Proyecto Final

Genetic Working Space

Alumnos

Ducret, Argentino (52194)

Gutierrez, Ignacio (54293)

Resumen

Genetic Working Space es una solución con algoritmos genéticos al problema de elegir una distribución de asientos para un conjunto de empleados, sujeto a restricciones. Para su implementación se utilizó un motor de algoritmos genéticos desarrollado en Java.

Los conceptos importantes vinculados a este problema son los proyectos, los empleados y el espacio de trabajo. El modelado del individuo consiste en un array de tamaño #proyectos que determina un orden para los proyectos, y una matriz de (#proyectos)x(#proyectos) que determina un nivel de afinidad entre proyectos. Dichas estructuras de datos permiten transformar de forma determinística un espacio de trabajo y un conjunto de personas en una asignación de asientos. La asignación de asientos puede ser luego utilizada para calcular el fitness. La cruza y la mutación consisten en transformaciones de las 2 estructuras de datos mencionadas.

Se realizaron pruebas con nombres de fantasía basándose en empresas, proyectos y espacios de trabajo reales. Los resultados obtenidos sugieren que los algoritmos genéticos son una herramienta prometedora para solucionar el problema planteado.

Introducción

El objetivo del proyecto final es implementar un algoritmo genético que dado una nómina de empleados con sus respectivos proyectos o tareas y una estructura jerárquica del lugar físico donde se desempeñarán, se logre asignarles un espacio de trabajo de una forma eficiente teniendo en cuenta la cercanía entre trabajadores de un mismo proyectos.

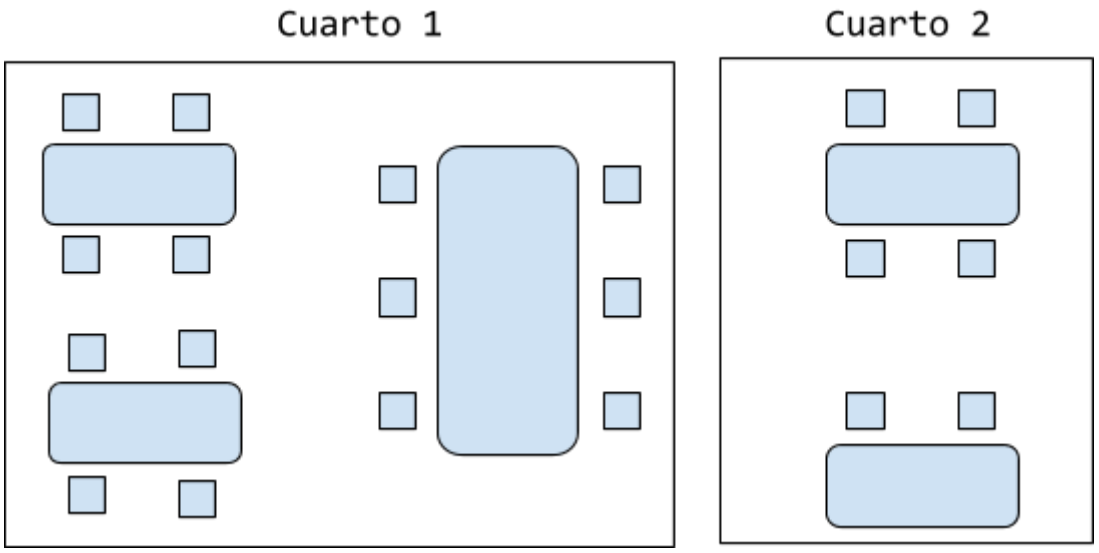
Este es un problema recurrente en empresas que tienen un flujo grande de entrada y/o salida de personal, teniendo así que utilizar horas de trabajo cada cierto corto periodo de tiempo para buscar una arquitectura de trabajo cómoda para los miembros de la empresa que se encuentran en la misma área, proyecto, tareas, etc, puedan estar cerca entre sí. Lo cual no es una tarea sencilla, ya que el tamaño de los grupos pueden variar, el tiempo de vida de cada proyecto es diferente, al alcanzar el límite de personas se deben expandir y toda la estructura cambia, entre otros problemas.

Nos inspiramos en Wolox, empresa de desarrollo de software con un gran crecimiento, que fue el lugar donde ambos trabajamos y ocurre el problema antes mencionado volviéndose cada vez más difícil realizar una asignación de lugares eficiente debido a la rápida ampliación.

En las siguientes secciones se detallara un problema de ejemplo para familiarizarse con la problemática, el proceso de desarrollo o implementación, estructuras y algoritmos utilizados, solución de problemas, cálculo del fitness y por último se analizará y detallaran resultados del algoritmo. Además se detallarán los problemas encontrados a lo largo del proyecto y se finalizará con las conclusiones correspondientes al rendimiento del algoritmo desarrollado.

