Approximation Algorithms for Dynamic Resource Allocation

Vivek F. Farias [*] Benjamin Van Roy [†‡]

March 8, 2005

Abstract

We consider a problem of allocating limited quantities of M types of resources among N independent activities that evolve over T epochs. In each epoch, we assign to each activity a task which consumes resources, generates utility, and determines the activity's subsequent state. We study the complexity of, and approximation algorithms for, maximizing average utility.

1 Introduction

We consider a problem of allocating limited quantities of M types of resources among N independent activities that evolve over T time periods. During each period, a task is assigned to each activity. A task consumes resources, generates utility, and determines the subsequent state of the activity. The goal is to maximize average utility.

As a motivating context, consider dynamic allocation of hundreds of computers and human experts among tens of thousands of information processing activities, each of which involves a sequence of tasks such as document translation, natural language processing, speech recognition, document comparison, and web crawling. Such activities might be managed, say, at a news organization such as the Associated Press. There are potentially many ways to carry out each activity, each of which may require different resources and generate a different level of utility. Further, as major news events occur, the portfolio of activities can change significantly, and the organization may require rapid reallocation of resources.

In this paper, we propose a model that aims to capture salient features of this problem. Our problem bears some similarities with the decision-CPM problem [2, 5], which is known to be intractable [3]. Our problem also encompasses the project selection problem studied in [4].

We prove that the associated optimization problem is NP-hard to approximate within any constant factor. We then propose and study two polynomial-time approximation algorithms which guarantee small errors in different regimes. The first leads to an error of no more than $\overline{U}MT/N$, where $\overline{U}$ is the maximal time-averaged utility that can be generated by an activity. The second algorithm promises $O(\overline{U}\sqrt{N\ln(MT)}/\underline{R})$ error, where $\underline{R}$ is the available quantity of the scarcest resource.

Our bounds are worst-case and additive. Computational experiments on randomly drawn problem instances suggest that the approximate solutions generally result in moderate percentage losses in parameter regimes for which the algorithms are designed.

The remainder of this paper is organized as follows. In Section 2 we present the problem formulation and compare it with models studied in [2, 4]. In Section 3 we show that the corresponding optimization problem is NP-hard to approximate to within any constant factor. We then present two equivalent integer programming formulations in Section 4. Section 5 presents linear programming relaxations to these integer programs. One has a polynomial number and the other an exponential number of decision variables. We show that an optimal solution to the second can be efficiently computed using an optimal solution to the first. Section 6 establishes integrality properties of vertices of the second linear program. and presents an approximation algorithm that provides useful performance guarantees in the $MT \ll N$ regime. Then, in Section 7, we discuss a randomized rounding approach to the problem that uses ideas from [8]. This approach provides

[*]Electrical Engineering, Stanford University

[†]Electrical Engineering and Management Science and Engineering, Stanford University

[‡]Corresponding author: Terman 315, Stanford University, Stanford, CA 94305-4023. email:bvr@stanford.edu

approximation guarantees that can be useful when $MT > N$. In Section 8 we discuss some computational experience with randomly drawn problem instances. That section also introduces a useful integer programming heuristic for problems in the $MT \ll N$ regime. Finally, in Section 9, we present extensions of the model.

2 Problem Formulation

We consider the allocation of M resource types (indexed $m = 1, \ldots, M$) to N activities (indexed $n = 1, \ldots, N$) over T time periods (indexed $t = 1, \ldots, T$). In each time period, a quantity $R_m \in \Re$ of each m^{th} resource is available for allocation. Unused resources do not accumulate; the quantity of resources remains fixed over time. At the beginning of each time period, the state of each activity takes on a value in a finite set $\mathcal{S} = \{1, \ldots, |\mathcal{S}|\}$ and for each activity, a task is selected from a finite set $\mathcal{A} = \{1, \ldots, |\mathcal{A}|\}$.

Let $x_{n,t} \in \mathcal{S}$ and $a_{n,t} \in \mathcal{A}$ denote the state of and task assigned to the nth activity in the tth time period. The state of the activity at the beginning of the $(t+1)$th time period is given by $f_n(x_{n,t}, a_{n,t})$. Further, during the tth time period, the activity generates utility $u_n(x_{n,t}, a_{n,t})$ and consumes resources $r_{n,m}(x_{n,t}, a_{n,t})$. We assume that there is a distinguished task $a' \in \mathcal{A}$ that generates no utility and consumes no resources; i.e., $u_n(x, a') = 0$ and $r_{n,m}(x, a') = 0$ for all n, m, and x. This can be thought of as an option to idle, however, it is not essential to assume $f_n(x, a') = x$.

The objective is to maximize utility, averaged over activities and time. In particular, we have the following optimization problem, which we will refer to as the dynamic resource allocation (DRA) problem:

$$\begin{array}{lll} \max & \frac{1}{NT} \sum_t \sum_n u_n(x_{n,t}, a_{n,t}) & \\ \text{s. t.} & \sum_n r_{n,m}(x_{n,t}, a_{n,t}) \leq R_m & \forall t, m, \\ & f_n(x_{n,t}, a_{n,t}) = x_{n,t+1} & \forall n, t. \end{array}$$

The decision variables are the actions $a_{n,t}$. Note that the problem data consist of functions $u_n : \mathcal{S} \times \mathcal{A} \mapsto \Re$, $r_{n,m} : \mathcal{S} \times \mathcal{A} \mapsto \Re$, $f_n : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$, scalar resource quantities R_m, and initial states $x_{n,1}$. Hence, the problem data is encoded as $\Theta(|\mathcal{S}|^2|\mathcal{A}|N + |\mathcal{S}||\mathcal{A}|NM)$ real numbers. We assume that these numbers are rational so that the input representation is finite.

