05/04/2015

Practica 2

Metaheurísticas basadas en una única solución

Miguel Morales Carmona

Contenido

Descripción de los algoritmos

Búsqueda local

Se trata de una búsqueda que parte de una solución actual y se pretende seleccionar una de su entorno para continuar. Utiliza la vecindad para realizar este proceso e implementa unas condiciones (a elegir por nosotros) para llegar un máximo global, ya que su peor inconveniente es que tiende a caer en óptimos locales en problemas grandes.

Hay dos variantes en las que podemos enfocar este tipo de búsqueda :

-First improvement : Evalúa el vecindario hasta encontrar uno que mejore la solución actual.

-Best improvement : Evalúa todo el vecindario de la solución actual. Algo bastante costoso.

Enfriamiento simulado

Se trata de un algoritmo de búsqueda por entorno que su probabilidad de parada se ve determinada por una aceptación basada en la termodinámica. Esto busca que no llegue a soluciones peores que las actuales y queden estancados en otros óptimos locales que no mejoren la solución que ya tengamos.

Si un movimiento se hace hacia uno peor es porque la aceptación intuye una posible mejora tras este paso.

Para esta fórmula de aceptación usamos T que es la temperatura, la probabilidad con la que aceptaremos una solución vecina peor que la actual. En cada ciclo esta variable va disminuyéndose, haciendo cada vez más difícil un movimiento peor. Por lo que se presupone que al final del algoritmo estaremos más cerca del óptimo global y lo que evita así es que se aleje del mismo.

Pseudocódigos de generación de vecinos

En el caso del MMDP, la generación de vecinos se realiza de la siguiente manera:

```
indice = 0;
para i = 0 hasta tamaño
        si solucion[i]==1 entonces
                para j = 0 hasta tamaño
                        si soluciones[j] == 0 entonces
                                indice++
                                Si indiceVecino==indice
                                        nuevo=intercambio(i,j)
                                        devuelve nuevo
                                finsi
                        finsi
                finpara
        finsi
finpara
```

En el caso de TSP, la generación de vecinos se realiza de igual manera pero con movimiento conocidos como 2-opt:

```
aux = sol->solucion[(aristas[i]+1)%tamaño]
sol->solucion[(aristas[i]+1)%tamaño]= sol->solucion[(aristas[i]+2+j)%tamaño];
sol->solucion[(aristas[i]+2+j)%tamaño] = aux;
```

Para generar el vector aristas o candidatos, debemos intentar mejorar la distancia, almacenando esta información en un vector, ordenándolo y eligiendo sólo los elementos que nos mejoran.

Pseudocódigos de métodos de busqueda

MMDP

BLb/BLf

Se ha utilizado la misma función para ambos, pues es aprovechable gran parte del código

```
mejorObjetivo <- evaluarSolucion(solActual);
 repetir
        mejora <- false;
        para i = 1 hasta (data.nd - data.md) * data.md and iter < (data.nd * 50)
                tmp <- generarVecindario(solActual, i);
                iter++;
                si evaluarSolucion(tmp) > mejorObjetivo
                        mejorVecino <- tmp;
                        mejorObjetivo <- evaluarSolucion(tmp);
                        mejora <- true;
                        Si es BLf
                                return mejorVecino
                finSi
        finPara
        solActual <- mejorVecino;
mientras iter < data.nd * 50 and mejora //Para parar el BLb
```

ES

En el código fueron incluídas mejoras extra, que se salen del algoritmo y algunas no sé a ciencia cierta si mejoran o no la solución o el tiempo de ejecución, pero el esqueleto principal de Enfriamiento Simulado en MMDP es el siguiente :

```
temperatura <- 10 * evaluarSolucion(sol) / -log(0.3);
mejorSolucion <- solActual <- sol;
repetir
         iterInterno <- iterMejora <- 0;
        nums <- aleatorioEnteroSinRepeticion(1, (data.nd - data.md) * data.md, (data.nd
- data.md) *    data.md );
        repetir
                vecino <- generarVecindario(solActual, nums[iterInterno]);
                 incrF <- evaluarSolucion(solActual) - evaluarSolucion(vecino);
                si incrF < 0 or (((float) rand()) / RAND_MAX) <= exp(-
                incrF/temperatura)
                        solActual <- vecino;
                         iterMejora
                        si evaluarSolucion(vecino) > evaluarSolucion(mejorSolucion)
                                mejorSolucion <- vecino;
                                mejora <- true;
                        finSi
                finSi
        iterInterno++;
        iter++;
        mientras (iter < data.nd * 50 && iterInterno < data.nd && iterMejora < data.nd *
0.1 && iterInterno < (data.nd - data.md) * data.md); //Condicion de parada interna
        temperatura <- temperatura*alpha;
mientras (mejora && iter < data.nd * iterRelaTamano && temperatura <= tFinal);
//Condicion de parada externa
devolver mejorSolucion;
```

