Percolation and Critical Probability

Nicolás Rodríguez Perdomo,[1]

[1] *Estudiante de Física de la Universidad Nacional de Colombia*

[*] *nirodriguezp@unal.edu.co*

Luis Alejandro Rojas Gutierrez,[2]

[2] *Estudiante de Ingeniería Civil de la Universidad Nacional de Colombia*

[*] *lrojasg@unal.edu.co*

Samuel Jacobo Garavito Segura,[3]

[3] *Estudiante de Matemáticas de la Universidad Nacional de Colombia*

[*] *sgaravito@unal.edu.co*

Jose Esteban Beleno Barroso,[4]

[4] *Estudiante de Física de la Universidad Nacional de Colombia*

[*] *jbelenob@unal.edu.co*

Abstract: El siguiente informe documenta la simulación de un proceso de percolación. Por un lado, se discute acerca de la probabilidad de la existencia de un cluster percolante en función del tamaño y probabilidad de llenado de una malla bidimensional al igual que el comportamiento del tamaño promedio del cluster percolante más grande. Por otro lado, se estudia la eficiencia computacional de la implementación realizada mediante herramientas de optimización y profiling.

Key Words: Simulación · Cluster · Percolación · Probabilidad · Optimización.

Date: 20 de Mayo de 2022.

Introducción

La percolación está relacionada con el paso de una sustancia, elemento o información de una frontera a otra, generalmente a través de un filtro. En este proyecto se simulará el llenado aleatorio de una malla bidimensional para simular el fenómeno de percolación de sitio.

El procedimiento es el siguiente. Se escoge una probabilidad de llenado p, y se recorrerá los sitios de una matriz de tamaño $L \times L$ uno por uno llenando cada componente dependiendo de dicha probabilidad. Teniendo en cuenta solamente vecinos inmediatos (arriba-abajo-izquierda-derecha), el anterior llenado hará que aparezcan clusters de vecinos con tamaños específicos que dependerán de esa probabilidad p. Si la probabilidad es baja, pocos sitios quedarán llenados y estarán disconexos. Si la probabilidad de llenada es alta, muchos sitios estarán llenos y se podran ver regiones de vecinos que atraviesan todo el sistema. A estos grupos de vecinos conectados se les llama clusters, y si hay un cluster que pasa de lado a lado en el sistema se dice que hay cluster percolante. Existe una probablidad crítica, p_c, después de la cual siempre aparecerá un cluster percolante (que une de arriba a abajo o de izquierda a derecha), cuyo tamaño será cada vez mayor a medida que aumenta $p > p_c$, limitado por el tamaño finito del sistema.

En el presente informe se responderá y discutirá los siguientes interrogantes.

- ¿Cuál es la probabilidad $P(p,L)$ de que exista un cluster percolante, en función de p y de L? Esto se resumirá en una figura de P en función de p, para diferentes L.
- ¿Cuál es el tamaño promedio del clúster percolante más grande, $s(p,L)$ en función de p y L? El tamaño del cluster es simplemente el número de sitios que pertenecen al cluster y estara normalizado al tamaño total del sistema, $L \times L$.

- ¿Qué tan óptima puede llegar a ser la implementación realizada para la simulación del proceso de percolacion?. Esto se realizará por medio de herramientas de optimización y profiling.

El informe cuenta con cuatro (4) secciones: *Introducción, Metodología, Análisis y Resultados, y Conclusiones.* Respectivamente, corresponden a: una introducción al contexto del problema, muestra de como se abordo el problema, resultados obtenidos y su interpretación ante el problema, y por ultimo las conclusiones del problema.

Metodología

La implementación del algoritmo se realizó en el lenguaje de programación **C++** y las representaciones gráficas de los resultados de la simulación por medio de la herramienta de Gnuplot. Por otro lado, para el estudio computacional del algoritmo se emplearon las banderas de optimización -O0 -O1 -O2 -O3, y como bandera de profiling $-pg$.

Simulación y Algoritmo

Para el proceso de simulación del llenado aleatorio de una malla bidimensional se ha empleado una distribución *Bernoulli* de parámetro p. Dada una matriz cuadrada de tamaño $L \times L$, esta se recorre componente por componente para determinar su valor como éxito (1) o fracaso (0) en ese espacio.

Ya hecho esto, inicia el algoritmo para la determinación de clusters. Se expondrá visualmente con un ejemplo en concreto para lograr una mejor comprensión. Tomando la notación usual de una matriz:

1	1	0	1
0	1	0	1
1	1	1	0
1	0	0	1

Paso 1.
Se llena la matriz con 0's y 1's.

2	1	0	1
0	1	0	1
1	1	1	0
1	0	0	1

Paso 2.
Se inicia el recorrido de la matriz donde en la primera componente se encuentra con un 1, este pasa a ser un 2.

2	3	0	1
0	1	0	1
1	1	1	0
1	0	0	1

Paso 3.
Se buscan los vecinos de la primer componente. Ya que en la componente a_{12} hay un 1, este pasa a ser un 3.

2	3	0	1
0	4	0	1
1	1	1	0
1	0	0	1

Paso 4.
Se buscan los vecinos **no previstos** de la componente a_{12}. Ya que en la componente a_{22} hay un 1, este pasa a ser un 4.

2	3	0	1
0	4	0	1
1	5	1	0
1	0	0	1

Paso 5.
Se buscan los vecinos **no previstos** de la componente a_{22}. Ya que en la componente a_{32} hay un 1, este pasa a ser un 5.

2	3	0	1
0	4	0	1
6	5	6	0
1	0	0	1

Paso 6.
Se buscan los vecinos **no previstos** de la componente a_{32}. Ya que en las componentes a_{31} y a_{33} hay un 1, estos pasan a ser un 6.

2	3	0	1
0	4	0	1
6	5	6	0
7	0	0	1

Paso 7.
Se buscan los vecinos **no previstos** de las componentes a_{31} y a_{33}. Como a_{33} no tiene vecinos, el proceso muere para esta rama. Por otro lado, para a_{31}, ya que en la componente a_{41} hay un 1, este pasa a ser un 7.

2	3	0	1
0	4	0	1
6	5	6	0
7	0	0	1

Paso 8.
Se buscan los vecinos **no previstos** de la componente a_{41}. En este punto ya no se encuentran mas vecinos, por ende el proceso muere para esta rama.

2	2	0	1
0	2	0	1
2	2	2	0
2	0	0	1

Paso 9.
Al no haber mas vecinos, se ha encontrado un cluster. Para delimitarlo, todas las componentes mayores al valor de la primera componente donde inicio el proceso del cluster, en este caso 2, se asignan con dicho valor.

2	2	0	3
0	2	0	4
2	2	2	0
2	0	0	1

Paso 10.
El algoritmo continua buscando 1's. Como en la componente a_{14} hay uno, este pasa a ser un 3. Este método de asignación de valores, en el momento del escaneo del cluster evita repetir componentes. Adicionalmente, como a_{24} es vecino y es 1, pasa a ser un 4.

2	2	0	3
0	2	0	3
2	2	2	0
2	0	0	1

Paso 11.
Puesto que la componente a_{24} no posee vecinos no previstos, el proceso muere para esta rama. Para delimitarlo, todas las componentes mayores al valor de la primera componente donde inicio el proceso del cluster, en este caso 3, se asignan con dicho valor.

2	2	0	3
0	2	0	3
2	2	2	0
2	0	0	4

Paso 12.
El algoritmo continua buscando 1's. Como en la componente a_{44} hay uno, este pasa a ser un 4. Al no tener vecinos y ser la ultima componente el algoritmo termina.

