Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 01-047

Approximation Solutions for the Resource Management Problem Using the General Cover Problem

Mihaela Cardei, David Maccallum, Sai Chen, Ionut Cardei, and Ding-zhu Du

December 13, 2001

Approximation Solutions for the Resource Management Problem Using the General Cover Problem

Mihaela Cardei, David MacCallum, Sai Chen
Ionut Cardei, Ding-Zhu Du

{mihaela, dmac, schen, ionut, dzd}@cs.umn.edu
Computer Science Department, University Of Minnesota, Minneapolis, MN 55455

Abstract

Time-critical applications and multimedia systems require a mechanism to arbitrate access to shared resources. Resource Management systems provide the necessary services to applications for admission control and adaptation. This paper defines a model for the Resource Management Problem and describes two near-optimal approximation methods for criticality-based selection scheduling of competing applications. The resource management allocation scheme is designed to follow a set of goals. In this paper we focus on maximizing the number of higher-criticality sessions admitted, where a session is an instance of an application executing on a system, using a set of resources, such as CPU, memory and disk IO. First we reduce the Resource Management Problem to the General Cover Problem and then present two approximation solutions, one using a greedy algorithm with the performance ratio of $1 + \ln(\max_{1\le j\le n}\sum_{i=1,m} a_{ij})$ and one using a linear programming algorithm with the performance ratio of $\max_{1\le i\le m}\sum_{j=1,n} a_{ij}$.

1. Introduction

The current advent of powerful computers, fast networks and multimedia consumer products have made possible a whole new class of applications. The PC will be the center of a personal network connecting various devices bringing a new user experience. Consider a scenario where the user runs a video monitoring (surveillance) application, consisting of multiple cameras sending high-rate video streams to the computer, a control application for the air conditioner, while listening to web radio stations, and playing Quake Arena with his online friends. All these applications must respond to relevant events in a short time – they are time critical, and are characterized by their different requirements for shared computer resources, such as CPU, disk IO, memory, and network resources. The amount of resources allocated for an application determines the quality of its execution, usually referred as Quality of Service (QoS). In addition, some applications may be more important than others. In our example the surveillance application becomes critical when it detects movement and its quality (frame rate or detection accuracy) must be maximized. Each application is also characterized by a criticality level. A frequent classification divides applications by criticality in three groups: critical, essential and best effort.
In general, the available computer and network resources are not enough to satisfy all applications at their best QoS. Therefore, multimedia applications accept resources to be allocated in a range [min, max] and provide a user-acceptable QoS even with minimal resources. A Resource Management System is responsible with dividing system resources among competing applications and gives precedence to applications with higher criticality. Before an application starts execution, it submits its resource requirements and criticality level to the resource manager. The resource manager (RM) determines availability and assigns resources to the new application that would satisfy at least the minimum QoS. When there are not enough available resources, the RM takes resources away from lower criticality applications and allocates them to the new application. With resources allocated, the new application is ready to be scheduled for execution on the system.

The method to allocate and distribute multiple resources to many applications is not trivial. It can have a big impact on overall system utilization and application QoS.
This paper defines a mathematical model for the problem of resource management for multiple resources (RMP) and describes two solutions based on the greedy method and on linear programming approximation of the General Cover Problem, that are close to the optimal solution.

This paper continues with a description of the application model in section 2. Then, the problem definition and the two solutions are listed in section 3. Section 4 presents some related work and section 5 concludes the paper with some final remarks.

2. Application Model

We define an application session to be an instance of an application executing on a system. Each application session uses a set of resources, such as CPU, memory and disk IO. For instance, a video processing system captures 30 frames/s, filters and compresses the images and then saves them to the disk. Each resource of type i is modeled as a bucket with limit r_i., for $i \in \{1, \ldots, m\}$, where m is the number of resources types in the system. Each session j demands a certain amount a_{ij} of resource type i, for each session $j \in \{1,\ldots, n\}$.
The limited resource constraint for resource i and for all sessions in the system can be stated as:

$$\Sigma_{j=1,n}\, a_{ij}\, s_j \leq r_i, \text{ for each } i \in \{1,\ldots,m\},$$

where $s_j = 1$ if the system schedules session j, and 0 if session j cannot be admitted due to unavailable resources.
The resource management allocation scheme is designed to follow a set of goals. For instance, some feasible goals are maximizing overall system utilization or maximizing QoS for higher critical sessions. In this paper we focus on maximizing the total number of higher-criticality sessions admitted. This can be stated as:

$$\text{Maximize } U = c_1 s_1 + c_2 s_2 + \ldots + c_n s_n,$$

where $c_j > 0$ is a weight factor associated to session j. If c_j corresponds to the session criticality, then maximizing U actually maximizes the overall criticality of the admitted sessions.