TSP

BLb/BLf

Lo he realizado de manera muy parecida al de MMDP pero evitando que se coja él mismo, y los colindantes, condición indispensable cuándo hablamos de búsqueda en un grafo como en este problema. Esto vuelve un poco más complejo y largo el código. El pseudocódigo es el siguiente:

```
mientras continuar=true
        si es BLf entonces
                mientras condicion=true
                        si i es menor que 50 entonces
                                w=random%tamaño
                        finsi
                        si no está seleccionado el nodo w entonces
                                si w no es i, ni i+1, ni i-1 entonces condicion=false;
                                sino
                                        w=(w+3)%tamaño;
                                        condicion=false
                                finsi
                        si estaba seleccionado el nodo w entonces i++
                        Si i es mayor que 50 entonces
                                w=i%tamaño;
                                i++
                        finsi;
                finpara
        sino es BLf entonces
                w=w%tamaño
        finsi
        si w = tamaño entonces continuar=false y w=0 //evitar violación de segmento
        nuevo=generar(actual, i, w)
        si evaluar(nuevo) < mejor entonces
                solmejor=nuevo
                mejor=evaluar(nuevo)
                si es BLf return mejor
        finsi
        j++
        si j es mayor que 1000*tamaño entonces continuar=false
        i++
        w++
finpara
return mejor
```

ES

```
mientras continuar1=true
        actual=nuevo
        mientras continuar2=true
                x=0
                w=rand()%tamaño
                si w no es i, ni i-1 ni i+1 entonces continuar2=false
                sino w=(w+3)%tamaño
                siguiente=generar(original, i, w);
                nuevaSolucion=evaluar(siguiente,)
                actualSolucion=evaluarComparativa(nuevo,nuevaSolucion,i,w)
                decrimiento=nuevaSolucion-actualSolucion

                si el decrecimiento es menor que 0 o la función de temperatura se cumple
                        nuevo=siguiente
                si nuevaSolucion < actualSolucion entonces
                        nuevo = siguiente
                si x es mayor O j mayor que 0,1*tamaño que tamaño continuar=false
                i++; x++; j++;
                siguiente=nueva;
        finpara
        Temperatura=temperatura*alpha
        si i es mayor que 1000*tamaño, actual y nuevo son iguales o se ha superado la
temperatura maxima, entonces continuar1=false
devuelve nueva;
```

Análisis de resultados

MMDP

	Algoritmo BL b		Algoritmo BL f		Algoritmo ES	
Fichero	Desv	Tiempo	Desv	Tiempo	Desv	Tiempo
GKD-Ia_11_n10_m4.txt	0.00	0.014	0.00	0.012	0.00	0.013
GKD-Ia_12_n10_m4.txt	0.00	0.013	0.00	0.015	0.00	0.013
GKD-Ia_1_n10_m2.txt	-6.63	0.015	-6.63	0.012	0.00	0.015
GKD-Ia_21_n10_m8.txt	0.00	0.012	0.00	0.013	0.00	0.014
GKD-Ia_22_n10_m8.txt	0.00	0.015	0.00	0.012	0.00	0.014
GKD-Ia_2_n10_m2.txt	-3.04	0.012	-6.08	0.014	0.00	0.013
GKD-Ia_31_n15_m4.txt	-6.47	0.014	-3.94	0.013	-5.58	0.016
GKD-Ia_32_n15_m4.txt	0.00	0.014	-4.81	0.013	-3.48	0.014
GKD-Ia_41_n15_m9.txt	0.00	0.015	0.00	0.015	-0.53	0.013
GKD-Ia_42_n15_m9.txt	0.00	0.014	0.00	0.012	0.00	0.013
GKD-Ia_46_n15_m12.txt	0.00	0.014	0.00	0.012	0.00	0.014
GKD-Ia_47_n15_m12.txt	0.00	0.013	0.00	0.016	0.00	0.012
GKD-Ia_56_n30_m9.txt	-5.11	0.016	-4.24	0.012	-3.48	0.015
GKD-Ia_57_n30_m9.txt	-2.96	0.015	-1.29	0.016	-2.40	0.014
GKD-Ia_61_n30_m12.txt	-3.81	0.013	-1.31	0.013	-0.87	0.015

