Luis Alberto Casillas Santillán
Inteligencia Artificial
Noviembre 2013

Indicaciones para la **Etapa IV** en los diferentes rubros posibles.

Muy importante, la entrega de la etapa final implica también la entrega de los reportes para cada etapa **los cuales deben colocarse en el repositorio específico del curso en Moodle**. El reporte por etapa debe incluir una pequeña introducción descriptiva de la etapa, una propuesta de solución (incluyendo algunas ilustraciones y diagramas en el caso de circuitos), porciones del sistema como partes importante de código fuente (ya sea en lenguaje de alto nivel para la gente de SW o rutinas críticas de ensamblador o alguna de las clases de HDL utilizado [o bien, C empotrado]), pantallas para la gente de SW mostrando algún momento de la funcionalidad y fotografías del avance para la gente que fabricó algún dispositivo. Esto para cada etapa, finalmente incluir una conclusión general que hable del aprovechamiento de la experiencia haciendo estos sistemas. Todos deben entregar este documento que agrupa las etapas, es una cuestión crítica para conseguir la nota.

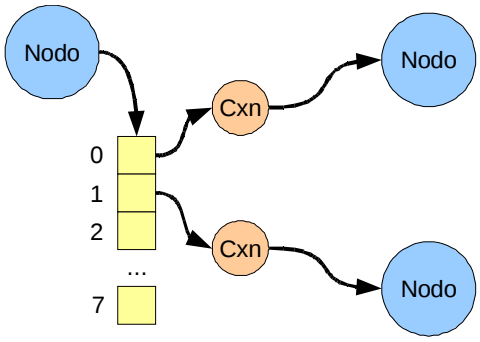
La fecha de entrega es:
4-Diciembre-2013

Etapa IV. Software. Por medio de aprendizaje de máquina, “Kibus” descubre un “buen” camino para volver a casa, sin tener al principio la mínima noción de su ubicación…

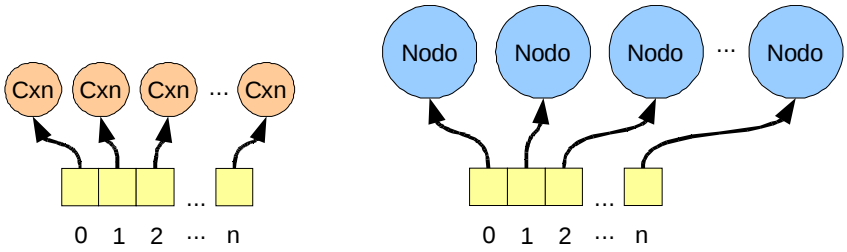
1. En esta etapa el personaje “Kibus” desconoce la ubicación de la casa. Sólo sabe que ha llegado a ella cuando se mueve a la posición donde está la casa, “se topa con ella”.
2. En una posición se coloca la casa y en otra diferente el personaje. Ambas elegidas por el usuario. Hay, como siempre, obstáculos aleatorios o colocados manualmente. El personaje no tiene la menor idea de donde podría estar la casa y se mueve al azar hasta llegar a ella, sólo que en este caso se lleva un registro de las acciones realizadas para llegar a la meta.
3. Requerirá ir de la posición inicial a la casa varias veces para descubrir / confeccionar el mejor camino, podrían ser miles de recorridos si el bosque es grande. Esto demanda que el sistema tenga dos modos de trabajo, uno con *simulación gráfica*, mostrando en la GUI todas las acciones y otra modalidad en la que se hacen los recorridos en *modo ciego*. Se supone que cuando corre en modo ciego ejecuta el mismo algoritmo que cuando muestra simulación gráfica, pero este modo de trabajo permite ejecutar más rápido los recorridos. Este comportamiento de recorrer una y otra vez el bosque mientras lo aprende será llamado de ahora en adelante “entrenamiento”, en independencia de mostrar simulación o hacerlo oculto.
4. Cada recorrido del entrenamiento consiste de moverse al azar hacia posiciones disponibles hasta llegar a la casa. Este movimiento aleatorio luce como un aparente caos, pero en realidad es bueno. Al deambular por el bosque, el personaje “Kibus” conoce su configuración. La desventaja que puede esto tener (además del tiempo que consumen tantos recorridos aleatorios) es que la máquina de números aleatorios (random) no sea muy eficiente y/o tendenciosa. Podría pensarse en fabricar una máquina que sea más diversa, es sencillo si