Problema ejemplo

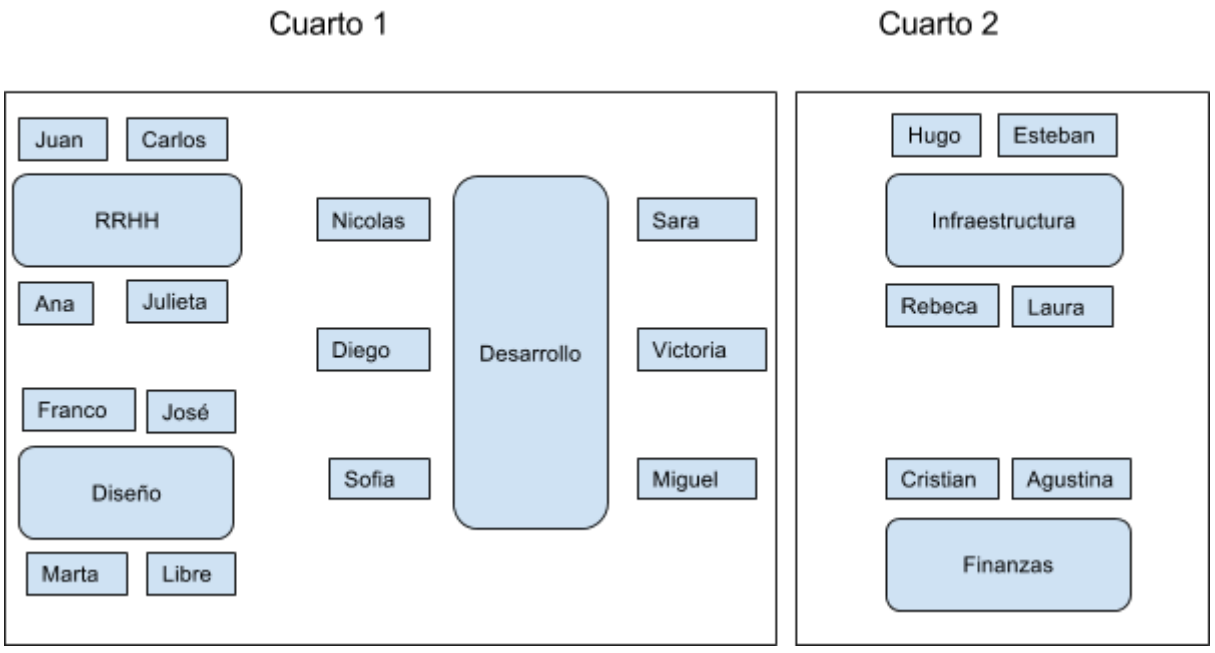
Supongamos que tenemos una empresa cuya arquitectura consiste de dos cuartos, el primero con dos mesas con cuatro puestos de trabajo y una mesa con seis puestos, por otro lado el segundo cuarto tiene una mesa de cuatro lugares y otra de solo dos como se puede ver en la imagen.



Además, esta empresa cuenta con 19 empleados los cuales están distribuidos en diferentes proyectos como se detalla a continuación:

1. Juan: RRHH
2. Carlos: RRHH
3. Ana: RRHH
4. Julieta: RRHH
5. Cristian: Finanzas
6. Agustina: Finanzas
7. Nicolas: Desarrollo
8. Diego: Desarrollo
9. Sofia: Desarrollo
10. Sara: Desarrollo
11. Victoria: Desarrollo
12. Miguel: Desarrollo
13. Hugo: Infraestructura
14. Laura: Infraestructura
15. Rebeca: Infraestructura
16. Esteban: Infraestructura
17. Franco: Diseño
18. José: Diseño
19. Marta: Diseño

Una de las posibles soluciones podría ser la siguiente:



Si bien este ejemplo es sencillo de resolver, ya que los equipos entran sin problema en las mesas se espera que el algoritmo logre resolver arquitecturas de este estilo sin problemas, y otras más complejas con solapamiento de proyectos y con espacios ajustados encontrando una de las soluciones óptima o una cercana a alguna de estas.

Implementación

El proyecto se realizó utilizando Java y el gestor de dependencias Maven, a su vez se incluyeron las siguientes dependencias:

1. xchart: Librería sencilla y liviana para graficar información. Se utilizada para brindar información sobre mayor fitness por generación, población promedio y fitness promedio.
2. jgraphx: Esta dependencia se encarga de realizar todo el manejo de grafos y la visualización de los mismos.