GKD-Ia_62_n30_m12.txt	-13.23	0.012	-6.32	0.014	-8.24	0.012
GKD-Ia_66_n30_m18.txt	-1.97	0.013	-2.93	0.014	-4.43	0.015
GKD-Ia_67_n30_m18.txt	-3.82	0.012	-3.35	0.014	-7.36	0.013
GKD-Ia_71_n30_m24.txt	-1.54	0.013	0.00	0.014	-3.85	0.013
GKD-Ia_72_n30_m24.txt	0.00	0.013	-0.51	0.013	-0.51	0.016
GKD-Ic_10_n500_m50.txt	-39.97	1.052	-16.96	9.182	-19.05	2.140
GKD-Ic_11_n500_m50.txt	-44.10	1.050	-11.64	9.354	-14.94	2.415
GKD-Ic_12_n500_m50.txt	-38.17	1.064	-7.41	9.224	-14.63	2.549
GKD-Ic_13_n500_m50.txt	-44.95	1.059	-17.26	8.907	-14.86	2.467
GKD-Ic_14_n500_m50.txt	-46.91	1.054	-12.16	9.272	-15.99	2.526
GKD-Ic_15_n500_m50.txt	-51.30	1.055	-12.04	9.276	-21.42	2.189
GKD-Ic_16_n500_m50.txt	-46.71	1.053	-11.17	9.296	-14.45	2.681
GKD-Ic_17_n500_m50.txt	-43.60	1.050	-8.83	9.161	-16.80	2.289
GKD-Ic_18_n500_m50.txt	-44.96	1.055	-16.97	9.312	-16.77	2.298
GKD-Ic_19_n500_m50.txt	-53.13	1.065	-17.55	8.786	-16.01	2.495
GKD-Ic_1_n500_m50.txt	-46.24	1.048	-8.55	8.920	-16.27	2.470
GKD-Ic_20_n500_m50.txt	-48.58	1.049	-11.48	9.447	-20.15	2.068

GKD-Ic_2_n500_m50.txt	-40.99	1.046	-18.19	8.756	-19.52	2.181
GKD-Ic_3_n500_m50.txt	-43.94	1.047	-13.71	9.328	-19.02	2.156
GKD-Ic_4_n500_m50.txt	-44.10	1.096	-9.94	9.240	-16.46	2.557
GKD-Ic_5_n500_m50.txt	-47.17	1.045	-14.40	9.178	-15.75	2.408
GKD-Ic_6_n500_m50.txt	-44.79	1.050	-14.08	8.843	-13.97	2.597
GKD-Ic_7_n500_m50.txt	-50.38	1.055	-19.97	8.742	-16.69	2.371
GKD-Ic_8_n500_m50.txt	-49.01	1.056	-10.57	9.305	-14.71	2.410
GKD-Ic_9_n500_m50.txt	-42.32	1.056	-16.36	9.084	-14.93	2.360

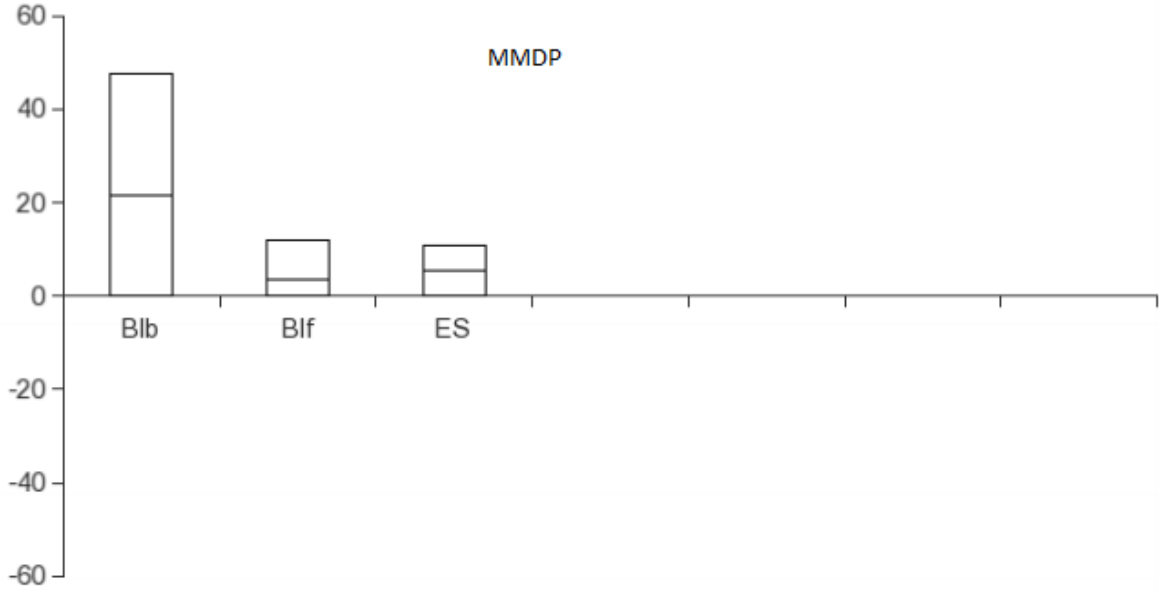
Media	-24.00	0.53	-7.77	4.57	-9.33	1.20
Desviación Típica	22.13	0.53	6.53	4.62	7.76	1.20
Máximo	0.00	1.10	0.00	9.45	0.00	2.68
Mínimo	-53.13	0.01	-19.97	0.01	-21.42	0.01

