ESCAPE

Emergency System for Calculating Adaptive Pedestrian Evacuations

Manual del usuario

Por:

Luis Moya
Departamento de Ingeniería, Pontificie Universidad Católica del Perú (PUCP), Perú

Julio Ramírez
Departamento de Ingeniería, Pontificie Universidad Católica del Perú (PUCP), Perú

Erick Mas
Departamento de Ingeniería, Pontificie Universidad Católica del Perú (PUCP), Perú

Shunichi Koshimura
Departamento de Ingeniería, Pontificie Universidad Católica del Perú (PUCP), Perú

Financiado por CONCYTEC-PROCIENCIA
(Contrato Número PE501078853-2022)

Abril 2025

Índice general

Parte I

Referencias

0.1. Lista de comandos

0.1.1. `nodesFile`

Atributo	Valor	Detalle
Tipo	`String`	Tipo de dato utilizado para este comando.
Por defecto	`nodes.csv`	Nombre predeterminado del archivo que contiene la información de intersecciones.
Descripción	Nombre del archivo de datos de las intersecciones.	

0.1.2. `linksFile`

Atributo	Valor	Descripción
Tipo	`String`	Tipo de dato utilizado para este comando.
Por defecto	`links.csv`	Nombre predeterminado del archivo que contiene la información de las calles.
Descripción	Nombre del archivo de datos de las calles.	

0.1.3. `populationsFile`

Atributo	Valor	Descripción
Tipo	`String`	Tipo de dato utilizado para este comando.
Por defecto	`population.csv`	Archivo predeterminado que contiene la población.
Descripción	Nombre del archivo de datos de las personas.	

0.1.4. `startTime`

Atributo	Valor	Descripción
Tipo	`Integer`	Tipo de dato utilizado para este comando.
Por defecto	`0`	Valor predeterminado utilizado para definir tiempo de inicio
Descripción	Es el tiempo de inicio de la evacuación, este debe ser ingresado en segundos.	

0.1.5. `process`

Atributo	Valor	Detalle
Tipo	`String`	Tipo de dato utilizado para este comando.
Por defecto	`calibration`	Valor predeterminado utilizado para definir el proceso de simulación.
Descripción		Indica el proceso de la simulación.
- Calibration	En este proceso el código necesita entrenar al algoritmo `Sarsa` para que tome sus propias decisiones. El proceso de calibración permite explorar y experimentar decisiones para almacenar los resultados del aprendizaje, este proceso puede repetirse según la cantidad de simulaciones deseadas.	
- Trained	En este proceso el algoritmo `Sarsa` utiliza lo que aprendió. Solo tiene permite utilizar el aprendizaje anterior y en base a ello utilizarlo para modelar una simulación.	

0.1.6. `stopAt`

Atributo	Valor	Detalle
Tipo	`String`	Tipo de dato utilizado para este comando.
Por defecto	`endTime`	Opción predeterminada utilizada para definir el tiempo final de evacuación.
Descripción	Opciones de terminar el tiempo de parar la evacuación. Solo implementado una forma, `endTime`.	

0.1.7. `endTime`

Atributo	Valor	Detalle
Tipo	`Integer`	Tipo de dato utilizado para este comando.
Descripción	Es para introducir el último tiempo de evacuación en el que se requiere realizar la simulación.	

0.1.8. `stopSimulationAt`

Atributo	Valor	Detalle
Tipo	`String`	Define el tipo de dato como cadena de caracteres.
Por defecto	`endNumberSimulation`	La cantiadad de simulaciones para terminar.
Descripción	Opciones para terminar las simulaciones. Implementadas dos formas: `endNumberSimulation` y `addNumberSimulation`.	

0.1.9. `endNumberSimulation`

Atributo	Valor	Detalle
Tipo	`Integer`	Define el tipo de dato como cadena de caracteres.
Descripción	Es la cantidad de simulaciones que se realizará, se introduce la última iteración de la simulación. Tener en cuenta que es obligatorio colocar este valor.	

0.1.10. `meanRayleight`

Atributo	Valor	Detalle
Tipo	`Integer`	Tipo de dato utilizado para este comando.
Por defecto	`420`	Valor predeterminado utilizado al número promedio de Rayleight.
Descripción	Es el número promedio de la distribución Rayleight (único parámetro modificable de la distribución). Este permite calcular el factor de escala para obtener un número según la distribución mencionada. El valor debe ser ingresado en segundos.	

0.1.11. `numberLinkDivision`

Atributo	Valor	Detalle
Tipo	`Integer`	Tipo de dato utilizado para este comando.
Por defecto	`10`	Valor predeterminado para subdivision de una calle.
Descripción	Es el número de particiones que se le realizan a las calles para obtener un conteo de peatones.	

