Metaheurísticas para el Diseño de Redes Multioverlay Robustas

L. Gómez, F. Casalongue, G. Lasalt

Abstract— **Las redes overlay son una realidad instaurada desde hace un buen tiempo en las organizaciones y la vida diaria. Este tipo de redes poseen la particularidad de estar formadas por dos (o más) capas. En este proyecto se pretende atacar el problema del diseño óptimo en costos de una red multioverlay tolerante a fallas. En particular interesa resolver el problema para un escenario real planteado por la empresa Antel, la compañía nacional de telecomunicaciones. En este escenario se plantea el diseño de una red de datos MPLS construida sobre una infraestructura de transporte de múltiples tecnologías (SDH y DWDM).**

Puede demostrarse que el problema a tratar es NP-difícil, por lo que se utiliza un enfoque basado plenamente en la aplicación de distintas metaheurísticas para resolverlo. Concretamente, se exploran las técnicas Tabu Search y Algoritmos Genéticos, así como un algoritmo híbrido de ambas. Con el objetivo de mejorar los tiempos de ejecución se experimentó con técnicas de paralelismo sobre el Algoritmo Genético y el híbrido desarrollados. La evaluación de las técnicas exploradas marca una clara ventaja del Tabu Search sobre el Algoritmo Genético y el híbrido implementados.

Palabras clave— **Overlay, Redes, MPLS, Metaheurísticas, Algoritmos Genéticos, Tabu Search.**

I. INTRODUCCIÓN

En este documento se ataca un problema real de diseño de redes presentado por la empresa de telecomunicaciones ANTEL (Administración Nacional de Telecomunicaciones). Este problema es, en pocas palabras, la definición del diseño óptimo de una red de datos virtual de tipo MPLS [1] construida sobre una red de transporte de tecnologías DWDM/SDH [2]. Los planes de ANTEL son la expansión de la red MPLS al interior del país, en base a objetivos comerciales concretos y estimaciones de demanda para el año 2013. La motivación principal es entonces, que la red a diseñar sea óptima en costos, de manera que la empresa pueda cumplir con estos objetivos al menor precio posible. La red de transporte es la misma infraestructura sobre la que funciona la red telefónica de ANTEL y la empresa no tiene planes de modificarla (por lo cual se considerará como fija).

Los costos a considerar en la implementación de la red son básicamente los costos de mantenimiento de la red de transporte. Se considerará que parte de estos costos se trasladan a la red de datos diseñada, por lo que se deberá minimizar el uso de la infraestructura existente por parte de la red a construir.

La complejidad del problema no radica sólo en tener que diseñar una red overlay óptima en costos, sino que además la red deberá cumplir con ciertas restricciones o condiciones de demanda de datos (capacidad) y conectividad. Las restricciones de demanda indican que la red deberá permitir rutear cierta cantidad mínima de datos entre dos nodos diferentes, mientras que las restricciones de conectividad pretenden asegurar que ante un fallo simple de la red de transporte (caída de un enlace de transporte) la red continuara conectada y cumpliendo con las restricciones de demanda.

Puede demostrarse que el problema Minimum Weight Two Connected Network [3] es reducible al problema de optimización que se intenta resolver. El Minimum Weight Two Connected Network Problem se encuentra clasificado dentro de los problemas NP-difíciles [3], lo que hace que el caso de estudio también sea NP-difícil. No se conocen algoritmos que resuelvan los problemas NP-difíciles de manera exacta en tiempo polinomial. Teniendo esto en cuenta, se decide llevar a cabo un trabajo basado en algoritmos aproximados y no en metodologías exactas. En particular, se opta por centrar el trabajo en el desarrollo de diferentes metaheurísticas.

II. PRESENTACION DEL PROBLEMA

El problema a resolver, se puede definir a grandes rasgos como el diseño robusto y óptimo de una red de datos multi-overlay MPLS, sobre una infraestructura de transporte fija.

Ahora bien, para profundizar en el tema, es fundamental tener una noción de lo que es una red overlay. Una red overlay es una red construida "encima" de otra red, que denominaremos red base. Los nodos de dicha red se dice que están conectados por "enlaces virtuales" o "enlaces lógicos". Cada uno de estos enlaces virtuales se corresponde con un camino en la red base, quizá usando varios enlaces de la misma.

El primer overlay en el problema surge de la interacción de dos redes:

- La red de datos
- La red de transporte

La red de datos está construida sobre la red de transporte, de manera que cada enlace de la red de datos se corresponde con un camino de la red de transporte (eventualmente formado por más de un enlace de transporte). Se presenta un ejemplo en la figura 1. Las líneas negras son los enlaces de físicos, o de transporte. Las líneas continuas en color representan enlaces virtuales, que de aquí en más se denominarán enlaces de datos. Las líneas punteadas representan los caminos físicos que usan los enlaces de datos.

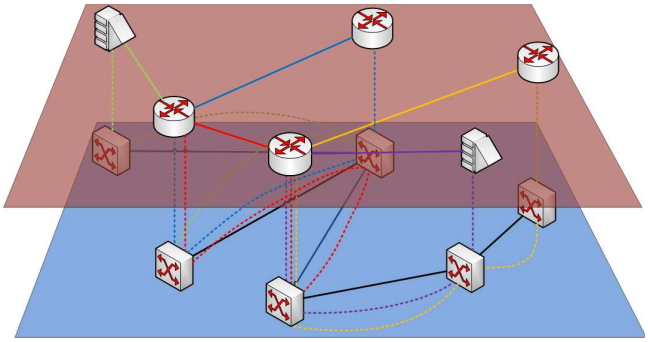

Fig. 1. Una red overlay de dos capas.

La red de transporte es una red fija en el modelo. En otras palabas, las características de la red de transporte son un dato de la realidad y no se puede influir en las mismas. Las tecnologías existentes en esta red para el caso de estudio son SDH y DWDM. Se asume que se cuenta con ambas tecnologías en todos los enlaces de la red de transporte.

A diferencia de la red de transporte, la red de datos es una red a construir y formará parte del problema decidir su topología e interacción con la red de transporte. En términos generales la red de datos es una red MPLS y dichas redes son overlay, ya que funcionan con circuitos virtuales. Por ende, en el problema a tratar existe un segundo overlay, que es dado por la interacción de los enlaces definidos por los circuitos virtuales MPLS y la red de datos en sí. Un circuito virtual en MPLS se denomina LSP (Label Switched Path) [4]. Además, en el contexto de las redes robustas, MPLS permite definir un LSP primario y un LSP secundario en caso de falla [5]. Estos conceptos se pueden apreciar en el ejemplo de la figura 2.

Igual que antes las líneas finas en color representan enlaces de datos. La línea roja continua y gruesa representa un circuito virtual MPLS. El LSP asociado al mismo se representa en rojo punteado.

Teniendo ahora un poco definidas las entidades del problema y sus interacciones se puede ahondar más en el objetivo. En pocas palabras, resolver el problema implica lo siguiente:

- Diseñar la red de datos. Esto implica definir qué enlaces existirán en dicha red y que capacidad tendrán.
- La red de datos debe ser tolerante a fallas simples de la red de transporte. Ante cualquier falla simple todo el tráfico MPLS debe seguir siendo enrutable.
- La red de datos a construir tiene un costo asociado que es la suma del costo de cada enlace de datos. El costo de cada enlace se calcula como el producto entre los kilómetros del camino de mapeo y el costo por kilómetro asociado al ancho de banda al que se dimensionó el enlace. Nos interesa que este costo sea mínimo.
- Se deben definir las rutas que se usarán en MPLS para cada escenario de falla simple en la red de transporte. Si bien para muchos escenarios esto puede no ser muy relevante (ya que MPLS cuenta con un ruteo automático basado en ISIS [6]), para muchos otros sí lo es (ya que el problema de ruteo con restricciones de capacidad es NP-Difícil [7]).

