Material Teórica XV parte 1

Temario

- Heurísticas para el problema de coloreo de grafos
- Heurísticas para el problema de la mochila

Soluciones heurísticas para el problema de coloreo de grafos

¿Qué es el problema de coloreo de grafos?

- ***El problema de coloreo nace con la cartografía***
- ***Se quiere asignar un color a cada país***
- ***Para poder distinguir las divisiones políticas los países limítrofes deben tener colores distintos***
- ***Se busca usar la mínima cantidad de colores***

Definición formal del problema

- ***Dado un grafo no dirigido G = (V, E) con un conjunto V de vértices y E de aristas, un coloreo válido de G se corresponde a una partición de V en* k *conjuntos independientes, el objetivo es dado un G encontrar el menor* k *tal que G tenga un k-coloreo válido***
- ***Llamaremos a este* k *mínimo χ(G) y diremos que es su número cromático***
  - **Dado G, hallar χ(G) es NP-Hard**

Modelo matemático

Minimizar

$$\sum_{j=1}^{n} w_j$$

Sujeto a

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \in V$$

$$x_{ij} + x_{kj} \leq w_j \quad si\ [i,k] \in E \quad \forall j = 1, \dots, n$$

$$x_{ij}, w_j \in \{0,1\} \quad \forall i \in V \quad \forall j = 1, \dots, n$$

Resolución del problema de coloreo de grafos

Para resolver de manera exacta el problema de coloreo de grafos se resuelve primero el problema como si las variables sean continuas. Esto se hace para todos los problemas de programación lineal entera, se comienza resolviendo la relajación lineal (que es resolver el problema como si las variables fueran continuas).

El problema es que cuando se resuelve el problema de coloreo de grafos como si las variables fueran continuas SIEMPRE DA QUE CON DOS COLORES ALCANZA, no importa cuántos colores sean necesarios.

Esto es algo que se da particularmente con el problema de coloreo de grafos, con otros problemas (como el problema de la mochila), la relajación lineal da un resultado que sirve como cota superior para la solución (es un resultado no posible pero nos dice que el óptimo entero tendrá un funcional peor, por eso sirve como cota).

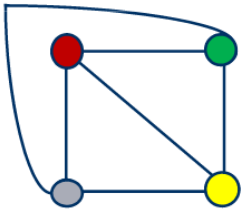
El hecho de que la relajación lineal dé que se necesitan dos colores se da por las siguientes restricciones:

$$x_{ij} + x_{kj} \leq w_j \quad si\ [i,k] \in E \quad \forall j = 1, \ldots, n$$

Cuando las variables pueden tomar valor continuo, aunque menores e iguales que uno (como es en la relajación lineal), Xij vale 0,5 y Xkj vale 0,5 con lo que con dos colores alcanza (pinta todos los nodos mitad con el color 1 y mitad con el color 2, y así cumple la restricción)

Resolución por heurísticas del problema de coloreo de grafos

Teorema de Appel-Hanke (1976): Un grafo plano requiere a lo sumo 4 colores para colorear sus nodos de forma que no haya vértices adyacentes del mismo color.



Si el grafo no es plano,
puede requerir tantos colores como vértices haya.

Heurísticas

Heurísticas golosas:

- ***Son rápidas***
- ***No suelen tener buenos resultados***
- ***Se usan para crear soluciones iniciales de metaheurísticas más complejas***
- ***En esta categoría se destacan***
  - **Largest saturation degree (DSATUR)**
  - **Recursive largest first (RLF)**