El algoritmo ha terminado. Los clusters han sido clasificados y se obtiene una nueva matriz, ahora determinaremos si existe cluster percolante. Para dicha tarea, se implementa una función la cual recibe como argumento esta nueva matriz, inicialmente se recorre la primera fila de dicha matriz para encontrar un cluster . Si lo encuentra, recorre la ultima fila para determinar si existe o no una componente del mismo cluster, esto mediante los valores asignados durante el algoritmo. Si también lo encuentra, existe cluster percolante, si no, no existe. Esto para encontrar los clusters percolantes verticales. Para clusters percolantes horizontales es análogo, se estudia para las columnas.

Finalmente, para determinar el tamaño del cluster percolante, se implemento una función cuyos argumentos son la nueva matriz y el valor identificativo del cluster, esta cuenta cuantas componentes concuerdan con dicho valor. El proceso se realiza para todo cluster percolante. El máximo de estos números corresponde al tamaño de cluster percolante mas grande.

Contenido del Repositorio

En el repositorio de GitHub: **2022-i-herrcomp-proyectointermedio-grupo-28-herr-comp** encontrará cuatro (4) directorios y cuatro (4) archivos. Los cuatro directorios corresponden a **Códigos principales, Datos, Gráficos y Latex.** En el primer directorio se encuentra todo el código necesario para el correcto funcionamiento de la simulación y algoritmo de clusters percolante; este cuenta con los siguientes archivos cada uno con su asignación

descrita a continuación:

- **Makefile:** Makefile que compila, ejecuta, crea datos y figuras, y realiza tests.
- **percolation_problem.h**: Header que involucra la declaración de todas las funciones implementadas en **percolation_problem.cpp**.
- **calculating_probabilities.cpp:** cpp encargado del reporte de promedios y desviaciones estándar.
- **largest_cluster_size.cpp:** cpp encargado de la determinación del tamaño del cluster percolante mas grande.
- **percolation_problem.cpp:** cpp que involucra la implementación de todas las funciones necesarias en simulación del proceso de percolacion. Las funciones son:
  - ***void fill_matrix(std::vector <long> & data, int seed, double p)*:** Llena la matriz con probabilidad de llenado p. *seed* es la semilla de la distribución que usa para llenar la matriz.
  - ***void print_matrix(const std::vector <long> & data)*:** Imprime en pantalla los elementos de la matriz.
  - ***long find_nonclassified_cluster(std::vector <long> & data)*:** Es una función que va a ser útil para la clasificación de los clusters. Encargada de buscar el primer elemento de un cluster que no ha sido clasificado (es decir un \1") y retornar la posición de ese elemento.
  - ***long cluster_size(const std::vector <long> & data, const long cluster_num)*:** Retorna el tamaño del cluster número $\backslash cluster\_num$".
  - ***void classify_clusters(std::vector <long> & data)*:** Esta función realiza el algoritmo de clasificar los clusters.
  - ***void attach_neighbours(std::vector <long> & data, const long N)*:** Clasificando un cluster dado, la función se encarga de ir añadiendo elementos al cluster si sus vecinos están llenos.
  - ***bool is_percolating(const std::vector <long> & data, const long cluster_num)*:** Determina si el cluster número $cluster\_num$ es percolante.
  - ***bool is_there_a_percolating(const std::vector <long> & data)*:** Determina si hay algún cluster percolante en la matriz.
  - ***long largest_percolating_cluster_size(const std::vector <long> & data)*:** Retorna el tamaño de cluster percolante más grande. Si no hay cluster percolante entonces retorna 0.
  - ***long number_of_last_cluster(const std::vector <long> & data)*:** Como los cluster son enumerados, esta función retorna el número del último cluster clasificado.
  - ***double compute_mean(const std::vector <double> & data)*:** Retorna la media de un conjunto de medidas.
  - ***double standard_deviation(const std::vector <double> & data, double mean)*:** Retorna la desviación estándar de un conjunto de medidas.
  - ***void compute_mean_and_standard_deviation_for_percolating_probability(long size, double probability, int reps_per_single_calculation, int number_of_calculations, int seed)*:** Imprime en pantalla la probabilidad de que aparezca un cluster percolante si se llena la matriz con una probabilidad p con su desviación estándar.$reps\_per\_single\_calculation$ indica cuantas veces se repite la simulación para tomar la medida 1 sola vez. $number\_of\_calculations$ indica cuantas veces se calcula la probabilidad, y *seed* es una semilla que va a permitir controlar las simulaciones.
- **percolation_test.cpp**: cpp encargado de probar las funciónes que detectan clusters y que calculan el tamaño del cluster percolante más grande por medio de catch.

  single_simulation.cpp: cpp encargado de hacer el calculo de la probabilidad de que aparezca un cluster percolante dados el tamaño de la matriz y la probabilidad de llenado.

Por otro lado, en el directorio tanto de **Datos como de Gráficos**, encontrara todos los resultados de los casos específicos estudiados. Por ultimo, en el directorio **Latex** encuentra todo el código LaTeX y sus dependencias. En cuanto a los cuatro archivos restantes, corresponden al pdf del informe, descripciónes y txt del profiling.

profiling-report.txt: Reporte de profiling *(flat profile)* para el proceso de simulación $p = 0.611$ y $L = 128$.

Análisis y Resultados

Probabilidad de Cluster Percolante

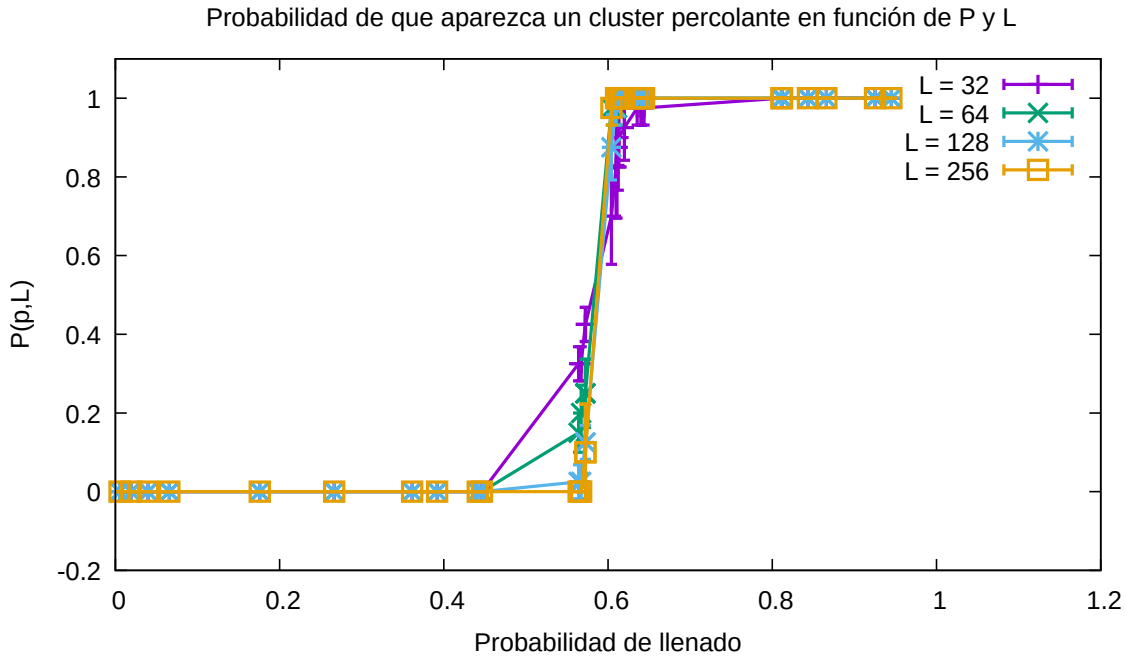

Fig. 1. Gráfico de $P(p,L)$ para valores de p en el intervalo $[0,1]$

.

