5 Metrical Task Systems

A *metrical task system* consists of a *metric space* M, whose members are called the *states* of the system, and a set of *tasks*, where each task is a cost function on M.

Given a metrical task system, our problem is to design an online algorithm for fulfilling a sequence of tasks which are given to us online.

If we are in a given state, say x, and a task τ is requested, we must choose a state y (which may be x), move to y, and then service the task. The cost of this move is then $||x, y|| + \tau(y)$, where $||x, y||$ denotes the cost of changing state from x to y, *i.e.,*the distance between x and y in the metric space M.

5.1 The Paging Problem as an MTS

The k-paging problem is an example of a metrical task system. Let $\mathcal{P}$ be the set of all possible pages. Let $\mathcal{M}$ be the set of all subsets of $\mathcal{P}$ of order k, *i.e.,*all possible caches. Then $\mathcal{M}$ is a metric space, where the distance between two possible caches is the number of replacements needed to change one into the other. Formally, if $X, Y \in \mathcal{M}$, then $||X, Y||$ is defined to be $k - |X \cap Y|$. For example, if $k = 5$, $X = \{a, b, c, d, e\}$ and $Y = \{b, c, d, f, g\}$, then $||X, Y|| = 2$.

The tasks of the k-paging problem are requests to individual pages; *i.e.,*each member $p \in \mathcal{P}$ corresponds to the task of requesting p.

The paging problem is an example of a *forcing task system*, defined to be a task system where every cost is either 0 or ∞. If X is the current cache, the cost of servicing the request p is 0 if $p \in X$, and ∞ otherwise. Since paying ∞ is impossible, when p is requested, we must "move" to a cache which contains p.

5.2 The Ice Cream Problem

A vendor sells two kinds of ice cream, vanilla and chocolate. Each customer requests one gallon of one kind of ice cream, and the vendor cannot predict what kind the next customer will request. He makes the ice cream online, *i.e.,*when the customer requests it.

The vendor has an ice cream making machine, which has two states: V and C. It costs one dollar to change the states of the machine. If the machine is state V, he can make a gallon of vanilla for one dollar using the machine, or he can make a gallon of chocolate for four dollars by hand. If the machine is state C, he can make a gallon of chocolate for two dollars using the machine, or he can make a gallon of vanilla for two dollars by hand.

In this case, the metric space M consists of only two states, V and C, and the distance between them is 1. Calling the tasks v and c, the following table gives the tasks as functions on M.

	V	C
v	1	2
c	4	2

Figure 5.1: Tasks for the Ice Cream MTS

We can list several possible strategies for the ice cream problem.

1. Leave the machine in state V at all times.
2. Leave the machine in state C at all times.
3. If the machine is in the "wrong" state at the time of a request, change state, otherwise do not change state.
4. If there are two consecutive requests with the machine in the "wrong" state, change state. In any other situation, do not change state.
5. If the machine is in state V and the request is c, change state. If the machine is in state C and two consecutive requests are both v, change state upon receiving the second request. In any other situation, do not change state.

Exercise 5.1 *Find the competitiveness of each of the above five strategies.*

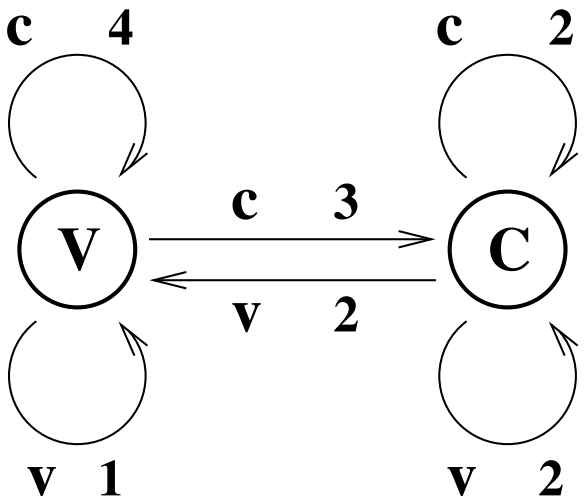

Figure 5.2: Diagram of the Ice Cream MTS

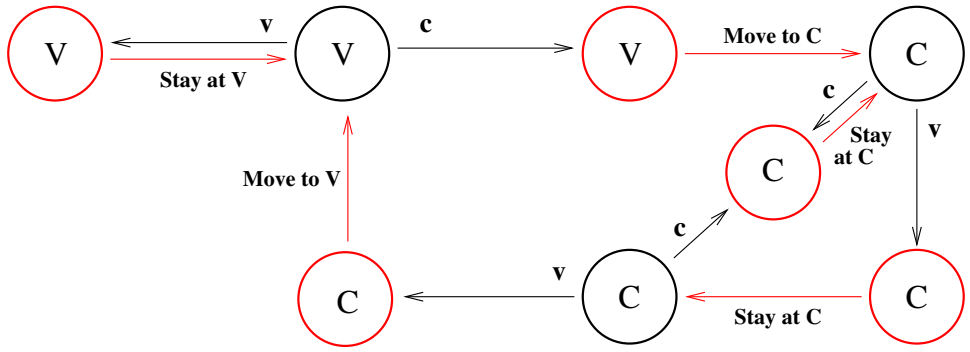

Figure 5.3: Strategy 5 for the Ice Cream MTS

5.3 Work Functions

How can we systematically analyze the ice cream problem, or for that matter, any metrical task system?

