Modelling the 3-Coloring of Serial-Parallel Graphs Via Incremental Satisfiability

C. López-Ramírez, and G. De Ita

***Abstract*—A novel algorithm is presented for the 3-coloring on parallel-serial graphs. Our proposal is based on the logical specifications (using conjunctive normal forms) of the constraints for a valid 3-coloring of a serial-parallel graph, and afterward, to apply incremental satisfiability in order to build efficiently, a valid 3-coloring. Our proposal builds an initial satisfiable conjunctive formula φ, where SAT(φ) requests an exponential time. However, when new clauses are added to φ, then the colors of the vertices of each serial-parallel component of the input graph are determined in automatic and incremental way. Our procedure is deterministic and it finds a valid 3-coloring. It has a linear-time complexity on the size of the graph. We also show the correctness of our algorithm.**

***Index Terms*—SAT Problem, Incremental Satisfiability, Serial-Parallel Graphs, 3-Coloring.**

I. Introducción

Uno de los problemas fundamentales en el razonamiento automático es el problema de Satisfactibilidad (SAT), que revisa si una fórmula lógica F es (o no) satisfactible. SAT es también una tarea relevante en otros problemas como: estimar el grado de creencia, revisar o actualizar creencias, en la explicación abductiva, en el diagnóstico lógico y en otros procesos propios de la Inteligencia Artificial (IA) [1].

SAT es un problema teórico importante, y fue el primer problema reconocido en la clase de complejidad NP-Completo. SAT continúa siendo un problema fundamental para aclarar la frontera entre problemas de las clases de complejidad P y NP-Completo. En particular, el caso 2-SAT, que determina la satisfactibilidad de una dos Forma Normal Conjuntiva (2-FNC), es un caso importante del problema SAT por resolverse en tiempo polinomial.

A pesar de la dificultad teórica del problema SAT, los procedimientos actuales de decisión, conocidos como solucionadores SAT, han sido sorprendentemente eficientes.

Los solucionadores SAT se han usado en diversas aplicaciones industriales. Tales aplicaciones rara vez se limitan a resolver sólo un problema de decisión, ya que en general, una sola aplicación requiere que se resuelva una secuencia de problemas relacionados, por lo que manejan una secuencia de problemas relacionados como una instancia del problema de satisfactibilidad incremental (ISAT).

Cristina López-Ramírez - estudiante del doctorado LKE de la Facultad de Cs. de la Computación - BUAP. cristyna2001@hotmail.com.

Guillermo De Ita Luna - profesor de la Facultad de Cs. de la Computación, Universidad Autónoma de Puebla. deitaluna63@gmail.com.

Las mejoras en el rendimiento de los algoritmos para ISAT han resultado ser una característica crucial para los solucionadores SAT [2].

El problema ISAT considera como entrada una secuencia de cláusulas: F_0, F_1,…, F_n, empezando con una fórmula inicial satisfactible F_0. Cada F_i resulta de un cambio en la fórmula anterior F_{i-1} impuesto por el 'mundo exterior'.

Aunque el cambio puede ser una restricción (añadir cláusulas) o una relajación (eliminar cláusulas), nos centraremos en el caso de restricción, al considerar solo adiciones de nuevas cláusulas. El proceso de agregar cláusulas se termina cuando se llega a la insatisfactibilidad, o no hay más cláusulas que añadir.

Una estrategia que se aplica al diseñar algoritmos para ISAT es preservar las estructuras computacionales formadas cuando se procesaron fórmulas anteriores, lo que permite por ejemplo, reconocer subfórmulas comunes que van apareciendo en las fórmulas de entrada, lo que permite la reutilización de la misma información a través de la secuencia de fórmulas.

Diferentes métodos se han aplicado para resolver ISAT, entre ellos, variaciones del procedimiento de ramificación y límites, denotado como Métodos IDPL, que generalmente se basan en el método clásico de Davis-Putnam-Loveland (DPL)[3]. En un procedimiento IDPL, al agregar nuevas cláusulas, se trata de mantener el árbol de búsqueda generado previamente. Se ha mostrado que IDPL se ejecuta más rápido que DPL para un gran conjunto de problemas SAT.

Whittemore et al. [4] definió ISAT como la resolución de cada subfórmula dentro de una secuencia finita de fórmulas. Wieringa [2] presentó una variedad de métodos secuenciales y paralelos para resolver ISAT, así como algunas de sus aplicaciones. Nadel [5] presenta una variación a ISAT bajo la hipótesis de cláusulas que fueron modificadas por variables de decisión; y de todas las cláusulas inferidas que dependen de algunos de los supuestos que incluyen su negación.

ISAT es de interés a una gran variedad de aplicaciones que necesitan ser procesadas en un entorno evolutivo. Este es el caso de aplicaciones como: planificación y programación reactiva, optimización combinatoria dinámica, revisión de fallas en circuitos combinatorios, satisfacción de restricciones dinámicas y aprendizaje en entornos dinámicos [6].

Por otro lado, el coloreo de grafos es quizás, uno de los problemas combinatorios más populares en la teoría de grafos [7]. En particular, la 3-colorabilidad sobre grafos planos es un conocido problema NP-completo. Una línea de investigación al considerar instancias de grafos planos es hallar cotas superiores en el número de colores a emplear.

Demostramos aquí que nuestra propuesta algorítmica, basada en especificar con formas normales conjuntivas las restricciones de un 3-coloreo sobre grafos serial paralelo, y emplear técnicas de satisfactibilidad incremental, efectivamente acota superiormente a 3 el número de colores para hallar un coloreo válido de un grafo plano serial paralelo.

Además de que utilizar satisfactibilidad incremental al modelar el coloreo de grafos serial-paralelo, proporciona un procedimiento efectivo, de complejidad lineal en tiempo, para hallar un 3-coloreo sobre este tipo de grafos.

II. Preliminares

Sea $X = \{x_1, \ldots, x_n\}$ un conjunto de n variables Booleanas, es decir que pueden tomar únicamente los valores de verdad $\{0,1\}$. Una literal es una variable x_i (x_i^1) o la variable negada $\neg x_i$ (x_i^0).

Una cláusula es una disyunción de literales distintas. Una k-cláusula es una cláusula con k literales. Una fórmula Booleana en forma normal conjuntiva (FNC) es una conjunción de cláusulas no tautológicas. F es una FNC monótona positiva, si todas las variables en F aparecen en forma no negada. Una k-FNC es una FNC conteniendo solamente k-cláusulas. ($\leq k$)-FNC denota una FNC conteniendo cláusulas con a lo más k literales.

