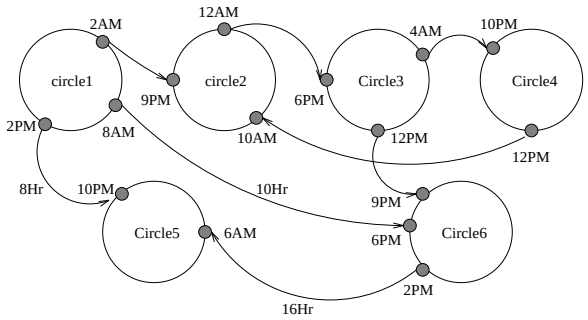
C-12.21

There are two possible solutions:

1.
   - For each airport $a_i \in \mathcal{A}$, draw a circle $circle_i$ to represent the time (24 hours).
   - For each flight $f \in \mathcal{F}$, find the origin airport a_o, the destination airport a_d, the departure time t_d, and the arrival time t_a. Draw a vertex v_1 on $circle_o$ marked the time t_d, also draw a vertex v_2 on $circle_d$ marked the time $(t_a+\mathrm{c}(a_d))$. Draw an directed edge from v_1 to v_2 with weight $t_a+\mathrm{c}(a_d)$-t_d (of course we must compute the correct weight (flight time) in case like the departure time is 10:00PM and the arrival time is 1:00AM next day). The direction of edges on a circle should be clockwise.
   - Now we have a new graph like the figure below:



   Note that we have included the minimum connecting time in the total flight time. We can start the modified new flights without worrying about the connecting times.
   Then to solve this problem is to find the shortest path from the first vertex on $circle_a$ (origin airport a) representing time t or after t to one vertex on $circle_b$ (destination airport b) in the graph. The flight sequences can be obtained from the shortest path from origin to destination.
2. The following algorithm finds the minimum travel time path from airport $a \in \mathcal{A}$ to airport $b \in \mathcal{A}$. Recall, we must depart from a at or after time t. Note, "$\oplus$" and "$\preceq$" are operations which we must implement. We define these operations as follows: If $x \oplus y = z$, then z is the point

in time (with date taken into consideration) which follows x by y time units. Also, if $a \preceq b$, then a is a point in time which precedes or equals b. These operations can be implemented in constant time.

initialize fringe to empty
initialize incoming_flight(x) to nil for each $x \in \mathcal{A}$
earliest_arrival_time(a)=t
earliest_arrival_time(x)=∞ for $x \in \mathcal{A}$ where $x \neq a$
repeat
    remove from fringe airport x such that $earliest\_arrival\_time(x)$ is soonest
    if $x \neq b$ then
        if $x = a$ then
            time_can_depart=t
        else
            time_can_depart=earliest_arrival_time(x) $\oplus$ c(x)
        for each flight $f \in \mathcal{F}$ such that $a_1(f) = x$
            if $time\_can\_depart \preceq t_1(f)$ and $t_2(f) \preceq earliest\_arrival\_time(a_2(f))$ then
                earliest_arrival_time($a_2(f)$)=$t_2(f)$
                incoming_flight($a_2(f)$)=f
                add $a_2(f)$ to the fringe if it is not yet there
until $x = b$

initialize *city* to b
initialize $flight\_sequence$ to *nil*
while (city$\neq a$) do
    append $incoming\_flight(city)$ to $flight\_sequence$
    assign to *city* the departure city of $incoming\_flight(city)$

reverse $flight\_sequence$ and output

The algorithm essentially performs Dijkstra's Shortest Path Algorithm (the cities are the vertices and the flights are the edges). The only small alterations are that we restrict edge traversals (an edge can only be traversed at a certain time), we stop at a certain goal, and we output the path (a sequence of flights) to this goal. The only change of possible

consequence to the time complexity is the flight sequence computation (the last portion of the algorithm). A minimum time path will certainly never contain more that M flights (since there are only M total flights). Thus, we may discover the path (tracing from b back to a) in $O(M)$ time. Reversal of this path will require $O(M)$ time as well. Thus, since the time complexity of Dijkstra's algorithm is $O(M \log N)$, the time complexity of our algorithm is $O(M \log N + M + M) = O(M \log N)$. Note: Prior to execution of this algorithm, we may (if not given to us) divide $\mathcal{F}$ into subsets $F_1, F_2, \ldots, F_N$, where F_i contains the flights which depart from airport i. Then when we say "for each flight $f \in \mathcal{F}$ such that $a_1(f) = x$", we will not have to waste time looking through flights which do not depart from x (we will actually compute "for each flight $f \in F_x$"). This division of $\mathcal{F}$ will require $O(M)$ time (since there are M total flights), and thus will not "slow down" the computation of the minimum time path (which requires $O(M \log N)$ time).