DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Sanguijuelas

28 de noviembre de 2014

Métodos Numéricos
Trabajo Práctico Nro. 1

Integrante	LU	Correo electrónico
Martin Carreiro	45/10	martin301290@gmail.com
Kevin Kujawski	459/10	kevinkuja@gmail.com
Juan Manuel Ortíz de Zárate	403/10	jmanuoz@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Índice

1. Resumen

Mediante el manejo de matrices se buscará modelar y resolver un problema de la realidad. Cómo la realidad está compuesta de infinitas variables dicha modelización implicará una inevitable discretización. Es decir trabajar con una cantidad acotada de variables del problema (sólo las relevantes).

El problema en cuestión es representar un parabrisas al cual se le adhieren sanguijuelas que le producen calor en un radio de las mismas. Queremos evitar que el parabrisas llegue a un punto crítico, que implicaría la destrucción de este. El punto crítico ocurre cuando el punto del centro tienen más de 235^oC. Para evitar que llegue al punto crìtico debemos deshacernos de las sanguijuelas pero no tenemos la energía para destruirlas a todas, por lo que tenemos que achicar la cantidad de sanguijuelas a eliminar de forma tal que el parabrisas no se rompa.

La modelización del mismo no es trivial. Es más, podría decirse que este proceso es mucho más complejo y costoso que la solución en sí. ¿Por qué? porque la creación de la matriz que represente al parabrisas y el cálculo de las temperaturas en cada punto, si no se usa un buen método, podría llegar a demorar mucho tiempo.

Por esto es que a lo largo del TP haremos mucho foco en como representar en forma de matriz el parabrisas, como este puede llegar a tener propiedades que utilizaremos para ahorrar el espacio ocupado. Una vez que tengamos esto, necesitamos saber si realmente estámos bajo un punto crítico. Esto estará determinado por la posición central de la matriz que discretiza el parabrisas $(altura/2, ancho/2)$ (donde altura y ancho son las cantidad de celdas correspondientes) y su temperatura (si es mayor o menor a 235^oC) para saber así si necesitamos deshacernos de las sanguijuelas para poder seguir navegando.

Una vez que tengamos esto plantearemos casos de test interesantes para analizar el tiempo de decisión de una situación de peligro y que el método elegido para deshacernos de las sanguijuelas nos minimiza la cantidad a destruir.

2. Introducción teórica

Dado que la modelización utlizada para este problema es un sistema de ecuaciones y que en base a algunos datos que nos dan (posición de las saniguijuelas, dimensiones y temperaturas de los bordes) tenemos que calcular los valores de muchas celdas, tenemos que buscar la forma de modelar esto óptimamente. Lo primero que se nos viene a la mente cuando tenemos esto es el método de eliminación gaussiano. Este transforma la matriz de coeficientes en una matriz triangular superior y luego mediante back substitution se pueden obtener todos sus valores. Si bien este método no funciona siempre veremos que para nuestro caso puede otorgarnos soluciones aceptables.

Veamos ahora más en detalle como funciona cada uno de estos algoritmos.

2.1. Algoritmo de eliminación gaussiana

Este algoritmo consisnte en restar multiplos de filas entre sí, comenzando desde la primera, con el objetivo de lograr todos ceros por debajo de la diagonal y numeros disintos de cero en la diagonal, es decir una matriz triangular superior. En el caso que me quedase un 0 en la diagonal el algoritmo comprende la opción de pivoteo (mas adelante demostraremos que en realidad no necesitamos) que consiste en intercambiar filas entre sí para que esto no suceda.

Veamos ahora algún ejemplo concreto con la siguiente matriz de 3×3:

$$\begin{pmatrix} 4 & 8 & 9 \\ 1 & 5 & 2 \\ 2 & 3 & 1 \end{pmatrix}$$

Hago

F2 = F2 - 1/4 (F1) y F3 = F3 - 1/2 (F1)

$$\begin{pmatrix} 4 & 8 & 12 \\ 0 & 3 & -1 \\ 0 & -1 & -5 \end{pmatrix}$$

y ahora

F3 = F3 - -1/3 (F2)

$$\begin{pmatrix} 4 & 8 & 12 \\ 0 & 3 & -1 \\ 0 & 0 & -31/3 \end{pmatrix}$$

Si entendemos la matriz original como un sistema de ecuaciones con 3 variables igualadas a distintos resultados, como podemos ver a continuación:

$$\begin{pmatrix} 4x & 8y & 9z \\ 1x & 5y & 2z \\ 2x & 3y & 1z \end{pmatrix}$$

Con esta nueva matriz triangulada podríamos calcular facilmente el valor de cada variable ya que de

la última ecuación podemos despejar facilmente el valor de Z, reemplazarlo en las otras ecuaciones y así calcular las otras variables.

Por último cabe destacar que la complejidad computacional de esta trinagulación es de O(n^3). Lo cual puede llegar a un cálculo sumamente extenso si nuestro n es considerablemente importante. Por esto es que decidimos pensar un poco mas la situación y encontrarle la vuelta para resolver en un tiempo razonable.

2.2. Matriz banda

Matriz banda es un caso especial de las matrices que ocurre cuando estas concentran toda su información cerca de la diagonal, más especificamente, la matriz tiene en su mayoría elementos nulos y los que no son nulos se concentrar a izquierda y derecha de la diagonal de la matriz, siendo el máximo de estas distancias la que determinaran el ancho de la banda con que se denomine a la matriz y buscandose que las mismas sean relativamente pequeñas en comparación del ancho de la matriz. Por ejemplo, si a lo sumo tiene p elementos no nulos a la izquierda y q a la derecha, esta matriz tendrá una banda de $p + q + 1$. Hay casos particulares para denominar estas matrices, por ejemplo, cuando $p = q = 0$ esta se denomina diagonal, y en caso de $p = q = 1$ es una tridiagonal. La ventaja principal que proporcina este tipo de matriz es la optimización a la hora de su almacenamiento, ya que como la mayoría de los elementos son nulos y los no nulos tiene un ancho fijo, es posible guardarlos en una matriz mucho menor con solo el tamaño de la banda como ancho, ahorrando gran cantidad de memoria.

Aca podemos ver un ejemplo de una matriz banda:

Matriz banda tridiagonal

$$\begin{pmatrix} d & q & 0 & 0 & 0 & 0 \\ p & d & q & 0 & 0 & 0 \\ 0 & p & d & q & 0 & 0 \\ 0 & 0 & p & d & q & 0 \\ 0 & 0 & 0 & p & d & q \\ 0 & 0 & 0 & 0 & p & d \end{pmatrix}$$

En el ejemplo se puede ver como se forma la banda sobre la diagonal, con la mayoría de elementos nulos fuera de la misma. Por lo tanto para ahorrar espacio se puede optimizar su almacenamiento de la siguiente forma:

Almacenamiento sólo de la banda de la matriz original

$$\begin{pmatrix} 0 & d & q \\ p & d & q \\ p & d & q \\ p & d & q \\ p & d & q \\ p & d & 0 \end{pmatrix}$$

En la sección Desarrollo mostraremos como aprovecharemos las ventajas de este tipo de estructura para resolver el problema planteado de una forma más eficiente.

