Budget-Constrained Bulk Data Transfer
via Internet and Shipping Networks*

Brian Cho and Indranil Gupta
Department of Computer Science
University of Illinois at Urbana-Champaign
{bcho2, indy}@illinois.edu

ABSTRACT

Cloud collaborators wish to combine large amounts of data, in the order of TBs, from multiple distributed locations to a single datacenter. Such groups are faced with the challenge of reducing the latency of the transfer, without incurring excessive dollar costs. Our Pandora system is an autonomic system that creates *data transfer plans* that can satisfy latency and cost needs, by considering transferring the data through both *Internet and disk shipments*. Solving the planning problem is a critical step towards a truly autonomic bulk data transfer service. In this paper, we develop techniques to create an optimal transfer plan that *minimizes transfer latency subject to a budget constraint*. To systematically explore the solution space, we develop efficient binary search methods that find the optimal shipment transfer plan. Our experimental evaluation, driven by Internet bandwidth traces and actual shipment costs queried from FedEx web services, shows that these techniques work well on diverse, realistic networks.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks

General Terms

Algorithms

Keywords

cloud computing, wide-area data transfer, data-intensive computing

---

1. INTRODUCTION

Cloud computing has the potential to allow groups of collaborators to collect datasets and run computations in an ad-hoc manner. However, a significant obstacle to such collaboration is the high cost and long latency involved in exchanging large datasets from distributed locations. Paying for high bandwidth links can be expensive for cloud providers, which is passed on to the user as data transfer rates. In addition, with current Internet bandwidths, sending large datasets is excessively time consuming. For example, sending a 1 TB forensics dataset from Boston to the Amazon S3 storage system cost $100 and took several weeks [11].[1] To remedy this problem, we have previously proposed the broad approach of transferring data according to a *cooperative transfer plan* using *both the Internet and disks shipped in packages* [7]. To make up for a lack of sufficient Internet bandwidth, collaborators have the option of moving their data into disks, and sending them as packages through a shipment service provider (e.g. UPS or FedEx).

We are developing a system called Pandora (People and Networks Moving Data Around) which automates the creation of cooperative transfer plans: information is gathered about available Internet and shipment links, and this data is used as input to algorithms that solve for optimal data transfer plans. A preliminary version of the Pandora service is at `http://hillary.cs.uiuc.edu`. This is the most critical step towards the vision of an end-to-end autonomic Pandora service that not only plans, but executes, e.g. initiates data transfers across sites, and manages, e.g. tracks shipments through web services, the entire data transfer process. The planning component collects Internet and shipment link information through bandwidth measurements [25] and by querying shipment service provider Web Services [3, 5]. Cloud collaborators supply parameters (e.g. data size and location) and constraints for a transfer, and Pandora processes this information to produce an optimal transfer plan using novel algorithms. Previously in [7] we developed algorithms for finding a transfer plan that obeys a latency deadline constraint, while minimizing the dollar cost of transferring the data.

In this paper, we solve the new problem of finding the fastest transfer plan given a strict constraint on the dollar cost budget. This *budget-constrained* problem requires new solution techniques that are explored in-depth in this paper.

For instance, finding a fast transfer time within a given budget is important for research collaborations. Consider
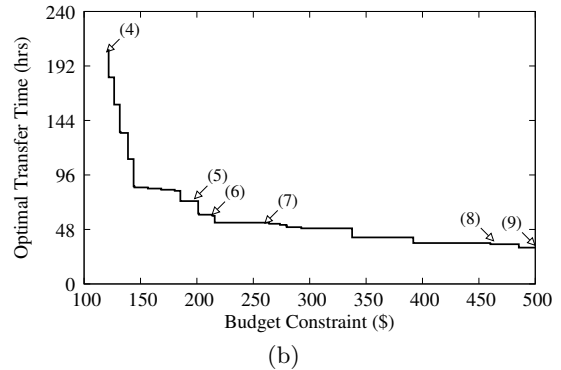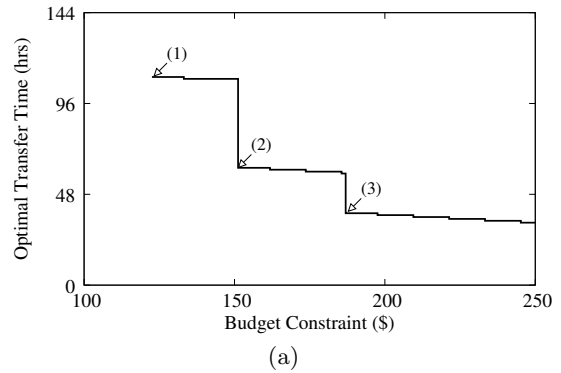
---

a researcher who would like to use sets of geological data collected at distributed sites. She would like to get the data transferred to a cloud service provider (data center) to run computations as quickly as possible. On the other hand, she has a limited amount of money to spend on the data transfer, e.g. she may be working under a grant with limited funds. By solving the budget-constrained transfer problem, Pandora can help the researcher get timely results with a limited budget. Similar examples apply to other cloud users such as consultants, freelancers, newsroom staff, etc. In our algorithms, we solve for the *optimal* transfer plan that meets the constraints while minimizing total transfer latency, so that cloud users can make the most of their funds.

Our algorithms for finding an optimal solution must contend with several major challenges. First, there are many different transfer strategies, and therefore the solution space is massive. Second, each transfer option has different characteristics: an Internet link between a pair of sites has a unique *bandwidth*; further there are several shipment options between a pair of sites (e.g. Ground, Two-Day, Overnight), each with its own *transit time* and *cost* depending on the geographic location of the sites. Third, each site also has characteristics: the time it takes to unpack, plug in, and transfer data from a disk varies by location. If the site is a service provider, charges may apply to incoming Internet bandwidth and disk handling. Fourth, the *scheduled time* when a transfer should begin is important. Finally, we observed that the optimal solution strategy is highly sensitive to the budget constraint. The best transfer plan may look very different for a transfer budget that is slightly smaller or slightly larger.

Figure 1 shows an example that highlights these challenges. We plot the optimal transfer time as a function of the specified budget constraint, for two different transfer scenarios. The sites for these scenarios are shown in Table 1 (with more details in Section 4). In Figure 1a we transfer 2 TB of data from a single source at unm.edu, to a single sink at uiuc.edu, while in Figure 1b we transfer 2 TB of data from eleven different sources (sites 1 through 11 in Table 1), each with 0.18 TB. The two plots show that the optimal transfer time differs widely between different budget constraints. We highlight some of the points in these plots and summarize their use of disk shipment in Figure 1c.

By focusing on a single source and sink, Figure 1a shows the general tradeoffs between shipment options. Faster shipment options can result in a faster transfer time, but require a larger cost budget. This tradeoff affects the shape of the optimal budget-constrained solution curve. The solutions can be divided into three sections by the points (1), (2), and (3). As the budget constraint is increased, these solution points show, respectively, the cut-off when the budget is sufficient to complete the transfer via Ground shipment (from (1) to (2)), a faster Two-Day shipment (from (2) to (3)), and the fastest Overnight shipment ((3) and beyond). Within each section, the transfer time continues to fall as the budget is increased, and more and more money is spent sending data on Internet links, in parallel to the disk shipments.

