

4-2004

# A Cost-Effective Critical Path Approach for Service Priority Optimization in the Grid Computing Economy

Mei LIN


Singapore Management University, [mli@smu.edu.sg](mailto:mli@smu.edu.sg)

Zhangxi LIN

Texas Tech University

**DOI:** <https://doi.org/10.1109/ITCC.2004.1286597>

Follow this and additional works at: [https://ink.library.smu.edu.sg/sis\\_research](https://ink.library.smu.edu.sg/sis_research)

 Part of the [Computer Sciences Commons](#), and the [Management Information Systems Commons](#)

---

## Citation

LIN, Mei and LIN, Zhangxi. A Cost-Effective Critical Path Approach for Service Priority Optimization in the Grid Computing Economy. (2004). *ITCC '04: Proceedings of the International Conference on Information Technology: Coding and Computing: April 5-7, 2004, Las Vegas*. 100-104. Research Collection School Of Information Systems.

**Available at:** [https://ink.library.smu.edu.sg/sis\\_research/1726](https://ink.library.smu.edu.sg/sis_research/1726)

This Conference Proceeding Article is brought to you for free and open access by the School of Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email [libIR@smu.edu.sg](mailto:libIR@smu.edu.sg).

# A Cost-Effective Critical Path Approach for Service Priority Optimization in the Grid Computing Economy

Mei Lin

Department of Computer Sciences  
The University of Texas at Austin  
Austin, TX 78712  
meilin@mail.utexas.edu

Zhangxi Lin

The Rawls College of Business Administration  
Texas Tech University  
Lubbock, TX 79409-2101  
zlin@ba.ttu.edu

## Abstract

*The advancement in the utilization and technologies of the Internet has led to the rapid growth of grid computing; and the perpetuating demand for grid computing resources calls for an incentive-compatible solution to the imminent QoS problem. This paper examines the optimal service priority selection problem that a grid computing network user will confront. We model grid services for a multi-subtask request as a prioritized PERT graph and prove that the localized conditional critical path, which is based on the cost-minimizing priority selection for each node, sets the lower bound for the length of cost-effective critical path that commits the optimal solution. We also propose a heuristic algorithm for relaxing the nodes on the non-critical paths with respect to a given critical path.*

**Keywords:** *Grid computing, computing power economy, network resource pricing, quality of service, PERT/CPM, time-cost tradeoff*

## 1. Introduction

The fast adoption of Internet-based grid computing technology indicates a booming of the application that calls for the incentive compatible mechanism to guarantee required quality of service (QoS). It has been well accepted that pricing will be the effective resolution to the grid computing resource allocation problem. A grid computing network with priced services can be considered a digital economy consisting of firms (servers), products (CPU services, application services, software services and data services), and consumers (users) [1][8][18]. So far, two different economic setups for the implementation of network resource pricing have been proposed, aiming at different economic objectives: social welfare maximization and profit maximization. The former can

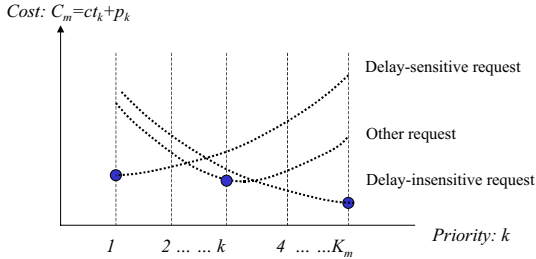
be applied to a network computing economy with centralized management that cares about the overall benefits for both users and service providers. The latter is for profit-making service providers selling their network computing services to the consumers in the network. However, these research efforts are focused on the service provider side problem – how to price their services. The user side problem was not well tackled, in which users will inevitably face a complicated decision problem – how to choose proper priority for each of the services that jointly fulfill their grid computing tasks in order to minimize the total cost with regard to price and throughput delay costs. This paper is intended to explore how to optimize service priority selections for a multi-subtask request in a grid network with prioritized services at different prices. The scope of this paper will be limited to the discussion of a sub-optimum solution for the critical path problem in a prioritized PERT graph because of the NP-hard nature of this problem.