Una variable $x \in X$ aparece en una fórmula F si x o $\neg x$ es un elemento de F. Usamos $v(X)$ para representar las variables involucradas en el objeto X, donde X puede ser una literal, una cláusula, o una FNC. Por ejemplo, para la cláusula $c = \{x_1 \vee \neg x_2\}$, $v(c) = \{x_1, x_2\}$. Lit(F) es un conjunto de literales involucradas en F, es decir si $X = v(F)$, entonces $\text{Lit}(F) = X \cup \neg X = \{x_1, \neg x_1, \ldots, x_n\}$.

Una asignación s para F es una función Booleana $s: v(F) \rightarrow \{0,1\}$. Una asignación puede también considerarse como un conjunto de literales no complementarias. Si $l \in s$, entonces $\neg l \notin s$, en otras palabras, s asigna verdadero a l y falso a $\neg l$, o viceversa. Sea c una cláusula y s una asignación, c es satisfactible por s si y solo si $(c \cap s) \neq \emptyset$. De otra forma, si para todo $l \in c$, $\neg l \in s$, entonces s falsifica a c.

Sea F una FNC, F es satisfactible por una asignación s si cada cláusula en F es satisfactible por s. F se contradice por s si cualquier cláusula en F es falsificada por s. Un modelo de F es una asignación para $v(F)$ que satisface a F, denotado como: $s \vdash F$. Un contramodelo de F es una asignación para $v(F)$ que contradice F. Si $n = |v(F)|$, entonces hay 2^n diferentes asignaciones definidas sobre $v(F)$. Denotamos al total de asignaciones de F, por $S(F)$.

Si $F_1 \subset F$ es una fórmula que consiste de algunas cláusulas de F, y $v(F_1) \subset v(F)$, una asignación sobre $v(F_1)$ es un asignamiento parcial sobre $v(F)$. Si $n = |v(F)|$ y $m = |v(F_1)|$, cualquier asignación sobre $v(F_1)$ tiene 2^{n-m} extensiones como asignaciones de $v(F)$. Si s asigna valores lógicos a todas las variables en F, entonces s es una asignación total de F.

El problema SAT consiste en determinar si F tiene un modelo. SAT(F) denota el conjunto de modelos de F, entonces $\text{SAT}(F) \subseteq S(F)$. El conjunto $\text{FAL}(F) = \text{S}(F) \backslash \text{SAT}(F)$ consiste de las asignaciones de $S(F)$ que falsifican a F. Para cualquier fórmula proposicional F, $S(F) = \text{SAT}(F) \cup \text{FAL}(F)$.

III. Reducción de formas normales conjuntivas

Al tratar el problema SAT($K \wedge \varphi$), introducimos reglas para simplificar una FNC, manteniendo sólo las subfórmulas que determinan la satisfactibilidad de la fórmula original. Por ejemplo, para una FNC es común eliminar todas las cláusulas redundantes, así como cláusulas tautológicas, cláusulas repetidas y cláusulas con literales puras.

Regla de cláusula subsumida: Dadas dos cláusulas C_i, C_j de una FNC F, si $\text{Lit}(C_i) \subseteq \text{Lit}(C_j)$, entonces C_j es subsumida por C_i, C_j y pude ser eliminada de F, porque toda asignación que satisface a C_j también satisface a C_i, esto es $\text{SAT}(C_j) \subseteq \text{SAT}(C_i)$. Por lo que es suficiente sólo mantener a C_i en la FNC. Aún más, esta regla de cláusula subsumida puede combinarse con resolución para simplificar la FNC φ, como se muestra en el lema siguiente:

Lema 1. Sea $(x, y) \in K$ y una cláusula $(\neg x, y, z) \in \varphi$ entonces el resolvente (y, z) es una cláusula binaria que puede ser adicionada a K y su cláusula padre $(\neg x, y, z)$ puede ser eliminada de φ.

Prueba. Tenemos que $(x \vee y)$ y $(\neg x \vee y \vee z) \equiv y \vee ((x \wedge \neg x) \vee (x \wedge z)) \equiv y \vee (x \wedge z) \equiv (x \vee y) \wedge (y \vee z)$, y si estas 2 últimas cláusulas son preservadas en K entonces la cláusula $(\neg x, y, z)$ puede ser borrada de φ, porque ésta es subsumida por $(y \vee z) \in K$ y por lo tanto, el conjunto de modelos de $(K \wedge \varphi)$ es preservado sin cambios.

Regla de resolución acotada: Sean $(x \vee y \vee z_1)$ y $(\neg x \vee y \vee z_2)$ 2 cláusulas ternarias de una fórmula φ, llamamos a $(y \vee z_1 \vee z_2)$ la regla de resolución explícita de las cláusulas anteriores. Sean $(x \vee y \vee z_1)$, $(x \vee y \vee z_2)$ y $(\neg z_1 \vee \neg z_2 \vee z_3)$ 3 cláusulas ternarias de una fórmula de entrada F llamamos a $(x \vee y \vee z_3)$ la regla de la resolución implícita de las 3 cláusulas anteriores.

Nótese que la regla anterior aplica resolución sobre 3-cláusulas, pero sólo si los resolventes no tienen más de 3 literales. La regla de resolución es también útil para mover cláusulas de φ a K. Por ejemplo, si φ contiene cláusulas tipo: (x, y, z) y $(\neg x, y, z)$, entonces éstas pueden ser eliminadas de φ y la cláusula (y, z) se adiciona a K. Otra regla común que tiene que ser adaptada para nuestro propósito, es:

Regla de la literal Pura: Sea F una FNC, $l \in \text{Lit}(F)$ si , l aparece en F pero $\neg l$ no aparece en F.

Si una cláusula contiene una literal pura, tal cláusula puede ser eliminada de F, manteniéndose el valor lógico de F. Porque si la literal l es verdadera, la cláusula conteniendo l es también verdadera, por tanto esto puede ser eliminado de F.

Sin embargo estas reglas tienen que ser aplicadas con cuidado en nuestro problema, ya que en el proceso de adicionar nuevas cláusulas φ a K, las literales inicialmente puras podrían ser no puras en $K \cup \varphi$. Por lo tanto, para eliminar cláusulas con literales puras deben ser aplicadas con un alcance local, trabajando en cada instancia $(K \wedge \varphi)$. Pero, si un nuevo conjunto de cláusulas φ_{i+1} es considerado, entonces todas las literales puras que fueron eliminadas en $(K \wedge \varphi)$ deben ser retornadas a φ_{i+1}.