One way to view the DRA problem involves drawing N directed acyclic graphs, each representing one activity. The nodes of each graph are partitioned into $T + 1$ subsets, each representing a time period. The $(T + 1)$th subset consists of a single node representing termination of the activity. The t^{th} subset (for $t \leq T$) consists of up to $|\mathcal{S}|$ nodes, representing the possible states of the activity at that time. Edges represent potential state transitions, each edge pointing from a node in a tth period to one in a $(t + 1)$th period. Each edge is also associated with a bundle of resources consumed and utility generated during the transition. The DRA problem is to determine trajectories through the activity graphs that maximize aggregate utility subject to resource constraints.

Our problem bears similarities with the decision-CPM problem [2]. Decision-CPM requires finding a directed acyclic subgraph (specified by a suitable, possibly non-linear constraint set) of an activity graph which is itself an arbitrary directed acyclic graph. Edges have associated costs and the goal is to minimize total cost incurred. Our problem may be viewed as a discrete-time decision-CPM problem wherein the activity graph is a forest that decomposes into a set of N separate activity graphs. We are required to find paths in each of these to maximize aggregate utility. Resource constraints preclude a trivial decomposition of the problem. These resource constraints impose an activity selection problem similar to that discussed in [4], which considers the selection of an activity portfolio wherein each activity is associated with a time-dependent resource consumption profile and resource consumption is constrained by a prescribed budget.

3 Computational Complexity

What quality of approximations might we expect? We use a result of Chekuri and Khanna [1] that establishes inapproximability of the packing integer program to show that the DRA problem is hard to approximate to within any constant factor.

Theorem 1. *For any $\alpha < \infty$, the DRA problem is NP-hard to approximate to within a factor of α.*

Proof: We consider a restricted class of DRA problems, with $T = 1$ and $|\mathcal{A}| = 2$. The first of the two tasks consumes a quantity $C_{mn} \in [0, 1]$ (rational) of the resource and generates utility $u_n \in [0, 1]$ (rational), while the second consumes

nothing and earns no utility. There are $R_m \in [1, \infty)$ units of each mth resource. This DRA problem is equivalent to the packing integer program:

$$\begin{array}{ll} \max & \frac{1}{N} u' a \\ \text{s.t.} & Ca \leq R \\ & a \in \{0,1\}^N \end{array}$$

By Theorem 6 of [1], for any $\alpha < \infty$, this problem is NP-hard to approximate to within a factor of α. □

Our proof above reduces a packing integer program to a DRA problem with the same input size. In particular, optimal solutions of a packing integer program with N variables and M constraints correspond to optimal solutions of a DRA problem with M resource types and N activities. Theorem 6 of [1] is proved via a reduction of maximum independent set to the packing integer program problem where $N \ll M$, which illustrates the hardness of the DRA problem in the $N \ll M$ regime. It may still be possible to generate good approximations in other regimes.

4 Integer Programming Formulations

The approximation algorithms we will develop revolve around two integer programs (IP), each of which solves the DRA problem, and their linear programming (LP) relaxations. We present these two IPs in this section.

4.1 The Edge IP

For each n, x, a, and t, let $\phi_{n,x,a,t}$ be an indicator taking value 1 if at time t, activity n is in state x and assigned task a, and 0 otherwise. Further, for each n

$$F_n(\overline{x}, \overline{a}, x) = \begin{cases} 1 & \text{if } f_n(\overline{x}, \overline{a}) = x, \\ 0 & \text{otherwise.} \end{cases}$$

The DRA problem can then be solved via the following 0-1 integer program:

$$\begin{array}{lll} \max & \frac{1}{NT} \sum_{n,x,a,t} u_n(x,a) \phi_{n,x,a,t} & \\ \text{s.t.} & \sum_{n,x,a} r_{n,m}(x,a) \phi_{n,x,a,t} \leq R_m & \forall t, m, \\ & \phi_{n,x,a,1} = 0, & \forall n, a, x \neq x_{n,1}, \\ & \sum_{\overline{x},\overline{a}} F_n(\overline{x}, \overline{a}, x) \phi_{n,\overline{x},\overline{a},t-1} = \sum_a \phi_{n,x,a,t} & \forall n, x, t > 1, \\ & \sum_{x,a} \phi_{n,x,a,t} \leq 1 & \forall n, t, \\ & \phi_{n,x,a,t} \in \{0,1\} & \forall n, x, a, t. \end{array}$$

The variables of this IP represent flows along edges of activity graphs. We will refer to this IP as the *Edge IP*. We digress to note that the above program permits that for some n, $\phi_{n,x,a,t} = 0$ for all x, a in *every* epoch; in such a case setting $\phi_{n,x_{n,1},a',1} = 1$, and selecting the idling action a' in each subsequent epoch $t > 1$ for activity n will also be a feasible solution of equal value to the above integer program and the corresponding DRA problem.

4.2 The Trajectory IP

It is useful to consider another IP that has as its variables flows along possible trajectories in each activity graph. Here, a trajectory for the nth project is a sequence of T state-task pairs: $(x_{n,1}, a_{n,1}), (x_{n,2}, a_{n,2}), \ldots, (x_{n,T}, a_{n,T})$. Note that there are $|\mathcal{A}|^T$ possible trajectories per activity. We assume that these trajectories are indexed by $j \in \{1, \ldots, |\mathcal{A}|^T\}$, denoting trajectory j of activity n by $(x^j_{n,1}, a_{n,1}), \ldots, (x^j_{n,T}, a^j_{n,T})$.

