DISEÑO Y ANÁLISIS DE ALGORITMOS
Algoritmos constructivos y búsquedas por entornos

Parallel Machine Scheduling Problem with Dependent Setup Times

Alberto Cruz Luis
alu0101217734@ull.edu.es

1. Introducción

En esta práctica se implementaran algoritmos constructivos y búsquedas por entornos para el problema de planificador de máquinas paralelas con dependientes tiempos de setup. El programa se implementara en C++.

2.Algoritmos usados para resolver el problema

Para ello usaremos 4 tipos de algoritmos que iré explicando más adelante. Greedy, GRASP, Multiarranque y GVNS.

2.1 Greedy

```
Seleccionar la m tareas j1, j2, .., jm con menores valores de t0j para ser introducidas en las primeras posiciones de los arrays que forman la solucion S;
S = {A1 = {j1}, A2 = {j2}, .., Am = {jm}};
repeat
    S ∗ = S;
    Obtener la tarea-maquina-posicion que minimiza el incremento del TCT;
    Insertarla en la posicion que corresponda y actualizar S ∗ ;
until (todas las tareas han sido asignadas a alguna maquina)
Devolver S ∗ ;
```

2.2 GRASP

```
procedure grasp()
InputInstance();
for GRASP stopping criterion not satisfied →
    ConstructGreedyRandomizedSolution(Solution);
```

```
LocalSearch(Solution);
UpdateSolution(Solution,BestSolutionFound);
rof;
return(BestSolutionFound)
end grasp;
```

```
procedure ConstructGreedyRandomizedSolution(Solution)
Solution = {};
for Solution construction not done →
MakeRCL(RCL);
s = SelectElementAtRandom(RCL);
Solution = Solution ∪ {s};
AdaptGreedyFunction(s);
rof;
end ConstructGreedyRandomizedSolution;
```

2.3 Multiarranque

```
Procedure Búsqueda con Arranque Múltiple
Begin
    Genera (Solución Actual);
    Mejor Solución := Solución Actual;
    Repeat Búsqueda Local(Solución Actual);
        If Objetivo(Solución Actual) < Objetivo(Mejor Solución)
        then
            Mejor Solución := Solución Actual;
            Genera (Solución Actual);
    Until (Criterio de parada)
End.
```

2.4 GVNS

```
SH(X, k, X0)
X 0 ← X;
p ← n´umero de v´ertices en el ciclo;
r ← 0;
repeat
    Seleccionar al azar (i, j) tal que i, j ∈ {2, . . . , n};
    if (i < p and j > p) then Xij ← X 0 \ {vi} ∪ {vj};
    if (i > p and j < p) then Xij ← X 0 \ {vj} ∪ {vi};
    if (i > p and j > p) then Xij ← X 0 ∪ {vj};
    if (i < p and j < p) then Xij ← X 0 \ {vi};
    X0 ← Xij ;
    r ← r + 1
until r = k;
return X0 ;
```

```
VND(X, Xmej)
repeat
    Xmej ← X;
    k ← 1;
    repeat
        X0 ← arg min { f ( Y ) : Y ∈ N k ( X ) };
        if f ( X 0 ) < f ( X ) then
            X ← X 0;
            k ← 1;
        else
            k ← k + 1;
        end
    until k = kmax ;
until f ( X ) > f ( Xmej);
return Xmej ;
```

```
GVNS(X, Xmej)
Xmej ← X;
repeat
    k ← 1;
```

```
        repeat
                SH( Xmej , k , X0 );
                VND( X0 , X00 );
                if f ( X00 ) < f ( Xmej ) then
                        Xmej ← X00;
                        k ← 1;
                else
                        k ← k + 1;
                end
        until k = kmax;
until Criterio de parada;
return Xmej;
```

3. Implementación del programa.

El programa se ha realizado en C++ usando POO(programación orientada a objetos).

Este programa se basa en la siguiente jerarquización de clases.

La clase Graph nos servirá para guardar la información del fichero .txt del que leeremos la información del problema.

class Graph;

Estas 3 clases son para poder guardar la solucion a nuestro problema del que estará compuesto de un array de maquinas y a su vez cada maquina tendra un array de tareas.

class Solution;
class Machine;
class Task;

La clase TaskScheduler será el planificador de tareas que será el que decidirá que tipo de algoritmo utilizara.

class TaskScheduler;

La clase Experiment servirá para ejecutar el problema con un cierto algoritmo y poder guardar en ella cierta información como puede ser el tiempo que tarda ese experimento.

class Experiment;

La clase Algorithm y sus derivaciones servirán para implementar cada uno de los algoritmos que estaremos utilizando para resolver nuestro problema.

class Algorithm;
class Greedy;
class GRASP;
class MultiBoot;
class GVNS;

La clase NeighbourAlgorithm y sus derivaciones servirán para implementar cada uno de los algoritmos que estaremos utilizando para ejecutar la búsqueda local. Con ellas se propondrá una serie de formas de como calcular todos los vecinos correspondientes a una solución.

class NeighbourAlgorithm;
class ExchangeOwnMachine;
class ExchangeExternalMachine;
class ReinsertionExternalMachine;
class ReinsertionOwnMachine;

4. Resultados obtenidos

En este apartado estaremos ejecutando los diferentes problemas con los algoritmos implementados.