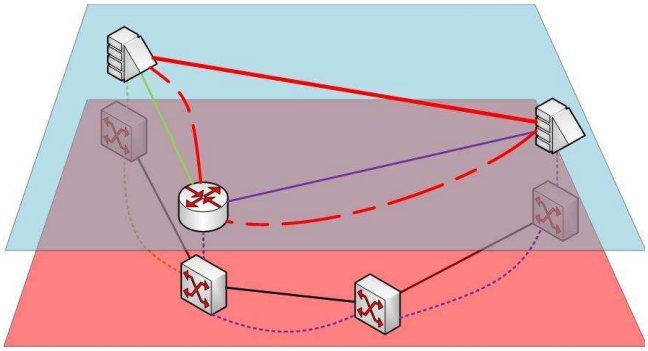

Fig. 2. Una red multicapa.

III. Modelo Matemático

El modelo matemático que se manejó en el trabajo fue extraído de la tesis de maestría de C. Risso [14], en la que se maneja el mismo problema. En este documento se presentará una versión muy simplificada a efectos ilustrativos.

Los siguientes son datos del problema:

- : Representa el grafo de transporte y que será fijo en el modelo.
- : Grafo de elementos candidatos a incluir en la red de datos. Todos las aristas y vértices son opcionales salvo los vértices terminales ( ).
- : Matriz de demanda entre nodos terminales.
- : Un enlace de datos no puede dimensionarse a cualquier capacidad. El conjunto representa el conjunto de capacidades que puede tomar un enlace de datos. Por convención se establece que y se interpreta como no usar el enlace.
- : Devuelve los kilómetros de un conjunto de enlaces de transportes.
- : La dimensión que se le da a un enlace de datos impacta en el costo que el mismo tiene asociado. La función devuelve el costo por kilómetro para una capacidad determinada.

Conociendo los datos presentados, el objetivo es determinar las siguientes entidades:

- dimensión del enlace .
- mapeo a transporte del enlace
- ruteo entre los terminales y para la falla de transporte .

El objetivo es que el costo de la red sea mínimo, lo cual se puede expresar de la siguiente manera:

Está claro que los tres componentes a determinar deben

cumplir una serie de restricciones que no expresarán aquí de manera formal. Las más relevantes de estas restricciones son que para cada falla simple de la red de transporte, ninguno de los enlaces de la red de datos se sobrecargue y que todas las demandas puedan ser satisfechas.

IV. Modelo de ruteo

A. Introducción

A la hora de enrutar en la red MPLS, existen varias maneras de asegurar la tolerancia ante una falla simple de transporte. Existen dos modelos de ruteo que interesa evaluar: el ruteo de camino primario-alternativo y el ruteo de un camino independiente por falla.

La manera más simple de asegurar conectividad ante una falla en un enlace de transporte es asegurar la existencia de un par de caminos disjuntos a nivel de aristas (en datos y transporte) para cada par de nodos edge con demanda. Se denominará al primero de estos caminos como "camino primario". Este camino es el que se utilizará en un caso de funcionamiento normal de la red. El segundo camino, o camino alternativo, se utilizará solo en caso de que alguno de los enlaces de transporte de los que depende el camino primario falle.

El ruteo de camino primario-alternativo, si bien permite la implementación de una amplia variedad de soluciones, para algunos escenarios existen soluciones factibles que no pueden ser modeladas. La alternativa es que para cada escenario de falla se rediseñen todos los caminos de la red. En esto se basa el modelo de n-caminos (que también se denominará como modelo de un camino por falla).

La ventaja sustantiva que tiene el modelo de n-caminos frente al de dos, es que permite generar soluciones que con dos caminos no son modelables. Existe la posibilidad de encontrar mejores soluciones con este modelo, dado que en el mismo se trabaja sobre un universo más amplio. Por otro lado, una posible consecuencia es que cualquier algoritmo de búsqueda (metaheurística o no) funcione de manera menos eficiente.

Otro punto interesante a tener en cuenta cuando se comparan ambos modelos de ruteo, es el costo de implementación de una solución generada con los mismos. El ruteo que manejarán los algoritmos desarrollados puede o no implementarse de manera directa en la red. Una posibilidad es que en la implementación se utilicen algoritmos de ruteo automático (como ISIS) sobre la red de datos definida. Sin embargo, recordemos que el problema de ruteo con capacidades es NP-difícil [7]. Por ende, existen escenarios factibles que no pueden ser manejados con algoritmos como el ISIS. Para esos escenarios la única alternativa es definir de manera manual el ruteo de la red.

En resumen, la implementación automática del ruteo puede ser imposible en determinados escenarios. Para estos escenarios, se puede usar la información de ruteo almacenada en la representación. Sin embargo, hay que tener en cuenta que esto añade un costo en horas hombre en la programación de los routers. Si bien este costo no está estimado, se sabe que el modelo de un "camino por falla" tiene costos de implementación manuales por mucho mayores al del "camino primario – alternativo".

B. Evaluación del modelo de ruteo

Considerando las diferencias existentes entre ambos modelos de ruteo se decidió realizar un estudio para medir el "sesgo" del modelo de camino-primario alternativo. Con el término "sesgo" se pretende hacer referencia a la incapacidad de este modelo de manejar determinadas soluciones. En particular se espera obtener el "sesgo al óptimo". Este se define como la distancia existente entre el óptimo que se puede obtener con el modelo de dos caminos y el óptimo real (que sí se puede obtener con el modelo de "un camino por falla"). Se considera que esta métrica es fundamental a la hora de escoger entre los dos modelos. Si el sesgo al óptimo es pequeño, la opción predilecta sería el modelo de camino primario-alternativo, ya que sus costos de implementación deberían ser inferiores. En cambio, si el sesgo al óptimo es muy grande, sería tentadora la elección del modelo de n-caminos.

La primera dificultad que se encuentra es que el sesgo que se intenta medir en el modelo de dos caminos depende del problema. Sin embargo, si se logra medir el sesgo en una gran diversidad de problemas, es de esperar que se obtenga una buena métrica de cómo se comporta en el caso promedio.

En otras palabras, para analizar cuál de los modelos (dos o N caminos) es el más aconsejable para usar, se deben comparar las soluciones optimas obtenidas con cada modelo para un conjunto de problemas de prueba. Para encontrar la solución óptima global es necesario utilizar un algoritmo exacto, por lo que se diseñaron e implementaron algoritmos de backtracking que devuelven la solución óptima. Se realizaron dos algoritmos, uno para cada modelo.

Dado que el tiempo de ejecución de los algoritmos de backtraking crece muy rápidamente con el tamaño del problema, se utilizaron escenarios pequeños (de cuatro a seis estaciones) para realizar la evaluación. Estas instancias fueron generadas con un algoritmo de generación automática de escenarios diseñado por el equipo de trabajo. Con el fin de obtener una buena medición del sesgo y no caer en las particularidades de algunos escenarios, se generaron cinco problemas distintos para cada número de estaciones, por lo que se evaluaron quince escenarios en total.

Los resultados que se muestran en la tabla 1 reflejan que los costos obtenidos por el modelo de N caminos son iguales o mejores que los obtenidos por la representación de 2 caminos. Esto es debido a que (como se mencionó anteriormente) el espacio de búsqueda en N caminos es mucho más grande existiendo la posibilidad de encontrar soluciones de mejor calidad. Sin embargo la falencia que tiene el modelo de N caminos es el costo computacional que comprende explorar el espacio de búsqueda. Si se compara el tiempo de cómputo entre ambos modelos, se puede apreciar que para el caso de camino primario-alternativo es del orden de décimas de segundo y para el caso de N caminos llega a ser incluso días. Se observa que hay un gap no despreciable (9,39% de

Modelo de 2 caminos			
Estaciones	Escenario	Tiempo (seg)	Costo
4	4_0	0	15098,10
4	4_1	0	15517,00
4	4_2	0,01	7876,50
4	4_3	0,01	8684,80
4	4_4	No factible	No factible
5	5_0	0,15	8006,30
5	5_1	0,09	16419,70
5	5_2	0,07	23476,10
5	5_3	0,5	27499,90
5	5_4	0,01	14248,50
6	6_0	0,42	17192,60
6	6_1	0,03	19841,70
6	6_2	0,26	12950,50
6	6_3	1,13	27635,60
6	6_4	0,06	15723,50

Modelo de N caminos			
Estaciones	Escenario	Tiempo (seg)	Costo
4	0	0,01	15098,10
4	1	0,01	11516,8
4	2	0,01	7876,5
4	3	0,01	8684,8
4	4	No factible	No factible
5	0	mayor a 48hs	No finalizo
5	1	1861,42	15635
5	2	1657,38	15871
5	3	507,378	18974,4
5	4	0,01	14248,5
6	0	mayor a 48hs	No finalizo
6	1	0,33	19841,7
6	2	1969,25	12950,5
6	3	mayor a 48hs	No finalizo
6	4	mayor a 48hs	No finalizo

	Igual costo
	Costo (2 caminos) > Costo(N caminos)

Tabla 1: Resultados evaluación modelos de ruteo

promedio) entre el espacio de búsqueda de dos caminos y el de N caminos, lo que apunta a que se elija este último para el desarrollo de los algoritmos. Sin embargo, se deben tener en cuenta los siguientes factores:

- Una solución de tipo N caminos puede resultar difícil de implementar en la práctica.
- El tiempo de cómputo de un algoritmo que corra en el espacio de N caminos debería ser mucho mayor que uno que corra en el espacio de dos caminos.

