Lab10-Turing Machine

Algorithm and Complexity (CS214), Xiaofeng Gao, Spring 2020.

∗ If there is any problem, please contact TA Yiming Liu.
∗ Name:Hongjie Fang Student ID:518030910150 Email: galaxies@sjtu.edu.cn

1. Design a one-tape TM M that computes the function $f(x, y) = x \mod y$, where x and y are positive integers ($x > y$). The alphabet is $\{1, 0, \square, \triangleright, \triangleleft\}$, and the inputs are x 1's, $\square$ and y 1's. Below is the initial configuration for input $x = 7$ and $y = 3$. The result $z = f(x, y)$ should also be represented in the form of z 1's on the tape with the pattern of $\triangleright 111 \cdots 111 \triangleleft$.

Initial Configuration

	$\triangleright$	1	1	1	1	1	1	1	$\square$	1	1	1	$\triangleleft$	
	$\uparrow$													
	q_S													

(a) Please describe your design and then write the specifications of M in the form like $\langle q_S, \triangleright\rangle \to \langle q_1, \triangleright, R\rangle$. Explain the transition functions in detail.

(b) Please draw the state transition diagram.

(c) Show briefly and clearly the whole process from initial to final configurations for input $x = 7$ and $y = 3$. You may start like this:

$$(q_s, \underline{\triangleright}1111111\square111\triangleleft) \vdash (q_1, \triangleright\underline{1}111111\square111\triangleleft) \vdash^* (q_1, \triangleright1111111\underline{\square}111\triangleleft) \vdash (q_2, \triangleright1111111\square\underline{1}11\triangleleft)$$

(Note that for simplicity, we write $(q_1, \triangleright\underline{1}111111\square111\triangleleft) \vdash^* (q_1, \triangleright1111111\underline{\square}111\triangleleft)$ if the corresponding transaction repeats on multiple inputs with the same state.)

Solution. Here are my answers to the sub-questions.

(a) Here are the specific transitions of M:

$$\begin{array}{lll}
\langle q_S, \triangleright\rangle \to \langle q_1, \triangleright, R\rangle & & \\
\langle q_1, 1\rangle \to \langle q_1, 1, R\rangle & \langle q_1, \square\rangle \to \langle q_1, \square, R\rangle & \langle q_1, \triangleleft\rangle \to \langle q_2, \triangleleft, L\rangle \\
\langle q_2, 1\rangle \to \langle q_3, \triangleleft, L\rangle & \langle q_2, \square\rangle \to \langle q_5, 1, L\rangle & \\
\langle q_3, 1\rangle \to \langle q_3, 1, L\rangle & \langle q_3, \square\rangle \to \langle q_4, \square, L\rangle & \\
\langle q_4, 1\rangle \to \langle q_1, \square, R\rangle & \langle q_4, \square\rangle \to \langle q_4, \square, L\rangle & \langle q_4, \triangleright\rangle \to \langle q_7, \triangleright, R\rangle \\
\langle q_5, 1\rangle \to \langle q_6, 1, R\rangle & \langle q_5, \square\rangle \to \langle q_5, 1, L\rangle & \langle q_5, \triangleright\rangle \to \langle q_6, \triangleright, R\rangle \\
\langle q_6, 1\rangle \to \langle q_1, \square, R\rangle & & \\
\langle q_7, 1\rangle \to \langle q_8, 1, L\rangle & \langle q_7, \square\rangle \to \langle q_7, 1, R\rangle & \langle q_7, \triangleleft\rangle \to \langle q_8, \triangleleft, L\rangle \\
\langle q_8, 1\rangle \to \langle q_h, \triangleleft, S\rangle & &
\end{array}$$

(Method Explanation) The main method of my design is that we repeat trying to eliminate y "1"s in x's representation, until there is not enough "1"s in x's representation, and the rest of "1"s in x's representation are the result of $x \mod y$. What's more, we do not use "0" in the design of the Turing Machine; therefore it also works on a simple alphabet $\Gamma' = \{1, \square, \triangleright, \triangleleft\}$.

