Topological Optimization of Fault-Tolerant Networks meeting Reliability Constraints

Sebastián Laborde

Tutor de Tesis

Dr. Ing. Franco Robledo

Director Académico

Prof. Ing. Omar Viera

Tesis de Maestría - Programa de Posgrado PEDECIBA Informática
Instituto de Computación - Facultad de Ingeniería
Universidad de la República

25 de enero de 2021

Motivación

- Los servicios de fibra al hogar *(Fiber-To-The-Home - FTTH)* tienen una gran penetración mundial brindando a los clientes finales transferencias de datos a altas velocidades.
- La cantidad de aplicaciones y servicios sobre internet ha crecido exponencialmente.
- Es mandatorio escalar de forma inteligente la infraestructura de red.
- Dado que el despligue de la fibra optica es una importante inversión económica, el diseño topológico de las redes *FTTH* debe continuar considerandose.

Objetivos I

- Problema de optimización combinatoria motivado por el diseño topológico de redes de comunicaciones con restricciones de confiabilidad.
- El objetivo es interconectar nodos distinguidos, llamados terminales, utilizando un nivel adecuado de redundancia y de forma simultanea, satisfacer las restricciones de confiabilidad.
- En el análisis de confiabilidad nos enfrentamos a fallas aleatorias en los componentes del sistema. Se considera el modelo como realista hostil, donde tanto nodos como aristas pueden fallar.

Objetivos II

- Encontrar una solución de costo mínimo que alcance un umbral de confiabilidad, donde tanto los nodos como los enlaces pueden fallar con probabilidades dadas.
- Entender el trade-off entre costo-confiabilidad, y como la confiabilidad aumenta naturalmente agregando niveles de redundancia entre los nodos terminales.
- Entender el impacto en la confiabilidad de las redes solución al aumentar o disminuir las probabilidades de falla elementales tanto en nodos como enlaces.
- Pertenece a la clase de problemas *NP-Hard*.
- Se propone una solución que resuelve el problema de forma apróximada con una metodología *GRASP/VND* para el problema de optimización y para el análisis de la confiabilidad el método *RVR*.

Generalized Steiner Problem with Node-Connectivity Constraints and Hostile Reliability (GSPNCHR)

Definition (GSPNCHR)

Consider a simple undirected graph $G = (V, E)$, terminal-set $T \subseteq V$, link-costs $\{c_{i,j}\}_{(i,j)\in E}$ and connectivity requirements $R = \{r_{i,j}\}_{i,j\in T}$. Further, we assume that both links and non-terminal (Steiner) nodes fail with respective probabilities $P_E = \{p_e\}_{e\in E}$ and $P_{V-T} = \{p_v\}_{v\in V-T}$. Given a reliability threshold p_{min}, the goal is to build a minimum-cost topology $G_S \subseteq G$ meeting both the connectivity requirements R and the reliability threshold: $R_K(G_S) \geq p_{min}$, being $K = T$ the terminal-set.

GSPNCHR

$$\text{mín} \sum_{(i,j)\in E} c_{i,j} x_{i,j}$$

$$s.t. x_{ij} \geq y^{u,v}_{(i,j)} + y^{u,v}_{(j,i)} \forall (i,j) \in E, \forall u,v \in T, u \neq v \quad (1)$$

$$\sum_{(u,i)\in E} y^{u,v}_{(u,i)} \geq r_{u,v} \ \forall\, u,v \in T,\ u \neq v \quad (2)$$

$$\sum_{(j,v)\in E} y^{u,v}_{(j,v)} \geq r_{u,v} \ \forall\, u,v \in T,\ u \neq v \quad (3)$$

$$\sum_{(i,p)\in I(p)} y^{u,v}_{(i,p)} - \sum_{(p,j)\in I(p)} y^{u,v}_{(p,j)} \geq 0,\ \forall p \in V - \{u,v\},\ \forall u,v \in T,\ u \neq v \quad (4)$$

