Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la Computación
CC5901-01 - Práctica Profesional II

Informe de Práctica Profesional II

NIC Chile Research Labs

Ivana Francisca Bachmann Espinoza
2 de octubre de 2015

Rut: 18.120.263-7
Carrera: Ingeniería Civil en Computación
Correo: ivana.bachmann@gmail.com
Celular: (+56) 9 6591103

Índice general

Capítulo 1

Resumen

El trabajo fue realizado en el laboratorio de investigación NIC Chile Research Labs. Este consistió en el desarrollo de una médida de robustez en redes para contribuir en el proyecto de creación de una medida multidimensional de resiliencia en redes.

Las redes están presentes en nuestra vida de múltiples formas desde redes sociales hasta redes de comunicaciones como Internet. Estas han sido ampliamente estudiadas como redes complejas de arcos y nodos que describen sus estructuras, relaciones, etc. Internet como red se divide en una multiplicidad de aristas donde destacamos la capa lógica y la física. Que este tipo de redes se mantenga estable y sin riesgo de fallo, es decir que sean robustas es un campo que se estudia actualmente, sin embargo la robustez de Internet hasta ahora ha sido estudiada siempre desde una perspectiva a la vez. Se hace necesario entonces el desarrollo de una medida de robustez integrada para comprender realmente qué tan frágil es una red y es por esto que se genera el proyecto para crear una medida multidimensional de robustez en redes.

El índice desarrollado tienen por objetivo proveer una forma simple y certera de identificar elementos riesgosos dentro de una red, o sea, elementos que de sufrir una falla o ataque y desconectarse del resto de la red pudiesen cortar a dicha red. La creación de un índice como el mencionado permitiría complementar a otras medidas existentes para desarrollar finalmente una medida precisa y confiable de la robustez de una red. El indice se desarrolló en tres etapas, la etapa de investigación o recolección de información, la etapa de diseño y la etapa de desarrollo y prueba del índice, siendo cada una de las etapas crítica en el desarrollo del índice final.

El trabajo realizado sobre el desarrollo del índice va a ser presentado en la XXXIV Conferencia Internacional de la Sociedad Chilena de Ciencias de la Computación.

En cuanto a los aprendizajes obtenidos se destaca el conocimiento adquirido en el desarrollo de investigación y el trabajo guiado por investigadores con experiencia.

Capítulo 2

Introducción

2.1. Lugar de trabajo

La práctica profesional serealizó en NIC Chile Research Labs[1], laboratorio de investigación aplicada en las áreas de redes computacionales, sistemas distribuidos y protocolos de Internet. Se encuentra ubicado en Av. Almirante Blanco Encalada #1975, Santiago, Chile. El trabajo se realizó de forma presencial en las instalaciones del laboratorio, entre en los meses de Junio y Julio.

2.2. Grupo de trabajo

En NIC Chile Research Labs trabajan investigadores de tiempo completo, desarrolladores experimentados, profesionales de otras áreas y estudiantes tanto de pregrado como postgrado. En particular para el trabajo realizado en la práctica se contó con la guía de dos investigadores, Javier Bustos doctor en ciencias de la computación y Patricio Reyes doctor en matemáticas.

2.3. Equipos y Software

2.3.1. Software

El software utilizado para desarrollar el trabajo fue *Python* como lenguaje de programación, la librería para manejo de redes *igraph* y el sistema operativo *Ubuntu*. A continuación se presenta una breve descripción los elementos más relevantes para el desarrollo y comprensión del trabajo realizado:

1. **igraph:** igraph corresponde a una librería que reúne un conjunto de herramientas para el análisis de redes. Se puede programar en los lenguajes **R**, **Python** y **C/C++**. Se decidió usar ésta librería para el desarrollo de la práctica pues *igraph* es *open source* y provee herramientas simples y eficientes para crear y analizar redes.

   Dentro de *igraph* destacamos a la clase *Graph* que nos permite modelar grafos. La clase *Graph* permite acceder a arcos y nodos mediante *Graph.es* y *Graph.vs* respectivamente, estas retornan listas de arcos y nodos con sus propiedades. Los arcos y nodos pueden poseer propiedades asociadas como nombres, métricas u otras propiedades y estas pueden ser accesadas a través *Graph.es["propiedad"]* o *Graph.vs["propiedad"]*. Además *igraph* provee métodos para construir grafos con propiedades conocidas como grafos k**-regulares**, grafos **completos**, grafos **estrella**, entre otros. Provee también métodos para calcular métricas conocidas como **betweenness** para arco y nodo, **grado**, etc[2].

2. **Python:** Python es un lenguaje mutiparadigma que soporta tanto programación imperativa como programación orientada a objetos. Corresponde a un lenguaje de tipado dinámico con la capacidad de correr en múltiples plataformas. En el contexto de esta práctica se dicidió utilizar este lenguaje pues soporta la librería *igraph* además este lenguaje tiene una sintaxis simple, legible y ya se contaba con experiencia programando en este lenguaje [3].

2.3.2. Equipo

En cuanto a los equipos computacionales, el laboratorio provee tanto equipos estacionarios como portátiles para aquellos que se encuentran trabajando o realizando prácticas. Para el proyecto desarrollado en particular se utilizó un equipo estacionario el cual fue provisto por el laboratorio.

