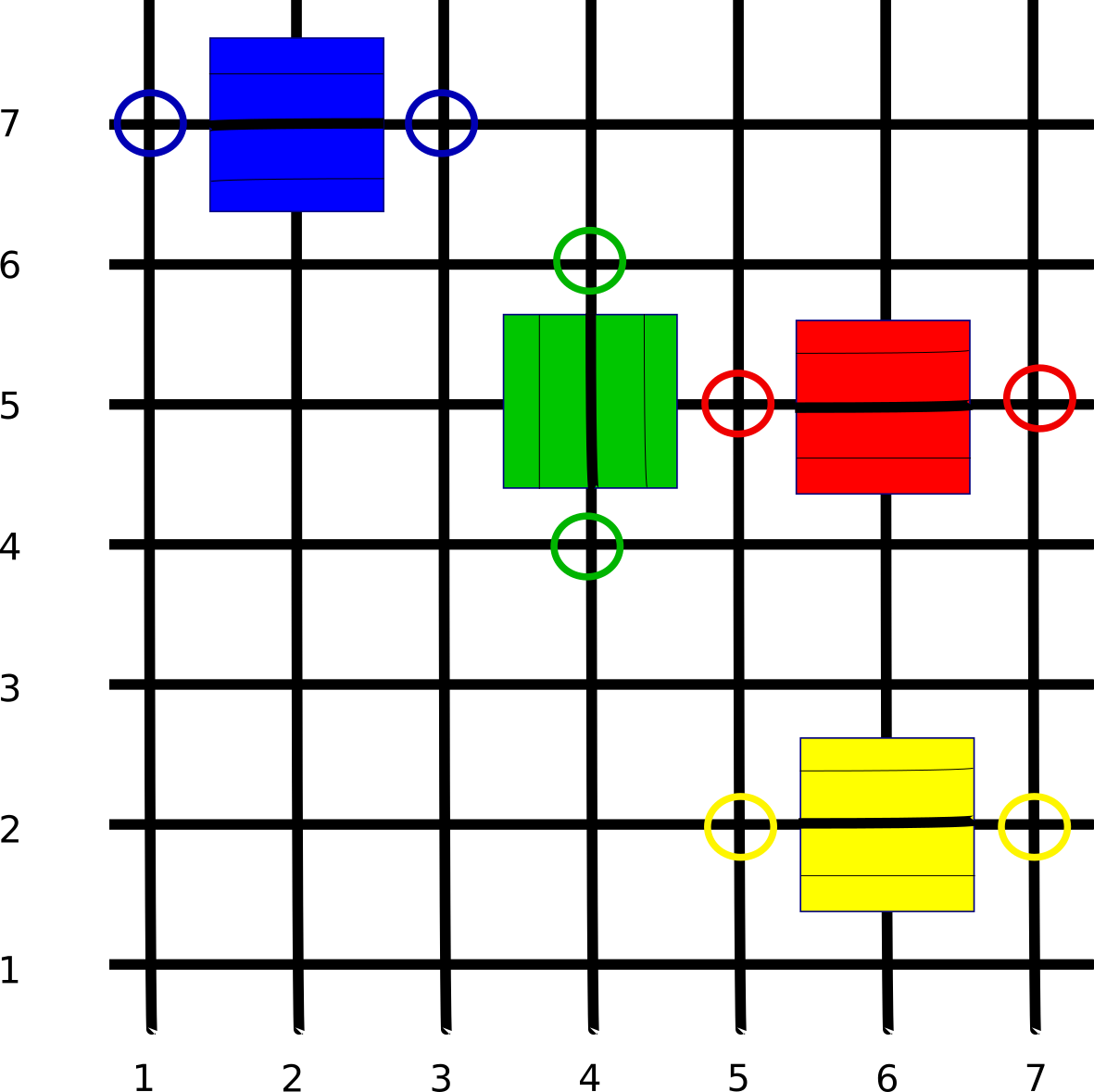
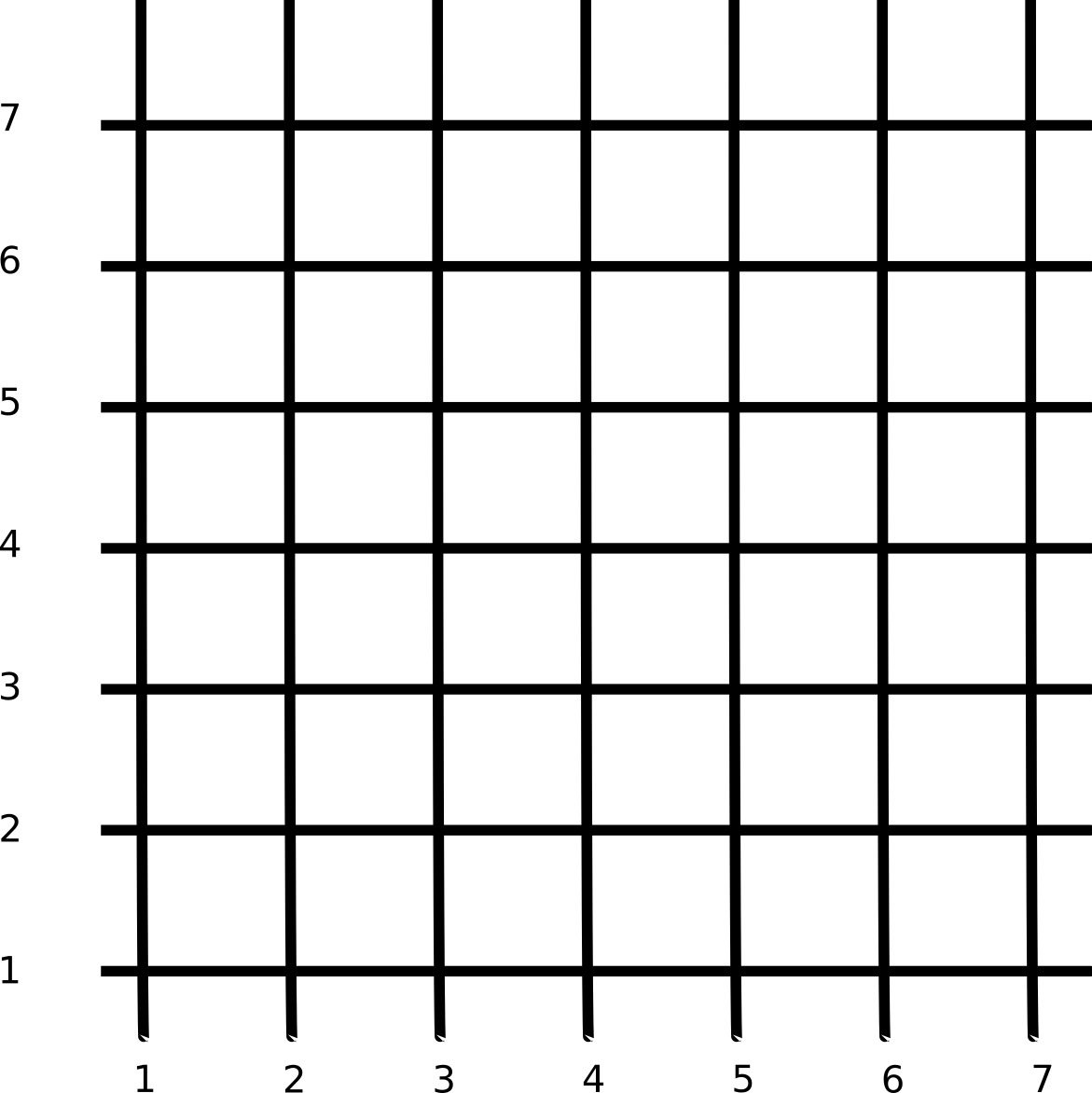
Planeación de Ruta usando Ant Colony Optimization Meta-Heuristic