For each n and j, let $\theta_{n,j}$ be an indicator taking value 1 if activity n follows trajectory j and 0 otherwise. The DRA problem can then be solved by the following *trajectory IP*:

$$\begin{array}{lll} \max & \frac{1}{NT} \sum_{n,t,j} u_n(x^j_t, a^j_t) \theta_{n,j} & \\ \text{s.t.} & \sum_{n,j} r_{n,m}(x^j_t, a^j_t) \theta_{n,j} \leq R_m & \forall t, m, \\ & \sum_j \theta_{n,j} \leq 1 & \forall n, \\ & \theta_{n,j} \in \{0,1\} & \forall n, j. \end{array}$$

Analogous to the case of the Edge IP, we note that the above formulation permits that for some n, $\theta_{n,j} = 0$ for all j; in such a case setting $\theta_{n,j} = 1$ for some j such that $a^j_{n,t} = a'$ for all t will also be a feasible solution of equal value to the above integer program and the corresponding DRA problem.

5 LP Relaxations

Our approximation algorithms entail solving a relaxation of the trajectory IP (which we will refer to as the *trajectory LP*) wherein we relax the constraint $\theta_{n,j} \in \{0,1\}$ to $\theta_{n,j} \geq 0$. Since the number of decision variables is exponential in T, straightforward formulation and solution of the trajectory LP is impractical. Instead, we develop an algorithm that efficiently solves the trajectory LP in two steps. The first step is to solve a relaxation of the edge IP (which we will refer to as the *edge LP*), wherein we relax the constraint $\phi_{n,x,a,t} \in \{0,1\}$ to $\phi_{n,x,a,t} \geq 0$. This LP involves $NT|\mathcal{S}||\mathcal{A}|$ decision variables and can therefore be solved in polynomial time.

Each decision variable of the edge LP is associated with an edge in an activity graph. For any optimal solution of the trajectory LP, there is a feasible solution to the edge LP with equal objective value. The latter can be generated by assigning to each edge a value equal to the sum of values associated with trajectories that traverse the edge.

The converse is also true. That is, for any optimal solution of the edge LP, the trajectory LP has a feasible solution with equal objective value. Consequently, letting z^*_{ELP} be the optimal objective value of the edge LP, we have the following result:

Theorem 2.

$$z^*_{\text{ELP}} = z^*_{\text{TLP}}$$

We will now present a polynomial time algorithm that computes an optimal solution to the trajectory LP, given an optimal solution to the edge LP. In addition to completing the proof of the above lemma, this algorithm provides the second step of our algorithm for solving the trajectory LP.

Let ϕ^* be an optimal solution of the edge LP. Fix n. Flow conservation implies that if $\phi^*_{n,x,a,t} > 0$ for some edge (x,a,t), the edge must be part of a trajectory j for which $\phi^*_{n,x^j_t,a^j_t,t} > 0$ for all t.

We consider an iterative algorithm that computes an optimal solution θ^* of the trajectory LP. The algorithm is initialized with $\theta = 0$ and $\phi = \phi^*$ and terminates with $\theta = \theta^*$ and $\phi = 0$. In each iteration, the algorithm identifies the edge $(\overline{x},\overline{a},\overline{t}) = \text{argmin}\{\phi_{n,x,a,t} \geq 0\}$ with minimal positive value and a jth trajectory for which $(x^j_{\overline{t}}, a^j_{\overline{t}}) = (\overline{x},\overline{a})$ and $\phi_{n,x^j_t,a^j_t,t} \geq \phi_{\overline{x},\overline{a},\overline{t}}$ for all t. Then, the algorithm sets $\theta_{n,j} := \phi_{n,\overline{x},\overline{a},\overline{t}}$ and for each t, updates $\phi_{n,x^j_{n,t},a^j_{n,t},t} := \phi_{n,x^j_{n,t},a^j_{n,t},t} - \phi_{n,\overline{x},\overline{a},\overline{t}}$; that is, values along the trajectory are deducted from the edge LP solution and added to the trajectory LP solution. Because the sequence of edge LP solutions is monotonically decreasing and each iteration sets one component to zero, the algorithm terminates in no more than $|\mathcal{S}||\mathcal{A}|T$ iterations. Further, after each iteration, we have

$$\sum_{x,a,t} u_n(x,a)\phi_{n,x,a,t} + \sum_{t,j} u_n(x^j_t, a^j_t)\theta_{n,j} = \sum_{x,a,t} u_n(x,a)\phi^*_{n,x,a,t}.$$

It follows that, upon termination,

$$\sum_{t,j} u_n(x^j_t, a^j_t)\theta_{n,j} = \sum_{x,a,t} u_n(x,a)\phi^*_{n,x,a,t}.$$

If we apply the algorithm we have described to each of the N activities, we arrive at an optimal solution to the trajectory LP with at most $N|\mathcal{S}||\mathcal{A}|T$ nonzero valued variables. Note that θ^* need not be represented exhaustively; only storage of the nonzero variables is required.

6 Using an Optimal Vertex

Our first approximation algorithm converts an optimal solution θ^* of the trajectory LP to an optimal vertex. Task assignment is then carried out based on this optimal vertex.