The Resource Management Problem (RMP) can be formulated as :

$$\text{Maximize } U = c_1 s_1 + c_2 s_2 + \ldots + c_n s_n,$$

while satisfying the next set of constraints:

$$\Sigma_{j=1,n}\, a_{ij}\, s_j \leq r_i, \text{ for each } i \in \{1,\ldots,m\},$$

This is known to be an NP hard optimization problem[1].

This model can be applied to periodic applications, too. The resource utilization per unit time for resource i by a session j is $a_{i\,j} \cdot rate_j$. The optimization problem for periodic applications reduces to RMP without loss of generality.

3. Reducing the Generalized Resource Management Problem to the General Cover Problem

The generalized Resource Management Problem (RMP) can be written as:

(RMP) maximize $c_1s_1 + c_2s_2 + \ldots + c_ns_n$

subject to:

$$\sum_{j=1,n} a_{ij}s_j \le r_i \quad , i = 1, 2, \ldots m$$

$$s_j \in \{ 0, 1 \} \text{ for } j = 1, 2, \ldots n$$

This problem is characterized by m resources with the maximum available capacity r_i ($i = 1 \ldots m$) and n sessions. Each session i has a weight (or criticality) c_i and has associated a *0-1* variable s_i such that the session is scheduled if and only if $s_i = 1$.

The problem presented in [1] is a particular case for $i = 4$ (4 resources) and $c_j = 1$ for $j = 1..n$ (each session has an equal weight).

RMP is an NP-hard optimization problem, and we want to find an approximation solution that is near optimal.

Next we reduce the RMP to the General Cover Problem (GCP).
Let $x_j = 1 - s_j \qquad j = 1..n$

The RMP becomes:

maximize $n - \sum_{j=1,n} c_j x_j$

subject to:

$$\sum_{j=1,n} a_{ij}x_j \ge \sum_{j=1,n} a_{ij} - r_i \quad , \quad i = 1, 2, \ldots m$$

$$x_j \in \{ 0, 1 \} \text{ for } j = 1, 2, \ldots n$$

If we note $b_i = \sum_{j=1,n} a_{ij} - r_i$ then the RMP reduces to the GCP:

(GCP) *minimize* $\sum_{j=1,n} c_j x_j$

subject to:

$$\sum_{j=1,n} a_{ij}x_j \ge b_i \qquad i = 1, 2, \ldots m$$

$$xj \in \{ 0, 1 \} \text{ for } j = 1, 2, \ldots n$$

The transition between solutions of RMP and GCP is *O(n)*.

Next we present two approximation algorithms for the GCP. Section 3.1 shows a greedy algorithm with performance ratio $1 + ln(max_{1 \le j \le n} \sum_{i=1,m} a_{ij})$ and running time $O(n^2 m)$. Section 3.2 presents a linear programming algorithm with ratio $max_{1 \le i \le m} \sum_{j=1,n} a_{ij}$ and running time $O(n^2 m)$.

3.1 An Approximation Solution of the General Cover Problem (GCP) Using a Greedy Algorithm

In this section we present a greedy algorithm for solving the GCP, then we find its performance ratio and running time. This algorithm is also presented in [2].

Greedy Algorithm GCP_G:

```
I = { 1, 2, … m }
J = { 1, 2, … n }

while I ≠ ∅ do
     find j* ∈ J such that
          ∑i∈I a'ij* / cj* = maxj∈J (∑i∈I a'ij / cj)
     where a'ij = min ( aij, bi - ∑j∉J aij );
     I ← I – { i | a'ij* = bi - ∑j∉J aij };
     J ← J – { j* };
endwhile;
output the feasible solution xG as follow:
  xGj = 0 for j∈ J and xGj = 1 for j∉ J.
```

This algorithm has the running time $O(mn^2)$. The next theorem proofs the performance ratio of this algorithm.

Theorem 1
The greedy algorithm GCP_G produces an approximation solution within a factor of $1 + ln(max_{1 \le j \le n} \sum_{i=1,m} a_{ij})$ from the optimal.

Proof:
We overview here the proof as suggested in [2].
Suppose that the algorithm selects *1, 2, … k* as j^*, in this order, during the computation.
For each $j \le k$, define the *i*-weight of x_j by:

$$w(i, j) = \frac{a''_{ij}\, c_j}{\sum_{i=1}^{m} a''_{ij}}$$

where

$$a''_{ij} = \min(a_{ij}, \max(0, b_i - \sum_{j'=1}^{j} a_{ij'}))$$

For each $j > k$ and *i* with $a_{ij} > 0$, define the *i*-weight of x_j by:

$w(i, j) = \max_{1 \le j' \le k} w(i, j')$

For each *i* and *j* with $a_{ij} = 0$, define the *i*-weight of x_j by:
w(i,j) = 0

The first observation is that $\sum_{j=1}^{n} c_j x_j^G = \sum_{j=1}^{k} c_j = \sum_{j=1}^{k} \sum_{i=1}^{m} w(i, j)$.