## 2. Prioritized PERT/CPM Model

Consider a grid network with prioritized computing services, including CPU, bandwidth, data access and software sharing. These services are priced in accordance with their usage load. We conceptualize all these kinds of services into logical servers. Users submit *requests* that are divided into multiple independent or dependent subtasks to the grid computing network for services.

Different service requests have different levels of urgency and different values to users. The higher priority service costs more, but saves more time which reduces the delay cost. It is obvious that if a subtask is extremely sensitive to service delay, the user must choose the highest priority for it, or vice versa (Figure 1). In many situations, a user can choose a moderate priority, not too high but good enough for a subtask to

minimize the cost and they can coordinate the selections at different servers in order to maximize their overall utility. We assume that the pricing information and server throughput time regarding the services are available to users so that they can select service quality properly according to the information.



**Figure 1:** The optimum service priority to different types of service request (the greater  $k$ , the lower the priority)

Based on the above description, we use an AON (activity-on-node) PERT graph to represent the order of processing the subtasks [12]. The starting node and the ending node,  $b$  and  $e$  respectively, are symbolic with throughput time 0; the other nodes are activity nodes, denoted by the set  $M$ . Each node represents a subtask to be process by a logical server; and the arcs indicate the sequence in which the subtasks are processed. Thus, this PERT graph is essentially an acyclic direct graph and is denoted as  $G = (V, E)$ , where  $V = \{b, e\} \cup M$ , and  $E$  is the set of ordered pairs representing arcs/edges.

We assume that a node  $m \in M$  provides  $K_m$  classes of priorities, with  $k_m = 1$  being the highest, the total cost of a given subtask at node  $m$  for each priority can be estimated, and that the higher priority requires less time and is more expensive. Thus, for a given subtask at any node  $m$  with priority  $k_m$ , we construct a triplet  $(k_m, t_{k_m}, p_{k_m})$ , where  $t_{k_m}$  and  $p_{k_m}$  are the throughput time and the associated cost at priority  $k_m$  respectively. We also assume that each unit service delay will incur a cost  $c$  to the user. So, the expected cost with a priority service at node  $m$  is  $C_m = ct_{k_m} + p_{k_m}$  if the delay at the node impacts the overall throughput time. We name this graph as a prioritized PERT graph. A question of interest is how to minimize the total expected cost by choosing the optimum  $k_m$  at each node.

**Definition 1:** *Cost-effective critical path* (CECP), denoted as  $(P, E_P) \subset G$ , where  $P \subset V$  and  $E_P \subset E$ , is the critical path of  $G$ , in which the priority of each node is configured to yield the minimum total cost of service.

If a CECP  $(P, E_P)$  can be identified, in which both  $t_{k_m}$  and  $p_{k_m}$  are optimized, the user can apply the time constraints preset by the CECP to solve for the optimum  $k_m$  in the remaining nodes by minimizing

$p_{k_m}$  as long as  $t_{k_m}$  is within the constraint range set by the CECP. Accordingly, we can define a non-critical path as a series of nodes connected one after another, whose start node and end node are the only ones on the CECP. Based on this discussion, the user cost minimization problem can be defined in a general form:

$$\text{Min}_{P \subset V, k_m} C = \sum_{s \in P} \text{Min}_{k_s} (ct_{k_s} + p_{k_s}) + \sum_{m \in P} p_{k_m}$$

s.t.  $\sum_{n \in N} t_{k_n} \leq \sum_{s \in P_N} p_{k_s}$  for every non-critical path  $(N, E_N)$ ,  $N \subset V$  and  $E_N \subset E$ , which starts and ends in the nodes in the CECP.

To solve the problem we can reformulate the above as an integer programming problem [12]:

$$\text{Minimize } C = \sum_{m \in G} \sum_{k_s=1}^{K_s} [x_{k_s} (y_m c * t_{k_s} + p_{k_s})]$$

Subject to the following constraints:

- $\sum_{k_m=1}^{K_m} x_{k_m} = 1$ , s.t.  $m \in G$ ,  $x_{k_m}$  is binary;
- $y_m = 1, \forall m \in P$ , where  $(P, E_P)$  is one of the paths between nodes  $b$  and  $e$ ;  $y_m = 0, \forall m \notin P$ ;
- Given any path  $(Q, E_Q) \subset (P, E_P)$  from node  $a$  to  $b$ , and for any other path  $(N, E_N)$  starting and ending with  $a$  and  $b$ ,  $\sum_{s \in Q} t_{k_s} \geq \sum_{m \in N} t_{k_m}$ .