Debido a esto se optó finalmente por utilizar el modelo de dos caminos en el desarrollo de los algoritmos

Generar la solución inicial s

$mejorSol = s$

Mientras NO se cumpla la condición de parada, **Hacer**

Generar el conjunto V^* de soluciones $s_i = s \oplus m_i$ / $m_i \notin LT$ *o bien* s_i satisface el criterio de aspiración.

Elegir la mejor solución $s' \in V^*$ (respecto a la función f a optimizar)

Actualizar LT con m_i

$s = s'$

Si $f(s') < f(mejorSol)$, **entonces** $mejorSol = s'$

Cada "cierto tiempo", intensificar y/o diversificar

Fin Mientras

Fig. 3. Pseudocódigo de la búsqueda tabú

V. METAHEURISTICAS

A. *Tabu Search*

Se trata de una técnica relativamente simple y se la considera una metaheurística basada en trayectoria [8]. La idea básica es mantener un historial no muy extenso indicando las soluciones que el algoritmo ya visitó en el pasado. Este historial es usado como una lista "tabú" y su objetivo es evitar que el algoritmo visite varias veces una misma solución e incluso que se explore más de una vez un mismo vecindario. Se muestra un pseudocódigo de esta técnica en la figura 3.

Dado que el almacenamiento de todas las soluciones que el algoritmo ya visitó puede resultar sumamente costoso, se suelen codificar por ejemplo las características básicas de los vecindarios para luego poder filtrar y evitar que se visiten nuevamente. En cada iteración se toma la mejor solución del vecindario que no esté en la lista tabú. Sin embargo, existen excepciones a esta regla que están dadas por los denominados criterios de aspiración. Si una solución del vecindario está en la lista tabú pero verifica un criterio de aspiración, entonces es tenida en cuenta a la hora de elegir la siguiente solución.

Con frecuencia se añaden al funcionamiento básico del algoritmo procesos de intensificación y/o diversificación. La idea de los procesos de intensificación es explotar las características de las mejores soluciones encontradas hasta el momento. Por el contrario, los procesos de diversificación buscan explorar soluciones o lugares del espacio de búsqueda aún no recorridos.

B. *Algoritmos Genéticos*

Los algoritmos genéticos son una técnica basada en población e inspiradas en la teoría de la evolución natural [9] [10]. Esta técnica trabaja iterativamente sobre una población de soluciones o individuos que son inicializados generalmente de forma aleatoria (aunque pueden usarse por ejemplo técnicas constructivas como Greedy para obtener una solución inicial).

Inicio de algoritmo

Generar una población inicial

Calcular fitness de cada individuo

Repetir

Para (Tamaño población) / 2

Seleccionar dos individuos de la anterior generación para el cruce

Cruzar los dos individuos con cierta probabilidad para obtener dos hijos

Mutar los dos descendientes con cierta probabilidad

Calcular la función de fitness para ambos individuos

Insertarlos en la nueva generación

Fin

Hasta (Condiciones de terminación)

Fin algoritmo.

Fig. 4. Pseudocódigo de un algoritmo genético

Cada iteración comprende tres fases principales: Selección, Reproducción (y mutación) y Reemplazo.

La selección es el proceso por el cual se eligen los individuos de la población para ser reproducidos. Esta selección se realiza teniendo en cuenta una función de aptitud o función de fitness que da la idea de cuan apto es un individuo para resolver el problema. Algunos de los tipos de selección más conocidos son selección proporcional, selección por ranking y selección por torneo [11]. Los individuos seleccionados en el proceso anterior son combinados para formar los nuevos individuos de la próxima generación. Luego de dicha reproducción se deben seleccionar cuales serán los individuos que formaran parte de la nueva generación. Esta nueva generación puede contener individuos de la población anterior dependiendo del algoritmo de reemplazo. Este proceso se ilustra en el pseudocódigo de la figura 4.

VI. METAHEURÍSTICAS APLICADAS AL PROBLEMA

A. Algoritmo Genético

En esta sección se desarrollará el diseño del AG propuesto. Para la especificación completa del diseño del AG es necesario definir ciertos componentes:

- Representación de la soluciones
- Función de fitness
- Selección
- Cruzamiento
- Mutación

Dadas las características del problema manejado, los operadores evolutivos no son necesariamente correspondidos de forma directa con los del algoritmo genético simple. Además de los operadores evolutivos de selección y cruzamiento, se presentan tres mutaciones aleatorias que modifican caminos en datos y en transporte. Adicionalmente,

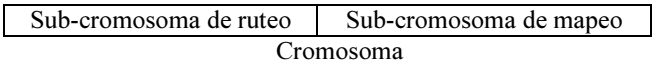

Fig. 5. Estructura cromosoma

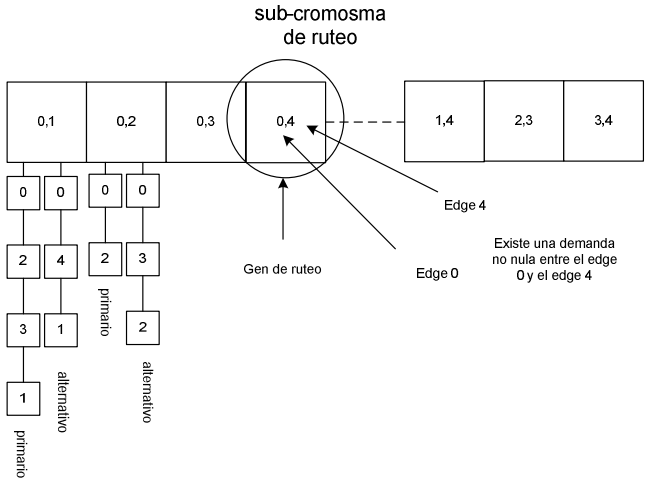

Fig. 6. Subcromosoma de ruteo

con el fin de mejorar la eficacia del algoritmo, se introducen dos operadores que funcionan como búsquedas locales.

1) Selección

Para definir cualquier operador de selección en un algoritmo genético, es indispensable definir algún tipo de función de fitness. Considerando que el problema a tratar es un problema de minimización del costo de una red, surge como opción natural usar el inverso del mismo como función de fitness. Para poder obtener valores de fitness para el caso de costo cero, se propone una variante en la que se le suma uno al denominador de la misma.

$$f(x) = \frac{1}{C(x) + 1}$$

Es razonable creer que cualquiera de los operadores de selección tradicionales son buenos candidatos para aplicar en el problema. Entre estos se optó por el operador de selección de ruleta, por ser el más utilizado en la práctica.

2) Representación de las soluciones

Definir una codificación o representación de la solución es necesario para la implementación de algunas metaheurísticas, como algoritmos genéticos. Las representaciones tradicionales para algoritmos genéticos, basadas en cadenas de símbolos, se consideran inapropiadas para este problema. Esta consideración se basa en el hecho de que diseñar operadores que estén basados en este tipo de representaciones y además mantengan la factibilidad de las soluciones es una tarea muy compleja en este problema.

La representación que se eligió para el modelo de camino primario alternativo se basa en definir la solución de manera implícita a través de estas rutas. Esta idea está inspirada en el proyecto de grado de Calegari [12].