La planeación de ruta fue realizada basada en un algoritmo llamado Ant Colony Optimizaton[[1]](#footnote-1), el cual da solución al problema del viajero, que consiste en visitar un número total de locaciones recorriendo la menor distancia posible; no obstabte, el problema que se le trata de dar solución es distinto, ya que, más allá de intentar recorrer un número de locaciones de manera aleatoria, se necesita recorrer los tuneles en orden, partiendo de una esquina para llegar a la esquina contraria, cruzando por los tuneles rojo, azul, verde y amarillo. Debido a esto, un algoritmo derivado de Ant Colony Optimization es usado, conocido como Ant Colony Optimization Meta-Heuristic[[2]](#footnote-2), el cual consiste en hacer que agentes virtuales conocidos como “Hormigas” realicen la busqueda de una ruta entre dos puntos por distintas generaciones hasta que una ruta lo suficientemente buena, con base en criterios dados, sea encontrada o que un número *n* de épocas hayan sido usadas para ejecutar dicha búsqueda.

Supongase un entorno como en la Figura 1a , donde un plano coordenado derecho es asignado. En la Figura 1b se observa el área con los cuatro túneles colocados en distintas posiciones, las posiciones de las entradas libres al tunel se muestran encerradas en círculos con el color perteneciente al tunel.



a) b)

Figura 1. a) Entorno en donde el móvil se desplazara desde las coordenadas (1,1) a (7,7). b) Entorno con los tuneles colocados y encerrados con círculos las entradas de cada tunel con su respectivo color.

**Desarrollo**

Un objeto de la clasee **BMWRoute** debe ser creado, dando como entrada una lista conteniendo las posiciones de cada una de las entradas de cada tunel de manera ordenada. Un ejemplo de lista con base en la Figura 1b sería como se muestra a continuación

Posiciones = [ ( 5,5) , (7,5), ( 1,7) , (3,7), ( 4,4) , (4,6), ( 5,7) , (7,2) ]

Una vez que las posiciones han sido dadas, el método BMW.Start() es llamado, comenzando la búsqueda de ruta. Primero, dadas las coordenadas de las entradas a los tuneles, es posible encontrar las coordenadas en las cuales cada tunel esta colocado, siguiendo la fórmula

Coordenadas de Tunel = [ (Punto1.x + Punto2.x)/2 , (Punto1.y + Punto2.y)/2 ]

Estas coordenadas son tratadas como obstáculos, ya que dichas posiciones no pueden ser alcanzadas a menos que sea el turno de dicho tunel a ser recorrido.