2.4. Situación Previa

Previo a la práctica el proyecto se encontraba en su etapa de concepción, es decir aún no se habían realizado avances sobre la idea original. Por esto la practicante se integró al proyecto en su etapa inicial de desarrollo.

2.5. Descripción general del trabajo realizado

El objetivo del trabajo realizado fue el de *aportar* en el desarrollo de una métrica de resiliencia en redes que integre las múltiples dimensiones de la misma de forma que se pueda tener una visión completa de la resistencia a fallos real de la red.

El aporte de la practicante consitió en el desarrollo de una métrica de robustez de redes que permita identificar rápidamente puntos críticos de fallo en una red.

Para llevar a cabo el trabajo este fue dividido en tres etapas que se describen brevemente a continuación.

1. **Investigación:** Dado que el trabajo consistió en aportar en el desarrollo de una métrica de robustez, se comenzó investigando sobre redes, sus distribuciones, definiciones de robustez y métricas existentes.
2. **Diseño:** Una vez realizada la investigación se procedió a idear una forma de aportar en el proyecto. Para ello se identificaron aquellos puntos que la métricas existentes no tomaban en cuenta respecto a la medición de robustez de una red y con estos puntos se diseñó una nueva métrica.
3. **Desarrollo y pruebas:** Luego de definir formalmente la métrica se realizaron pruebas en redes simuladas para probar su efectividad encontrando puntos débiles en redes versus otras métricas conocidas y comunmente utilizadas para medir robustez.

Capítulo 3

Trabajo realizado

Las redes de comunicaciones naturalmente se ven afectadas por diversas otras redes que interactúan con ellas o se manifiestan a su alrededor, como por ejemplo las redes electricas o el hecho de que la red de Internet está compuesta por más de una capa (red física, red lógica). Un cambio en estas redes "circundantes"puede provocar modificaciones en la red de comunicaciones original, a pesar de que cambios en la red de comunicaciones original no necesariamente afecta a las demás redes a su alrededor. Luego podemos considerar a dichas redes como objetos multidimensionales, pues su comportamiento no depende únicamente de ellas mismas y su topología, si no que también se ven afectadas por el comportamiento de las redes a su alrededor. Es por esto que nace la idea del proyecto de desarrollar una métrica que mida la resistencia a fallo, o robustez, de una red de comunicaciones que considere esta multidimensionalidad.

Como se ha mencionado previamente el trabajo consistió en aportar en el desarrollo de esta métrica multidimensional de resiliencia en redes. Dado que los factores que afectan a una red de comunicaciones pueden ser típicamente modelados también como redes el trabajo desarrollado consistió en encontrar o crear una métrica que permitiese medir apropiadamente la robustez de una red cualquiera. Para llevar a cabo esto se procedió a desarrollar el trabajo en las etapas que describiremos a continuación.

3.1. Investigación

El primer paso para realizar el trabajo solicitado fue el de investigar sobre redes, métricas y definiciones de robustez. Durante la investigación se buscó información en múltiples lugares y se leyó una gran variedad de papers. Con el objetivo de presentar de forma clara el trabajo realizado se describirán aquellos puntos que jugaron un rol vital durante el desarrollo de la práctica.

3.1.1. Redes

Una ***red*** corresponde a una estructura compuesta por **nodos** y **enlaces**. Cada nodo se puede relacionar con otro a través de un enlace, de esta forma las redes pueden utilizarse para representar estructuras como redes de comunicación, redes de influencias,redes sociales (donde cada nodo puede representrar una persona y cada enlace puede representar la relación *" amigo de"*),etc.

3.1.2. Grado

Dado un nodo en una red se define su **grado** como el número de **vecinos** que éste posee, donde sus *vecinos* son todos aquellos nodos a los que se une a través de un enlace. [5]

3.1.3. Componentes conexas

Dada una red se definen sus **componentes conexas** como las *sub-redes* tales que cada par de nodos en la *sub-red* se encuentran conectados por un camino de enlaces [6]. (ver figura 4.1).

3.1.4. Distribución Power-law

Una función **power-law** corresponde a la función definida por $f(x) = Cx^{-\alpha}$ con C y α constantes. En base a esta función se pueden definir redes cuya distribución de grados sigue una distribución **power-law** para un cierto exponente α.

La distribuciones de grados *power-law* son usualmente utilizadas para modelar redes sociales y redes de comunicaciones como puede ser la red de Internet. Estas redes son conocidas como *scale-free networks* o *redes de escala libre* [7]. (ver figura 4.2)

3.1.5. Métricas conocidas

Se muestran a continuación algunas de las métricas utilizadas comunmente para medir la robustez de una red que se consideraron como parte de la investigación y forman parte del trabajo desarrollado. Estas medidas se conocen como *centrality measures* o *medidas de centralidad* pues entregan una noción de que tan "central"se encuentra un nodo en una red [8], en particular estas médidas se calculan nodo a nodo. Para ver más métricas estudiadas se puede revisar el paper de Ellen van der Meer, *Comparing measures of network robustness* [9].