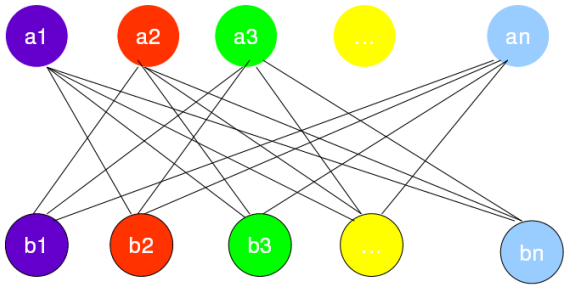
Un ejemplo de algoritmo Greedy

Algoritmo greedy heurístico: **O(V)**

```
función Coloreo ( Grafo G(V,A) )
{
   i = 1;
   while (grafo no coloreado) {
          Elegir un color c_i
          Colorear todos los vértices que se pueda con c_i
           a partir de un vértice arbitrario (esto es, todos
           los vértices que no sean adyacentes a un vértice
           ya coloreado con c_i)
          i = i + 1;
    }
}
```

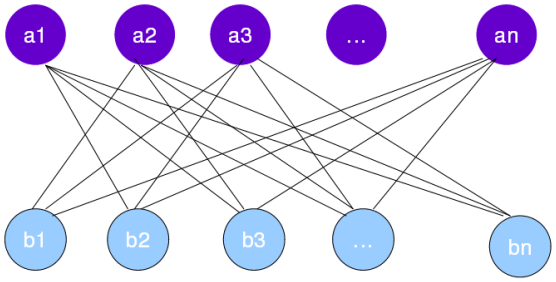
En las heurísticas Greedy, el orden en el cual se van recorriendo los nodos es decisivo. Veamos dos ejemplos para el mismo problema:

Secuencia de vértices <a1,b1,a2,b2,..an,bn>



n colores!!!

Secuencia de vértices <a1,a2,..an,b1,b2,..bn>



Coloreo óptimo:2 colores

Heurísticas de búsqueda local:

- ***Tabucol***
  - **Primera aplicación de Tabú a coloreo**
  - **Se fue mejorando con el tiempo, se usa como subcomponente de metaheurísticas más complejas**
- ***Y muchas otras:***
  - **Simulated Annealing, Iterated Local Search, Reactive Partial Tabu Search, GRASP, Variable Neighborhood Search, Variable Space Search, Clustering-Guided Tabu Search**

Algoritmo propuesto: MACOL

- ***Se busca un k-coloreo para un* k *dado, en caso de encontrar un coloreo válido usando k colores se puede buscar un nuevo k'-coloreo con* k' < k*, a medida que se achica* k *el problema se vuelve más difícil***

- ***Se trabaja sobre un algoritmo memético, estos son algoritmos de población donde se agrega aprendizaje por separado a algunos o todos los individuos.***
  - **En este caso se usa un algoritmo genético combinando Tabu Search.**

Puntos interesantes

- ***Propone cruza multi-padre***
- ***Toma una definición de distancia entre dos k-coloreos y la extiende a distancia entre un k-coloreo y una población***
- ***Propone una nueva función de evaluación que tiene en cuenta tanto la calidad de la solución como la diversidad de los individuos***

El objetivo es evitar la convergencia prematura que caracteriza a muchos algoritmos genéticos

Estructura general de MACOL

Algorithm 1. Pseudo-code of the Memetic Algorithm for k-Coloring

Input: Graph G, number of colors k, population size p
Output: The best k-coloring S^* found so far
$P = \{S_1, \ldots, S_p\} \leftarrow$ Initial_Population( ) /* Section 2.3 */
for $i = \{1, \ldots, p\}$ **do**
  $S_i \leftarrow$ Tabu_Search(S_i) /* Section 2.4 */
end for
$S^* = arg\,min\{f(S_i), i = 1, \ldots, p\}$
repeat
  Randomly choose m individuals $\{S_{i1}, \ldots, S_{im}\}$ from $P(2 \leqslant m \leqslant p)$
  $S_0 \leftarrow$ Adaptive_Multi-Parent_Crossover $(S_{i1}, \ldots, S_{im})$ /* Section 2.5 */
  $S_0 \leftarrow$ Tabu_Search(S_0) /* Section 2.4 */
  if $f(S_0) < f(S^*)$ **then**
    $S^* = S_0$
  end if
  $\{S_1, \ldots, S_p\} \leftarrow$ Pool_Updating$(S_0, S_1, \ldots, S_p)$ /* Section 2.6 */
until Stop condition met