Then for each column j, with m' such that $a_{1j}, \ldots a_{m'j} > 0$ and $a_{m'+1j} = \ldots = a_{mj} = 0$,

$$\sum_{i=1}^{m} w(i, j) \le c_j \left(\frac{a_{1j}}{a_{1j}} + \frac{a_{2j}}{a_{1j} + a_{2j}} + \ldots + \frac{a_{m'j}}{a_{1j} + \ldots + a_{m'j}}\right) \le c_j \left(1 + \ln \sum_{i=1}^{m'} a_{ij}\right)$$, see[2]

Suppose that *x** is the optimal solution of the GCP. Then

$$\sum_{j=1}^{k} w(i, j) \le \sum_{x_j^*=1} w(i, j)$$

This follows directly from the definition of *w(i,j)* for *j>k*.
Let's note by APP the result produced by the GCP_G and by OPT the result produced by the optimal solution.

$$\text{APP} = \sum_{j=1}^{k} c_j = \sum_{j=1}^{k} \sum_{i=1}^{m} w(i, j) = \sum_{i=1}^{m} \sum_{j=1}^{k} w(i, j) \le \sum_{i=1}^{m} \sum_{x_j^*=1} w(i, j)$$

$$= \sum_{x_j^*=1} \sum_{i=1}^{m} w(i, j) \le \sum_{x_j^*=1} c_j \left(1 + \ln \sum_{i=1}^{m'} a_{ij}\right) \le \max_{1 \le j \le n} \left(1 + \ln \sum_{i=1}^{m} a_{ij}\right) \sum_{x_j^*=1} c_j$$

$$= \left(1 + \ln \max_{1 \le j \le n} \sum_{i=1}^{m} a_{ij}\right) OPT$$

3.2 An Approximation Solution of the General Cover Problem (GCP) Using a Linear Programming Solution

In this section we present a linear programming approach for solving the GCP, then we find the performance ratio and running time. This method is also presented in [2].

The Integer Programming form of the General Cover Problem (IP_{GCP}) can be stated as following:

min $c_1x_1 + c_2x_2 + \ldots + c_2x_n$
subject to:

$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \ge b_1$
$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \ge b_2$
……
$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \ge b_m$

where $x_1, x_2, \ldots x_n \in \{0,1\}$
and $c_j, a_{ij}, b_i \ge 0$ $(i=1,2,\ldots m,\ j=1,2,\ldots n)$

The IP problem is transformed to a linear programming (LP) problem using the relaxation method, by replacing the constraints $x_j \in \{0,1\}$ to $0 \le x_j \le 1$.

First, we note that b_i and c_j may be assumed to be positive. In fact, if $b_i=0$, them we can remove the i^{th} constraint. If $c_j=0$, then we may set $x_j=1$ and remove corresponding column.

Suppose $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$ is an optimal solution of the LP, then we obtain an approximation solution $x^A = (x_1^A, x_2^A, \ldots, x_n^A)$ for the IP using the following rounding technique:

Let $x_j^A=1$ if $x_j^* \ge a$
Let $x_j^A=0$ if $x_j^* < a$

So the Algorithm for the General Cover Problem can be summarized as:
First find an optimal solution x^* for the LP
Output the following approximation x^A for the IP_{GCP}:
$x_j^A=1$ if $x_j^* \ge a$ and $x_j^A=0$ if $x_j^* < a$.

Next we want to determine the values of a for which
$a_{i1}x_1^A + a_{i2}x_2^A + \ldots + a_{in}x_n^A \ge b_i$, $i=1\ldots m$ hold.

Since $a_{11}x_1^A + a_{12}x_2^A + \ldots + a_{1n}x_n^A$

$$= \sum_{x_j^* \ge a} a_{1j} \ge \sum_{x_j^* \ge a} a_{1j} x_j^*$$

$$\ge \sum_{j=1}^{n} a_{1j} x_j^* - \sum_{x_j^* < a} a_{1j} x_j^* \ge b_1 - \sum_{x_j^* < a} a_{1j} x_j^*$$

Here, we need to know whether $b_1 - \sum_{x_j^* < a} a_{1j} x_j^* \ge b_1$ is true.

Since b_1 is an integer, if $\sum_{x_j^* < a} a_{1j} x_j^* < 1$, then

$b_1 - \sum_{x_j^* < a} a_{1j} x_j^* \ge b_1$ holds, which means

$a_{11}x_1^A + a_{12}x_2^A + \ldots + a_{1n}x_n^A \ge b_1$ holds.

