Proceedings of the 2007 IEEE Symposium on
Foundations of Computational Intelligence (FOCI 2007)

Task Scheduling on Flow Networks with Temporal Uncertainty

Ping Hu, Meaghan Dellar, and Chenyi Hu
Computer Science Department
University of Central Arkansas
Conway, AR 72035

Abstract

This study was motivated by the large-scale evacuations and aftermath relief efforts caused by hurricanes Katrina and Rita in 2005. During large-scale natural disasters, both demands and uncertainties on transportation networks can be pushed to the extreme.

The objective of this study is to optimally schedule tasks on a flow network that has temporal uncertainty modeled with interval valued time costs. We apply a fuzzy partial order relation for intervals to extend the Edmonds-Karp max-flow min-cost algorithm in this paper. Then, using the greedy approach, we propose task optimal schedule algorithms on a flow network with temporal uncertainties by utilizing its full capacity with the least possibility of delay in terms of the fuzzy partial order relations of intervals. In addition to the task scheduling algorithms, simple case studies are also provided in this paper.

Key words*: flow network, interval fuzzy partial order, task scheduling*

1. INTRODUCTION

1.1 Motivations of this study

The problem to be studied in this paper is to schedule multiple tasks on a flow network with temporal uncertainty. In 2005, hurricanes Katrina and Rita wrought havoc on the southern states along the Gulf coast. The anticipatory and resultant evacuation orders caused unusually high traffic volumes on the surrounding highways and interstates while many of them had significantly reduced capacity or became completely inaccessible. More roadway congestion arose in the aftermath due to the large influx of hurricane relief aid sent to cities in the Gulf coast such as New Orleans. In the wake of these hurricane disasters in the United States, it has become apparent that further studies need to be done for scheduling multi-tasks in a flow network with uncertainties. During large-scale natural disasters, both demands and uncertainties on transportation networks can be pushed to extreme. However, in general, uncertainties are often associated with networks. For example, the demands on a roadway may vary; and traffic delays can be cause by road constructions, traffic accidents, etc. Therefore, the studies in this paper may have much broader applications.

1.2 Existing model and algorithms

There is a large amount of literature pertaining to the study of flow networks, task scheduling, and multi-objective optimization with uncertainties. Weighted graphs are often used to model networks. Algorithms developed by Dijkstra [3], Bellman-Ford [2], Floyd-Warshall [5] and others find shortest paths in a connected weighted network. The topological ordering of a directed acyclic graph can be used to schedule tasks that have dependencies. The Ford-Fulkerson algorithm [6] finds the maximum flow in a network with single- source and destination, and satisfies capacity constraints of the network. When studying uncertainties associated with scheduling algorithms, in addition to classical statistic, probabilistic and stochastic approaches, new technologies such as fuzzy logic [17], interval computing [16], and genetic algorithms have been applied. All of these provide us with the necessary background knowledge for this study.

1.3 Objectives of this study

This study involves scheduling multiple tasks with temporal constraints on a flow network. A task t on a flow network is a job to be sent from a source S to a destination D. To complete a task, it requires available capacity of the network and also consumes time. Since the time cost may involve uncertainty, in this study, we apply intervals to model temporal uncertainty in a flow network. The objective of this study is to fully utilize the capacity of a flow network and simultaneously minimize the possibility of overall delay.

The rest of this paper is organized as the following. In section 2, we introduce the model of flow network with temporal uncertainty in this paper. In section 3 we briefly review related background knowledge including Ford-Fulkerson algorithm, Edmonds-Karp algorithm, and interval fuzzy partial order relations. In section 4, we investigate task scheduling algorithms on a flow network with temporal uncertainty with sample case studies. We conclude this paper with section 5.

2. FLOW NETWORK WITH TEMPORAL UNCERTAINTY

2.1 Flow network

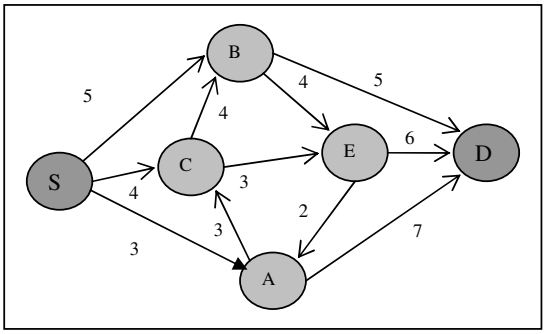


Figure 1: A sample flow network

In this study we use the flow network model proposed by Ford and Fulkerson in [6] (1962). In that model, a flow network is a directed graph G = (V, E) without parallel edges. There are two distinguished nodes: S (source) and D (sink) in the network. For each edge e ∈ E, there is an constant c(e) that represent the maximum allowable flow capacity on that edge in the given direction. Figure 1 above is a sample flow network. The flow capacity from the node B to the node E is 4.