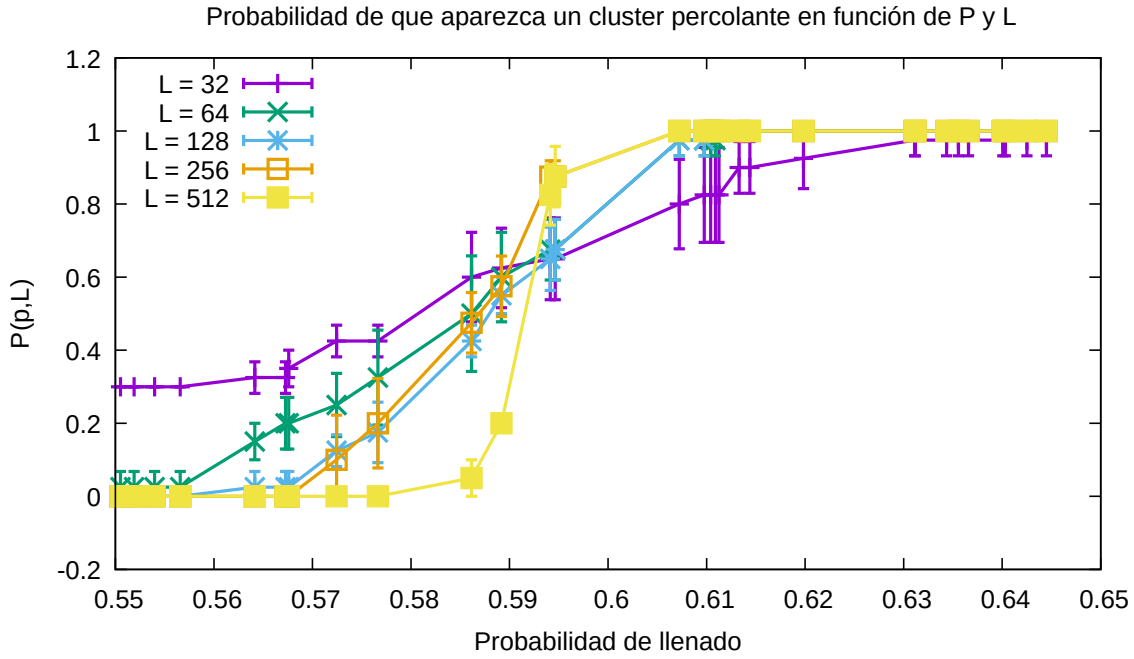

Fig. 2. Gráfico de $P(p,L)$ para valores de p en el intervalo $[0.55,0.65]$

.

Para cada probabilidad p se ejecutaron diez (10) repeticiones de simulación para obtener $P'(p,L)$. Este proceso se repitió cuatro (4) veces con el propósito de adquirir una mayor precisión; a partir de esos nuevos resultados se obtuvo su promedio, que corresponde a $P(p,L)$, y desviación estándar (σ). Cabe resaltar que a mayor fuese el numero de repeticiones tanto para calcular $P'(p,L)$ como para calcular $P(p,L)$, los datos nos proporcionarían estadísticas cada vez mas confiables, no obstante, demandarían una capacidad de computo mas elevada; con estos números se logra un equilibrio.

Inicialmente, mediante un **rápido escaneo visual a la figura 1.** es posible evidenciar varios comportamientos que vale la pena resaltar. De primera mano, vemos que entre mayor sea el tamaño de la malla, tarda mas en ocurrir la aparición del primer cluster percolante, sin embargo, también estas son las primeras en alcanzar el valor critico $p_c(L)$. Ligado a lo anterior, entre mayor es el tamaño de la malla, la distribución de $P(p,L)$ posee menor variación, los datos se van acumulando en un intervalo $[0.55+\delta, 0.65-\delta]$ con $0<\delta$. Es por esto que para cada $L \in \{32,64,128,256,512\}$ se analizara el comportamiento de las probabilidades en ese intervalo. Para cada tamaño L se estudio el comportamiento de $P(p,L)$ con 30 probabilidades p.

La **figura 2.** corresponde a un resumen gráfico del análisis de los resultados que se presentaran en seguida. Al ser una simulación, los resultados obtenidos son aproximaciones de los datos teóricos.

Resultados para valores de p entre 0.55 y 0.65

i	p_i	$P(p,L)$	σ (std)
1	5.5051591501e-01	3.e-01	0.e+00
2	5.5190247412e-01	3.e-01	0.e+00
3	5.5397803623e-01	3.e-01	0.e+00
4	5.5657095388e-01	3.e-01	0.e+00
5	5.6414641729e-01	3.25e-01	4.3301270189e-02
6	5.6726953251e-01	3.25e-01	4.3301270189e-02
7	5.6758612351e-01	3.5e-01	5.0000000000e-02
8	5.7245059269e-01	4.25e-01	4.3301270189e-02
9	5.7664737833e-01	4.25e-01	4.3301270189e-02
10	5.8614963416e-01	6.0000000000e-01	1.2247448714e-01
11	5.8917661955e-01	6.25e-01	1.0897247359e-01
12	5.9413485334e-01	6.5e-01	1.1180339887e-01
13	5.9463492008e-01	6.5e-01	1.1180339887e-01
14	6.0723564919e-01	8.e-01	1.2247448714e-01
15	6.0975562059e-01	8.25e-01	1.2990381057e-01
16	6.1041319373e-01	8.25e-01	1.2990381057e-01
17	6.1090355983e-01	8.25e-01	1.2990381057e-01
18	6.1125253432e-01	8.25e-01	1.2990381057e-01
19	6.1329630996e-01	9.e-01	7.0710678119e-02
20	6.1439142632e-01	9.e-01	7.0710678119e-02
21	6.1984202394e-01	9.25e-01	8.2915619759e-02
22	6.3110621048e-01	9.75e-01	4.3301270189e-02
23	6.3119477503e-01	9.75e-01	4.3301270189e-02
24	6.3435281189e-01	9.75e-01	4.3301270189e-02
25	6.3556209451e-01	9.75e-01	4.3301270189e-02
26	6.3658827551e-01	9.75e-01	4.3301270189e-02
27	6.4006214549e-01	9.75e-01	4.3301270189e-02
28	6.4031785264e-01	9.75e-01	4.3301270189e-02
29	6.4251531592e-01	9.75e-01	4.3301270189e-02
30	6.445003845e-01	9.75e-01	4.3301270189e-02

Table 1. Resultados para $L = 32$

Para el tamaño $L = 32$, se observa que en el intervalo $[0.55, 0.65]$ no se alcanza el p_c, pero se esta muy cerca de hacerlo. Por otro lado, se ven bastantes $P(p,L)$ repetidos para varias p, este factor es importante para la explicación que se dará mas adelante. La desviación estándar es nula para los primeros cuatro valores y después oscila entre los ordenes de 10^{-1} y 10^{-2} como pasara en la mayoría de los casos siguientes, esto indica que las probabilidades utilizadas para obtener el promedio no varían mucho y son fiables. Continuaremos para el siguiente tamaño en aras de iniciar a hacer comparaciones.

i	p_i	$P(p,L)$	σ (std)
1	5.5051591501e-01	2.5e-02	4.3301270189e-02
2	5.5190247412e-01	2.5e-02	4.3301270189e-02
3	5.5397803623e-01	2.5e-02	4.3301270189e-02
4	5.5657095388e-01	2.5e-02	4.3301270189e-02
5	5.6414641729e-01	1.5e-01	5.e-02
6	5.6726953251e-01	2.e-01	7.0710678119e-02
7	5.6758612351e-01	2.e-01	7.0710678119e-02
8	5.7245059269e-01	2.5e-01	8.6602540378e-02
9	5.7664737833e-01	3.25e-01	1.2990381057e-01
10	5.8614963416e-01	5.e-01	1.5811388301e-01
11	5.8917661955e-01	6.0000000000e-01	1.2247448714e-01
12	5.9413485334e-01	6.75e-01	8.2915619759e-02
13	5.9463492008e-01	6.75e-01	8.2915619759e-02
14	6.0723564919e-01	9.75e-01	4.3301270189e-02
15	6.0975562059e-01	9.75e-01	4.3301270189e-02
16	6.1041319373e-01	9.75e-01	4.3301270189e-02
17	6.1090355983e-01	9.75e-01	4.3301270189e-02
18	6.1125253432e-01	1.e+00	0.e+00
19	6.1329630996e-01	1.e+00	0.e+00
20	6.1439142632e-01	1.e+00	0.e+00
21	6.1984202394e-01	1.e+00	0.e+00
22	6.3110621048e-01	1.e+00	0.e+00
23	6.3119477503e-01	1.e+00	0.e+00
24	6.3435281189e-01	1.e+00	0.e+00
25	6.3556209451e-01	1.e+00	0.e+00
26	6.3658827551e-01	1.e+00	0.e+00
27	6.4006214549e-01	1.e+00	0.e+00
28	6.4031785264e-01	1.e+00	0.e+00
29	6.4251531592e-01	1.e+00	0.e+00
30	6.445003845e-01	1.e+00	0.e+00