6.1 Vertices of the Trajectory LP

Each activity is associated with $|\mathcal{A}|^T$ variables, 1 type 2 constraint, and $|\mathcal{A}|^T$ type 3 constraints. It is easy to see that at most $|\mathcal{A}|^T$ among these type 2 and type 3 constraints can be binding. Further, if $|\mathcal{A}|^T$ of these type 2 and type 3 constraints *are* binding, all $|\mathcal{A}|^T$ variables associated with the activity are integer valued.

There are a total of $N|\mathcal{A}|^T$ variables. Hence, at any vertex, at least $N|\mathcal{A}|^T$ constraints are binding. No more than MT of these can be of type 1. Hence, at least $N|\mathcal{A}|^T - MT$ of these must be of types 2 or 3. It follows that no more than MT activities can have non-integer valued variables. We state this fact as a lemma, which is very similar to a well-known result from [7].

Lemma 1. *At any vertex of the trajectory LP, no more than MT activities are associated with non-integer values.*

Given an optimal solution θ^*, an optimal vertex can be computed by solving a linear program involving only the nonzero-valued components of θ^*, which is equivalent to solving the trajectory LP with all other variables constrained to be equal to zero. Since the number of nonzero components is polynomial in the input size, this can be done in polynomial time.

6.2 Task Assignment

To convert an optimal vertex of the trajectory LP to a suboptimal feasible solution of the DRA problem, consider idling during every period in each activity for which (1) there are associated non-integer-valued variables or (2) every variable is equal to zero. The variables associated with each other activity assign value 1 to a selected trajectory and 0 to all others, and we simply assign tasks to follow the selected trajectory. It is easy to see that this results in a feasible solution to the DRA problem. Further, if

$$\overline{U} = \max_{n,j} \frac{1}{T} \sum_t u_n(x_t^j, a_t^j),$$

the utility is at least the objective value of the trajectory LP minus $\overline{U}MT/N$. Since the objective value of the trajectory LP exceeds that of the DRA problem, this also offers a performance loss bound.

We conclude our discussion in terms of a theorem. Let z^*_{DRA} and z^*_{TLP} denote the optimal objective values of the DRA problem and the trajectory LP, respectively. Let $\tilde{z}_{\mathrm{TLP}}$ denote the objective value associated with the suboptimal solution to the DRA that we have described.

Theorem 3. *A feasible solution to the DRA problem with an objective value of*

$$\tilde{z}_{\mathrm{TLP}} \geq z^*_{\mathrm{TLP}} - \frac{\overline{U}MT}{N} \geq z^*_{\mathrm{DRA}} - \frac{\overline{U}MT}{N},$$

can be computed in polynomial time.

7 Randomized Rounding

In Section 5, we presented a means of producing a sparse optimal solution to the trajectory LP. This sparsity enables construction of feasible solutions to the trajectory IP via the randomized rounding techniques of [8]. Such solutions are near optimal when $\underline{R} \gg \sqrt{N}$, where $\underline{R} = \min_m R_m$. That is, the availability of the scarcest resource is large relative to the square root of the number of projects.

Let $\theta^*_{n,j}, j \in \{1, \ldots, |\mathcal{A}|^T\}, n \in \{1, \ldots, N\}$, be a basic optimal solution to the trajectory LP. If one were to randomly assign trajectory j to activity n with a probability slightly less than $\theta^*_{n,j}$, and assign the idling trajectory with the remaining probability, then one might hope to hope to generate a feasible solution to the DRA with value close to z^*_{TLP}. This is the spirit of the randomized rounding approach we pursue in this section.

7.1 An Existence Result

We will first show that there exists a feasible solution to IP that is within an additive error of $O(\overline{U}\sqrt{N \ln MT}/\underline{R})$ of the optimal solution to the trajectory-LP. Towards this end we introduce some notation and preliminary results. For some fixed scalar ν, $\nu \in (0, 1]$, we define the following random variables:

- For each m, n, t, let $R^\nu_{m,n,t}$ represent the random quantity of resource m consumed by activity n in time t if activity n were assigned trajectory j with probability $\theta^*_{n,j}\nu$ and the idling trajectory with probability $1 - \nu \sum_j \theta^*_{n,j}$. Note that $R^\nu_{m,n,t}$ has support $\{(r_{n,m}(x_t^j, a_t^j); j = 1, \ldots, |\mathcal{A}|^T\} \cup \{0\}$ and takes value $r_{n.m}(x_t^j, a_t^j)$ with probability $\nu\theta^*_{n,j}$ and 0 with the remaining probability.

- For each n, let U^ν_n represent the random utility earned by activity n if it were assigned trajectory j with probability $\nu\theta^*_{n,j}$ and the idle trajectory with probability $1 - \nu \sum_j \theta^*_{n,j}$. Note that U^ν_n has support $\{u_n(x_t^j, a_t^j); j = 1, \ldots, |\mathcal{A}|^T\} \cup \{0\}$ and takes value $u_n(x_t^j, a_t^j)$ with probability $\nu\theta^*_{n,j}$ and 0 with the remaining probability.

Now observe that if one were to construct a random solution to the DRA wherein activity n were assigned trajectory j with probability $\nu\theta^*_{n,j}$ and the idle trajectory with probability $1-\nu\sum_j \theta^*_{n,j}$, then $\sum_{n=0}^{N} U^v_n/NT$ corresponds to the value of such a randomly generated solution while $R_m - \sum_{n=0}^{N} R^v_{m,n,t}$ is the quantity of resource m left unused at time t. The following concentration results will be useful in establishing existence of a feasible solution with objective value close to z^*_{TLP}.

