COMSOAL

COmputer **M**ethod of **S**equencing **O**perations for **A**ssembly **L**ines

- Algoritmo para la resolución de problemas SALB-1

-> conozco el TC, busco determinar el número de estaciones de trabajo.

Algunas consideraciones iniciales:

- Selección aleatoria de tareas a las ET.
- Se busca la mejor solución “a prueba y error”
- Se prueban diferentes combinaciones de asignaciones de tareas a ET, y se va guardando la mejor solución.
- La “mejor solución” hallada por el algoritmo se actualizará sólo si la solución actual mejora le eficiencia de la considerada como mejor hasta el momento.

Implementación del algoritmo

Qué tenemos que lograr con la programación de COMSOAL.

1-Definir parámetros y variables.

2-Determinar qué tareas se encuentran habilitadas para ser asignadas. Esto implica que puedan ingresar por tiempo a la ET bajo análisis y también por precedencias.

3-Determinar qué tarea es la que va a ingresar a la ET. Elección aleatoria.

4-Asignar la tarea a la ET bajo análisis. Actualizar tiempos de la ET.

5-Actualizar el grupo de tareas candidatas.

6-Repetir hasta que todas las tareas estén asignadas a una ET.

7-Calcular la eficiencia de la solución actual y guardar la solución sólo si mejora la eficiencia de la solución considerada como la mejor hasta el momento.

8-Display de la mejor solución obtenida. (max eficiencia)

1-Definir de parámetros y variables

```
//DEFINICIÓN DEL TC
TC=40; // tiempo de ciclo determinado

//DEFINICIÓN DEL NÚMERO TOTAL DE ACTIVIDADES Y DURACIÓN DE CADA UNA
nact=15; //cantidad de actividades a asignar

duracion = [10 12 7 8 20 4 11 6 9 12 15 13 9 8 9]; //duracion de cada actividad

//MATRIZ DE RELACIONES DE PRECEDENCIA. vale 1 si j precede a i

predecesores = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 1 1 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 1 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 1 1 1 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 1 0 1 0 0 0 0 0 0;
    0 0 0 0 0 0 1 1 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 1 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 1 1 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 1 0 0;
    0 0 0 0 0 0 0 0 0 0 0 1 0 1 0];

//Definición de variables, vectores y matrices requeridos para expresar la solución final
eficienciamaxima=0; //variable de eficiencia
solucionestActs=zeros(nact,nact);//matriz solución que guarda qué actividades se relizan en
                                 //cada ET, colocando indice que identifica tarea

solucioncapET=zeros(1,nact); // vector solución que indica la capacidad de cada ET
solucionnET=0; // solucion de la cantidad de ET
```

Dimensión [nact,nact]

Forma de comparar soluciones

Suma de los tiempos de las tareas asignadas

Máxima cantidad de ET posibles=nact

1 4 7 3 6

2 5 8

Cantidad de soluciones que se van a comparar

De la solución en cada iteración

```
//Comienzo de cada iteración
for g=1:500; //cantidad de iteraciones que se quieren realizar

opAsig= zeros (1,nact); // vector inicial en cero que indica 1 en la posición i,
                        // si la tarea i ya fue asignada a alguna ET

capET=zeros(1,nact); //vector que indica la capacidad          de cada ET.



estActs=zeros(nact,nact); //matriz p guardar que actividades se realiza en
                          //cada ET, colocando indice que identifica tarea

nET=1; //inicialización del índice utilizado para recorrer las ET, donde se guarda
       //el número de ET utilizadas.

w= zeros(1,nact); // operaciones sin predecedores o cuyos predecedores
                  // ya fueron asigandos con tiempos de operacion menores o
                  // iguales al TO de la ET.

asignacionTerminada=0; // variable para terminar el lazo while.

I=0; // variable aux utilizada para guardar las acts que realiza cada ET.
```

