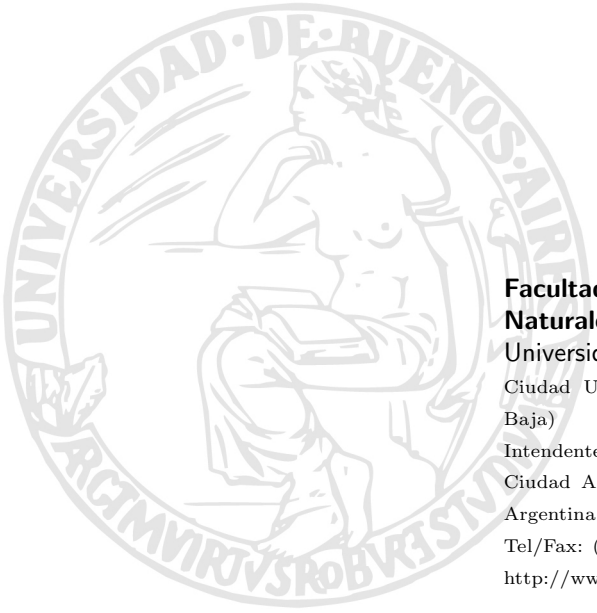
DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

TP1 - Bactracking

September 4, 2017 Algoritmos y Estructuras de Datos III

Integrante	LU	Correo electrónico
Pawlow, Dante	449/12	dante.pawlow@gmail.com



Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2610 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (++54 +11) 4576-3300

http://www.exactas.uba.ar

1 Introducción

En el presente trabajo práctico se estudia la eficiencia de la técnica algorítmica conocida como backtracking para resolver un ejercicio de optimización.

El objetivo del ejercicio es conocer el máximo número de agentes en los que se puede confiar dada una encuesta interna del grupo. En dicha encuesta cada agente puede responder si otro agente le parece confiable, desconfiable o neutral (no responde). El conjunto de agentes confiables que se debe obtener debe contar con las siguientes características: ningún agente dentro del conjunto puede decir que alguien externo es confiable ni desconfiar de alguien dentro del conjunto.

La técnica de backtracking se basa en encontrar todas las posibles soluciones válidas del problema, mediante un proceso exhaustivo. Se le puede agregar restricciones sobre los subproblemas que se deduce que no son óptimos (llamadas podas) para reducir el espectro de búsqueda de soluciones y, por lo tanto, el tiempo de cómputo.
Una poda de validez es aquella que reduce el espectro de búsqueda de posibles soluciones recortando aquellos caminos que resultan en soluciones inválidas.
Una poda de optimización, en cambio, recorta caminos que resultan en soluciones válidas, pero menos óptimas que una ya encontrada.

2 Algoritmos

Los algoritmos de backtracking con poda de esta sección se construyeron en base al de backtracking puro, por lo que comparten muchas características.

Para resolver el problema se modeló la estructura Agente que se reconoce unívocamente por su número de ID. Cada agente tiene dos conjuntos de enteros que representan los agentes en los que confía y en los que no.
Trabajar con conjuntos facilita muchas de las funciones involucradas en los algoritmos y permite una relación directa entre la abstracción con la que estamos trabajando y la implementación.

2.1 Backtracking puro

Para un agente dado existen dos posibilidades: que sea confiable o desconfiable. El algoritmo de backtracking recorre todas las posibles soluciones. Una posible solución se construye agregando un agente por vez al grupo de confiables o al de desconfiables, lo que se puede abstraer a un árbol binario de decisiones.
Este algoritmo genera una posible solución dejando de lado todas las soluciones trivialmente inválidas, por ejemplo: que un agente esté en ambos conjuntos o que un agente no pertenezca a ninguno. Además, cada vez que se agrega un agente se verifica antes de continuar que el conjunto parcial resultante sea un conjunto válido.