Lemma 2. *For any $\epsilon > 0$ let $\nu = 1 - \frac{1}{\underline{R}}\sqrt{\frac{N\ln(MT+1+\epsilon)}{2}}$. If $\nu \geq 0$, then we have for each m, t:*

$$P\left[\sum_{n=0}^{N} R^v_{m,n,t} \geq R_m\right] \leq \frac{1}{MT+1+\epsilon}$$

Lemma 2 gives us an upper-bound on the probability that the resource constraint for resource m in time t is violated by our randomly generated solution.

Lemma 3. *Given $\epsilon > 0$, let ν be as in the previous Lemma and assume $\nu \geq 0$. Let $\delta = \frac{\overline{U}}{\nu z^*_{TLP}}\sqrt{\frac{\ln(MT+1+\epsilon)}{2N}}$. Then,*

$$P\left[\sum_{n=0}^{N} U^v_n/NT \leq \nu(1-\delta)z^*_{\text{TLP}}\right] \leq \frac{1}{MT+1+\epsilon}$$

Lemma 3 gives us an upper-bound on the probability that the value of our randomly generated solution to the DRA is less than $\nu(1-\delta)z^*_{\text{TLP}} = z^*_{TLP} - \left[\frac{\overline{U}}{\underline{R}} + \frac{\overline{U}}{N}\right]\sqrt{\frac{N\ln(MT+1+\epsilon)}{2}}$.
The proofs of both Lemmas rely essentially on the Hoeffding bound [6], and have been omitted for brevity.
Armed with Lemmas 3 and 2, we are ready to prove our main result on the existence of a feasible solution to the DRA with value close to z^*_{TLP}:

Lemma 4. *Given $\epsilon > 0$ let ν be as in Lemma 2. If $\nu \geq 0$, then there exists a feasible solution to IP with value*

$$\geq z^*_{TLP} - \overline{U}\sqrt{2N\ln(MT+1+\epsilon)}/\underline{R}$$

Proof: From Lemma 2, we have that for each m, t:

$$P\left[\sum_{n=0}^{N} R^v_{m,n,t} \geq R_m\right] \leq \frac{1}{MT+1+\epsilon}$$

That is, the probability that a randomly generated solution violates the resource constraint for resource m in time t is less than $1/(MT+1+\epsilon)$.
Let δ be as in Lemma 3. Then, we also have that:

$$P\left[\sum_{n=0}^{N} U^v_n/NT \leq \nu(1-\delta)z^*_{\text{TLP}}\right] \leq \frac{1}{MT+1+\epsilon}$$

That is, the probability that the value of a randomly generated solution is less than $z^*_{TLP} - \left[\frac{\overline{U}}{\underline{R}} + \frac{\overline{U}}{N}\right]\sqrt{\frac{N\ln(MT+1+\epsilon)}{2}}$ is less than $1/(MT+1+\epsilon)$. Thus the probability that any of these events occur

$$P\left[\cup_{(m,t)}\{\sum_{n=0}^{N} R^v_{m,n,t} \geq R_m\} \cup \{\sum_{n=0}^{N} U^v_n/NT \leq \nu(1-\delta)z^*_{\text{TLP}}\}\right]$$

is, by the union bound, less than

$$P\left[\sum_{n=0}^{N} U^v_n/NT \leq \nu(1-\delta)z^*_{\text{TLP}}\right] + \sum_{m,t} P\left[\sum_{n=0}^{N} R^v_{m,n,t} \geq R_m\right] \leq \frac{MT+1}{MT+1+\epsilon}$$

Consequently we have that the probability that no constraint is violated and that the value of the randomly generated solution is greater than $z^*_{TLP} - \left[\frac{\overline{U}}{\underline{R}} + \frac{\overline{U}}{N}\right]\sqrt{\frac{N\ln(MT+1+\epsilon)}{2}}$ is

$$\begin{aligned} &P\left[\sum_{n=0}^{N} R^v_{m,n,t} \leq R_m \ \forall m,t; \sum_{n=0}^{N} U^v_n/NT \geq \nu(1-\delta)z^*_{\text{TLP}}\right] \\ =&1 - P\left[\cup_{(m,t)}\{\sum_{n=0}^{N} R^v_{m,n,t} \geq R_m\} \cup \{\sum_{n=0}^{N} U^v_n/NT \leq \nu(1-\delta)z^*_{\text{TLP}}\}\right] > \frac{\epsilon}{MT+1+\epsilon} \end{aligned} \tag{1}$$

That is, we have the existence of a feasible solution to IP with value

$$\geq z^*_{TLP} - \left[\frac{\overline{U}}{\underline{R}} + \frac{\overline{U}}{N}\right]\sqrt{\frac{\ln(MT+1+\epsilon)}{2N}}$$