0.1.12. `exploration`

Atributo	Valor	Detalle
Tipo	`Integer`	Tipo de dato utilizado para este comando.
Por defecto	cercano a `0.2`	Valor predeterminado utilizado el porcentaje de exploración al final de la simulación.
Descripción	Es el número de exploración final. Al transcurrir las simulaciones, el ratio de exploración irá cambiando hasta llegar al número ingresado.	

0.1.13. graphicPrintout

Atributo	Valor	Detalle
Tipo	`Logical`	Tipo de dato utilizado para este comando.
Por defecto	`Yes`	Activado para exportación de variables.
Descripción	Sirve para exportar las variables de posición, velocidad, cantidad de evacuados.	

0.1.14. graphicPrintoutPeriod

Atributo	Valor	Detalle
Tipo	`Integer`	Tipo de dato utilizado para este comando.
Por defecto	`1`	Valor predeterminado asociado al atributo.
Descripción	Es el intervalo de tiempo que se debe introducir para saber cada cuanto tiempo se exportan las variables. Es el intervalo de exportación. Puede ser mayor que 1. Debe colocarse en segundos.	

0.1.15. listingPrintoutPeriod

Atributo	Valor	Detalle
Tipo	`Integer`	Tipo de dato utilizado para este comando.
Por defecto	`1`	Valor predeterminado asociado al atributo.
Descripción	Determine cada cuanto tiempo se mostrará los resultados para el terminal. Permite visualizar en tiempo de ejecución.	

0.1.16. pedestrianCountPeriod

Atributo	Valor	Detalle
Tipo	`Integer`	Tipo de dato utilizado para este comando.
Por defecto	`1`	Valor predeterminado asociado al atributo.
Descripción	Determine cada cuanto tiempo se calculará en conteo de personas en las calles.	

0.1.17. `computationContinued`

Atributo	Valor	Detalle
Tipo	`Logical`	Tipo de dato que indica verdadero o falso.
Por defecto	`No`	Valor predeterminado asignado al atributo.
Descripción	Determina si los cálculos serán independientes a un resultado previo.	

0.1.18. `previousComputationFile`

Atributo	Valor	Detalle
Tipo	`String`	Define el tipo de dato como cadena de caracteres.
Por defecto	`population.csv`	Nombre del archivo predeterminado utilizado.
Descripción	Nombre del archivo que contiene resultados de una simulación previa realizada en la misma ciudad. Esta última simulación provee de valores de estados experimentados para la nueva simulación.	

0.1.19. `totalEvacuatedCount`

Atributo	Valor	Detalle
Tipo	`Logical`	Tipo de dato que indica verdadero o falso.
Por defecto	`Yes`	Valor predeterminado asignado al atributo.
Descripción	Permite exportar un archivo con la cantidad de personas evacuadas totales en el tiempo de evacuación del proceso de `trained`.	

0.1.20. `pythonVersion`

Atributo	Valor	Detalle
Tipo	`Logical`	Tipo de dato que indica verdadero o falso.
Por defecto	`No`	Valor predeterminado asignado al atributo.
Descripción	Permite utilizar datos de aprendizajes obtenidos de la versión de Python.	

Parte II

Manuel de usuario

Capítulo 1

Introducción

1.1. Presentación del sofware ESCAPE®

El código ESCAPE®, utiliza el algoritmo de `Sarsa` para resolver ecuaciones de velocidad tiempo para evaluar las rutas de escape ante la llegada de un tsunami a las costas de las cuidades.

Los principales resultados de la evacuación de cada peatón son los siguientes:

- Tiempo de evacuación
- Posición
- Velocidad

1.2. Características Principales

Las características principales del software son las siguientes:

- `Software`: Desarrollado en C++, un lenguaje de programación de bajo nivel que garantiza un rendimiento más eficiente y rápido.
- `Evacuaciones`: Facilita la planificación y ejecución de evacuaciones urbanas.
- `Tiempo de evacuación`: Permite calcular con precisión el tiempo total requerido para evacuar a los habitantes de una ciudad.
- `Puntos de destino`: Proporciona herramientas para identificar y establecer puntos estratégicos de evacuación en áreas urbanas.

1.3. Contactos

Si se tiene alguna duda durante el uso del software, puede contactarnos enviando un email y tener una asistencia técnica.