(Transitions Explanations) Here are some detailed explanations.

Step 1. Find out the last "1" in the representation of y using the following transitions:

$$\langle q_S, \triangleright\rangle \to \langle q_1, \triangleright, R\rangle$$
$$\langle q_1, 1\rangle \to \langle q_1, 1, R\rangle \qquad \langle q_1, \square\rangle \to \langle q_1, \square, R\rangle \qquad \langle q_1, \triangleleft\rangle \to \langle q_2, \triangleleft, L\rangle$$

Step 2a. If found, change this "1" to "$\triangleleft$" using the following transition, and go to Step 3.

$$\langle q_2, 1\rangle \to \langle q_3, \triangleleft, L\rangle$$

Step 2b. If not found, change all the "$\square$" except the first to "1" using the following transitions, and go back to Step 1.

$$\langle q_2, \square\rangle \to \langle q_5, 1, L\rangle$$
$$\langle q_5, 1\rangle \to \langle q_6, 1, R\rangle \qquad \langle q_5, \square\rangle \to \langle q_5, 1, L\rangle \qquad \langle q_5, \triangleright\rangle \to \langle q_6, \triangleright, R\rangle$$
$$\langle q_6, 1\rangle \to \langle q_1, \square, R\rangle$$

Step 3. Find out the last "1" in the representation of x using the following transitions.

$$\langle q_3, 1\rangle \to \langle q_3, 1, L\rangle \qquad \langle q_3, \square\rangle \to \langle q_4, \square, L\rangle$$
$$\langle q_4, \square\rangle \to \langle q_4, \square, L\rangle$$

Step 4a. If found, change this "1" to "$\square$" using the following transition, and go to Step 1.

$$\langle q_4, 1\rangle \to \langle q_1, \square, R\rangle$$

Step 4b. If not found, change all the "$\square$" except the last to "1" and change the last to "$\triangleleft$" using the following transitions, and the process is finished.

$$\langle q_4, \triangleright\rangle \to \langle q_7, \triangleright, R\rangle$$
$$\langle q_7, 1\rangle \to \langle q_8, 1, L\rangle \qquad \langle q_7, \square\rangle \to \langle q_7, 1, R\rangle \qquad \langle q_7, \triangleleft\rangle \to \langle q_8, \triangleleft, L\rangle$$
$$\langle q_8, 1\rangle \to \langle q_h, \triangleleft, S\rangle$$

(b) The state transition diagram of my Turing Machine is as follows (Fig. 1).