We assume without loss of generality that $\underline{R} < N$ (for otherwise every solution to the DRA is feasible, trivializing IP), so that, we have the existence of a feasible solution to IP with value

$$\geq z^*_{TLP} - \overline{U}\sqrt{2N\ln(MT+1+\epsilon)}/\underline{R}$$

□

7.2 Task Assignment via Randomized Rounding

Our discussion thus far suggests the following natural randomized algorithm which may be used to generate a feasible solution to IP, of the quality indicated by Lemma 4. Given a solution $\theta^*_{n,j}, j \in \{1,\ldots,J\}, n \in \{1,\ldots,N\}$ to the TLP, the algorithm assigns (independently for each n) trajectory j to activity n with probability $\nu\theta^*_{n,j}$ and assigns the idle trajectory with probability $1 - \nu\sum_j \theta^*_{n,j}$. Since solutions to the TLP have $N + MT$ positive components cf. Lemma 1, this procedure takes $\Theta(N + MT)$ time. In the event that the resultant solution is not feasible, or the value of the resultant solution is less than $z^*_{TLP} - \left[\frac{\overline{U}}{\underline{R}} + \frac{\overline{U}}{N}\right]\sqrt{\frac{\ln(MT+1+\epsilon)}{2N}}$, the solution is discarded and the procedure repeated. Now by (1) the probability that the rounded solution does not need to be discarded is greater than $\frac{\epsilon}{MT+1+\epsilon}$ so that the expected number of iterations required is less than $\frac{MT+1+\epsilon}{\epsilon}$. We conclude our discussion in this section with the following theorem. Let the random variable $\tilde{z}_{\text{RAND}}$ denote the objective value associated with the suboptimal solution to the DRA that the procedure we have described terminates on.

Theorem 4. *Given $\epsilon > 0$, if $\frac{1}{\underline{R}}\sqrt{\frac{N\ln(MT+1+\epsilon)}{2}} \leq 1$ then a feasible solution to the DRA problem with an objective value of*

$$\tilde{z}_{\text{RAND}} \geq z^*_{\text{TLP}} - \overline{U}\sqrt{2N\ln(MT+1+\epsilon)}/\underline{R} \geq z^*_{\text{DRA}} - \overline{U}\sqrt{2N\ln(MT+1+\epsilon)}/\underline{R},$$

can be computed in randomized polynomial time.

8 Computational Experience

The bounds of Theorems 3 and 4 are worst-case and additive. In this section, we present computational experiments on randomly drawn problem instances. The results suggest that the approximate solutions generally result in moderate percentage losses in parameter regimes for which they are designed. Alongside performance, we report statistics on compute time to demonstrate that the algorithms scale well to large problems. Finally, we propose an integer programming heuristic that, for problems in the $MT \ll N$ regime, further improves approximations without requiring much additional compute time.

8.1 Generative Model

We consider random problem instances with $|\mathcal{S}| = 10$, $|\mathcal{A}| = 3$, $N = 100$, and various values of M and T. For fixed $N, |\mathcal{S}|, |\mathcal{A}|, M$ and T, we generate the activity graph for a single project as follows. Connect the start state for project n

	M	5		10		20	
T		*Approx.*	*Time*	*Approx.*	*Time*	*Approx.*	*Time*
5	*Opt.Vert.*	0.889 +/- 0.005	8.308 +/- 0.187	0.774 +/- 0.016	15.91 +/- 0.331	0.605 +/- 0.016	31.21 +/-0.731
	Heur.	0.971 +/- 0.009	4.107 +/- 0.077	0.915 +/- 0.019	7.082 +/- 0.379	0.916 +/- 0.030	16.52 +/- 1.490
	Rounding.	0.758 +/- 0.017	8.416 +/- 0.201	0.713 +/- 0.019	16.07 +/- 0.346	0.708 +/- 0.016	31.62 +/- 0.741
10	*Opt.Vert.*	0.771 +/- 0.012	42.01 +/- 0.639	0.620 +/- 0.012	115.3 +/- 2.996	0.380 +/- 0. 016	296.8 +/- 17.16
	Heur	0.986 +/- 0.002	21.76 +/- 2.715	0.975 +/- 0.003	132.2 +/- 28.41	0.968 +/- 0.022	1663 +/- 564.8
	Rounding	0.709 +/- 0.017	42.31 +/- 0.611	0.682 +/- 0.014	116.1 +/- 3.019	0.679 +/- 0.017	298.2 +/- 170.9
20	*Opt.Vert.*	0.614 +/- 0.011	239.6 +/- 5.790	0.365 +/- 0. 015	741.4 +/- 24.90	0.127 +/- 0.012	1196 +/- 22.52
	Heur.	***	***	***	***	***	***
	Rounding	0.701 +/- 0.020	240.7 +/- 5.793	0.686 +/- 0.018	743.5 +/- 24.90	0.668 +/- 0.018	1201 +/- 22.50

Table 1: Performance for the optimal vertex and rounding algorithms, as well as the heuristic for varying values of M and T. Approximation ratios reported are lower bounds computed from the value of the LP-relaxations. *** indicates that the heuristic was unsuccessful in finding a feasible solution after a day of computation. Computation time reported in seconds for SunBlade 2000 machines with 2GB of RAM.

(which is selected uniformly at random from $\mathcal{S}$), $x_{n,1}$, to k randomly selected states from $\mathcal{S}$ where k is selected uniformly at random between 1 and $|\mathcal{A}|$. Each of these k states represent the reachable states at $t = 2$. We repeat the procedure for each of these k states and continue in this fashion, to construct an activity graph spanning T epochs. For each edge in the activity graph, we set resource consumption of each of the M resources to some random value distributed uniformly on $[0, 1]$. We also set the utility of the edge to some random value distributed uniformly on $[0, 1]$. N project-graphs are generated in this manner. We set $R_m = 0.5N$ for each m. Such a model, while fairly general, allows us transparent control on two parameters required to identify the regime of the algorithms' applicability. First, we are assured that $E[z^*_{DRA}]/\overline{U} \geq 0.25$ where the expectation is over problem instances, and second, that $\underline{R}/\sqrt{N} \geq 5$ for $n \geq 100$.