A continuación se detalla en pseudo-codigo la estructura del algoritmo genetico:

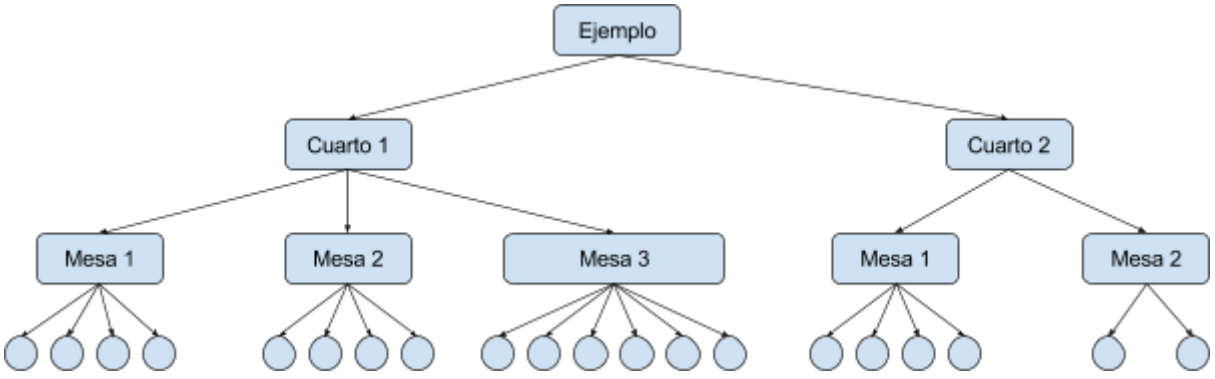
```
Generar una población inicial
while No se cumpla la condición de corte do
       Seleccionar individuos para la reproducción
       Cruzar dichos individuos
       Mutar o no a aquellos hijos
       Generar nueva población mediante reemplazo
end
```

En las siguientes secciones se las estructuras utilizadas para representar a un cromosoma y los métodos utilizados por el algoritmo genético.

Estructura arquitectura de trabajo

La representación de la estructura de la empresa se modelo con una árbol N-ario, en el cual las hojas representan al espacio de trabajo y los nodos intermedios representan la jerarquía de la arquitectura de la empresa.

El ejemplo mencionado anteriormente se visualizaría de la siguiente forma:



Una de las ventajas de esta representación es que cada hoja queda identificada de forma única al utilizar el camino desde la raíz hasta la hoja como ID. Por ejemplo: “Ejemplo-Cuarto

1-Mesa 3-Asiento 4”, la única restricción es que el nombre de los espacios de trabajo no se repitan en una misma mesa.

En este ejemplo, se definió una semántica para la jerarquía de la empresa pero en otros casos los nodos intermedios pueden representar otra estructura como edificios, pisos, departamentos, entre otros.

Además, existe la posibilidad de que las hojas no estén a la misma altura supongamos que en el ejemplo anterior una mesa se divide en lado derecho y lado izquierdo agregando un nivel más de profundidad en el árbol.

Archivos para generar la arquitectura de trabajo

En esta sección se detallara como es el formato de los archivos utilizados para indicarle al algoritmo como interpretar la estructura de la empresa a estudiar y cómo se distribuyen sus empleados en los diferentes proyectos.

Archivo de organización

Este archivo es el encargado de indicar la representación de la empresa a nivel físico, en otras palabras cuantos edificios, cuartos, mesas, espacios de trabajo, entre otros le corresponden a la empresa.

Reglas a tener en cuenta:

- Se utilizó un formato donde cada línea representa un nodo del árbol al cual se le puede dar cualquier nombre.
- Las líneas que tengan la misma cantidad de tabs antes del nombre del nodo indica que esos nodos son hermanos.
- En el caso de tener un tab más que el nodo de la línea anterior indica que este nodo es hijo.
- El archivo debe tener un nodo raíz.

El archivo correspondiente al ejemplo antes visto es el siguiente:

```
Ejemplo
   Cuarto 1
      Mesa 1
         Silla 1
         Silla 2
         Silla 3
         Silla 4
      Mesa 2
         Silla 1
         Silla 2
         Silla 3
         Silla 4
```

```
      Mesa 3
         Silla 1
         Silla 2
         Silla 3
         Silla 4
         Silla 5
         Silla 6
   Cuarto 2
      Mesa 1
         Silla 1
         Silla 2
         Silla 3
         Silla 4
      Mesa 2
         Silla 1
         Silla 2
```

Archivo de empleados

Finalmente el archivo utilizado para indicar los empleados de la organización cuenta con el siguiente formato:

- Una línea por empleado indicando el nombre del mismo y sus proyectos.
- Los proyectos se indican luego del nombre del empleado y deben comenzar con #.
- Los nombres de los empleados no se pueden repetir.

El siguiente archivo representa a los empleados del ejemplo visto anteriormente.

```
Juan #RRHH
Carlos #RRHH
Ana #RRHH
Julieta #RRHH
Cristian #Finanzas
Agustina #Finanzas
Nicolas #Desarrollo
Diego #Desarrollo
Sofia #Desarrollo
Sara #Desarrollo
Victoria #Desarrollo
Miguel #Desarrollo
Hugo #Infraestructura
Laura #Infraestructura
Rebeca #Infraestructura
Esteban #Infraestructura
Franco #Diseño
José #Diseño
```

```
Marta #Diseño
```

Estructura individuo

La estructura de un individuo o cromosoma se conforma principalmente de tres elementos:

1. **Orden**

Representa el seguimiento en el cual se les asignarán a las personas de un mismo proyecto sus puestos de trabajo. Está representado por un array de `int`, ya que se identificó de forma unívoca a los proyecto con un número entero positivo.