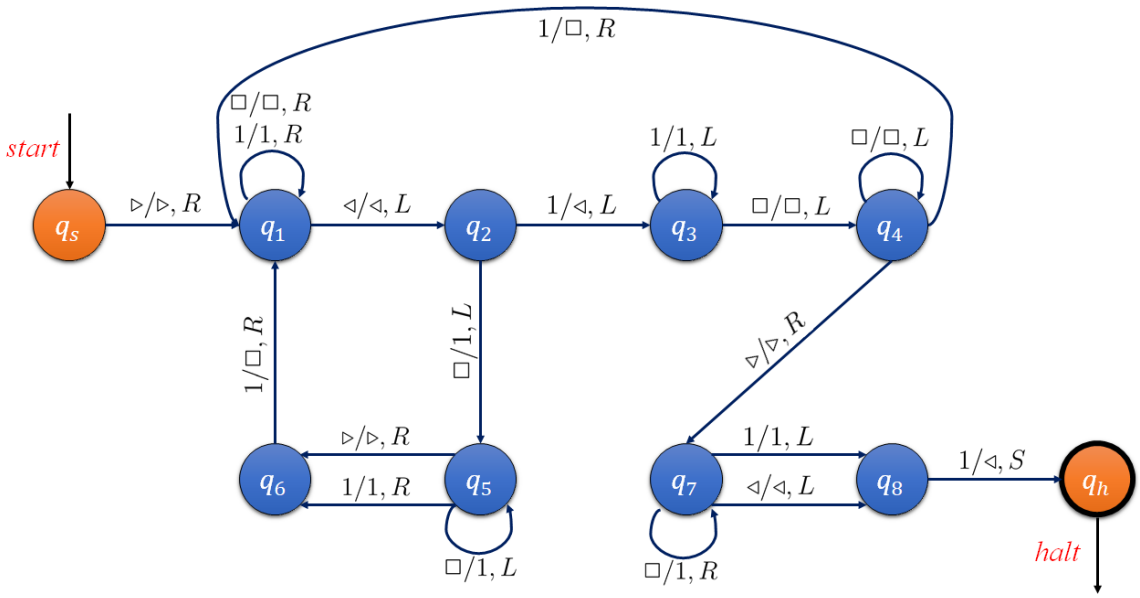

Figure 1: The state transition diagram

(c) The whole process from initial to final configurations for input $x = 7$ and $y = 3$ is as follows (some useless parts of the tape are automatically ignored, and we will use the blue color to mark them one step before they are ignored).

$(q_s, \underline{\triangleright}1111111\square111\triangleleft) \vdash (q_1, \triangleright\underline{1}111111\square111\triangleleft) \vdash^* (q_1, \triangleright1111111\square111\underline{\triangleleft})$
$\vdash (q_2, \triangleright1111111\square111\underline{\triangleleft}) \vdash (q_3, \triangleright1111111\square11\underline{1}\triangleleft\triangleleft) \vdash^* (q_3, \triangleright1111111\underline{\square}11\triangleleft)$
$\vdash (q_4, \triangleright1111111\underline{1}\square11\triangleleft) \vdash (q_1, \triangleright111111\square\underline{\square}11\triangleleft) \vdash^* \cdots\cdots$ (the same pattern (pattern 1))
$\vdash^* (q_1, \triangleright1111\square\underline{\square}\square\square\triangleleft) \vdash^* (q_1, \triangleright1111\square\square\square\underline{\triangleleft}) \vdash (q_2, \triangleright1111\square\square\square\underline{\square}\triangleleft)$
$\vdash (q_5, \triangleright1111\square\square\underline{\square}1\triangleleft) \vdash^* (q_5, \triangleright111\underline{1}1111\triangleleft) \vdash (q_6, \triangleright1111\underline{1}111\triangleleft)$
$\vdash (q_1, \triangleright1111\square\underline{1}11\triangleleft) \vdash^* \cdots\cdots$ (the same pattern (pattern 2))
$\vdash^* (q_1, \triangleright1\square\underline{1}11\triangleleft) \vdash^* \cdots\cdots$ (the same pattern (pattern 1))
$\vdash^* (q_1, \triangleright\square\underline{\square}11\triangleleft) \vdash^* (q_1, \triangleright\square\square11\underline{\triangleleft}) \vdash (q_2, \triangleright\square\square1\underline{1}\triangleleft) \vdash (q_3, \triangleright\square\square\underline{1}\triangleleft\triangleleft) \vdash (q_3, \triangleright\square\underline{\square}1\triangleleft)$
$\vdash (q_4, \triangleright\underline{\square}\square1\triangleleft) \vdash (q_4, \underline{\triangleright}\square\square1\triangleleft) \vdash (q_7, \triangleright\underline{\square}\square1\triangleleft) \vdash^* (q_7, \triangleright11\underline{1}\triangleleft) \vdash (q_8, \triangleright1\underline{1}1\triangleleft)$
$\vdash (q_h, \triangleright1\underline{\triangleleft}1\triangleleft) \Rightarrow (q_h, \triangleright1\underline{\triangleleft})$

In the process above,

- pattern 1 means eliminating the last "1" of both x's representation and y's representation;
- pattern 2 means eliminating y "1"s in x's representation.

The final tape is $\cdots\triangleright 1\underline{\triangleleft}\cdots$, which indicates $7 \mod 3 = 1$, and it is correct.

□

2. Assume there's a Turing Machine M using alphabet $\Gamma : \{\triangleright, \square, a, b, \cdots, z\}$. We can simulate M by a Turing Machine $\tilde{M}$ using alphabet $\tilde{\Gamma} : \{\triangleright, \square, 0, 1\}$. Please transform the instruction $\langle q, i\rangle \to \langle q', j, R\rangle$ in M into its corresponding form in $\tilde{M}$.