8.2 The Optimal Vertex Algorithm

We will call our two algorithms the *optimal vertex algorithm* and the *randomized rounding algorithm*. We first discuss the performance of the optimal vertex algorithm. We would like to study percentage losses associated with problems for which $MT \ll N$, this being the regime in which Theorem 3 assures small additive loss. Table 1 illustrates that for problems from our generative model, moderate percentage losses result for instances with values of MT/N as high as $1/2$. Further the algorithm requires only a few seconds to solve each instance. The performance of the algorithms deteriorates when MT grows larger than N.

8.3 The Randomized Rounding Algorithm

For the randomized rounding algorithm, we would like to examine percentage losses for problems in which MT is large relative to N. Table 1 illustrates that moderate percentage losses are obtained for instances with values of MT/N as large as 4. Further the algorithm requires only a few minutes to solve each instance. Even for the choices of ϵ as small as 10^{-5}, the algorithm required just a single rounding iteration. This is attributed to the fact that the bounds used to compute the ν and δ parameters are themselves extremely conservative. In contrast, a commercial integer programming solver (CPLEX) employing heuristics to find a first feasible solution quickly, was unable to produce a feasible solution even after a day of computation for most instances in the $MT > N$ regime.

8.4 A Task Assignment Heuristic for the $MT \ll N$ regime

We also believe that the use of a generic integer programming technique such as branch-and-bound with a tree exploration heuristic that emphasizes finding feasible solutions quickly is likely to perform very well on problems in the $MT \ll N$ regime. This is because the root node relaxation for the Edge integer program has very few fractional variables cf. Corollary 1 bellow, and a simple depth first search on the fractional variables is guaranteed to produce a feasible solution no worse than that produced by our algorithm.

Corollary 1. *There exists a basic optimal solution to the Edge LP wherein at least $N - MT$ projects have integral flows and at most $2MT^2$ variables are fractional.*

Proof: Lemma 1 shows the existence of an optimal, possibly over-complete basis for the Edge LP wherein at least $N - MT$ projects have integral flows and at most $2MT^2$ variables are fractional. Since there must be a subset of columns of such an over-complete optimal basis that forms an optimal basis, and since this subset necessarily includes columns corresponding to projects with integer solutions, the claim follows. □

One heuristic we experimented with along these lines, is presented in Table 1. After computing an optimal vertex to the trajectory LP, the heuristic fixes integer valued variables in the edge LP, and in place of the task assignment procedure of Section 6.2, uses the first feasible solution from a generic integer programming solver to resolve fractional edges. As illustrated in Table 1, the heuristic takes very little additional time, and produces significantly better approximations for problems in the $MT < N$ regime. As the value of MT/N increases, the heuristic quickly become computationally infeasible.

9 Resource Roles and other Extensions

We introduce the concept of a *role*. The execution of a task requires certain roles be filled by available resources. As an illustration, returning to the example of the Associated Press, the role of translating Arabic to English might be filled by a translator (the resource) that is bilingual in English and Arabic, or by one that is trilingual is English, Arabic and Hindi. The optimal vertex algorithm can be extended to accommodate this.

More formally, we consider the model of Section 2, where in place of M resource types, we have M roles. We will assume that at each of the T epochs we have L resource units indexed (indexed $l = 1, \ldots, L$), where each resource is associated with a qualification set $q_l \subseteq \{1, \ldots, M\}$ that represents the roles unit l may be assigned to. Here, we let $r_{n,m}(x, a)$ denote the number of *roles* of type m that need to be filled by the nth project if it were in state x and took action a. Matching resources to roles in time t may be viewed as a transportation problem with L supply nodes each with a supply of 1, and M demand nodes each with a demand determined by the choice of actions for each project in time t. The f^t_{lm} variables in the edge LP and trajectory LP below correspond to the flow variables for the transportation problem for time t; an edge from resource l to role m exists iff $m \in q_l$. We have the following analogue of the edge LP (ELP'):

$$
\begin{array}{lll}
\max & \frac{1}{NT} \sum_{n,x,a,t} u_n(x,a)\phi_{n,x,a,t} & \\
\text{s.t.} & \sum_{n,x,a} r_{n,m}(x,a)\phi_{n,x,a,t} \leq \sum_l f^t_{lm} & \forall t, m \\
& \phi_{n,x,a,1} = 0, & \forall n, a, x \neq x_{n,1}, \\
& \sum_{\overline{x},\overline{a}} F_n(\overline{x},\overline{a},x)\phi_{n,\overline{x},\overline{a},t-1} = \sum_a \phi_{n,x,a,t} & \forall n, x, t > 1, \\
& \sum_{x,a} \phi_{n,x,a,t} \leq 1 & \forall n, t, \\
& \sum_m f^t_{lm} \leq 1 & \forall l, t. \\
& f^t_{lm} \geq 0 & \forall l, m, t. \\
& \phi_{n,x,a,t} \geq 0 & \forall n, x, a, t.
\end{array}
$$

We also have the following analogue to the trajectory-LP (TLP'):

$$
\begin{array}{lllll}
\max & \frac{1}{NT} \sum_{n,t,j} u_n(x^j_t, a^j_t)\theta_{n,j} & & & \\
\text{s.t.} & \sum_{n,j} r_{n,m}(x^j_t, a^j_t)\theta_{n,j} \leq \sum_l f^t_{lm} & \forall t, m & \text{(type 1)} \\
& \sum_m f^t_{lm} \leq 1 & \forall l, t. & \\
& f^t_{lm} \geq 0 & \forall l, m, t. & \\
& \sum_j \theta_{n,j} \leq 1 & \forall n, & \text{(type 2)} \\
& \theta_{n,j} \geq 0 & \forall n, j. & \text{(type 3)}
\end{array}
$$

