\in A''} v_{ik} - \sum\limits_{(k,j)\in A''} v_{kj} = w_k, & k = 0, \ldots, n \\
& v_{ij} \geq 0, & \forall\ (i,j) \in A''
\end{array} \tag{27}
$$

Since the preprocessing procedure takes only $\mathcal{O}(n^2)$, the complexity of the solution procedure will be determined by solving MCFR. In the next theorem we see that an optimal solution to MCFR provides an optimal solution to MCF.

Theorem 4 *If v^* is optimal for MCFR than u^* is optimal for MCF where,*

$$
u^*_{ij} \quad = \quad \begin{cases} v^*_{ij} & \text{if } (i,j) \in \mathcal{A}'', \\ 0 & \text{otherwise.} \end{cases} \tag{28}
$$

Proof. Suppose v^* is optimal for MCFR and has objective value z^*. Also let u^* be as in (28). Since v^* is feasible for MCFR with objective value z^*, u^* is feasible for MCF with objective value z^*. Hence $z' \leq z^*$, where z' is the optimal objective value for MCF. To show that u^* is optimal for MCF we show net that $z' \geq z^*$.

To that end we will construct an optimal solution u' to MCF with the property

$$
u'_{0i} = 0\ \forall\ (0,i) \in \mathcal{A}' - \mathcal{A}''\ \textit{and}\ u'_{i0} = 0\ \forall\ (i,0) \in \mathcal{A}' - \mathcal{A}'' \tag{29}
$$

Consider an arbitrary optimal solution u to MCF. If u satisfies (29) then set $u' = u$. Otherwise, there exists a node k such that either $u_{0k} > 0$ where either (i) edge $(0,k) \in \mathcal{A}' - \mathcal{A}''$ or (ii) $u_{k0} > 0$ and $(k,0) \in \mathcal{A}' - \mathcal{A}''$. Consider case (i): Let $P_k = a_0 a_{i_1} a_{i_2} \ldots a_{i_l}$ be a critical path determining the early-start time of activity a_k. Then create u' so that,

$$
u'_{ij} \quad = \quad \begin{cases} u_{ij} + u_{0k} & \text{if } (i,j) \in \{(0,i_1),(i_1,i_2)\ldots(i_{l-1},i_l)\}, \\ 0 & \text{if } (i,j) = (0,k), \\ u_{ij} & \text{otherwise.} \end{cases} \tag{30}
$$

Essentially u' is the same as u except that the flow along u_{0k} is rerouted along a longest path P_k from 0 to k. Hence u' is feasible for MCF. To see that u' is optimal we compare the objective function value

of u' to that of u. Observe that

$$\begin{aligned} z' &= \sum_{(i,j)\in A} -p_j u_{ij} - \sum_{i=1}^{n}(r_i+p_i)u_{i0} + \sum_{j=1}^{n} d_j u_{0j} && \text{(because } u \text{ is optimal)} \\ &= \sum_{(i,j)\in A} -p_j u'_{ij} - \sum_{i=1}^{n}(r_i+p_i)u'_{i0} + \sum_{j=1}^{n} d_j u'_{0j} && \text{(by definition of } u') \\ &\quad - (r_{i_1} + p_{i_1} + p_{i_2} + \ldots + p_{i_{l-1}} + p_{i_l} + r_k + p_k) \\ &= \sum_{(i,j)\in A} -p_j u'_{ij} - \sum_{i=1}^{n}(r_i+p_i)u'_{i0} + \sum_{j=1}^{n} d_j u'_{0j} && \text{(because } (0,k) \in \mathcal{A}' - \mathcal{A}'' \text{ and } u \text{ is optimal)} \end{aligned}$$

Hence, in case (i) u' obtains the optimal objective value z'. For case (ii) a symmetric argument using the path that determines the late-start time of a_k can reduce the flow along $(k,0)$ down to 0. Repeating this procedure for every node k that violates (29) we obtain a solution u' satisfying (29) in at most $2n$ iterations.

Using u' we now construct v' by setting

$$v'_{ij} = u'_{ij} \;\forall\; (i,j) \in \mathcal{A}''; \; 0 \; \textit{otherwise}. \tag{31}$$

By (29) and the optimality of u' for MCF, v' is feasible for MCFR with objective value z'. It follows that $z^* \leq z'$. Together with $z^* \geq z'$ obtained earlier this yields $z^* = z'$. Since u^* is feasible for MCF and attains value z^*, u^* is optimal for MCF. This completes the proof of the theorem. □

The previous theorem establishes that solving MCFR is equivalent to solving MCF. Since the set $\mathcal{A}''$ is a subset of $\mathcal{A}'$ the average performance of the minimum cost flow algorithms on MCFR is expected to be better than that of MCF.