Solution. If we map $a, b, \cdots, z$ to $1, 2, \cdots, 26$ in order, then we need five 01-bit to represent the letters. Thus, i can be represented as 01001, and j can be represented as 01010. Therefore, when using transition $\langle q, i\rangle \to \langle q', j, R\rangle$ in Turing Machine $\tilde{M}$, the initial tape state is as follows (Fig. 2).

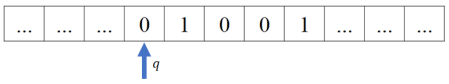

Figure 2: The initial tape state of $\tilde{M}$ before transition

And the final tape state is as follows (Fig. 3)

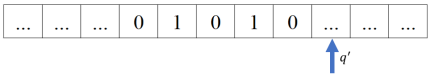

Figure 3: The final tape state of $\tilde{M}$ after transition

Here are the corresponding transitions of transition $\langle q, i\rangle \to \langle q', j, R\rangle$ in $\tilde{M}$.

$\langle q, 0\rangle \to \langle q_0^r, 0, R\rangle$
$\langle q_0^r, 1\rangle \to \langle q_{01}^r, 1, R\rangle$
$\langle q_{01}^r, 0\rangle \to \langle q_{010}^r, 0, R\rangle$
$\langle q_{010}^r, 0\rangle \to \langle q_{0100}^r, 0, R\rangle$
$\langle q_{0100}^r, 1\rangle \to \langle q_{01001}^r, 1, S\rangle$
$\langle q_{01001}^r, 1\rangle \to \langle q_{01010}^{\prime wR}, 1, S\rangle$
$\langle q_{01010}^{\prime wR}, 0/1\rangle \to \langle q_{0101}^{\prime wR}, 0, L\rangle$
$\langle q_{0101}^{\prime wR}, 0/1\rangle \to \langle q_{010}^{\prime wR}, 1, L\rangle$
$\langle q_{010}^{\prime wR}, 0/1\rangle \to \langle q_{01}^{\prime wR}, 0, L\rangle$
$\langle q_{01}^{\prime wR}, 0/1\rangle \to \langle q_0^{\prime wR}, 1, L\rangle$
$\langle q_0^{\prime wR}, 0/1\rangle \to \langle q_5^{\prime R}, 0, S\rangle$
$\langle q_5^{\prime R}, 0/1\rangle \to \langle q_4^{\prime R}, 0/1, R\rangle$
$\langle q_4^{\prime R}, 0/1\rangle \to \langle q_3^{\prime R}, 0/1, R\rangle$
$\langle q_3^{\prime R}, 0/1\rangle \to \langle q_2^{\prime R}, 0/1, R\rangle$
$\langle q_2^{\prime R}, 0/1\rangle \to \langle q_1^{\prime R}, 0/1, R\rangle$
$\langle q_1^{\prime R}, 0/1\rangle \to \langle q', 0/1, R\rangle$

We can divide the transitions above into three stages: reading stage, transition stage, writing stage and moving stage. Here are the reading stage, transition stage, writing stage and moving stage of the instruction $\langle q, i\rangle \to \langle q', j, R\rangle$ (Fig. 4, Fig. 5, Fig. 6 and Fig. 7).