3. Desarrollo

3.1. Matriz normal

Como una primera forma de resolución planeamos el parabrisas como un sistema de ecuaciones (más adelante veremos que el mismo presenta una organización de los elementos no nulos que se denomina "*matrizbanda*", pero por ahora no tendremos en cuenta esto y por consiguiente no realizaremos ninguna optimización al respecto), este se vería representado como una matriz, donde cada columna y cada fila representa a una posición especifica en el parabrisas pero que a su vez cada fila simboliza las ecuaciones que planteo para esa posición. Por lo tanto la diagonal de la matriz representa al coeficiente de esa posición en el parabrisas. En base a lo que explicamos anteriormente y como la temperatura de cada celda vacía es el promedio de las 4 contiguas, al plantear la ecuación:

$$t_{ij} = (t_{i+1j} + t_{ij+1} + t_{i-1j} + t_{ij-1})/4$$

$$4t_{ij} = t_{i+1j} + t_{ij+1} + t_{i-1j} + t_{ij-1}$$

$$-4t_{ij} + t_{i+1j} + t_{ij+1} + t_{i-1j} + t_{ij-1} = 0$$

Nos queda que los coeficientes de la celda actual es -4 y de las contiguas un 1, igualando todo a cero. Por lo tanto en cada fila de la matriz calculamos la posición que deberia quedar cada celda contigua en la matriz aplanada, quedando el sistema de ecuaciones de la siguiente forma:
(donde los espacios vacios representan blancos)

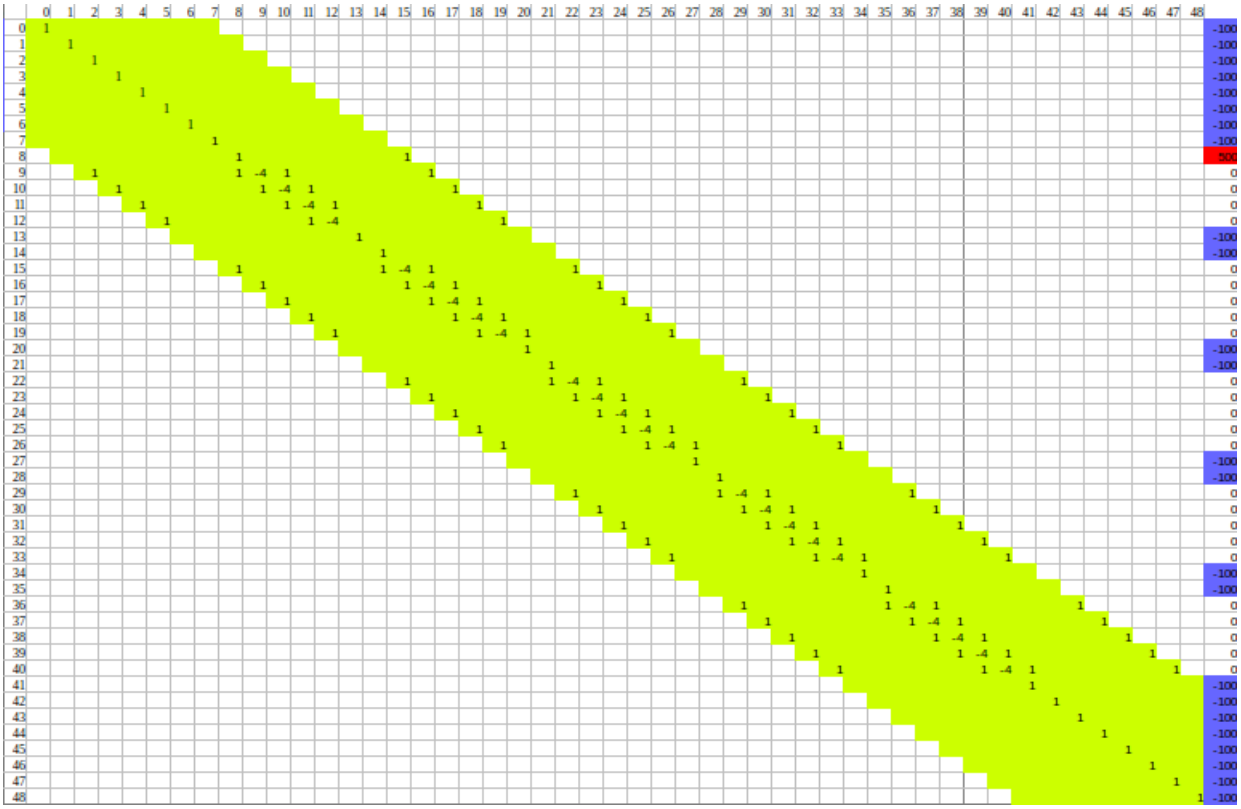


Figura 1: Matriz de ecuaciones del ejemplo

Por lo tanto la solución a este sistema planteado se hará por medio del clásico método de resolución de Eliminación Gaussiana que consta de la eliminación progresiva de variables en el sistema de ecuaciones, hasta tener sólo una ecuación con una incógnita. Una vez resuelta esta, se procede por sustitución regresiva hasta obtener los valores de todas las variables.

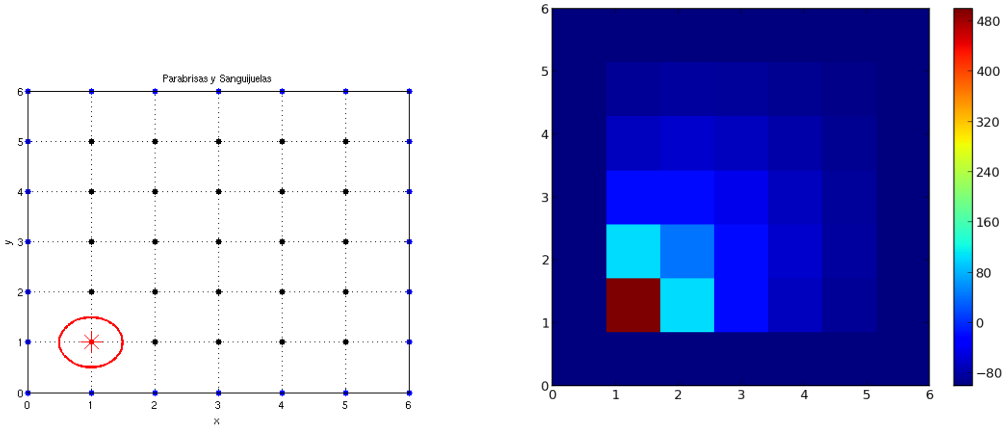
```
Class Windshield{
    resolveByGaussianElimination(){
         gaussianElimination();
         backSubstitution();
    }
    gaussianElimination(){
        for rows k already been reduced
            iMax = findPivot();
            Swap rows k and imax;
            Force 0's in column A[k+1..n-1][k];
    }
    backSubstitution(){
        for row from last to first:
            calculateFromNextOneExceptForLast()
    }
}
```

3.2. Matriz Banda

En la sección anterior presentamos una forma de resolver nuestro problema representando el parabrisas como un sistema de ecuaciones y aplicando Eliminación Gaussiana. En este sección propondremos una optimización a esto, pero para esto primero necesitamos ver con qué cosas estamos tratando y para eso veamos el siguiente ejemplo:

En este caso es un parabrisas de 6x6 granulado en 1 y una sanguijuela de radio 1 con temperatura de 500 en la primera posición.

El parabrisas se vería representado asi (hay que tener en cuenta que en realidad hay que espejarlo horizontalmente ya que para nosotros el eje de coordenadas se encuentra arriba a la izquierda):



Analizando la matriz del sistema de ecuaciones que se genera a partir de la representación del Parabrisas observamos que presenta las características de la matriz "*banda*" explicada anteriormente. Este es un hecho importante ya que las matrices banda tienen características especiales de las cuales es posible la optimización temporal y espacial para resolver el sistema de ecuaciones con eliminación gausseana, principalmente el hecho de que no es necesario realizar intercambios de filas en el mismo, aunque esto viene sumado al hecho de que la matriz es además semidefinida positiva y asegurándonos esta ventaja.