Table 2. Resultados para $L = 64$

Note que para este caso, $L = 64$, en el intervalo se alcanza $p_c(64) = 6.1125253432e-01$ con p_{18}. También, observe que para cada p, $P(p,L)$ se repite menos, es decir $P(p,L)$ en proporción, adopta mas valores que para el caso de $L = 32$. Por otra parte, comparando con los primeros p_i de $L = 32$, la certeza de que aparezca el primer cluster percolante disminuyo, como ya habíamos previsto desde el estudio de la **figura 1.**. Este comportamiento sera mas evidente a medida que L aumente.

Para el tamaño $L = 128$ se observan cambios significativos. Nuevamente, se alcanza $p_c(128) = 6.1041319373e-01$ en el intervalo con p_{16}, una medida muy similar a la de $p_c(64)$. Asimismo, la certeza de la aparición del primer cluster ahora es nula para los primeros cuatro valores, es a partir de p_5 que inicia el incremento. De aquí en adelante, se introducirá el concepto de **intervalo critico para** L dado por $[p', p_c]_L$ donde p' es la probabilidad a partir de la cual se tiene certeza de que aparece el primer cluster percolante y p_c la probabilidad critica. Se denotara $|[p', p_c]_L|$ como la longitud del intervalo. Para este caso en concreto $[p', p_c]_{128} = [5.6414641729e-01, 6.1041319373e-01]$ y $|[p', p_c]_{128}| = 4.626677644e-02$. No se había introducido este concepto anteriormente

puesto que no era muy claro para $L = 32, 64$.

i	p_i	$P(p,L)$	σ (std)
1	5.5051591501e-01	0.e+00	0.e+00
2	5.5190247412e-01	0.e+00	0.e+00
3	5.5397803623e-01	0.e+00	0.e+00
4	5.5657095388e-01	0.e+00	0.e+00
5	5.6414641729e-01	2.5e-02	4.3301270189e-02
6	5.6726953251e-01	2.5e-02	4.3301270189e-02
7	5.6758612351e-01	2.5e-02	4.3301270189e-02
8	5.7245059269e-01	1.25e-01	4.3301270189e-02
9	5.7664737833e-01	1.75e-01	8.2915619759e-02
10	5.8614963416e-01	4.25e-01	4.3301270189e-02
11	5.8917661955e-01	5.5e-01	5.e-02
12	5.9413485334e-01	6.5e-01	8.6602540378e-02
13	5.9463492008e-01	6.75e-01	8.2915619759e-02
14	6.0723564919e-01	9.75e-01	4.3301270189e-02
15	6.0975562059e-01	9.75e-01	4.3301270189e-02
16	6.1041319373e-01	1.e+00	0.e+00
17	6.1090355983e-01	1.e+00	0.e+00
18	6.1125253432e-01	1.e+00	0.e+00
19	6.1329630996e-01	1.e+00	0.e+00
20	6.1439142632e-01	1.e+00	0.e+00
21	6.1984202394e-01	1.e+00	0.e+00
22	6.3110621048e-01	1.e+00	0.e+00
23	6.3119477503e-01	1.e+00	0.e+00
24	6.3435281189e-01	1.e+00	0.e+00
25	6.3556209451e-01	1.e+00	0.e+00
26	6.3658827551e-01	1.e+00	0.e+00
27	6.4006214549e-01	1.e+00	0.e+00
28	6.4031785264e-01	1.e+00	0.e+00
29	6.4251531592e-01	1.e+00	0.e+00
30	6.445003845e-01	1.e+00	0.e+00

Table 3. Resultados para $L = 128$

Llegados a $L = 256$, el comportamiento que se ha venido evidenciando se ratifica nuevamente. Para este caso $[p', p_c]_{256} = [5.7245059269e-01, 6.0723564919e-01]$ y $|[p', p_c]_{256}| = 3.47850565e-02$. Con estos dos conceptos, se resumen los resultados que han venido siendo explicado para los L anteriores.

Anteriormente, se menciono el hecho de que a medida que L aumenta, $P(p,L)$ se repite menos, es decir $P(p,L)$ en proporción, adopta mas valores que para el caso de $L-1$. Esto se debe a que a medida que aumenta el tamaño de la matriz las alternativas de llenado de un espacio con 1 son mayores. No es lo mismo, llenar una matriz de dimensión $n = 2$ con una probabilidad de 0.65 a llenar una de $n = 100$ con la misma probabilidad por el simple hecho de que al menos el 50% de la misma debe corresponder a $1's$, es decir, dos (2) componentes, no obstante en caso de no existir aun el cluster percolante, solo existen dos espacios los cuales se pueden llenar, mientras que para $n = 100$ bajo una probabilidad de 0.65 al menos 50 componentes corresponden al valor de 1. Tomando el peor de los casos, el caso de la **distribución en ajedrez** *(explicado posteriormente)* se tienen 15 casillas mas para poder determinar un cluster percolante. A medida que el tamaño aumente se tienen mas casillas y mas posibles combinaciones para poder determinar un cluster percolante.

Después de presentar los resultados para cada L, este hecho quedara aun mas claro.

i	p_i	$P(p,L)$	σ (std)
1	5.5051591501e-01	0.e+00	0.e+00
2	5.5190247412e-01	0.e+00	0.e+00
3	5.5397803623e-01	0.e+00	0.e+00
4	5.5657095388e-01	0.e+00	0.e+00
5	5.6414641729e-01	0.e+00	0.e+00
6	5.6726953251e-01	0.e+00	0.e+00
7	5.6758612351e-01	0.e+00	0.e+00
8	5.7245059269e-01	1.e-01	1.2247448714e-01
9	5.7664737833e-01	2.e-01	1.2247448714e-01
10	5.8614963416e-01	4.75e-01	8.2915619759e-02
11	5.8917661955e-01	5.75e-01	8.2915619759e-02
12	5.9413485334e-01	8.75e-01	4.3301270189e-02
13	5.9463492008e-01	8.75e-01	4.3301270189e-02
14	6.0723564919e-01	1.e+00	0.e+00
15	6.0975562059e-01	1.e+00	0.e+00
16	6.1041319373e-01	1.e+00	0.e+00
17	6.1090355983e-01	1.e+00	0.e+00
18	6.1125253432e-01	1.e+00	0.e+00
19	6.1329630996e-01	1.e+00	0.e+00
20	6.1439142632e-01	1.e+00	0.e+00
21	6.1984202394e-01	1.e+00	0.e+00
22	6.3110621048e-01	1.e+00	0.e+00
23	6.3119477503e-01	1.e+00	0.e+00
24	6.3435281189e-01	1.e+00	0.e+00
25	6.3556209451e-01	1.e+00	0.e+00
26	6.3658827551e-01	1.e+00	0.e+00
27	6.4006214549e-01	1.e+00	0.e+00
28	6.4031785264e-01	1.e+00	0.e+00
29	6.4251531592e-01	1.e+00	0.e+00
30	6.445003845e-01	1.e+00	0.e+00

Table 4. Resultados para $L = 256$

Finalmente, para $L = 512$, $[p', p_c]_{512} = [5.8614963416e-01, 6.0723564919e-01]$ y $|[p', p_c]_{512}| = 2.108601503e-02$. En este punto, vistos ya todos los casos de L es posible redactar la siguiente conjetura.