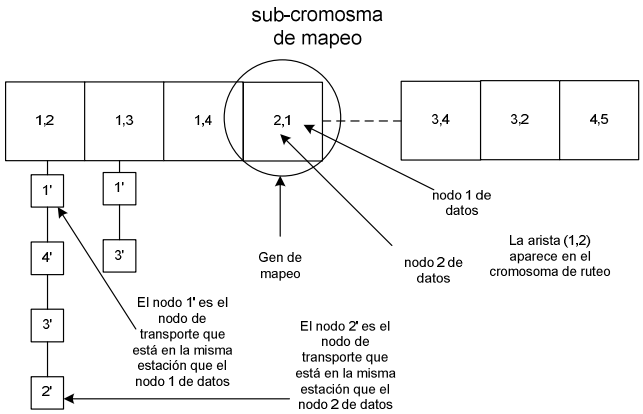

Fig. 7. Subcromosoma de mapeo

La representación de dos caminos (o de camino primario-alternativo) está compuesta de dos subcomponentes: el componente de ruteo (basado en listas de caminos) y el componente de mapeo. A la dupla formada por ambos se le denominará cromosoma, de manera de utilizar la terminología propia de algoritmos genéticos.

El componente de ruteo, o sub-cromosoma de ruteo, está conformado por una lista de elementos que denominaremos genes de ruteo. Cada uno de estos genes está asociado a un par de nodos edge con demanda no nula entre sí. La demanda asociada a un par de edges se deriva de los accesos conectados a los mismos.

A su vez cada uno de estos genes de ruteo tiene dos sub-componentes: el camino primario y el camino alternativo. Tanto el camino primario, como el alternativo son listas de nodos de datos. El camino primario representa el camino que será utilizado en todos los escenarios, salvo en los que la falla de transporte se encuentre sobre este camino. Para esos casos, se utilizará el camino secundario.

Superponiendo todos los caminos de todos los genes del sub-cromosoma de ruteo, se obtiene la red de datos. La capacidad de las aristas se calcula tomando el mínimo ancho de banda disponible que soporte el máximo flujo de datos alcanzado para todos los escenarios (es decir, para todas las fallas de transporte posibles).

El componente de mapeo, o sub-cromosoma de mapeo, se encarga de definir el mapeo a transporte de cada una de las aristas de la solución (definidas implícitamente por el sub-cromosoma de ruteo). Todo gen del sub-cromosoma de mapeo está asociado a una arista de datos ad presente en la solución (definida implícitamente por el sub-cromosoma de ruteo) y contiene una lista ordenada de nodos de transporte, que representa el mapeo a transporte de esa arista ad.

Con la unión de ambos componentes se puede obtener toda la información sobre la solución:

3) Generación de soluciones factibles

Se desarrolló un algoritmo capaz de crear y reparar soluciones factibles que será utilizado a lo largo de este trabajo y en particular tiene la función de actuar como mecanismo de inicialización del algoritmo genético. El algoritmo desarrollado es del tipo greedy aleatorio, aunque esto no debe entenderse de forma estricta, ya que el mismo puede deshacer partes ya construidas (lo que no ocurre en un enfoque greedy aleatorio tradicional).

```
genesRuteo = genes de ruteo en orden de demanda decreciente
Para k = 0 hasta (cantidad_genes – 1)
genRuteo = genesRuteo[k]
Si genRuteo no está completo entonces
        Intento = 0
        Mientras (genRuteo no esté completo) && intento < maxIntentos
                intentarConstruirGen(genRuteo)
                Intento = intento + 1
        Fin mientras
        Si genRuteo no está completo entonces
                vaciarUnGenProblematico
                k = 0
        Fin si
Fin si
Fin para
```

Fig. 8. Pseudocódigo de la generación de soluciones factibles

El esqueleto del algoritmo es, a grandes rasgos, un bucle donde en cada iteración se intenta generar un gen de ruteo. La construcción de los genes de mapeo es subordinada a los de ruteo. En otras palabras, la construcción de un gen de ruteo invoca la generación de todos los genes de mapeo que necesita. Es importante notar que los genes se eligen siempre en orden de demanda decreciente, por lo que primero son generados los genes con mayor demanda. Esto es debido a que rutear genes con poca demanda asociada en una red con poca capacidad remanente es mucho más fácil que hacerlo con genes de mayor demanda.

La generación de un gen de ruteo puede fallar por diversas causas. Una de las mismas puede ser que la capacidad de los enlaces disponibles no pueden rutear la demanda necesaria, o porque ninguno de estos se puede mapear en transporte. Cada gen se intenta construir un número determinado de veces. Si se agota el número máximo de intentos se analiza qué genes de ruteo ya construidos impiden la construcción del nuevo gen. Este conjunto se denominará de aquí en más “genes problemáticos”.

No se encontró un mecanismo para determinar este conjunto de forma exacta, por lo que se diseño una heurística que lo determina de forma aproximada. La heurística desarrollada se puede resumir en los siguientes pasos:

- Si al momento de escoger el siguiente enlace de datos durante el proceso de construcción de un gen, ninguno de los mismos soporta la capacidad de la demanda a rutear, se identifican los genes que utilizan estos enlaces como posibles genes problemáticos.
- En caso de no haberse identificado ningún posible gen

problemático durante el proceso de construcción, se considera que todos los genes de la solución son problemáticos.

- Luego que el conjunto de posibles genes problemáticos queda definido, se elige algún gen problemático de manera aleatoria y se lo "vacía". Cuando se "vacía" un gen de ruteo, se eliminan los caminos primarios y secundarios. Además, también se eliminan los genes de mapeo asociados a enlaces que solo aparecían en el gen de ruteo vaciado, en caso de que existan. Este proceso se repite hasta que la construcción del gen es terminada.

En la figura 8 se muestra un pseudocódigo del esqueleto del algoritmo, donde se ilustran los conceptos presentados. Como se puede apreciar, una vez que la construcción de un gen de ruteo falla y se elimina un gen problemático, se resetea la variable k. Este comportamiento asegura que los primeros k genes de mayor demanda siempre están construidos.

Cuando el algoritmo tiene que agregar una nueva arista a la solución, lo hace sorteando entre una lista de candidatos. En dicho sorteo cada arista tiene una cierta probabilidad de ser seleccionada, esta probabilidad depende directamente de una función de aptitud. Para las aristas de datos se le da más probabilidad a las aristas que aportan menor costo a la solución, mientras que para las aristas de transporte se le da más probabilidad a las aristas que minimizan los kilómetros al destino.

4) Cruzamiento

Dada la codificación del problema no es posible utilizar los operadores de cruzamiento tradicionales por lo que se realizó el diseño de un operador de cruzamiento específico.

A diferencia de los operadores de cruzamiento tradicionales, donde dos padres dan lugar a dos hijos, el operador de cruzamiento diseñado obtiene un hijo de un conjunto de N padres. Este diseño se debe a que durante el cruzamiento surgen ciertas inconsistencias que dejan al hijo incompleto (el material genético restante se debe regenerar). Usando múltiples padres, en lugar de dos, se minimizan dichas inconsistencias. Esto se verá en detalle más adelante.

El individuo hijo se genera en etapas, donde cada etapa consiste en agregar un gen de ruteo a dicho individuo. Para cada gen que se agrega, se selecciona uno de los padres al azar y se copia el contenido del gen en cuestión del padre al hijo. Agregar este nuevo gen de ruteo al hijo implica agregar todos los genes de mapeo que estén asociados al mismo y no estén presentes en el individuo hijo. Este procedimiento de construcción por partes de un individuo se continúa hasta que todos los genes del cromosoma de ruteo se hayan copiado.