Es importante notar, que a los primeros proyectos en dicho array se les podrán asignar una gama más amplia de espacios de trabajo, pues una vez que se asigna un asiento o puesto este queda ocupado imposibilitando el uso del mismo para los siguientes proyectos. En apartados posteriores se detallara el algoritmo utilizado para ubicar a los proyectos en el espacio físico que les corresponda a cada uno.

2. **Afinidad**

Además del orden, es importante establecer si se tiene la necesidad de posicionar cerca a dos proyectos para esto se utiliza la afinidad. Se la implementó utilizando una matriz de int de `n x n` (n denota la cantidad de proyectos), donde la fila `i-ésima` se puede interpretar como las afinidades ordenas de forma decreciente del proyecto `i` (número de fila) con el resto de los proyectos incluyéndose a sí mismo. Por ejemplo, dada la fila 2 de la matriz de afinidades `[1, 2, 3, 4]` significa que el proyecto 2 tiene mayor afinidad con el 1, luego con el 2, 3 y 4.

Este concepto de afinidad se utiliza a la hora de asignar lugares de trabajo a un proyecto en particular. Primero como ya se explicó se utiliza el orden para establecer a cuáles se “sentaran” primero luego la afinidad indica cerca de qué proyecto se tiene que ubicarlo. Esto no aplica para el primer proyecto a posicionar (ya que todos los lugares de trabajo están libres) o si la mayor afinidad la tiene con sigo mismo.

3. **Restricciones**

Por último, las restricciones son un mapa donde las keys denotan a los proyectos y los valores almacenados son las personas que trabajan en dicho proyecto.

Estos valores solo se utilizan a la hora de calcular el fitness de un individuo lo cual se explicará en apartados posteriores.

Algoritmo para sentar personas

Sirve para determinar qué asiento termina ocupando cada persona, tomando como input:

- archivo de organización
- archivo de empleados

- array de orden
- matriz de afinidades

3Inicialmente se genera el árbol que representa el espacio de trabajo a partir del archivo de organización. Luego se consiguen las personas y los proyectos a los que pertenecen a partir del archivo de empleados. El algoritmo sienta a los proyectos en orden, según indica el array de orden. Por cada proyecto busca las mejores opciones para sentarlo (nodos a partir de los cuales alcanza el espacio y se aprovecha bien, es decir, se minimiza la cantidad de asientos vacíos que que quedan a partir del nodo elegido). De las distintas opciones que se encuentran se tiene que elegir 1 sola, y para hacerlo primero se fija si de los proyectos con el cual el actual (el que está siendo sentado) tiene afinidad, alguno ya fue sentado. Si hubiere, toma el de mayor afinidad y lo usa como referencia. Entre las distintas opciones que teníamos antes tomamos la que minimiza la distancia al nodo de referencia, y es allí donde se sienta el proyecto. La forma en la que se sienta el proyecto es consultando las hojas disponibles que existen a partir del nodo elegido. Todas las personas del proyecto a sentar son sentados en sentido de lectura en las hojas disponibles. Este proceso se repite por cada proyecto y el final del algoritmo obtenemos un mapeo (persona -> hoja) que respeta el array de orden y de afinidades.

Criterio de corte

Los criterios de corte que se utilizaron son cuatro, y basta con que uno se cumpla para dar por terminado la ejecución del algoritmo. Los mismos son los siguientes:

1. Límite de generaciones: El algoritmo termina al alcanzar el número de generaciones límite.
2. Fitness deseado: Al obtener un cromosoma con el fitness deseado se finaliza el proceso.
3. Por estructura: Cuando la siguiente generación no evoluciona un porcentaje suficiente se detiene la búsqueda.
4. Por contenido: Si luego de n generacion no se logro superar el mejor fitness se devuelve este máximo como resultado.

Métodos de selección

Los métodos de selección reciben como parámetro una lista de individuos y un valor k, retornando k individuos seleccionados según el método utilizado.

A continuación se listan las selecciones disponibles, para más información ver detalle en el anexo.

1. Elite
2. Ruleta
3. Universal
4. Boltzmann
5. Torneo determinístico
6. Torneo probabilístico

7. Ranking
8. Aleatorio

Método de apareamiento

Para seleccionar los pares de padres para luego realizar la cruza, se utiliza un algoritmo estocástico de pairing. El cual consiste en formar parejas sin repetir los individuos, en otras palabras ningún individuo seleccionado se va a aparejar más de una vez.