Figure 1b shows optimal transfer solutions for a network with many sites. The transfer options used in Figure 1c shows a sliver of the massive solution space. Deciding which of the diverse transfer options to use is sensitive to the budget constraint and has a critical impact on the shipment

(a)

(b)

	$	Hrs	Shipment Links		TB
1	120	109	unm.edu→uiuc.edu	Ground	2.00
2	150	59	unm.edu→uiuc.edu	Two-Day	2.00
3	185	31	unm.edu→uiuc.edu	Overnight	2.00
4	120	206	indiana.edu→uiuc.edu	Ground	2.00
5	200	73	unc.edu→uiuc.edu	Overnight	1.88
6	215	60	wustl.edu→indiana.edu	Two-Day	0.09
			unc.edu→uiuc.edu	Overnight	1.73
7	260	54	wustl.edu→uiuc.edu	Two-Day	0.13
8	460	36	wustl.edu→uiuc.edu	Overnight	0.19
			unc.edu→uiuc.edu	Overnight	0.50
9	500	32	indiana.edu→uiuc.edu	Overnight	0.20
			wustl.edu→uiuc.edu	Overnight	0.19
			unc.edu→uiuc.edu	Overnight	0.19

(c)

Figure 1: Optimal transfer times given budget constraints. Figure (a) shows the transfer of 2 TB from site 1 in Table 1, while (b) shows the same data sent from sites 1-11. Table (c) summarizes the disk shipments involved for points labeled in (a) and (b). Internet transfers are not shown.

time. For example, point (5) and (7) only have a difference of $60 in transfer budget, yet the shipment time is reduced by 19 hours. The strategies differ in many ways – not only are the shipment options used different, Overnight in (5) and Two-Day in (7), but also the size of data in the disks, 1.88 GB and 0.13 GB respectively, and even the where the shipment originates, unc.edu and wustl.edu.

Before we discuss our solutions, we would like to clarify a couple of points. First, our back-of-the-envelope calculations for the CCT testbed [4] show that only one to two disk sizes are preferable for ease of management (e.g. to enable hot-swapping disks into CCT). Since the focus of this paper is on solving the planning problem, a further discussion of the disk population is beyond our scope. Secondly, we aim to

derive optimal solutions no matter what the setting. Due to the heterogeneity of dataset distribution and bandwidth across sites (as seen in Table 1), degenerate solutions (e.g. ship all or transfer all over Internet) would be sub-optimal.

2. PROBLEM FORMULATION

Pandora's algorithms for creating data transfer plans require a clear representation of the various inputs. In this section, we describe how the inputs are modeled into a concise graph representation (more detail can be found in our previous work [7]). We model the sites and data transfer networks as the nodes and edges of a directed graph. We then incorporate the graph along with data sizes and locations into a formal dynamic flow network [10]. This formulation is useful because the network defines the solution space. A feasible solution is one that obeys network constraints *across time*.

2.1 Graph model

2.1.1 Internet and Shipping links

The edges on our graph represent the transportation of data through the network of Internet and disk shipment links. Each link has a *capacity*, a *cost*, and a *transit time*. An Internet link has a constant capacity equal to the average available bandwidth, a transit time set to zero (since millisecond latencies are negligible), and a cost of zero (except when terminating in the sink).

Shipping links that transfer disks in packages have entirely different properties. First, there are many levels of service, e.g. Overnight, Two-day, Ground, etc. We treat each level of service as a distinct link. For each shipping link, the cost grows with the number of disks, but this growth is not linear to the amount of data inside each disk. For example, sending either 0.2 TB or 1.8 TB has the same cost (using 2 TB disks), but sending 2.2 TB has a higher cost because an additional disk is needed. Thus, a shipment link has a cost that follows a step function of the amount of data transferred, capacity that is infinite, and a transit time on the order of hours.

Our model captures the salient features for planning a transfer in a compact form. These features will remain relevant during the execution of the transfer plan. The cost and transit time of links will rarely change during the execution of the transfer because these are values that are queried from the providers themselves. Likewise, capacity should not change significantly – [11] has shown that the throughput for large transfers across the Internet is very stable. Only small transfers suffer from variable bandwidth, however Pandora deals with large bulk transfers.

2.1.2 Site bottlenecks

Merely knowing the capacity, cost, and transit time of each link is insufficient to model the transfer networks. There are also end-site characteristics. For Internet links, an end-site is restricted by incoming and outgoing bandwidth capacities. For instance, an end-site can only receive at the data rate of the incoming bandwidth capacity, even if individual links have unused bandwidth. Our model imposes these end-site Internet capacities. In the case of disks, the data sent does not enter a site instantaneously when a package arrives. Rather, an individual at the receiving end must remove the disk from its packaging, then plug in the disk and transfer the actual bytes. Our model accounts for this.
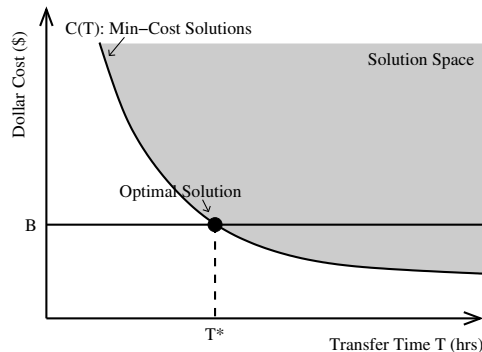

Figure 2: A sketch of the solution space according to transfer time and dollar cost. The optimal solution satisfies the budget constraint B in the shortest possible transfer time T^*.

2.2 Data Transfer Over Time

Our graph model is formalized as a dynamic flow network $\mathcal{N}$ consisting of the set of directed edges A and vertices V discussed above, along with the following attributes on these elements. Each edge in the graph $e \in A$ has a capacity u_e, transit time function τ_e, and a cost function c_e, which is *a linear function for Internet links and a step function for shipment links*.

The budget-constrained transfer problem is then modeled on this network. The budget constraint is given as a scalar value B. Data is represented by assigning a demand attribute D_v to all vertices $v \in S^+$ that are data sources. The destination vertex is assigned a negative $D_v = -\sum_{u \in S+} D_u$.

In a solution to the above problem, flow is assigned to each edge at each time unit θ, denoted as $f_e(\theta)$. These units of flow must meet capacity and conservation constraints. These constraints are similar to those for standard network flow, except they are defined across all $\theta \in [0, T)$, where T is the time when all demands are met, i.e., the net flow that has left each vertex is equal to its demand D_v.

