Appendix I

Schema of the Giffler And Thompson Algorithm for Schedule Generation

Giffler() /* Loop for reading or input job data and initialisation */
BEGIN:
input : no. of jobs n; no. of WC's m;
for ( i = 1 to n)
{
for k = 1 to m
{
Input: j (WC no. for kth operation of i); Ti (processing time of i on WC j);
}
Input: Di (Due date of job i);
}
set: old datum time ODT = 0; q = n * m; p = 1;
initialise array 'x' = {0} ; set : k = 1 ;
/* Starting table with the entry of all immediate operations in array x_{pq} */
for( i = 1 to n)
{
find : j (WC no. Corresponding to operation k = 1); q = ((j - 1)*n + i); x_{pq} = T_q;
}
START:
find: datum time 'DT' [minimum of (x_{pq} ≥ ODT)];
find: Conflict WorkCell no. cfm (WC number 'j' corresponding to x_{pq} = DT);
find: number of conflict jobs Ncfj (i.e. the jobs Ci with (x_{pq} ≥ DT) in the cfm);
/* Selection of one job from contenders and returns the conflict job Ci */
If Ncfm > 1, then
{
Conflict fn();
Goto LOAD;
}
else set the job as Ci in cfm; /* Procedure for loading the job Ci on the cfm*/
LOAD:
for the job Ci selected
{
find: k, j and q ;
set: $A = x_{pq}$;
check: Is there any more operations for $C_i$?
if 'yes', then
{
    get: WC no. $j$ for $(k+1)^{th}$ operation of the job $C_i$;
    find: processing time $T_{Cj}$;
    $qq = (j - 1) * n + C_i$;
    set: $x_{p(qq)} = x_{p(qq)} + T_{Cj}$;
}
for all other jobs contended for the cfm
{
    find: $k, j, q$;
    set: $x_{pq} = A + T_q$;
}
find: maximum value of the revised $x_{pq}$ and set as MAX;
if (MAX = DT and all tasks are over) then goto PRINT:
set: $ODT = DT$; goto START:
PRINT: /* Loop for the derivation of schedule i.e. Start and Finish time */
for ($i = 1$ to $n$)
{
    for $k = 1$ to $m$
    {
        find: WC number $j$;
        $q = (j - 1) * n + i$;
        $FT_{ik} = x_{pq}$;
        $ST_{ik} = x_{pq} - T_q$;
    }
}
END:
Conflict_fn() /* Sub program for resolving conflict */
{ select a job among the contenders arbitrarily;
    return the selected job as $C_i$;
}