Índice general

1.. Introducción 1
1.1. Descripción 1
1.2. Primeros pasos 1

2.. Desarrollo 3
2.1. Definición de similitud y complementariedad 3
2.1.1. Papers 3
2.1.2. Autores 3
2.2. Generación de bundles 3
2.3. Selección de bundles 3
2.3.1. Selección golosa 4
2.3.2. Selección de a pares 4
2.3.3. Selección proporcional 4

3.. Resultados 6
3.1. Comparando soluciones 6
3.2. Papers 6
3.3. Autores 7

4.. Conclusiones 9

1. INTRODUCCIÓN

1.1. Descripción

Con los resultados obtenidos del paper *"Composite Retrieval of Diverse and Complementary Bundles"* se decidió implementar búsquedas de soluciones para la base de datos *"A Data-Driven Journey through Software Engineering Research"*

1.2. Primeros pasos

Todo Pasa.
Julio G.

Se utilizó la base de datos de *"A Data-Driven Journey through Software Engineering Research"* con el objetivo de obtener conjunto de bundles complementarios. La base de datos contiene información sobre cerca de 7777 papers y sus 9865 autores, además cuenta con un perfil para cada paper, llamado `topicProfile` que ofrece una clasificación en porcentajes de que tema se refiere cada paper. Los temas son:

- Adaptive Systems
- Algorithm
- Architectures
- Artificial Intelligence
- Autonomic Systems
- Concurrency
- Database
- Distributed Systems
- Education
- Embedded
- Empirical Software Engineering
- Evolution
- Formal Methods
- Hardware
- Human Computer Interaction
- Information Systms
- Knowledge Engineering
- Languages
- Maintenance
- Methodologies
- Metrics
- Models
- Operating Systems
- Performance
- Process
- Product Lines
- Program Analysis
- Program Comprehension
- Real Time Systems
- Reliability
- Requirements
- Reverse Engineering
- Security
- Services
- Software Quality
- Synthesis
- Testing
- Visualization

La generación de soluciones de bundles se realizó, como propone el paper *"Composite*

Retrieval of Diverse and Complementary Bundles", maximizando la función objetivo:

$$\sum_{1\leq i\leq k}\sum_{u,v\in S_i}\gamma s(u,v)) \; + \sum_{1\leq i\leq j\leq k}(1-\gamma)(1-\max_{u\in S_1, v\in S_j} s(u,v))$$

Dónde la función s representa la similitud entre dos items y γ $(0<\gamma<1)$ es un parámetro para ponderar entre la calidad de un bundle (intra) y la separación entre ellos (inter).
A partir de información provista por la base de datos y con la definición de funciones de similitud, que se verán más adelante, tanto para autores y papers se generaron conjuntos de bundles para los siguientes criterios:

- Papers de tópicos similares que se presentaron en conferencias de distintos lugares.
- Autores similares de distintos lugares.

2. DESARROLLO

2.1. Definición de similitud y complementariedad

Como paso preliminar a la búsqueda de las soluciones se definió la noción de similitud y complementariedad para los grupos de elementos.

2.1.1. Papers

La similitud entre papers se definió como la distancia entre los `topicProfile` de cada uno. Cada perfil se interpreta como un vector de `n` posiciones, dónde cada posición pertenece a un tópico particular. Entonces la similitud entre dos papers la interpretamos como el ángulo vectorial entre dos vectores.
La complementariedad de dos papers se definió al lugar de presentación de dicho paper.

2.1.2. Autores

Para determinar la similitud entre los autores se creó un perfil de cada uno. Para lograrlo lo primero que se hizo fue tener en cuenta todos los perfiles de papers en los que participaron cada uno de ellos. Al igual que con los papers, el perfil de los autores se representó con un vector. Este vector se cálculo sumarizando los vectores de cada uno de los papers en el que participó.
El paso siguiente fue determinar la similitud de los autores, la primer aproximación fue similar al cálculo de similitud de los papers y calculamos los ángulos entre todos los vectores. Como similitud alternativa decidimos restar el perfil de cada autor componente a componente y calcular el valor de su norma, así de esta forma a mayor norma mayor similitud entre autores.
La complementariedad de dos papers se definió al lugar de pertenencia del autor en cuestión.

