Demostracion de NP-completitud de k-COLOR

De 3-SAT a 3-COLOR y de 3-COLOR a k-COLOR (k >= 3)

Fecha: 14/01/2026

Resumen

Se demuestra que el problema de decision k-COLOR (dado un grafo G, decidir si admite una coloracion propia con a lo sumo k colores) es NP-completo para todo entero fijo k >= 3. La prueba tiene dos partes: (i) k-COLOR pertenece a NP porque una coloracion propuesta se verifica en tiempo O(|E|); (ii) k-COLOR es NP-duro. Para (ii) se muestra primero que 3-COLOR es NP-completo construyendo una reduccion polinomial desde 3-SAT mediante gadgets (paleta de colores, gadget de variable y un gadget de clausula basado en un OR-gadget). Luego se reduce 3-COLOR a k-COLOR para k >= 3 agregando un clique de tamaño k-3 conectado a todos los vertices del grafo original, forzando que dicho grafo use solo 3 colores.

1. Definiciones basicas

1.1 Problemas de decision, lenguajes y complejidad

En complejidad computacional se modela un problema de decision como un lenguaje L, es decir, un conjunto de cadenas sobre un alfabeto finito. Una instancia x pertenece a L si y solo si la respuesta correcta es SI. Se dice que un problema es decidible si existe un algoritmo que termina en todas las entradas y responde correctamente.

1.2 La clase NP

Definicion (NP). Un lenguaje L pertenece a NP si existe un verificador V(x,y) que corre en tiempo polinomial y una funcion polinomial p(.) tal que para toda entrada x:

x pertenece a L si y solo si existe un certificado y con |y| <= p(|x|) y V(x,y) acepta.

Interpretacion: las soluciones candidatas se pueden verificar rapidamente (en

tiempo polinomial), aunque encontrarlas pueda ser dificil.

1.3 Reducciones polinomiales

Definicion (reduccion many-one en tiempo polinomial). Para lenguajes A y B, decimos que A se reduce a B en tiempo polinomial (A <=_p B) si existe una funcion f computable en tiempo polinomial tal que, para toda x, x pertenece a A si y solo si f(x) pertenece a B. Si A <=_p B y B esta en P, entonces A tambien esta en P.

1.4 NP-duro y NP-completo

Definicion (NP-duro). Un lenguaje H es NP-duro si para todo L en NP se cumple L <=_p H. Definicion (NP-completo). Un lenguaje C es NP-completo si (i) C pertenece a NP y (ii) C es NP-duro. En la practica, para probar NP-dureza basta reducir un problema ya conocido NP-completo al problema objetivo.

2. El problema k-COLOR y pertenencia a NP

2.1 Definicion de k-COLOR

Entrada: un grafo no dirigido G=(V,E) y un entero k>=1.
Pregunta: existe una funcion c:V->{1,...,k} tal que para toda arista (u,v) en E se cumple c(u) != c(v)?
Una tal funcion c se llama k-coloracion propia, y decimos que G es k-coloreable.

2.2 k-COLOR pertenece a NP

Teorema. Para todo k, k-COLOR pertenece a NP.

Demostracion. El certificado es una asignacion de colores c:V->{1,...,k}. El verificador recorre todas las aristas (u,v) en E y comprueba que c(u) != c(v). Esto requiere O(|E|) comparaciones, por tanto es tiempo polinomial en el tamaño de la entrada. Luego k-COLOR esta en NP. QED.

3. NP-dureza de 3-COLOR mediante 3-SAT

3.1 El problema 3-SAT

En 3-SAT se recibe una formula booleana en CNF donde cada clausula tiene exactamente tres literales. Ejemplo: (x OR not y OR z) AND (not x OR y OR w). La pregunta es si existe una asignacion de Verdadero/Falso a las variables que haga verdadera toda la formula. Es un resultado clasico que 3-SAT es NP-completo (a partir de SAT y el teorema de Cook-Levin).

3.2 Objetivo de la reduccion

Construiremos, para cada formula 3-CNF phi, un grafo G_phi tal que:

- si phi es satisfacible, entonces G_phi es 3-coloreable;
- si G_phi es 3-coloreable, entonces phi es satisfacible.

La construccion es polinomial en el tamaño de phi, por lo que esto define una reduccion 3-SAT <=_p 3-COLOR.

3.3 Paleta de colores (gadget base)

Creamos tres vertices especiales etiquetados T, F y B y los conectamos formando un triangulo completo. En toda 3-coloracion propia, estos tres vertices deben recibir colores distintos. Como los nombres de los colores son arbitrarios, renombramos los tres colores para que col(T)=T, col(F)=F y col(B)=B. A partir de aqui, hablamos de 'el color T' para referirnos al color asignado al vertice T, etc.