Después de que se calculan las coordenadas de cada tunel, se inicializa la lista de nombre **orderedPoints** con la primera posición siendo la coordenada de partida (1,1), para después ser ingresado la coordenada de tunel siguiente que se encuentra más cercana al inicio, en este ejemplo sería la posición (5,5), y se agrega a la lista **orderedPoints** seguido por la coordenada de la otra entrada del tunel, que es (7,5); después se obtiene la coordenada del siguiente tunel más cercana a la última coordenada agregada a la lista **orderedPoints,** siendo la coordenada (7,5). Este proceso se repite hasta que todas las posiciones han sido ordenadas y al final el punto final (7,7) es agregado a la lista. El resultado una lista del ejemplo actual es:

**orederPoints**  = ( (1,1) , (5,5) , (7,5) , (3,7) , (1,7) , (4,6) , (4,4) , (5,2) , (7,2) , (7,7) )

Después, una matriz que representa el área mostrada en la Figura 1b es creada, si bien el plano coordenado es de 7,7; la matriz que se crea es de 9,9 debido a que se agregan laas fronteras que son prohibidas. A continuación se muestra una matriz representativa del área

**Area** = [0 0 0 0 0 0 0 0 0

0 1 1 1 1 1 1 1 0

0 1 1 1 1 1 1 1 0

0 1 1 1 1 1 1 1 0

0 1 1 1 1 1 1 1 0

0 1 1 1 1 1 1 1 0

0 1 1 1 1 1 1 1 0

0 1 1 1 1 1 1 1 0

0 0 0 0 0 0 0 0 0]

Después las posiciones de los túneles son agregadas. Los ceros representan las posiciones que no deben ser alcanzadas por el móvil.

**Area** = [0 0 0 0 0 0 0 0 0

0 1 0 1 1 1 1 1 0

0 1 1 1 1 1 1 1 0

0 1 1 1 0 1 0 1 0

0 1 1 1 1 1 1 1 0

0 1 1 1 1 1 1 1 0

0 1 1 1 1 1 0 1 0

0 1 1 1 1 1 1 1 0

0 0 0 0 0 0 0 0 0]

Una vez que el área es terminada, un objeto tipo **Colony** es inicializado dando como entradas el número de objetos **Ant** a ser creados, el número de épocas a ejecutar el algoritmo, la posición inicial, la posición final, el Área a explorar, constantes de evaporación de feromona, constante para refuerzo de feromona y valor de feromona inicial).

Col = Colony(10, 10, orderedPoints[i], orderedPoints[i+1], Area, 0.001, 1,0.1)

Los puntos que son entregados al objeto **Colony** como inicio y objetivo son en parejas, el primer punto de partida es la esquina en la cual el móvil es colocado al inicio y el primer punto objetivo es la entrada del tunel número uno más cercana; la segunda posición de partida es la entrada del tunel número uno que no ha sido usada y la segunda posición objetivo es la entrada del tunel número dos más cercana al tunel uno y el proceso sigue sucesivamente hasta que la última pareja es conformada por una entrada del tunel número cuatro y las coordenadas 7,7 que son la posición final del recorrido. En el ejemplo, estás parejas estarían dadas por las siguientes parejas

[1 1] [5 5]

[7 5] [3 7]

[1 7] [4 6]

[4 4] [5 2]

[7 2] [7 7]

Dentro de la clase **Colony**  *n* objetos **Ant** son creados e inicializados recibiendo como entrada la posición de donde comenzará la búsqueda y el punto donde la búsqueda termina. Después de esto, la búsqueda de la posición objetivo comienza. Este proceso es ejemplificado con el área mostrada en la Figura 1b.

La primera ruta a ser encontrada es desde la posición (1,1) a la posición (5,5). Está búsqueda sera realizada por cada uno de los objetos tipo **Ant**. Supóngase que el primero objeto tipo **Ant**  comienza su recorrido, éste se encuentra en la posición (1,1); para tomar su siguiente posición un proceso de decisión es realizado de manera probabilística. Cada uno de los objetos tipo **Ant** solo tiene permitido moverse trazando cruces, como se muestra en la Figura 2, los puntos verdes son las posiciones a las cual el objeto **Ant** puede tener acceso.