Conjetura: Sea $[p', p_c]_L$ el intervalo critico para L y $|[p', p_c]_L|$ su longitud. Sea $\{|[p', p_c]_i|\}_{i\in\mathbb{N}}$ una sucesión en $\mathbb{R}$. Entonces esta la sucesión es convergente y $\lim_{i\to\infty} |[p', p_c]_i| = 0$.

La sucesión se ha visto que es decreciente, $|[p', p_c]_{128}| > |[p', p_c]_{256}| > |[p', p_c]_{512}|$ y es acotada inferiormente de manera trivial por 0. Experimentalmente, esta sucesión es convergente y pareciera que tiende a un valor muy pequeño, por ende se postula a 0 como candidato a ser ese limite.

En aras de sustentar aun mejor esta conjetura, a continuación se abordara el problema desde una perspectiva **TEÓRICA**. Dada una matriz cuadrada P de tamaño n con $n \in \mathbb{N}$ definida de la siguiente manera:

$$P = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix}$$

i	p_i	$P(p,L)$	σ (std)
1	5.5051591501e-01	0.e+00	0.e+00
2	5.5190247412e-01	0.e+00	0.e+00
3	5.5397803623e-01	0.e+00	0.e+00
4	5.5657095388e-01	0.e+00	0.e+00
5	5.6414641729e-01	0.e+00	0.e+00
6	5.6726953251e-01	0.e+00	0.e+00
7	5.6758612351e-01	0.e+00	0.e+00
8	5.7245059269e-01	0.e+00	0.e+00
9	5.7664737833e-01	0.e+00	0.e+00
10	5.8614963416e-01	5.e-02	5.e-02
11	5.8917661955e-01	2.e-01	0.e+00
12	5.9413485334e-01	8.2500000000e-01	8.2915619759e-02
13	5.9463492008e-01	8.75e-01	8.2915619759e-02
14	6.0723564919e-01	1.e+00	0.e+00
15	6.0975562059e-01	1.e+00	0.e+00
16	6.1041319373e-01	1.e+00	0.e+00
17	6.1090355983e-01	1.e+00	0.e+00
18	6.1125253432e-01	1.e+00	0.e+00
19	6.1329630996e-01	1.e+00	0.e+00
20	6.1439142632e-01	1.e+00	0.e+00
21	6.1984202394e-01	1.e+00	0.e+00
22	6.3110621048e-01	1.e+00	0.e+00
23	6.3119477503e-01	1.e+00	0.e+00
24	6.3435281189e-01	1.e+00	0.e+00
25	6.3556209451e-01	1.e+00	0.e+00
26	6.3658827551e-01	1.e+00	0.e+00
27	6.4006214549e-01	1.e+00	0.e+00
28	6.4031785264e-01	1.e+00	0.e+00
29	6.4251531592e-01	1.e+00	0.e+00
30	6.445003845e-01	1.e+00	0.e+00

Table 5. Resultados para $L = 512$

Se estipula una probabilidad de llenado de la matriz es p. Dada esta probabilidad, el numero de a_{ij} con $1 \leq i, j \leq n$ tal que $a_{ij} = 1$ esta altamente ligado a ella. Esta afirmación suena obvia, sin embargo observe el siguiente caso. Si $n = 100$, dada p habrán al menos $\lfloor p \cdot 100 \rfloor$ componentes iguales a 1. Si $n = 1000$, dada p habrán al menos $\lfloor p \cdot 1000 \rfloor$ componentes iguales a 1. Con el fin de obtener la probabilidad mínima de que aparezca el primer cluster percolante analizaremos el siguiente caso :

$$P = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

La proporción de $1's$ en el sistema debe ser al menos de $\frac{1}{n}$. Por consiguiente la probabilidad a partir de la cual es seguro que puede aparecer el primer cluster percolante esta dada por $p(n) = \frac{1}{n}$. Cabe aclarar que existen $p_i < \frac{1}{n}$ tales que puede aparecer un cluster percolante, no obstante para estas no existirá siempre la **posibilidad** de que este cluster aparezca. Teniendo este $p(n)$, para que ocurra el caso de la matriz de arriba, la probabilidad de que se llenen n casillas seguidas esta dada por $\left(\frac{1}{n}\right)^n$. Vea que dados $n, m \in \mathbb{N}$ tales que $n < m$, se tiene que $\left(\frac{1}{m}\right)^m < \left(\frac{1}{n}\right)^n$. Es por esta misma razón que en la **figura 1.** se evidencia el comportamiento de que a medida que aumenta el

tamaño de la malla, tarda mas en aparecer el primer cluster percolante. Ahora, forzaremos la aparición del cluster.

Supongamos que n es par y $p = 0.5$. Consideraremos el peor caso en el que no se ha formado un cluster percolante y los $1's$ están distribuidos en la matriz como un tablero de ajedrez. Es decir:

$$P = \begin{pmatrix} 1 & 0 & 1 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 1 \\ 1 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & 1 & \vdots \\ 0 & 1 & 0 & \cdots & 1 \end{pmatrix}$$

Alcanzado este punto, para que se de la aparición del cluster, por lo menos deben haber $\frac{n}{2}$ $1's$ mas:

$$P = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & 1 & 0 & \cdots & 1 \\ 1 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & 1 & \vdots \\ 0 & 1 & 0 & \cdots & 1 \end{pmatrix}$$

De este modo, para que se logre esto $0.5 + \frac{1}{2n} < p_c(n)$. No obstante este caso precisa de una probabilidad $(p)^{\frac{n}{2}}$, muy baja, lo cual nos lleva a concluir que $p_c(n) \cong 0.5 + \frac{1}{2n} + \varepsilon(L)$ para $0 < \varepsilon(L)$. Este resultado es una aproximación bastante acertada pues teóricamente $p_c = 0.59271$. Complementando la conjetura, cuando $\lim_{i\to\infty} |[p', p_c]_i| = 0$ se tiene que $p' = p_c$. Esta parte del estudio se realizo para matrices de dimensión par, puesto que solo hay una forma de distribuirlas de la forma del ajedrez. Para matrices impares existen dos formas.

A continuación se estudiara el tamaño del cluster percolante.

Tamaño de cluster percolante

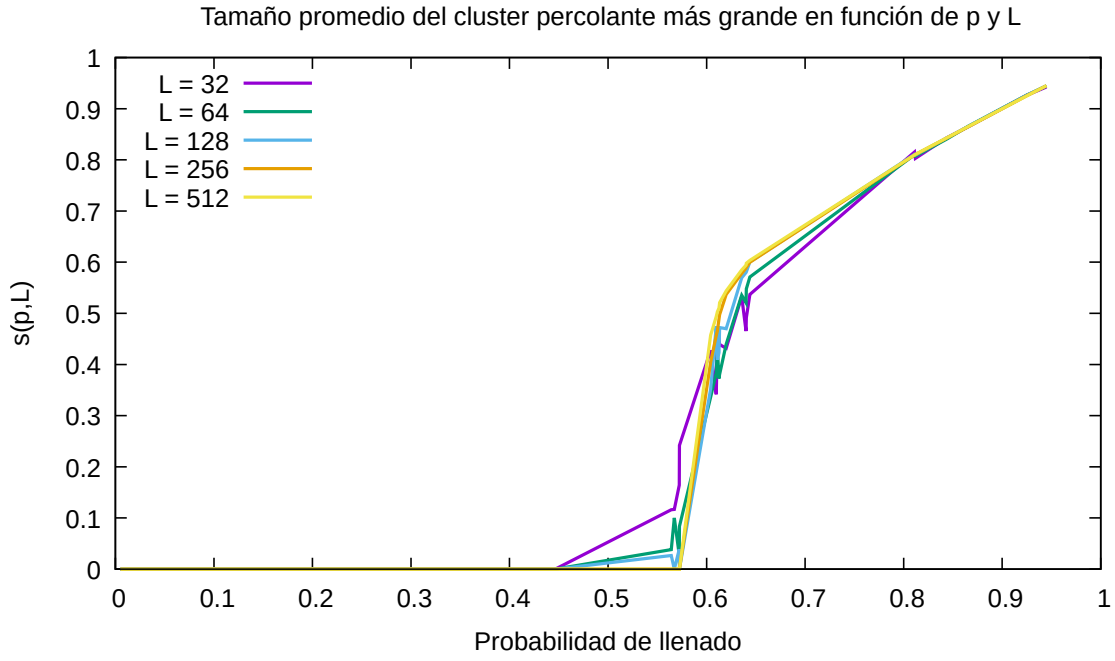

Fig. 3. Gráfico de $s(p, L)$ para valores de p en el intervalo $[0, 1]$

.