La regla principal a aplicar en nuestra propuesta algorítmica, es propagación unitaria. Para introducirla, definimos primero la reducción de una fórmula por una literal, conocido como resolución unitaria.

Definición 1. Sea F una FNC y x una literal de F. La resolución de F por x (forzar x), denotado por $F[x]$, es la fórmula generada de F por las siguientes 2 reglas:

a) Remover de F las cláusulas conteniendo x (regla de cláusula subsumida).
b) Remover $\neg x$ de las cláusulas restantes (regla de resolución unitaria).

Forzar x en F es también llamada reducción unitaria. La reducción por un conjunto de literales puede establecerse inductivamente de la siguiente manera: sea $s = \{l_1, l_2, \ldots, l_k\}$ una asignación parcial de $v(F)$. La reducción de F por s se define por aplicar resolución unitaria iterativamente, para l_i, $i = 1, \ldots k$. Lo que significa reducir F por l_1 y luego por l_2 denotado como: $F[l_1, l_2]$ y así sucesivamente. Cuando $s = \emptyset$, $F[s] = F$.

Ejemplo 1. Sea $F = \{\{x_1, \neg x_2\}, \{x_1, x_2\}, \{x_1, x_3\}, \{\neg x_1, x_3\} \{\neg x_2, x_4\}, \{\neg x_2, \neg x_4\}, \{x_2, x_5\}, \{x_3, \neg x_5\}\}$. $F[x_2] = \{\{x_1\}, \{x_1, x_3\}, \{\neg x_1, x_3\}, \{x_4\}, \{\neg x_4\}, \{x_3, x_5\}\}$. Si $s = \{x_2, \neg x_3\}$, $F[s] = \{\{x_1\}, \{x_1\}, \{\neg x_1\}, \{x_4\}, \{\neg x_4\}, \{\neg x_5\}\}$.

Sea F una FNC y s una asignación parcial de F. Si un par de cláusulas unitarias contradictorias son obtenidas cuando $F[s]$ es calculada, entonces F es falsificada por s. Si durante el cálculo de $F[s]$, nuevas cláusulas unitarias son generadas, la asignación parcial s se extiende añadiéndole las cláusulas unitarias encontradas, es decir $s = s \cup \{u\}$, donde $\{u\}$ es una cláusula unitaria. Así que $F[s]$ puede reducirse nuevamente utilizando las nuevas cláusulas unitarias. El proceso iterativo anterior es generalizado, y llamamos a este proceso iterativo *Propagación Unitaria*(F, s). Por simplicidad se abrevia *Propagación Unitaria*(F, s) por $UP(F, s)$.

Consideraremos F como una fórmula conjuntiva tipo: $(K \wedge \varphi)$ y s una asignación parcial de F. Denotamos como $K' = UP(F, s)$ a la nueva fórmula conjuntiva que queda como resultado de aplicar $UP(F, s)$. Observe que si s falsifica a F entonces K' contiene la cláusula nula y en este caso, decimos que $F[s]$ es insatisfactible. Y cuando s satisface a F, K' es vacía.

De ahora en adelante, consideremos que K es una 2-FNC y φ una 3-FNC. En primer lugar, aplicamos las reglas de simplificación descritas anteriormente, con el fin de reducir φ y extender K, manteniendo el valor lógico de $(K \wedge \varphi)$.

IV. Grafos Planos

Sea $G = (V, E)$ un grafo conectado no dirigido y sea $n = |V|$ y $m = |E|$. Los grafos planos juegan un papel importante tanto en la teoría de grafos como en la de dibujo de grafos. Los grafos planos tienen varias propiedades interesantes: por ejemplo, son grafos dispersos, 4-coloreables, y su estructura interna puede describirse de manera breve y elegante [7].

Un dibujo Γ de un grafo plano G asigna cada vértice v a un punto distinto $\Gamma(v)$ del plano y cada arista $\{u, v\}$ a una línea simple o curva de Jordan $\Gamma(u, v)$ con los puntos finales $\Gamma(u)$ y $\Gamma(v)$.

Un dibujo es plano si no hay cruce de aristas, excepto posiblemente en puntos finales comunes. Un grafo es plano si admite un dibujo plano. Un dibujo plano divide el plano en regiones conectadas llamadas caras. La cara sin límites usualmente es llamada *cara externa* o cara del exterior. Si todos los vértices son incidentales a la cara exterior, el grafo es llamado *outerplanar*.

Un grafo no conexo es plano si y solo si todas sus componentes conectadas son planas. Quizás la propiedad más reconocida sobre grafos planos es la que establece el Teorema de Euler, que muestra que los grafos planos son escasos. Es decir, dado un grafo plano con n vértices, m aristas y f caras, se cumple que $n - m + f = 2$.

Un corolario simple que se deriva del teorema de Euler, es que para un grafo plano maximal con al menos tres vértices, donde cada cara es un triángulo ($2m = 3f$), tenemos $m = 3n - 6$, y por lo tanto, para cualquier grafo plano se cumple que $m \leq 3n - 6$. Si el grafo es un árbol, entonces $m = n - 1$.

La primera caracterización de grafos planos se debe a Kuratowski [8], y establece que un grafo es plano si y solo si no contiene un subgrafo que es una subdivisión de K_5 o $K_{3,3}$, donde K_5 es el grafo completo de orden 5 y $K_{3,3}$ es el grafo bipartito completo con 3 vértices en cada uno de los conjuntos de la partición. Un resultado posterior equivalente reescrito en términos de grafo *menores*, es el teorema de Wagner's [9] que establece que un grafo G es plano si y solo si no tiene K_5 o $K_{3,3}$ como un *menor*, es decir, K_5 o $K_{3,3}$ no se pueden obtener de G al contraer algunas aristas, eliminando algunas aristas y eliminando algunos vértices aislados. Obsérvese que las dos caracterizaciones son diferentes ya que un grafo puede admitir K_5 como un menor sin tener un subgrafo que es una subdivisión de K_5 [10].

El coloreo de grafos planos se ha investigado ampliamente y representa un área de interés relevante tanto en la teoría de grafos como en la teoría de la complejidad, ya que involucra la frontera entre procedimientos computacionales eficientes e intratables.