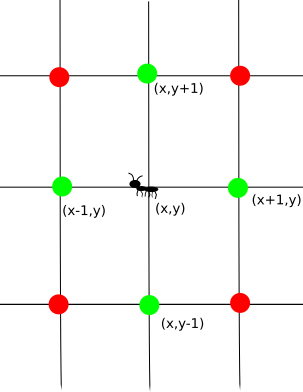
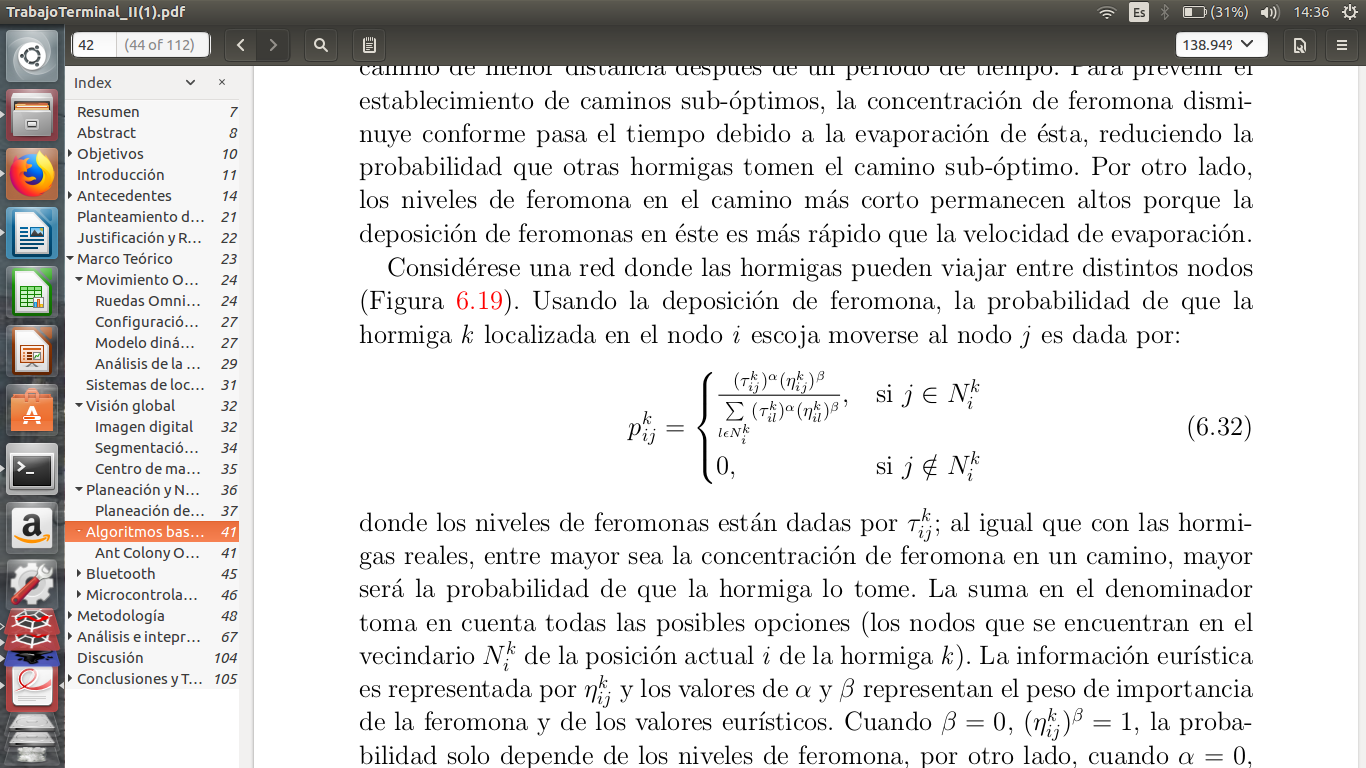


Figura 2. Posiciones a las cuales el objeto tipo **Ant** puede tener acceso. Cada uno de estos puntos en color verde tienen sus coordenadas con base en la posición del objeto **Ant**.

Para determinar la siguiente posición, se calcula cuán probable es que alguno de los puntos en el vecindario del objeto **Ant** sea elegido utilizando la siguiente ecuación



donde *pkij* es la probabilidad de que el nodo *j* sea elegido por el objeto k tipo **Ant** encontrado en la posición i, τij es el valor de feromona encontrado en el nodo, ηij es el inverso de la distancia entre el nodo j y la posición objetivo, α y β son constantes de peso, es decir, el factor en el que la feromona o la distancia intervendrá en la probabilidad. Esta probabilidad solo es calculada para los nodos encontrados en el vecindario del nodo i, que es la posición del objeto **Ant**. Una vez que las probabilidades se tienen, se elige alguna de las posibles siguientes posiciones tomando en cuenta la probabilidad de cada posición.