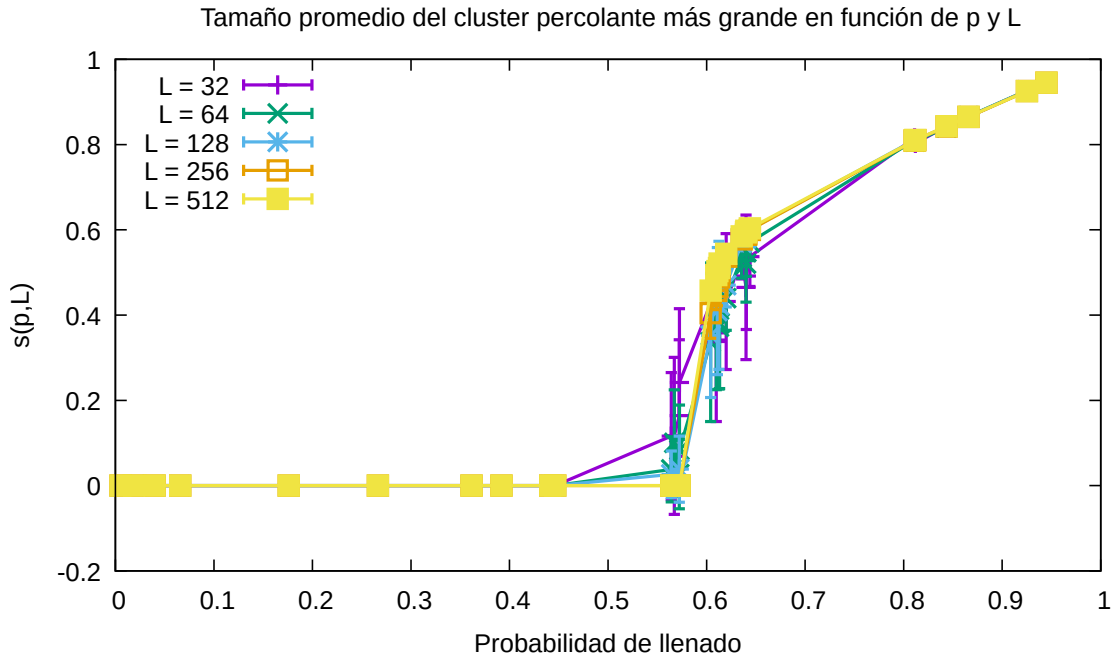

Fig. 4. Gráfico de $s(p,L)$ para valores de p en el intervalo $[0.00, 1.00]$ Con marcadores en los datos

.

Para cada probabilidad p se ejecutaron diez (10) repeticiones de simulación para obtener $s'(p,L)$. Este proceso se repitió cuatro (4) veces con el propósito de adquirir una mayor precisión; a partir de esos nuevos resultados se obtuvo su promedio, que corresponde a $s(p,L)$, y desviación estándar (σ). Cabe resaltar que a mayor fuese el numero de repeticiones tanto para calcular $s'(p,L)$ como para calcular $s(p,L)$, los datos nos proporcionarían estadísticas cada vez mas confiables, no obstante, demandarían una capacidad de computo mas elevada; con estos números se logra un equilibrio.

Como se puede observar en la **figura 3.** sin importar el tamaño de la matriz, todas presentan el mismo comportamiento, en el cual para probabilidades de llenado menores al $0,4$ el tamaño promedio del cluster percolante es nulo, debido a que no existe ningún cluster percolante para dichas probabilidades, tal como se evidencia en la anterior sección, en el intervalo desde 0,45 hasta 0,58 podemos observar un cambio gradual en el cual empiezan a aparecer clusters percolantes en las matrices, pero estos crusters no poseen un gran tamaño al compararse al tamaño de la matriz en la que se generaron, en donde el cluster percolante mas grande para estas probabilidades solo posee alrededor del 10% de las celdas en la matriz, luego, para el intervalo entre el 58% y el 65% podemos apreciar un crecimiento desmesurado de los clusters mas grandes en la matriz, una hipótesis que podría explicar este comportamiento es que en este rango de probabilidades los clusters que antes estaban aislados en varias partes de la matriz de repente se logran conectar los unos con los otros, lo que da como resultado un crecimiento acelerado en el tamaño de los crusters percolantes, por ultimo, para el intervalo entre el 65% y el 95% observamos un crecimiento lineal con una pendiente de la recta ligeramente menor al 100%, esto nos da a entender que el cluster percolante que se esta generando con dichas probabilidades de llenado posee en su interior la mayor parte de las celdas activas dentro de la matriz.

La **figura 4.** corresponde a un resumen gráfico del análisis de los resultados que se presentara en seguida. Al ser una simulación, los resultados obtenidos son aproximaciones de los datos teóricos.

Resultados para todas las matrices

La matriz de $L = 4$ es única entre las demás ya que es la única en la que observamos que se da una matriz percolante para una probabilidad de llenado menor al 35% , este hecho es atribuible a que al ser una matriz de tamaño tan bajo fue posible general un camino con una cantidad baja de celdas activas.

n	p_i	$s(p,L)$	σ (std)
4	0.00515915005919126	0	0
4	0.01902474123894857	0	0
4	0.03978036226448075	0	0
4	0.06570953877525139	0	0
4	0.1758612351493777	0	0
4	0.2664737832701232	0.06875000000000001	0.14375
4	0.3614963416156809	0	0
4	0.3917661954564606	0.0375	0.1125
4	0.4413485334192322	0.09375	0.2000976324197765
4	0.4463492007614787	0.11875	0.2475662790042296
4	0.5641464172912782	0.475	0.2769814975770042
4	0.5672695325116448	0.3	0.2888555002072836
4	0.572356491861326	0.43125	0.3003253444183491
4	0.5724505926942517	0.31875	0.3416984233209161
4	0.6041319372559434	0.41875	0.2905409824792365
4	0.6097556205861673	0.43125	0.3192398197280534
4	0.6109035598280564	0.53125	0.3000650971039451
4	0.6125253431985588	0.3875	0.2692582403567252
4	0.6132963099618162	0.4125	0.2825663638864329
4	0.6198420239381081	0.5875	0.2455860338048563
4	0.635562094509897	0.50625	0.1904968831766021
4	0.6400621454922257	0.55	0.2368411915187052
4	0.6403178526380329	0.41875	0.2592808564086442
4	0.6439142632400989	0.6125	0.2401171589037319
4	0.8110621047570959	0.825	0.06123724356957944
4	0.8119477502995482	0.70625	0.13125
4	0.8435281188537787	0.83125	0.1154136582038712
4	0.8658827550634677	0.80625	0.284289135388604
4	0.9251531591658583	0.86875	0.09035520184250602
4	0.9450038449806873	0.925	0.07288689868556625