3.4 Gadget de variable

Para cada variable xi agregamos dos vertices, uno etiquetado xi y otro etiquetado not xi. Agregamos aristas (xi, not xi), (xi, B) y (not xi, B). Como ambos son adyacentes a B, ninguno puede usar el color B; por tanto cada uno usa T o F. Y como xi y not xi son adyacentes entre si, necesariamente toman colores opuestos. Interpretamos: xi es Verdadero si col(xi)=T, y xi es Falso si col(xi)=F. Esta regla define una asignacion booleana consistente para cada 3-coloracion del grafo.

3.5 OR-gadget y gadget de clausula

Para cada clausula C=(a OR b OR c), donde a,b,c son literales (vertices ya presentes), agregamos un gadget que implementa la operacion OR. La idea es construir un subgrafo con un vertice de salida s (representando a OR b OR c) que cumpla:

- Si a,b,c tienen color F, entonces s queda forzado a color F.
- Si al menos uno de a,b,c tiene color T, entonces el gadget puede colorearse de modo que s tenga color T.

Luego conectamos el vertice de salida s a los vertices F y B (agregamos aristas (s,F) y (s,B)). Con eso, en cualquier 3-coloracion, s no puede ser F ni B, asi que necesariamente col(s)=T. Por las propiedades anteriores, esto implica que en cada clausula al menos un literal debe ser T.

Definicion del OR-gadget. Usaremos el siguiente subgrafo con entradas a,b,c y salida s. Se agregan seis vertices internos: p, q, x, u, v y s (donde s es la salida). Las aristas son:

(a,p), (b,q), (p,q), (p,x), (q,x), (x,u), (u,v), (c,v), (u,s), (v,s).

Este OR-gadget aparece en apuntes de cursos y textos de NP-completitud; una presentacion concreta con la misma estructura y propiedades puede encontrarse, por ejemplo, en notas de la Universidad de Illinois (CS374) y materiales introductorios similares.

3.6 Propiedades del OR-gadget

Lema 1 (comportamiento cuando a=b=c=F). Si col(a)=col(b)=col(c)=F y a,b,c no pueden usar el color B, entonces en cualquier 3-coloracion del OR-gadget se tiene col(s)=F.

Demostracion. Supongamos col(a)=col(b)=col(c)=F. Como p es adyacente a a, se cumple col(p) != F; por tanto col(p) pertenece a {T,B}. Analogamente col(q) pertenece a {T,B}. Como p y q son adyacentes, deben usar colores distintos, asi que {col(p), col(q)} = {T,B}. El vertice x es adyacente a p y a q, por lo que no puede usar ni T ni B; el unico color restante es F. Luego col(x)=F. El vertice u es adyacente a x, asi que col(u) != F y por tanto col(u) pertenece a {T,B}. El vertice v es adyacente a u y a c (que es F), asi que col(v) != col(u) y col(v) != F; por tanto col(v) es el unico color en {T,B} distinto de col(u). En particular, {col(u), col(v)} = {T,B}. Finalmente, s es adyacente a u y a v, por lo que no puede usar ni T ni B; el unico color posible es F. Concluimos col(s)=F. QED.

Lema 2 (comportamiento cuando alguno es T). Si al menos uno de a,b,c tiene color T (y los literales estan restringidos a {T,F}), entonces existe una 3-coloracion del OR-gadget en la que col(s)=T.

Demostracion (constructiva, por casos). Consideramos dos casos principales.

- Caso A: c = T. Elegimos colores para u y v de modo que s pueda ser T. Si col(u) y col(v) son F y B (en algun orden), entonces s puede tomar T. Como v es adyacente a c=T, v no puede ser T. Si asignamos v=F, entonces u puede ser B (o viceversa) de forma que u != v y las restricciones se satisfacen. Los vertices p,q,x se colorean para representar (a OR b): si alguno de a,b es T, se puede elegir x=T; si ambos son F, entonces x=F. En ambos subcasos, siempre podemos escoger u != x. Esto completa una coloracion con s=T.

- Caso B: c = F y (a OR b) = T. Primero coloreamos el subgrafo izquierdo para obtener x=T: si a=T, ponemos p=F (para evitar T) y q=B; entonces x=T porque es adyacente a F y B. Si b=T se hace simetricamente. Con x=T y c=F, elegimos u=F (u es adyacente a x, asi que puede ser F) y v=B (v es adyacente a u=F y a c=F, asi que no puede ser F; puede ser B). Entonces s, adyacente a u=F y v=B, puede ser T. QED.

Observacion: si c=F y (a OR b)=F, entonces a=b=c=F, lo cual cae en el Lema 1. Por tanto, los dos casos cubren exactamente el escenario en el que al menos uno de a,b,c es T.

3.7 Correctitud global de la reduccion 3-SAT <=_p 3-COLOR

Teorema. Para toda formula 3-CNF phi, phi es satisfacible si y solo si el grafo G_phi construido es 3-coloreable.

Demostracion.