We are looking for flows that meet the above constraints, while reducing the transfer time. That is, our objective is: *Minimize T*. Since our problem is budget-constrained, we must also meet the constraint that the sum of all costs across all times $\sum_{e \in A, \theta \in [0,T)} c_e(f_e(\theta))$ is less than the budget B.

3. SOLUTION

From the solution space of data transfer plans, we are looking for a solution that minimizes transfer latency subject to a budget constraint. In Figure 2 we sketch the solution space according to the transfer time and dollar cost of the solutions. The entire solution space is colored in gray. The horizontal line shows the budget constraint B of the problem. The solutions we are interested in are in the area on or below the line. The optimal solution is precisely the solution with the smallest transfer time in this area. We denote the time of the optimal solution as T^*.

Our problem is to find the optimal solution among the solution space. The strategy we adopt is to make use of the line in Figure 2 that separates the solution space from the non-solutions space. We denote as a function, $C(T)$, the minimum cost among all solutions with transfer time T. This allows us to leverage our previous algorithms in [7] to

Algorithm 1 Binary Search Functions

// type = {$orig$, lb, ub}
function binary_findinterval(init_head, budget, type):
head = tail = init_head
while cost > budget **do**
  cost, endtime = solve_mincost(deadline, type)
  head = tail
  tail = deadline
  deadline = deadline * 2
end while
return head, tail

---

function binary_search(head, tail, budget, type):
while head < tail **do**
  midpoint = $\lfloor$(head+tail)/2$\rfloor$
  cost, endtime = solve_mincost(midpoint, type)
  if cost > budget **then**
    head = midpoint+1
  else
    tail = endtime
  end if
end while
return head, tail

Algorithm 2 Two-Step Min-Cost Binary Search

1: **function twostep_mincost_binary_search(budget):**
2:   head, tail = binary_getinterval(1, budget, $orig$)
3:   head, tail = binary_search(head, tail, budget, $orig$)
4:   **return** tail

---

find the value $C(T)$ for any given T. Using these algorithms, one naive approach may be to compute $C(T)$ for each value of $T = 1, 2, ...$ until we find the smallest value T such that $C(T) \leq B$. In fact, this strategy produces the optimal solution at T^*. However, this approach can be prohibitively expensive. This motivates our development of faster search strategies, which we do next.

3.1 Two-Step Binary Search using Deadline-Constrained Minimum Cost Solutions

The function $C(T)$ is monotonically decreasing. This can be seen by considering that the solution $C(T)$ can be replicated at time $T + 1$, so the value of $C(T + 1)$ by definition will be at most $C(T)$. We make use of this property to create an efficient *binary search algorithm* on $C(T)$ to find the optimal solution.

This binary search is illustrated in Algorithm 1 and 2. It runs in two main steps defined in line 2 and 3 in Algorithm 2. The first part of our search finds both an upper bound and lower bound on the optimal transfer time T^*. It does so by computing values of $C(T)$ for *exponentially increasing values* of T until a solution that meets the budget constraint is found. The first value of T that meets the budget constraint is used as the upper bound, while the immediately previous tried value of T is the lower bound. Notice that this requires computing exactly $\lceil logT^* \rceil + 1$ deadline-constrained minimum cost solutions.

Thereafter, a binary search is performed in the interval between the lower bound and upper bound. After each iteration of the search, the search interval for the next iteration
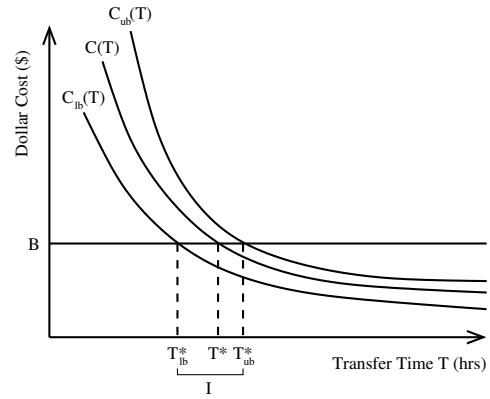

Figure 3: $C_{ub}(T)$, $C_{lb}(T)$, and $C(T)$ minimum cost solution curves.

Algorithm 3 Bounded Min-Cost Binary Search

1: **function bounded_mincost_binary_search(budget):**
2:   head_{ub}, tail_{ub} = binary_getinterval(1, budget, ub)
3:   head_{ub}, tail_{ub} = binary_search(head_{ub}, tail_{ub}, budget, lb)
4:   head_{lb}, tail_{lb} = binary_search(1, tail_{ub}, budget, lb)
5:   head, tail = binary_search(tail_{lb}, tail_{ub}, budget, $orig$)
6:   **return** tail

---

is selected depending on the value of $C(T)$: the upper half of the current interval is selected when $C(T)$ is greater than B, and then lower half is selected otherwise.

The binary search step can add, at most, $\lceil logT^* \rceil$ deadline-constrained minimum cost computations. Thus, the total number of minimum cost computations required is only $2\lceil logT^* \rceil + 1$ in the worst case.

However, while the *number* of computations grows only logarithmically with the optimal transfer time T^*, the actual *time* of computation grows at a much higher rate with T^*. This is because the time required to compute each constrained minimum cost solution grows with the size of the problem which is determined by the deadline time T [7]. Thus performing a search by solving many of these minimum cost computations can become expensive.

3.2 Bounded Binary Search using Strong Lower and Upper Bounds

We can reduce the computation time of binary search if we reduce the number of computations of the function $C(T)$, especially when T becomes large. In this section, we show how to reduce the binary search interval around T^*, which is when T becomes the largest. We accomplish this by introducing two forms of *bounding functions*, $C_{ub}(T)$ and $C_{lb}(T)$. These bound the original function from above and below, respectively. More concretely, these bounding functions obey the relationship $C_{lb}(T) \leq C(t) \leq C_{ub}(T)$. Like $C(T)$, they are monotonically decreasing functions of T.

Figure 3 sketches these bounding functions. We denote the time T where each bounding function intersects with the budget constraint B as T_{lb}^*, and T_{ub}^*. From the definition of the bounding functions, we have $T_{lb}^* \leq T^* \leq T_{ub}^*$. Thus, if we know the values of T_{ub}^* and T_{lb}^*, they define an interval $I = [T_{lb}^*, T_{ub}^*]$ where T^* must reside in.

We can build an efficient binary search framework around

these bounding functions, as shown in Algorithm 3. This algorithm replaces the exponentially increasing search done directly on $C(T)$ in Algorithm 2 with a pair of searches for T_{ub}^* and T_{lb}^*. First, the value of T_{ub}^* is found in a similar way to T^* in Algorithm 2. The exponentially increasing search done to find initial bounds for $C_{ub}(T)$ involves exactly $\lceil log T_{ub}^* \rceil + 1$ computations of C_{ub}. The binary search phase given the initial bounds takes at most $\lceil log T_{ub}^* \rceil$ computations of C_{ub}. Next, after we have found T_{ub}^*, we search for T_{lb}^*. We can skip the exponentially increasing search by using T_{ub}^* itself as the upper bound. The binary search phase involves at most $\lceil log T_{ub}^* \rceil + 1$ computations of C_{lb}. Finally, we perform a binary search for T^* on the interval $[T_{lb}^*, T_{ub}^*]$. This involves at most $\lceil log(T_{ub}^* - T_{lb}^*) \rceil + 1$ computations of C.