Table 6. Resultados para $L = 4$

n	p_i	$s(p,L)$	σ (std)
32	0.00515915005919126	0	0
32	0.01902474123894857	0	0
32	0.03978036226448075	0	0
32	0.06570953877525139	0	0
32	0.1758612351493777	0	0
32	0.2664737832701232	0	0
32	0.3614963416156809	0	0
32	0.3917661954564606	0	0
32	0.4413485334192322	0	0
32	0.4463492007614787	0	0
32	0.5641464172912782	0.11611328125	0.148966887257355
32	0.5672695325116448	0.116796875	0.184130179586802
32	0.572356491861326	0.1642578125	0.1780141385067512
32	0.5724505926942517	0.2419921875	0.1731540873085592
32	0.6041319372559434	0.42783203125	0.09246603058493312
32	0.6097556205861673	0.3416015625	0.1912486396691306
32	0.6109035598280564	0.4478515625	0.05704529471756687
32	0.6125253431985588	0.44931640625	0.09081973496120131
32	0.6132963099618162	0.43896484375	0.1008475152333232
32	0.6198420239381081	0.4318359375	0.1593765557905068
32	0.635562094509897	0.5330078125	0.04833303692606222
32	0.6400621454922257	0.46513671875	0.1694278524783767
32	0.6403178526380329	0.49169921875	0.1254388734203469
32	0.6439142632400989	0.53701171875	0.06998683533173204
32	0.8110621047570959	0.81552734375	0.01190004618659793
32	0.8119477502995482	0.8029296875	0.01191086022538228
32	0.8435281188537787	0.84306640625	0.01046381136424972
32	0.8658827550634677	0.865234375	0.01658432009012014
32	0.9251531591658583	0.9265625	0.00604838381531329
32	0.9450038449806873	0.9423828125	0.009150552168064437

Table 7. Resultados para $L = 32$

Lo que hace especial la matriz de tamaño $L = 32$ es que para el intervalo entre $pi = 56\%$ y $pi = 58\%$ se observa que el cluster percolante mas grande es significativamente mas grande que el que presentan el resto de tamaños, la sospecha al igual que en el caso de $L = 4$ es que el tamaño relativamente bajo de la matriz le permite conectar un mayor numero de celdas.

La matriz de $L = 64$ presenta un comportamiento intermedio entre las matrices de $L = 32$ y las matrices de un mayor tamaño.

Para las matrices de tamaño $L = 128, 256, 512$ podemos apreciar el mismo comportamiento, el cual fue descrito a detalle al comienzo de esta sección.

n	p_i	$s(p,L)$	σ (std)
64	0.00515915005919126	0	0
64	0.01902474123894857	0	0
64	0.03978036226448075	0	0
64	0.06570953877525139	0	0
64	0.1758612351493777	0	0
64	0.2664737832701232	0	0
64	0.3614963416156809	0	0
64	0.3917661954564606	0	0
64	0.4413485334192322	0	0
64	0.4463492007614787	0	0
64	0.5641464172912782	0.038232421875	0.07646609094608665
64	0.5672695325116448	0.1002197265625	0.1245804062417597
64	0.572356491861326	0.027099609375	0.081298828125
64	0.5724505926942517	0.0836669921875	0.10518699309963
64	0.6041319372559434	0.3364013671875	0.1860116842686391
64	0.6097556205861673	0.3835205078125	0.1585291933106376
64	0.6109035598280564	0.4303466796875	0.04571379838517686
64	0.6125253431985588	0.372607421875	0.1459700361492031
64	0.6132963099618162	0.38076171875	0.1531167865029517
64	0.6198420239381081	0.43974609375	0.07540450684816616
64	0.635562094509897	0.5354736328125	0.04861354829340135
64	0.6400621454922257	0.52080078125	0.09024005558248932
64	0.6403178526380329	0.5478515625	0.057067107333208
64	0.6439142632400989	0.5708984375	0.04314693611733603
64	0.8110621047570959	0.8099609375	0.00782789620619056
64	0.8119477502995482	0.8072021484375	0.006090856011649831
64	0.8435281188537787	0.840283203125	0.004396971978458096
64	0.8658827550634677	0.8661865234375	0.004916206607738648
64	0.9251531591658583	0.9266357421875	0.002960551133525849
64	0.9450038449806873	0.9442626953125	0.002295571093343067

Table 8. Resultados para $L = 64$

n	p_i	$s(p,L)$	σ (std)
128	0.00515915005919126	0	0
128	0.01902474123894857	0	0
128	0.03978036226448075	0	0
128	0.06570953877525139	0	0
128	0.1758612351493777	0	0
128	0.2664737832701232	0	0
128	0.3614963416156809	0	0
128	0.3917661954564606	0	0
128	0.4413485334192322	0	0
128	0.4463492007614787	0	0
128	0.5641464172912782	0.026531982421875	0.05476723750212475
128	0.5672695325116448	0	0
128	0.572356491861326	0.03870849609375	0.07749094587166469
128	0.5724505926942517	0	0
128	0.6041319372559434	0.3522216796875	0.1454714577337838
128	0.6097556205861673	0.474932861328125	0.03143628209923148
128	0.6109035598280564	0.409356689453125	0.1490586096827757
128	0.6125253431985588	0.42305908203125	0.1502080482867319
128	0.6132963099618162	0.47205810546875	0.04997206443973522
128	0.6198420239381081	0.4700927734375	0.05096157043522467
128	0.635562094509897	0.569580078125	0.009509387034027178
128	0.6400621454922257	0.57991943359375	0.01911492471277496
128	0.6403178526380329	0.581011962890625	0.01555462845159308
128	0.6439142632400989	0.600555419921875	0.006003529786894868
128	0.8110621047570959	0.81083984375	0.004084415013555822
128	0.8119477502995482	0.809576416015625	0.003464243848748536
128	0.8435281188537787	0.8425048828125	0.003030344798421552
128	0.8658827550634677	0.86556396484375	0.002382123253366443
128	0.9251531591658583	0.926361083984375	0.00195069161462388
128	0.9450038449806873	0.945086669921875	0.0008505813475878061

Table 9. Resultados para $L = 128$

n	p_i	$s(p,L)$	σ (std)
256	0.00515915005919126	0	0
256	0.01902474123894857	0	0
256	0.03978036226448075	0	0
256	0.06570953877525139	0	0
256	0.1758612351493777	0	0
256	0.2664737832701232	0	0
256	0.3614963416156809	0	0
256	0.3917661954564606	0	0
256	0.4413485334192322	0	0
256	0.4463492007614787	0	0
256	0.5641464172912782	0	0
256	0.5672695325116448	0	0
256	0.572356491861326	0	0
256	0.5724505926942517	0	0
256	0.6041319372559434	0.4045318603515625	0.05890929736595225
256	0.6097556205861673	0.4567672729492188	0.04418845573374557
256	0.6109035598280564	0.4656509399414063	0.04331331999479548
256	0.6125253431985588	0.4887802124023438	0.03086243977146106
256	0.6132963099618162	0.497430419921875	0.02422756219330999
256	0.6198420239381081	0.5370620727539063	0.007911782774473596
256	0.635562094509897	0.5777816772460938	0.008899961942361312
256	0.6400621454922257	0.588433837890625	0.006228319951521895
256	0.6403178526380329	0.590045166015625	0.007364969692365334
256	0.6439142632400989	0.59970703125	0.00477961784129965
256	0.8110621047570959	0.8094436645507812	0.001323528223688747
256	0.8119477502995482	0.81044921875	0.001850346975071405
256	0.8435281188537787	0.8423171997070312	0.001277543424171102
256	0.8658827550634677	0.8648391723632812	0.001254462717296016
256	0.9251531591658583	0.9255905151367188	0.001259349377196368
256	0.9450038449806873	0.9450225830078125	0.0008265138283661402