El algoritmo empieza con una llamada de "inicialización" a la siguiente función:

Algorithm 1: Backtracking puro

input : confiables: conjunto de agentes confiables; noConfiables: conjuntos de agentes no confiables; índice
output: número máximo de agentes en los que se puede confiar.

```
1  if ya agregué a todos los agentes then
2  |  return si es válido: el tamaño de confiables, si no: 0;
3  end
4  Aumento el índice;
5  if Agregar el agente(índice) a confiables produce un conjunto válido then
6  |  Llamo recursivamente a esta función con el nuevo confiables y el viejo
   |    noConfiables;
7  end
8  if Agregar el agente(índice) a noConfiables produce un conjunto válido
     then
9  |  Llamo recursivamente a esta función con el viejo confiables y el nuevo
   |    noConfiables;
10 end
11 Retorno el máximo de los dos caminos;
```

La forma de verificar que una solución parcial es válida es viendo que la intersección entre los agentes en los que no confía un agente del grupo de confiables y el grupo de confiables sea vacía y haciendo lo mismo con los que confía y el grupo de noConfiables.

2.2 Backtracking con poda de validez

Una poda de validez debe evitar recorrer ramas del árbol de soluciones recortando soluciones parciales que, pese a que no sean necesariamente inválidas, llevan a soluciones finales inválidas.
En este caso, la poda elegida es la siguiente: cuando se agrega un agente al conjunto de agentes confiables se verifica que los agentes en los que confía y desconfía sean consistentes con los del resto de los agentes en el conjunto. De esta manera, se evita seguir explorando una posible solución en la que, por ejemplo, dos agentes del grupo de confiables tienen opiniones distintas sobre un tercero. Esta poda se agrega a las validaciones realizadas en el backtracking puro.

Algorithm 2: Función *esConsistente()*

input : confiables: conjunto de enteros que representa a los agentes confiables; agentes: conjunto de todos los agentes
output: Respuesta booleana

```
1  if confiables es vacío then
2  |  Return verdadero
3  end
4  for por cada agente a en confiables do
5  |  for por cada agente b en confiables do
6  |  |  if a confía en agentes noConfiables de b o desconfía en los
   |  |    agentes confiables de b then
7  |  |  |  Return falso;
8  |  |  end
9  |  end
10 end
11 Return verdadero;
```

2.3 Backtracking con poda de optimización

Una poda de optimización evita recorrer los caminos que producen resultados válidos pero no óptimos.
La poda que se eligió es la siguiente: se almacena en una variable global la cantidad máxima de agentes en los que se puede confiar encontrada en un momento dado (llamada *maxConfiables*, empezando en 0). Antes de agregar un agente al conjunto de confiables se verifica si la cantidad de agentes que queda

por asignar es menor a la necesaria para sobrepasar a *maxConfiables*, se poda la rama de soluciones. Cuando se encuentra un resultado mejor que el actual, *maxConfiables* se actualiza.

Como ambas podas pueden coexistir, se agregaron juntas al algoritmo de backtracking puro. El algoritmo resultante es el siguiente:

Algorithm 3: Backtracking con ambas podas aplicadas

input : confiables: conjunto de agentes confiables; noConfiables: conjuntos de agentes no confiables; índice
output: número máximo de agentes en los que se puede confiar.

```
1  if ya agregué a todos los agentes then
2  |  return si es válido: el tamaño de confiables, si no: 0;
3  end
4  if con los agentes restantes no se llega a maxConfiables then
5  |  return 0;
6  end
7  Aumento el índice;
8  if Agregar el agente(índice) a confiables produce un conjunto válido y es
     consistente then
9  |  Llamo recursivamente a esta función con el nuevo confiables y el viejo
   |    noConfiables;
10 end
11 if Agregar el agente(índice) a noConfiables produce un conjunto válido y
     es consistente then
12 |  Llamo recursivamente a esta función con el viejo confiables y el nuevo
   |    noConfiables;
13 end
14 if el máximo retornado es mayor que maxConfiables then
15 |  Modifico maxConfiables con el nuevo valor;
16 end
17 Retorno el máximo de los dos caminos;
```