Using this strategy, the worst case number of deadline-constrained minimum cost computations required for the original function is changed from $2\lceil log T^* \rceil + 1$ to $\lceil log(T_{ub}^* - T_{lb}^*) \rceil + 1$. On the other hand, additional computations to solve bounding functions are required. Thus, for bounding functions to reduce the binary search computation time, they must be cheap to compute, and their values must not be too far from each other.

3.3 Lower-bound and Upper-bound Networks

If the bounding functions are cheap and tight, the bounded binary search solution approach could be very fast. In this section, we construct bounding functions for the deadline-constrained minimum cost problem. We prove that these functions strongly bound the original problem. Later, in Section 4 we show that these constructions are indeed cheap and tight.

Before we present the bounding formulations, we first review relevant parts of the time-expanded networks concept used to solve the minimum cost problem with a deadline T. More details can be found in our previous work [7].

In Figure 4a we show a data transfer network that contains a single Internet transfer link between node A and B that can transfer one unit of flow at each time unit, and a single shipping option between node B and C that can transfer four units of flow with a transit time of four time units. This network is expanded into a time-expanded network in Figure 4b, which explicitly represents the passing of time while transferring data in the network. The time-expanded network consists of *holdover edges* between every time point that represents the holding of data during that time (vertical edges), *Internet transfer edges* at every time point (solid horizontal edges), and *shipping transfer edges* at each relevant pick-up and delivery time (dashed slanted edges). In this network, the capacity constraints allow a total of three units of data to be transferred from node A to node C during the transfer time of $T = 6$.

We produce *lower bound (LB)* and *upper bound (UB)* variants to the original time-expanded network (denoted as *TEN*) by combining Internet transfer edges at neighboring time points. We systematically group k successive Internet transfer edges into a single edge with a capacity k times the original. Combining edges makes the problem smaller, thus meeting the desirable property of cheap computation. We construct each of the variant networks in a way such that the solutions of a deadline T-constrained minimum cost solution on the variants obey the relationship $C_{lb}(T) \le C(t) \le C_{ub}(T)$. This makes them suitable as

(a) Dynamic Flow Network (b) Time-Expanded Network (TEN)

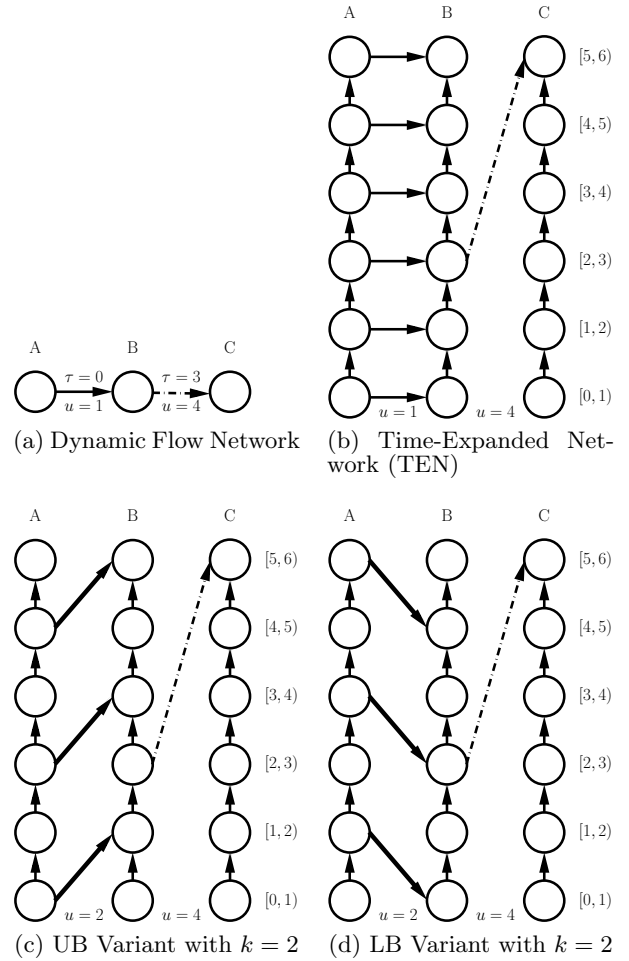(c) UB Variant with $k = 2$ (d) LB Variant with $k = 2$

Figure 4: An example dynamic flow network and the construction of its time-expanded network and variants.

bounding functions. In addition, we also argue that the construction allows variant solutions to be close to the solutions of the original TEN network.

We show the construction of the bounding variants using the example in Figure 4. We first walk through the UB variant. Groups of k Internet edges are combined into a single edge. For example with $k = 2$, the edges $(A_{[0,1)}, B_{[0,1)})$ and $(A_{[1,2)}, B_{[1,2)})$ with capacity 1 are combined into the single edge $(A_{[0,1)}, B_{[1,2)})$ with capacity 2. The combined single edge faithfully represents the total amount of flow that can be sent during the entire k time steps. Intuitively, this is why the UB variant can act as a bound for the TEN solutions. This relationship is given formally in the following Lemma.

LEMMA 3.1. *The relationship* $C(T) \le C_{ub}(T)$ *holds for all T, where $C(T)$ is the T-deadline constrained minimum cost solution to the TEN network, and $C_{ub}(T)$ is the T-deadline constrained minimum cost solution to the UB network.*

PROOF. The Lemma is proved by showing that any solution on the UB network can be transformed into a solution in the TEN network with the same cost. Consider the Internet transfer edges $e(t)$ with capacity u_e and cost per flow

c_e at each time point $t \in [0, T)$ in the TEN network. Each group with k of these edges with $t = [ak, a(k+1))$ make up a UB edge $e_{ub}(a)$ with capacity $u_e k$ and cost per flow c_e. Consider a solution with flow through $e_{ub}(a)$ of size $F \leq u_e k$. We can convert this flow into the TEN network by assigning $f_e(t) = c_e$ to the edges with $t = [ak, ak + \lfloor F/k \rfloor]$, and then assigning any remaining flow to $f_e(ak + \lfloor F/k \rfloor + 1)$. This obeys the capacity constraints of the TEN network edges, while maintaining the same total cost on these edges.

Thus, if we have the T-deadline constrained minimum cost solution to the UB network, $C_{ub}(T)$, we can transform this solution into a solution with the same cost in the TEN network. This solution must have a cost greater than or equal to $C(T)$, by the definition of $C(T)$ as the minimum cost with transfer time T. We have shown $C(T) \leq C_{ub}(T)$. $\square$

On the other hand, the combined edge in the UB network misses opportunities for sending flow between some nodes. This is the tradeoff we pay for constructing a smaller network that is easier to compute. For example, $(A_{[2,3)}, B_{[3,4)})$ does not allow the flow across A and B at time $[2, 3)$ that the TEN network allows. Still, our technique of grouping edges in distinct groups of k mitigates the effect of missing opportunities by keeping them within a fixed time frame k. This is aimed at maintaining the UB solution close to that of the TEN network.