2.2. Generación de bundles

Para la generación de bundles se usaron diferentes algoritmos, cada uno con sus ventajas.
El algoritmo jerárquico `HAC` comienza con cluster conformado por un solo ítem y en cada paso une los dos clusters que maximizan su similitud y que respeten las restricciones, que en estos casos fueron no superar una cierta cantidad de elementos por bundles.
El algoritmo `BOBO-x` genera los clusters a partir de un ítem, llamado pivote, y luego de forma golosa selecciona los ítems que maximizan la función intra-cluster. La x representa el número de bundles que va a generar
El algoritmo `BOBO-Ex` es similar a `BOBO-x` pero prueba con todos los ítems como pivots para la generación de bundles.

2.3. Selección de bundles

Se utilizaron tres estrategias de selección de bundles para generar las soluciones finales.

2.3.1. Selección golosa

Para la selección de los bundles se utilizó un algoritmo goloso que comienza seleccionando el cluster con mayor valor intra y en cada iteración se elige el cluster que maximiza la función objetivo.
Durante las primeras pruebas notamos que todas las soluciones contenían varios bundles en común teniendo en cuenta que se utilizaron distintos valores gammas. Supusimos que esto se debía a que en el momento de hacer la selección en todos los casos, sin importar el gamma, se seleccionaba el mismo bundle (que es el de mayor intra). Al siguiente paso al seleccionar el bundle que maximiza la función objetivo, ocurre que el valor intra del bundle es mucho más significativo que el inter. Por lo tanto, lo que ocurre es que en los primeros pasos se seleccionan los mismos bundles para todos los gammas sin tener en cuenta que cuando se optó por un gamma cercano a cero se pedía que la solución sea más diversificada.

2.3.2. Selección de a pares

Para mejorar el proceso de selección se decidió modificar el algoritmo para que en vez de realizar la selección de a un bundle, se generan pares de bundles y el algoritmo goloso selecciona el que maximiza la función objetivo.

Algorithm 1 Selección de bundles de a pares

Require: Aca va la entrada del algoritmo.
Ensure: Aca va la salida del algoritmo.

```
1: while alguna conodicion do
2:    Descripcion con palabras.
3:    if alguna conodicion copada then
4:       return devuelve algo
5:    else
6:       alguna asignacion o algo
7:    end if
8: end while
9: for alguna otra condicion do
10:   if alguna conodicion then
11:      hace algo
12:      return devuelve algo
13:   end if
14: end for
15: return devuelve otra cosa
```

2.3.3. Selección proporcional

Además como tercera opción de selección se implementó un algoritmo proporcional que en cada paso se ponderan los resultados de la función que calcula el intra y el inter restante, intentando “adivinar ”el valor de las próximas iteraciones y de esta manera dar más importancia en los primeras iteraciones al valor del inter.

Algorithm 2 Selección de bundles proporcional

Require: Aca va la entrada del algoritmo.
Ensure: Aca va la salida del algoritmo.

```
1: while alguna conodicion do
2:     Descripcion con palabras.
3:     if alguna conodicion copada then
4:         return devuelve algo
5:     else
6:         alguna asignacion o algo
7:     end if
8: end while
9: for alguna otra condicion do
10:    if alguna conodicion then
11:        hace algo
12:        return devuelve algo
13:    end if
14: end for
15: return devuelve otra cosa
```

3. RESULTADOS

3.1. Comparando soluciones

Para comparar las la calidad de las distintas soluciones, además del valor objetivo se compara con la cantidad de elementos iguales en toda la solución y la cantidad igual de elementos para cada bundle. De esta manera se observa que tan "parecidas"son las soluciones y con más detalle que tan "parecidos"son los bundles. Entre los diferentes algoritmos buscamos la cantidad de elementos iguales en cada una de las soluciones y también la cantidad de elementos iguales por bundle.