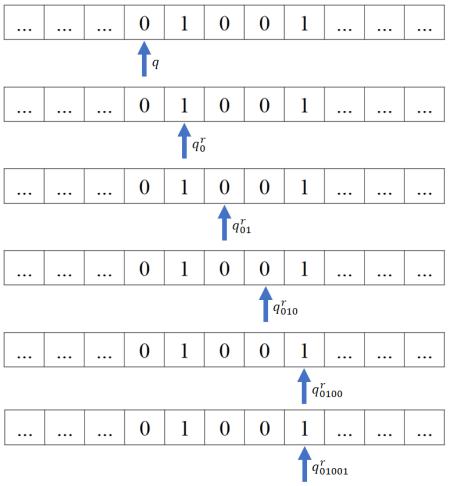

Figure 4: The reading stage of the given instruction

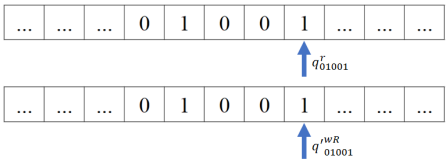

Figure 5: The transition stage of the given instruction

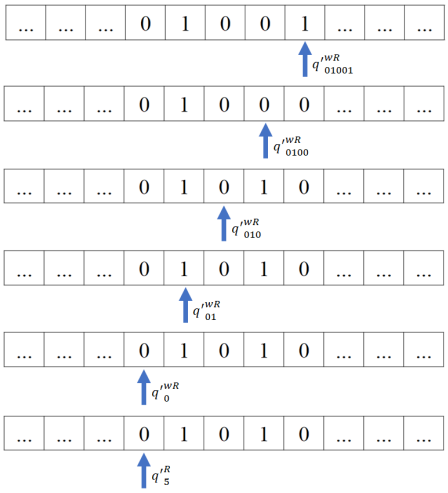

Figure 6: The writing stage of the given instruction

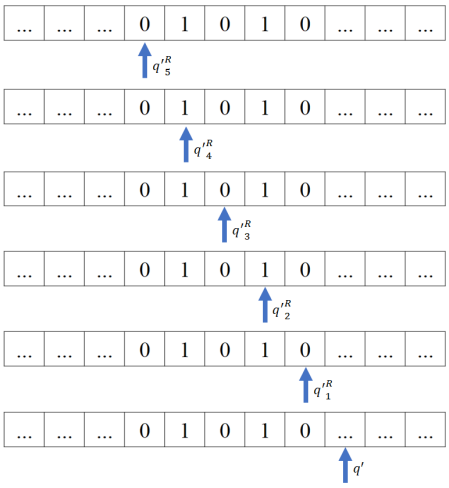

Figure 7: The moving stage of the given instruction

If we denote the five continuous blocks which contains the data as a *macro-block*, then according the figures above, the reading stage is basically reading the data represented in macro-block; the transition stage is basically perform the corresponding transition of M in $\tilde{M}$; the writing stage is basically writing the data to the macro-block; and the moving stage is basically moving to the next macro-block. Therefore, the transitions above of $\tilde{M}$ have the same function as the original transition $\langle q, i\rangle \to \langle q', j, R\rangle$ of M. □

3. **Wireless Data Broadcast System.** In a Wireless Data Broadcast System (WDBS), data items are repeatedly broadcasted in cycle on different channels. Denote $D = \{d_1, d_2, \cdots, d_k\}$ as data items, each d_i with length l_i (as time units), and $\mathbf{C} = \{C_1, C_2, \cdots, C_n\}$ as broadcasting channels. Fig. 8 illustrates a WDBS with 25 data items and 4 channels. Once a channel finishes broadcasting current cycle, it will repeat these data again as a new cycle. E.g., a possible broadcasting sequence of C_1 could be $\{d_6, d_{12}, d_1, d_{18}, d_7, d_6, d_{12}, d_1, d_{18}, d_7, \cdots\}$

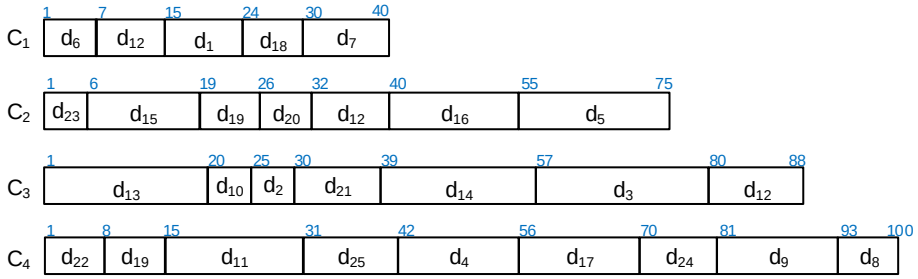

Figure 8: An Example Scenario of Wireless Data Broadcast System.