GSPNCHR

$$\sum_{(s,i)\in E} x_{s,i} \leq M\hat{x}_s,\ \forall s \in V - T \tag{5}$$

$$R_K(G_S(\{x_{ij}\})) \geq p_{min} \tag{6}$$

$$x_{(i,j)} \in \{0,1\}\ \forall (i,j) \in E \tag{7}$$

$$\hat{x}_i \in \{0,1\}\ \forall i \in V - T \tag{8}$$

$$y^{u,v}_{(i,j)} \in \{0,1\}\ \forall (i,j) \in E,\ \forall u,v \in T,\ u \neq v \tag{9}$$

Resultados Teóricos

- Se demuestra la complejidad computacional del problema *(NP-hardness)*.
- Se modela el problema para instancias particulares con $r_{ij} = k, k \geq 2$, tanto para arista conectividad como para nodo conectividad.
- Se encuentran cotas inferiores para las formulaciones anteriores aplicando teoría de dualidad y relajación lagrangiana.
- Formulación de programación matemática del problema para las versiones k nodo-conectividad y k arista-conectividad aplicando los teoremas de BienStock y Monma.
- Se proponen algoritmos de orden polinomial para resolución del problema en forma exacta de casos particulares respecto a conectividad y topología de grafos.

Trabajos Relacionados

- Se dedicó un capítulo entero donde se expone el relevamiento del estado del arte de la temática de esta tesis: Confiabilidad en redes, Diseño topologico de redes, GRASP, VNS, RVR y otras como Crude Monte Carlo.
- Son escasos los trabajos que abordan conjuntamente la optimización topológica de la red bajo restricciones de confiabilidad.
- VNS y RVR han sido utilizados con gran éxito en problemas de optimización combinatoria.

Publicación

1. A GRASP/VND Heuristic for the Generalized Steiner Problem with Node-Connectivity Constraints and Hostile Reliability to be published in the Proceedings of the 8th International Conference on Variable Neighborhood Search (ICVNS March 2021). Khalifa University, Abu Dhabi, U.A.E. The article will be published by Springer in the Lecture Notes in Computer Science (LNCS) series.

GRASP/VND

- Utilizamos una metodología GRASP/VND.
- GRASP y VND son metaheurísticas bien conocidas que se han utilizado con éxito para resolver muchos problemas difíciles de optimización combinatoria.
- GRASP es un poderoso proceso de arranque múltiple que opera en dos fases. Se construye una solución factible en una primera fase, cuyo vecindario luego se explora en la fase de búsqueda local.
- VND explora varias estructuras de vecindad en un orden determinista. Su éxito se basa en el simple hecho de que las diferentes estructuras de vecindad no suelen tener el mismo mínimo local. Así, la solución resultante es simultáneamente una solución localmente óptima en todas las estructuras vecinas.

Esquema General

Network Design

Alg 1 $sol = NetworkDesign(G_B, iter, k, p_{min}, P_E, P_{V-T}, simiter)$

```
1:  i ← 0; P ← ∅; sol ← ∅
2:  while i < iter do
3:      ḡ ← Construction(G_B, P, k)
4:      g_sol ← VND(ḡ, P)
5:      reliability ← RVR(g_sol, P_E, P_{V-T}, simiter)
6:      if reliability > p_min then
7:          sol ← sol ∪ {g_sol}
8:      end if
9:  end while
10: return sol
```

