

Práctica 4: Backtraking

José Antonio Álvarez Ocete

Doble Grado en Ingeniería Informática y Matemáticas

23 de mayo de 2018

Descripción del problema: el Viajante de Comercio



Solución branch and bound - I

Solución branch and bound - II

Función de estimación utilizada:

```
double estimacion(vector<int> &candidates, vector<vector<double>> &cities){
    double result = 0;
    double row_min;

    for(int i = 0; i < cities.size()-1; i++){
        if(!visitado(i,candidates)){
            result += rowMin(i, cities[i]);
        }
    }
    return result;
}
```

Solución inicial

Para decidir la ciudad de inicio para el recorrido empleamos la siguiente función:

```
int farthest(vector<pair<double, double> > &coordinates){
    pair<double, double> centro(0,0);
    double dist, max = 0;
    int mejor = 0;

    for(int i = 0; i < coordinates.size(); i++){
        centro.first += coordinates[i].first;
        centro.second += coordinates[i].second;
    }

    centro.first /= coordinates.size();
    centro.second /= coordinates.size();

    for(int i = 0; i < coordinates.size(); i++){
        dist = distance(coordinates[i], centro);
        if(dist > max){
            max = dist;
            mejor = i;
        }
    }

    return mejor;
}
```

Backtracking

Solución de backtracking implementada:

```
void backtracking(int pos, pair<double, vector<int>> sol,
    vector<vector<double>> &cities, pair<double, vector<
    int>> &best){
    if(pos < cities.size()){
        for(int i = 1; i < cities.size(); i++){
            if(!visitado(i, sol.second)){
                sol.second[pos] = i;
                sol.first = compute_length(
                    sol.second, cities);
                if(sol.first < best.first)
                    backtracking(pos +
                        1, sol, cities,
                        best);
            }
        }
    } else if(best.first < best.first)
        best = sol;
}
```

Para estudiar la eficiencia empírica de los resultados se ha utilizado un ordenador con las siguiente características:

- CPU: Intel Pentium G3258 (2) @ 3.200GHz
- Memoria RAM: 7876MiB
- Kernel: 4.13.0-36-generic
- OS: Linux Mint 18.3 Sylvia x86_64

Cuadro: Comparativa de longitud y tiempo (seg) para cada algoritmo

Longitud	Branch and bound	Backtracking
19.9247	3E-06	2E-06
51.6735	6E-06	3E-06
59.4593	1.7E-05	5E-06
56.4986	5E-05	1.6E-05
119.193	0.000148	6.2E-05
150.371	0.000919	0.000281
255.254	0.001052	0.000763
361.223	0.003113	0.003867
464.618	0.015933	0.021076
476.531	0.014079	0.049719
586.246	0.193942	3.49537
755.058	0.046548	0.676337
1040.99	0.234971	9.97742

Comparativa gráfica

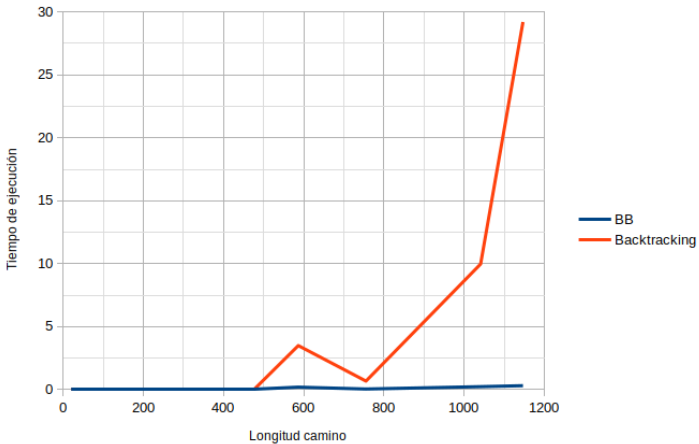


Figura: Comparativa de los algoritmos de backtracking y branch and bound

Comparativa con los algoritmos greedy

Cuadro: Resultados para un algoritmo greedy

19.9247	4E-06
51.6735	2E-06
59.4593	1E-06
62.8659	2E-06
123.26	2E-06
150.371	3E-06
300.216	3E-06
361.223	3E-06
494.908	2E-06
476.531	4E-06
794.865	4E-06
581.918	4E-06
1005.65	5E-06
1070.15	5E-06

Comparativa de caminos - I

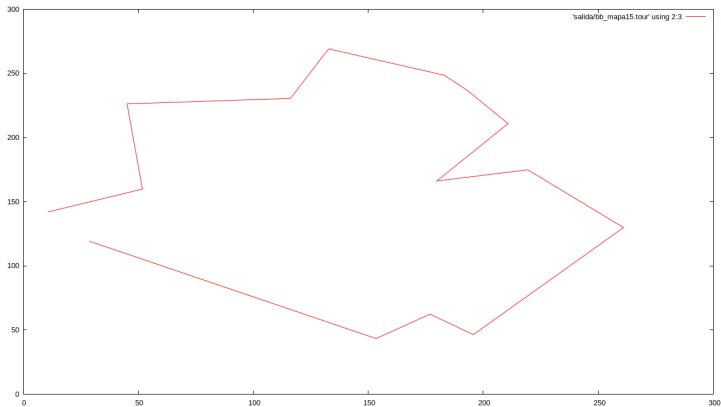


Figura: Algoritmo de branch and bound

Comparativa de caminos - II

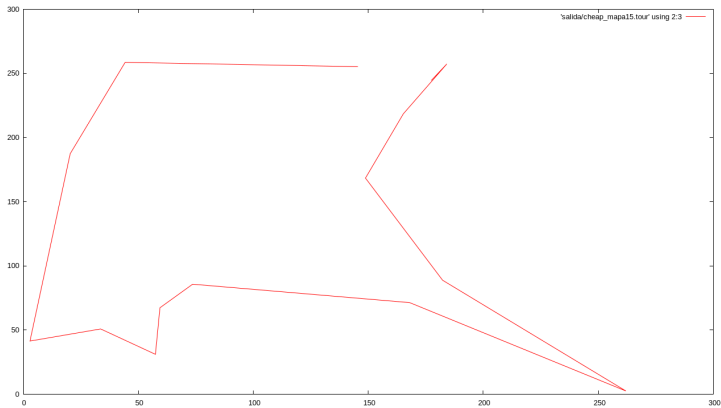


Figura: Algoritmo greedy

Conclusiones