The LB variant is constructed in a similar way to the UB variant. However, in the case of the LB variant, the combined edge is drawn in the opposite direction of time. In the case of the LB variant, any flow in the TEN network can be faithfully represented by the combined edges of the LB network. This gives us the Lemma:

LEMMA 3.2. *The relationship $C_{lb}(T) \leq C(T)$ holds for all T, where $C_{lb}(T)$ is the T-deadline constrained minimum cost solution to the LB network, and $C(T)$ is the T-deadline constrained minimum cost solution to the TEN network.*

PROOF. The Lemma is proved similarly to the previous one. We show that any solution on the TEN network can be transformed into a solution in the LB network with the same cost. Consider the Internet transfer edges $e(t)$ with capacity u_e and cost per flow c_e at each time point $t \in [0, T)$ in the TEN network. Each group with k of these edges with $t = [ak, a(k+1))$ make up a LB edge $e_{lb}(a)$ with capacity $u_e k$ and cost per flow c_e. Consider a solution in the TEN network with flow through the edges in $t = [ak, a(k+1))$. The sum of the flows is less than the combined capacities, i.e., $F = \sum_{t \in [ak, a(k+1))} f_e(t) \leq u_e k$. We can convert this solution into a LB solution by assigning in the LB network $f_e(a) = F$. This obeys the capacity constraints of the LB network edges, while maintaining the same total cost on these edges.

Thus, if we have the T-deadline constrained minimum cost solution to the TEN network, $C(T)$, we can transform this solution into a solution with the same cost in the LB network. This solution must have a cost greater than or equal to $C_{lb}(T)$, by the definition of $C_{lb}(T)$ as the minimum cost with transfer time T. We have shown $C_{lb}(T) \leq C(T)$. $\square$

However, in the case of the LB variant, the combined edge can artificially model flow being sent backwards in time.

Index	Site	BW	Index	Site	BW
Sink	uiuc.edu	-	6	mtu.edu	33.1
1	unm.edu	82.9	7	ufl.edu	7.6
2	unc.edu	78.5	8	rochester.edu	6.9
3	indiana.edu	73.8	9	umn.edu	5.9
4	utexas.edu	70.7	10	ncsu.edu	4.6
5	duke.edu	64.6	11	wustl.edu	2.0

Table 1: Sites used in experiments. BW is the measured available bandwidth (Mbps) to the Sink.

Setting	Site Index	Data Size (per site)
Uniform and *Half Shipment*	1-11	0.18 TB
Skewed	1, 3, 5, 7, 9, 11 2, 4, 6, 8, 10	0.3 TB 0.04 TB

Table 2: Size of source data at each site for the various experimental settings.

In Figure 4d we can send two units of flow on the edge $(A_{[3,4)}, B_{[2,3)})$ while obeying the flow constraints in the LB network. Yet, this is physically impossible. We must be particularly careful that the LB variant solution does not stray far from the TEN solution because of these time-traveling edges. Thus, our construction uses the same global k value across all pairs of sites. This restricts the time that flow can travel back in time to only k units. For example, once a flow has reached the time $[3, 4)$ at any site, it cannot then travel back to a time before $[2, 3)$.

3.4 Using Partial Results

Our solution techniques are developed to produce optimal solutions. When planning a potentially long and expensive transfer, we believe that it is useful to solve for an optimal solution. However, users may still want the option to begin a transfer with sub-optimal results for transfer plans that take a very long time to compute. In this case, we can stop the binary search, and present the user with the solution with the shortest transfer time, among those that meet the budget constraint. In Algorithm 2, at least one constraint satisfying solution will be available when the exponential search stage is complete. In Algorithm 3, a constraint satisfying solution is also available after the first stage. This is due to the result of Lemma 3.1. A minimum cost solution to the UB variant can be converted to a constraint satisfying solution of the TEN network. In our experiments, we observe how fast partial results become available, and how close to optimal their transfer times are.
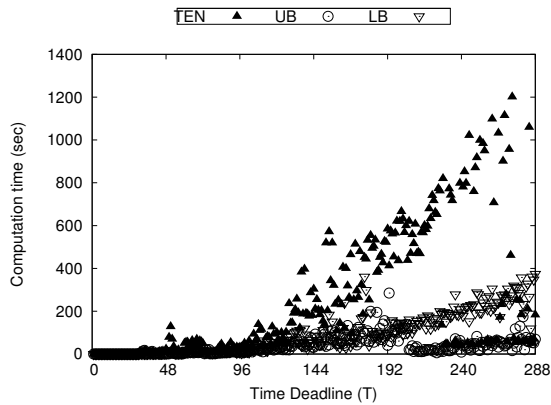
4. EXPERIMENTAL RESULTS

In this section, we use trace-driven experiments to evaluate Pandora's techniques presented in Section 3. We focus on computation time as the main metric. The value of optimal solutions solved by Pandora were presented in Figure 1 – in addition, we evaluate the quality of partial results.
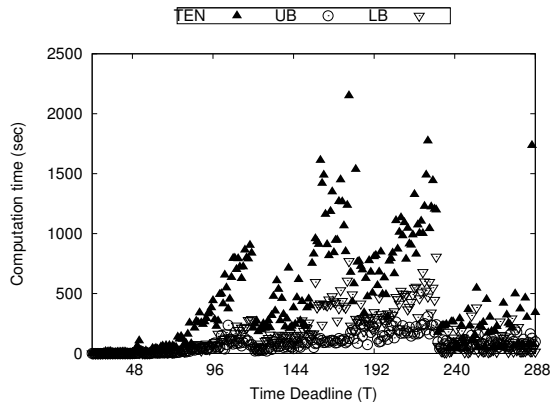
4.1 Experimental Setup

Our experiments were driven by trace data on actual Internet and shipment networks between academic sites. The basic topology we used had a single sink at uiuc.edu and 11 additional sites at .edu domains as listed in Table 1.
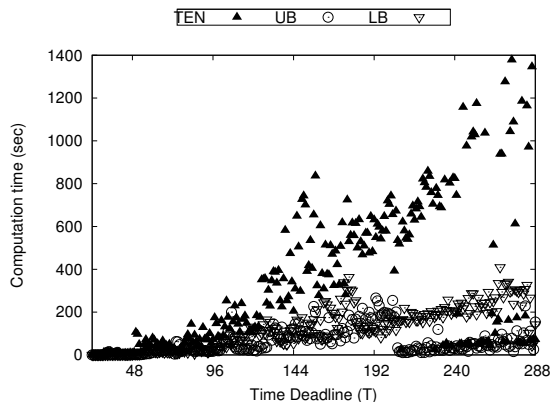
The Internet bandwidth between the sites was derived from PlanetLab available bandwidth traces measured using