Antes de agregar un nuevo gen de ruteo se debe verificar primero que el camino primario y secundario sigan siendo disjuntos en transporte. Puede existir una intersección de ambos en el nuevo individuo, ya que ciertos mapeos de aristas en el cromosoma hijo (agregados en pasos anteriores) puede que sean distintos a los del padre. Además del chequeo anterior se debe realizar un chequeo de capacidad. Puede suceder que al agregar los caminos de un gen de ruteo exista una arista que ya fue utilizada por algún gen anterior, por ende

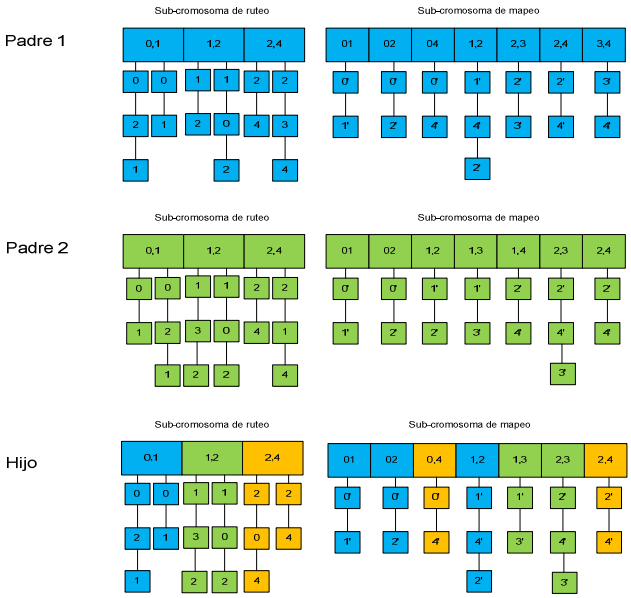

Fig. 9. Ejemplo de un posible cruzamiento

al agregarle la demanda del nuevo gen el enlace podría quedar extra saturado.

Ambos problemas pueden llegar a impedir que un gen de ruteo sea agregado al nuevo individuo. Para intentar resolver esto se ideo se sortea un nuevo padre del cual extraer el gen en caso de que aparezca alguna inconsistencia. El procedimiento se repite hasta lograr agregar dicho gen sin problemas o hasta alcanzar un límite de iteraciones predefinido.

Si finalmente no se pudo agregar el gen, el cromosoma de ruteo queda incompleto y la solución es no factible. En este caso se utiliza el método greedy aleatorio de reconstrucción de soluciones descrito en la sección "Generación de soluciones factibles", para reparar la solución.

Este comportamiento es ilustrado en la figura 9, donde se cruzan dos padres para generar un individuo. Los colores del hijo muestran de qué padre vino cada gen. El gen de ruteo [2,4] aparece en naranja porque no pudo ser insertado desde ninguno de los dos padres (los caminos primario y alternativo no son disjuntos en el hijo). El mismo fue generado por el algoritmo de reparación, lo que conllevó la generación de los genes de ruteo [0,4] y [2,4].

5) Mutación de datos

Esta es una mutación aleatoria que reconstruye un gen al azar del cromosoma de ruteo. Como su nombre lo indica, esta mutación tiende a modificar los caminos en la capa de datos, sin modificar mapeos de los enlaces ya existentes. Sin embargo, es posible que el mapeo de un enlace cambie si el mismo sólo es usado por el gen que se reconstruye.

El funcionamiento de la misma es bastante simple. Básicamente selecciona de forma aleatoria un gen del cromosoma de ruteo y lo vacía. Este proceso de vaciado elimina también los genes de mapeo cuyo enlace asociado ya no es usado por ningún gen de ruteo. A continuación reconstruye tanto el camino primario como el secundario desde cero. Para completar la solución se invoca un método

greedy aleatorio de reconstrucción que intentara encontrar un par de caminos que nuevamente hagan factible a la solución. Este método es explicado en detalle en la sección de "Generación de soluciones factibles".

6) Mutación de transporte

Aunque también es una mutación aleatoria como la de datos, a diferencia de ésta su objetivo es cambiar parcialmente el mapeo de uno o más enlaces. El operador selecciona con probabilidad uniforme un conjunto de genes de mapeo y a cada uno intenta cambiarle solo una porción del mapeo.

7) Mutación enlace tabú

Dada una solución se selecciona al azar un enlace de datos y es identificado como tabú durante el proceso de mutación. La idea consiste en eliminar dicho enlace de la solución junto con todos los genes asociados. Hecho esto tenemos que el individuo ahora se encuentra incompleto, por lo que debe ser reconstruido. Para esto es utilizado el método greedy aleatorio de reconstrucción expuesto en la sección "Generación de soluciones factibles". En proceso de reparación se intenta reconstruir la solución pero sin utilizar en ningún momento el enlace eliminado (de ahí el nombre de enlace tabú).

8) Mejor mutación datos

Este operador cae en el grupo de búsqueda local, dado que intenta obtener la mejor configuración vecina dentro de lo posible. No es estrictamente una búsqueda local ya que no siempre devuelve el mejor individuo dentro de una vecindad, sino el mejor individuo de entre N vecinos.

El funcionamiento es muy simple: se modifica una cantidad de veces el individuo actual usando la Mutación Datos y se devuelve la mejor solución encontrada.

9) Mejor mutación transporte

El funcionamiento es muy similar que el de la Mejor Mutación Datos, solo que en este caso la vecindad está determinada por la Mutación Transporte. Al igual que en el caso anterior se exploran N soluciones vecinas a la actual usando la Mutación Transporte y se devuelve la mejor.

B. Tabu Search

Como un camino alternativo al algoritmo evolutivo y con la intención de poder comparar resultados es que se optó por llevar a cabo la instanciación de un Tabu Search.

El diseño de este algoritmo reutiliza en gran parte lo construido para el AG. Para poner un ejemplo, se reutiliza la representación de la solución como mecanismo base para la manipulación de soluciones. Adicionalmente se tiene que también fueron diseñados algoritmos de mutación y búsquedas locales para perturbar soluciones halladas. Estos algoritmos pueden ser usados sin mayor adaptación como algoritmos de búsqueda local en un Tabu Search.

La búsqueda local elegida para ser usada en cada iteración del tabú fue la Mejor Mutación Datos. Esta decisión se basa en que dicha mutación puede potencialmente modificar cualquier componente de la solución.

El diseño del Tabu Search fue considerablemente sencillo en dos sentidos. Por un lado, se tiene que la construcción de esta metaheurística tanto en la etapa de diseño como de desarrollo no insumió demasiado esfuerzo. Esto es justamente porque la mayoría del trabajo ya había sido pensado, diseñado y desarrollado para el algoritmo genético. Por otro lado, se decidió crear una versión realmente simple del Tabu Search disminuyendo el trabajo adicional de su inclusión en el proyecto.

Otro punto clave a tratar es la identificación de las soluciones dentro de la lista tabú. Durante el transcurso de la ejecución de esta técnica metaheurística es necesario verificar casi constantemente si una solución pertenece al historial recorrido. Generalmente no se suelen almacenar las soluciones visitadas en la lista tabú, sino que se crean "claves" o "identificadores" de cada una o incluso de su vecindario para hacer más simple esta verificación. En el presente trabajo se hizo una codificación de las soluciones que las identifica por completo. No es más que una cadena de caracteres conteniendo datos sobre qué caminos usa cada pareja de nodos, mapeos a transporte, etc.

C. Algoritmo híbrido

Con la idea de obtener una metaheurística que combine las fortalezas del algoritmo genético y la búsqueda tabú diseñadas, se decidió crear una metaheurística híbrida. En el algoritmo híbrido, la búsqueda tabú queda sometida al algoritmo genético, insertándose en el mismo como un nuevo operador evolutivo.

El principal inconveniente de hacer esta incorporación es la pérdida de performance en la ejecución del algoritmo. De cualquier manera, el tiempo de cómputo invertido en el operador de búsqueda tabú puede ser regulado con la probabilidad de ejecución del mismo. Con esta idea, se calibraron los parámetros del operador de manera que permitieran una exploración importante para un número reducido de soluciones. Finalmente, se fijaron el número de iteraciones y el tamaño de la vecindad en las constantes 60 y 40 respectivamente.

VII. ANALISIS EXPERIMENTAL

En esta sección se persiguen dos objetivos: la correcta calibración de los parámetros de las metaheurísticas, así como la comparación de la calidad de los resultados entre las distintas técnicas presentadas.