An S-T flow is a function *f* that satisfies the capacity and conservation constraints as described below:

- $\forall e \in E: \quad 0 \leq f(e) \leq c(e)$
- $\forall v \in V\backslash\{S, D\}: \quad \sum_{in} f(e) = \sum_{out} f(e)$

The value of a flow is the sum of all outgoing flow f(e) from the source S.

Ford and Fulkerson proposed their well-known algorithm to find the maximum flow [6]. Edmonds and Karp then improved overall efficiency of the Ford-Fulkerson algorithm [4]. By attaching another constant to an edge in a flow network to model the cost of a flow on that edge, Edmonds and Karp proposed their maximum flow and minimum cost algorithm [4] (1972). We review these algorithms in the section 3 and then extend them in the section 4 of this paper.

2.2 Flow network with temporal uncertainty

In real world applications, time is consumed when a flow passes from one vertex to another through a link (or path) in a flow network. Therefore, in scheduling tasks on a flow network we should take time into consideration. Temporal costs have characteristics that are significantly different from monetary cost in addition to irreversibility and compatibility. For example, the monetary cost for shipping cargo from A to B can be related to its load. However, the time needed for driving an 18 wheeler and a light sedan from A to B may have no significant difference at all. Another special characteristic of temporal cost on the flow network can be its uncertainty. For example, the driving time from A to B may not be exactly two hundred minutes but mostly between 190 and 210 minutes pending on the road conditions or the departure time. To model this kind of uncertainty, we use intervals for time consumptions on a flow network in this study.

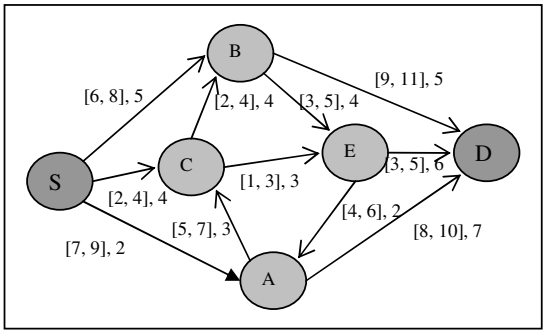


Figure 2: A flow network with temporal uncertainty

The term *interval* used in this paper means a set of all real numbers between its lower (left) and upper (right) bounds provided. If the left and right endpoints are equal, the interval is *trivial* and is the same as a real number. In the rest of this paper we will use boldfaced letters to denote intervals. The left and right endpoints of an interval **a** are specified by the same latter but with subscripts L and R, respectively. Hence, an interval $\mathbf{a} = [a_L, a_R] = \{x \in \Re : a_L \leq x \leq a_R\}$ with its lower and upper bounds as a_L and a_R, respectively.

Figure 2 is a sample flow network with uncertain cost specified by intervals associated with the links.

3. RELATED BACKGROUND KNOWLEDGE

To schedule tasks on flow networks with interval temporal uncertainty, we need related flow network algorithms and interval partial order relations. We review them briefly in this section.

3.1 The task scheduling problem

A task *t* on a flow network is a job to be sent from the source S to the destination D. To simplify our study, we consider only the required capacity resource, f_t, and arrival deadline d_t associated with t.

As mentioned earlier in this paper, the objective of this study is to schedule tasks on a flow network with temporal uncertainty that fully utilizes the capacity of the network and minimizes the possibility of total delay.

Let J be a collection of such tasks to be scheduled. The task scheduling problem is to:

- schedule all tasks in J on a given flow network and meet all temporal constraints if it possible;
- otherwise, schedule tasks in J within the maximum flow capacity and minimizing the overall possible delay.

3.2 Ford-Fulkerson and Edmonds-Karp algorithms

Ford and Fulkerson reported their work on finding a maximum flow on a flow network. The key concept of their algorithm is called the augmented path.

Let N be a flow network, which is specified by a graph G, its capacity function c, source S, and sink D. Furthermore, let f be a flow of N. Given an edge e = (u, v) of G directed from a vertex u to another vertex v, the residual capacity from u to v, forward direction, with respect to the flow f, denoted by $\Delta_f(u,v)$, is defined as

$$\Delta_f(u,v) = c(e) - f(e)$$

And the residual capacity from v to u, in the backward direction of the edge (u, v) is defined as

$$\Delta_f(v,u) = f(e)$$

The residual capacity is defined as

$$\Delta_f(e) = \begin{cases} c(e) - f(e) & \text{if } e \text{ is a forward edge} \\ f(e) & \text{if } e \text{ is a backward edge} \end{cases}$$