(a) Uniform

(b) Skewed

(c) Half Priced Shipment

Figure 5: Computation times for computing the T-deadline constrained minimum cost for TEN, UB, and LB networks with $k = 4$.
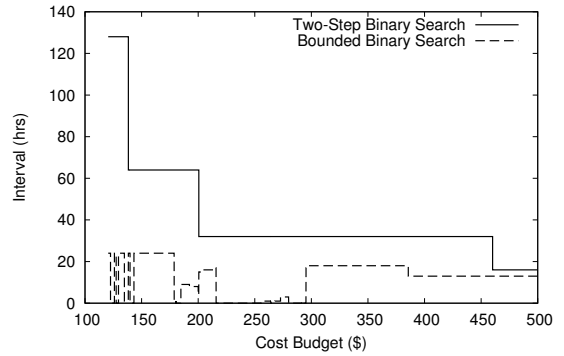

Figure 6: The difference in the binary search interval, using the Two-Step Binary Search in Algorithm 2 and the Bounded Binary Search in Algorithm 3.

from each of 11 quantiles based on the measured bandwidth between .edu domains and the sink. They serve as our data sources. We use three different experimental settings. These are chosen to show how our algorithms cope with diverse, realistic environments. As shown in Table 2, they are: *Uniform*, which places 2 TB of data uniformly at each source (0.18 TB each); *Skewed*, which places 1.8 TB at sources 1, 3, 5, 7, 9, and 11 (0.3 TB each) and 0.2 TB at sources 2, 4, 6, 8, and 10 (0.04 TB each); and *Half Shipment*, which places data in the same way as the Uniform setting, but cuts the shipment link costs to half of their real values. Where not mentioned, we use the Uniform setting.
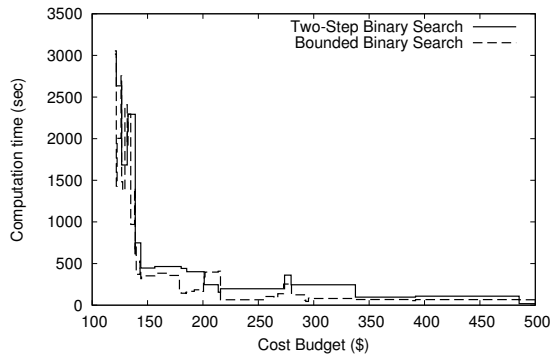
4.2 UB and LB network Microbenchmarks

Our first set of experiments, shown in Figure 5, compares the computation times for solving the T-deadline constrained minimum cost problem on the original time-expanded networks (TEN) against the same computation on UB and LB networks. There is much variation in the measured computation times. Still, generally for all three settings, the computation time forms an upward trend with increasing T. Also, the TEN network computations make up the majority of high computation times. The UB and LB network computations are comparatively cheaper. Thus, the UB and LB networks meet the criteria for a bounding function of being cheaper to compute than the original function.
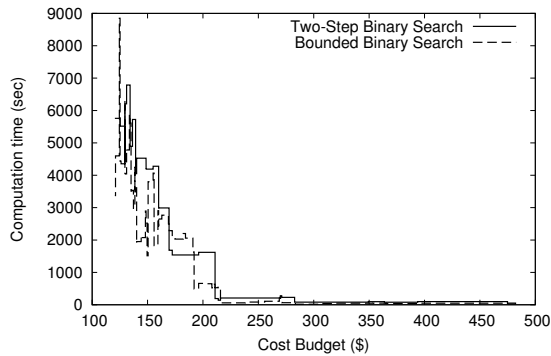
In Figure 6 we investigate whether the bounding functions are sufficiently tight. We show the interval that is searched by the TEN network for both algorithms. For Algorithm 3, this is the distance between T_{lb}^* and T_{ub}^* (used as parameters in line 5), while for Algorithm 2, this is the interval found in the first step (used as parameters in line 3). This interval is large when the budget constraint is strict, because it grows with T^*. In contrast, the interval found with the bounding functions does not grow with T^*. Thus, the bounding functions should be useful, especially when there is a very tight budget constraint.
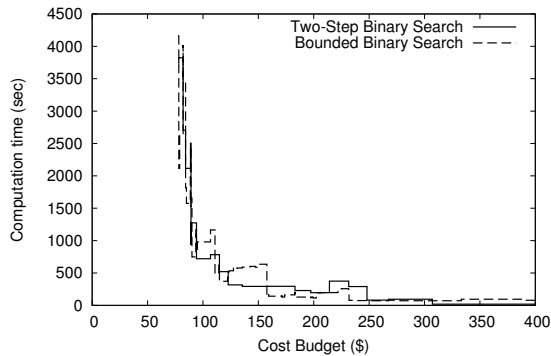
4.3 Binary Search Strategies

In this section, we look at the computation time for finding an optimal solution to the budget-constrained transfer problem using our binary search strategies. We show the computation times of the two-step binary search and bounded binary search in Figure 7. We find that, in general, more stringent cost budgets imply a longer time to compute. This
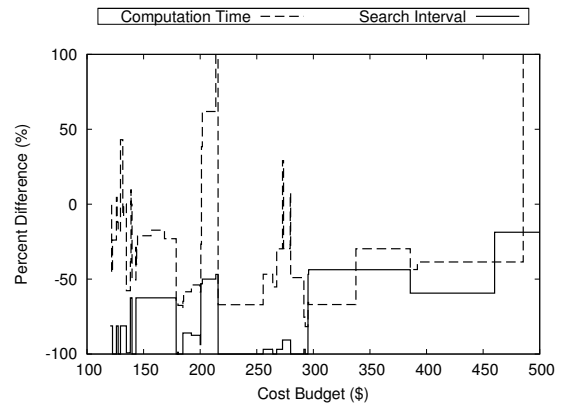
the Spruce measurement tool [21] by the Scalable Sensing Service (S^3) [25] (at 12:32 pm on Nov 15, 2009). We obtained real shipping cost and time data between all sites by using FedEx SOAP (Simple Object Access Protocol) web services [3], with site addresses provided by a whois lookup to the domains. For service charges at the sink, we used Amazon AWS's published costs.

Sites 1 through 11 were chosen evenly by taking one site

(a) Uniform

(b) Skewed

(c) Half Priced Shipment

Figure 7: Computation times using different search strategies.

Budget Range	Pearson Coefficient
[0, 150]	0.30
[0, 175]	0.28
[0, 200]	0.53
[0, 225]	0.50
[0, 250]	0.50
[0, 275]	0.53
[0, 300]	0.60
[0, 325]	0.60
[0, 350]	0.11

Figure 8: The difference between the computation time of bounded and two-step binary search strategies is correlated with the difference in the size of the interval between the strategies. The positive correlation is strong in the first half of the budget range.

is intuitive because a stringent cost budget will have a larger value of T^*, which in turn means that both the number of search iterations increases, and larger time points for the minimum cost problem will have to be solved.