A. Plataforma de ejecución y desarrollo

Los algoritmos presentados en la sección anterior fueron implementados en C++, utilizando la versión 4.1.2 del compilador estándar GNU. Para el caso de los algoritmos genéticos, se utilizó la biblioteca de esqueletos MALLBA como base para la implementación. La biblioteca MALLBA resuelve la comunicación entre procesos de manera transparente utilizando la tecnología MPI. En este caso, se

utilizó MPICH en su versión 1.2.7 como implementación de MPI.

Se utilizó una plataforma distinta para cada etapa de la evaluación experimental. Para la fase de parametrización, se utilizaron equipos con las siguientes características:

- Intel(R) Pentium(R) D CPU 2.80GHz
- 1 GB memoria RAM
- Fedora Core 10 (Linux Kernel 2.6.27.5-117.fc10.x86_64)

En el resto de la evaluación experimental, se realizaron ejecuciones sobre el cluster de Facultad de Ingeniería, UdelaR. Si bien la arquitectura completa de este cluster no es uniforme [13], la estructura principal sí lo es y en este trabajo solo se utilizaron algunos nodos de la misma. La estructura principal del cluster está formada por 9 nodos con las siguientes características:

- 2 chips Quad core Xeon E5430, 2x6 MB caché, 2.66 GHz, 1.333 MHz FSB
- 8 GB memeria RAM
- Arquitectura 64 bits
- Adaptador de red dual (2 puertos Gigabit Ethernet)

B. Instancias de prueba

Para cualquier evaluación experimental de un algoritmo se necesita un conjunto de escenarios de prueba, sobre el que se pueda extraer conclusiones. A estos efectos, se toma como base el conjunto de prueba utilizado por C. Risso [14], compuesto por diferentes escenarios a futuro para la red MPLS de la empresa ANTEL.

Aunque el principal interés en este trabajo es la evaluación de los algoritmos usando el conjunto de prueba presentado por C. Risso, el mismo resulta sumamente inapropiado para la calibración de los algoritmos. También dificulta otros experimentos realizados a los largo del documento, como la evaluación del operador de cruzamiento. Para esta etapa es más apropiado un conjunto más heterogéneo de escenarios, de manera que la calibración de los algoritmos sea aplicable a un conjunto de problemas lo más amplio posible. Además, aún sin tener este objetivo a la vista, la calibración de los algoritmos utilizando estos escenarios sería imposible desde un punto de vista práctico, ya que requeriría un inmenso tiempo de cómputo dada la gran complejidad de los mismos. Con esta idea en mente, se construyó un conjunto de escenarios propio para usar en estas etapas.

C. Generador de escenarios

Para automatizar la generación de instancias de prueba (escenarios de aquí en más), se diseñó e implementó un algoritmo que hace esta tarea en forma aleatoria en base a ciertos parámetros de probabilidad y tamaño de la red. Llamaremos a este algoritmo de aquí en adelante "generador de escenarios consistentes".

Es necesario tener en cuenta que para que un escenario sea factible debe cumplir diversas condiciones. Además de las condiciones de factibilidad, nos interesa que los escenarios cumplan una serie de requisitos para que los mismos presenten características de posibles escenarios reales. Por ejemplo, se prohíbe la generación de redes donde el largo de algún enlace de transporte sobrepase cierto valor.

Algunas otras condiciones que se impusieron son:

- Una estación debe tener algún nodo más aparte del nodo de transporte (nodo de datos o nodo terminal).
- El largo de los enlaces de transporte debe ser menor a una constante dada. Llamaremos MAX_LARGO_ENLACE a dicha constante.
- Toda estación debe tener dos estaciones a distancia menor que MAX_LARGO_ENLACE.

Escenario	#Nodos de datos	# Requisitos de tráfico	# Aristas de transporte	# Aristas de datos candidatas
1	80	78	214	365
2	56	58	164	286
3	80	107	214	365
4	56	58	164	286
5	68	78	190	341
6	56	58	164	286
7	68	107	190	341
8	56	58	164	286
9	68	107	190	341
10	56	58	164	286
11	68	107	190	341
12	56	58	164	286

Tabla 2: Características de escenarios de ANTEL

D. Escenarios de ANTEL

La principal motivación del proyecto, como se planteó en la introducción, es la optimización de la topología de la red MPLS que la empresa ANTEL espera implementar para cumplir con una serie de compromisos. En la actualidad, sólo la red de transporte es un dato completamente definido de este problema. En el trabajo realizado por C. Risso [14], se examinan una serie de posibles escenarios distintos para la red de datos, en base a la variación de los requerimientos que no están completamente determinados. En el presente trabajo se toma como base esta serie de escenarios para realizar la evaluación experimental. Al mismo tiempo, esto también ofrece un punto de referencia para comparar los resultados obtenidos por las metaheurísticas con los resultados obtenidos en la tesis de C. Risso.

Los escenarios de C. Risso tienen un orden de 130 a 159 estaciones, donde entre 56 y 79 de las mismas poseen nodos de datos. Estos escenarios están clasificados según cuatro variables binarias, de las cuales se entiende que ANTEL tiene mayor control (pueden tener por ende influencia en decisiones estratégicas de la empresa) y que son fundamentales para el backbone IP/MPLS.

La arquitectura es quizás la variable más influyente al momento de determinar la topología de la red. Se consideran entonces dos tipos de escenarios en este sentido. En los escenarios de tipo napL (escenarios pares) se mantiene la

Operador	Probabilidad
Cruzamiento	60%
Mutación datos	1%
Mutación transporte	1%
Mejor mutación Transporte	50%
Mejor mutación Datos	90%
Mutación enlace tabú	10%
Mutación tabú	5%

Tabla 3: Configuración utilizada en la parametrización del tamaño de población

arquitectura actual. Esto significa tunelizar el trafico IP del interior hacia la red IP Pública, la cual tiene presencia sólo en el área metropolitana de Montevideo (AMM). Por el contrario, la arquitectura napH (escenarios impares) comprende el extender la red IP Pública hacia el interior evitando que el tráfico internacional pase necesariamente por el AMM. Intuitivamente el costo de la arquitectura napH es menor que el de la napL dado que disminuiría el uso de la infraestructura de transporte. Sin embargo el cambio a esta arquitectura acarrearía un costo operativo importante, por lo que el cambio debe estar bien justificado.

En la **Error! Reference source not found.** 2 se muestran las características principales de cada uno de los escenarios evaluados. Vale recalcar que los datos mostrados por la tabla incluyen el procesamiento de los escenarios que se menciona en el párrafo anterior. El número de requisitos de tráfico hace referencia a la cantidad de pares de nodos de datos con demanda asociada.

E. Calibración del algoritmo híbrido

La calibración de los algoritmos estocásticos en general es una tarea importante, ya que permite obtener el máximo rendimiento de la técnica implementada. Un proceso de calibración formal puede exigir tiempo y recursos computacionales importantes, por lo que sólo se invirtió este tiempo para el algoritmo híbrido desarrollado. El resto de las técnicas presentadas se calibraron de manera informal y los parámetros escogidos para las mismas se presentan en la siguiente sección.

Pensando en la calidad de la calibración se decidió realizar la misma mediante varios escenarios relativamente pequeños y con diferentes características. De esta manera se puede contar con escenarios representativos de una gran variedad de problemas. Esto permite configurar el algoritmo para que generalmente obtenga buenos resultados para diferentes problemas en vez de especializarse en un solo escenario.

Para la ejecución fue necesario generar un número elevado de escenarios de prueba con diferentes densidades y variando entre 12 y 25 nodos cada uno. Del total de escenarios generados se tomaron 6 diferentes con alrededor de 15 nodos (se decidió no utilizar escenarios más grandes porque no aportaría demasiado para la calibración y sin embargo enlentecería el proceso). Para cada escenario se realizaron 10 ejecuciones para poder tomar un estimativo mas real sobre el tiempo y el fitness que obtiene cada combinación de parámetros.