MACOL: Initial Population

- ***Se usa una versión randomizada de la heurística DANGER***
  - **El próximo vértice a colorear se elige en base a su índice de "riesgo", este se desprende del grado del vértice y los colores asignados previamente.**
  - **El color a asignar se toma con el mismo criterio, se busca el color que es "menos probable" que sus vecinos requieran**
  - **La randomización consiste en tomar los valores de riesgo como una probabilidad de elección del color o vértice**
- ***Si el nuevo k-coloreo es muy parecido a los ya obtenidos se lo descarta y se busca uno nuevo***
- ***Los k-coloreos no son válidos (si alguno lo fuera el problema está resuelto)***

Se probó también con poblaciones generadas totalmente al azar

- **No se observó una diferencia real en el coloreo final obtenido**
- **La mejor calidad de las soluciones iniciales (comparadas con soluciones al azar) ayuda a que el costo computacional de las primeras generaciones sea menor**

MACOL: Tabu Search

- ***Se define la función de evaluación* f *como la suma de los vértices en conflicto, i.e. vértices adyacentes con el mismo color***
- ***Se obtiene un vecino de un k-coloreo dado cambiando el color de un vértice en conflicto***
  - **La cantidad de k-coloreos vecinos está acotada por *O(f(s) x k)***
  - **Para un vértice *u* con color original *i* y nuevo color *j* llamamos a este movimiento *(u,i,j)***
- ***Una vez que se efectúa el movimiento* (u,i,j) *se prohíbe al vértice* u *volver al color* i *por las siguientes* l *iteraciones***
- ***En este paper se utiliza* l = μ * f(S) + r(10)**
  - ***r(10)* es un número random de 1 a 10**
  - ***μ* se fija en 1**
- ***La condición de corte es la cantidad de iteraciones***

MACOL: Adaptive Multi-Parent Crossover

- ***AMPaX ("Adaptive Multi-Parent Crossover") es una extensión de GPX "Greedy Partition Crossover"***

- ***Dado un un k-coloreo* i *definimos las particiones* {Vi1, Vi2, ..., Vik} *donde un* Vij *contiene a los vértices coloreados con color* j *en el k-coloreo* i**

GPX

- **Input: S1 y S2 con particiones {V11, V12, ..., V1k} y {V21, V22, ..., V2k} respectivamente**
- **Output: S0 con particiones {V01, V02, ..., V0k}**
- **Pseudocódigo**
  - **Para j = 1 hasta k hacer**
    - ***Se toma la partición más grande de S((i mod 2) + 1) y se la asigna a S0 como V0j***
    - ***Se quitan los vértices contenidos en V0j de las particiones de S1 y S2***
  - **Fin Para**
  - **Agregar los vértices restantes a particiones de S0 al azar**

MACOL: Adaptive Multi-Parent Crossover

- ***Diferencias entre AMPaX y GPX***
  - **AMPaX no se limita a usar solo dos padres, en este paper se hacen pruebas con 2, 4 y r[2, ..., 6] (un número al azar entre 2 y 6), se elije esta última opción**
  - **AMPaX no define de antemano de que padre se tomará la partición sino que toma al mejor dándole al algoritmo una visión global**
  - **Para agregar diversidad luego de que un padre es elegido para pasar su partición se lo coloca en una lista tabú por *q* iteraciones**
    - **En este paper se toma *q = m / 2* donde *m* es la cantidad de padres**