Métodos de cruza

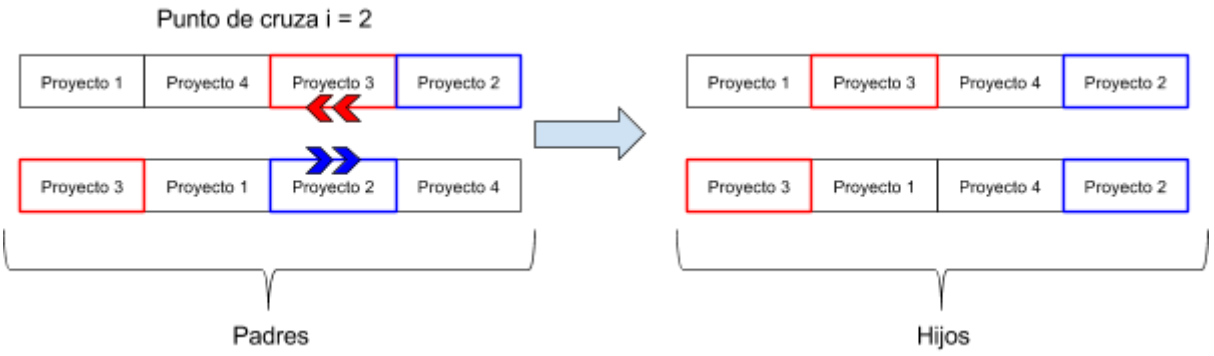
Se tuvo que implementar un método de cruza adaptado para este problema, ya que los formatos tradicionales no se corresponden a este problema. En secciones anteriores se explicó cómo se modeló la estructura de un individuo, dicho esto podemos pensar la cruza como un intercambio de información entre dos individuos, para realizar este intercambio se tiene que tener en cuenta tanto el orden como la afinidad.

Dado dos cromosomas se eligen de forma aleatoria índices uno para el array de orden y otros n para cada fila de la matriz de afinidades. Luego el siguiente proceso se repite para cada vector (el de orden y para cada fila de afinidad):

Dados dos cromosomas se aplica el mismo proceso para su array de orden y para cada vector que conforma la matriz de afinidades, siendo el proceso aplicado el siguiente:

1. Se elige de forma aleatoria un índice para realizar el intercambio de información sobre los valores i-ésimos de los vectores de los padres.
2. Se obtiene del array 1 el valor correspondiente al índice calculado.
3. En el array 2, se busca el índice correspondiente al valor recuperado en el paso anterior. (Notar que ambos arrays tendrán los mismos valores pero probablemente en diferente orden)
4. Se mueve el valor del array 1 hacia el lado en que se encuentra el valor en el array 2 (o no se realiza ningún movimiento si estos dos valores coinciden en el mismo índice). Entiéndase mover como intercambiar o shiftear dos elementos consecutivos en un array hacia la izquierda o derecha según corresponda.

En el siguiente gráfico se detalla un ejemplo sencillo de cómo se realiza la cruza implementada, solo para el array de orden el mismo proceso se realiza de igual forma para las filas de la matriz de afinidades.



En este algoritmo de cruza se intenta aprovechar la información valiosa que tiene el otro individuo, es decir si un cromosoma prioriza el orden o afinidad de cierto proyecto tal vez si el otro intentar copiar este comportamiento (shifteando el proyecto un lugar hacia derecha o izquierda) podría mejorar su fitness.

Métodos de mutación

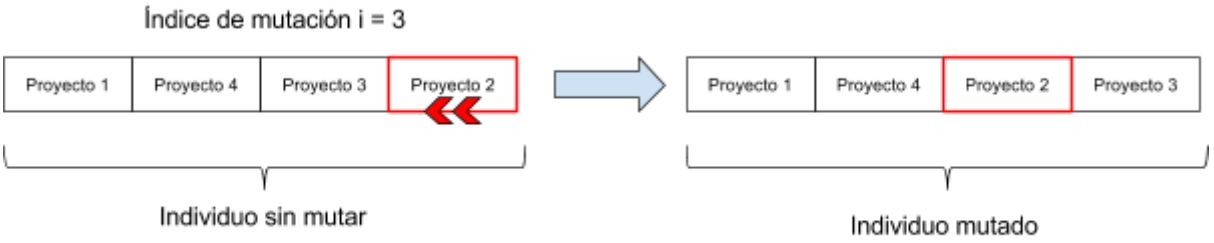
La principal característica de una mutación es que introduce nuevos genes a la población para así evitar convergencia prematura o en otras palabras que todos los individuos sean parecidos. Pero en este problema tanto la estructura de orden como afinidad ya cuentan con todos los valores lo único que puede cambiar es el orden de estos.

Es posible que ciertos proyectos no se logren priorizar si no se aplican algoritmos de mutación ya que puede suceder que toda la población priorice forma similar. El proceso utilizado se encarga de alterar el orden de los elementos de los arrays (orden y afinidad) de la siguiente forma:

1. Calcula un índice de forma aleatoria.
2. Shiftea o mueve un lugar hacia la izquierda el elemento correspondiente al índice calculado en el paso anterior.