1. **Degree centrality:** Corresponde al número de enlaces que posee un nodo con los demás [5]. Un nodo que se conecta con muchos otros nodos se considera "*más central*" que un nodo de grado bajo.

2. **Betweenness centrality:** Corresponde a la fracción de caminos de largo mínimo que pasan por un determinado nodo. [10]. Si muchos caminos mínimos pasan por un nodo podemos considerar que dicho nodo es "*central*".

3. **Harmonic centrality:** Es el promedio *"harmonico"* del inverso de las distancias entre el nodo al cual le calculamos su *harmonic* y el resto. A menor sea la distancia a otros nodos mayor será este promedio [11]. En particular esta medida permite considerar incluso a aquellos nodos de la red que no es posible alcanzar desde el nodo original simplemente considerando ésta distancia como infinita.

3.1.6. R-index

El **R-index** o medida de robustez **R** corresponde a una medida de robustez que considera el tamaño de la componente conexa más grande de una red luego de que se removieran nodos o enlaces de ésta, es decir, luego de haber sufrido un ataque.

El **R-index** se define como

$$R = \frac{1}{N} \sum_{i=1}^{N} \sigma(\frac{i}{N})$$

Donde N es el número de nodos en la red y $\sigma(\frac{i}{N})$ corresponde al tamaño de la componente conexa más grande de la red luego de remover i nodos o enlaces bajo *alguna estrategia* (por ejemplo, remover los i nodos con mayor grado) [12].

3.2. Diseño

Una vez realizado el proceso de investigar se detectó que si bien las medidas de centralidad proveían un buen acercamiento para encontrar puntos débiles en una red, muchas de estas son complejas y no permiten encontrar los puntos que causan el *mayor* daño a la red, lo que en este caso se traduce a encontrar aquellos puntos que de ser desconectados de la red provocan un mayor número de nodos desconectados de la componente conexa más grande. Por esto se decidió proceder a desarrollar un índice apropiado.

3.2.1. Objetivo del índice

El criterio para determinar si un posible índice era suficientemente bueno para ser considerado se definió de acuerdo a si cumplía o no ciertas condiciones. Estas condiciones son:

1. Ser capaz de detectar con *facilidad* aquellos puntos o nodos en una red que de ser desconectados causarían la *mayor* pérdida de nodos.
2. Distinguir entre aquellos puntos que causan pérdida real de nodos al ser desconectados de los que no.

3.2.2. Definición

Luego de realizar un proceso iterativo en el que se desarrollaron varias métricas posibles, se probaron teóricamente y finalmente se descartaron al no cumplir el objetivo, la practicante llegó a la definición de un indice que en efecto cumple lo deseado.

Este índice mide el impacto que causaría la desconexión de un nodo comparando las componentes conexas que quedan *luego* de desconectar dicho nodo, donde desconectar se refiere a eliminar todos los enlaces en los que participa este nodo. Para esto se compara el tamaño de la componente conexa más grande con el tamaño del resto de las componentes conexas. En este caso se considera que el tamaño de una componente conexa corresponde al número de nodos que posee. A más similares en tamaño sean las componentes conexas respecto a la componenete conexa más grande mayor es el valor del índice. En caso de que la desconexión del nodo sólo genere 1 nueva componente conexa (es decir sólo la correspondiente al nodo desconectado) entonces el valor índice es 0. Así el índice en efecto informa de que tan grave es desconectar un nodo, y en caso que al desconectar el nodo no se pierdan otros nodos el índice les otroga el valor 0 permitiendo distinguir aquellos elementos que causarían daño real de aquellos que no.

El índice fue bautizado como "*Miuz*" y formalmente se define como: Sea $\mathcal{N}$ una red de tamaño N, sea $\mathcal{C}(\mathcal{N} \setminus n)$ el conjunto de componentes conexas de $\mathcal{N}$ luego de desconectar el nodo n de la red. El índice *Miuz* del nodo n en la red $\mathcal{N}$ es:

$$Miuz_{\mathcal{N}}(n) = \begin{cases} \frac{\sum_{c \in \mathcal{C}(\mathcal{N} \setminus n)} \|c\|}{max_{c \in \mathcal{C}(\mathcal{N} \setminus n)} \|c\|} - 1, \text{Si } \|\mathcal{C}(\mathcal{N} \setminus n)\| \neq \|\mathcal{C}(\mathcal{N})\| + 1 \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{Si no} \end{cases} \tag{3.1}$$

Donde $\|c\|$ corresponde al tamaño de la componente conexa c, $c \in \mathcal{C}(\mathcal{N} \setminus n)$. El índice $Miuz_{\mathcal{N}}(\cdot)$ toma valores entre 0 y $N - 1$, pudiendo alcanzar su máximo valor en la red estrella de N nodos, donde al desconectar el nodo central la red queda dividida en exactamente N componentes conexas de igual tamaño.