### 3. Discrete Time-cost Tradeoff Problem

The problem defined in the last section falls into the time-cost tradeoff problem category [5]. However, it differs from the traditional time-cost tradeoff problem in that it takes into account the delay cost without applying the budget or deadline constraints.

Denote  $\Theta$  as the set containing all possible combinations of priority selections for the nodes in the prioritized PERT graph  $G$ . Let  $\sigma \in \Theta$  be a configuration of the network with the selection of a particular priority for each activity node:

$$\sigma = \{(k_m, m), \forall m \in M\}$$

Let  $T(\sigma)$  and  $C(\sigma)$  be the overall time and cost for the configuration  $\sigma$ . The traditional time-cost tradeoff problem with the budget constraint is to minimize the total time with a cost constraint,  $b$ :

$$\text{Find } \sigma^{lc} \text{ such that } T(\sigma^{lc}) = \min_{\sigma \in \Theta} \{T(\sigma) : C(\sigma) \leq b\}.$$

The version with the deadline constraint is to minimize the total time given a due time,  $d$ :

Find  $\sigma^{cl'}$  such that  $C(\sigma^{cl'}) = \min_{\sigma \in \Theta} \{C(\sigma) : T(\sigma) \leq d\}$ .

The above two versions were proved to be NP-hard [5] and have attracted a lot research effort in finding a feasible algorithm. Our basic formulation is identical to that of the traditional time-cost tradeoff problem in that at each node, the higher priority has a shorter throughput time and a higher price. But, in addition to the cost incurred from processing jobs at the nodes, we also introduce a delay cost, which can be considered zero in the traditional problems.

#### 4. Critical Path Optimization Strategy

**Definition 2:** *Conditional critical path (CCP)* is the critical path for a given configuration of priority selections for the prioritized PERT graph.

We can solve a CCP by setting each node at the priority level that minimizes the local cost  $C_m$ . We name this CCP as *Localized CCP* or LCCP. We understand that this local optimization may not necessarily minimize the total service costs for whole network. However, it can provide some information for searching the optimization solution more efficiently.

Let  $\sigma_0$  be a configuration of the network, by which every node has been selected a priority that minimizes the local cost. We have:

$$\sigma_0 = \{(k_m, m) \mid (c \cdot t_{k_m} + p_{k_m}) \leq (c \cdot t_{j_m} + p_{j_m}) \\ 1 \leq j_m \leq K_m, \forall m \in M\}$$

Let  $\sigma$  be the optimal configuration after relaxing the non-critical paths within the time constraints set by the LCCP based on  $\sigma_0$ .

Denote the resulting LCCP as a set of nodes  $P$ , such that  $P \subseteq G$ ; and let  $\sigma_P$  be the configuration of the nodes on the CCP:

$$\sigma_P = \{(k_m, m) \mid (k_m, m) \in \sigma \wedge m \in P\}, \sigma_P \subseteq \sigma.$$

The total throughput time of configuration  $\sigma$  is the sum of the throughput time of the nodes on the critical path. Let us denote the total throughput time as,

$$T(\sigma) = \sum_{m \in P} t_{k_m} = \sum_{(k_m, m) \in \sigma_P} t_{k_m}.$$

And the total cost of the configuration  $\sigma$  is,

$$C(\sigma) = \sum_{(k_m, m) \in \sigma_P} (c \cdot t_{k_m} + p_{k_m}) + \sum_{(k_m, m) \in \sigma \wedge m \notin \sigma_P} p_{k_m}.$$

**Theorem 1** The total throughput time of the optimal configuration is no less than  $T(\sigma)$ .

The proof of this theorem is based on the following two corollaries.

**Corollary 1.1** Given a LCCP  $P$  with configuration  $\sigma_P \subseteq \sigma$ , if  $\exists \sigma_P' \subseteq \sigma'$ , such that  $C(\sigma') \leq C(\sigma)$

and  $T(\sigma') < T(\sigma)$ , then  $\exists \sigma_P'' \subseteq \sigma''$ , such that  $C(\sigma'') \leq C(\sigma')$ ,  $T(\sigma'') > T(\sigma)$ .