MACOL: Adaptive Multi-Parent Crossover

Algorithm 2. Pseudo-code of the Adaptive Multi-Parent Crossover Operator

Input: m parent individuals $\{S_1, \ldots, S_m\}$ $(m \geqslant 2)$
Output: An offspring individual $S_0 = \{V_{01}, \ldots, V_{0k}\}$
Set forbidden length for each parent: $q(S_i) = 0,\ i = 1, \ldots, m$
for $i = 1, \ldots, k$ **do**
  Build the ith color class V_{0i} of S_0 as follows:
  Indicate the non-forbidden color classes: $\widehat{V} = \{V_{uv} | , q(S_u) = 0\}$
  Find out the maximal cardinality class: $V_{u^* v^*} = arg\,max\{|V_{uv}\|, V_{uv} \in \widehat{V}\}$
  Let V_{0i} be the set of vertices in S_{u^*} with color v^* : $V_{0i} = V_{u^* v^*}$
  Remove all the vertices in V_{0i} from m parent individuals: $V_{uv} = V_{uv} \setminus V_{0i},\ u = 1, \ldots, m, v = 1, \ldots, k$
  Update forbidden length: $q(S_u) = q(S_u) - 1\ (q(S_u) > 0)$ and $q(S_{u^*}) = \lfloor m/2 \rfloor$
end for
Randomly choose a color class for each unassigned vertex.

MACOL: Pool Updating

Definiciones

- ***La distancia entre dos k-coloreos Si, Sj es la cantidad mínima de movimientos (cambio de color a un vértice) para que Si sea igual a Sj***
  - **Calculan esta distancia de forma exacta resolviendo un problema de matching entre los vértices de los k-coloreos**
- ***Distancia entre un k-coloreo y una población***
  $D_{iP} = \min\{d_{ij} \mid S_j \in P, j \neq i\}$
- ***Score de un k-coloreo dentro de una población***
  $h_{iP} = f(S_i) + e^{\beta / DiP}$

Se define* $\beta = \lambda * n$ *y* $\lambda = 0{,}08$*,* n *es la cantidad de vértices

- ***El score propuesto busca equilibrar calidad con variedad***
- ***En las primeras soluciones tanto*** $f(S_i)$ ***como*** D_{iP} ***son muy grandes por lo tanto que el primer término (calidad) es predominante en*** h_{iP}
- ***A medida que se mejoran las soluciones la diversidad disminuye y entra en juego, nuevas soluciones buenas pueden descartarse si son parecidas a la población***

De esta forma se evita la convergencia prematura

MACOL: Pool Updating

Algorithm 3. Pseudo-code of the Pool Updating Rule.

Input: Population $P = \{S_1, \ldots, S_p\}$ and offspring k-coloring S_0
Output: Updated Population $P = \{S_1, \ldots, S_p\}$
Tentatively add S_0 to Population $P : P' = P \cup \{S_0\}$
for $i = 0, \ldots, p$ **do**
  Calculate the distance between S_i and P' $(D_{i,P'})$ according to Eq. (3)
  Calculate the goodness score of S_i $(h_{i,P'})$ according to Eq. (4)
end for
Identify the k-coloring with the largest value of the goodness score: $S_w = arg\,max\{h_{i,P'} | i = 0, \ldots, p\}$
if $S_w \neq S_0$ **then**
  Replace S_w with $S_0 : P = P \cup \{S_0\} \setminus \{S_w\}$
else
  if $rand(0, 1) < 0.2$ **then**
    Identify the second worst k-coloring: $S_{sw} = arg\,max\{h_{i,P'} | i = 0, \ldots, p, S_i \neq S_w\}$
    Replace the second worst k-coloring S_{sw} with $S_0 : P = P \cup \{S_0\} \setminus \{S_{sw}\}$
  end if
end if

Tests

Parametros

Parameters	Section	Description	Values
p	2.1	Size of population	20
α	2.4	Depth of TS	100,000
m	2.5	Number of parents for crossover	$r[2,\ldots,6]$
P_r	2.6	Probability for accepting worse offspring	0.2
λ	2.6	Parameter for goodness score function	0.08

Se utilizan los grafos DIMACS

Computational results of MACOL algorithm on the *difficult* DIMACS challenge benchmarks within 5 CPU hours, except for graphs *C2000.5* and *C4000.5*.