3 Método experimental y análisis de resultados

3.1 Complejidad teórica

Los algoritmos de backtracking se caracterizan por poder representarse abstractamente como una búsqueda exhaustiva sobre un árbol de soluciones. Dado que forman y recorren todo el árbol, típicamente tienen complejidad de orden exponencial $\mathcal{O}(2^n)$ o mayor.
Los algoritmos presentados parten de una base de backtracking con dos posibles caminos para cada agente (confiable o no confiable). Esto se puede representar con un árbol binario, efectivamente produciendo una complejidad mínima de $\mathcal{O}(2^n)$. Además, se debe tener en cuenta la complejidad aportada por las verificaciones de validez. Estas verificaciones realizan una comparación entre agentes, en el peor caso comparando todos contra todos, lo que produciría una complejidad $\mathcal{O}(n^2)$. Por último, se debe tomar en cuenta la cantidad de encuestas realizadas a: en peor caso se recorrerían todas, agregando un factor extra a la complejidad.

La complejidad propuesta para estos algoritmos sería entonces: $\mathcal{O}(2^n n^2 a)$.

Es importante notar que las podas no reducen la complejidad en peor caso, dado que pueden no ejecutarse. Por otro lado, sí suelen reducir el tiempo promedio de ejecución.

3.2 Complejidad empírica

Para realizar un análisis empírico de complejidad, se midieron los tiempos de ejecución de cada uno de los algoritmos para tamaños de entrada variables.

3.2.1 Encuestas aleatorias

El primer experimento que se llevó a cabo fue tomando una cantidad de agentes n de 1 a 300 y generando encuestas aleatorias para cada uno de ellos.
Las encuestas se generaron agregando una cantidad aleatoria (como máximo n) de agentes al conjunto de *confiables* de cada agente dado. De la misma manera, se generó el conjunto de *noConfiables* de cada agente, removiendo los que estuviesen presentes en *confiables*, para evitar inconsistencias.
El experimento se corrió 100 veces para cada n, generando nuevas encuestas para cada corrida. Se recopilaron datos de tiempo de ejecución para cada algoritmo, se realizó un promedio y se obtuvo la *figura 1*.

Calculando el *coeficiente de determinación* (r^2) se obtuvo que las curvas graficadas se aproximan con mayor confianza con una función polinomial. Esto parece contradecir la hipótesis inicial de que los algoritmos poseen una complejidad $\mathcal{O}(2^n)$. Además, se puede observar que los tres algoritmos no parecen demostrar una diferencia significativa en tiempos de ejecución.

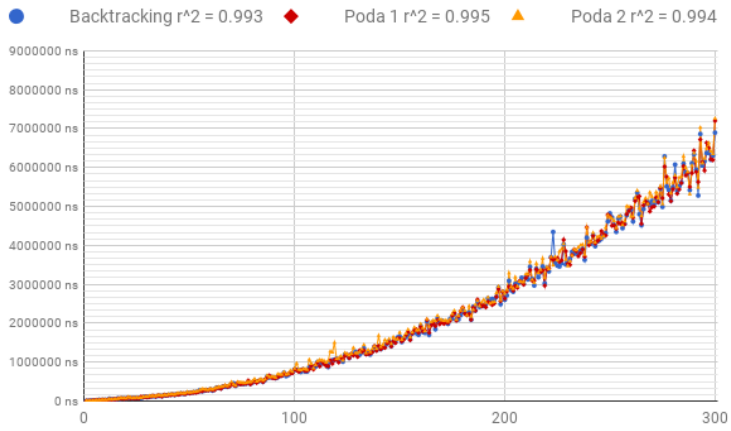


Figure 1: encuestas exclusivamente positivas.

Haciendo un análisis más detallado, se observó que los resultados (el número máximo de agentes en los que se puede confiar), rara vez superan el 1. Se plantea, entonces, la hipótesis de que nos encontramos ante un mejor caso para el algoritmo de backtracking, ocasionado por una cantidad muy grande de encuestas, lo que lleva a menos conjuntos válidos posibles y permite recortar ramas del árbol de soluciones muy rápidamente.