If a mobile client requires a subset of data items $D_q \subseteq D$ from this WDBS, he/she must access onto one channel, wait for the appearance of one required item, and switch to another channel if necessary. Each "switch" requires one time slot. For example, Lucien wants to download $\{d_1, d_3, d_5\}$, as shown in Fig. 9. He firstly accesses onto C_1 at time slot 1, then download d_1, d_3 respectively during time slots 2 to 5, and then switch to C_3 at time slot 6 (note that he cannot download d_5 from C_2 because of the switch constraint), and download d_5 during time slots 7 to 8. We define *access latency* as the period when a client starts downloading, till the time he/she finishes. As a result, the overall access latency for Lucien is 7 in this example.

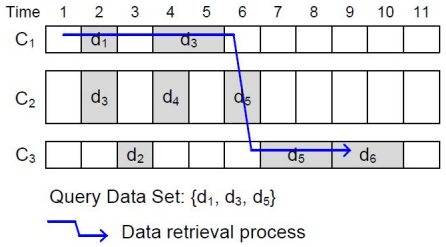

Figure 9: An Example Scenario of Query of a Client.

Each operation (download/wait/switch) needs energy consumption. To conserve energy, a client hopes to use minimum amount of energy to download all required items in D_q, which means that he/she waits to minimize both access latency and switch numbers. Unfortunately, these two objectives conflict with each other naturally. Fig. 10 exhibits such a scenario. To download $D_q = \{d_1, d_2, d_3, d_4\}$, if we start from C_2, in Option 1 we can switch to C_1 for d_1 immediately after downloading d_3, return back to C_2 for d_4, and to C_1 again for d_2. Such option costs 3 switches and 7 access latency. While in Option 2, we stay at C_2 lazily for d_3 and d_4, and then switch to C_1 for d_2 and d_1. Such option costs 1 switches and 12 access latency.

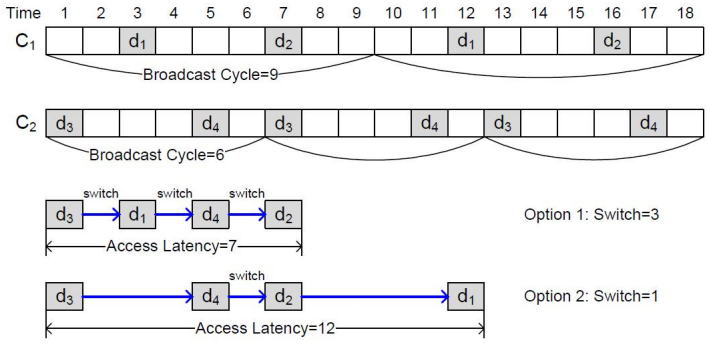

Figure 10: Confliction between Access Latency and Switch Number.

Once we want to minimize two conflictive objectives simultaneously, we have three possible ways (similar as Segmented Least Squares told in Dynamic Programming Lecture). Now it is your turn to complete the formulation of this optimization, we name it as Minimum Constraint Data Retrieval Problem (MCDR), with the following sub-questions.

(a) If we add an additional switch parameter h, please define the MCDR (Version 1) completely as a search problem.

(b) If we add an additional latency parameter t, please define the MCDR (Version 2) completely as a search problem.

(c) If we set dimensional parameters α to switch number, and β to access latency, we can combine two objectives together linearly as a new concept "cost". Please define the Minimum Cost Data Retrieval Problem (MCDR, Version 3) correspondingly.

(d) Please give the decision versions of sub-questions (a), (b) and (c).

Solution. Here are my answers to the sub-questions.

(a) We make some definitions about switch and access latency first.

Definition 1 (Switch)**.** *A switch is changing the current channel to another channel, and it will cost one slot.*

Definition 2 (Access latency)**.** *Downloading each of the data items needs some time slots, and the number of time slots is equal to the length of the data item; every switch needs one time slot; we also need some extra time slots to waiting for target data items. The access latency is the sum of these time slots.*

Here is the definition of the search problem of MCDR (Version 1).