Let π be a path from S to D that is allowed to traverse edges in either the forward or backward direction. The residual capacity $\Delta_f(\pi)$ of a path π is the minimum residual capacity of its edge, that is,

$$\Delta_f(\pi) = \min_{e \in \pi} \Delta_f(e)$$

If $\Delta_f(\pi) > 0$, then π is called an augment path. A value of total flow then can be increased by adding the minimum residual capacity on each forward edge and subtract it from every backward edge in the augment path. By exhaustively finding augment paths, one may increase the total flow to the maximum within the capacity constraints. This results, in pseudo code, the Ford-Fulkerson's maximum flow algorithm as the follow:

```
Algorithm FordFulkersonMaxFlow(N)
   for all  e ∈ G.edges()  setFlow(e, 0)
   while G has an augmenting path p
      {compute residual capacity D of p }
      D ← ∞
      for all edges e ∈ p{compute residual
         capacity d of e }
        if e is a forward edge of p
           d←getCapacity(e)-getFlow(e)
        else {e is a backward edge }
            d ← getFlow(e)
        if d < D
            D ← d
       {augment flow along p }
      for all edges e ∈ p
        if e is a forward edge of p
            setFlow(e, getFlow(e) + D)
        else {e is a backward edge }
            setFlow(e, getFlow(e) - D)
```

Edmonds and Karp attached a positive cost constant on each edge of a flow network. Let C be a cycle, with both forward and backward edges, in a maximum flow network. For each e ∈ C, one can find its residual capacity as usual. The product of the residual capacity and the cost on the edge is called the *residual cost*. The sign of a residual cost is determined as the follows: positive if the edge is forward; otherwise negative. If the total product on the cycle is negative, the cycle is called an *augmented cycle*. By adjusting flow on each edge of the augment cycle, the total cost will be reduced without effect the total flow. The Edmonds-Karp maximum flow and minimum cost algorithm in pseudo code is as the follow:

```
Algorithm EdmondsKarpAlgo(N)
Find max-flow FordFulkerson's algorithm
while G has an augmenting cycle C
    for all edges e ∈ C
        if e is a forward edge
            setFlow(e, getFlow(e) + D)
        else {e is a backward edge }
            setFlow(e, getFlow(e) - D)
```

3.3 Intervals and their fuzzy partial order relations

Interval computing was proposed by Moore [16] in the late of the 1950's. In addition to the above described left and right endpoints representations, an interval can be represented by its midpoint and its radius as well. The midpoint of an interval $\mathbf{a} = [a_L, a_R]$ is denoted by $m(\mathbf{a}) = (a_L + a_R)/2$; and the radius of $\mathbf{a}$ is denoted by $r(\mathbf{a}) = (a_R - a_L)/2$. Hence, an interval $\mathbf{a} = [a_L, a_R] = [m(\mathbf{a}) - r(\mathbf{a}), m(\mathbf{a}) + r(\mathbf{a})]$.

Basic binary interval arithmetic operations for intervals, $\mathbf{a} = [a_L, a_R]$ and $\mathbf{b} = [b_L, b_R]$ are defined as:

$\mathbf{a} + \mathbf{b} = [a_L + b_L, a_R + b_R]$

$\mathbf{a} - \mathbf{b} = [a_L - b_R, a_R - b_L]$

$\mathbf{a} \ominus \mathbf{b} = [a_L - b_L, a_R - b_R]$ provided $r(\mathbf{a}) \geq r(\mathbf{b})$

$\mathbf{a} * \mathbf{b} = [\min(a_L b_L, a_L b_R, a_R b_L, a_R b_R), \max(a_L b_L, a_L b_R, a_R b_L, a_R b_R)]$

$\mathbf{a} / \mathbf{b} = [\min(a_L/b_L, a_L/b_R, a_R/b_L, a_R/b_R), \max(a_L/b_L, a_L/b_R, a_R/b_L, a_R/b_R)]$, provided that $0 \notin \mathbf{b}$.

The above interval arithmetic definitions also imply the following properties:

$m(\mathbf{a} + \mathbf{b}) = m(\mathbf{a}) + m(\mathbf{b})$

$m(\mathbf{a} - \mathbf{b}) = m(\mathbf{a}) + m(\mathbf{b})$

$m(\mathbf{a} \ominus \mathbf{b}) = m(\mathbf{a}) - m(\mathbf{b})$

There is a very rich literature in interval analysis far beyond the scoop of this paper. For more information about interval computing, interested readers may refer the comprehensive website maintained by Kreinovich [11].