3.3. Desarrollo y pruebas

Para probar la efectividad del índice *Miuz* encontrando los puntos débiles de una red de realizaron pruebas para compararlo con otras métricas usadas para probar la robustez de una red como lo son *harmonic*, *betweenness* y *degree*. Para esto se implementó un algoritmo en Python capaz de calcular el *Miuz* para cada nodo de una red. Para obtener las demás medidas se utilizó la librería igraph en Python, en particular los métodos *degree()*, *betweenness()* de la clase *Graph* y además se implementó una función para calcular el *harmonic* de una red nodo a nodo.

3.3.1. Implementación

Para realizar las pruebas de efectividad primero se implementó un algoritmo para calcular el *Miuz* para cada nodo en un grafo. Respecto a las demás medidas a probar, salvo por *harmonic*, no fue necesario implementarlas pues igraph las provee. El algoritmo para calcular el *Miuz* se implementó de la siguiente manera:

1. Dado un grafo debemos recorrer uno a uno sus nodos y calcular su *Miuz*.

```
def setMiuz(originalGraph):
    # inicializar variables
    for i in range(numberOfNodes):
        # calculo de miuz nodo a nodo
```

2. Dentro del calculo del índice se pueden omitir aquellos nodos con grados menor o igual a 1 pues estos siempre tienen índice *Miuz* igual a 0 al no poder crear nuevas componentes conexas al ser desconectados de la red (salvo por la componente compuesta por ellos mismos).

```
def setMiuz(originalGraph):
    # inicializar variables
    for i in range(numberOfNodes):
        notIsolatedNode = len(originalGraph.neighbors(i)) > 1
        if notIsolatedNode:
            # calculo de miuz nodo a nodo
```

3. Para calcular el miuz de aquellos nodos con grado mayor que 1 se crea una copia del grafo original, se desconecta el nodo, se aplica la fórmula correspondiente en caso de que las componentes conexas aumenten al desconectar el nodo y se le asigna el valor del índice al nodo correspondiente.

```
def setMiuz(originalGraph):
    # inicializar variables
    for i in range(numberOfNodes):
        notIsolatedNode = len(originalGraph.neighbors(i)) > 1

        if notIsolatedNode:
            graphCopy = makeGraphCopy(originalGraph)
            initialNumberOfConnectedComponents = len(graphCopy.components(mode=STRONG))
            disconnectNode(graphCopy,i)
```

```
            currentConnectedComponents = graphCopy.components(mode=STRONG)
            numberOfConnectedComponents = len(currentConnectedComponents)

            if numberOfConnectedComponents > initialNumberOfConnectedComponents + 1:
                # inicializar variables #
                # generar suma de tamanos de las componentes conexas sin contar a la componente conexa mas
                    grande #
                miuz = suma/sizeOfLargestComponent
                originalGraph.vs[i]["miuz"] = miuz
```

Y luego se implementó un algoritmo para calcular *harmonic* simplemente siguiendo la fórmula y utilizando el método *shortest_paths()* de la clase *Graph* de la librería:

```
def setHarmonic(graph):
        numberOfNodes = len(graph.vs)
        distances = graph.shortest_paths()
        sum = 0
        harmonics = []
        for i in range(numberOfNodes):
                for j in range(numberOfNodes):
                        if i != j:
                                sum += 1.0/distances[i][j]
                harmonics.append((1.0/(numberOfNodes-1))*sum)
                sum = 0
        graph.vs["harmonic"] = harmonics
```

3.3.2. Simulaciones y pruebas

Para analizar qué tan efectivo es el nuevo índice identificando puntos débiles en una red respecto a los mencionados anteriormente, se realizaron pruebas de simulación de ataques en redes de distribución power law con coeficiente $\alpha \in \{2{,}15, 2{,}2, 2{,}3\}$. Esto con el objetivo de simular la red de Internet.

Conjunto de datos

Para realizar la simulación de ataques bajo las distintas métricas se generó un conjunto de redes de distribución power law con coeficiente $\alpha \in \{2{,}15, 2{,}2, 2{,}3\}$. Para cada α se generaron 50 redes diferentes cada una de 1000 nodos.

Dado que igraph no provee herramientas para generar una red con distribución power law se implementó la función *generatePowerLawGraph* que genera las redes requeridas. Luego de generar las redes, estas fueron almacenadas en archivos para su posterior uso. A continuación se describe la implementación de *generatePowerLawGraph*:

1. Con el objetivo de generar una distribución de grados que siga una power law se creó la función *getDegreesPowerlaw* que utilizando *random* de Python genera una lista de grados de

nodos de largo n que siguen distribución Power law para un determinado *alpha*.

```
def getDegreesPowerlaw(n, alpha):
    # ... #
```

2. Utilizando el método *Degree_Sequence* de la clase *Graph* provista por igraph que recibe una distribución de grados y genera un grafo, se implementó *generatePowerLawGraph* que genera un grafo de n nodos con distribución Power law de coeficiente *alpha* con un error *epsilon*.

```
        nodeDegrees = getDegreesPowerlaw(n,lamda)
        results = powerlaw.Fit(node_degrees, discrete=True)
        alpha = results.power_law.alpha
        difference = math.fabs(alpha - lamda)
        while True:
            while(difference > epsilon):
                    # recalcular difference #
            # se intenta generar un grafo con results
            # esto puede fallar
            try:
                        g = Graph.Degree_Sequence(node_degrees, method="vl")
                        return g
                except Exception, e:
                        difference = epsilon + 1
                        pass
```