3.2.1. Optimización

El problema de representar la matriz original es que la mayoría de los valores son 0 y solo importan los elementos de la banda de la matriz, por lo tanto una forma de optimizar espacialmente es solo guardar la banda, con lo que se logra reducir considerablemente el espacio en esta representación, ya que suponiendo que se tiene un parabrisas de n filas y m columnas, el tamaño de la matriz original sería de $(n*m)^2$, mientras que con la optimización de matriz banda queda de tamaño $(n*m)*(2*m+1) \approx (n*m^2)$.

El método para construir la representación optimizada de la matriz banda es simple, se guarda la banda en una matriz pero rotandola de forma que quede vertical, quedando en el centro los elementos de la diagonal dependiendo de lo que haya en esa posición, ya que en el caso de que sea vacía esta deberá tener coeficientes de las posiciones adyacentes. Además se adiciona una columna más para representar la temperatura en esa posición. Aúnque uno podría pensar que ya que solo hay 5 elementos no nulos máximo en cada fila se podria optimizar guardando esos 5 coeficientes, pero debido a que luego se le aplicará el método de Back Substitution de la Eliminación Gausseana como método para resolver el sistema, esto hará que se generen valores no nulos dentro de la banda, por lo tanto el tamaño de esta matriz deberá ser lo suficiente como para poder trabajar sobre él y almacenar los valores parciales. La matriz banda tendra hasta un maximo de 5 valores no nulos por fila dependiendo el tipo de posición de la misma, ya que si en la posición hace referencia al borde frío, esta fila tendra un 1 como valor central y un -100 en la columna de resultados, esto se deduce a partir de la expresión

$$T(x,y) = -100^o\text{C} \qquad \text{si } x = 0, m \text{ ó } y = 0, n. \tag{1}$$

Donde m es ancho y n alto.
El caso para el cual la posición contenga una sanguijuela será similar al de frío, ya que deberá respetar la ecuación

$$T(x,y) = temp_{sanguijuela} \tag{2}$$

Y por último, en base a las ecuaciones que deben respetar las posiciones vacías vistas anteriormente, en la posición central que representa a la posición actual quedará un -4 y en las de arriba-abajo-izquierda-derecha un 1 en cada una, siendo referenciadas en la matriz banda por las posiciones de la izquierda y derecha de la central, y las que estan a distancia de la banda, por lo tanto, en las puntas de la fila de matriz, y al final de todo un 0 en la columna de resultado. Por lo tanto, cada fila que represente a una posición vacía tendra una distribución inicial de la siguiente forma:

$$\begin{pmatrix} 1 & 0 & \dots & 0 & 1 & -4 & 1 & 0 & \dots & 0 & 1 \end{pmatrix}$$

El algoritmo consta de dos etapas, la construcción de la matriz que representara la banda y la resolución del problema utilizando la misma.

Construcción de la matriz banda

```
por cada posición del parabrisas //O(N*M)
       pos = fila posicion + columna posicion * ancho;

              si en esa posición hay una SANGUIJUELA:

                    matrizBanda[pos][ancho] = 1;
                    resultados[pos]         = temp_sanguijuela;

              si esa posición es borde FRIO:

                    matrizBanda[pos][ancho] = 1;
                    resultados[pos]         = -100;

              en caso de que esa posición sea VACIA:

                    matrizBanda[pos][ancho]   = -4;
                    matrizBanda[pos][ancho-1] = 1;
                    matrizBanda[pos][0]       = 1;
                    matrizBanda[pos][ancho+1] = 1;
                    matrizBanda[pos][ancho*2] = 1;
                    resultados[pos]           = 0;
```

Continuando con el ejemplo que propusimos cuando empezamos esta sección, la matriz banda resultante sería la siguiente:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0								1									-100
1								1									-100
2								1									-100
3								1									-100
4								1									-100
5								1									-100
6								1									-100
7								1									-100
8								1									500
9	1						1	-4	1							1	0
10	1						1	-4	1							1	0
11	1						1	-4	1							1	0
12	1						1	-4	1							1	0
13								1									-100
14								1									-100
15	1						1	-4	1							1	0
16	1						1	-4	1							1	0
17	1						1	-4	1							1	0
18	1						1	-4	1							1	0
19	1						1	-4	1							1	0
20								1									-100
21								1									-100
22	1						1	-4	1							1	0
23	1						1	-4	1							1	0
24	1						1	-4	1							1	0
25	1						1	-4	1							1	0
26	1						1	-4	1							1	0
27								1									-100
28								1									-100
29	1						1	-4	1							1	0
30	1						1	-4	1							1	0
31	1						1	-4	1							1	0
32	1						1	-4	1							1	0
33	1						1	-4	1							1	0
34								1									-100
35								1									-100
36	1						1	-4	1							1	0
37	1						1	-4	1							1	0
38	1						1	-4	1							1	0
39	1						1	-4	1							1	0
40	1						1	-4	1							1	0
41								1									-100
42								1									-100
43								1									-100
44								1									-100
45								1									-100
46								1									-100
47								1									-100
48								1									-100

Figura 2: Matriz banda del ejemplo

Aquí se puede observar mejor como se distribuyen los coeficientes de la matriz original que pertenecen a la banda sobre esta nueva representación de ella, y como se "*verticaliza*" la misma. Como sucedió anteriormente, los espacios en blanco representan los valores nulos.

Hasta este punto tenemos construida la representación de la matriz original en una forma que optimice el almacenamiento de la misma, ahora solo queda resolverla de una forma también optima.

Resolución del sistema

Este algoritmo se basa basicamente en el algoritmo de eliminación gausseana para triangular y luego realizar un back sustitution para obtener el valor de los coeficientes. Lo que es remarcable es el hecho de como trabaja en la matriz banda vertical como si fuese la banda en diagonal de la matriz original.

```
//PRIMERO LA HAGO TRIANGULAR SUPERIOR
PARA CADA FILA DE LA BANDA, DE ARRIBA A ABAJO:

   SI NO ES VACIO SE QUE ABAJO HAY TODO CERO, CONTINUO;
   SI ES VACIA DIAGONALIZO:
   {
         centro = bandMatrix[i][n];
         actual = bandMatrix[i+h][n-h];
         multiplicador = actual / centro;
         fila[i+h] - fila[i];
         resultado[i+h] -= resultado[i] * multiplicador;
    }

 // EN ESTE PUNTO YA TENGO LA MATRIZ TRIANGULADA SUPERIORMENTE
 //AHORA SOLO FALTA REALIZAR
 // BACK SUBSTITUTION
 PARA CADA FILA i DE LA BANDA, DE ABAJO A ARRIBA
        fila = i / n;
        columna = i % n;
        for( int h = 1; h <= n; h++){
        // Recorro la banda en forma diagonal
        // Equivale a recorrer en forma vertical con la matriz normal
        // COMO ES BANDA ME FIJO SI EN LA DIAGONAL IZQ INF HAY DISTINTO DE 0 PARA PIVOTEAR
             centro = bandMatrix[i][n];
             actual = bandMatrix[i-h][n+h];
             multiplicador = actual / centro;

             fila[i-h] -=  fila[i] * multiplicador; // Opero entre filas
             resultado[i-h] -= resultado[i]* multiplicador;

             // y por último actualizo los valores del parabrisas con las temperaturas actuales
             Parabrisas[fila][columna].temp = actual
       }
```

4. Eliminar sanguijuelas

La resolución del problema de evitar el punto crítico en el centro del parabrisas consiste en la eliminación de sanguijuelas. Suponiendo la existencia de n sanguijuelas, la cantidad de subconjuntos a probar son 2^n. Es un número que crece rápidamente y si bien se puede resolver por fuerza bruta (ayudados por Backtracking y derivados), suponemos que encontrar el mínimo tiene un orden exponencial en la cantidad de sanguijuelas.

Finalmente, planteamos dos tipos de solución a modo de comparación. Estos serán detallados más adelante. Ambas son heurísticas por lo mencionado anteriormente.