El Teorema de los cuatro colores [11] [12] [13] afirma que cualquier grafo plano es 4-coloreable y resuelve una conjetura que por más de un siglo, fue el problema no resuelto más famoso en la teoría de grafos y quizás, en todas las matemáticas [14]. Mientras que un grafo plano libre de triángulos es 3-colorable [15], y tal 3-coloreo se puede encontrar en tiempo lineal [16] [17] [18].

A. Grafos Serial-Paralelo

Un grafo serial-paralelo es un tipo de grafo plano. Los grafos serial-paralelo se forman a través de composiciones de grafos de dos terminales. Un grafo de dos terminales, denotado TTG, es un grafo con 2 vértices distinguidos: s, t llamados fuente(s) y objeto (t) (ver Figura 1).

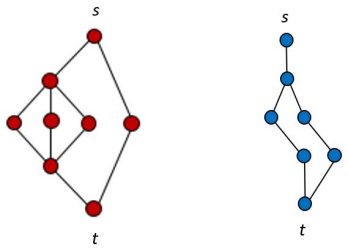

Fig. 1. Ejemplo de TTG's.

De entre las diferentes definiciones para un grafo serial-paralelo, nos basaremos principalmente en la propuesta por Eppstein [19].

Una composición paralela (Figura 2) $P_c = P_c\ (x, y)$ de dos TTG's: x y y es un nuevo TTG creado de la unión disjunta de los grafos a través de fusionar los vértices fuentes de x, y y para crear la nueva fuente de P_c y fusionando los vértices objetivos de x y y para crear el nuevo objetivo de P_c.

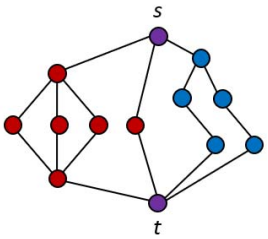

Fig. 2. Composición Paralela.

La composición serie (Figura 3) $S_c = S_c\ (x,y)$ de los TTG`s x, y, se forma al fusionar el vértice objetivo de x con el vértice fuente (s) de y. El vértice fuente de x se convierte en la fuente de S_c y el vértice objetivo de y se convierte en el objetivo de S_c.

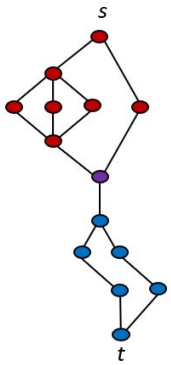

Fig. 3. Composición Serial.

Un grafo serial-paralelo de dos terminales TTSPG es un grafo que puede ser construido por una secuencia de composiciones: serial y/o paralelo, iniciando a partir de un conjunto de copias del grafo de una sola arista K_2, identificando sus dos vértices terminales.

La 3-colorabilidad de grafos planos (incluyendo grafos triangulares) es reconocida como un problema NP-completo [20]. En esta línea de trabajo, es que se consideran instancias de grafos planos donde se pueda encontrar demostraciones sucintas para determinar cotas superiores sobre el número de colores a usar. En este esfuerzo, mostraremos que nuestra propuesta algorítmica efectivamente acota superiormente a 3 el número de colores para hallar un coloreo válido de un grafo plano tipo serial-paralelo.

Un grafo TTSPG permite representar de forma natural circuitos (eléctricos, electrónicos, de comunicación, etc.) que combinan composiciones seriales y paralelas. También nos permiten representar estructuras lógicas como son los diagramas binarios ordenados de decisión (OBDD).

Los OBDD son una de las estructuras de datos más ampliamente usadas al realizar operaciones básicas sobre funciones Booleanas, así como en aplicaciones del área de verificación de circuitos, síntesis, y revisión de modelos. Por ejemplo, en [21], los OBDD's son utilizados para la construcción de árboles generadores con pesos mínimos (MST) usados para evitar de mejor manera el particionamiento de redes de transmisión, en lugar de utilizar árboles basados en ligas de transmisión de rango común. Hay una línea de investigación para usar MST's, basados en grafos serial – paralelo, para el ruteo y la reducción de energía en el contexto de redes inalámbricas y redes adhoc [21] [22].

La característica principal de la topología de los OBDD's es que estos son grafos tipo serial – paralelo, por lo que el diseñar algoritmos específicos para estas topologías, como es el caso del algoritmo que presentamos, puede ser de utilidad en otros procesos donde se aplican los OBDD's.

Otro tipo de aplicación es cuando se considera una transmisión tipo *unicast* donde se tiene un flujo de transmisión de datos uno a uno. En este caso se considera que el flujo se realiza desde un único emisor a un conjunto de receptores, como por ejemplo de un servidor a un grupo de trabajo de una LAN. Aunque varios usuarios puedan solicitar la misma información al servidor al mismo tiempo, en el entorno unicast el servidor responderá a las peticiones de los usuarios enviando la información a cada usuario. Este tipo de transmisión se modela vía grafos serial paralelo, con una sola fuente (que fungiría como el servidor) y varios vértices sumideros o vértices objetivo (los usuarios).

Al aplicar nuestro algoritmo de 3-coloreo sobre grafos serial paralelo G, con una fuente y varios sumideros, bastará con unir todos los sumideros a un sumidero general formándose un nuevo grafo G_1. A G_1 se le aplica nuestro algoritmo y el 3-coloreo resultante para G_1 continuará siendo un 3-coloreo para todo subgrafo de G_1, inclusive para G que ha modelado la comunicación unicast.

V. Propuesta Algorítmica

Nuestra propuesta algorítmica para construir un 3-coloreo de un grafo serial-paralelo se basa en transformar el grafo de entrada a una fórmula tipo $\varphi_0 = (K \wedge \varphi)$, definiendo así la primer fórmula φ_0 en un proceso de satisfactibilidad incremental. Posteriormente, se irán agregando nuevas cláusulas para conformar una serie de fórmulas φ_i i=1,…, k. Los modelos para $((K \wedge \varphi_0) \wedge \varphi_1 \wedge \ldots \varphi_k)$ determinarán un 3-coloreo válido del grafo serial-paralelo de entrada.

Nuestro algoritmo se basa en la reducción de un 3-coloreo de un grafo a una fórmula $(K \wedge \varphi)$ justificado por el lema siguiente.

Lema 2. El 3-coloreo de un grafo es polinomialmente reducible al problema SAT$(K \wedge \varphi)$, con K una 2-FNC y φ una 3-FNC.