consideramos que sólo debe elegir al azar entre ocho alternativas; pero debe cuidarse que cada una de ellas tenga una probabilidad de 0.125 de aparecer.

5. Durante el entrenamiento, el personaje memoriza las conexiones que utilizó para ir en el recorrido actual de la celda de inicio a la celda meta (la casa). Las ubicaciones de estas celdas no cambian durante todo el conjunto de recorridos que constituyen el entrenamiento. Al concluir un recorrido se cuentan la cantidad de conexiones utilizadas y ello se utilizará para lograr implantar un aprendizaje competitivo que se explica más adelante.
6. La estructura de datos recomendada luce como se muestra en la siguiente gráfica:



7. Adicionalmente, existe dos estructuras lineales, una que recuerda los Nodos que se han visitado y otra que recuerda las conexiones que se han recorrido. Las estructuras lucen como muestra la imagen a continuación:



8. El vector de conexiones visitadas abona directamente para el aprendizaje. Las ***n*** conexiones visitada se comparan contra una media histórica. La media histórica se calcula dinámicamente a partir del máximo histórico y el mínimo histórico. El máximo y el mínimo van cambiando sobre la marcha: cuando se ha realizado un recorrido menor al mínimo ese se convierte en el nuevo mínimo, cuando se ha realizado un recorrido mayor al máximo se convierte en el nuevo máximo. Los valores iniciales para el máximo y el mínimo son iguales al valor del primer recorrido completado (Bootstrap).
9. Una vez que concluye un recorrido (incluso el primero), se calcula la media histórica (máximo histórico + mínimo histórico, dividido por dos) y se compara contra el tamaño del recorrido recién terminado (al concluir el primer recorrido, la media y el tamaño del último recorrido resultarán iguales, con base en el punto 8 de esta lista). Si el recorrido es mayor a la media, se aplica un castigo; si el recorrido es menor a la media, se aplica un premio. La magnitud del premio o el castigo resultan del valor absoluto de restar la media histórica del tamaño del último recorrido. Cuando se ha aplicado el premio o castigo que amerite, se procede a verificar si los límites deben ser ajustados y no antes.

