Ampliación de la Programación

M Procesadores

Backtracking

M Procesadores: Backtracking

Resolución

En cada etapa k decidimos el procesador en el que realizamos el trabajo k supuesto que ya tengamos decididos los k-1 trabajos anteriores.

- **Test Solución**: Etapa *N* (cuando hayamos tomado las decisiones para todos los trabajos).
  - **Comprobaremos** entonces si hemos o no mejorado nuestra mejor solución (*Test Fracaso –No Fracaso-*, para la mejor solución).
- **Test Fracaso**: No hay en la construcción de una solución, siempre podemos hacer el trabajo *k* en uno de los procesadores. Utilizaremos sólo el test usado para decidir si es una mejor solución.
- **Generación de Descendientes**: Para el trabajo *k* tenemos M alternativas, hacerlo en cada uno de los procesadores.

M Procesadores: Backtracking

Resolución Mejorada

En cada etapa k decidimos el procesador en el que realizamos el trabajo k supuesto que ya tengamos decididos los k-1 trabajos anteriores.

- **Test Solución**: Etapa *N* (cuando hayamos tomado las decisiones para todos los trabajos).
  - **Comprobaremos** entonces si hemos o no mejorado nuestra mejor solución (*Test Fracaso –No Fracaso-*, para la mejor solución).
- **Test Fracaso**: Si en una determinada etapa la solución que estamos construyendo *no es* mejor que la mejor que hemos construido hasta el momento, *no seguimos*.
- **Generación de Descendientes**: Para el trabajo *k* tenemos M alternativas, hacerlo en cada uno de los procesadores. Bastará con un **for**

```
for (p=0; p<M; p++)  // p procesador que realiza el trabajo k
```

M Procesadores: Backtracking

Resolución

```
#include <stdio.h>
#define DIMMAX 20
#define INFINITO 10000
#define TRUE 1
#define FALSE 0

  typedef struct {
     int coste;
     int trabajo [DIMMAX]; // trabajo[i] es del procesador que hace el trabajo i
   } TipoSolucion;

  void MProcesadores (int k, int tproc[DIMMAX], TipoSolucion *sol, TipoSolucion *solOptima,
                                 int M, int N, int tiempo[DIMMAX][DIMMAX]);

  void Inicializa (int tproc[DIMMAX], TipoSolucion *sol, int M);


  void Salida (TipoSolucion sol, int M, int N, int tiempo[DIMMAX][DIMMAX]);
  void Entrada (int *M, int *N, int tiempo[DIMMAX][DIMMAX]);
```

M Procesadores: Backtracking

Resolución

```
void Inicializa (int tproc[DIMMAX], TipoSolucion *sol, int M) {
   int i;

   for (i=0; i<M; i++) tproc[i] = 0;
   sol->coste = INFINITO;
}


int maximo (int tproc[DIMMAX], int M) {
   int i, max;

   max = tproc[0];
   for (i=1; i<M; i++) if (max < tproc[i]) max = tproc[i];
   return max;
}
```

M Procesadores: Backtracking

Resolución

```
int main () {

   int tiempo [DIMMAX][DIMMAX]; // matriz de costes
   int M, N;
   TipoSolucion sol, solOptima;
   int tproc [DIMMAX];

   Entrada(&M,&N,tiempo);

   Inicializa (tproc,&solOptima,M);

   MProcesadores(0,tproc,&sol,&solOptima,M,N,tiempo);

   Salida(solOptima,M,N,tiempo);

   return 0;

}
```

M Procesadores: Backtracking

Resolución

```
void MProcesadores (int k, int tproc[DIMMAX], TipoSolucion *sol, TipoSolucion *solOptima,
                    int M, int N, int tiempo[DIMMAX][DIMMAX]) {
   int p;

  sol->coste = maximo (tproc,M);
  if (sol->coste < solOptima->coste)
    if (k == N) // Hemos construido una solución, y es mejor que la mejor anterior, actualizamos
      *solOptima = *sol;
    else {
      for (p=0; p<M; p++) {
        // lo hace el procesador p
        tproc[p] += tiempo[p][k];
        sol->trabajo[k] = p;
        MProcesadores (k+1,tproc,sol,solOptima,M,N,tiempo);

        // deshacemos para la siguiente iteración
        tproc[p] -= tiempo[p][k];
      }
}
```

M Procesadores: Backtracking

Resolución

```
void Entrada (int *M, int *N, int tiempo[DIMMAX][DIMMAX]) {
  int i, j;

  printf("Introduzca el número de trabajos "); scanf("%d",N);
  printf("Introduzca el número de procesadores "); scanf("%d",M);
  for (i=0; i<*M; i++) {
    printf("\nDuración de los trabajos en el procesador %d (separados por un espacio)\n",i);
    for (j=0; j<*N; j++)
      scanf(" %d", &tiempo[i][j]);
  }

  printf("\n\n");
  for (i= 0; i<*M; i++) {
    for (j=0; j<*N; j++)
      printf(" %4d",tiempo[i][j]);
    printf("\n");
  }

}
```

M Procesadores: Backtracking

Resolución

```
void Salida (TipoSolucion sol, int M, int N, int tiempo[DIMMAX][DIMMAX]) {
   int i, p;

   for (p= 0; p<M; p++) {
            printf("\nTrabajos realizados por el procesador %d: ", p);
      for (i=0; i<N; i++)
         if (sol.trabajo[i] == p) printf("%d  (+ %d ) ", i, tiempo[p][i]);
      printf("\n");
   }
   printf("\n\nEl coste de la solucion es: %d",sol.coste);

}
```