In this study, we are interested in the relations of time intervals. In studying temporal relationships of two time intervals, Allen listed 13 possible cases [1] (1983) qualitatively. Krokhin et al further studied the relations in [14] (2003) and indicated that the relations between intervals could be $2^{13} = 8192$ possible unions of the 13 basic interval relations. For readers' convenience, here we list Allen's 13 basic cases between two time intervals in Figures 3-9:

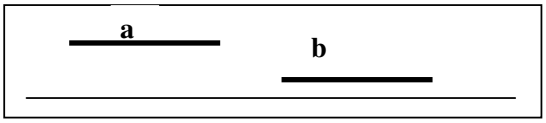


Figure 3: **a** precedes **b**, and **b** preceded by **a**

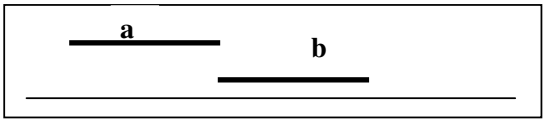


Figure 4: **a** meets **b,** and **b** met by **a**

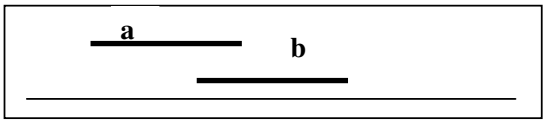


Figure 5: **a** overlaps **b**, and **b** overlapped by **a**

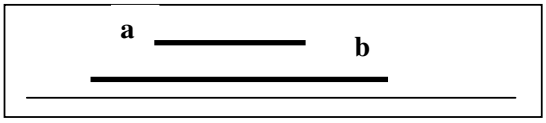


Figure 6: **a** during **b,** and **b** includes **a**

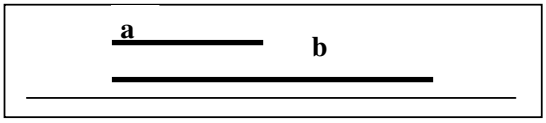


Figure 7: **a** starts **b,** and **b** started by **a**

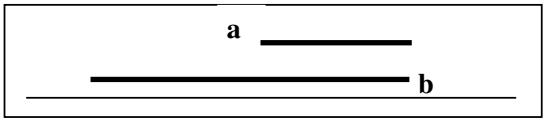


Figure 8: a finishes **b,** and **b** finished by **a**

Figure 9: **a** equals **b**

The above relations are qualitative. However, we need a way to quantitatively compare time intervals in a flow network with interval valued temporal uncertainty. In other words, if **a** and **b** are two intervals, we need to be able to determine if '**a** is less than **b**'. This calls for a partial order relationship between intervals. As we can see that in the figures the above, the statement '**a** is less than **b**' could not be represented in traditional binary logic anymore. For example, in Figure 6, there are some a $\in$ **a**, and b $\in$ **b**, such that a < b; and there are also some a' $\in$ **a**, and b' $\in$ **b**, such that a' > b'. Therefore, the statement '**a** is less than **b**' could be both true and false but in different degrees.

Applying fuzzy logic proposed by Zadeh [17], we defined an interval binary operator in [12] that returns a real number between zero and one as the follow:

Definition 1: Let $\prec$ be a binary operator for two nontrivial intervals **a** = (a_L , a_R) and **b** = (b_L , b_R) such that ($\mathbf{a} \prec \mathbf{b}$) =

1	if $a_R < b_L$
1^-	if $a_L \le b_L \le a_R < b_R$
0.5	r (**b**) = r (**a**) and $a_L = b_L$
$(b_R - a_R)/ 2[r(\mathbf{b}) - r(\mathbf{a})]$	if $b_L \le a_L < a_R \le b_R$, and r (**b**) > r (**a**)

Note: The definition above also works when **a** and **b** are trivial intervals. When both of them are trivial intervals, i.e. $a_L = a_R$ and $b_L = b_R$, the definition returns 1 if $a_R < b_L$, and 0.5 if $a_L = b_L$. It is in consistent with the ordering relation of real numbers. When only one of them is trivial, say **a** is trivial, the definition returns appropriate fuzzy memberships as well.

In this definition, it clearly shows us when **a**'s right point is less than **b**'s left point, **a** is always less than **b** without any questions, so it has the membership 1. When $a_L \le b_L$ and $a_R < b_R$, except the overlap [b_L, a_R], all points in **a** are less than points in **b**. Also, any point *x* in the overlap [b_L, a_R] is in both **a** and **b**. A point *x* is not less than itself. Therefore, it is reasonable to believe that **a** is still less than **b**, so it has the membership 1^-. If r (**b**) = r (**a**) and $a_L = b_L$, then **a** is the same as **b**. Hence, **a** is equally less and greater than **b. S**o the memberships should be the same as 0.5. When $b_L \le a_L < a_R \le b_R$ that means **a** is enclosed in **b**, so for the statement '**a** is less than **b**' the membership function that returns one when $a_L = b_L$ and $a_R < b_R$, and zero when $a_L > b_L$ and $a_R = b_R$.

Using the above definition, we can also define another binary operator for intervals:

Definition 2: Let $\succ$ be a binary operator for two nontrivial intervals **a** = (a_L, a_R) and **b** = (b_L, b_R) such that ($\mathbf{a} \succ \mathbf{b}$) = 1 – ($\mathbf{a} \prec \mathbf{b}$).