	Grupo 1			Grupo 2	
	Mutación Datos	Mutación Transporte	Mutación Enlace tabú	Mejor Mutación Datos	Mejor Mutación Transporte
C1	1%	1%	1%	90%	50%
C2	1%	1%	1%	85%	30%
C3	5%	5%	5%	90%	50%
C4	5%	5%	5%	85%	30%

Tabla 4: Posibles configuraciones de operadores a elegir

Parámetro	Valor
Cantidad de islas	4
Población de cada isla	13
Cantidad de hijos por generación/isla	39
Modelo de migración	Sincrónico
Ejecutar migración cada	3 iteraciones
Número de individuos a migrar	1
Operador de selección	Selección de torneo
Tamaño del torneo de selección	3
Operador de reemplazo	Reemplazo por torneo
Tamaño del torneo de reemplazo	5

Tabla 5: Configuración modelo de migración

Se decidió hacer la calibración de parámetros en dos etapas diferentes. En una primera fase se define la cantidad de individuos y en la siguiente las probabilidades de los operadores de mutación. A pesar de que la calibración de los operadores y del tamaño de población no es independiente, fue necesario considerarlas de esta forma para reducir el número de ejecuciones.

Se decidió utilizar tres valores diferentes para el tamaño de la población, variando entre 30, 50 y 70 individuos. Se determinó que no es de mayor aporte utilizar más de 70 individuos ni utilizar menos de 30 (dado que es el rango de valores generalmente utilizados). Las probabilidades usadas para los operadores evolutivos en la calibración del tamaño de población se presentan en la tabla 3. El caso que obtuvo mejores resultados fue el de 50 individuos. Con 70 no se observan mejoras en el fitness a pesar de que hubo un incremento en el tiempo de cómputo, y con 30 individuos se obtiene una mejora en el tiempo de ejecución pero no se logran buenos resultados en el fitness.

En la segunda etapa de la calibración, en vez de tomar los operadores de manera independiente, se decidió ligarlos en subgrupos que varían de manera conjunta. Aunque esto reduce la calidad de la calibración, la convierte en un procedimiento viable, debido a la mejora del tiempo de ejecución.

Los grupos en los que se separan los operadores están formados por las mutaciones aleatorias (mutación datos, mutación transporte y mutación enlace tabú) y las mutaciones de búsqueda local (mejor mutación datos y mejor mutación transporte). El cruzamiento y la mutación tabú se mantuvieron estáticos en 60% y 0.5% de probabilidad respectivamente.

Para cada grupo se definieron sólo dos valores posibles, estableciendo así cuatro casos de prueba en vez de tres como en la primera parte. En la tabla 4 se muestran los valores escogidos para cada caso de estudio.

Se hicieron 10 ejecuciones independientes por cada caso y escenario. Los mejores resultados fueron obtenidos por el caso 1, por lo que se uso esta configuración para la evaluación final

Parámetro	Valor
Tamaño del vecindario	50
Tamaño lista tabú	100
Cantidad de iteraciones	5000

Tabla 6: Configuración final del Tabú Search

de los algoritmos.

Se asume por simplicidad que todos los parámetros escogidos para la versión serial son también los mejores para la versión paralela. En particular el tamaño de la población y la cantidad de hijos de cada generación se dividirán equitativamente entre el número de islas. El resto de los parámetros que son específicos del modelo de paralelismo se calibraron mediante experimentos informales y se presentan en la **Error! Reference source not found.** 5.

F. Calibración de las otras técnicas desarrolladas

Como se explicó en la sección anterior, se realizó una calibración formal sólo para el algoritmo híbrido desarrollado, mientras que para el resto de las técnicas desarrolladas se realizó solo una calibración informal.

Para el caso del algoritmo genético puro se tomó como base la calibración realizada para el algoritmo híbrido, modificando luego los valores de ciertos parámetros en base a experimentos informales. En cuanto a los parámetros del modelo de migración de la versión paralela, se decidió trabajar con los mismos parámetros que fueron escogidos para el algoritmo híbrido

A diferencia de las otras técnicas presentadas, el Tabú Search presentado tiene mucho menos parámetros a calibrar. Los parámetros escogidos fueron calibrados de manera informal y se muestran en la tabla 6.

G. Evaluación Final

La evaluación final de los algoritmos presentados se realizó sobre los escenarios planteados por C. Risso [14]. Considerando que el costo computacional de los algoritmos puede llegar a ser muy grande y los recursos a disposición finitos, se realizó una primera evaluación experimental aplicando todas las técnicas sólo sobre los escenarios de arquitectura napL. Estos escenarios son los más fáciles de resolver entre los presentados por C.Risso, dado que la red MPLS es encargada de rutear mucho menos tráfico que en la arquitectura napH. Los escenarios 2, 6 y 10 son iguales a los 4, 8 y 12 en cuanto a la red MPLS, por lo que sólo los primeros tres serán tenidos en cuenta.

Las cuatro variantes que se decidió evaluar fueron las siguientes:

- Tabú Search
- AG Secuencial
- AG Paralelo
- AG Híbrido Paralelo

Dada la naturaleza estocástica de estos algoritmos, es fundamental realizar múltiples ejecuciones de cada técnica. Las características del Tabu Search hacen que sea una técnica liviana con un costo computacional relativamente bajo, lo que permitió realizar quince ejecuciones para cada uno de los tres escenarios escogidos. No se puede decir lo mismo de las distintas versiones del algoritmo genético, por lo que se realizaron seis ejecuciones de cada variante por cada escenario.

Escenario 02					
	Mejor Costo	Costo Promedio	Desv. estándar	Tiempo Promedio (hs)	T. al ópt. prom. (hs)
Tabú	856484	950559	51375	4,18	2,92
AG	907166	963387	48189	110,83	98,49
AGP	892923	934017	35051	51,26	47,34
Hibrido	910099	946100	25555	52,54	44,89
Escenario 06					
	Mejor Costo	Costo Promedio	Desviación estándar	Tiempo Promedio (hs)	T. al ópt. prom. (hs)
Tabú	718124	825313	56643	4,80	3,90
AG	767269	794204	14794	106,61	91,78
AGP	774398	810455	34659	40,75	35,16
Hibrido	757447	851511	56324	60,95	60,55
Escenario 10					
	Mejor Costo	Costo Promedio	Desviación estándar	Tiempo Promedio (hs)	T. al ópt. prom. (hs)
Tabú	725089	801393	42677	4,66	3,70
AG	808756	835013	22351	126,61	104,31
AGP	769039	798543	18664	43,04	40,34
Hibrido	766666	788260	20914	60,95	48,29

Tabla 7: Resultados obtenidos en escenarios napL.

	Speedup	Eficiencia
Escenario 02	2,16	0,54
Escenario 06	2,62	0,65
Escenario 10	2,94	0,74

Tabla 8: Valores de Speedup y eficiencia entre la versión serial y paralela del AG.

Los resultados obtenidos se presentan en la tabla 7. Se muestra en celeste el mejor valor obtenido en cada caso y en gris el que se encuentra en segunda posición.

Como muestra la tabla 7, el tiempo promedio de ejecución del Tabu Search es cuando menos diez veces menor que el tiempo promedio para la variante más eficiente del AG. A su vez, en todos los casos el mejor costo es siempre obtenido por este algoritmo. Sin embargo, el mejor costo promedio es siempre uno de los peores, en comparación con el resto de las técnicas. Una posible interpretación es que el Tabú Search tiende a estancarse con más facilidad que el resto de los algoritmos. Surge como interrogante entonces si los mejores resultados del Tabú Search se deben a la mayor cantidad de corridas independientes realizadas para esta técnica, en lugar de la propia eficacia del algoritmo. En cualquier caso, queda claramente a la vista que esta técnica es eficaz y muy eficiente en comparación con las variantes del AG.

La tabla 8 muestra los valores de speedup y eficiencia entre el algoritmo genético puro serial y su versión paralela. Puede observarse que el speedup alcanzado es bastante sublineal (se utilizaron cuatro procesadores en el AGP) aunque no deja de ser importante.

Los resultados presentados hasta ahora cumplen la función de brindar métricas para poder comparar los algoritmos desarrollados. Sin embargo, los mismos no son comparables frente a los costos obtenidos por C. Risso. Esto surge porque en cada escenario se incluyen nodos “ficticios” para modelar

la red de AMM y afectan el costo de la solución obtenida del algoritmo, pero no el costo de la red MPLS.