Instances	n	n_e	*dens*	k^*	References	MACOL			
						k	#hit	*iter*	*time*(*m*)
DSJC250.5	250	15,668	0.50	28	[14,16,24,29,30,34]	28	20/20	2.6×10^6	< 1
DSJC500.1	500	12,458	0.10	12	[2,16,24,29,30,34]	12	20/20	5.3×10^7	< 1
DSJC500.5	500	62,624	0.50	48	[2,14,16,24,29,34]	48	20/20	2.1×10^6	22
DSJC500.9	500	112,437	0.90	126	[2,16,24,30,34]	126	20/20	1.9×10^8	95
DSJC1000.1	1000	49,629	0.10	20	[2,14,16,24,29,34]	20	16/20	3.5×10^7	108
DSJC1000.5	1000	249,826	0.50	83	[14,29,34]	83	20/20	2.2×10^8	47
DSJC1000.9	1000	449,449	0.90	224[a]	[14,16,24,34]	**223**	18/20	4.5×10^8	150
						224	20/20	7.9×10^7	45
DSJR500.1c	500	121,275	0.97	85	[12,24,29,30]	85	20/20	4.5×10^6	5
DSJR500.5	500	58,862	0.47	122	[29,33]	122	11/20	2.5×10^7	115
R250.5	250	14,849	0.48	65	[2,29,30]	65	20/20	2.7×10^5	4
R1000.1c	1000	485,090	0.97	98	[2,29,30,34]	98	20/20	7.5×10^5	8
R1000.5	1000	238,267	0.48	**234**	[29,33]	245	13/20	1.2×10^9	276
						246	16/20	5.8×10^8	152
						247	20/20	2.3×10^8	76
le450_15c	450	16,680	0.17	15	[2,12,16,24,29,30]	15	20/20	2.0×10^6	3
le450_15d	450	16,750	0.17	15	[2,12,16,24,29,30]	15	20/20	1.8×10^6	5
le450_25c	450	17,343	0.17	25	[2,29,30,34]	25	20/20	1.3×10^7	15
le450_25d	450	17,425	0.17	25	[2,29,30,34]	25	20/20	2.1×10^7	10
flat300_26_0	300	21,633	0.48	26	[12,29,30]	26	20/20	2.6×10^6	4
flat300_28_0	300	21,695	0.48	**28**	[2,24]	29	15/20	1.7×10^7	128
						30	20/20	4.6×10^6	13
flat1000_50_0	1000	245,000	0.49	50	[2,16,24,29,30]	50	20/20	3.2×10^5	5
flat1000_60_0	1000	245,830	0.49	60	[2,16,24,29,30]	60	20/20	6.3×10^5	9
flat1000_76_0	1000	246,708	0.49	82	[29,34]	82	20/20	7.2×10^7	68
						83	20/20	2.7×10^7	27
C2000.5	2000	999,836	0.50	151	[34]	**148**	1/5	8.2×10^8	2156
						149	3/5	6.6×10^8	1875
						150	5/5	3.9×10^8	1385
						151	5/5	3.2×10^8	867
C4000.5	4000	4,000,268	0.50	280	[12]	**272**	3/5	1.2×10^9	7165
						273	4/5	9.3×10^8	5274
						274	4/5	6.5×10^8	3786
						275	5/5	4.8×10^8	2943
						280	5/5	2.3×10^8	1546
latin_sqr_10	900	307,350	0.76	**98**	[30]	99	5/20	6.7×10^7	158
						100	17/20	1.3×10^7	35

[a] Very recently a 223-coloring was independently reported in [35] for graph *DSJC1000.9*.