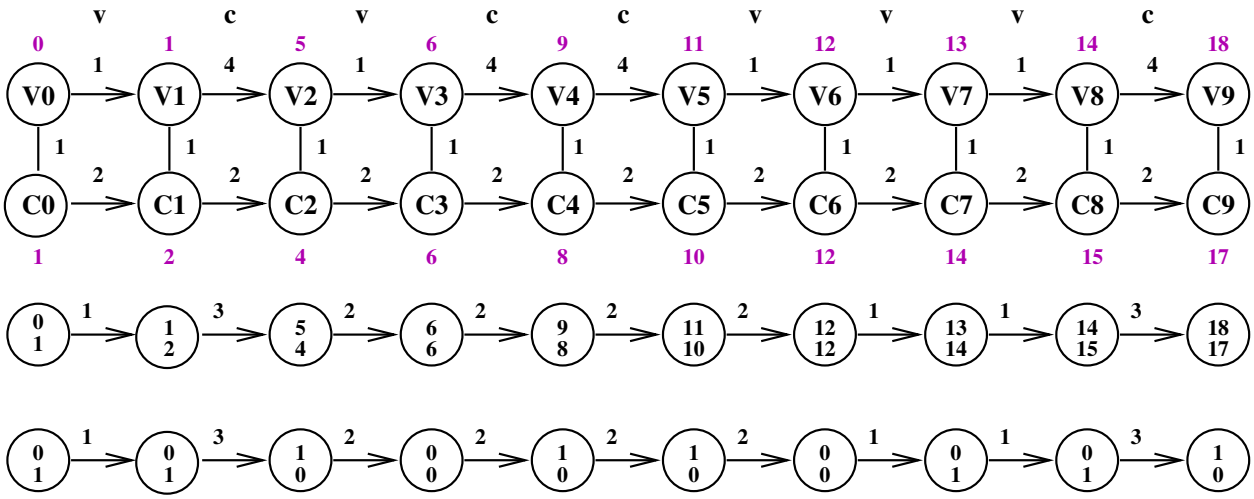

Figure 5.4: Computing optimal service for the Ice Cream Problem using dynamic programming. The third line shows work functions and amortized optimal cost. The fourth line shows offset functions and amortized optimal cost.

The optimal service of a given task sequence $\rho = \tau_1\tau_2\cdots\tau_n$ of an MTS can be computed using dynamic programming, provided the number of states of the MTS that need be considered is finite. For each $0 \leq t \leq n$, and each state x of the MTS, define $\omega^t(x)$ to be the optimal cost of servicing the request $\rho[1:t] = \tau_1\tau_2\cdots\tau_t$ and ending in state x. Thus, $\omega^0(x) = ||s, x||$, where s is the start state of the MTS, and ω^t for $t > 0$ is computed by the recurrence:

$$\omega^t(x) = \min_y \left\{\omega^{t-1}(y) + \tau_t(y) + ||y, x||\right\}$$

The values of $\omega^t(x)$ can be computed by solving the single source shortest path problem in the weighted directed graph, whose nodes consist of all pairs of the form (x, t) such that $0 \leq t \leq n$

and x is a state of the MTS, where the source node is $(s, 0)$, and where there are directed edges of two types:

- An edge from (x, t) to (y, t) of weight $||x, y||$ for all states x, y and all t.
- An edge from (x, t) to $(x, t+1)$ of weight $\tau_t(x)$ for any state x and all $t < n$.

We call ω^t the *work function after t steps*. The work function is a function whose domain is the set of states of the MTS. For example, given the request sequence *vcvccvvvc* for the ice cream problem, $\omega^5(V) = 11$ and $\omega^5(C) = 10$. Our algorithm cannot know the adversary's state at a given time, but after five steps, the algorithm knows that if the adversary is at V then he has paid 11, while if he is at C he has paid 10.

5.4 Amortized Optimal Cost

Let $\rho = \tau_1 \tau_2 \cdots \tau_n$ be a task sequence (also called a request sequence). Let $cost^t_{opt}$ be the optimal cost of servicing the first t tasks. Then the amortized optimal cost at step t is defined to be $\Delta^t cost_{opt} = cost^t_{opt} - cost^{t-1}_{opt}$.