2-Determinar qué tareas se encuentran habilitadas para ser asignadas a estaciones de trabajo.

```
while (asignacionTerminada==0)
    for i=1:nact; // recorrer todas las act p ver si pueden ingresar a w
        if(opAsig(i)==0) //si la operacion no fue aun asignada
            j=1;
            puedeEjecutarse=1; //var auxiliar que vale 1 si la act i
                               // esta en condiciones de ejecutarse
            while (j<=nact) //recorrer todas las actividades
?????
?????                   // si no tiene predecesores o el predecesor ya fue asignado
                else
                    j=nact + 1;
                    puedeEjecutarse=0;
                end
            end

            if (puedeEjecutarse==1 & (duracion(i)+ capET(nET)<= TC))
                // si act i cumple con los req de predecesores y
                //tiene una duracion menos al tiempo remanente en la ET
                //puede incluirse dentro de las candidatas
??????????
            end
        end
    end
```

Se comienza a recorrer la matriz

Valdrá 0, cuando la tarea no pueda ejecutarse

Salgo del while

Agregar la tarea al conjunto w.

3-Determinar qué tarea es la que va a ingresar a la ET. Elección aleatoria.

```
aux=sum(w); //variable auxiliar para determinar que actividad se va a seleccionar

if ((aux>0)) //si en w hay componentes
????    aleat= ??????? ; //se elige aleatoriamente una actividad
    k=0; //k e i variables auxiliares
    i=0;
    while (k<aleat) //se busca la actividad nro aleat de las asignadas a w
        i=i+1;
        k=k+w(i);
    end

    I=I+1;
```

Cantidad total de tareas posibles a asignar

k suma cuando w(i)=1
i guarda el índice de la tarea

Asigno una tarea más a la ET

4-Asignar la tarea a la ET bajo análisis. Actualizar tiempos de la ET.

```
        opAsig(i)=1; //se asigna la act elegida
        w(i)=0; // se quita de w
??????????? ; //actualizar capacidad ocupada en la estacion vigente
        estActs(nET,I)=i; //se guarda q act se hace en cada ET
```

5-Actualizar el grupo de tareas candidatas.

```
//Actualización del vector w
    for k=1:nact; //se actualiza w sacando las actividades donde su tiempo es mayor que la capacidad de la ET
        if((w(k)==1 & (duracion(k) + capET(nET)>  TC)))
            w(k)=0;
        end
    end
```

Quito del conjunto w las tareas que ya no ingresan por tiempo.
El conjunto w se volverá a formar (incorporando tareas que se habiliten por precedencias o en caso de una nueva ET por tiempos), pero como estas tareas ya tenían asignado un 1, se deben volver a 0.

6-Repetir hasta que todas las tareas estén asignadas a una ET.

```
        if (sum(opAsig)==nact) // se asignaron todas las actividades?
            asignacionTerminada=1;
        end

    else
        nET=nET+1;
        I=0;
    end
end
```

Si ya se superó o igualó TC, empiezo la asignación de tareas a otra ET

7-Calcular la eficiencia de la solución actual y guardar la solución sólo si mejora la eficiencia de la solución considerada como la mejor hasta el momento.

```
estActs;
capET;
nET;
TCT=max(capET);
aux2=sum(capET)/(TCT*nET); //cálculo de la eficiencia
if (aux2>eficienciamaxima) // para determinar si es mas eficiente que la anterior
    eficienciamaxima=aux2;
    solucionestActs=estActs;
    solucioncapET=capET;
    solucionEst=nET;
end

end;
```

La solución se actualiza sólo si mejora la eficiencia de la actual

8-Display de la mejor solución obtenida.

```
//se muestra la mejor solución
disp('La eficiencia maxima de la linea es:')
disp(eficienciamaxima)
disp('La distribucion de las tareas en cada estacion de trabajo es:')
disp(solucionestActs)
disp('La capacidad utilizada en cada estacion de trabajo es:')
disp(solucioncapET)
disp('La cantidad de estaciones de trabajo utilizada es:')
disp(solucionEst)
disp('El tiempo de ciclo de la linea es:')
disp(max(solucioncapET))
```