Table 10. Resultados para $L = 256$

n	p_i	$s(p,L)$	σ (std)
512	0.00515915005919126	0	0
512	0.01902474123894857	0	0
512	0.03978036226448075	0	0
512	0.06570953877525139	0	0
512	0.1758612351493777	0	0
512	0.2664737832701232	0	0
512	0.3614963416156809	0	0
512	0.3917661954564606	0	0
512	0.4413485334192322	0	0
512	0.4463492007614787	0	0
512	0.5641464172912782	0	0
512	0.5672695325116448	0	0
512	0.572356491861326	0	0
512	0.5724505926942517	0	0
512	0.6041319372559434	0.4575901031494141	0.01902967443244453
512	0.6097556205861673	0.4937828063964844	0.01299988704956715
512	0.6109035598280564	0.5032302856445312	0.01249339326777827
512	0.6125253431985588	0.5104511260986329	0.02087923586122324
512	0.6132963099618162	0.5210765838623047	0.004566370913563197
512	0.6198420239381081	0.5437637329101562	0.005749211536354477
512	0.635562094509897	0.5846958160400391	0.003692289703511073
512	0.6400621454922257	0.5942970275878906	0.002973620601841954
512	0.6403178526380329	0.5977935791015625	0.002198689052675443
512	0.6439142632400989	0.6035972595214844	0.002729509699906893
512	0.8110621047570959	0.8096721649169922	0.0007634696072146707
512	0.8119477502995482	0.8106674194335938	0.0006114097006059681
512	0.8435281188537787	0.8429557800292968	0.0005073713715773153
512	0.8658827550634677	0.8653705596923829	0.0003928362204951793
512	0.9251531591658583	0.9251663208007812	0.0005835484437323122
512	0.9450038449806873	0.9450424194335938	0.0006572133335012123

Table 11. Resultados para $L = 512$

Optimización

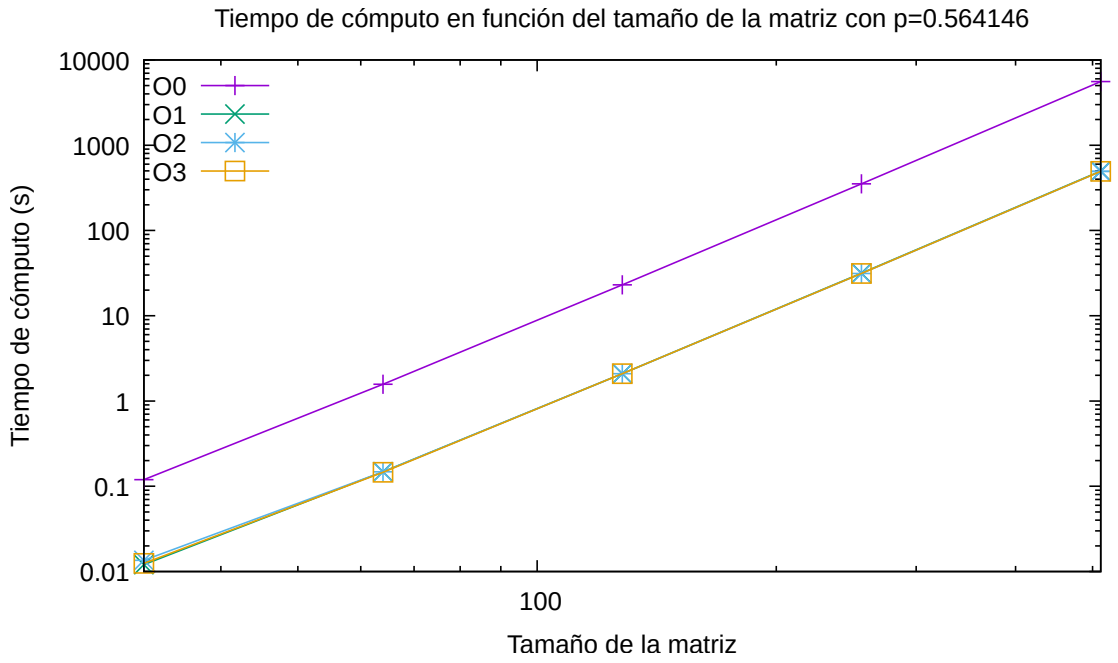

Fig. 5. Gráfico de tiempos de ejecución para las cuatro banderas de optimización en función del tamaño de la matriz.

.

En el siguiente gráfico se observa resumidamente, bajo una escala doble logarítmica, los tiempos de ejecución de la simulación en función del tamaño de la malla. Como dato importante para destacar, en todas las banderas se observa un comportamiento logarítmico lineal, algo no muy variable. Por otra parte, -O1, -O2 y -O3 ocurren con tiempos muy similares, mientras que -O0 ocurre con un tiempo bastante distante. Las rectas logarítmicas parecen casi paralelas lo que evidencia el gran beneficio de emplear banderas de optimización, pues para valores como $L = 512$ mas de 9000 segundos de diferencia es bastante significativo.

Profiling

$\%tiempo(s)$	$Funciones\ implementadas$
0.0	$fill\_matrix$
0.94	$find\_nonclassified\_cluster$
0.26	$cluster\_size$
44.34	$classify\_cluster$
0.2	$attach\_neighbours$
0.0	$is\_percolating$
0.0	$is\_there\_a\_percolating$
0.05	$largest\_percolating'_cluster\_size$
5.18	$number\_of\_last\_cluster$
0.0	$compute\_mean$
0.0	$standard\_deviation$
0.0	$compute\_mean\_and\_standard\_deviation\_for\_percolating\_probability$

Table 12. Porcentaje de tiempo usado por cada funcion

En esta tabla podemos observar los porcentajes de tiempo de ejecución de cada una de las funciones implementadas por los autores que son fundamentales para la simulación. El flat profile se obtuvo de compilar con la bandera -pg **calculating_probabilities.cpp** con los parámetros $p = 0.611$ y $L = 128$. Vemos que en total, las funciones ocupan un 50.97% de la ejecución del programa. El 49.03% restante corresponde a funciones ya internas de $C++$, funciones ya implementadas o no tan relevantes para la simulación. La función mas costosa computacionalmente es $classify\_clusters$, esto tiene sentido, pues es la que requiere de un barrido entero de la matriz junto con barridos internos para clasificar los clusters. Hubo funciones las cuales que de por si el tiempo de ejecución de maquina fue nulo o casi nulo, pues simplemente conllevan operaciones aritméticas como $compute\_mean, standard\_deviation$,etc.

Conclusiones

Como conclusiones obtenemos lo siguiente:

- El proceso de percolacion se encuentra fuertemente ligado al coeficiente del medio donde este ocurriendo el proceso, por ejemplo coeficiente de filtrado del suelo o también de un mismo colador o malla de percolacion.
- A medida que aumente el tamaño del agente de percolacion, el traspaso de ese medio poroso depende mas del coeficiente de percolacion que del mismo tamaño. Es decir, como ya se vio puede que el tamaño de la matriz aumente, sin embargo si se quiere lograr un buen filtrado es necesario de un coeficente intermedio, como por el ejemplo uno en el intervalo estudiado $[0.55, 0, 65]$, intermedio ya que para coeficientes muy altos ya pasaría a ser un tipo de obstrucción.
- El proceso de simulación practico concuerda bastante bien con un proceso de experimentación practica.
- La implementación del algoritmo empieza a ser costosa computacionalmente a partir de mallas bidimensionales de tamaños mayores a 1000.
- El comportamiento del cluster percolante mas grande dentro de la matriz es el mismo a partir de $L = 128$ en donde la matriz adquiere un tamaño lo suficientemente grande para que pequeñas variaciones en las celdas activas no tengan un gran impacto en el tamaño del cluster mas grande.
- Para probabilidades altas el cluster percolante mas grande esta conformado por casi la totalidad de las celdas activas en la matriz.
- La probabilidad critica es dependiente del tamaño de la malla bidimensional, sin embargo, cuando $L \longrightarrow \infty$, esta se estabiliza cercana al valor 0.59271 a tal punto de llegar a ser constante.