Correos electrónicos:
lmoya@pucp.edu.pe
julio.ramirez@pucp.edu.pe

1.4. Requisitos

- Lenguaje: C++ (se requiere soporte para C++17)
- Compilador: GCC 9.4 o superior / Clang 10.0 o superior
- Herramientas de Construcción:
  - GNU Make 4.2.1 (Makefile)
  - Plataforma: Ubuntu 20.04.6 LTS

Capítulo 2

Fundamentos

El algoritmo `Sarsa` (Estado-acción-recompensa-estado-acción) fue propuesto por Rummery y Niranjan bajo el nombre de "Modified Connectionist Q-Learning"(Q- Learning conexionista modificado) (MCQ-L). Siendo Rich Sutton, quien le asigna el nombre de `Sarsa` .

El nombre `Sarsa` se debe a la siguiente cadena: La función principal para actualizar el valor Q, depende del estado actual del agente "S 1 ", la acción elegida por el agente .ª 1 ", la recompensa R 2 "que obtiene el agente por elegir esta acción. El estado "S 2 .en el que entra el agente después de realizar esa acción y, por último, la siguiente acción .ª 2 "que elige el agente en su nuevo estado. Siendo el resultado el acrónimo de la quíntuple $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$.

2.1. Algoritmo Sarsa

El algoritmo `Sarsa` es el siguiente:

$$Q^{\text{new}}(S_t, A_t) \longleftarrow (1-\alpha)Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) \tag{2.1}$$

A un agente `Sarsa` se le conoce como algoritmo de aprendizaje porque interactúa con el entorno y actualiza la política en función de las acciones realizadas. El valor Q de un estado-acción se actualiza mediante un error y se ajusta con un índice de aprendizaje "α".

Los valores Q representan la posible recompensa recibida en el siguiente paso temporal por realizar la acción a en el estado s, más la recompensa futura descontada recibida de la siguiente observación del estado-acción.

Una ventaja que tiene `Sarsa` es que aprende por sí mismo los valores Q asociados a la adopción de la política que sigue.

2.2. Parámetros Sarsa

A continuación, se muestran los parámetros que se utilizan con `Sarsa` :

- `Índice de aprendizaje (α)`: Determina hasta qué punto la información recién adquirida anula la antigua. Un factor 0 hace que el agente no aprenda nada, mientras que un factor 1 hace que el agente tenga en cuenta únicamente la información más reciente.

- `Factor de descuento` (γ): Determina la importancia de las recompensas futuras. Un factor de descuento 0 hace que el agente sea oportunista o miope, considerando solo las recompensas actuales. Por otro lado, un factor cercano a 1 incentiva al agente a buscar recompensas elevadas a largo plazo. Si el factor de descuento es igual o superior a 1, el valor Q puede divergir.

- `Condiciones iniciales` ($Q(S_0, A_0)$): Dado que `Sarsa` es un algoritmo iterativo, antes de la primera actualización asume una condición inicial. Un valor inicial alto o infinito, también conocido como çondiciones iniciales optimistas", puede fomentar la exploración. La regla de actualización inicial asigna valores altos a todas las acciones, aumentando su probabilidad de elección. La primera vez que se realiza una acción, la recompensa "r" restablece las condiciones iniciales y se utiliza para fijar el valor de Q, permitiendo un aprendizaje inmediato en caso de recompensas deterministas.

Capítulo 3

Estructura y algoritmo del programa

3.1. Estructura del programa

El software ESCAPE® se ejecuta de acuerdo con la estructura que se muestra en la siguiente figura. Se muestra la información de entrada que necesita el programa, como las calles (link), intersecciones de las calles (node) y las personas (pedestrian), cada una con su identificador. Una vez se introduzca toda la información se realiza el procesamiento de datos que consiste.

Luego de procesar de datos, se tiene como salidas la demostración en tablas y/o figuras de la cantidad de evacuados vs tiempo, cantidad de evacuados vs puntos de evacuación y el total de evacuados vs número de simulación.

3.2. Lista de clases

Archivo	**Descripción**
`io`	Entrada y salida de archivos.
`directory`	Diccionario de comandos.
`node`	Definición de intersecciones en una ciudad.
`link`	Definición de calles en una ciudad.
`sublink`	Definición de una subsección de una calle.
`pedestrian`	Definición de peatones.
`stateMatrix`	Estructura de matriz de estados.
`sarsa`	Cómputo de algoritmo `Sarsa` .

Cuadro 3.1: Nombres de las clases utilizados y implementadas en el modelo.

3.3. Módulos del sofware

3.3.1. Módulo Node (Intersecciones)

El módulo Node presenta los siguientes miembros:

- `idNode`: Son los identificadores únicos de cada intersección.
- `coordenada`: Contiene el vector de ubicación de la intersección.
- `linkConnectionsPtr`: Es un vector de las calles interconectadas.
- `stateMatrixsExperimentadosPtr`: Es un vector de los estados experimentados.