**Proof:** The only way to shorten the total throughput time without changing the critical path is to raise priorities on the LCCP. This change alone, however, will also increase the total cost; thus, it is necessary to also lower the priorities for other nodes on the LCCP. If the resultant configuration has a lower total cost and throughput time, by un-raising the priorities raised, the total cost is further reduced, yielding a more optimal configuration with a longer total throughput time.

Assume there is a  $\sigma_P'$  derived from  $\sigma_P$ ,  $\exists (j_m, m) \in \sigma_P'$ , such that  $j_m < k_m$ , for  $(k_m, m) \in \sigma_P$ , and  $\exists (l_m, m) \in \sigma_P'$ , such that  $l_m > k_m$ , for  $(k_m, m) \in \sigma_P$ . Let  $H$  be the set of nodes on LCCP  $P$  that have priorities higher than the original configuration and the effected nodes on the non-critical path, and  $L$  be the set of nodes on critical path  $P$  that have priorities lower than the original configuration and the effected nodes on the non-critical path. Thus,

$$C(\sigma_H') > C(\sigma_H), C(\sigma_L') < C(\sigma_L), T(\sigma_H') < T(\sigma_H), \text{ and } T(\sigma_L') > T(\sigma_L).$$

Since  $C(\sigma') \leq C(\sigma)$  and  $T(\sigma') < T(\sigma)$ , then  $(C(\sigma_H') - C(\sigma_H)) \leq (C(\sigma_L) - C(\sigma_L'))$  and  $(T(\sigma_L') - T(\sigma_L)) < (T(\sigma_H) - T(\sigma_H'))$ .

Let  $\sigma''$  be  $\sigma'$  with the modification that  $\forall m \in H$ , use  $(k_m, m) \in \sigma$  (this is equivalent to letting  $\sigma''$  be  $\sigma$  with the modification that  $\forall m \in L$ , use  $(k_m, m) \in \sigma'$ ). Since  $C(\sigma_H') > C(\sigma_H)$  and  $C(\sigma_L') < C(\sigma_L)$ , so  $C(\sigma'') < C(\sigma') < C(\sigma)$ . And because  $T(\sigma_H') < T(\sigma_H)$  and  $T(\sigma_L') > T(\sigma_L)$ ,  $T(\sigma'') > T(\sigma) > T(\sigma')$ .

**Corollary 1.2** Given a LCCP  $P$ , if  $\exists \sigma_P' \subseteq \sigma'$ , where  $P' \neq P$ , such that  $C(\sigma') \leq C(\sigma)$  and  $T(\sigma') < T(\sigma)$ ,  $\exists \sigma_P'' \subseteq \sigma''$  such that  $C(\sigma'') \leq C(\sigma')$ ,  $T(\sigma') < T(\sigma'')$ , and  $T(\sigma) < T(\sigma'')$ .

This theorem states that for any configuration with a different critical path that has a lower total cost and a shorter total throughput time, there exists another configuration that has a lowest total cost and longest total throughput time among the three configurations. Thus, the total throughput time of the optimal solution is still lower-bounded by  $T(\sigma)$ .

**Proof:** In the new configuration, parts or all of LCCP  $P$  are non-critical paths to the shorter critical path  $P'$ . For any continuous section of  $P$  that is now a non-critical path and contains the nodes of higher or unchanged priorities, the cost of the sub-graph containing the corresponding section of the new critical

path  $P'$  and other effected nodes is higher than the original cost of that sub-graph. In order to have a lower total cost, there must also be parts of the  $P$  that are lengthened for the lower price, whether or not they are parts of the new critical path  $P'$ . And the cost of those sub-graphs is lower than the cost of the same nodes with the original configuration, such that the overall cost is lower than the original total cost. Thus, by undo the changes made in the higher-cost sub-graphs, the resulting configuration has a lower total cost compared to the old and new configurations, and a longer throughput time. The formal proof is similar to that of Corollary 1.1.

## 5. A Heuristic Algorithm for Optimizing Priority Selections on Non-Critical Paths

### 5.1. Node Contractions

After a sub-optimal CCP and its configuration are determined, the next problem is how to relax the time constraints of nodes on the non-critical paths to optimize the total cost. In order to improve the efficiency of problem solving process, we propose three graph contraction schemes in this section for reducing the complexity of the graph. By using contraction rules, we can recursively contract a large graph down to a smaller and simpler one by reducing each unit to one node, thus reduce the size of a large grid.