6 Conclusion

We have developed several algorithms for solving MWFP. In the table below we summarize these algorithms, their worst case performance and cite the corresponding section. The following notation is used in the table: $m = |\mathcal{A}''|$, $W = \sum_{i=1}^{n} |w_i|$, and $C = \max_i d_i$.

Problem MWF assumes infinite capacity much like CPM and the NPV models for projects. In the event of resource contention among activities, the problem of allocating activities to resources, and their scheduling, is of paramount importance in project scheduling and manufacturing scheduling. Our future research is focused on this extension and uses model MWF as a building block.

Algorithm	Complexity	Section
O-MWF	$\mathcal{O}(n^4)$	3
SMWF	$\mathcal{O}(Wn^2 + n^3)$	4
MCFR	$\mathcal{O}(nm\ log(n^2/m)\ log\ nC)$	5
	$\mathcal{O}(nm\ (log\ log\ W)\ log\ nC)$	5
	$\mathcal{O}((m\ log\ n)(m + n\ log\ n))$	5

Table 1: Summary of results

References

[1] Ahuja R.K., T.L. Magnanti, J.B. Orlin *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, New Jersey, 1993.

[2] Baker, K.R. *Elements of Sequencing and Scheduling*, Dartmouth Hanover, NH, 1993.

[3] Brucker, P. *Scheduling Algorithms*, Springer Verlag, NY, 1998.

[4] Chen J. and C.-Y. Lee. General Multi-processor Tasks Scheduling, *Naval Research Logistics*, 46:57-74, 1999.

[5] Demeulemeester, E. W., W. Herroelen and P. Van Dommelen. An Optimal Recursive Search Procedure for the Deterministic Unconstrained Max-npv Project Scheduling Problem, Research Report 9603, Department of Applied Economics, K. U. Leuven, 1996.

[6] Elmaghraby, S. E. and W. S. Herroelen. The Scheduling of Activities to Maximize the Net Present Value of Projects, *European Journal of Operational Research*, 49:35-49, 1990.

[7] Goldratt, E. M. *Critical Chain*, North River Press, MA 01230, 1997.

[8] Grinold, R. C. The Payment Scheduling Problem, *Naval Research Logistics Quarterly*, 19:123-136, 1972.

[9] Herroelen, W. S. and E. Gallens. Computation Experience with an Optimal Procedure for the Scheduling of Activities to Maximize the Net Present Value of Projects, *European Journal of Operational Research*, 65:274-277, 1993.

[10] Herroelen, W. S., P. Van Dommelen and E. Demeulemeester. Project Network Models with Discounted Cash Flows a guided tour through recent developments, *European Journal of Operational Research*, 100:97-121, 1997.

[11] Horn, W. A. Single-Machine Job Sequencing with Treelike Precedence Ordering and Linear Delay Penalties, *SIAM Journal on Applied Mathematics*, 23(2):189-202, 1972.

[12] Icmeli, O. and S.S. Erenguc. A Branch and Bound Procedure for the Resource Constrained Project Scheduling Problem with Discounted Cash Flows, *Management Science*, 42(10):1395-1408, 1996.

[13] Lawler, E. L. *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

[14] Pinedo, M. *Scheduling: Theory, Algorithms and Systems*, Prentice-Hall, NY, 1996.

[15] Russell, A. H. Cash Flows in Networks, *Management Science*, 16:357-373, 1970.

[16] Sepil, C. Comment on Elmagrhaby's and Herroelen's The Scheduling of Activities to Maximize the Net Present Value of Projects, *European Journal of Operational Research*, 73:185-187, 1994.

[17] Smith, W. E. Various Optimizers for Single Stage Production, *Naval Research Logistics Quarterly*, 3:59-66, 1956.

[18] Vairaktarakis, G. L. and S. S. Syam. Minimizing Activity Costs in General Project Networks with Alternative Resources, Working Paper, 2001.