- (=>) Suponga que phi es satisfacible y sea A una asignacion que satisface phi. Coloreamos el triangulo base con colores distintos y fijamos esos nombres como T,F,B. Para cada variable xi: si A(xi)=Verdadero ponemos col(xi)=T y col(not xi)=F; si A(xi)=Falso ponemos col(xi)=F y col(not xi)=T. Esto es valido porque el gadget de variable fuerza colores opuestos y evita el color B. Ahora tome una clausula C=(a OR b OR c). Como A satisface phi, al menos uno de sus literales es Verdadero, luego el vertice correspondiente tiene color T. Por el Lema 2, el OR-gadget de esa clausula puede colorearse con salida s=T. Con esto, toda la construccion queda 3-coloreada.
- (<=) Suponga ahora que G_phi es 3-coloreable. Definimos una asignacion booleana A por: A(xi)=Verdadero si col(xi)=T, y A(xi)=Falso si col(xi)=F. Esto esta bien definido porque en cada gadget de variable, xi y not xi estan conectados entre si y a B, por lo que toman colores opuestos en {T,F}. Considere una clausula C=(a OR b OR c) con su OR-gadget y salida s. Como s es adyacente a B y a F, s no puede ser B ni F; por tanto col(s)=T. Por el Lema 1, es imposible que col(a)=col(b)=col(c)=F, porque en ese caso se forzaria col(s)=F. Luego, al menos uno de a,b,c tiene color T, lo que implica que el literal correspondiente es Verdadero bajo A. Asi, cada clausula es satisfecha y A satisface phi. QED.

Como la construccion agrega un numero de vertices y aristas proporcional al numero de variables y clausulas (tamano O(n+m)), la reduccion es polinomial. Dado que 3-SAT es NP-completo y 3-COLOR esta en NP, concluimos que 3-COLOR es NP-completo.

4. De 3-COLOR a k-COLOR (k >= 3)

Ahora probamos que para todo k >= 3 fijo, k-COLOR es NP-duro incluso si ya sabemos que 3-COLOR lo es.

4.1 Construccion

Sea G=(V,E) una instancia de 3-COLOR. Construimos un grafo G' asi:

- Agregamos un conjunto C de (k-3) nuevos vertices.
- Conectamos todos los pares de vertices en C (C forma un clique de tamano k-3).
- Conectamos cada vertice de C con cada vertice de V (todos los vertices nuevos son adyacentes a todo G).

Esta construccion es claramente polinomial: se agregan k-3 vertices y O((k-3)^2 + (k-3)|V|) aristas.

4.2 Correctitud de la reduccion

Lema. G es 3-coloreable si y solo si G' es k-coloreable.

Demostracion.

- (=>) Si G es 3-coloreable, sea c una 3-coloracion de G con colores {1,2,3}. Coloreamos los vertices de C con colores nuevos 4,5,...,k (todos distintos). Como C es un clique, necesita colores distintos; y como cada vertice de C es adyacente a todo V, los colores 4..k no pueden aparecer en V, pero V ya usa solo {1,2,3}. Por tanto obtenemos una k-coloracion de G'.
- (<=) Si G' es k-coloreable, observe que el clique C tiene tamano k-3, asi que sus vertices deben recibir k-3 colores distintos. Como cada vertice de C es adyacente a todo V, ningun vertice de V puede usar un color usado en C. Por tanto, los vertices de V pueden usar a lo sumo k-(k-3)=3 colores. Esto induce una 3-coloracion de G (quizas usando 1,2,3 como nombres de esos colores).

Concluimos que 3-COLOR <=_p k-COLOR para todo k>=3.

5. Conclusiones: NP-completitud de k-COLOR

Ya tenemos:

- k-COLOR pertenece a NP (Seccion 2.2).
- 3-SAT <=_p 3-COLOR (Seccion 3), luego 3-COLOR es NP-completo.

- 3-COLOR <=_p k-COLOR para todo k>=3 (Seccion 4).

Por transitividad de las reducciones, 3-SAT <=_p k-COLOR. Entonces k-COLOR es NP-duro. Como ademas k-COLOR pertenece a NP, se concluye:

Teorema final. Para todo entero fijo k >= 3, k-COLOR es NP-completo. QED.

Referencias (lectura recomendada)

- S. Cook (1971). The complexity of theorem-proving procedures. Proceedings of the Third Annual ACM Symposium on Theory of Computing (STOC).
- R. Karp (1972). Reducibility among combinatorial problems. In Complexity of Computer Computations (Miller & Thatcher, eds.).
- M. Garey y D. Johnson (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman.
- C. Papadimitriou (1994). Computational Complexity. Addison-Wesley.
- M. Sipser (varias ediciones). Introduction to the Theory of Computation. Cengage.
- Apuntes de CS374 (University of Illinois): reduccion 3-SAT -> 3-COLOR mediante OR-gadgets (consultado como referencia pedagogica).