Para ajustar el error la función *generatePowerLawGraph* utiliza la función *fit* del paquete *powerlaw* de python, *fit* retorna el coeficiente α que mejor se ajusta a la distribución entregada. De esta forma *generatePowerLawGraph* genera distribuciones de grados hasta que la diferencia entre lo entregado por *fit* y el *alpha* recibido por la función tengan una diferencia inferior a *epsilon*

Ataques

Dado un índice se realizaron ataques *secuenciales* [12] de la siguiente forma:

1. Se calcula el índice para cada nodo en la red.
2. Se desconecta el nodo con el mayor valor para dicho índice.
3. Se calcula el R-index *parcial*. Es decir se calcula $\sigma(i/\mathcal{N})$ que corresponde al tamaño de la componente conexa más grande hasta el momento dividido por el tamaño original de la red, con $\mathcal{N}$ la red.
4. Se descarta el nodo recién desconectado y se vuelve al punto 1.

Para obtener los datos sobre los ataques se implementó un *script* que calcula y almacena los datos sobre los R-index parciales bajo los indices elegidos (*degree*, *harmonic*, *betweenness* y *miuz*). Este *script* fue originalmente programado sin optimizaciones particulares, sin embargo resultó ser demasiado lento para la cantidad de datos a analizar, por lo que fue re-implementado usando programación dinámica lo cual redujo considerablemente el tiempo requerido para los cálculos.

3.3.3. Resultados

El índice *Miuz* resultó ser más efectivo reduciendo el tamaño de la componente conexa más grande en *pocos golpes* que las demás medidas en todos los casos estudiados, llegando a reducir el tamaño de la componente conexa más grande a menos de un 40 % del tamaño de la red original en los primeros 15 nodos removidos por un ataque basado en *Miuz*. Sin embargo luego de reducir la componente conexa más grande a un tamaño aproximado de un 30 % del tamaño de la red original el resto de las medidas supera el desempeño del *Miuz*. Esta situación se puede apreciar con más detalle en las figuras 4.3 y 4.4. En las imágenes el *R-index* total corresponde al área debajo de cada curva, donde un área más pequeña significa que la red es menos resistente a un *ataque* bajo la medida que dicha curva representa.

Como se mencionó *Miuz* resulto ser más efectivo durante los primeros ataques, no necesitándose mucho más que 10 ataques para reducir la red a un 50 % de su tamaño original. En particular estos resulados demuestran que *Miuz* en efecto es capaz de encontrar los puntos que causan mayor pérdida de nodos en caso de ser desconectados (lo que se traduce a generar mayor pérdida en las simulaciones de ataques), e incluso demuestra ser mejor reconociendo estos puntos que otras medidas normalmente usadas *con el mismo propósito*. Esto último se puede observar mejor en las siguiente imágen 4.5.

Discusión

Considerando atacantes reales o situaciones de riesgo reales como posibles catástrofes, la cantidad de nodos en una red que efectivamente pueden desconectarse del resto de ésta es relativamente baja en comparación con el tamaño real de la red, por lo cual *Miuz* como índice cumple con el objetivo de identificar los peores puntos, dando pie a una posible mejora de la red en estos puntos, encontrando de manera efectiva aquellos nodos que de removerse provocarían un daño mucho mayor. Así incluso si luego de una cierta cantidad de nodos removidos el índice pasa a ser menos efectivo que otras medidas conocidas el índice *Miuz* sigue siendo útil, porque esto no ocurre hasta se han removido tantos nodos que la red ha dejado de ser funcional (la componente conexa más grande corresponde a un 30 % de la red original).

Capítulo 4

Conclusiones

Los resultados del trabajo presentado en este informe fueron considerados por los guías del proyecto como satisfactorios. El índice desarrollado cumplió con las pretenciones de servir para la posterior creación de una medida multidimensional de resiliencia en redes. El índice desarrollado y sus resultados además fueron aprobados y enviados a la XXXIV Conferencia Internacional de la Sociedad Chilena de Ciencias de la Computación a realizarse en las Jornadas Chilenas de la Computación 2015 [13].

Respecto al futuro del proyecto hay que destacar que aún se sigue trabajando en él, en el desarrollo de una medida de resiliencia en redes multidimensional, y que el uso índice desarrollado está contemplado en el desarrollo del proyecto. Además el índice *Miuz* se va a seguir investigando y desarrollando con miras a probar su efectividad no sólo sobre los nodos de un red sino que también sus enlaces.

Dentro del aprendizaje obtenido en el proceso de la práctica se destacan a nivel técnico el aprendizaje y descubrimiento de la librería *igraph* para el manejo y modelamiento de redes. También se destaca el aprendizaje a nivel del desarrollo de investigación, su proceso, sus etapas y sus métodos, y la experiencia que significa ahondar en un área en constante estudio y desarrollo como lo son las redes y finalmente ser capaz de realizar un pequeño aporte en su avance. Todas las tecnologías mencionadas fueron de suma importancia para llevar a cabo de forma apropiada y a tiempo el trabajo pedido a la practicante. Dentro de otros aspectos se aprendió sobre el trabajo bajo la tutela de investigadores con experiencia y el valor que aporta el mantener una comunicación fluida y constante con los mismos. Finalmente queda destacar el aprendizaje sobre la importancia que tiene para el desarrollo de una investigación la perseverancia, el aprendizaje constante, la creatividad e, incluso, la valentía que requiere el intentar algo nuevo sin saber realmente si va a llegar al resultado esperado.