Prueba. Sea $G = (V, E)$ un grafo donde $n = |V|$, $m = |E|$. Definimos las variables lógicas $x_{v,c}$ para indicar que al vértice $v \in V$ se le asigna el color $c \in \{1,2,3\}$. Para cada vértice $v \in V$, tres variables lógicas $x_{v,1}, x_{v,2}, x_{v,3}$ son creadas. Por tanto habrá $3*n$ variables Booleanas en $v(K) \cup v(\varphi)$.

Consideremos las restricciones que formarán a la fórmula K. Para cada arista $e = \{u,v\} \in E$, u y v deben tener un color

diferente. Esta restricción se modela mediante las siguientes tres cláusulas binarias: $(\neg x_{u,1} \vee \neg x_{v,1}) \wedge (\neg x_{u,2} \vee \neg x_{v,2}) \wedge (\neg x_{u,3} \vee \neg x_{v,3})$. Existen $3*|E|$ cláusulas binarias de esta clase.

Otra clase de restricción binaria que restringe el que cada vértice no debe tener más de un color, se modela por las siguientes tres cláusulas: $(\neg x_{v,1} \vee \neg x_{v,2}) \wedge (\neg x_{v,2} \vee \neg x_{v,3}) \wedge (\neg x_{v,3} \vee \neg x_{v,1})$, para cada vértice $v \in V$.

Se tendrán $3*|V|$ cláusulas binarias de esta clase. Ambos conjuntos de $3*(|V|+|E|)$ cláusulas binarias conforman la fórmula K que está en 2-FNC.

En este caso, SAT(K) no es suficiente para determinar un 3-coloreo de G, ya que aunque K fuera satisfactible, no se construye directamente un 3-coloreo de G a partir de SAT(K).

Para construir soluciones al 3-coloreo de G, una 3-FNC φ será formada por las cláusulas que modelan la restricción de que a cada vértice se le debe asignar al menos un color. Entonces, para cada vértice $v \in V$ la siguiente cláusula es generada: $(x_{v,1} \vee x_{v,2} \vee x_{v,3})$. φ tiene $|V|$ 3-cláusulas. Además, cada uno de las $3 * n$ variables de $v(K)$ tiene solo una ocurrencia en φ.

Esta reducción se realiza en tiempo polinomial sobre el tamaño n y m, ya que consiste en crear $3 * (n + m)$ cláusulas binarias para K y (n) cláusulas ternarias para φ. Además, se cumple que G tiene un 3-coloreo si y solo si $(K \wedge \varphi)$ es satisfactible.

Nótese que cada componente de SAT($K \wedge \varphi$) es polinomialmente resoluble. Por ejemplo, SAT(K) se resuelve en tiempo lineal sobre el tamaño $|K|$, al ser K una 2-FNC. Similarmente, al ser φ una 3-FNC, donde cada una de sus variables ocurre una sola vez, SAT(φ) es un problema trivial, ya que cada variable es pura en φ y por tanto φ es satisfactible. Sin embargo, la satisfactibilidad de la conjunción de ambas fórmulas: SAT($K \wedge \varphi$) es un problema NP-completo [1].

El carácter no determinista en el proceso de asignar un color a cada vértice de G en la reducción presentada (lema 2), se refleja al requerir un tiempo exponencial para resolver su respectiva instancia 3-SAT. Sin embargo, para topologías de grafos especiales, como es el caso de los grafos serial-paralelo, se pueden determinar nuevas restricciones que se van añadiendo a la fórmula inicial $(K \wedge \varphi)$, de forma que nos garantice el poder solucionar la nueva instancia de 3-SAT de una forma incremental, y requiriendo un tiempo de cómputo acotado polinomialmente.

Sea G un TTSPG (grafo serial-paralelo) cuya especificación de un 3-coloreo ha sido codificada como $\varphi_0 = (K \wedge \varphi)$, lo que resulta en que φ_0 es una 3-FNC. Para evitar una complejidad exponencial en tiempo al formar un 3-coloreo a partir de SAT(φ_0), es necesario ir agregando nueva información en el proceso de satisfactibilidad, lo que se logra al adicionar nuevas cláusulas que ayudan a determinar el 3-coloreo de G.

En este caso, los vértices claves para definir un 3-coloreo se conforman por los pares (s_i, t_i) de cada componente serial-paralelo. Cuando en cada camino de s a t no hay otras componentes paralelas, decimos que la componente (s, t) es simple, es decir, cada camino de s a t es un camino simple, y cada vértice del camino tendrá grado 2 (excepto para s y t).

Lema 3. Cada componente serial de un grafo serial-paralelo es 2-coloreable.

Prueba. Las partes seriales en un grafo serial-paralelo tienen vértices con grado 2 o menos y por lo tanto, las partes seriales en un TTG son 2 coloreables, y entonces pueden ser coloreados usando de manera alternada 2 colores desde s a t.

Asociamos a cada vértice v en G un conjunto de colores prohibidos, denotado por $tabu(v)$. Así, cuando a un vértice v se le asigna un color: $Color(v)$ entonces a todo vértice vecino u de v, se coloca en $tabu(u)$ el color asignado a v, ya que este color será ahora prohibido para u.

En una componente serial-paralela, si alguno de los vértices s o t es eliminado, entonces la componente se convierte en un árbol con raíz en s si t es eliminado, o viceversa.

El algoritmo que proponemos forma el conjunto $Fr=\{(s_0,_3),(s_1,_3 \vee t_1,_3), (s_2,_3 \vee t_2,_3),\ldots, (s_k,_3 \vee t_k,_3)\}$ conteniendo la cláusula $(s_i,_3 \vee t_i,_3)$ asociada a las dos terminales de cada componente serial-paralelo de G, ordenadas por su aparición en G, de las componentes más externas hacia las más internas. En el caso de una fuente s_i con varios sumideros $t_{i1}, \ldots, t_{ik}$, uniremos todos estos sumideros a un solo vértice objetivo t_i, formándose así el nuevo par (s_i, t_i).

Regla de asignación del color 3. Nuestra propuesta algorítmica asigna a cada uno de los componentes paralelos de un grafo, el color 3 al vértice s, o al vértice t, o a ambos. Nuestra propuesta inicia asignando color 3 al primer vértice fuente s_0, y esto se hace al adicionar la cláusula $(s_0,_3)$ a φ_0.

Lema 4. Si $Color(s_i) = 3$ entonces $3 \notin tabu(t_{i+1})$.

