Algorithm 1: Online Association Algorithm

Input: User i with time demand $\{T_{i1}, T_{i2}, ..., T_{iN}\}$ and priority w_i
Output: user association $\{x_{i1}, x_{i2}, ...x_{iN}\}$
Initialisation: $x_{ij} = 0, \quad \forall 1 \leq j \leq N$;

```
-if WaitingWork().empty==False and WaitingWork().QueueHead.priority ≥ w_i then
    WaitingWork.add(T_i1, T_i2, ..., T_iN; w_i)
else
    j_min = argmin_j([Σ_{k=1}^{M} x_kj.T_kj] + T_ij) /* choose the less loaded AP */
    if [(Σ_{k=1}^{M} x_{kj_min}.T_{kj_min}) + T_{ij_min}] ≤ 1 then
        x_{ij_min} = 1 ;
    else
        /* Time demand exceeds the remaining available time */
        ChooseAP=1, chooseDetail=priorityAllowDesassociate(AP1,T_i1, w_i) ;
        for j=1 to N do
            k=priorityAllowDesassociate(AP_j, T_ij, w_i) ;
            if k.response==True then
                /* The current candidate AP has less priority clients to be removed */
                if chooseDetail.response==False then
                    /* It's the first available AP */
                    chooseAP=j, chooseDetail=k ;
                else
                    if k.directDisassociate==True then
                        if chooseDetail.directDisassociate==False then
                            /* The First available AP allowing direct disassociation */
                            chooseAP=j, chooseDetail=k ;
                        else
                            /* choose the better AP through the less priority disassociated clients */
                            if last-elem.prio(k)> last-elem.prio(chooseDetail) then
                                chooseAP=j, chooseDetail=k ;
                    else
                        /* choose the better AP through the less priority disassociated clients */
                        if (chooseDetail.directDisassociate==False) and
                          (last-elem.prio(k)> last-elem.prio(chooseDetail)) then
                            chooseAP=j, chooseDetail=k ;
            else
                R_1 = k.period * c[i][j];
                R_2 = chooseDetail.period * c[i][chooseAP];
                if (chooseDetail.response==False and R_1 > R_2) then
                    /* The total demand of the current client is not available but currently, this is the
                       better AP */
                    chooseAP=j, chooseDetail=k ;
        if chooseDetail.response==False then
            /* The current client could not be associated immediately, because higher priority clients could not be
               disassociated, and should be put on the waiting queue. The client could choose to start working
               with the available current period */
            WaitingWork(T_i1, T_i2, ...T_iN, w_i) /* The waiting clients are put in a priority based queue,
               for the same priority clients a FIFO is used */
        else
            if chooseDetail.DirectDisassociate==False then
                /* less priority clients exist, but should be disassociated when they finish the current critical
                   task, the client is put in the queue and associated as possible, or could choose to start
                   working with the chosen AP with the current available period */
                WaitingWork(T_i1, T_i2, ...T_iN, w_i)
            else
                Disassociate(chooseAP,ChooseDetail.index) /* Disassociate clients from the
                   current index until the tail of the Queue of clients */
                j=chooseAP, x_ij = 1 ;
```