Bibliografía

[1] Sitio web de NIC Chile Research Labs: http://www.niclabs.cl/

[2] Manual de igraph para python: http://igraph.org/python/doc/tutorial/tutorial.html

[3] Python wiki: https://wiki.python.org/moin/

[4] Redes: https://en.wikipedia.org/wiki/Network_theory

[5] Grado: https://en.wikipedia.org/wiki/Degree_(graph_theory)

[6] Componentes conexas: https://en.wikipedia.org/wiki/Connected_component_(graph_theory)

[7] Distribución powerlaw, red de escala libre: https://en.wikipedia.org/wiki/Scale-free_network

[8] Centralidad: https://en.wikipedia.org/wiki/Centrality

[9] Paper: Comparing measures of network robustness, Ellen var der Meer. https://www.few.vu.nl/en/Images/werkstuk-meer_tcm39-280356.pdf

[10] Betweenness centrality: https://en.wikipedia.org/wiki/Betweenness_centrality

[11] Harmonic centrality: http://tuvalu.santafe.edu/~aaronc/courses/5352/fall2013/csci5352_2013_L4.pdf

[12] R-index: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3615130/?report=classic

[13] XXXIV Conferencia Internacional de la Sociedad Chilena de Ciencias de la Computación: http://www.jcc2015.inginformatica.cl/index.php/jcc/eventos

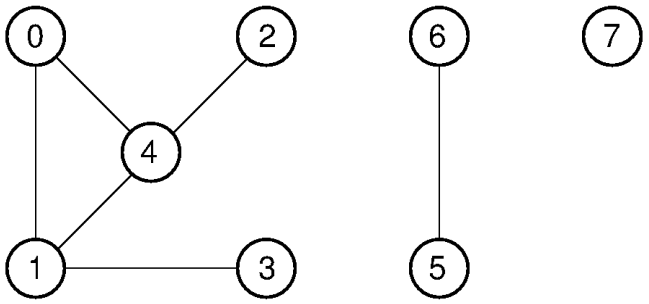


Figura 4.1: Grafo con 3 componentes conexas. La primera contiene a los nodos $\{0, 1, 2, 3, 4\}$, la segunda a los nodos $\{5, 6\}$ y la tercera al nodo $\{7\}$.

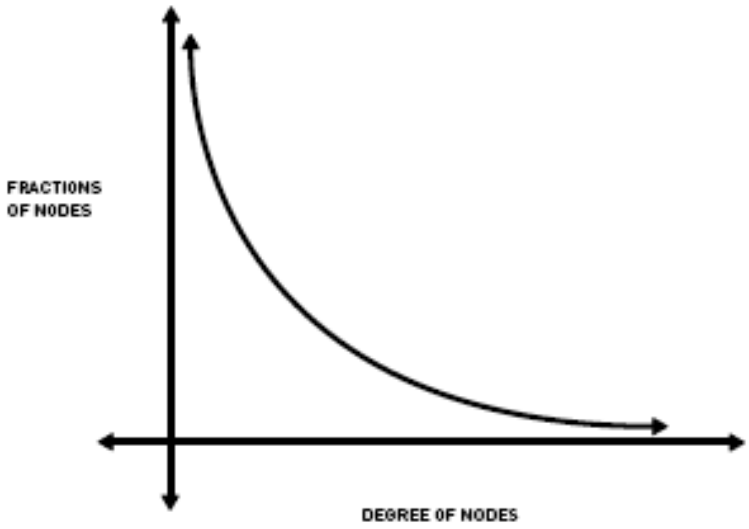
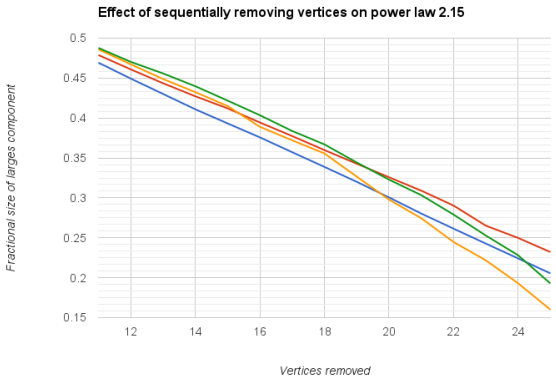
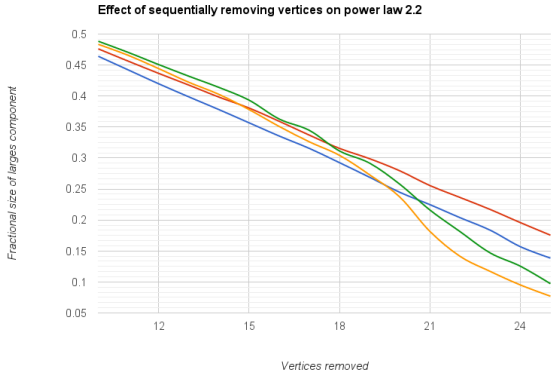