Prueba. (Por contradicción) Supongamos que se asigna $Color(s_i) = 3$, y que esto implica que $3 \in tabu(t_{i+1})$, entonces s_i sería adyacente a t_{i+1}. Pero el vértice s_{i+1} está entre s_i y t_{i+1} y por tal, s_i y t_{i+1} no son adyacentes.

Lema 5. Dado un subgrafo G conteniendo una sola dos-terminal (s, t), si se cumple $(s,_3 \vee t,_3)$ entonces el grafo G es 3-coloreable.

Prueba. Asuma que $\{s, t\} \notin E(G)$, se asigna $f(s) = f(t) = 3$. Sea ahora $G' = G-\{s, t\}$, G' es un grafo acíclico el cual es 2-coloreable (usando sólo los colores 1 y 2), y entonces G es 3-coloreable. Si $\{s, t\} \in E(G)$, sea $f(s)=3$ y $G'=G-\{s\}$ es un árbol con raíz en el nodo t. G' puede colorearse por niveles iniciando con el color 1 para t y usando los colores 2 y 1 por niveles hasta llegar a todos los nodos hoja de G', formándose así, un 3-coloreo para G.

Teorema de Exactitud. Si para toda dos-componente (s_i, t_i) de un grafo TTSPG G se cumple la cláusula: $(s_i,_3 \vee t_i,_3)$ entonces el grafo G es 3-coloreable.

Prueba. Por inducción sobre el número de dos componentes en el grafo G.

i) Para una sola dos-componente (s_1, t_1), este caso se demostró anteriormente (Lema 5).
ii) Se supone que la propiedad se cumple en todo grafo con n dos-componentes (Hipótesis Inductiva).
iii) Sea G un grafo con n+1 dos-componentes. Y sea (s_0, t_0) la componente más externa en G. Nuestro algoritmo asigna $f(s_0) = 3$, y $\forall y \in N(s_0)$ se marca $tabu(y)=tabu(y) \cup \{3\}$, formándose así, $G'=G-\{s_0\}$. G' contiene n dos-componentes, cumpliéndose la hipótesis inductiva para G'. Además, no puede darse el caso de que exista una dos-componente $(s_i, t_i) \in G'$ tal que $tabu(s_i) = tabu(t_i)=3$, ya que

esto implicaría que se tienen las aristas $\{s_0, s_i\}$ y $\{s_0, t_i\}$ en G, lo que contradice que (s_0, t_0) y (s_i , t_i) son dos-componentes distintas en G. Así que o bien se cumple $(s_{i,3})$ o bien $(t_{i,3})$, y por tanto también se cumple: $(s_{i,3} \vee t_{i,3})$. El argumento anterior itera para toda dos-componente $(s_i , t_i) \in G'$ y por tanto, toda dos-componente cumple las condiciones del lema anterior para ser 3-coloreable, y por tanto, G es también 3-coloreable.

Como toda cláusula $(s_{i,3} \vee t_{i,3})$, i=1,…,k se cumple, así como: $(s_{o,3})$, entonces SAT$((K \wedge \varphi_0) \wedge Fr)$ determina un 3-coloreo para cualquier grafo serial-paralelo TTSPG G.

Nuestra propuesta algorítmica inicia ejecutando $K'=UP((K \wedge \varphi_0), (s_{0,3}))$, lo que permite determinar de forma automática, los colores válidos dentro del conjunto {1,2,3} para algunos de los vértices de G. Si durante la ejecución de $K'=UP((K \wedge \varphi_0), (s_{0,3}))$ no se asigna color a alguna dos-componente (s_i , t_i), entonces de forma incremental y de acuerdo al orden definido en Fr, se adiciona una nueva cláusula unitaria (u), ejecutando ahora: $K'=UP(K',(u))$, donde (u) será $(s_{i,3})$, o bien $(t_{i,3})$, según los conjuntos tabús de las variables: s_i , t_i. Este proceso de adición de nuevas cláusulas unitarias nos lleva a un proceso de satisfactibilidad incremental.

Al terminar el proceso de satisfactibilidad incremental, se obtendrán los modelos para la fórmula lógica $((K \wedge \varphi_0) \wedge Fr)$, y estos modelos codifican un 3-coloreo para el grafo de entrada G.

Se cumple que para los vértices terminales (s_i , t_i) se satisface la cláusula $(x_{i,3} \vee x_{i,3})$. En tanto que para los caminos acíclicos en G, los que se forman al remover los anteriores vértices terminales, se satisfacen las variables $x_{v,2}$ o $x_{v,1}$ según al vértice $v \in v(G)$ se le haya asignado color 2 o 1, respectivamente. Por tanto, las variables $(x_{v,c})$ que toman valor verdadero en los modelos de $((K \wedge \varphi_0) \wedge Fr)$, determinan que al vértice $(v \in v(G))$ se le asigna el color c, con $c \in \{1,2,3\}$. Así, nuestra propuesta algorítmica encuentra de forma automática y basada en el teorema 1, un 3-coloreo sobre grafos serial-paralelos.

Realizamos el análisis matemático sobre la complejidad en tiempo de nuestra propuesta algorítmica.

De acuerdo al lema 2, se tiene que $|K| = 3*(|V|+|E|)$, esto es, K es de orden lineal con respecto a la longitud del grafo de entrada G. Mientras que $|F_r|$ se corresponde con el número de componentes serial-paralelo que hay en G.

Es conocido que el tiempo de ejecución de un 2-coloreo sobre componentes acíclicas es de orden lineal sobre la longitud del grafo. Por otro lado, se sabe que la ejecución del proceso de propagación unitaria $UP(K, (u))$ es de orden $O(|K|)$ [23]. Esto es el costo en tiempo que tenemos al hacer los llamados: $K=UP(K, (s_{i,3}))$ o al ejecutar $K=UP(K, (t_{i,3}))$. Nuestro algoritmo, hará un máximo de llamados al proceso UP de acuerdo al número de diferentes componentes serial-paralelo existentes en G.

Por tanto, la complejidad en tiempo de nuestro algoritmo es de orden $O(|K| * |F_r|)$, lo que resulta en un algoritmo de orden lineal en tiempo en base al tamaño del grafo de entrada y al número de componentes serial-paralelo que hay en el mismo grafo.