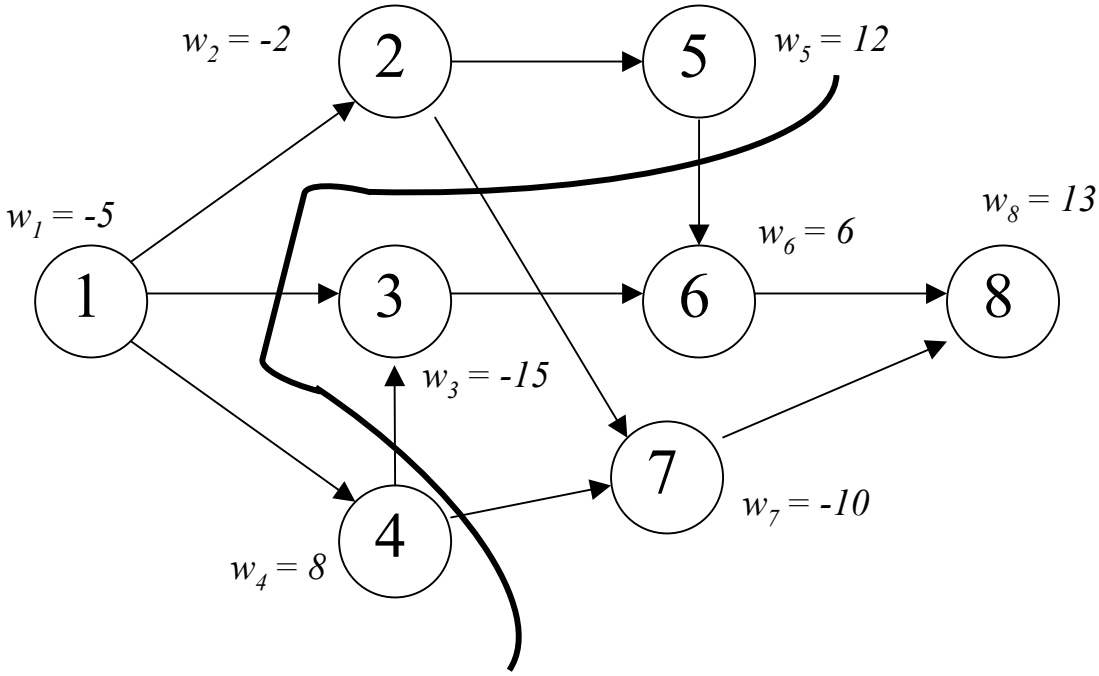

Figure 1: The network *N*.