En la tabla 9 se incluyen los costos de la red MPLS de las soluciones obtenidas por la búsqueda tabú (eliminando el costo de los enlaces de AMM) en los escenarios de arquitectura napL y se comparan con aquellos obtenidos por C. Risso. Se puede apreciar que los costos obtenidos en los tres escenarios de arquitectura napL están a menos de 10% del costo obtenido por C. Risso. Además, en el caso del escenario 02, se obtiene una pequeña ventaja de medio punto porcentual.

Visto que la búsqueda tabú obtuvo los mejores resultados sobre los tres escenarios evaluados, se decidió utilizar este algoritmo para resolver también los escenarios de arquitectura napH. Los resultados de esta evaluación se muestran en la tabla 10.

Para esta arquitectura los costos son de calidad sustantivamente inferior a los obtenidos con la arquitectura napL. El gap existente entre los costos obtenidos por el Tabú Search y los obtenidos por C. Risso está entre un 10% y un 35% salvo para el escenario 03 donde el mismo sube a un 45%. La explicación de esto probablemente yace en que el costo internacional es el principal contribuyente al costo de una solución y simplemente parece ser muy difícil enrutarlo de manera eficiente con el sistema de camino primario-alternativo.

Escenario	Costo MPLS Tabu	Costo MPLS Risso	Gap
02	529838	532896	-0,57%
06	489007	451360	8,34%
10	471790	451360	4,53%

Tabla 9: Costo MPLS arquitectura napL.

Escenario	Costo MPLS	Costo MPLS Risso	Costo Int.	Costo Int. Risso	Gap costo Total
01	1.439.494	1.233.168	10.875.000	7.975.000	33,73%
03	1.260.214	1.049.568	8.700.000	5.800.000	45,41%
05	951.321	713.856	5.800.000	5.075.000	16,63%
07	1.191.729	627.848	4.350.000	4.350.000	11,33%
09	976.144	823.160	6.525.000	5.075.000	27,18%
11	1.231.813	651.560	5.075.000	4.350.000	26,10%

Tabla 10: Evaluación del Tabu Search sobre la arquitectura napH

VIII. CONCLUSIONES Y TRABAJO A FUTURO

A. *Conclusiones*

En este documento se presentó la aplicación de diferentes metaheurísticas para atacar el problema del diseño óptimo de una red de datos virtual de tipo MPLS.

Se implementaron varias técnicas con la idea de explorar diferentes posibilidades a la hora de resolver el problema. En un principio se escogieron Tabu Search y algoritmos genéticos por presentarse como técnicas eficaces a la hora de resolver problemas complejos. Luego, con el objetivo de conseguir una técnica que tuviera las mejores características de ambas, se desarrolló un algoritmo híbrido.

Al comparar los resultados obtenidos por las técnicas desarrolladas, la conclusión más relevante seguramente es que algoritmos genéticos no parece ser una técnica muy apropiada para atacar este problema. La desventaja principal que presenta esta metaheurística en el contexto actual, surge de la dificultad de encontrar operadores de cruzamiento eficaces y en el poco rendimiento que presenta el algoritmo si no se usan operadores de búsqueda local. La versión paralela del AG obtuvo resultados sustantivamente mejores que la versión serial. La hibridación del algoritmo genético con el Tabu Search no obtuvo mayores dividendos, a pesar de que se dedicó más tiempo a la calibración de este algoritmo.

Aunque la representación de camino primario-alternativo puede facilitar la implementación de la red, se interpreta que la misma puede impactar en el costo de una solución. Esta interpretación surge del estudio realizado en la evaluación del modelo de ruteo y es confirmada por el gap existente entre los mejores costos obtenidos y aquellos reportados por C. Risso (donde se habilita un ruteo distinto por cada falla).

Los resultados sobre los escenarios de Antel difieren drásticamente de un tipo de arquitectura a otro. Esta observación surge del gap existente entre los resultados obtenidos con el Tabú y aquellos reportados por C. Risso en ambas arquitecturas. Esto está relacionado con la baja eficiencia del ruteo internacional con el modelo de camino primario-alternativo. Este modelo obliga a reservar siempre una segunda ruta para cada par de nodos con demanda. A menos que estas rutas se diseñen de una manera especial, esto implica rutear el doble de la capacidad demandada. Para que la capacidad reservada sea óptima, las rutas se deben diseñar de tal manera que las rutas alternativas se compartan y que ningún escenario de falla haga efectivas dos rutas alternativas que pasan por un mismo enlace.

En resumen, el modelo de ruteo de camino primario alternativo hace muy difícil el ruteo eficiente del tráfico internacional. La razón reside en que el mismo deja de ser apropiado cuando el número de demandas crece, sobre todo si la red de transporte no es muy densa.

Para finalizar las conclusiones, se considera que el Tabú Search desarrollado se presenta como una herramienta eficiente y eficaz para resolver los escenarios de arquitectura napL. Sin embargo, los resultados sobre la arquitectura napH no son muy alentadores. En general se presume que esta herramienta obtiene buenos resultados en redes con capa de transporte densa y con poca demanda a rutear.

B. *Trabajos a futuro*

Considerando las limitaciones de tiempo, quedaron pendientes algunas líneas de investigación que no se desarrollaron en este proyecto. Se propone incluir las siguientes propuestas en posibles trabajos posteriores:

- Considerando la mala calidad de los resultados en los escenarios de arquitectura napH, se propone trabajar con la representación de N-caminos. Otra opción podría ser utilizar el modelo de ruteo de N-Caminos aplicado a una representación basada en la topología de la solución y no en el ruteo.
- Considerando que el modelo de islas de los AGs obtuvo mejores resultados que la versión serial, se propone trabajar con paralelismo de grano más fino.
- Por último se propone trabajar con otras metaheurísticas que no fueron utilizadas en este trabajo, tales como VNS u otras técnicas basadas en trayectoria.

REFERENCIAS

[1] E. Osbourne y A. Simha, Traffic Engineering With MPLS, 1ra ed. Indianapolis: Cisco Press, 2002.

[2] V. Alwayn, Optical Network Design and Implementation, John Kane, Ed. Indianapolis, USA: Cisco Press, 2004.

[3] L. Monma Clide, S. Munson B., y R. Pulleyblank W., "Minimum-weight two-connected spanning networks," en Mathematical Programming., 1990, pags. 153-172.

[4] E. Rosen. (2001, Enero) Rfc-Editor. [Online]. http://www.rfc-editor.org/rfc/rfc3031.txt [Accedido: 31 de Agosto del 2010].

[5] E. P. Pan y E. G. Swallow. (2005, May) IETF Tools. [Online]. http://tools.ietf.org/html/rfc4090 [Accedido: 31 de Agosto del 2010].

[6] Network Working Group. (1990, Febrero) Internet FAQ Archives. [Online]. http://www.rfc-editor.org/rfc/rfc1142.txt [Accedido: 31 de Agosto del 2010].

[7] Ch. Chekuri, S. Khanna, y F. B. Shepherd, "Edge-Disjoint Paths in Planar Graphs with Constant Congestion," en Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, 2006, pags. 757 - 766.

[8] F. Glover, "Future paths for integer programming and links to artificial intelligence.," Computers and Operations Research, vol. 13, no. 5, pags. 533 – 549, 1986.

[9] M. Mitchell, An introduction to genetics algorithms. Cambridge: MIT Press, 1996.

[10] H. E. Magnago, Algoritmos evolutivos aplicados a problemas de diseño de redes confiables, Francisco Javier Lic. Díaz y Jose Luis Mg. Herneńdez, Eds. La Plata: Facultad de Inofrmatica, Universidad de la Plata, 2006.

[11] D. Goldberg, Genetic algorithms in search, optimization, and machine learning. New York: Addison-Wesley Longman Publishing Co, 1989.

[12] D. Calegari, "Algoritmos genéticos aplicados al diseño de una red de comunicaciones confiable," InCo, UdelaR, Montevideo, Proyecto de grado 2002.

[13] Cluster Fing. [Online]. www.fing.edu.uy/cluster [Accedido: 31 de Agosto del 2010]

[14] C. Risso, "Optimizacion de Costos en Redes Multicapas Robustas," Ingenieria Matematica, UdelaR, Montevideo, Tesis de Maestria 2010.