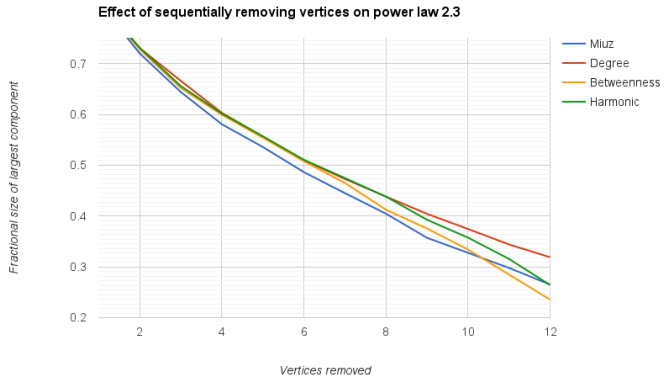
Figura 4.2: Distribución de grados power law.



(a) Evolución power law con α =2.15

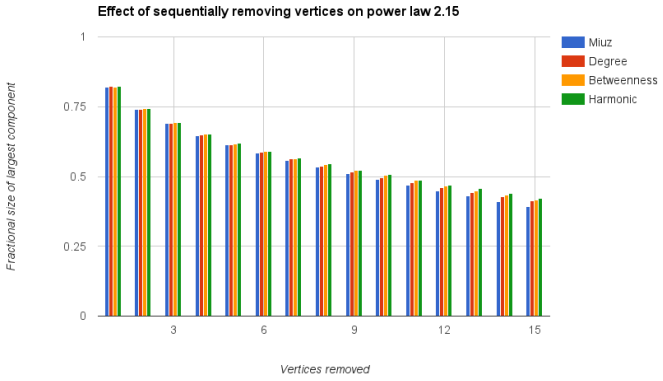


(b) Evolución power law con α =2.2

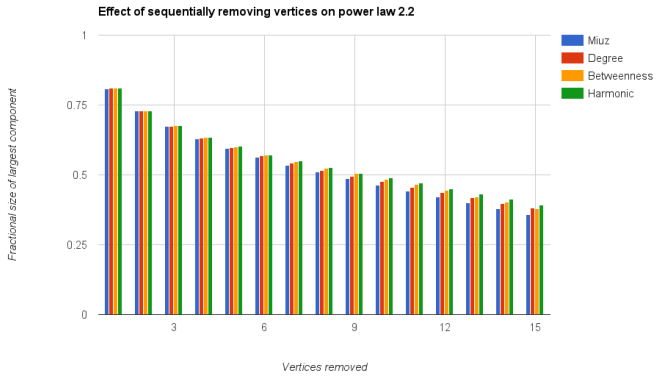


(c) Evolución power law con α =2.3

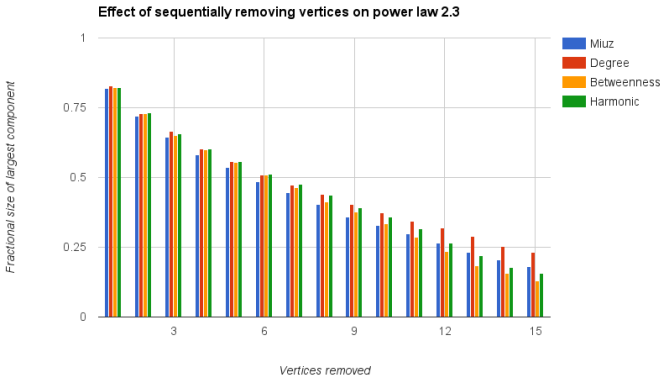
Figura 4.3: Efecto de un ataque secuencial en una red power law con $\alpha \in \{2{,}15, 2{,}2, 2{,}3\}$. En azul ataque por miuz, en amarillo ataque por betweenness, en verde ataque por harmonic, en rojo ataque por degree