Un algoritmo para el 3-coloreo de grafos serial-paralelo

Entrada: G - un grafo plano serial-paralelo.
Salida: Una función de 3-coloreo, tal que a cada vértice del grafo le asigna un color.
Proceso

1. Identificar vértices s(fuente) y t(objeto) por niveles de anidamiento, formándose el conjunto Fr.
2. Remover todo vértice de grado < 3.
3. Aplicar $K'=UP((K \wedge \varphi_0), (s_{0,3}))$
4. Mientras $(Fr \neq \emptyset)$ y $(K' \neq \emptyset$) hacer
5. Selecciona componente $(s_i, t_i) \in Fr$ del nivel más externo
6. If $(tabu(s_i) = 3)$ then { $Color(t_i) = 3$; $v= t_i$; $G = G- t_i$}
7. Else
8. { $Color(s_i) = 3$; $v= s_i$; $G = G- s_i$}
9. End if
10. $\forall y \in N(v) : tabu(y)=tabu(y) \cup \{3\}$
11. Aplicar $K'=UP(K', (v_3))$
12. End mientras
13. Aplica 2-coloreo sobre vértices de grado < 3
14. Return /* regresa un 3- coloreo del Grafo G */

Ilustremos la aplicación de nuestro algoritmo para el 3-coloreo de un grafo serial-paralelo, con el siguiente ejemplo.

Ejemplo 2

Consideremos el grafo serial-paralelo G de la Figura 4.

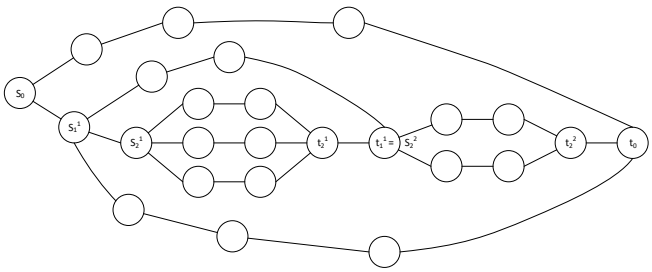

Fig. 4. Grafo serial-paralelo G.

Paso inicial: Reconocer los pares (s_i, t_i), removiendo los vértices de grado < 3, ya que son 2-coloreables como se ve en Figura 5.

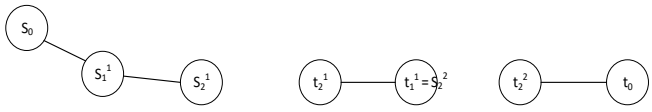

Fig. 5. Subgrafo con fuentes-objetos del grafo G.

Se empieza asignando color $f(s_0)= 3$, el cual se representa con el color azul, después se continúa asignando color 3 a cada componente paralela, como se muestra en la Figura 6.

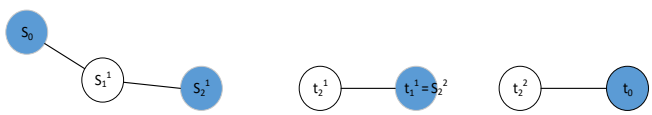

Fig. 6. Primera parte coloreada del Grafo G.

Recuperando vértices de grado < 3, ya que son 2-coloreables como se ve en la Figura 7.

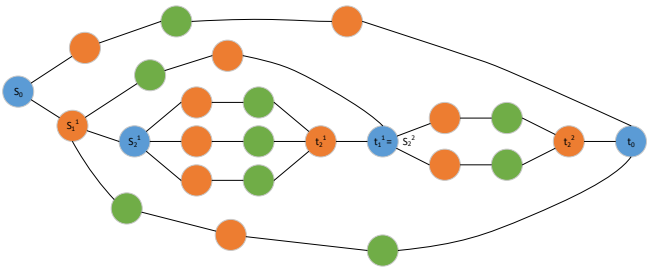

Fig. 7. Grafo G con un 3-coloreo.

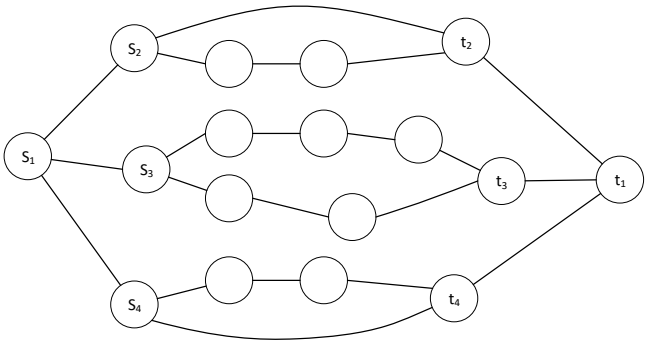

Fig. 8. Grafo serial paralelo de entrada G.

Ejemplo 3. Se reconoce toda pareja (s_i, t_i) de las componentes serial- paralelo del grafo de entrada (Figura 8). Se remueven los vértices de grado < 3 y se asigna el color 3 a la primera fuente (s_1), como se muestra en la Figura 9.

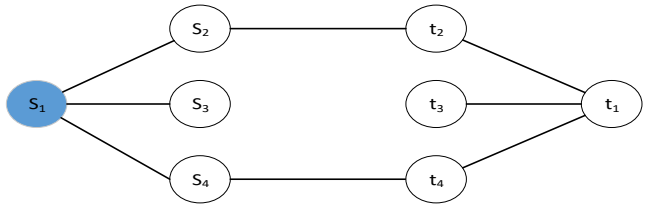

Fig. 9. G con primera fuente coloreada y vértices removidos.

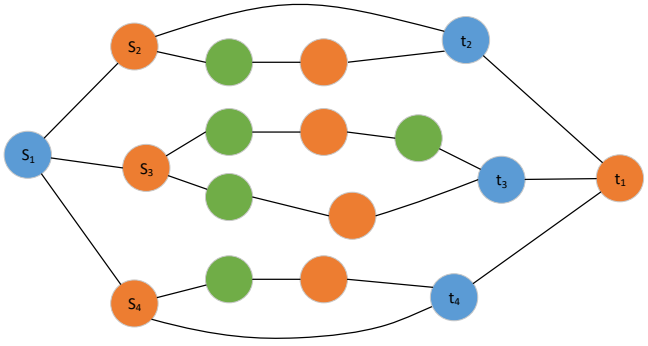

Fig. 10. Un 3-coloreo de G.

Como ningún vértice fuente de las demás dos componentes pueden tomar color 3, entonces los vértices objetivo tomarán el color 3. Cuando se recuperan los vértices de grado 2, se aplica un 2-coloreo sobre estos vértices y G queda coloreado como se muestra en la Figura 10.