Analogous to Theorem 2, we have the following theorem:

Theorem 5.

$$z^*_{\text{ELP}'} = z^*_{\text{TLP}'}$$

The proof of this result is identical to that presented in Section 5; given an optimal solution to ELP', it is possible using the flow decomposition procedure in that section to construct an optimal solution to TLP' in polynomial time. Furthermore it is not hard to see that the following analogue to Lemma 1 holds (the argument used in Lemma 1 applies identically).

Lemma 5. *At any vertex of TLP', no more than MT activities are associated with non-integer values.*

Finally, given an optimal solution $(\theta^*_{n,j}, f^{t*}_{lm})$ to TLP', we assign trajectories to activities exactly as described in Section 6.2. In doing so, we are assured of integral demands for each role, less than or equal to $\sum_l f^{t*}_{lm}$ for each m, t. Arriving at an allocation of resources to roles then requires the solution of MT transportation problems whose feasibility follows from the feasibility of $(\theta^*_{n,j}, f^{t*}_{lm})$. We conclude our discussion of this extension with the following theorem. Let $\tilde{z}_{\mathrm{TLP}'}$ denote the objective value associated with the suboptimal solution to the DRA problem with roles that we have described.

Theorem 6. *A feasible solution to the DRA problem with roles having an objective value of*

$$\tilde{z}_{\mathrm{TLP}'} \geq z^*_{\mathrm{TLP}'} - \frac{\overline{U}MT}{N} \geq z^*_{\mathrm{DRA}'} - \frac{\overline{U}MT}{N},$$

can be computed in polynomial time.

We close the paper mentioning a few immediate extensions of our model and results. Instead of assuming that the availability of a resource stays fixed over time, we may also model the vector of available resources evolving according to some time varying linear dynamical system. Denoting by R_t, the M-dimensional vector of available resources at time t, and assuming that at time t, R_t evolves according to the linear system $R_{t+1} = A_t R_t + b_t$, where $A_t \in \Re^{M\times M}$ and $b_t \in \Re^M$, and further that R_1 is known in advance, we may consider the following DRA problem with time-varying resource availability:

$$\begin{array}{lll} \max & \frac{1}{NT}\sum_t \sum_n u_n(x_{n,t}, a_{n,t}) & \\ \text{s. t.} & \sum_n r_{n,m}(x_{n,t}, a_{n,t}) \leq (R_t)_m & \forall t, m, \\ & f_n(x_{n,t}, a_{n,t}) = x_{n,t+1} & \forall n, t. \\ & R_{t+1} = A_t R_t + b_t & \forall t < T \end{array}$$

The natural analogues to the Edge and Trajectory LPs for the above problem have properties identical to those of the Edge LP and Trajectory LP explored in Sections 5 and 6; the proofs and algorithms of those sections apply identically. It is therefore possible to efficiently produce, for the DRA problem with time varying resource availability, a solution having value $\geq z^*_{\mathrm{DRA}} - \overline{U}MT/N$.

Finally, we note that there is no reason to assume, for either the optimal vertex algorithm or the rounding algorithm, that the functions $u_n : \mathcal{S}\times\mathcal{A} \mapsto \Re$, $r_{n,m} : \mathcal{S}\times\mathcal{A} \mapsto \Re$, $f_n : \mathcal{S}\times\mathcal{A} \mapsto \mathcal{S}$ are independent of t; such a dependence merely increases problem description length by a factor of T.

Acknowledgments

This research was supported by a supplement to NSF Grant ECS-9985229 provided by the Management of Knowledge Intensive Dynamic Systems Program (MKIDS). We thank Sid Berkowitz for stimulating discussions that inspired the problem formulation and Ashish Goel, Pete Veinott, and Yinyu Ye, for helpful discussions on approximation algorithms.

References

[1] Chandra Chekuri and Sanjeev Khanna. On Multidimensional Packing Problems. *SIAM Journal on Computing* , 33(4):837–851, 2004.

[2] W. Crowston and G. L. Thompson. Decision CPM: A Method for Simultaneous Planning, Scheduling, and Control of Projects. *Operations Research*, 15(3):407–426, 1967.

[3] P. De; E. J. Dunne; J. B. Ghosh and C. E. Wells. Complexity of the Discrete Time-Cost Tradeoff Problem for Project Networks. *Operations Research*, 45(2):302–306, 1997.

[4] G.L.Nemhauser and Z. Ullman. Discrete Dynamic Programing and Capital Allocation. *Management Science*, 15(9):494–505, 1969.

[5] Thomas J. Hindelang and John F. Muth. A Dynamic Programming Algorithm for Decision CPM Networks. *Operations Research*, 27(2):225–241, 1979.

[6] W. Hoeffding. Probability for sums of bounded random variables. *J. Amer. Stat.*, 58:13–30, 1963.

[7] Alan S. Manne. Programming of Economic Lot Sizes. *Management Science*, 4(2):115–135, 1958.

[8] Prabhakar Raghavan. Probabilistic construction of deterministic algorithms approximating packing integer programs. *Journal of Computer and System Sciences*, 37:130–143, 1988.