Teniendo esto en cuenta, se proponen los siguientes experimentos: uno con encuestas exclusivamente positivas (ningún agente desconfía de nadie) y otro con un número de encuestas acotadas para cada agente.

3.2.2 Encuestas exclusivamente positivas

Para este experimento se procedió de manera muy similar al anterior, pero evitando que se generen encuestas negativas. Los resultados se graficaron en la *figura 2* y se observó que un ajuste polinómico (en particular, cúbico) presentaba el coeficiente de determinación más alto. Esto se podría atribuir a que nos encontramos frente a un caso borde, dado que no hay encuestas negativas.

Lo más interesante de este caso, es que el backtracking puro tuvo tiempos significativamente mejores que los backtracking podados. Esto probablemente se dé porque los backtracking podados realizan una mayor cantidad de verificaciones sobre los datos, entonces pueden existir casos en los que el backtracking puro tenga un mejor rendimiento.

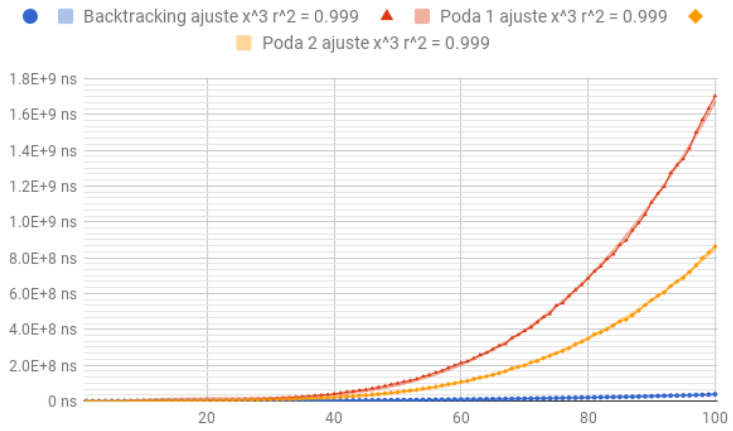


Figure 2: encuestas aleatorias.

3.2.3 Cantidad de encuestas restringida a $\frac{1}{5}$ de n

En este experimento las encuestas se generaron de manera aleatoria, similarmente a como se generaron en *3.2.1.* La diferencia radica en que la cantidad máxima de individuos sobre los cuales puede opinar un agente se redujo a $\frac{1}{5}$ de la cantidad total.

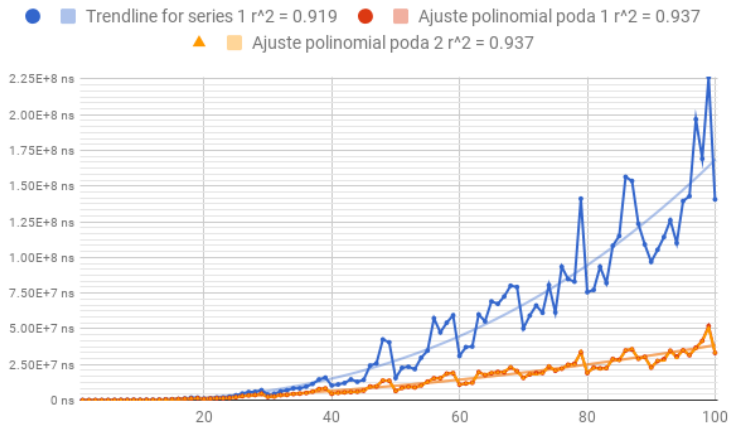


Figure 3: encuestas aleatorias restringidas.

Como se puede observar en la *figura 3*, los tiempos son menores que en el caso *3.2.2*, aunque con una distribución de datos mucho menos uniforme. Se observa, además, que la poda 1 y 2 presentan un tiempo promedio prácticamente idéntico.

Se sigue presentando, como en los casos anteriores, una tendencia al ajuste polinómico, aunque en este caso el coeficiente de determinación es mucho menos significativo.