Es importante destacar que este algoritmo cuenta con una probabilidad de ocurrencia, es decir se genera un número aleatoria entre 0 y 1 si este número es mayor a la probabilidad de ocurrencia se disparan los pasos mencionados en caso contrario no se realiza ninguna mutación.

A continuación se presenta un ejemplo de mutación, donde se puede ver que el proyecto 2 no es muy priorizado por este individuo pero el proceso introduce una variación priorizando un poco más.



Métodos de reemplazo

1. Método 1: Intercambia la generación actual por los hijos generados (en este método se utiliza k = N).

2. Método 2: Se selecciona utilizando alguno de los métodos de selección mencionados anteriormente N - k individuos de la generación actual y los k hijos generados pasan directamente a siguiente generación.

3. Método 3: Al igual que en el método 2 se utilizan métodos de selección para seleccionar N - K individuos de la generación actual y también se seleccionan k individuos de la generación actual con los k hijos generados.

Cálculo del fitness

El primer paso antes de calcular el fitness consiste en obtener un mapeo (persona -> hoja). La forma de hacerlo fue explicado en *Algoritmo para sentar personas.*

El cálculo de fitness total es una suma ponderada entre 2 funciones de fitness:

Distance Fitness

Es la suma del Distance Fitness Individual (por persona).
El Distance Fitness Individual es función de la suma de distancias que hay entre la persona y el resto de sus compañeros (personas con las que comparte al menos 1 proyecto).
Esta función de fitness favorece el acercamiento de las personas que comparten proyectos.

TMI Fitness

Es la suma del TMI Fitness Individual (por persona), TMI significa tamaño de mesa ideal.
El TMI Fitness Individual contempla si la persona fue sentada en un lugar apropiado para sentar a P, donde P es el conjunto de todas personas que comparten al menos 1 proyecto con la persona.
Esta función de fitness penaliza situaciones como sentar en una mesa de 2 personas a una persona que es parte de un proyecto con 10 personas.

Problemas encontrados

Inicialmente se planteó una representación diferente, la cual consiste en que cada individuo estaba conformado por un array de personas o empleados a los cuales se les podía asignar un espacio de trabajo y ademas tenian, de forma similar a como se explicó anteriormente, una serie de restricciones.

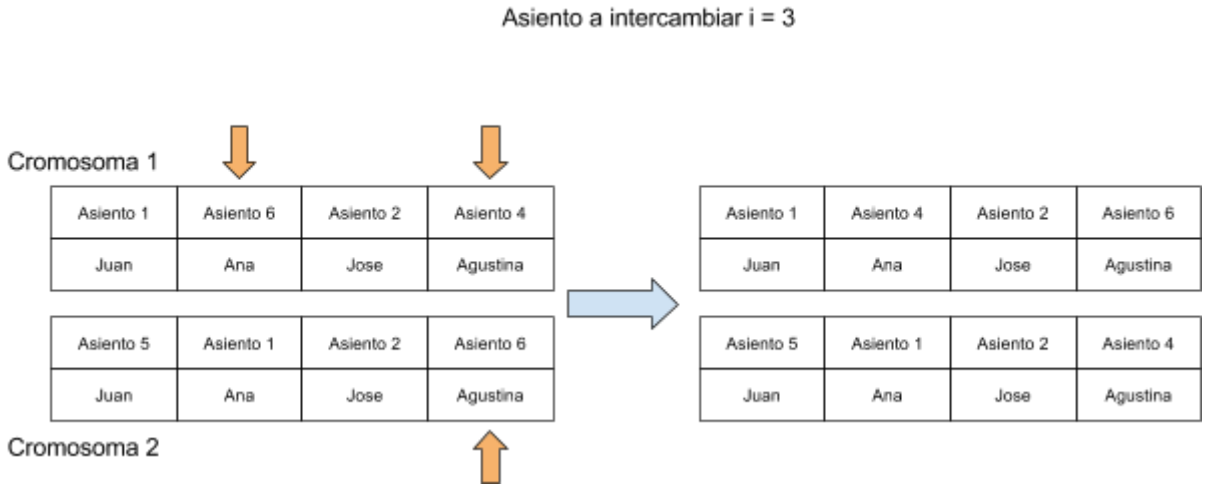
Un cromosoma válido es aquel que tiene en todos sus empleados asignado un asiento sin que este se repita con otro empleado, es decir que dos personas no pueden estar en el mismo lugar de trabajo y toda persona tiene que tener asignado un asiento.

Un posible cromosoma válido podría ser el siguiente:

Asiento 11	Asiento 2	Asiento 7	Asiento 4	Asiento 9	...	Asiento 12	Asiento 15
Juan	Carlos	Ana	Julieta	Cristian		José	Marta

Resumiendo, el algoritmo genético en esta situación se encargaba, a la hora de realizar una cruza o mutación, de intercambiar asientos entre dos personas teniendo en cuenta la información del otro cromosoma en el caso de la cruza o en el caso de la mutación simplemente intercambiaba dos lugares.