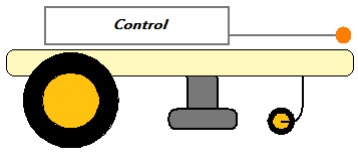
10. El vector de nodos visitados permite verificar que no se visiten durante cada recorrido de entrenamiento nodos ya visitados, a menos que no exista otra opción. Por ejemplo, en bloqueos por obstáculos.
11. Ambos vectores, el de conexiones y el de nodos, se limpian cada recorrido que se haga durante el entrenamiento. Lo cual no supone un problema, pues el aprendizaje logrado por ellos queda subyaciendo en los valores de conexión.
12. Cada objeto de conexión tiene un valor entero que representa su costo. Al principio es igual a MAXINT/2. Cuando participa en una trayectoria que se castigó, su costo aumenta… cuando participa en una trayectoria que se premió su costo se reduce. Este premio o castigo se tiene una magnitud equiparable a la diferencia absoluta entre el valor medio histórico y el valor actual del último recorrido. Como se explicó previamente.
13. El grafo que modela al bosque está al principio vacío, sólo consta de dos nodos y ninguna conexión. Los nodos y las conexiones entre éstos se crean conforme navega el personaje por el bosque. Esta característica permite que el grafo refleje en su topología una organización similar a la del bosque.
14. Los recurrentes recorridos usando esta política de entrenamiento propician que los caminos que conducen mejor a la casa, sean más frecuentemente premiados; mientras que aquellos que no conducen a la casa sean más frecuentemente castigados. Los premios y castigos provocan que las magnitudes de conexión entre nodos se modifiquen, reflejando así el entrenamiento recibido. Por ello, se necesitan varios recorridos.
15. Luego cuando el entrenamiento es terminado, los pesos forman una especie de embudo abstracto que indica el mejor camino para ir de cualquier parte del bosque a la casa. Puede entonces iniciar la etapa de explotación.
16. Para la explotación se trabajará una de las siguientes formas:
    a. Primero el mejor, algoritmo consistente en elegir simplemente la ruta con menor costo en el momento de salir de cada nodo. Este algoritmo tiene el riesgo de visitar nodos visitados e ingresar en ciclos infinitos. Para ello es preciso controlar los nodos visitados. Si de desea ir a un nodo ya visitado, se elige el nodo de menor costo siguiente. Si no existe ese nodo siguiente, se retrocede un nodo y se prueba otro nodo siguiente en el previo.
    b. Algoritmo de Dijkstra. Que permite en un grafo con pesos (como el que se ha construido durante el entrenamiento) encontrar el camino mínimo entre dos puntos.
17. Durante la explotación debe ser posible colocar al personaje en diferentes puntos del bosque y desde allí volver. Al volver preguntará si lo debe hacer usando “primero el mejor” o Dijkstra.

Etapa IV. Robot móvil “Tamborilero”. ¡Qué todo junto funcione!

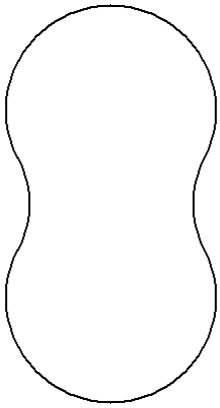
1. Actualmente el robot localiza superficies “ideales” y toca sobre ellas.
2. Hace falta la última porción, que genere sonido activamente como complemente a las percusiones. Una fuente de ritmos. Leer el documento Moodle sobre el tema.
3. Luego de un tiempo (cuando ha terminado de tocar con los percutores y los ritmos) el robot acciona un movimiento emergente para retirarlo de la superficie e inicia de nuevo la búsqueda de una superficie… y todo inicia de nuevo ☺

Etapa IV. Robot móvil de multi-comportamiento. Seguidor de líneas. Aunque al final, debe ser posible accionar cualquiera de las etapas de forma selectiva… DIP switch, posiblemente.

1. El robot móvil se mueve al frente mientras vira siguiendo lógicamente una línea pintada en el piso y a la vez va *sensando* con infrarrojo (IrDA) sigue detectando la presencia de algún obstáculo.



2. Cuando detecta un obstáculo, la única acción emergente en detenerse hasta que el obstáculo se retire.
3. La ruta se sigue por medio de sensores de luz que detectan la contrastante tonalidad en el piso. Cuando el robot no siente una línea frente a los sensores de su “panza”, hace un comportamiento emergente que eventualmente le llevará a una línea en el piso, como ir en reversa. Cuando la línea es finalmente detectada, inicia su rutina infinita de seguir línea y detenerse ante un obstáculo.
4. La forma de la línea se expresa a continuación:



Etapa IV. Robots para Recolección de Basura.

1. Cuando una determinada cantidad de elementos han sido recolectados, el robot debe liberarlos.
2. La idea es que el robot forme montículos de elementos que ha recolectado el robot y que previamente se encontraban dispersos.
3. Esta aproximación imita los mecanismos que usan las hormigas para proteger las larvas cuando el nido está comprometido.
4. La implementación concreta depende fuertemente de la configuración que se ha logrado al momento.
5. Del mismo modo, esta etapa podría ser redundante para el caso de personas que trabajan individualmente y han logrado una funcionalidad efectiva hasta la etapa III.