We have proved that the above binary operators are in fact are fuzzy partial order relations for intervals. We have also shown that the above two binary operators are continuous except only at one single point. Using them as fuzzy membership functions, we can determine if '**a** is less (or greater) than **b**' for two intervals. For more details about the fuzzy partial order relations, please refer to our paper [12] which appears in the same proceedings containing this paper.

Applying the fuzzy partial order relations we have also successfully extended shortest path and minimum spanning tree algorithms in [12].

4. TASK SCHEDULING

In this section we discuss task scheduling on a flow network with temporal uncertainties.

4.1 Flow with minimum interval temporal cost

As we have defined the task scheduling problem before, an immediate question one may have is what are the necessary conditions and sufficient conditions if all tasks in J can be scheduled on the network and meet all constraints. There are two types of constraints in the scheduling problem. One is the capacity constraint, and the other is the time constraints. It is obvious that, to be able to schedule all tasks, the total capacity required must be no more than the maximum flow capacity of the network. However, it is not that straight forward to find the conditions to meet the time constraints especially when there are temporal uncertainties associated with a

flow network. In order to do this, we extend the Edmonds-Karp algorithm to study minimum interval valued temporal costs.

Let f be a flow on a network G = (V, E). We define the time cost of the flow f, T(f), as the following:

$$T(f) = \sum_{e \in E} T(e) f(e)$$

where T(e) is the temporal cost of a flow f on the edge e. Among all flows that have the same value |f|, f* is said to be minimum temporal cost if T(f*) = min T(f).

To find a minimum temporal cost flow with given flow value, we can apply the concept of augment cycle described in Section 3 of this paper. However, we need to notice that the temporal cost on a edge T(e) is an interval now due to temporal uncertainty.

Let C be a cycle, with both forward and backward edges, in a value |f| flow network. For a forward edge e ∈ C, its *residual cost interval*, R(e), is the product of its capacity residual and time cost interval. If e is a backward edge, then its *residual cost interval* is the negative of the product. The residual cost interval of the cycle C, R(C), is the sum of the residual cost interval of all edges on C, i. e., R(C) = Σ R(e) for all e ∈ C. Through interval computing, we can find its *residual cost interval* for the cycle C. By comparing R(C) with zero as defined in Definition 1, R(C) ≺ 0, we obtain the fuzzy membership of that the cycle C is a fuzzy *augment cycle*. By repeatedly adjusting flow on each edge of all fuzzy augment cycles of fuzzy membership greater than 0.5, the possible total time cost will approach to its minimum while maintaining the value of the flow. We summarize the above discussion as:

Theorem 1: A maximum capacity flow f has possible minimum time cost among all flows of value | f | if and only if there is no augment cycle with fuzzy membership greater than 0.5.

Proof:

To prove the "if" part, we assume that f is a maximum flow but does not have a possible minimum time cost. Then, using the extended Edmonds-Karp algorithm, one can obtain a maximum flow with the possible minimum time cost. Therefore, there exists at least one fuzzy augmenting cycle with fuzzy membership greater than 0.5. This contradicts with the assumption that there is no augment cycle with fuzzy membership greater than 0.5.

To prove the "only if " part, suppose that there is an augmenting cycle of negative time cost with respect to f, then we can obtain another flow g by this augmenting cycle with value | f |, but g has less time cost than f, which is a contradiction.

Our goal is to shift flow from possible more time cost paths to less time cost paths until there is no such path available. If there is a flow that can be possibly shifted from a path p_1 to another path p_2 and reduces possible time cost, then p_1 and p_2 form an augment cycle with membership more than 0.5. Hence, by exhaustively shifting flow on fuzzy augment cycles with membership more 0.5, one approaches the possible minimum time cost flow. However, in practice, one may want to use an appropriate α-cut to obtain a reasonable approximation of the possible minimum time cost flow.

4.2 Task scheduling

The flow with possible minimum temporal cost discussed above paves the road for scheduling tasks that have a required capacity that is no greater than the allowable maximum flow of the network. For a given set of tasks J, assume its total capacity requirement is less than or equal to the max-flow of the network. We schedule them to meet the capacity constraint first, and apply residual cycles to find the possible minimum time cost of the entire flow. We then attempt to sort the temporal costs for each path for the flow, using the upper bounds of the time interval so we can determine the worst-case cost. (Note: This is a conservative approach. One may sort the path according to the lower bounds or the midpoints as an aggressive or average approach.) Let P = (p_1, p_2, p_3…p_m) be paths from S to D of the flow with minimum total time costs, and T(p_i) be the time cost associated with the path p_i. We arrive at:

$$T(p_1) \le T(p_2) \le \ldots \le T(p_m), \quad \text{and}$$

$$f(p_1) + f(p_2) + \ldots + f(p_m) = |f|.$$

Then, we can schedule the tasks according to their sorted deadlines on these sorted paths.