**Definition 3:** A node with only one immediate precedent node and one immediate subsequent node is called a *single-path node*. A subgraph formed by a series of two or more single-path nodes connected one after another is called a *straight unit*.

**Definition 4:** A node with more than one immediate precedent node or subsequent node is called a *junction node*. A junction node always belongs to more than one path. It cannot contract with any one of the annexed nodes if one of the nodes is also a junction node (Figure 2a), or if the annexed nodes do not have a common immediate node on another side even if all of them are single-path nodes (Figure 2b).

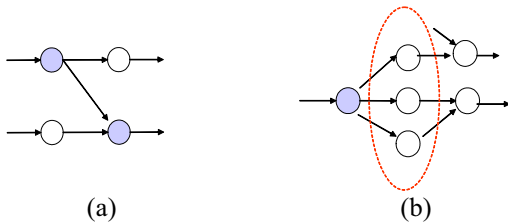


Figure 2: Junction nodes (shaded circle)

**Contraction Rule 1:** If the arc between two connecting nodes is the only arc on that side of each

node, then the two nodes can be merged into a single node (Figure 3). The triplet describing the new node is  $(k, t_k, p_k)$ , where priority  $k$  is one of the priority combinations of the priorities of the nodes in the unit, and  $t_k$  and  $p_k$  are the sums of the  $t$ 's and the  $p$ 's respectively of the priority combination  $k$ .

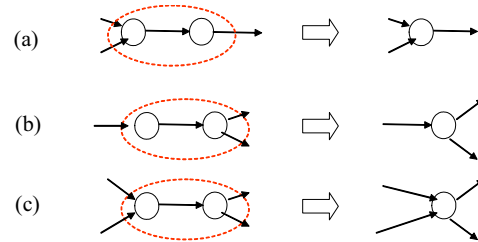


Figure 3: Contractions in three different situations

**Contraction Rule 2:** In a subgraph where there are multiple paths between two nodes and the paths do not have any arc connecting to any node outside of the path, these paths can be merged into a single node (Figure 4). If there are  $J$  paths and each path  $j$  is characterized by priority set  $K_j$ , throughput time set  $T_j$ , and price set  $P_j$ , the triplet  $(k, t_k, p_k)$  for the new node, where  $k \in K_j$ ,  $t_k \in T_j$ , and  $p_k \in P_j$ , can be defined as the following:

- $t_k$  is in the throughput time set  $T^* \subset (\cup_j T_j)$ , ranging  $[t_{min}, t_{max}]$ , where  $t_{min} = \max_j \{ \min_s t_{sj} \mid t_{sj} \in T_j \}$  and  $t_{max} = \max_j \{ \max_s t_{sj} \mid t_{sj} \in T_j \}$ .
- $k$  is the priority that matches the throughput time  $t_k$  originally defined for the associated node.
- $p_k = \sum_{j \in J} \min_{t_j \leq t_k, P_j} P_j$ .

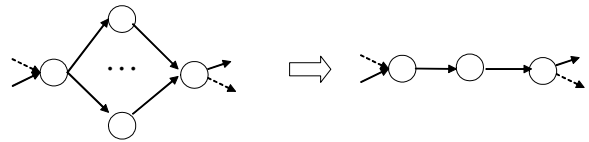


Figure 4: Contraction of parallel paths

A straight unit can be contracted into a single node by repeatedly using Contraction Rule 1, and a multiple path subgraph can also be contracted into a single node using Contraction Rule 2 and then Contraction Rule 1.

### 5.2. Node Priority Optimization

Once the graph is reduced, optimization of the non-critical paths can be accomplished in a more efficient and systematic fashion. Heuristically, the steps are

suggested as, 1) optimizing single-path nodes, 2) optimizing the junction nodes.

The rationale for the above particular order is that a junction node usually affects greater number of paths than a single-path node does; the cost reduction is generally less efficient in optimizing a node that elongates the throughput times of larger number of paths. Thus, even within a group of many junction nodes, it is preferable to optimize the junction node with lowest multiplicity, which is defined as the product of number of incoming arcs and number of outgoing arcs for a junction node. In brief, the larger the multiplicity, the more paths one junction node impacts, thus less desirable an optimization target.