Si observamos las métricas, podemos observar que BLb y BLf obtienen muy buenos resultados para instancias pequeñas mientras que ES los obtiene mejores para las grandes.

Sin embargo, BLf no empeora en la misma cantidad que BLb, esto se debe a que la aleatoriedad en casos grandes funciona de mejor manera. Pero también podemos observar que el coste computacional de BLf es mucho mayor, la explicación podría ser por la alta generación de números aleatorios, que es mucho más costosa que explorar todo el entorno con BLb.

Los estadísticos finales pueden llegar a ser engañosos, así que si tuviera que quedarme con alguno de estos algoritmos sería con BLf, pues es más constante aunque más costos. En caso de que el coste computacional fuera un problema, escogería ES ya que en casos reales, las instancias tienden a ser grandes.

Los valores son negativos porque es un caso de maximización. El boxplot de este problema está en la página siguiente.



TSP

	Algoritmo BL b		Algoritmo BL f		Algoritmo ES	
Fichero	Desv	Tiempo	Desv	Tiempo	Desv	Tiempo
att48.txt	466.50	0.036	365.37	0.018	270.26	0.037
berlin52.txt	73.34	0.038	33.67	0.025	28.19	0.035
ch130.txt	459.71	0.165	94.72	0.188	39.83	0.243
ch150.txt	552.30	0.222	107.50	0.248	54.83	0.311
d198.txt	869.75	0.348	193.62	0.419	103.65	0.521
dantzig42.txt	42.56	0.031	30.72	0.016	16.64	0.027
eil101.txt	270.35	0.099	48.62	0.103	40.41	0.107
eil51.txt	62.99	0.036	36.92	0.017	15.75	0.035

eil76.txt	164.89	0.068	49.21	0.046	26.63	0.058
fri26.txt	13.77	0.015	23.62	0.016	7.33	0.015
gr17.txt	13.53	0.012	11.89	0.014	6.27	0.012
gr202.txt	387.67	0.369	128.45	0.403	19.73	0.512
gr96.txt	322.85	0.104	70.48	0.104	34.73	0.128
kroA100.txt	387.72	0.100	85.56	0.112	45.69	0.149
kroA200.txt	880.29	0.359	187.83	0.431	71.63	0.595
p01.txt	25.32	0.012	25.20	0.014	10.54	0.012
pr76.txt	197.02	0.065	44.58	0.059	26.24	0.086
st70.txt	175.94	0.056	49.78	0.039	30.03	0.065
TSP						
Media	298.14	0.12	88.21	0.13	47.13	0.16
Desviación Típica	271.43	0.12	87.30	0.15	60.71	0.19
Máximo	880.29	0.37	365.37	0.43	270.26	0.60
Mínimo	13.53	0.01	11.89	0.01	6.27	0.01

En este caso los valores son positivos porque es un caso de minimización.

Este problema requiere menos coste computacional pero los resultados son bastante peores. Un caso que no puedo encontrar explicación es que manualmente los resultados han sido muy diferentes a los que he obtenido al ejecutar el script, posiblemente la semilla influyese mucho en el resultado.

Vemos que ES obtiene muy buenos resultados en este caso y que BLf no requiere tanto tiempo como en MMDP aún generando aleatorios de igual manera, sin embargo, sigue obteniendo los resultados mejores y más constantes. Excepto algunos casos, que quizás pudiera haber caído en un óptimo local.

BLb y ES tienen valores muy similares pero BLb tiene un máximo mucho más grande, por lo que la elección entre ellas dos, sobre todo para instancias grandes, depende mucho del problema y los datos.