Alg 2 $(sol, P) = Construction(G_B, C, R, k)$

1: $g_{sol} \leftarrow (S_D^{(I)}, \emptyset)$; $m_{i,j} \leftarrow r_{i,j}$; $P_{i,j} \leftarrow \emptyset, \forall i,j \in S_D^{(I)}$; $A_{i,j} \leftarrow 0, \forall i,j \in S_D^{(I)}$
2: **while** $\exists\, m_{i,j} > 0 : A_{i,j} < MAXATTEMPTS$ **do**
3: $(i,j) \leftarrow ChooseRandom(S_D^{(I)} : m_{i,j} > 0)$
4: $\overline{G} \leftarrow G_B \setminus P_{i,j}$
5: **for all** $(u,v) \in E(\overline{G})$ **do**
6: $\overline{c}_{u,v} \leftarrow c_{u,v} \times 1_{\{(u,v) \notin g_{sol}\}}$
7: **end for**
8: $L_p \leftarrow KSP(k, i, j, \overline{G}, \overline{C})$
9: **if** $L_p = \emptyset$ **then**
10: $A_{i,j} \leftarrow A_{i,j} + 1$; $P_{i,j} \leftarrow \emptyset$; $m_{i,j} \leftarrow r_{i,j}$
11: **else**
12: $p \leftarrow SelectRandom(L_p)$; $g_{sol} \leftarrow g_{sol} \cup \{p\}$
13: $P_{i,j} \leftarrow P_{i,j} \cup \{p\}$; $m_{i,j} \leftarrow m_{i,j} - 1$
14: $(P, M) \leftarrow GeneralUpdateMatrix(g_{sol}, P, M, p, i, j)$
15: **end if**
16: **end while**
17: **return** (g_{sol}, P)

Fase Búsqueda Local

VND

El objetivo es combinar una rica diversidad de vecindades para obtener una solución óptima local para cada vecindario. NetworkDesign considera una implementación clásica de VND, ordenando las respectivas búsquedas locales después de la fase de construcción. Se consideran tres estructuras de vecindad para construir VND.

- SwapKeyPathLocalSearch
- KeyPathLocalSearch
- KeyTreeLocalSearch

Definition (key-node)

Un key-node en una solución factible $v \in g_{sol}$ es un nodo de Steiner (no terminal) con grado mayor o igual a tres.

Definition (key-path)

Un key-path en una solución factible $p \subseteq g_{sol}$ es un camino elemental donde todos los nodos intermedios no son terminales con grado dos en g_{sol}, y los nodos extremos son terminales o key-nodes.

Definition (key-tree)

Sea $v \in g_{sol}$ un key-node perteneciente to a una solución factible g_{sol}. El key-tree asociado a v, denotado como T_v, es un árbol compuesto por todos los key-paths que se encuentran en un punto común (i.e., el key-node v).

Definition (Estructura de Vecindad para Swap Key-Paths)

Dado un key-path $p \subseteq g_{sol}$, una solución vecina para g_{sol} es $\hat{g}_{sol} = \{g_{sol} \setminus p\} \cup \{m\}$, siendo m el conjunto de nodos y enlaces que serán añadidos preservando la factibilidad de $\hat{g}_{sol}$.

Alg 3 $g_{sol} = SwapKeyPathLocalSearch(G_B, C, g_{sol}, P)$

```
1:  improve ← TRUE
2:  while improve do
3:      improve ← FALSE
4:      K(g_sol) ← {p_1, ..., p_h} /* Key-path decomposition of g_sol */
5:      while not improve and ∃ key-paths not analyzed do
6:          p ← (K(g_sol)) /* Path not analyzed yet */
7:          (g_sol, improve) ← FindSubstituteKeyPath(g_sol, p, P)
8:      end while
9:  end while
10: return g_sol
```

Definition (Estructura de Vecindad para Key-Paths)

Dado un key-path $p \in g_{sol}$, una solución vecina es $\hat{g}_{sol} = \{g_{sol} \setminus p\} \cup \{\hat{p}\}$, donde $\hat{p}$ es otro camino que conecta los extremos desde p. La vecindad de key-paths desde g_{sol} esta compuesta por la operación previa a los posibles miembros pertenecientes a $K_{g_{sol}}$.