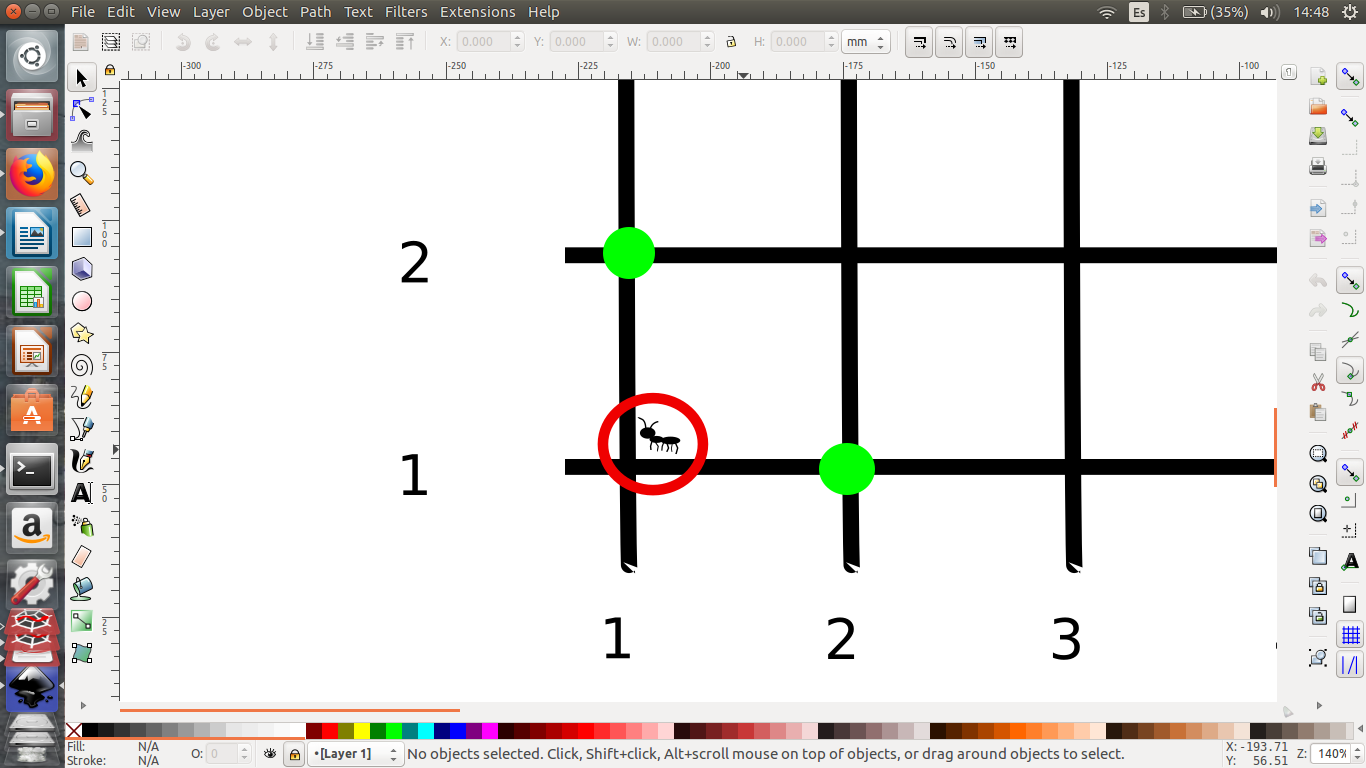


Figura 3. Posibles posiciones partiendo de la posición inicial.

Tomando como ejemplo la Figura 3, solo hay dos posibles posiciones a ser tomadas, los nodos (2,1) y (1,2). Debido a que es la primera iteración, las feromonas conservan el valor con el que se inicializaron, que es 0.1; el valor de α y β son constantes e iguales a 0.5. Tomando lo anterior en cuenta y que la posición destino en esta primera parte es (5,5), las probabilidades de cada uno de los puntos es como se muestra a continuación:

p1(1,1)(2,1) = [0.10.5 \* (1/dist((2,1), (5,5)))0.5 ] / {[0.10.5 \* (1/dist((2,1), (5,5)))0.5 ] + [0.10.5 \* (1/dist((1,2), (5,5)))0.5 ]} = 0.0548478 / (0.0548478 +0.0548478 ) = 0.5

p1(1,1)(2,1) = [0.10.5 \* (1/dist((1,2), (5,5)))0.5 ] / {[0.10.5 \* (1/dist((2,1), (5,5)))0.5 ] + [0.10.5 \* (1/dist((1,2), (5,5)))0.5 ]} = 0.0548478 / (0.0548478 +0.0548478 ) = 0.5

Debido a que ambas posiciones son equiprobables, la decisión será hecha de manera aleatoria. Supóngase que la posición (2,1) es elegida, el proceso de cálculo de probabilidades se repite de nuevo con las posiciones a las que se tiene acceso (Figura 4, puntos verdes) y que no hayan sido visitados (Figura 4, punto amarillo).