4.1. Punto Crítico

En un parabrisas real el punto crítico se determina por el punto exacto del medio del parabrisas, en la posición (largo/2, ancho/2), pero en nuestra estructura de datos que representa al parabrisas esta puede no llegar a tener un punto exacto dependiendo de la discretización que tenga el mismo. Por lo tanto decidimos tomar la celda de la posición (filas/2,columnas/2), ya que en nuestra discretización sería una buena aproximación al centro. En caso de haber fila o columna impar tomamos la parte entera de la división.

4.2. Solución Greedy

Esta solución consiste en ir seleccionando las sanguijuelas más cercanas al punto medio hasta que el punto central esté por debajo de 235 C°. Al no ser una solución exacta es posible que eliminar la más cercana al centro no pertenezca a la mejor solución, pero es intuitivo suponer que la sanguijuela que más cause calor al punto medio esté cerca del mismo.
Lo primero que se realiza es ordenar las sanguijuelas según su distancia al centro. Para esto utilizamos la función *sort*() de C++ que utiliza como comparador el valor pre-calculado de distancia al centro. El orden de esta parte es $O(n.log(n))$ en donde n es la cantidad de sanguijuelas.
Al ser una solución golosa (greedy), elige la siguiente sanguijuela siempre y cuando no se haya enfríado el punto medio y haya salido del estado crítico. La elección está dada por la sanguijuela más cerca todavía no elegida en una previa iteración.
Suponiendo que la complejidad de rehacer el cálculo es RA, la complejidad final es: $O(n.log(n) + n * RA)$. Notar que esa última n es en el peor caso que tengamos que eliminar todas las sanguijuelas, pero debería ser un $k < n$.

Algorithm 1 Solución Greedy

```
1: ordenoLasSanguijuelasPorCercaníaAlPuntoCrítico()
2: sanguijuelasAMatar ← []
3: do
4:     sanguijuelasAMatar.agregar(obtenerSanguijuelaMasCercana())
5: while laTemperaturaEsAlta()
6: devuelvo sanguijuelasAMatar
```

Algorithm 2 laTemperaturaEsAlta()

```
1: recalculoConMatrizBanda();
2: devuelvo (parabrisas.tempPuntoCritico() > 235)
```

4.2.1. Espiral vs centrico

Durante el desarrollo del algoritmo Greedy, nuestro primer acercamiento fue recorrer la matriz de forma espiral. El primer problema que encontramos fue que teníamos un cálculo muy grande para determinar dado un punto del parabrisas, qué sanguijuela era y que recorríamos partes de la matriz que no importaban (en especial los bordes). A su vez, tendíamos a alejarnos del objetivo principal que era preocuparnos por el centro. Es por eso que decidimos ordenar las sanguijuealas según su distancia al centro. Para ahorrar tiempo computacional, este dato está calculado en el momento de la creación de la sanguijuela para su posterior ordenamiento.

4.3. Solución Random

En esta solución seleccionamos de nuestro array de posiciones de sanguijuelas (que es el array recibido por parametro, osea con las posiciones sin discretizar) una al azar y la eliminamos. Luego ejecutamos de nuevo el cálculo de las temperaturas y chequeamos si el punto crìtico esta por debajo del 235 C^{o}.Si no lo está, elegimos otra al azar y repetimos el proceso hasta que lo esté.

Algorithm 3 RandomSolution

```
1: do
2:     matarSanguijuelaRandom()
3: while laTemperaturaEsAlta()
4: return sanguijuelasAEliminar
```

Algorithm 4 matarSanguijuelaRandom()

```
1: sanguijuelasAEliminar.agregar(removerSanguijuelaRandom(listaSanguijuelas)); ▷ All Leaches is loaded with matrix
2: recalculoConMatrizBanda();
```

5. Experimentación Y Resultados

5.1. Introducción

Realizamos los experimentos en tres computadoras con procesadores similares y se trató de mantener cualquier otro programa cerrado. Igualmente cada conjunto de test fue resuelto en la misma máquina para que las diferencias sean lo más dependendientes a los inputs y los procesos posible.

5.2. Temperatura en el punto crítico variando granularidad

En esta sección queremos ver que ocurre con la temperatura en el punto crítico para distintas granularidades de la misma instancia. Para tal fin vamos a presentar distintos tipos de instancias que pueden ser enriquecedoras para el analisis. Queremos aclarar que no nos pareció relevante el analisis con un sanguijuela sobre el punto crítico ya que la temperatura no sufriría variación alguna. Esto último es debido a que todavía no utilizaremos los metodos de eliminación de sanguijuelas para mantener el punto crítico a menos de 235º

Caso 1: 8 sanguijuelas rodeando al punto crítico.

Caso 2: Una sanguijuela muy cercana al punto crítico.

Caso 3: 5 sanguijuelas alineadas debajo del punto crítico.

Las condiciones del parabrisas que utilizaremos para los 5 casos seran similiares, un ancho y alto de 1024 (para tener un gran espacio de testeo y cantidad de divisores), con una temperatura de 400º para las sanguijuelas (para que influya pero que a su vez no sean tan determinantes) y un radio de 16 posiciones. El valor de las discretizaciones que elegimos para cada caso está limitado en función la cantidad de sanguijuelas ya que a medida que aumenta el valor la cantidad de posiciones disponibles en el parabrisas disminuye

Caso 1:

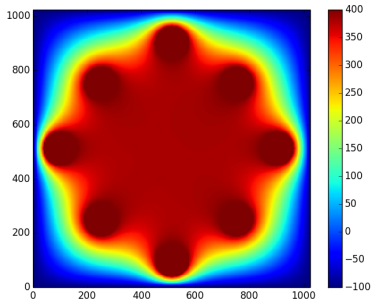


Figura 3: Granularidad: 4 - Temperatura: 382.021°

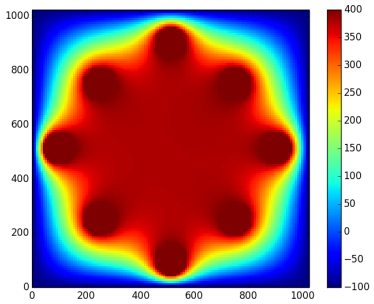


Figura 4: Granularidad: 8 - Temperatura: 380.138°

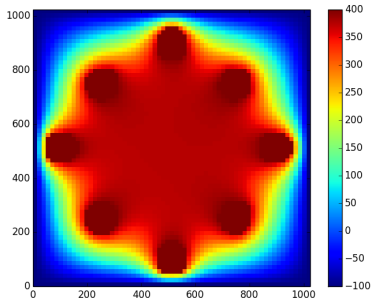


Figura 5: Granularidad: 16 - Temperatura: 375.96°

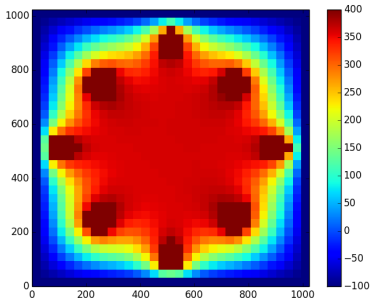


Figura 6: Granularidad: 32 - Temperatura: 362.425°

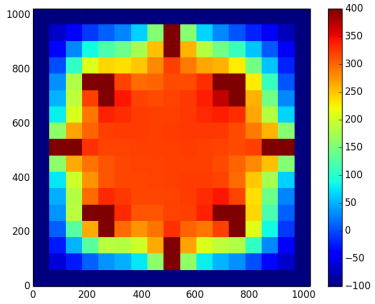


Figura 7: Granularidad: 64 - Temperatura: 321.288°

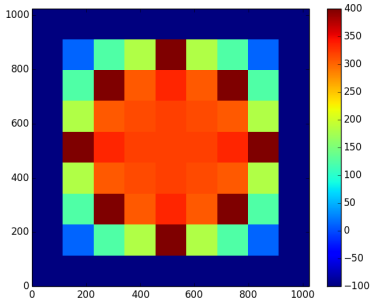


Figura 8: Granularidad: 128 - Temperatura: 268.117°

Caso 2:

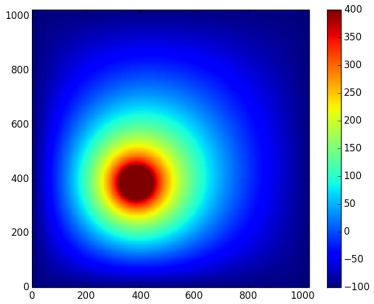


Figura 9: Granularidad: 4 - Temperatura: 166.282°

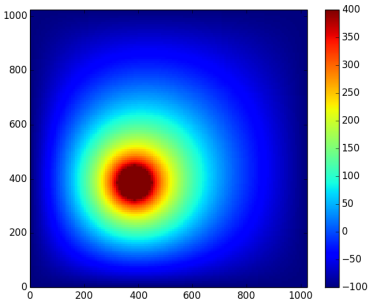


Figura 10: Granularidad: 8 - Temperatura: 163.331°

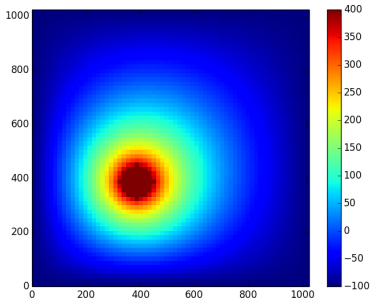


Figura 11: Granularidad: 16 - Temperatura: 158.461°

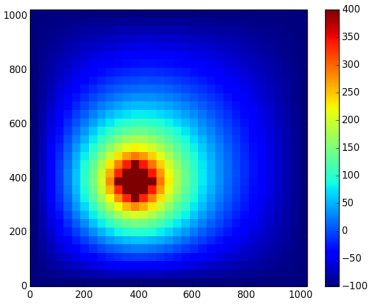


Figura 12: Granularidad: 32 - Temperatura: 153.939°

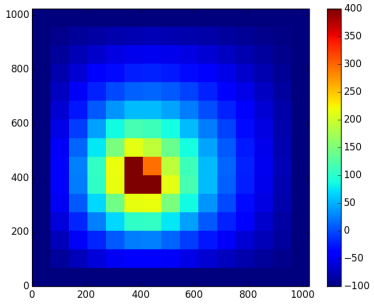


Figura 13: Granularidad: 72 - Temperatura: 138.874°

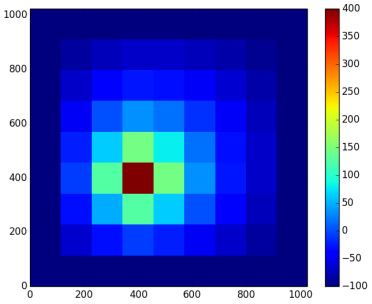


Figura 14: Granularidad: 128 - Temperatura: 81.2233°

Caso 3:

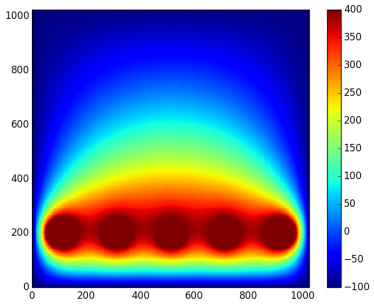


Figura 15: Granularidad: 4 - Temperatura: 154.532°

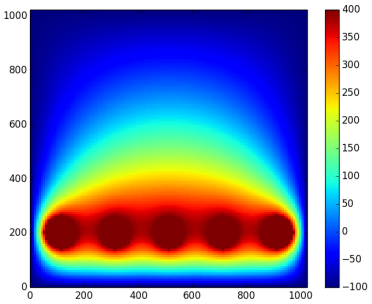


Figura 16: Granularidad: 8 - Temperatura: 153.194°

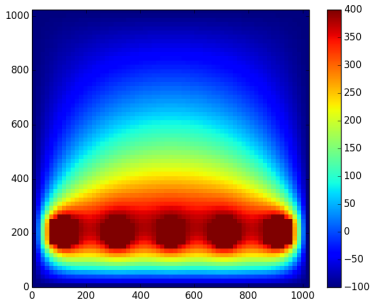


Figura 17: Granularidad: 16 - Temperatura: 150.55°

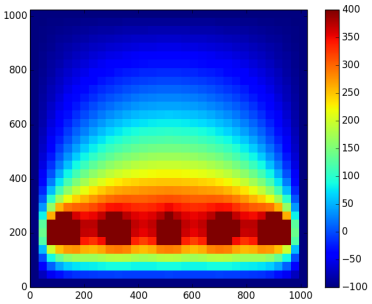


Figura 18: Granularidad: 32 - Temperatura: 146.013°

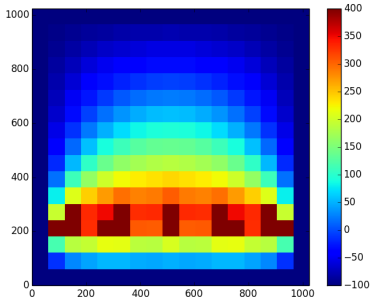


Figura 19: Granularidad: 64 - Temperatura: 137.51°

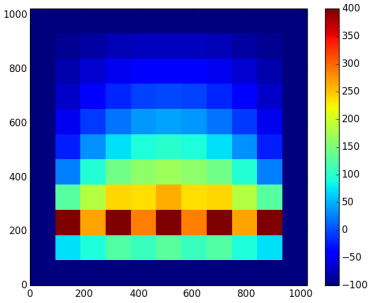


Figura 20: Granularidad: 100 - Temperatura: 100.115°

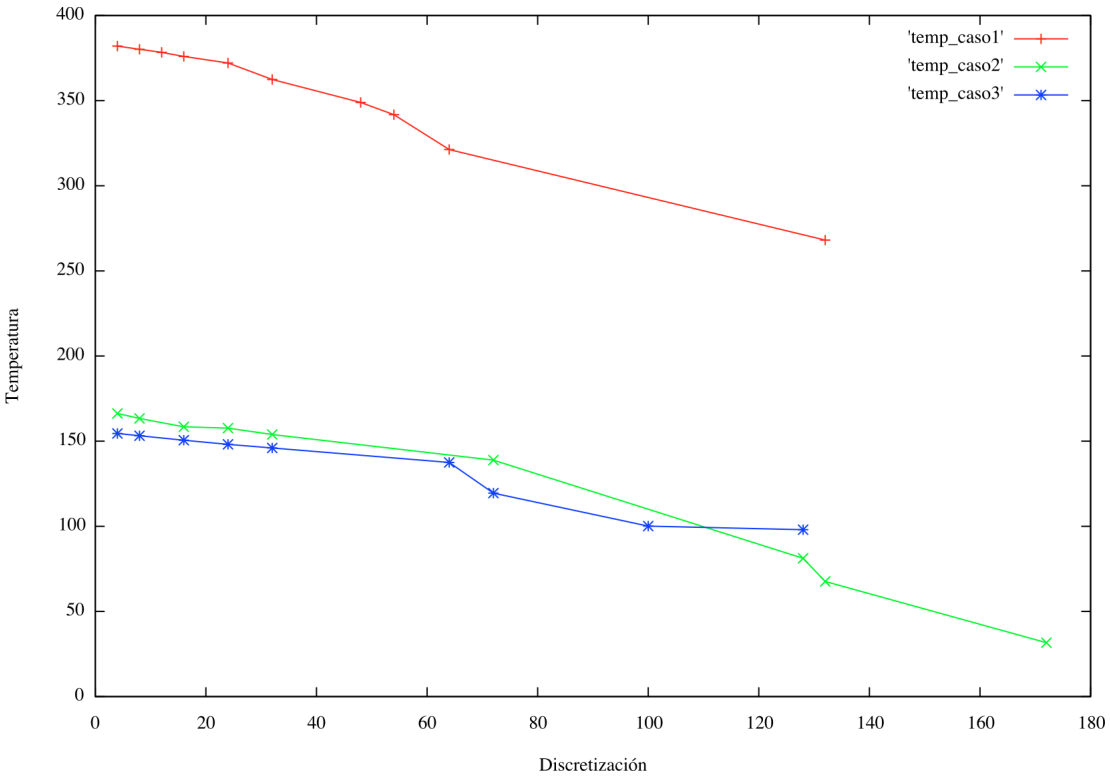


Figura 21: Comparación de temperaturas

Por lo que se puede observar en los gráficos de como va fluctuando la temperatura a medida que se aumenta el valor de discretización (que provoca disminución en la granuralidad) para los 3 casos de prueba y con el ultimo gráfico comparativo, como era de esperarse, la temperatura disminuye en el punto crítico, esto es debido a que a mayor granularidad las sanguijuelas tienen mas borde en el parabrisas para transferir el calor ya que dependiendo el radio de la sanguijuela provoca que ocupe más puntos en el parabrisas. Esto último del radio se puede ver cuando en los últimos gráficos de cada caso cuando la garrulidad es minima que las sanguijuelas pasan a ocupar una posición del parabrisas.

5.3. Tiempo de corrida variando granularidad

Utilizando los mismos casos de prueba que para la comparación de temperaturas medimos los tiempos que tomaban los dos algoritmos de resolución normal y banda. Los parámetros que utilizamos en esta oportunidad también son similares a la sección anterior, menos el valor de discretización que es el variable. Los valores que tomamos para este caso estan en el intervalo de [4;50] para el algoritmo de banda y [18;95] para el normal, y la razón de esto es que para valores muy chicos el tiempo de cómputo era tal para el algoritmo normal que el SO mataba el proceso y para valores muy grandes los dos algoritmos empezaban a acercarse al cero por lo que decidimos que era un buen número de limite superior.
A continuación el gráfico de tiempos:

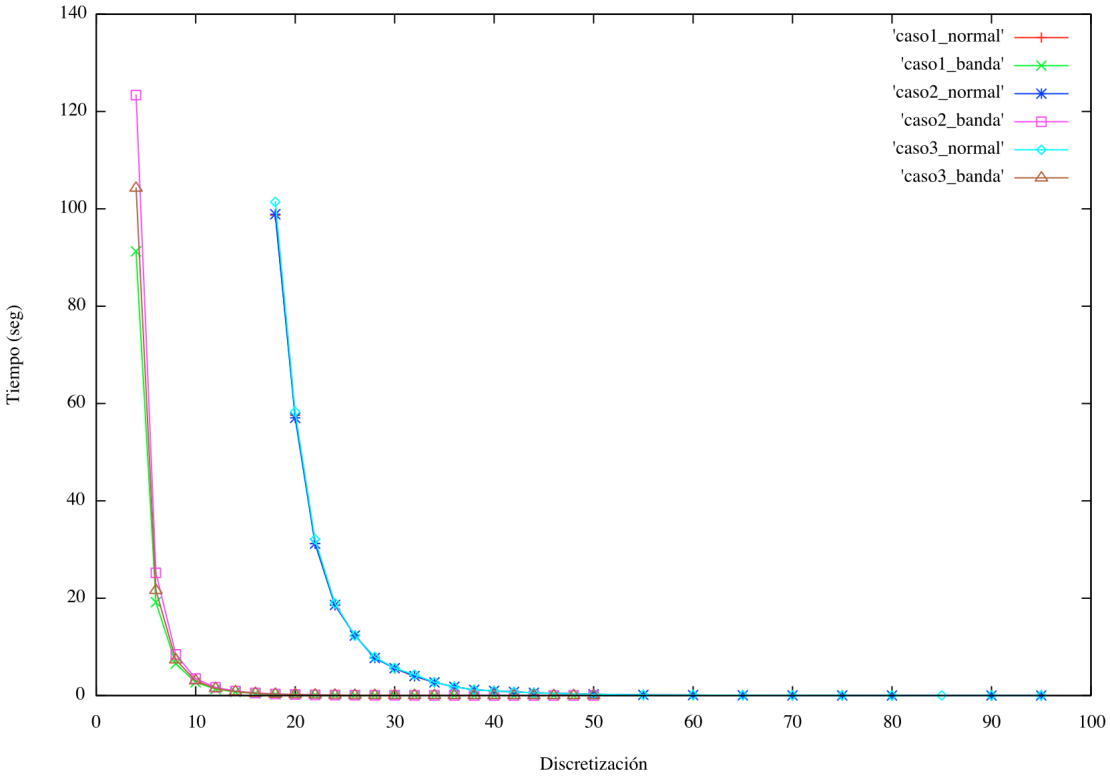


Figura 22: Comparación de tiempos

Por los resultados que obtuvimos se puede ver claramente que para un mismo algoritmo el tiempo de cómputo es similar en los 3 casos para cada valor de discretización, esto se debe a que el tamaño del parabrisas es el mismo. Además se puede ver que los dos algoritmos se comportan de una forma parecida (de forma exponencial a menor valor de discretización), con la diferencia de que los tiempos del algoritmo normal tardan 350 veces más aproximadamente, siendo esto esperado debido a que el algoritmo que utiliza la matriz banda optimiza cuestiones tanto espaciales como temporales

5.4. A matar sanguijuelas

A continuación analizaremos dos instancias de parabrisas con sanguijuelas para evaluar los resultados de las soluciones otorgadas por los algoritmos random y greedy.

5.5. Caso trivial

En esta instancia observaremos un caso en el que la única sanguijuela que se debe eliminar es la del centro. Esta se encuentra exactamente en el punto central. Adicionalmente hay otras 4 sanguijuelas en las puntas. Matar a estas no soluciona nada ya que la del centro esta aplicando una temperatura constante de 400 C°. Lo observaremos con 4 discretizaciones distintas para tener un mas amplio panorama.

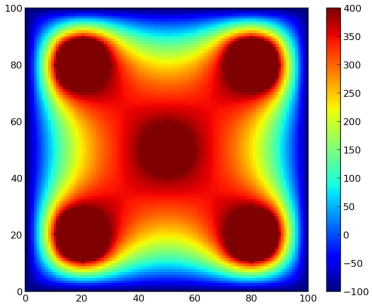


Figura 23: Granularidad 1

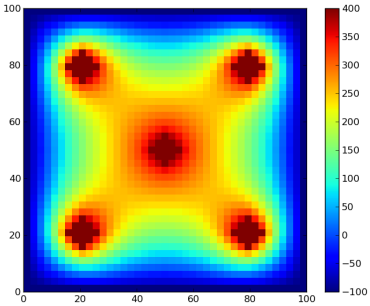


Figura 24: Granularidad 2.5

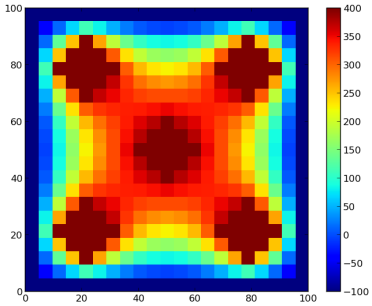


Figura 25: Granularidad 5

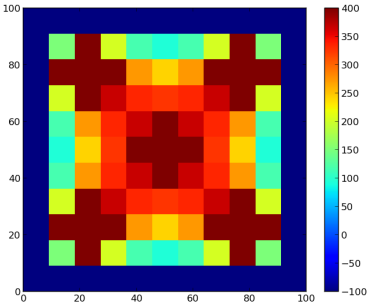


Figura 26: Granularidad 10

5.5.1. Resultados Random

Ahora veamos que obtenemos al aplicarle distintas veces el algoritmo de solución random explicado en el desarrollo (4.3) a la instancia con granularidad 2.5.