Comparison with the state-of-the-art algorithms in terms of the best results obtained[a].

Instances	k^*	k_{best}	Local Search Algorithms				Hybrid Algorithms						
			[24]	[8]	[6]	[2]	[12]	[30]	[14]	[16]	[29]	[13]	[34]
DSJC250.5	28	**28**	–	**28**	**28**	–	29	**28**	**28**	**28**	**28**	**28**	**28**
DSJC500.1	12	**12**	**12**	13	**12**	**12**	–	–	–	**12**	**12**	**12**	**12**
DSJC500.5	48	**48**	**48**	50	49	**48**	49	49	**48**	**48**	**48**	49	**48**
DSJC500.9	126	**126**	**126**	127	**126**	127	–	–	–	**126**	127	127	**126**
DSJC1000.1	20	**20**	**20**	21	–	**20**	–	–	**20**	**20**	**20**	21	**20**
DSJC1000.5	83	**83**	86	90	89	89	84	89	**83**	84	**83**	88	**83**
DSJC1000.9	224	**223**	224	226	–	226	–	–	224	224	224	228	224
DSJR500.1c	85	**85**	**85**	–	–	**85**	**85**	**85**	–	86	**85**	**85**	–
DSJR500.5	122	**122**	125	–	124	125	130	123	–	127	**122**	**122**	124
R250.5	65	**65**	–	66	–	66	69	**65**	–	–	**65**	**65**	–
R1000.1c	98	**98**	–	**98**	–	**98**	99	**98**	–	–	**98**	**98**	**98**
R1000.5	234	245	–	242	–	248	268	241	–	–	**234**	237	245
le450_15c	15	**15**	**15**	–	**15**	**15**	16	**15**	**15**	**15**	**15**	**15**	–
le450_15d	15	**15**	**15**	–	**15**	**15**	16	**15**	–	**15**	**15**	**15**	–
le450_25c	25	**25**	**25**	–	26	**25**	–	–	26	26	**25**	26	**25**
le450_25d	25	**25**	**25**	–	26	**25**	–	–	–	26	**25**	26	**25**
flat300_26_0	26	**26**	–	**26**	**26**	–	**26**	**26**	–	**26**	**26**	**26**	–
flat300_28_0	28	29	**28**	31	31	**28**	33	31	31	31	31	31	31
flat300_50_0	50	**50**	**50**	**50**	–	**50**	84	**50**	–	**50**	**50**	**50**	–
flat300_60_0	60	**60**	**60**	**60**	–	**60**	84	**60**	–	**60**	**60**	**60**	–
flat300_76_0	82	**82**	85	89	–	87	84	89	83	84	**82**	87	**82**
C2000.5	153	**148**	–	–	–	–	153	165	–	–	–	162	151
C4000.5	280	**272**	–	–	–	–	280	–	–	–	–	301	–
latin_sqr_10	98	99	–	–	99	–	106	**98**	–	104	101	99	–
Better	3	–	4	9	6	7	16	6	4	9	4	11	4
Equal	18	–	11	5	7	11	2	9	5	10	17	11	11
Worse	3	–	1	1	0	1	0	2	0	0	1	2	0
Total	24	24	16	15	13	19	18	17	9	19	22	24	15

[a] Notice that competitive results were also reported with a refined tabu search algorithm described in a new paper recently accepted for publication [35].