3.2. Papers

Originalmente la base de datos contenía unos 7777 papers, de los cuáles se tuvo que hacer una depuración, ya que había papers que no tenían ningún autor asociado o perfil creado. Luego de la depuración obtuvimos 4937 que cumplen los requisitos para la búsqueda de las soluciones.
Se genraron soluciones con las siguientes características:

Descripción:

- **Algoritmo de selección:** simple, por tuplas y proporcional
- **Algoritmo de generación:** `HAC` y `BOBO-x`, con $x \in (10, 160)$
- **γ:** $\in (0,1; 0,3; 0,5; 0,7; 0,9)$
- **Cantidad de bunldes:** 10
- **Cantidad de items por bundles:** 5

Como primera observación podemos ver la cantidad de bundles que se generan para cada algoritmo de producción y su tiempo de ejecución:
A continuación se muestran los valores de la función objetivo obtenidos:

Algoritmo	Bundles Generados	Tiempo de Ejecución (minutos)
HAC	2378	6
BOBO-10	100	2
BOBO-160	1600	5

Tab. 3.1: Tabla 1

Al comparar `BOBO-10` y `BOBO-160` usando la selección simple se observa que la diferencia entre los valores de la función objetivo de cada solución aumenta a medida que γ se acerca a 1. Suponemos que esto se debe a la estrategía `produce and choose`. El objetivo de la primer etapa es producir bundles con máximo valor de similitud. En el caso de que se requiera una solución con mayor separación entre bundles, al haber producido menos cantidad de bundles existen menos posibilidades para generar una solución más dispersa. Un mismo análisis se podría hacer si comparamos `BOBO-10` y `HAC`.

Algoritmo	γ 0,1	0,3	0,5	0,7	0,9
HAC	48,8922	58,7049	70,205	81,8331	93,7189
BOBO-10	30,5376	26,68	22,9482	23,2333	21,9347
BOBO-160	35,1979	34,4164	37,0669	40,1762	44,9824

Tab. 3.2: Tabla 2

ANALIZAR `BOBO-160` y `HAC`
Comparando los resultados obtenidos al realizar la selección de a un candidato contra la selección de a pares obtuvimos que para `BOBO-160` y `HAC` los tiempos aumentaron a 40 minutos y para `BOBO-10` a 2 minutos. En cuanto al valor de la función objetivo el único beneficiado fue `BOBO-10` ya que para `HAC` empeoró y para `BOBO-160` el aumento fue muy pequeño en comparación al incremento de tiempo.

3.3. Autores

Se genraron soluciones con las siguientes características:

Descripción:

- **Algoritmo de selección:** simple y proporcional
- **Algoritmo de generación:** `HAC` y `BOBO-x`, con $x \in (10, 160)$
- **γ:** $\in (0,1; 0,3; 0,5; 0,7; 0,9)$
- **Cantidad de bunldes:** 10 y 20
- **Cantidad de items por bundles:** 5 y 10

		Valor función objectivo		Duración de la ejecución (minutos)	
Algoritmo	gamma	Selección simple	Selección proporcional	Selección simple	Selección proporcional
HAC	0,1				
HAC	0,3				
HAC	0,5				
HAC	0,7				
HAC	0,9				
BOBO-160	0,1				
BOBO-160	0,3				
BOBO-160	0,5				
BOBO-160	0,7				
BOBO-160	0,9				

Tab. 3.3: Tabla 3

Para todas las soluciones obtenidas con la generación `HAC` y utilizando cualquiera de los dos algoritmos de selección de selección, todas las soluciones formaron los mismos bundles a pesar de que el γ sea distinto.
Se realizaron otras búsquedas de soluciones, excluyendo a cinco de los autores que están presentes en todas las soluciones. En las nuevas soluciones obtenidas se repite nuevamente

el comportamiento que para todos los γ, con las soluciones idénticas y valores de la función objetivo coinciden con los de las soluciones obtenidas sin excluir autores.
Con el algoritmo `HAC` todas las soluciones fueron idénticas para todos los γ. Esto se debe a que existen 40000 relaciones de similitud con valor uno. Con la heurística Produce and Choose, al momento de producir no se tiene en cuenta el γ por lo tanto para todos los γ en la etapa de producción se producen los mismos bundles.
Por otro lado todas las soluciones generadas por el algoritmo `HAC` prácticamente no comparten bundles similares con las demás soluciones, ni siquiera autores similares en toda la solución.

4. CONCLUSIONES