While optimizing the single-path nodes, due to prior graph reduction, some nodes are the contracted single nodes of unit graphs. Before diving into the algorithm of optimizing each unit type, it is necessary to specify the quantity associated with each node.

For any node  $m$ , let  $U$  be the set of non-critical paths that  $m$  belongs to; let  $l_u$  be the throughput time of any  $u \in U$ , and  $t_u$  be the throughput time of the section on the critical path that is between the beginning and ending nodes of  $u$  inclusively, we define *Allowable Relaxation* (AR) with regard to node  $m$  as:

$$AR_m = \inf\{t_u - l_u, \text{ for all } u \in U\}.$$

Additionally, for a node contracted from Contraction Rule 1, its AR equals to the AR of any of the two nodes contracted (note that the two nodes have the same AR); and for a node contracted from Contraction Rule 2, its AR is the largest AR among the paths.

It is also necessary to have a quantity indicating the rate of savings in cost with respect to increase in time from a higher priority to a lower one. In turn, we construct another quantity, a set of pairs, denoted  $B_m$  for any node  $m$ :

$$B_m = \{(r_{ij}, d_{ij}), \text{ where } r_{ij} = \frac{P_i - P_j}{t_j - t_i}, d_{ij} = t_i - t_j \mid k_m \leq i < K_m, k_m < j \leq K_m\}.$$

In addition, for a node that is contracted from Contraction Rule 1, its  $B$  is  $\bigcup B_m$  for the two nodes contracted; for a node that is contracted from Contraction Rule 2, its  $B$  is  $\bigcup B_m$ , for all nodes contracted.

The optimization procedures for different types of nodes are as follows:

#### A. Single-Path Node Optimization Procedures

1. Find the pair  $(r, d)$  in  $B_m$  or  $B$  (for contracted nodes), such that  $r$  is the largest provided that the corresponding  $d \leq AR$ .

2. Change the priority of the node according to the pair chosen.
3. Update set  $B_m$  or  $B$  to exclude the priority option used in step 1.
4.  $AR = AR - d$  for the node and other single-path nodes on the affected paths. If the resulting  $AR$  is less than the  $AR$  of any of the junction nodes on the affected path, change the  $AR$  of that junction node to this  $AR$ .
5. Repeat step 1-5, provided that  $AR \geq 0$  and there exists  $d$  in  $B_m$  or  $B$ , s.t.  $d \leq AR$ . Otherwise, exit.

#### B. Junction Node Optimization Procedures

1. Find the pair in  $\bigcup B_m$  or  $\bigcup B$  (for contracted nodes), for all remaining junction nodes whose  $AR \geq 0$ , such that  $r$  is the largest provided that the corresponding  $d \leq AR$ .
2. Change the priority of the chosen node.
3. Update  $B_m$  or  $B$  of the node to exclude priority option used in step 1.
4.  $AR = AR - d$  for the node and those of the other single-path nodes on the affected paths. If the resulting  $AR$  is less than the  $AR$  of any of the junction nodes on the affected path, change the  $AR$  of that junction node to this  $AR$ .
5. Repeat step 1-5, if there remains junction nodes with  $AR \geq 0$  and there exists  $d$  in  $\bigcup B_m$  or  $\bigcup B$ , s.t.  $d \leq AR$ . Otherwise, exit.

## 6. Summary

We have examined the optimal service priority selection problem that a grid computing user will confront. Grid services to a multi-subtask request are modeled as a prioritized PERT/CPM problem. Differing from the studies of the traditional time-cost tradeoff problem, our formulation added a delay cost in regard to the total throughput time. We defined a cost-effective critical path and proved that it sets a lower bound of optimum throughput time for the prioritized PERT/CPM problem. This effectively narrows the range of the potential optimum critical paths. The algorithm can be generalized to fit similar operations research problems, such as supply chain management.

#### References: (omitted due to limited paper size)

For a complete version as well as the reference list of this paper please see:

<http://geek.ba.ttu.edu/zlin/pdf/ITCC-grid.pdf> or contact the authors.