Lemma 5.1 *The sum of the amortized optimal costs is the optimal cost.*

Proof: Note that $cost^0_{opt} = 0$. Then $\sum_{t=1}^{n} \Delta^t cost_{opt} = \sum_{t=1}^{n} \left(cost^t_{opt} - cost^{t-1}_{opt} \right) = cost^n_{opt}$ □

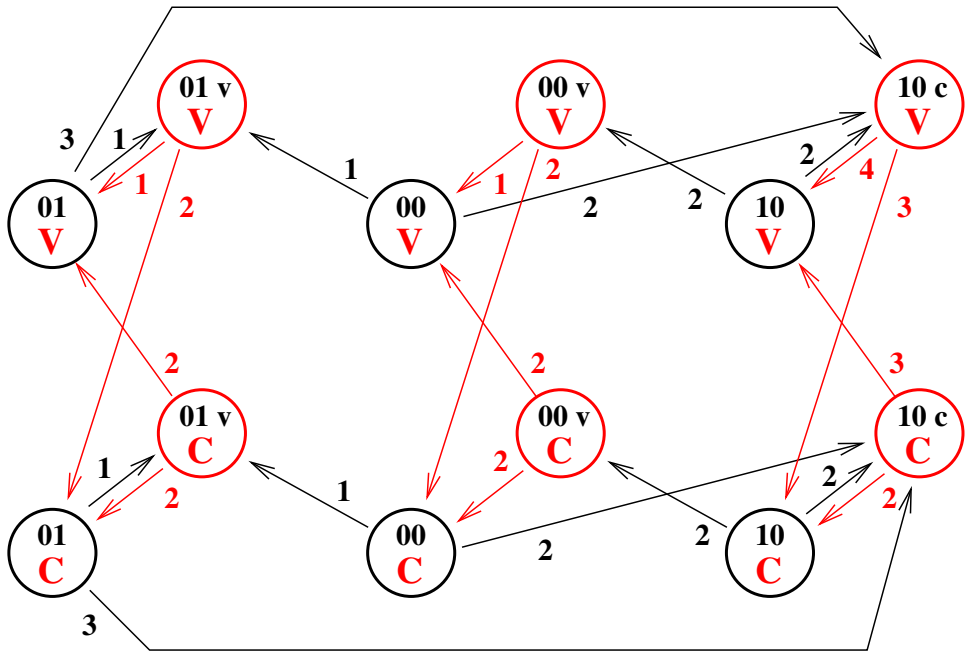

Figure 5.5: The Red-Black Game Equivalent to the Ice Cream Problem

Using the offset functions, we can reduce the problem of finding the competitiveness of finite metrical task system with integral costs to the problem of finding the competitiveness of a finite

red-black game. Figure 5.5 shows the red-black game for the ice cream problem. Each node shows the current offset function and the algorithm state, and in addition, each black node shows the current request. The red costs are the algorithm costs, and the black costs are amortized optimal costs.

Lemma 5.2 *In a finite red-black game, the competitiveness is always the ratio of red costs to black costs for some cycle.*

Exercise 5.2 *Find the competitiveness of the ice cream problem.*

Hint: there are a number of cycles in Figure 5.5. The ratio of the costs for each of these cycles is a simple fraction, ranging from 1 to 2. For example, the ratio of a cycle consisting of the four rightmost nodes in the figure, those with offset function 10, is $\frac{3}{2}$. One suggestion is to pick a cycle, compute the ratio, and then try to fill in the potentials. If that fails, pick a different cycle.

5.5 The Underemployed Watchman

A certain watchman has only one duty, namely to shut down a reactor in case of a nuclear attack against the United States. He has never had to do his duty, and we all hope that he never will. Yet, he must stay alert!

The President has devised a system for keeping the watchman awake. He sits in a room with two buttons, one on the North wall, and one on the South wall. There is a light next to each button. When any light flashes, the watchman must press the button next to that light within 30 seconds.

There are also two chairs in the room, one near the North wall, and the other near the South wall. Neither chair can be moved. Since the room is very small, he can reach either button from either chair, but it is much easier to reach the button nearer a chair than the button further away. The watchman must be sitting in a chair at all times, except for the brief time it takes to change chairs. If a light flashes, the watchman can do one of two things: either press the button from the chair that he's in, or move to the other chair and then press the button from there.

The watchman wants to minimize "effort," defined as follows.

1. It takes no effort to press the button next to the chair where he's sitting.
2. It takes one "unit" of effort to press the button farther from the chair where he's sitting.
3. It takes one "unit" of effort to change chairs.

Exercise 5.3 *Find the offset functions and construct the red-black game for the underemployed watchman problem, and compute the competitiveness.*