MACOL

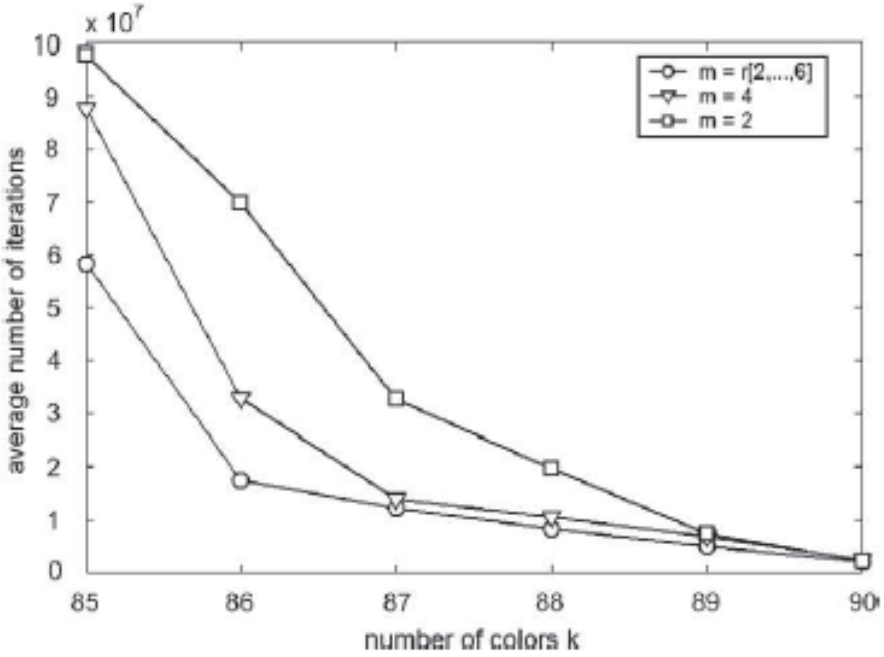


Fig. 2. Comparison between multi-parent crossover and 2-parent crossover.

Computational results of MACOL algorithm on the *easy* DIMACS benchmarks.

Instances	n	n_e	*dens*	k^*	MACOL			
					k	#*hit*	*iter*	*time*(*m*)
DSJC125.1	125	736	0.09	5	5	10/10	1.4×10^5	1
DSJC125.5	125	3891	0.50	17	17	10/10	4.8×10^4	3
DSJC125.9	125	6961	0.89	44	44	10/10	2.4×10^6	4
DSJC250.1	250	3218	0.10	8	8	10/10	6.9×10^5	2
DSJC250.9	250	27,897	0.90	72	72	10/10	5.5×10^6	3
R125.1	125	209	0.03	5	5	10/10	3.7×10^5	2
R125.1c	125	7501	0.97	46	46	10/10	2.8×10^6	5
R125.5	125	3838	0.50	36	36	10/10	3.2×10^4	1
R250.1	250	867	0.03	8	8	10/10	1.5×10^6	5
R250.1c	250	30,227	0.97	64	64	10/10	2.8×10^6	4
DSJR500.1	500	3555	0.03	12	12	10/10	3.3×10^5	4
R1000.1	1000	14,348	0.03	20	20	10/10	2.9×10^5	2
le450_15a	450	8168	0.08	15	15	10/10	2.7×10^5	2
le450_15b	450	8169	0.08	15	15	10/10	3.5×10^5	2
le450_25a	450	8260	0.08	25	25	10/10	1.8×10^5	4
le450_25b	450	8263	0.08	25	25	10/10	2.8×10^6	3
school1	385	19,095	0.26	14	14	10/10	8.8×10^5	6
school1_nsh	352	14,612	0.24	14	14	10/10	7.3×10^5	4
flat300_20_0	300	21,375	0.48	20	20	10/10	1.7×10^6	1

Tests: Pool Updating

DisQual: función propuesta
PoolWorst: sacar al peor de la población
ParentWorst: sacar al peor padre

Conclusiones

El éxito de MACOL frente a otras heurísticas de coloreo de grafos resalta el papel fundamental de los mecanismos de generación de diversidad para evitar la convergencia prematura y permitir explorar el espacio de soluciones.

Al momento de desarrollar una heurística para un problema determinado se debe analizar en detalle el balance entre intensificación y diversificación ya que de este puede depender el éxito o fracaso del proyecto.