Por ejemplo en la siguiente imagen se puede observar un ejemplo de cruza para la posición 3, en el cual la información que pasa del cromosoma 1 al 2 es que Agustina está sentada en el asiento 4 y en el caso contrario la información que pasa de un cromosoma al otro es que Agustina está sentada en el asiento 6. Luego cada cromosoma reubica a el asiento viejo de agustina si es que se generó una repetición de asientos con la nueva asignación, en el gráfico si el cromosoma 1 le asigna el asiento 6 a Agustina se repetiría con el asiento de Ana entonces se le asigna el asiento viejo de Agustina (el 4) a Ana pero en cambio en el cromosoma 2 no se genera ningún conflicto al realizar el cambio.



Es muy importante destacar que este enfoque es correcto y se puede utilizar en algoritmos genéticos pero en este caso en particular los resultados que se obtienen para ciertas configuraciones de trabajo complejas no son las esperadas. Esto sucede cuando existen personas que están asignadas en más de un proyecto y los espacios de trabajo son reducidos.

Si lo comparamos con la arquitectura explicada en secciones anteriores, lo que se propone es ubicar a los proyectos completos pero en este caso lo equivalente sería estar sentando persona por persona. Por esto mismo el algoritmo al no tratar a los proyectos como un todo no logra ubicar a todas las personas de un mismo área en el lugar correcto. Lo cual se soluciono cambiando el modelo que representa el problema al explicado.

Resultados

Caso 1

En este caso se estudia una estructura básica, similar al ejemplo inicial, donde el algoritmo con gran facilidad logra establecer a los proyectos uno por mesa como era de esperarse, ya que existen tantos proyectos como mesas con la capacidad de alojarlos.

Empleados

```
juan #avenida
jose #avenida
nicolas #benedette
julieta #benedette
belen #benedette
argentino #leverpass
ignacio #leverpass
sebastian #leverpass
santiago #leverpass
julian #coca-mola
mariano #coca-mola
lucas #coca-mola
manuela #coca-mola
cecilia #mandowals
diego #mandowals
gaston #mandowals
macarena #mandowals
felipe #mandowals
gonzalo #mandowals
```

Solución encontrada

```
Floor
  room1
    table1
      c1 (jose) [#avenida]
      c2 (juan) [#avenida]
      c3 (VACIO)
    table2
      c4 (julian) [#coca-mola]
      c5 (mariano) [#coca-mola]
      c6 (lucas) [#coca-mola]
      c7 (manuela) [#coca-mola]
  room2
    table3
      c8 (julieta) [#benedette]
      c9 (nicolas) [#benedette]
      c10 (belen) [#benedette]
    table4
      c11 (ignacio) [#leverpass]
      c12 (argentino) [#leverpass]
```

```
        c13 (santiago) [#leverpass]
        c14 (sebastian) [#leverpass]
  room3
    table5
        c15 (macarena) [#mandowals]
        c16 (felipe) [#mandowals]
        c17 (gonzalo) [#mandowals]
        c18 (cecilia) [#mandowals]
        c19 (diego) [#mandowals]
        c20 (gaston) [#mandowals]
```

Caso 2

En el siguiente caso se pretende estudiar una empresa donde todas las personas tiene más de un proyecto y existe mucho solapamiento entre ellos. Se puede observar que el resultado obtenido es coherente a las asignaciones de cada persona y además el algoritmo tiene en cuenta de qué forma es mejor agrupar los empleados de los diferentes proyectos para minimizar las distancia entre ellos ya que es muy complicado situarlos a todos juntos.

Empleados

```
nacho #kanpas #dev_kanpas
tino #kanpas #skate #dev_kanpas #design_skate
tom #kanpas #design_kanpas
prudi #kanpas #skate #dev_kanpas #dev_skate
gus #skate #design_skate
cristian #skate #dev_skate
tzova #lapiz #dev_lapiz
ponti #lapiz #dev_lapiz
anis #lapiz #design_lapiz
inox #lapiz #design_lapiz #mubi #design_mubi
sebas #lapiz #dev_lapiz #mubi #design_mubi
dinu #mubi #dev_mubi
alan #mubi #dev_mubi
ema #mubi #design_mubi
euge #verde #dev_verde
meli #verde #design_verde
pol #verde #dev_verde #mubi #dev_mubi
nardi #verde #design_verde #mubi #dev_mubi
conco #lapiz #kanpas
fefe #skate
juli #verde #dev_verde
juani #kanpas
```

Solución encontrada

```
wolox
  4
    A
      M1
        L1
          C1 (conco) [#lapiz, #kanpas]
```