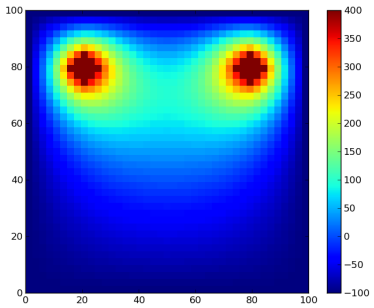


Figura 27: Resultado primer corrida

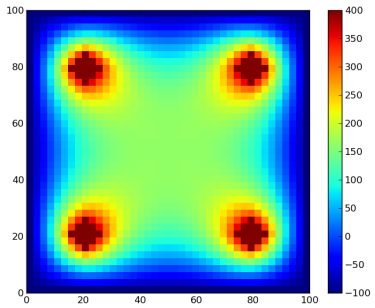


Figura 28: Resultado segunda corrida

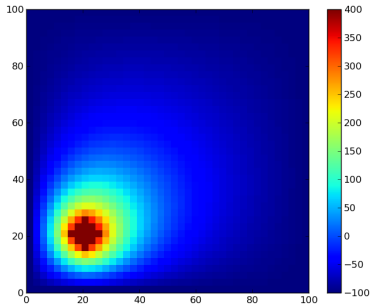


Figura 29: Resultado tercer corrida

Como podemos observar la solución óptima (la que menos sanguijuelas elimina), es la segunda. Pero como esta solución es completamente random en la primera corrida mata 2 sanguijuelas antes de elegir la correcta y en la tercera 4. Sólo en la segunda elije en el primer intento la sanguijuela correcta. Se aplicó también para todas las granularidades antes mencionadas y se observó el mismo comportamiento erratico a la hora de matar sanguijuelas.

5.5.2. Resultados Greedy

Veamos ahora para la misma matriz como se comporta nuestro otro algoritmo.

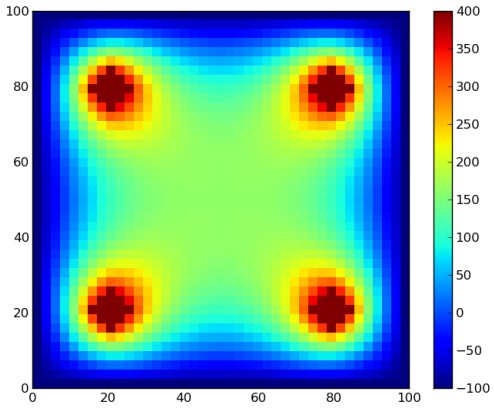


Figura 30: Resultado corrida con greedy

En todas las corridas obtuvimos el mismo resultado y demoraron lo mismo. Si bien con las otras granularidades el tiempo no fué el mismo (a menor granularidad, osea menor cantidad de celdas, menor tiempo de computo como vimos anteriormente) el resultado si lo fué. Era de esperarse, ya que efectivamente en este caso, la sanguijuela a eliminar era la del medio que naturalmente es la más cercana al punto crítico.

5.6. Caso solución no es el mas cercano

A continuación analizaremos un caso en el cual se debe eliminar una sanguijuela que no es la más cercana para efectivamente poder reducir la temperatura en el punto crítico. Lo chequearemos tanto para Greedy como para el Random, haciendo 3 corridas para este último ya que sabemos que su comportamiento es fluctuoso. Los datos exactos de este test son:
Medidas: 100 x 100
discretización:1
radio: 0.9
tempreratura sanguijuelas:450
Posiciónes sanguijuelas:(49, 38);(57, 60);(57, 37);(39, 37);(36, 47)

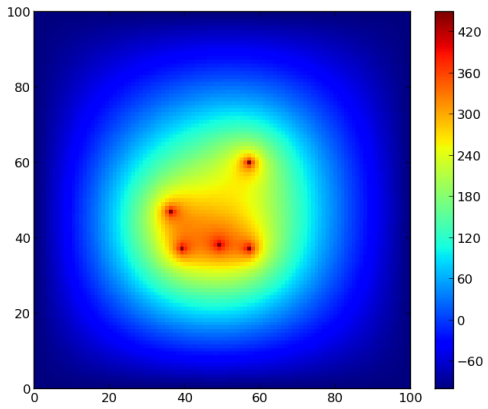


Figura 31: Parabrisas granularidad 1

Como podemos observar en las imágenes que están debajo, la solución greedy no es la óptima ya que mató a dos sanguijuelas y la random en algunos casos a una sola. Esto es debido a que la temperatura emitida por la sanguijuela mas cercana tiene un bajo impacto en el calor total recibido por el punto crítico, ya que este se encuentra 'rodeado' de sanguijuelas. Como la sanguijuela mas cercana tiene otras dos cerca de ella que también irradian calor al pc, si matamos a esta el punto central sigue recibiendo un considerable calor de la zona en la que se encontraba. En cambio si matamos alguna de las sanguijuelas que están mas aisladas esto cambia. Veamoslo en las soluciones obtenidas por nuestras algoritmos.

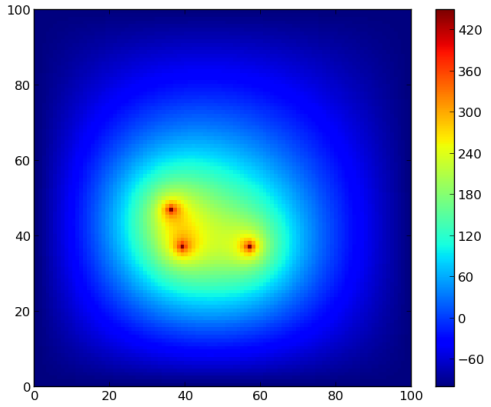


Figura 32: Solución Greedy

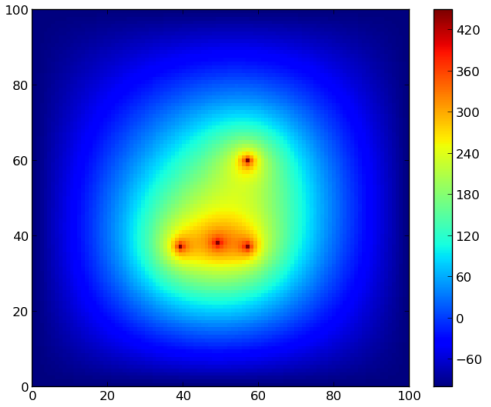


Figura 33: Solución Random 1

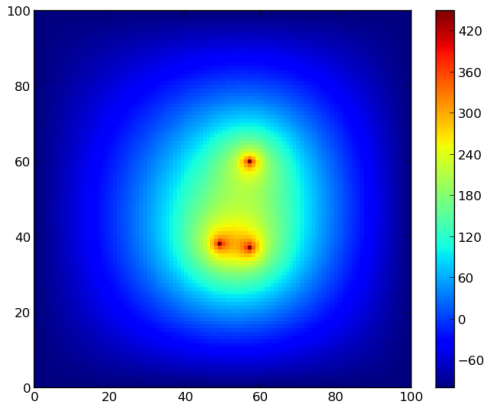


Figura 34: Solución random 2

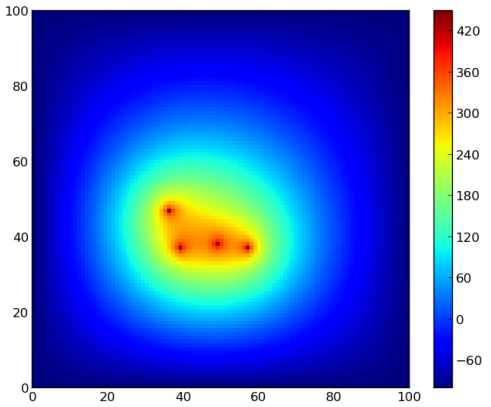


Figura 35: Solución random 3

El greedy mató primero a la sanguijuela de la posición (49,38) ya que era la mas cercana. Sin embargo si mataba primero la (57,60) la temperatura en el punto crítico disminuía lo suficiente. Esto podemos corroborarlo observando la tercera solución del Random. El random por otro lado nos dió 3 resultados distintos de los cuales dos son válidos. Cabe destacar sin embargo que el erroneo nos lo devolvió 2 veces, por lo que podemos decir que tuvo un 50 % de efectividad.