3.2.4 Cantidad de encuestas restringida a $\frac{1}{25}$ de n

Por último, se restringieron las encuestas máximas de cada agente a un número aleatorio entre 1 y $\frac{1}{25}$ de la cantidad total de agentes. Esta vez se tomó un n de 1 a 40, porque los tiempos de ejecución eran significativamente mayores que en los casos anteriores. Los resultados se graficaron en la *figura 4*.

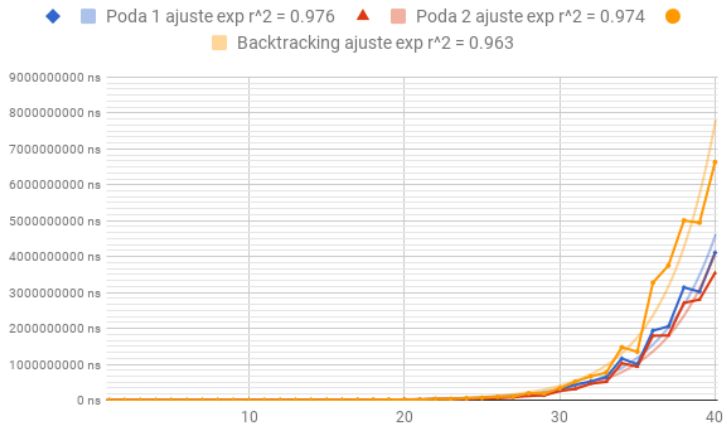


Figure 4: encuestas aleatorias aún más restringidas.

Como se puede observar, para este caso de prueba el ajuste más apropiado es el exponencial. Para mayor certeza, se graficaron los datos en escala logarítmica en la *figura 5*, resultando en un ajuste lineal. Esto confirma las hipótesis de que los algoritmos estudiados tienen una complejidad al menos $\mathcal{O}(2^n)$.

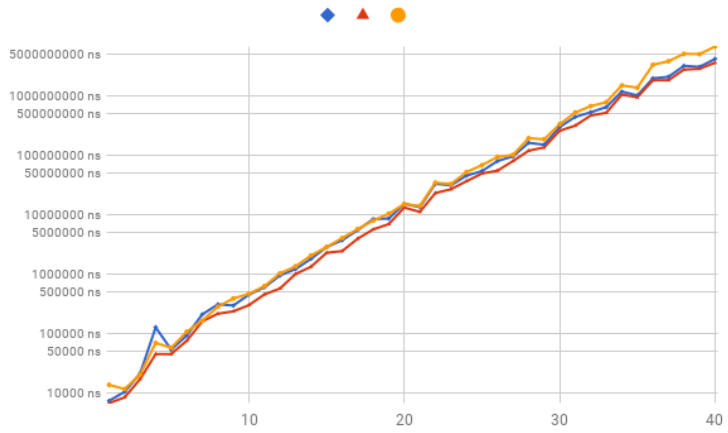


Figure 5: encuestas aleatorias aún más restringidas, escala logarítmica.

4 Conclusiones

A lo largo de este proceso experimental surgieron las siguientes hipótesis:

1. Los algoritmos presentados presentan complejidad exponencial.
2. Los algoritmos de backtracking podados son más eficientes que los de backtracking puro.

Por un lado se pudo comprobar que los algoritmos tienen complejidad (al menos) exponencial. Esto se pudo observar en el caso de prueba *3.2.4*.
Pese a esto, se observó que en muchos casos, los algoritmos tienen un rendimiento similar al de una complejidad polinómica. Esto puede darse porque hay casos bordes que presentan una complejidad mucho menor o porque el caso promedio del algoritmo es considerablemente mejor que el peor caso.

Por otro lado, se pudo ver en el caso *3.2.2* que aplicar una poda no necesariamente mejora el tiempo de ejecución de un algoritmo, e incluso puede pasar lo contrario.
Esto ocurre porque las podas pueden agregar procesos computacionalmente costosos y, aún así, tener que recorrer todo el árbol de soluciones. Las podas no reducen la complejidad en peor caso, peor pueden ayudar a reducir significativamente el tiempo de ejecución en el caso promedio (como se puede observar en el caso *3.2.3*.