Alg 4 $g_{sol} = KeyPathLocalSearch(G_B, C, g_{sol})$

```
1:  improve ← TRUE
2:  while improve do
3:      improve ← FALSE
4:      K(g_sol) ← {p_1, ..., p_h} /* Key-path decomposition of g_sol */
5:      while not improve and ∃ key-paths not analyzed do
6:          p ← (K(g_sol)) /* Path not analyzed yet, with extremes u and v */
7:          μ̂ ←< NODES(p) ∪ S_D \ NODES(g_sol) > /* Induced subgraph μ̂ */
8:          p̂ ← Dijkstra(u, v, μ̂)
9:          if COST(p̂) < COST(p) then
10:             g_sol ← {g_sol \ p} ∪ {p̂}
11:             improve ← TRUE
12:         end if
13:     end while
14: end while

15: return g_sol
```

Definition (Estructura de Vecindad para Key-Tree)

Se considera el key-tree $T_v \in g_{sol}$ con raíz key-node v. Un vecino de g_{sol} es $\hat{g}_{sol} = \{g_{sol} \setminus T_v\} \cup \{T\}$, siendo T otro árbol que reemplaza a T_v con hojas identicas.

Alg 5 $g_{sol} = KeyTreeLocalSearch(G_B, C, g_{sol})$

```
improve ← TRUE
while improve do
    improve ← FALSE
    X ← KeyNodes(g_sol) /* Key-nodes from g_sol */
    S̄ ← S_D \ NODES(g_sol)
    while not improve and ∃ key-nodes not analyzed do
        v ← X /* Key-node not analyzed yet */
        [g_sol, improve] ← GeneralRecConnect(G_B, C, g_sol, v, S̄)
    end while
end while
return g_sol
```

RVR

Alg 6 $RVR(G, K, p_v, p_e)$

1: **If** *terminals*=1, **return** 0
2: **Elseif** $\phi(G, K) = 1$, **return** 1.
3: **Else**
4: $D := GetKExtendedCut(G)$
5: $Q_D := AllFailedProb(D)$
6: $index := GetRandomItem(D)$
7: $c := D[index]$
8: $remove(G, D, index - 1)$
9: $add(G, c)$
10: $return\ Q_D + (1 - Q_D) \times RVR(G)$
11: **EndIf**

Instancias de prueba

Caracterización instancias

Instancias	Características de las instancias			
	$\|V\|$	$\|E\|$	Densidad	$\|E(C)\|$
c-fat200-1	200	1534	0.071	81
c-fat200-2	200	3235	0.163	306
c-fat200-5	200	8473	0.426	1892
c-fat500-1	500	4459	0.036	110
c-fat500-2	500	9139	0.073	380
c-fat500-5	500	23191	0.186	2304
c-fat500-10	500	46627	0.374	8930
p_hat300-2	300	21928	0.489	4637
p_hat300-3	300	33390	0.744	7740
keller5	776	225990	0.752	15184
MANN_a27	378	70551	0.990	31284
c125_9	125	69632	0.899	236406

Resultados obtenidos

	GRASP/VND		Algoritmo Genético		GAP
Instancias	$\|E(C)\|$ prom.	$T(s)$ prom.	$\|E(C)\|$ prom.	$T(s)$ prom.	(%)
c-fat200-1	81	0.37	81	6.4	0.0
c-fat200-2	306	0.81	306	7.5	0.0
c-fat200-5	1892	4.94	1892	12.5	0.0
c-fat500-1	110	2.46	110	16.15	0.0
c-fat500-2	380	5.83	380	14.3	0.0
c-fat500-5	2304	10.85	2304	20.36	0.0
c-fat500-10	8930	65.74	8930	32.59	0.0
p_hat300-2	4636.2	3659.39	4633.40	171.9	$\approx$0.0
p_hat300-3	7726.8	3992.42	7387.27	279.8	0.04
c125_9	2766	253.25	2737.2	5.0	0.01
keller5	15183.24	1167.64	12382	50.57	0.18
MANN_a27	31244.10	548.54	30405	46.49	0.03

Conclusiones

Conclusiones

- Aplicaciones diversas en diferentes áreas.
- Se demuestra la $\mathcal{NP}$-Completitud.
- Solución competitiva con las existentes y con tiempos de ejecución muy buenos.

Trabajo Futuro

Trabajo Futuro

- Aplicaciones reales en grandes superficies.
- Explorar la versión con pesos en las aristas, (WMCC).

Fin

Gracias por su atención.