Because

$$\sum_{x_j^* < a} a_{1j} x_j^* < a \sum_{j=1}^{n} a_{1j} < 1$$

if we choose $a < 1/(\sum_{j=1}^{n} a_{1j})$,

then $a_{11}x_1^A + a_{12}x_2^A + \ldots + a_{1n}x_n^A \geq b_1$ holds.

If we choose $a = 1/f$ where $f = max_{1 \leq i \leq m} \sum_{j=1}^{n} a_{ij}$,

then $a_{i1}x_1^A + a_{i2}x_2^A + \ldots + a_{in}x_n^A \geq b_i$ holds for any $i=1..m$.

Next we prove that the general cover problem has a polynomial time approximation with performance ratio $f = max_{1 \leq i \leq m} \sum_{j=1}^{n} a_{ij}$.

Since $\forall$ j, $x_j^A \leq f x_j^*$, we obtain:

$c_1x_1^A + c_2x_2^A + \ldots + c_2x_n^A$

$\leq f(c_1x_1^* + c_2x_2^* + \ldots + c_2x_n^*)$

$\leq f$[optimal value of optimal function IP]

So, x^A is an f-approximation of IP_{GCP}.

The running time of this algorithm includes 2 parts, which are the running time of getting x^* (solving LP problem) and running time of getting x^A from x^*.

Part 1:
Linear programming is in linear time. Different algorithm of LP has different time bound. For example, the randomized algorithm solves LP with m constraints and n variables in approximately $O(n^2m)$.

Part 2:
The running time to find f is $O(mn)$.
The running time to get x^A from x^* is $O(n)$.

So, the total running time is $O(n^2m)$.

4. Related work

There has been extensive research in the area of resource management optimization.
[3] defines a resource model with discrete operation points for multidimensional resources and introduces heuristic approximation algorithms based on convex hulls as well an optimal dynamic programming solution.
Our results on the resource management problem are best understood in the context of the two solutions proposed by Huang, Wan and Du [1]. Each solution uses a greedy algorithm, the first is based on linear programming and the second on the primal-dual theorem for linear programming. We will shortly overview here the basic idea of each algorithm.

The linear programming approach is straightforward. In the Integer Programming problem (IP), described as RMP in the previous section, the numbers s_j were constrained to be *0* or *1*. Linear Programming problem (LP) is obtained using the relaxation technique, allowing the s^*_j to be real numbers, $0 \leq s^*_j \leq 1$. So (IP) becomes a Linear Programming problem (LP), with the advantage that this problem is not NP hard, in fact it is in P (see [2]).

The LP-based approximation algorithm is as follow.
Step 1: Find an optimal solution for LP_k.
Step 2: Order the s^*_j, such that $s^*_1 \geq s^*_2 \geq \ldots \geq s^*_k$.
Step 3: Add sessions according to the order in step 2, if there are sufficient resources.
So, this approach begins with no session, and adds sessions until the available resources are exhausted.

The second method, Primal-Dual Base Removal has an opposite greedy approach:
it begins with all of the sessions, and remove sessions until a feasible solution is found.
The greedy algorithm describes a "greedy" order in which to remove sessions. The idea is to order the critical resources, and then to remove the session that uses the greatest amount of the most critical resource. In this way it arrives to a solution that has no violation of resources consumption.

Huang, Wan and Du used simulation results to argue that the LP-based algorithm does at least as well as the primal-dual based algorithm. Hence, they only analyze the former. Let APPROX be the solution obtained from the LP-based algorithm, and OPT be the optimum solution to the original problem. Then, APPROX ≥ OPT - 4. This means that the approximation solution schedules at most 4 sessions less than the optimal solution.

5. Conclusion

In recent years many system resource management and scheduling techniques have been developed to support criticality based applications.
We presented in this paper a mathematical model and solutions for the resource management problem. The notion of criticality is used to capture the semantics of application importance. We described here a new approach in solving the resource management problem: we first reduce the problem to the general cover problem and then solve it using two approximation methods. The first uses a greedy algorithm and the second uses a linear programming based algorithm. In this paper we also presented the performance ratio and running time for these two solutions. In practice we can employ the solution that provides better result based on the values of the parameters characterizing the problem.

References

[1] Huang, J., Wan P.-J., Du D.-Z. “Criticality and QoS-Based Multiresource Negotiation and Adaptation”, Real-Time Systems, 15, 249-273 1998.
[2] Du D.-Z. and Ko K.-I. “Design and Analysis of Approximation Algorithms”, 1998.
[3] Lee, C, Lehoczky J., Rajkumar R., Siewiorek D. “On Quality of Service Optimization with Discrete QoS Options”, in Proceedings of the IEEE Real-Time Technology and Applications Symposium, June 1999.