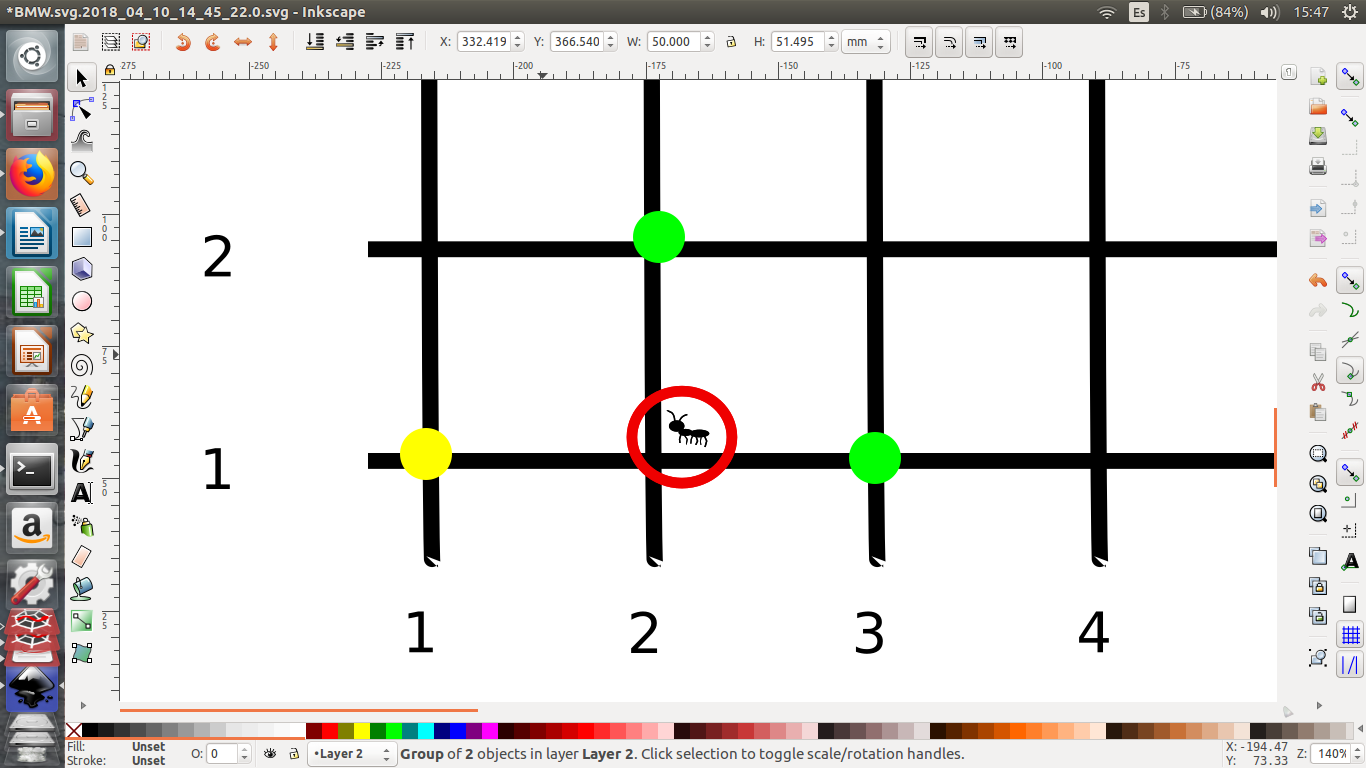
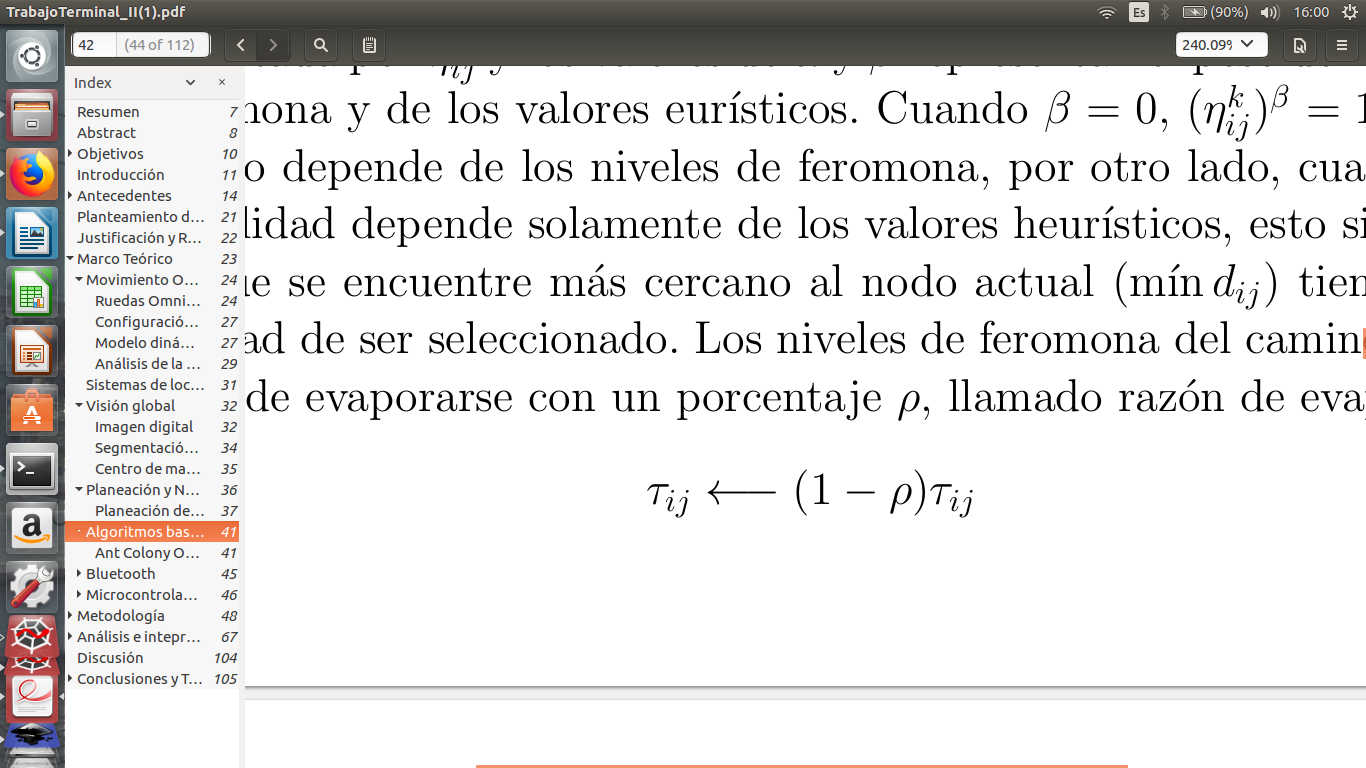