Definition 3 (MCDR Version 1, Search Problem)**.** *Given k data items $D = \{d_1, d_2, \cdots, d_k\}$ located in n different broadcasting channels $\mathbf{C} = \{C_1, C_2, \cdots, C_n\}$, and the target data items set $D_q \subseteq D$. Each data item d_i has its length l_i, and it is located in some positions in some channels. Each channel has a different broadcast cycle length. Given a switch parameter h, then the Minimum Constraint Data Retrieval Problem is to find a minimum access latency data retrieval schedule under the constraint of at most h switches.*

(b) Here is the definition of the search problem of MCDR (Version 2).

Definition 4 (MCDR Version 2, Search Problem)**.** *Given k data items $D = \{d_1, d_2, \cdots, d_k\}$ located in n different broadcasting channels $\mathbf{C} = \{C_1, C_2, \cdots, C_n\}$, and the target data items set $D_q \subseteq D$. Each data item d_i has its length l_i, and it is located in some positions in some channels. Each channel has a different broadcast cycle length. Given an access*

latency parameter t, then the Minimum Constraint Data Retrieval Problem is to find a minimum switch number data retrieval schedule under the constraint of at most t access latency.

(c) Then we define the cost of a data retrieval schedule as follows.

Definition 5 (Cost)**.** *Given a data retrieval schedule S, and two dimensional parameters α to switch number and β to access latency. Suppose the switch number of the schedule S is h, and the access latency of the schedule S is t, then the cost c of the schedule S is defined as follows.*

$$c \triangleq \alpha h + \beta t$$

Here is the definition of the search problem of MCDR (Version 3).

Definition 6 (MCDR Version 3, Search Problem)**.** *Given k data items $D = \{d_1, d_2, \cdots, d_k\}$ located in n different broadcasting channels $\mathbf{C} = \{C_1, C_2, \cdots, C_n\}$, and the target data items set $D_q \subseteq D$. Each data item d_i has its length l_i, and it is located in some positions in some channels. Each channel has a different broadcast cycle length. Given two dimensional parameters α to switch number and β to access latency, then the Minimum Cost Data Retrieval Problem is to find a minimum cost data retrieval schedule.*

(d) The decision versions of previous MCDR problems are as follows.

Definition 7 (MCDR Version 1, Decision Problem)**.** *Given k data items $D = \{d_1, d_2, \cdots, d_k\}$ located in n different broadcasting channels $\mathbf{C} = \{C_1, C_2, \cdots, C_n\}$, and the target data items set $D_q \subseteq D$. Each data item d_i has its length l_i, and it is located in some positions in some channels. Each channel has a different broadcast cycle length. Given a switch parameter h and an access latency parameter t, then the Minimum Constraint Data Retrieval Problem is whether there exists a data retrieval schedule whose access latency is no more than t, under the constraint of at most h switches.*

Definition 8 (MCDR Version 2, Decision Problem)**.** *Given k data items $D = \{d_1, d_2, \cdots, d_k\}$ located in n different broadcasting channels $\mathbf{C} = \{C_1, C_2, \cdots, C_n\}$, and the target data items set $D_q \subseteq D$. Each data item d_i has its length l_i, and it is located in some positions in some channels. Each channel has a different broadcast cycle length. Given an access latency parameter t and a switch parameter h, then the Minimum Constraint Data Retrieval Problem is whether there exists a data retrieval schedule whose switch number is no more than h, under the constraint of at most t access latency.*

Definition 9 (MCDR Version 3, Decision Problem)**.** *Given k data items $D = \{d_1, d_2, \cdots, d_k\}$ located in n different broadcasting channels $\mathbf{C} = \{C_1, C_2, \cdots, C_n\}$, and the target data items set $D_q \subseteq D$. Each data item d_i has its length l_i, and it is located in some positions in some channels. Each channel has a different broadcast cycle length. Given two dimensional parameters α to switch number and β to access latency and a cost parameter c, then the Minimum Cost Data Retrieval Problem is whether there exists a data retrieval schedule whose cost is no more than c.*

□