We illustrate the above idea with the example below:

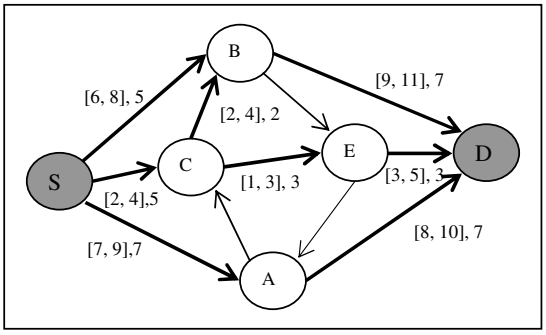


Figure 10: Minimum temporal cost flow and paths

In the figure above we have found the minimum time cost flow of volume 17 (in fact it is the maximum flow of the network.) Then using the Dijkstra's algorithm we find

p_1: S → C → E → D with T (p_1) = [2,4] + [1,3] + [3,5] = [6,12] and f (p_1) = 3.

p_2: S → C → B → D with T (p_2) = [2,4] + [2,4] + [9 , 11] = [13,19] and f(p_2) = 2.

p_3: S → B → D with T (p_3) = [6 , 8] + [9 , 11] = [15, 19] and f(p_3) = 5.

p_4: S → A → D with T (p_4) = [7 , 9] + [8 , 10] = [15 , 19] and f (p_4) = 7.

We list them in the table below:

	Path	Possible time cost on the path	Flux on the path
p_1	S-C-E-D	[6, 12]	3
p_2	S-C-B-D	[13, 19]	2
p_3	S-B-D	[15,19]	5
p_4	S-A-D	[15,19]	7

Suppose we have n tasks that need to be scheduled with capacity requirements and deadlines. We sort the tasks according to their deadlines, and get a series of tasks t_1, t_2, …, t_n. The task i has a deadline $d(t_i)$ for i = 1, 2, …, n and $d(t_1) \le d(t_2) \le \ldots \le d(t_n)$.

Finally, we schedule the tasks with the greedy approach by assigning the tasks with earliest deadline to the path with minimum time costs.

For example , suppose we have three tasks that need to be scheduled on the flow in Figure 10. They are task1 with $f(t_1) = 5$ and deadline $d(t_1) = 21$, task2 with $f(t_2) = 2$ and deadline $d(t_2) = 14$, task3 with $f(t_3) = 6$ and deadline $d(t_3) = 23$. Then as we discussed the above we sort the tasks by deadline and we get $d(t_2) < d(t_1) < d(t_3)$. So we first assign task2 on path1 which is the shortest time cost path available. Now path1 only has one unit available. Then we assign task1 on path1 with 1 unit and on path 2 with 2 units and on path 3 with 2 units. At last we assign task 3 on path3 with 3 units and on path4 with 3units. Now all tasks have been scheduled and meet the time constraint. We summarize the above as a scheduling algorithm:

```
Algorithm TaskScheduling

  On a minimum time cost flow network:
  Sort m paths such that
    T(p1) ≤ T(p2) ≤ … ≤ T(pm),
  Sort n jobs by deadline such that
   d(t1) ≤ d(t2) ≤ … ≤ d(tn)
  For j = 1 to n
     Assign job j to the path with shortest
    time cost still have free capacity. If
    the path does not have enough flow
    capacity, use the next path.
```

Now we have a scheduling algorithm for multiple tasks on a flow network with temporal uncertainty. Here we list conditions for which all constraints can be satisfied in the scheduling:

1. Let the maximum flow capacity of the network be $|f_{max}|$. The necessary condition on the capacity is:
$$f(t_1) + f(t_2) + \ldots + f(t_n) \le |f_{max}|.$$
2. Let t_i be a task being scheduled on path p_j. The necessary temporal condition is that for all i = 1, 2, …n and j = 1, 2, …, m,
$$T(p_j) < d(t_i).$$
3. There is sufficient capacity for paths to schedule all tasks and meet condition 2.

If all of the above conditions can be met then we say tasks are satisfactorily schedulable on the network crisply. Otherwise, the tasks may only be scheduled without a guarantee of satisfying all constraints. For example, when $d(t_i)$ falls in the interval of $T(p_j)$ then t_i has a possibility to meet or to miss its deadlines with different degrees due to temporal uncertainty of the network.