When we compared the effectiveness of both strategies, we found that in the majority of cases the computation time of using the bounded binary search strategy is less than that using the two-step binary search strategy. This pattern is true for all three experimental settings, despite the differences in the computation time for each deadline T in Figure 5. This suggests that the bounded binary search strategy is useful for solving transfer problems on a wide range of networks.
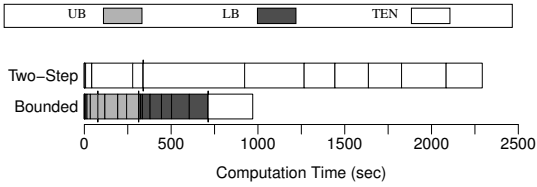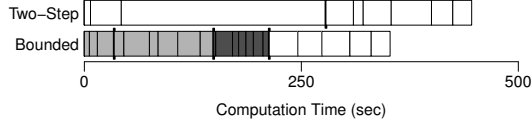
The comparison also shows variance in the relative difference in computation time for the bounded binary search and two-step binary search strategies. Some of this variance can be seen as a natural consequence of the variance we observed in Figure 5 for minimum cost computation running times. Yet, we are able to determine in Figure 8 that the difference in computation is correlated to the difference in interval size for the final minimum cost binary search. The plot shows the relative percentage difference of both the computation time and interval length. The shape of the curves look roughly correlated. We quantify this correlation using the Pearson product-moment correlation coefficient statistic [16]. The Pearson coefficient is a value between -1 and 1, that is used to measure the strength of linear dependence. A strong positive correlation would have a coefficient close 1. We apply the Pearson correlation across various ranges of the budget constraint. We observe from Figure 8 that there is a somewhat strong correlation of 0.5 when looking at the first half of the budget range (up to \$320), for which computation times are relatively high. Since values are significantly positive, we can conclude that the bounded binary search is effective because the bounding functions limit the number of computations required for the final minimum cost binary search.
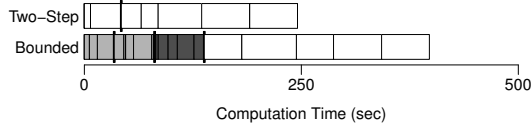
In Figure 9, we break down the computation time of the binary search techniques into their various stages, for a few specific cost budgets. Each slot is color-coded to represent the minimum cost computation of a network variant. The computations are divided into stages of the binary search algorithms by vertical lines. Algorithm 2 has two stages (lines 2 and 3) while Algorithm 3 has four stages (lines 2 through 5). Both Figure 9 (a) and (b) show the common case where the bounded binary search is better. In (a), finding
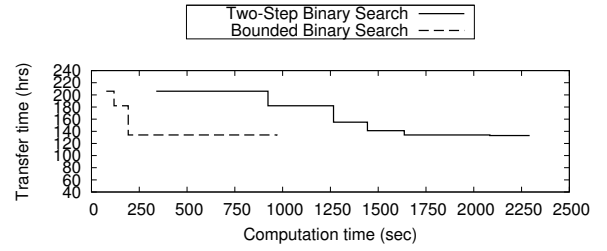
78

(a) Cost Budget = $135

(b) Cost Budget = $150

(c) Cost Budget = $210

Figure 9: Timeline of the minimum cost computations taken in Algorithm 2 and Algorithm 3. Each slot represents a computation. The slots are grouped according to stage, which are presented in lines 2-3 of Algorithm 2 and lines 2-5 of Algorithm 3.
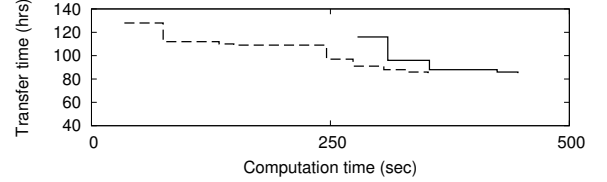
the value of T_{ub}^* for the bounded binary search finishes a little before the upper limit is found in the two-step binary search. Then, given these upper limits, the binary search for T_{lb}^* reduces the binary search interval much faster than the binary search on the original problem. In this case, the interval is small enough that only a single minimum cost computation of the original problem is required in the last stage. In (b), the first three stages of the bounded binary search finish before the first stage of the two-step binary search. In this case, using the bounding functions allowed us to find a tighter interval in a shorter amount of time for the final stage. Finally, (c) shows an example of the less common case where the computation time for the bounded binary search takes longer than the two-step version. In this case, the exponential search stage of the two-step binary search finishes much earlier than the first three stages of the bounded binary search. Then, in the final stage of the bounded binary search, the computation time of each minimum cost took longer than the unbounded search. This can happen because of the variations in computation times shown in Figure 5. However, despite these variances, we have observed (previously in Figure 7) that the bounded binary search is a better option in most cases.

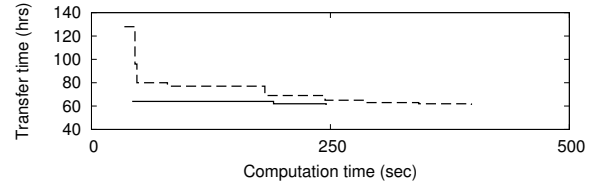## 4.4   Partial Result Solutions

As our final set of experiments, we look at the effectiveness of our binary search algorithms in producing non-optimal partial results. Recall that for both binary search strategies, the first feasible solution is found after their respective first stages. The results in Figure 9 show that at least one sub-optimal solution that meets constraints becomes available much earlier than the final optimal solution. For example,

(a) Cost Budget = $135

(b) Cost Budget = $150

(c) Cost Budget = $210

Figure 10: Evolution of best transfer time during binary search.
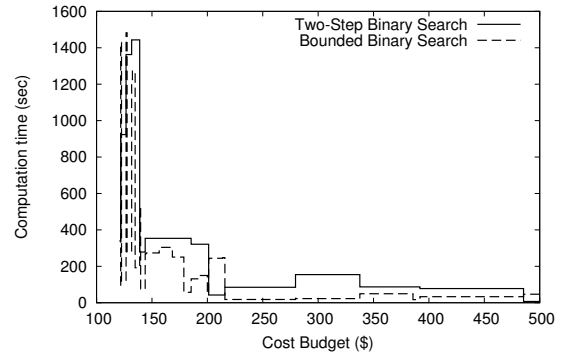

Figure 11: Computation time needed to find a solution within 10% of the optimal.

in Figure 9a, a solution becomes available in less than 350 seconds for the two-stage binary search, and in less than 100 seconds for the bounded binary search.

For partial results, we are interested not only in when a solution becomes available. It is important also to consider the quality of the solution. We show this in Figure 10. We plot the evolution of the best solution's transfer time as the binary search progresses. We observe that the bounded binary search finds solutions with short transfer time much faster than the two-stage strategy. Intuitively, this is because the feasible solutions on the UB network are feasible solutions on the original TEN network.