VI. CONCLUSIONES

Hemos diseñado un nuevo algoritmo basado en el problema de satisfactibilidad incremental (ISAT) para resolver el problema del 3-coloreo sobre grafos serial-paralelo. La propuesta inicia con una fórmula proposicional en forma conjuntiva φ_0 que especifica las restricciones que un 3-coloreo válido requiere sobre G - un grafo serial-paralelo.

Si se busca un modelo de φ_0, se requeriría un tiempo exponencial de cómputo para el 3-coloreo de G, pero al ir agregando nuevas cláusulas unitarias y binarias a φ_0, se va determinando de forma incremental, los colores de los vértices de cada componente serial-paralelo del grafo de entrada. Nuestra propuesta algorítmica se basa en aplicar propagación unitaria, siendo la cláusula unitaria una de las dos terminales de las componentes paralelas, lo que resulta en un método para hallar un 3-coloreo válido del grafo G, que es de complejidad lineal en tiempo en base a la longitud del grafo de entrada y al número de componentes serial-paralelo del mismo grafo.

Hemos demostrado la exactitud de nuestra propuesta algorítmica, mostrando que la cota superior sobre el número de colores para un grafo plano serial-paralelo es de 3.

AGRADECIMIENTOS

Los autores agradecen el apoyo otorgado por parte del CONACyT y SNI-México. También se agradece el apoyo de la Facultad de Ciencias de la Computación de la Benemérita Universidad Autónoma de Puebla.

REFERENCIAS

[1] G. De Ita, J. R. Marcial-Romero and J. A. Hernández, "The Incremental Satisfiability Problem for a Two Conjunctive Normal Form", Electronic Notes in Theoretical Computer Science, vol. 328, pp. 31-45, 2016.

[2] S. Wieringa, "Incremental Satisfiability Solving and its Applications", PhD thesis, Department of Computer Science and Engineering, Alto University Pub., 2014.

[3] M. Davis, G. Logemann and D. Loveland, "A machine program for theorem proving". Communications of the Association for Computing Machinery, pp. 394-397, 1962.

[4] J. Whittemore, K. Joonyoung and K.A. Sakallah, "SATIRE: A New Incremental Satisfiability Engine", In Proceedings of the 38th Design Automation Conference (DAC)-ACM, Las Vegas - USA, pp. 542-545, 2001.

[5] A. Nadel, V. Ryvchin and O. Strichman, "Ultimately Incremental SAT", Proc. SAT 2014, LNCS vol. 8561, pp. 206–218, 2014

[6] M. Mouhoub and S. Sadaoui, "Systematic versus non systematic methods for solving incremental satisifiability", Int. J. on Artificial Inteligence Tools, vol. 16, no.1, pp.543-551, 2007.

[7] P. Cortese and M. Patrignani, "Planarity Testing and Embedding", Press LLC, 2004.

[8] K. Kuratowski, "Sur le problème des courbes gauches en topologie", Fund. Math., vol. 15, pp.271–283, 1930.

[9] K. Wagner, "Über eine Eigenschaft der ebenen Komplexe", Mathematische Annalen, vol. 114, pp.570–590, 1937.

[10] F. Haray and W. T. Tutte, "A dual form of Kuratowski's theorem". Canad. Math. Bull, vol 8, pp.17–20, 1965.

[11] K. Appel and W. Haken. “Every planar map is four colourable, part I: discharging”, Illinois J. Math., vol 21, pp. 429–490, 1977.
[12] K. Appel, W. Haken, and J. Koch, “Every planar map is four colourable, part II: Reducibility”, Illinois Journal of Mathematics, vol 21 pp.491–567, 1977.
[13] N. Robertson, D.P. Sanders, P.D. Seymour, and R. Thomas, “The four color theorem”, J. Combin. Theory Ser. B, vol.70, pp. 2–4, 1997.
[14] F. Harary, “Graph Theory”. Addison-Wesley, Reading, Mass, 1969.
[15] H. Grötzsch.”Ein Dreifarbensatz für dreikreisfreie Netzeauf der Kugel“, Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur. Reihe vol. 8, pp. 109-120, 1959.
[16] Z. Dvorak, K. Kawarabayashi, and R. Thomas, “Three- coloring triangle free planar graphs in linear time”, In Claire Mathieu, editor, SODA, SIAM, pp 1176–1182, 2009.
[17] J. Stacho, “3-Colouring AT-Free Graphs in Polynomial Time”, vol. 64, pp.384-399, 2012.
[18] Z. Dvorak, D. Kral and R.Thomas, “Three-coloring triangle-free graphs on surfaces VII. A linear-time algorithm”, 2016.
[19] D. Eppstein, “Parallel recognition of series-parallel graphs”, Information and Computation, vol. 98, no.1, pp. 41-55, 1992.
[20] M. R. Garey, D. S. Johnson, and L. Stockmeyer, “Some simplified NP-complete graph problems”, Theoretical Computer Science, vol. 1, no.3, pp.237–267, 1976.
[21] C. Y. Márquez, I. L. Yáñez, O. C. Nieto and A. J. A. Cruz, “BDD-based Algorithm for the Minimum Spanning Tree in Wireless Ad-hoc Network Routing”, IEEE Latin America Transactions, vol. 11, no. 1, 2013.
[22] C. Toh, “Maximum Battery Life Routing to Support Ubiquitous Mobile Computing in Wireless Ad Hoc Networks”, IEEE Communications Magazine, vol. 21, no. 4, pp. 138-147, 2001.
[23] H. Zhang and M.E. Stickel, “An efficient algorithm for unit propagation”, Proc. Fourth Int. Symp. on Artificial Intelligence and Mathematics, Florida,1996.

Cristina López Ramírez es profesor - investigador de la División Académica de Informática y Sistemas (DAIS) de la Universidad Juárez Autónoma de Tabasco. Áreas de interés: Problemas de Satisfactibilidad y Sistemas Inteligentes. Actualmente es estudiante de doctorado en Language & Knowledge Engineering (LKE) en la Fac. de Cs. de la Computación en la Benemérita Universidad Autónoma de Puebla (BUAP), México.

Guillermo De Ita Luna es doctor en Ing. Eléctrica por el CINVESTAV- IPN. Ha trabajado como desarrollador y consultor en sistemas de bases de datos y sistemas de información geográfica. Ha sido Profesor investigador de tiempo completo por 27 años en la Facultad de Ciencias de la Computación, BUAP, Puebla. Miembro del Sistema Nacional de Investigadores de México.