(a) Evolución power law con α =2.15

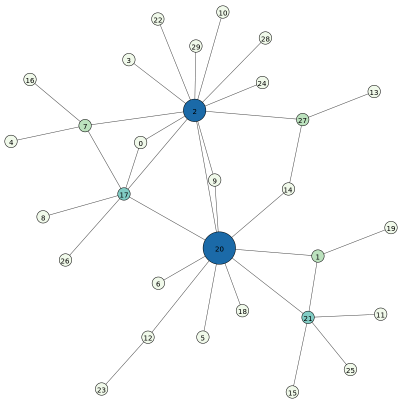


(b) Evolución power law con α =2.2

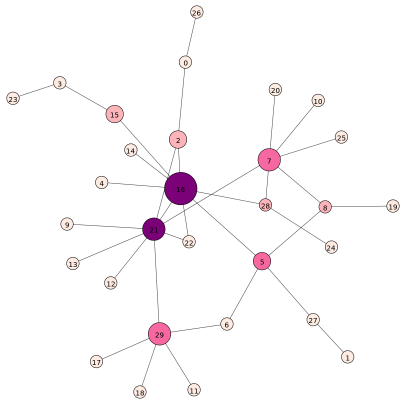


(c) Evolución power law con α =2.3

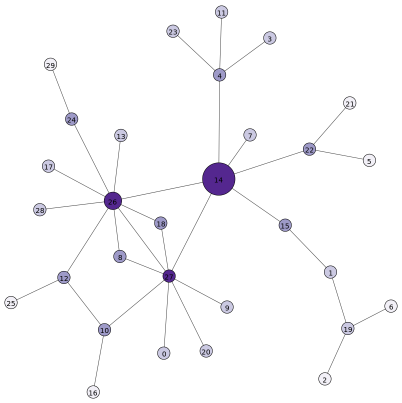
Figura 4.4: Efecto de un ataque secuencial en una red power law con $\alpha \in \{2{,}15, 2{,}2, 2{,}3\}$.



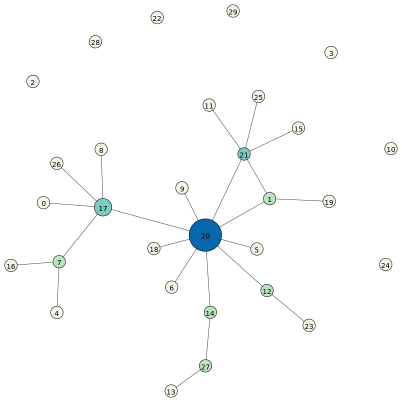
(a) Red original A, color codificado por degree centrality



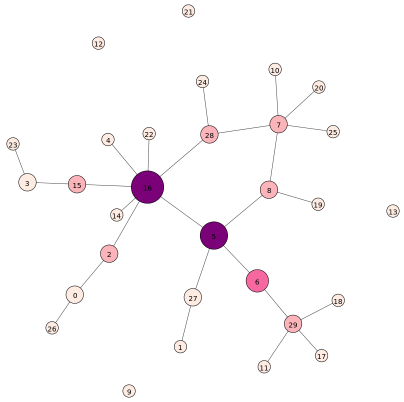
(b) Red original B, color color codificado betweenness centrality



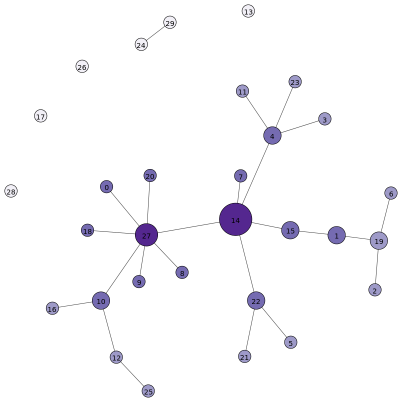
(c) Red original C, color codificado por harmonic centrality



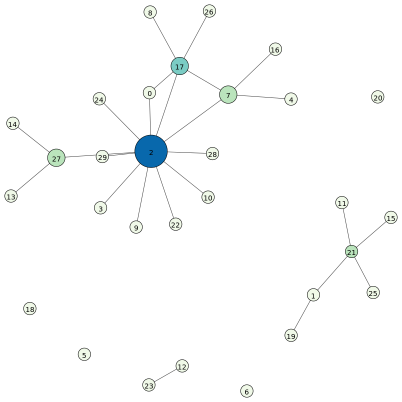
(d) Red A, luego de desconectar el nodo de mayor degree centrality



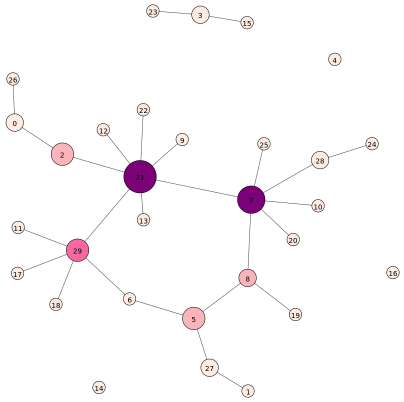
(e) Red B, luego de desconectar el nodo de mayor betweenness centrality



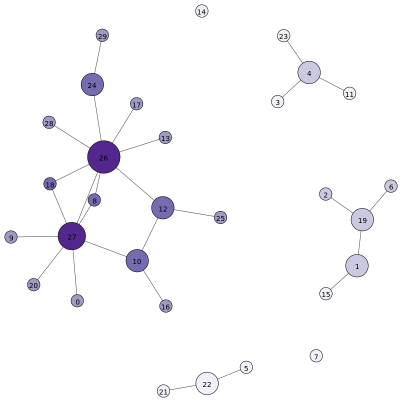
(f) Red C, luego de desconectar el nodo de mayor harmonic centrality



(g) Red A, luego de desconectar el nodo de mayor *Miuz*



(h) Red B, luego de desconectar el nodo de mayor *Miuz*



(i) Red C, luego de desconectar el nodo de mayor *Miuz*

Figura 4.5: Efecto de los diferentes ataques. El tamaño muestra que tan alto es el *Miuz* y el color refleja la medida de centralidad. Las medidas de centralidad y *Miuz* fueron recalculados luego de desconectar el nodo.