Hence, we need to define a function to evaluate the uncertainty. Let f_{ij} be the flow for task i running on path j and the upper bound of $T(p_j)$ be $T_R(p_j)$. The fuzzy membership of $d(t_i)$ less than $T(p_j)$, meeting the deadline, is $d(t_i) \succ T(p_j)$ as defined in Definition 2. Let $E_{ij} = f_{ij} * [T_R(p_j) - d(t_i)] * [d(t_i) \succ T(p_j)]$. For example if task t_i has deadline 15 and the path j task i running has time cost [13, 17] and the flow running on this path is 3 units, then $E_{ij} = 3 * (17 - 15) * (15 \succ [13,17]) = 3 * 2 * 0.5 = 3$. Then, the scheduling objective is to maximize ΣE_{ij}. It can also been defined as a dual minimization problem.

4.3 An alternative approach for task scheduling

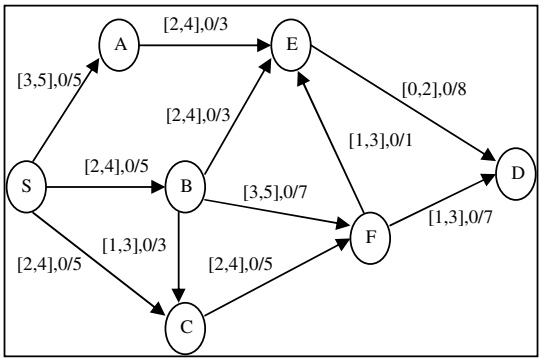
The schedule algorithm discussed above works when the total capacity requirement is no more than the maximum capacity of the flow network. Instead of using

```
Algorithm MinCostFlow(N):
 Input: Weighted flow network N = (G,S,w,D,t)
 Output: A maximum flow with minimum cost f for N
 for each edge e ∈ N do
  f(e) ← 0
 for each vertex v ∈ N do
  d(v) ← 0
 stop ← false
 repeat
  compute the weighted residual network R_f
  for each edge (u,v) ∈ R_f do
     w'(u,v) ← w(u,v) + d(u) – d(v)
  for each edge (u,v) ∈ R_f do
     w'(u,v) ← w(u,v) + d(u) – d(v)
  run Dijkstra's algorithm on R_f using the weights w'
  for each vertex v ∈ N do
     d(v) ← distance of v from s in R_f
  if d(t) < +∞ then
     {π is an argumenting path with respect to f}
     {Compute the residual capacity Δf (π) of π}
     Δ ← +∞
     for each edge e ∈π do
     if Δf (e) < Δ then
     Δ ← Δf (e)
     {Push Δ ← Δf (π) units of flow along path π}
     for each edge e ∈π do
           if e is a forward edge then
             f(e) ← f(e) +Δ
           else
             f(e) ← f(e) – Δ        {e is a backward edge}
 else
   stop ← true {f is a maximum flow of minimum cost}
```

Ford-Fulkerson's algorithm to find the maximum flow first, Edmonds and Karp proposed the algorithm below that effectively finds maximum flow and minimum cost on a flow network with successive shortest path approach. Here is the pseudo code for non-interval costs.

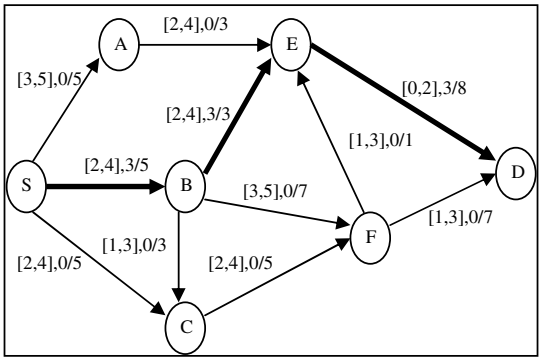
Similar to our previous discussion, we can use the extended interval Dijkstra's algorithm [12] and find the maximum flow with minimum total time cost.

Here is an example:

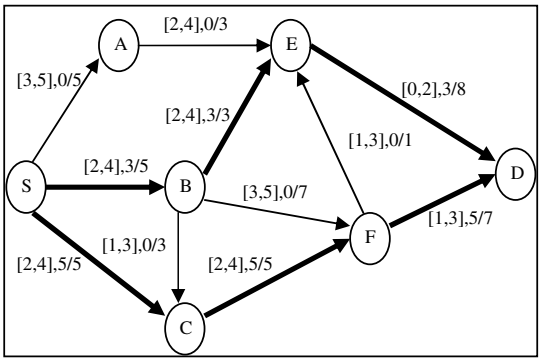


Suppose we have a network as the above. In the network each edge is labeled with T(e), f(e)/c(e).