In Figure 11 we show how quickly the binary search strate-

gies converge to near-optimal solutions across all cost budgets. We plot the computation time needed to get a solution that is within 10% of the optimal transfer time. Compared with Figure 7a, we see a significant decrease in computation time required. For lower cost budgets, the computation time is less than half of finding the optimal time. Thus, using partial results is an attractive technique for users that wish to decrease computation time.

5. RELATED WORK

Previous work has looked at bulk data transfers in the Grid [6, 17], and on PlanetLab [19]. Researchers have found that the use of intermediate nodes can help speed up bulk data transfer. Yet, the scale of TBs in today's infrastructures challenges us to think of new approaches.

We are not the first to propose shipping data using physical media. Jim Gray's SneakerNet [13], the PostManet project [12], and DOT [23] all make use of physical shipment. Also, in industry, services such as Amazon's Import/Export provide interfaces to use physical disk shipment [1]. Our work combines these physical shipping networks with Internet links to reduce cost and latency of cooperative bulk transfers.

There are many success stories running ad-hoc computations in industry clouds [2]. Meanwhile, in the scientific domain one of the main obstacles is that data sets are very large and distributed across many organizations [20, 24]. Our work is aimed at alleviating the large cost and long latency of transferring this data.

The budget-constrained data transfer problem presented is related to the quickest transshipment and evacuation problems posed on dynamic flow networks [10, 15, 9]. Our problem is different, because it contains edges with step function costs. Step function costs are considered in similar problems in operations research [22], such as [14] and [18]. Yet these problems do not take into account transfer latencies. Thus, finding an optimal solution is more challenging than the aforementioned works. Supply chain management considers the efficient transporting of physical goods across many different transportation networks as a key strategic goal [8]. As far as we know, this literature has not considered either bulk data or the Internet as a transportation network.

In our previous work, we looked at the deadline-constrained minimum cost problem [7]. We leverage this previous work as a building block for our algorithms.

6. CONCLUSION

In this paper, we have formulated and solved the problem of finding the fastest bulk data transfer plan given a strict budget constraint. We first characterized the solution space, and observed that the optimal solution can be found by searching through solutions to the deadline-constrained minimum cost problem [7]. Based on these observations, we devised a two-step binary search method that will find an optimal solution. We then developed a bounded binary search method that makes use of bounding functions that provide upper- and lower bounds. We presented two instances of bounding functions, based on variants of our data transfer networks, and proved that they do indeed provide bounds. Finally, we evaluated our algorithms by running them on realistic network inputs. We found that our techniques significantly reduce the time needed to compute solutions.

7. REFERENCES

[1] "Amazon Web Services," Website, http://aws.amazon.com/.
[2] "Amazon Web Services: Case Studies," Website, http://aws.amazon.com/solutions/case-studies/.
[3] "Federal Express Developer Resources Center," Website, http://fedex.com/us/developer/.
[4] "Illinois Cloud Computing Testbed (CCT)," Website, http://cloud.cs.illinois.edu/.
[5] "UPS Online Tools," Website, http://www.ups.com/e_comm_access.
[6] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, and I. Foster, "The globus striped gridftp framework and server," in *Proc. of ACM/IEEE Supercomputing (SC)*, 2005, pp. 54–.
[7] B. Cho and I. Gupta, "New Algorithms for Planning Bulk Transfer via Internet and Shipping Networks," in *Proc. of IEEE ICDCS*, 2010.
[8] S. Chopra and P. Meindl, *Supply Chain Management: Strategy, Planning, and Operation*, 4th ed. Pearson Prentice Hall, 2010, ch. 13.
[9] L. Fleischer and M. Skutella, "Quickest Flows Over Time," *SIAM J. Computing*, vol. 36, no. 6, pp. 1600–1630, 2007.
[10] L. R. Ford and D. R. Fulkerson, "Constructing Maximal Dynamic Flows from Static Flows," *Operations Research*, vol. 6, no. 3, pp. 419–433, 1958.
[11] S. Garfinkel, "An evaluation of Amazon's Grid computing services: EC2, S3 and SQS," Tech. Rep. TR-08-07, Aug 2007.
[12] N. Garg, S. Sobti, J. Lai, F. Zheng, K. Li, R. Y. Wang, and A. Krishnamurthy, "Bridging the digital divide: storage media + postal network = generic high-bandwidth communication," *Trans. Storage*, vol. 1, no. 2, pp. 246–275, 2005.
[13] J. Gray and D. Patterson, "A conversation with Jim Gray," *ACM Queue*, vol. 1, no. 4, pp. 8–17, 2003.
[14] K. S. Hindi, A. Brameller, and K. M. Hamam, "Solution of fixed cost trans-shipment problems by a branch and bound method," *International Journal for Numerical Methods in Engineering*, vol. 12, no. 5, pp. 837–851, 1978.
[15] B. Hoppe and E. Tardos, "The Quickest Transshipment Problem," *Math. Oper. Res.*, vol. 25, no. 1, pp. 36–62, 2000.
[16] M. G. Kendall and A. Stuart, *The advanced theory of statistics*. Hafner Publishing Co., 1961, vol. II.
[17] G. Khanna, U. Catalyurek, T. Kurc, R. Kettimuthu, P. Sadayappan, I. Foster, and J. Saltz, "Using overlays for efficient data transfer over shared wide-area networks," in *Proc. of ACM/IEEE SC*, 2008.
[18] H.-J. Kim and J. N. Hooker, "Solving fixed-charge network flow problems with a hybrid optimization and constraint programming approach," *Annals of Operations Research*, vol. 115, pp. 95–124, 2002.
[19] K. Park and V. S. Pai, "Scale and performance in the CoBlitz large-file distribution service," in *Proc. of USENIX NSDI*, 2006, pp. 29–44.
[20] L. Ramakrishnan, K. R. Jackson, S. Canon, S. Cholia, and J. Shalf, "Defining future platform requirements for e-Science clouds," in *Proc. of the ACM Symposium on Cloud Computing (SoCC)*, 2010.
[21] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in *Proc. of ACM SIGCOMM IMC*, 2003, pp. 39–44.
[22] H. A. Taha, *Operations Research: An Introduction*, 8th ed. Pearson Prentice Hall, 2007, ch. 5.
[23] N. Tolia, M. Kaminsky, D. G. Andersen, and S. Patil, "An architecture for internet data transfer," in *Proc. of USENIX NSDI*, 2006, pp. 19–19.
[24] P. Watson, "e-Science in the Cloud with CARMEN," in *Proc. of International Conference on Parallel and Distributed Computing Applications and Technologies*, 2007.
[25] P. Yalagandula, P. Sharma, S. Banerjee, Sung-Ju.Lee, and S. Basu, "S^3: A scalable sensing service for monitoring large networked systems," in *Proc. of ACM SIGCOMM INM (Workshop)*, Sep. 2006.