5.6.1. Variando la granularidad

Como vimos anteriormente, la temperatura en el punto crítico varía según la granularidad que le demos. Veamos entonces que pasa si aumentamos la granularidad (H = 0.8 en vez de 1).

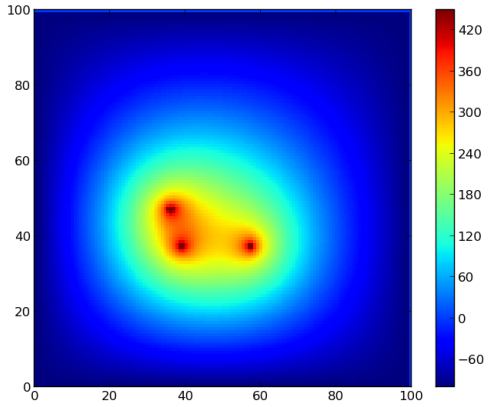


Figura 36: Solución Greedy

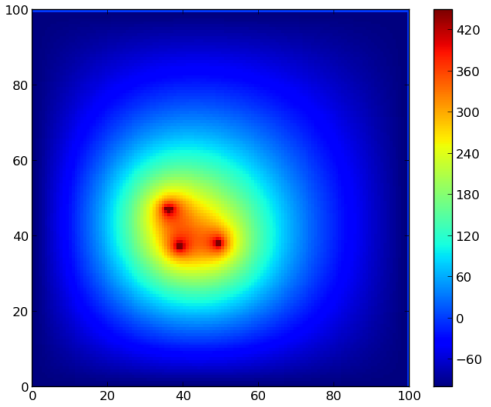


Figura 37: Solución Random 1

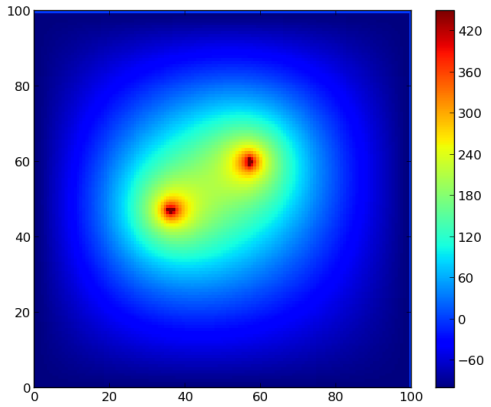


Figura 38: Solución Random 2

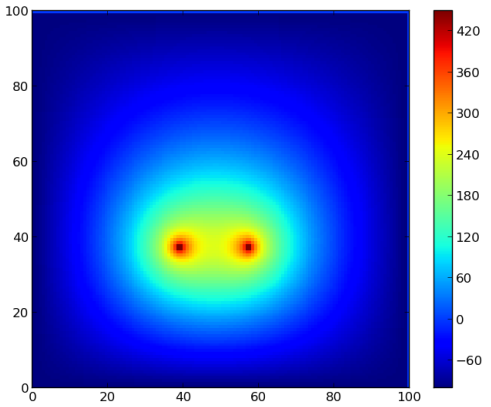


Figura 39: Solución Random 3

Como le aumentamos la granularidad aumentó la temperatura del punto central de tal forma que provocó que tampoco bastase eliminando una sola sangijuela (sea cual sea esta), obteniendo así la misma cantidad de sangijuelas tanto en al solución óptima como en la greedy. El Random esta vez devolvió un sólo resultado correcto ya que los otros dos mataron sanguijuelas de más.

6. Discusión

6.1. ¿Por que random?

Cuando empezamos a pensar otras soluciones alternativas a la golosa. Pensamos primero en soluciones totalmente suboptimas, como ir sacando desde el borde hacia el centro. Pero considerabamos que era tan malo que quitaba cualquier tipo de anàlisis serio. Luego también pensamos en ir sacando una del centro y una del borde alternadamente, pero no tenía tampoco algún justificativo conceptual alguno. Ahí fué entonces cuando pensamos en la solución random. Que claramente no responde a ningún compartamiento definido. Nos resultó mejor esto ya que las otras opciones pensadas o directamente estaban mal propuestas a proposito o no tenían sentido alguno. Además este comportamiento erratico podría llegar a darnos resultados interesantes o que rompiensen con nuestras intuiciones.

7. Conclusiones

7.1. Normal vs Banda

De acuerdo a la experimentación realizada y a los resultados obtenidos no cabe duda que para resolver el problema planteado la mejor solución es utilizar el algoritmo de matriz banda. Principalmente porque los dos algoritmos proporcionan resultados iguales y exactos que resuelven el problema planteado, pero el algoritmo de banda los resuelve en tiempo y espacio extremadamente menor, y esta diferencia se diferencia aún mas a medida que aumenta el tamaño del parabrisas y aumenta la granularidad del mismo, hasta el punto en que al algoritmo normal deja de resolverlo en un tiempo razonable o excede la memoria provocando que se detenga el proceso que ejecuta. Por este motivo se concluye totalmente que el problema se debe solucionar mediante el algoritmo de banda.

7.2. Random vs Greedy

Como observamos en los resultados para el caso en el que la sanguijuela/s se encuentran en el medio claramente es mejor el greedy, ya que siempre mata a la sanguijuela correcta a diferencia del random que sólo lo hizo en uno de los 3 intentos. Por otro lado si tenemos en cuenta los tiempos que pueden llegar a demorar recalcular las temperaturas, esto le da todavía una mayor importancia, ya que se reduce considerablemente el tiempo de ejecución.

Por otro lado en el caso en el que la sanguijuela a sacar no era la más cercana, el random en promedio obtuvo mejores resultados. Sin embargo al aumentarle la granularidad esto se revertió nuevamente debido a que aumentó la temperatura del PC y esto provocó que si o si fuese necesario matar por lo menos dos sanguijuelas, entre ellas la mas cercana.
Lo que pensamos con respecto a esto es que hay casos bastantes puntuales en los que el greedy falla y el random podría devolver una mejor solución, aunque esto también depende del porcentaje de sanguijuelas que se puedan sacar para solucionar el problema. Es decir, si de 5 sanguijuelas por ejemplo la solución no está en sacar la mas cercana sino cualquiera de las otras 4, el random sería el mejor, pero si la correcta fuera una sola, el random volvería a ser tan o mas ineficiente que el greedy ya que tendría una probabilidad de 1/5 de elegir la correcta la primera vez.
Osea, el random se comportaría mejor que el greedy sólo en el caso de que la sanguijuela a sacar no sea la más cercana y haya otras n/2 (o más) posibles para sacar de forma que resuelvan el problema.

Finalmente concluimos que los casos en los que el random sería mejor que el greedy son tan puntuales y nuestro problema, en principio, es tan genérico que son despreciables. Por lo que el greedy termina siendo claramente mejor.