Figura 4. Desde la posición actual (2,1), las posibles siguientes posiciones son (3,1) y

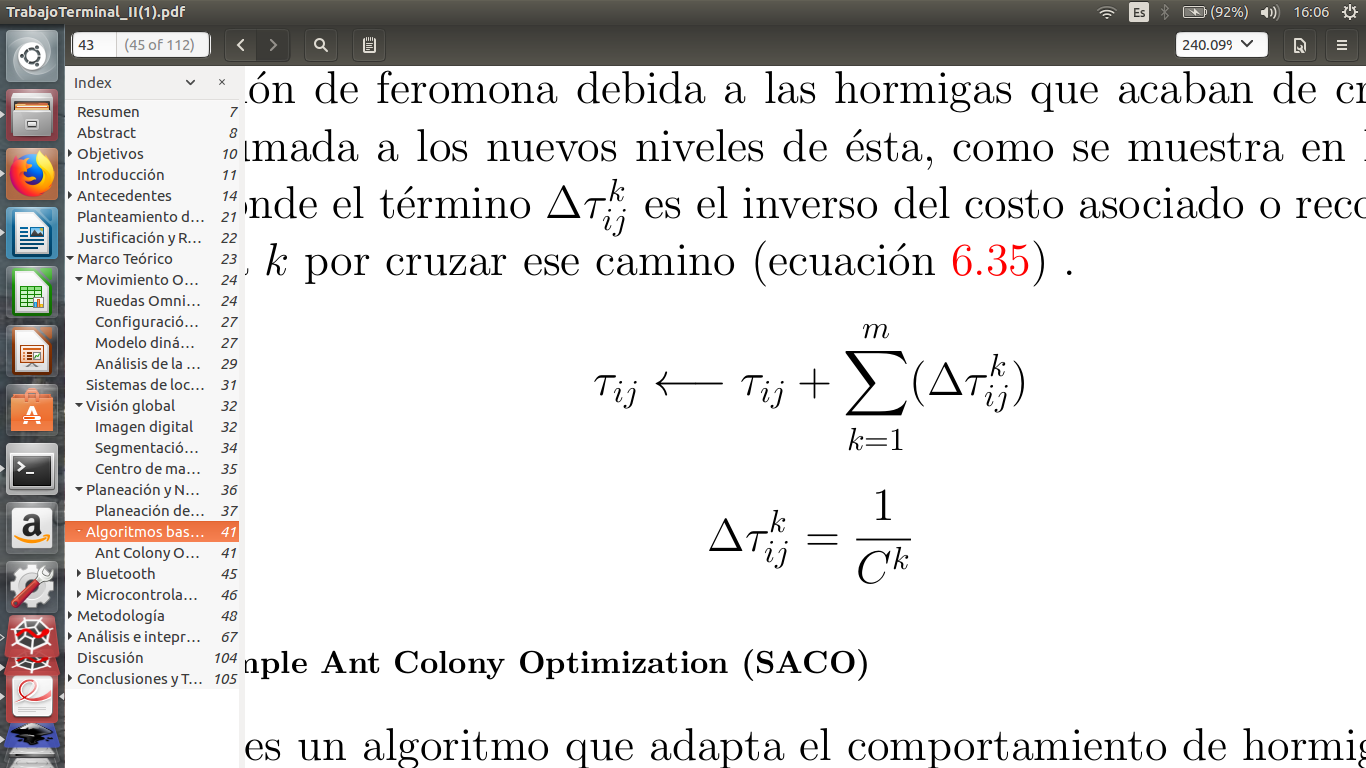
(2,2). La posición (1,1) no es una opción ya que ya fue visitada y el algoritmo prohibe

repetir dicha posición.

Este proceso se repite sucesivamente hasta que el objto de la clase **Ant** llega a la posición deseada; en este momento el objeto **Ant** en turno termina su trabajo y el siguiente comienza al igual que el anterior, buscando una serie de posiciones desde (1,1) hasta (5,5). Una vez que los *n* objetos **Ant** han encontrado la posición deseada, la ruta de menor número de posiciones es elegida para que dos procesos que afectan las feromonas depositadas en el área toman lugar, evaporación y refuerzo. El primero conciste en hacer que la feromona depositada en el área sea reducida un factor ρ en función de la ecuación siguiente:



Una vez que la evaporación ha sido realizada, el refuerzo entra en acción. El refuerzo de feromona conciste en incrementar la feromona en las posiciones que forman la ruta que es formada por el menor número de posiciones. El refuerzo está dado por la siguiente ecuación



Donde Δτij es el inverso del número de posiciones que componen la ruta encontrada.

Cuando se llega a este punto se dice que una iteración o época a terminado. Este proceso se repite tantas veces como le haya sido indicado al algoritmo.

Una vez que el número de épocas ha sido agotado, la ruta de menor número de posiciones es almacenada y un nuevo punto de inicio y posición de interes son ingresados al algoritmo, siguiendo el ejemplo mostrado en la Figura 1b, las posiciones de inicio y objetivo serían (7,5) y (3,7) respectivamente.