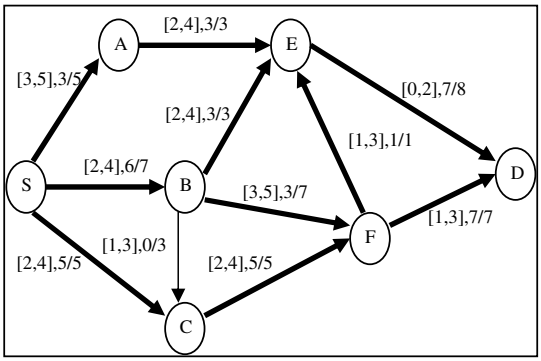
Using the successive shortest path algorithm we first find the shortest argument path S→B→E→D and the saturated edge of this path is BE. So, we add 3 units of flow on this path and get:



Then we find the next shortest argument path is S→C→F→D and the saturated edge of this path is SC and CF. So, we add 5 units of flow on this path then we get



We continue until we get the minimum time cost max-flow.



The maximum flow with minimum time cost of the example can be described in the table below:

Modify times	Shortest path	Cost of shortest path di	Flux of shortest path xi	The saturated path (i--j)
1	S-B-E-D	[4,10]	3	BE
2	S-C-F-D	[5,11]	5	SC and CF
3	S-A-E-D	[5,11]	3	AE
4	S-B-F-D	[6,12]	2	FD
5	S-B-F-E-D	[6,14]]	1	FD

After we obtained the maximum flow with minimum interval cost, we can then schedule the tasks as described in section 4.3. When the total capacity required is more than the maximum flow, one should establish a scheme to prioritize all the tasks and then keep the capacity requirement of to be scheduled tasks to no more than the maximum flow.

5. CONCLUSIONS AND FURTURE WORK

In this paper, we studied algorithms that schedule tasks on a capacity flow network with capacity and uncertain temporal constraints.

We modeled temporal uncertainty with time intervals. By applying fuzzy partial order relations for intervals, we extended Edmonds-Karp maximum flow and minimum cost algorithms for task scheduling. We discussed the necessary conditions and sufficient conditions to satisfy all scheduling constraints. Algorithms for crisply and fuzzily schedulable tasks are presented in Section 4.

In this study both tasks and flow networks are static. However, in practice, they can be dynamic. Scheduling a dynamic collection of tasks on a dynamic flow network can be more challenge and more interesting.

Acknowledgement: This research is partially supported by NSF Grant CCF-0202042.

Two student co-authors of this paper have been supported by the university graduate assistantship and Arkansas State Undergraduate Research Fellowship.

REFERENCES

[1] Allen, J., *Maintaining knowledge about temporal intervals*, ACM Communications 26, pp.832–843, 1983.

[2] Bellman, R. *On a routing problem*, Quarterly of Applied Mathematics, 16(1), pp.87-90, 1958.

[3] Dijkstra, E. W. *A note on two problems in connexion with graphs*. Numerische Mathematik, Vol. 1, pp. 269-271, 1959.

[4] Edmonds J. and Karp R., *Theoretical improvements in the algorithmic efficiency for network flow problems*, J. of ACM, 19, pp. 248-264, 1972.

[5] Floyd, R. W. *Algorithm 97: Shortest path*, Communication of ACM, Vol. 5, No. 6, pp. 345, 1962.

[6] Ford, L. R., Fulkerson, D. R. *Flows in Networks*, Princeton University Press, 1962.

[7] Fishburn, P. C. *Interval Orders and Interval Graphs: A study of Partially Ordered Sets*, Wiley, New York, 1985.

[8] Gilmore, P. C. and Hoffman, A. J. *A characterization of comparibility graphs and interval graphs*, Canadian Journal of Mathematics, 16, pp.539-548, 1964.

[9] Golumbic, M. C. *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, 1980.

[10] Goodrich, M. and Tamassia, R. *Algoritm Design*, John Wiley & Sons, 2002.

[11] Interval Computations, http://www.cs.utep.edu/interval-comp/main.html

[12] Hu, P. and Hu, C. *Fuzzy partial-order relations for intervals and interval weighted graphs*, Proceedings of IEEE 2007 Symposium on Foundations of Computational Intelligence, Honolulu, HI, 2007.

[13] Klein, G.; Moskowitz, H.; Ravindran, A; *Interactive multi-objective optimization under uncertainty*, Management Science; 36, pp. 58-75, 1990.

[14] Krokhin A., Jeavons P., and Jonsson P. Reasoning about temporal relations: the tractable subalgebras of Allen's interval algebra, J. of the ACM, 50, pp. 591–640, 2003.

[15] Kruskal, J. B. *On the shortest spanning subtree and the traveling salesman problem*, Proceedings of the American Mathematical Society 7, pp.48–50, 1956.

[16] Moore , R. E. *Method and Application of Interval Analysis*, SIAM, Philadelphia, 1979.

[17] Zadeh, L.A. The Concept of a Linguistic Variable and Its Application to Approximate Reasoning I, II, III. Information Sciences, 8, pp. 199-251; 9, pp. 43-80, 1975.