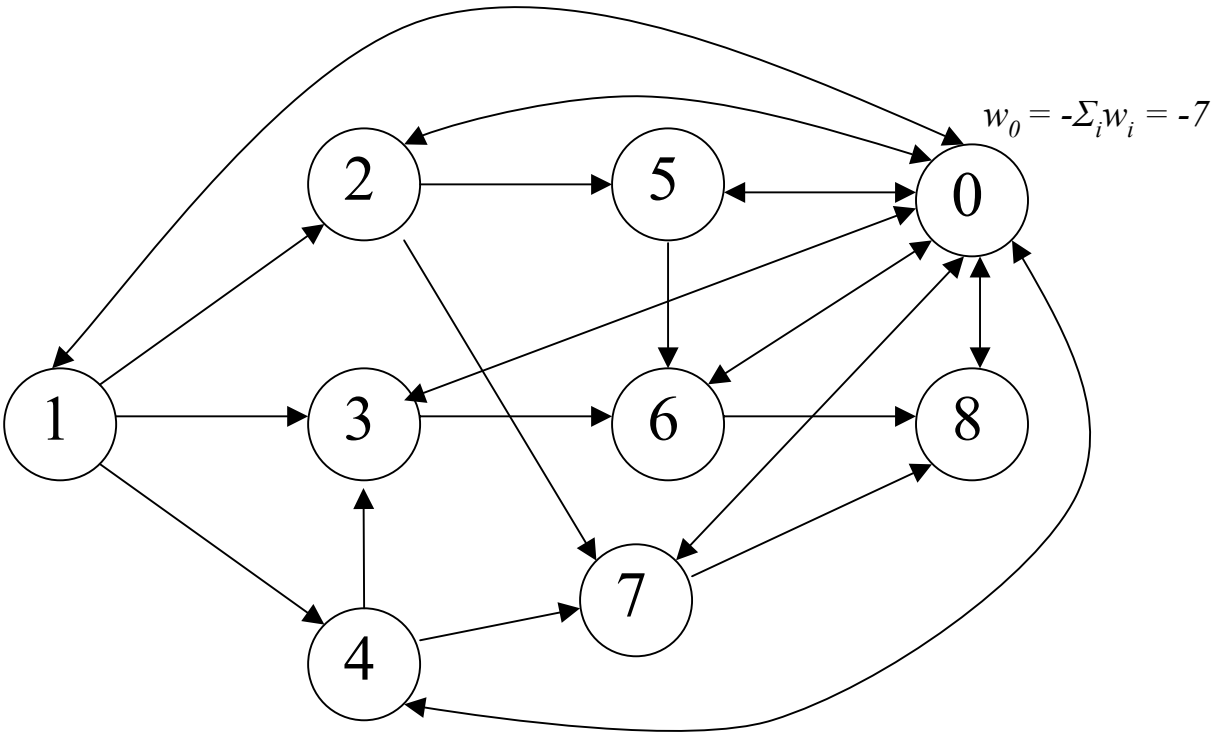

Figure 2: The network *N* with node 0.

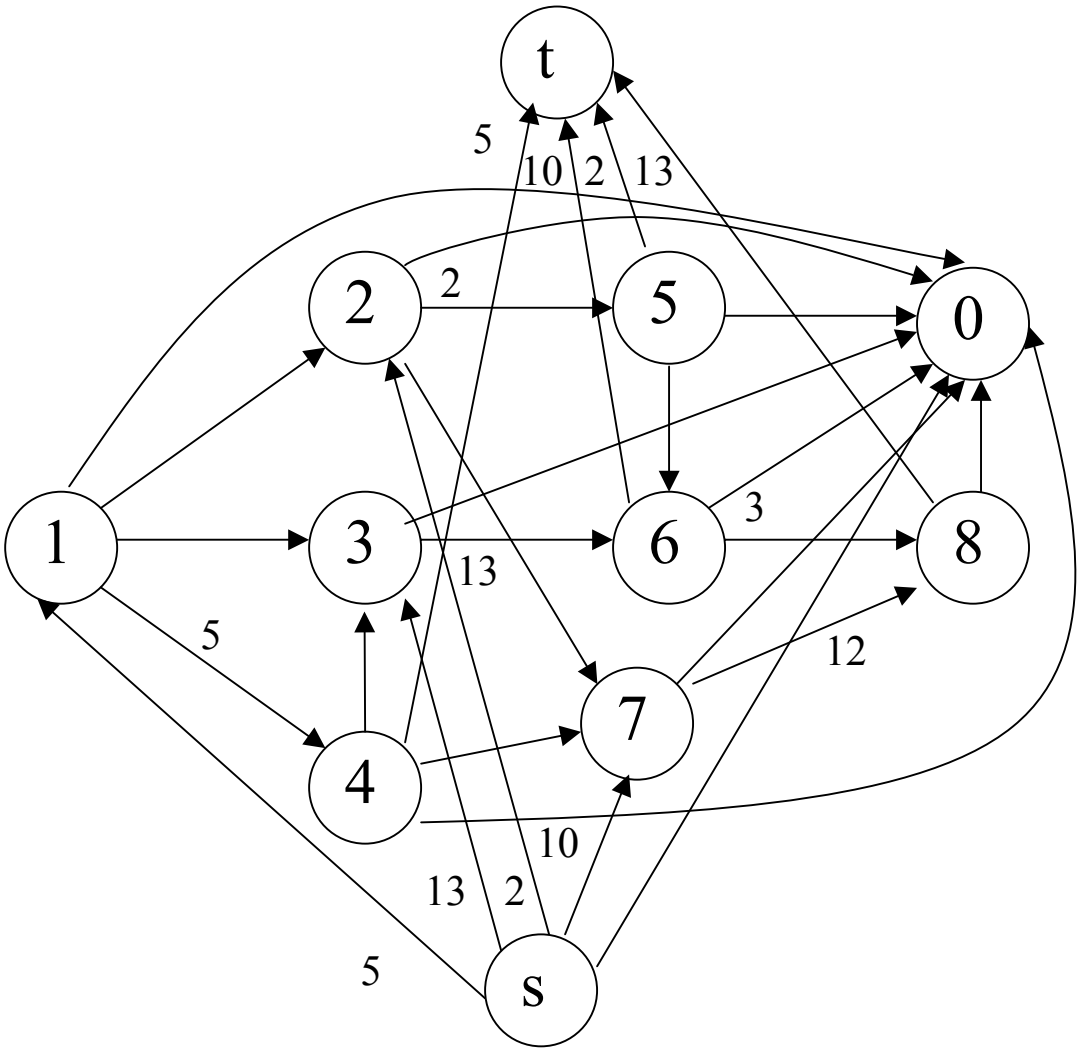

Figure 3: The maxflow network *N'*.