C2 (anis) [#lapiz, #design_lapiz]
C3 (Vacío)
L2
C1 (tino) [#kanpas, #dev_kanpas, #skate, #design_skate]
C2 (nacho) [#kanpas, #dev_kanpas]
C3 (Vacío)
M2
L1
C1 (dinu) [#mubi, #dev_mubi]
C2 (alan) [#mubi, #dev_mubi]
C3 (Vacío)
L2
C1 (tom) [#design_kanpas, #kanpas]
C2 (juani) [#kanpas]
C3 (Vacío)
M3
L1
C1 (prudi) [#dev_skate, #kanpas, #dev_kanpas, #skate]
C2 (cristian) [#dev_skate, #skate]
C3 (Vacío)
L2
C1 (ponti) [#lapiz, #dev_lapiz]
C2 (tzova) [#lapiz, #dev_lapiz]
C3 (Vacío)
M4
L1
C1 (Vacío)
C2 (Vacío)
C3 (Vacío)
L2
C1 (Vacío)
C2 (Vacío)
C3 (Vacío)
B
M1
C1 (euge) [#dev_verde, #verde]
C2 (meli) [#verde, #design_verde]
C3 (juli) [#dev_verde, #verde]
C4 (pol) [#dev_verde, #mubi, #dev_mubi, #verde]
C5 (nardi) [#mubi, #dev_mubi, #verde, #design_verde]
C6 (Vacío)
M2
C1 (Vacío)
C2 (Vacío)
5
A
M1
C1 (Vacío)
C2 (Vacío)
M2
C1 (sebas) [#lapiz, #mubi, #dev_lapiz, #design_mubi]
C2 (ema) [#mubi, #design_mubi]
C3 (inox) [#lapiz, #mubi, #design_lapiz, #design_mubi]
B
M1
C1 (Vacío)
C2 (Vacío)
C3 (Vacío)
C4 (Vacío)
M2

C1 (fefe) [#skate]
C2 (gus) [#skate, #design_skate]
C3 (Vacío)
C4 (Vacío)

M3
C1 (Vacío)
C2 (Vacío)
C3 (Vacío)
C4 (Vacío)

Conclusión

Los resultados obtenidos confirman que los algoritmos genéticos son una herramienta prometedora en materia del problema seleccionado. En las pruebas realizadas se obtuvieron soluciones que son coherentes, incluso en casos caóticos donde una persona puede pertenecer a más de un proyecto. Este método para hallar una forma de sentar a las personas aporta más valor cuando la cantidad de proyectos es muy elevada (superior a 15 aproximadamente), debido a que es en esos casos que el problema se vuelve difícil de resolver a mano. Queda pendiente analizar los beneficios de introducir variabilidad en el orden en el que se sientan los miembros de un mismo proyecto.

Anexo

Metodos de seleccion

Elite

Se seleccionan los k individuos con mayor fitness y se los retorna. Éste método no devuelve individuos repetidos si la población no tiene individuos repetido.

Ruleta

Se calculan las aptitudes relativas acumuladas de cada individuo. Luego se generan k randoms y se eligen los k “ganadores” que verifican que su aptitud relativa acumulada es la menor en superar el valor del k-ésimo número random.

$$p_i = \frac{f(i)}{\sum_{i=1}^{n} f(i)}$$

$$q_0 = 0$$

$$q_i = \sum_{j=1}^{i} p_j$$

Universal

Similar al método ruleta, pero solo se utiliza un número random y se generan k números random a partir de este. Luego se procede de la misma forma que en el método anterior para seleccionar los individuos.

$$r_j = \frac{r + j - 1}{k}, \; j \; \varepsilon \; [1,k]$$

Boltzmann

Se calcula el valor esperado de cada individuo. Además se utiliza una función decreciente para indicar el valor actual de la temperatura T, en nuestro caso se decrementa en uno la temperatura por cada generación transcurrida. El valor inicial de la temperatura se puede configurar.
Luego se calcula los valores esperados acumulados y se procede de la misma forma que en los métodos anteriores para buscar los individuos seleccionados.

$$f(i,t) = \frac{e^{f(i)/T}}{<e^{f(i)/T}>^t}, \text{ donde } <>^t \text{ denota acumular los valores de la poblacion } t$$

Torneo determinístico

Se realizan pequeños torneos entre grupos de m individuos elegidos al azar, el ganador es el de mayor rendimiento. Este proceso se repite k veces.

Torneo probabilístico

Similar al torneo determinista pero los grupos son de dos individuos y es 0.75 probable que gane el individuo de mayor rendimiento. También se repite el procedimiento k veces.

Ranking

Se ordenan de menor a mayor por desempeño los individuos a seleccionar, luego se calcula los ranking relativos acumulados para cada individuo. Para calcularlo se utilizó la fórmula que se indica muestra más abajo, donde i es la posición del individuo y n es la cantidad de individuos. Luego se seleccionan k individuos utilizando números random igual que en el método de ruleta.

$$r(i,n) = \frac{i}{(n * (n+1)) / 2}$$

Aleatorio

Como el nombre lo indica se seleccionan k individuos al azar.