Una vez que todas las posiciones han sido ingresadas al algoritmo y éste ha encontrado una ruta, todas las rutas de los anteriores recorridos son unidas y regresadas en un vector.

**Resultados:**

La lista de puntos que es entregada con base en el ejemplo de la Figura 1b es la siguiente

**Ruta** = [[1, 1], [2, 1], [3, 1], [2, 1], [3, 1], [3, 2], [4, 2], [5, 2], [5, 3], [5, 4], [5, 5], (6.0,5.0), [7, 5], [7,6], [6, 6], [6, 7], [5, 7], [4, 7], [3, 7], (2.0, 7.0), [1, 7], [1, 6], [2, 6], [3, 6], [4, 6], (4.0, 5.0), [4, 4], [4, 3], [5,3], [5, 2], (6.0, 2.0), [7, 2], [7, 3], [7, 4], [7, 5], [7, 6], [7, 7]]

En la Figura 5 se muestra la ruta que el móvil seguiría si la Figura 1b fuera el entorno con el cual se está trabajando.

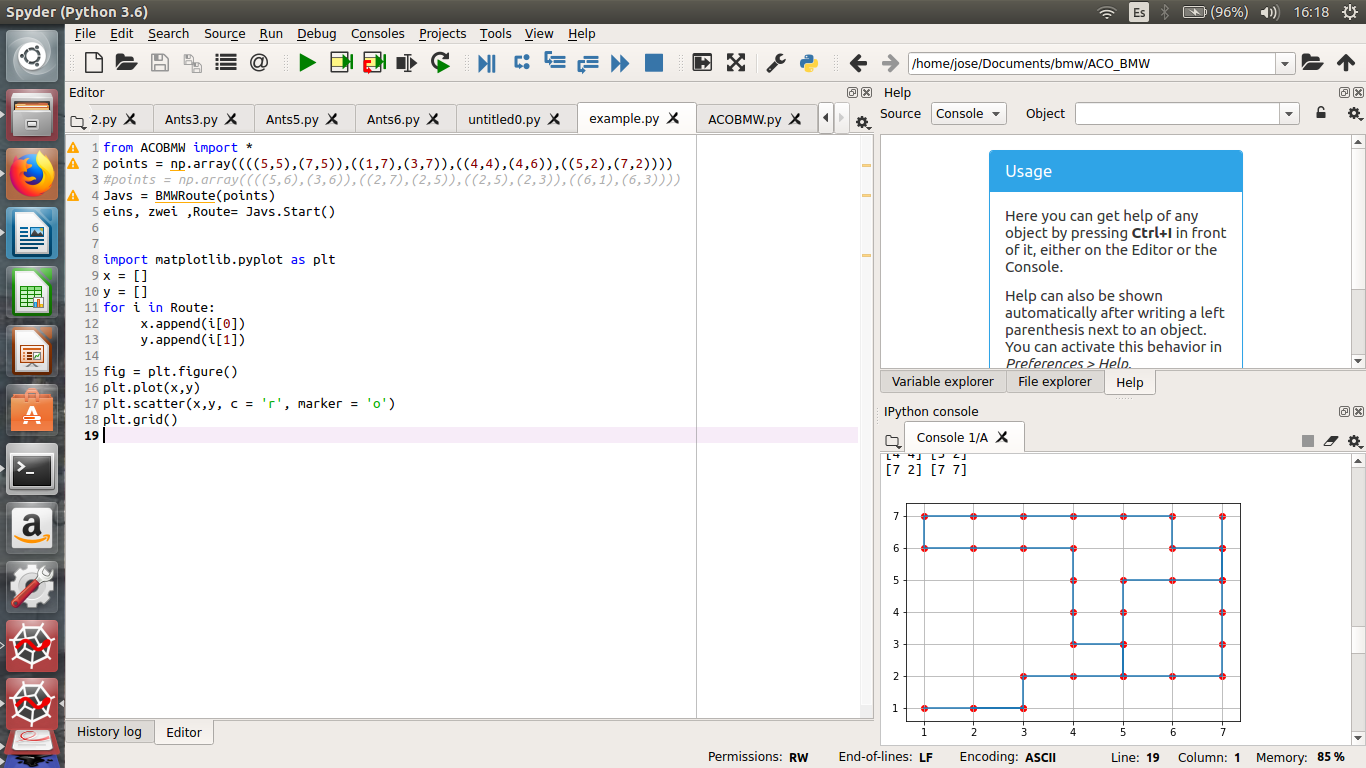


Figura 5. Ruta final entregada por el algoritmo

1. M. Dorigo, V. Maniezzo, Colorni A., Ant System: Optimization by a Colony of Cooperating Agents, Cybernetics, Vol. 26, No. 1, 1996. [↑](#footnote-ref-1)
2. M.A. Porta Garcia, O. Montiel , O. Castillo, R. Sepulveda ,P. Melin , Path planning for autonomous mobile robot navigation with antcolony optimization and fuzzy function evaluation, Analysis and Design of Intelligent Systems using Soft Computing Techniques SP, Volume 41, pp 790-798, 2009. [↑](#footnote-ref-2)