Greedy

Problema	Número de Tareas	Número de Máquinas	TCT	CPU
I40j_2m_S1_1.txt	40	2	13739	0.756973 ms
I40j_4m_S1_1.txt	40	4	7001	0.689912 ms
I40j_6m_S1_1.txt	40	6	5130	0.748705 ms
I40j_8m_S1_1.txt	40	8	4269	0.921791 ms

Multiarranque

ExchangeExternalMachine

Problema	Número de Tareas	Número de Máquinas	TCT	CPU
I40j_2m_S1_1.txt	40	2	14838	46.9651 ms
I40j_4m_S1_1.txt	40	4	8148	35.4772 ms
I40j_6m_S1_1.txt	40	6	5181	44.7298 ms
I40j_8m_S1_1.txt	40	8	4906	27.4973 ms

ReinsertionOwnMachine

Problema	Número de Tareas	Número de Máquinas	TCT	CPU
I40j_2m_S1_1.txt	40	2	15744	91.5602 ms
I40j_4m_S1_1.txt	40	4	8127	27.2803 ms

I40j_6m_S1_1.txt	40	6	6065	13.3088 ms
I40j_8m_S1_1.txt	40	8	5190	33.2319 ms

ReinsertionExternalMachine

Problema	Número de Tareas	Número de Máquinas	TCT	CPU
I40j_2m_S1_1.txt	40	2	17199	8.69665 ms
I40j_4m_S1_1.txt	40	4	8824	2.32823 ms
I40j_6m_S1_1.txt	40	6	6144	1.12037 ms
I40j_8m_S1_1.txt	40	8	5351	1.33062 ms

GRASP

ExchangeExternalMachine

Problema	Número de Tareas	Número de Máquinas	LRC	TCT	CPU
I40j_2m_S1_1.txt	40	2	2	14384	69.5569 ms
I40j_4m_S1_1.txt	40	4	2	7833	54.6331 ms
I40j_6m_S1_1.txt	40	6	2	5466	48.7848 ms
I40j_8m_S1_1.txt	40	8	2	4455	45.8329 ms
I40j_2m_S1_1.txt	40	2	3	15281	61.1461 ms
I40j_4m_S1_1.txt	40	4	3	8206	52.4122 ms

I40j_6m_S1_1.txt	40	6	3	5724	48.5988 ms
I40j_8m_S1_1.txt	40	8	3	4568	44.232 ms

ReinsertionOwnMachine

Problema	Número de Tareas	Número de Máquinas	LRC	TCT	CPU
I40j_2m_S1_1.txt	40	2	2	14227	138.213 ms
I40j_4m_S1_1.txt	40	4	2	7920	51.1515 ms
I40j_6m_S1_1.txt	40	6	2	5589	31.5967 ms
I40j_8m_S1_1.txt	40	8	2	4455	26.7514 ms
I40j_2m_S1_1.txt	40	2	3	14825	139.12 ms
I40j_4m_S1_1.txt	40	4	3	8167	55.3467 ms
I40j_6m_S1_1.txt	40	6	3	5817	33.1905 ms
I40j_8m_S1_1.txt	40	8	3	4676	27.1721 ms

ReinsertionExternalMachine

Problema	Número de	Número de	LRC	TCT	CPU

	Tareas	Máquinas			
I40j_2m_S1_1.txt	40	2	2	14559	21.4279 ms
I40j_4m_S1_1.txt	40	4	2	7432	17.3136 ms
I40j_6m_S1_1.txt	40	6	2	5632	15.6205 ms
I40j_8m_S1_1.txt	40	8	2	4467	15.5946 ms
I40j_2m_S1_1.txt	40	2	3	14211	21.0538 ms
I40j_4m_S1_1.txt	40	4	3	7823	18.2375 ms
I40j_6m_S1_1.txt	40	6	3	5564	16.0543 ms
I40j_8m_S1_1.txt	40	8	3	4448	15.4398 ms

GVNS

Problema	Número de Tareas	Número de Máquinas	Kmax	TCT	CPU
I40j_2m_S1_1.txt	40	2	2	14838	126.952 ms
I40j_4m_S1_1.txt	40	4	2	8148	47.5465 ms
I40j_6m_S1_1.txt	40	6	2	5128	62.6472 ms
I40j_8m_S1_1.txt	40	8	2	4906	38.7305 ms
I40j_2m_S1_1.t	40	2	3	14838	298.665 ms

xt					
I40j_4m_S1_1.txt	40	4	3	8148	251.359 ms
I40j_6m_S1_1.txt	40	6	3	5128	233.845 ms
I40j_8m_S1_1.txt	40	8	3	4906	144.99 ms