Los miembros `idNode` y `coordenada` se identifican con el archivo de intersecciones; mientras que, `linkConnectionsPtr` y `stateMatrixsExperimentadosPtr`, son miembros que se obtienen a medida que el software procesa los datos.

Asignación de valores de intersección

Para introducir los valores de las intersecciones se necesita un archivo `csv` donde se encuentren los valores de identificación de calles, posición en x y posición en y.

El archivo de control `controlDict` tiene el comando `nodesFile` para especificar el nombre del archivo, por default este es `node.csv`.

A continuación, se muestra un ejemplo del cambio de nombre del archivo de intersecciones:

```
1 nodesFile: interseccionesLima.csv
```

3.3.2. Módulo Link (Calles)

El módulo Link presenta los siguientes miembros:

- `idLink`: Identificador único de una calle.
- `node1Ptr`: Puntero a la primera intersección de la calle.
- `node2Ptr`: Puntero a la segunda intersección de la misma calle.
- `length`: Largo de la calle.
- `width`: Ancho de la calle.
- `orientacionLink`: Orientación de la calle.
- `anchoSubdivision`: Ancho de cada subdivisión de la calle.
- `cantidadSubdivisiones`: Cantidad de subdivisiones de la calle, usadas para el cálculo de densidad.
- `densityLevel`: Densidad de la calle, calculada como la máxima densidad entre sus subdivisiones.
- `subdivisiones`: Vector de la clase `subLink`, que representa las divisiones de la calle.

Asignación de valores de calles

Para introducir los valores de las calles se necesita un archivo donde se coloque un número identificador de la calle, dos puntos de inicio y fin de la calle, la longitud y el ancho de la calle. El archivo de control `controlDict` tiene el comando `linksFile` para especificar el nombre del archivo, que por default es `link.csv`.

A continuación, se muestra un ejemplo del cambio de nombre del archivo de calles:

```
linksFile: callesLima.csv
```

Opciones para la subdivisión de una calle

Existen dos opciones para subdividir una calle:

1. `anchoSubdivision`
2. `cantidadSubdivisiones`

Por default se encuentra activado la opción `anchoSubdivisión`. Esta opción permite dividir la calle de acuerdo a una longitud fija de cada subdivisión y con ello se calcula la cantidad de subdivisiones. En caso la cantidad de subdivisiones no es un entero se procede a redondear y se vuelve a calcular un nuevo ancho, el cual es muy cercano al anterior, esto se realiza debido a que no se puede tener una cantidad de subdivisiones con valor decimal.

Por otro lado, la segunda opción permite ingresar un número de divisiones a la calle y con ello se calculará un ancho único cada subdivisión.

A continuación, se muestra un ejemplo de la primera opción que aparece por default para la subdivisión de una calle:

```
opcionSubdivision: anchoSubdivision
```

Ancho de subdivisiones

Para poder utilizar esta opción no es obligatorio elegir `anchosubdivision` en el comando `opcionSubdivisión`, debido a que ya esta por defecto esa opción.

Permite colocar el valor de la longitud de las subdivisiones. Los valores pueden ser enteros o decimales, pero no fracciones.

A continuación, se muestra un ejemplo para la subdivisión de una calle:

```
anchoSubdivision: 12.5
```

Cantidad de subdivisiones

Para poder utilizar esta opción es necesario elegir `cantidadSubdivisiones` en el comando `opcionSubdivisión`.

El comando `cantidadSubdivisiones` permite dividir la calles en una cantidad entera de subdivisiones.

A continuación, se muestra un ejemplo de uso del comando:

```
cantidadSubdivisiones: 20
```

3.3.3. Módulo nodeDestino (Puntos de evacuación)

El módulo `nodeDestino` presenta los siguientes miembros:

- `maxPersonasEvacuadas`: Es la cantidad de personas que puede evacuar en este sitio. Por defecto, esta opción está desactivada, por lo tanto, cada punto de evacuación tiene una capacidad infinita. Para activar esta opción y limitar la capacidad, se debe modificar el archivo de intersecciones colocando el valor 2, seguido de la cantidad máxima de evacuados para ese punto.

Impresión de imagen de Evacuados vs Tiempo para diferentes simulaciones

Muestra una línea con la información del total de personas evacuadas en todo el tiempo de evacuación para ciertas simulaciones. Se agrega el número deseado de simulación con el comando totalEvacuadosVsSimulacionAt, mostrando cómo evoluciona el aprendizaje del algoritmo. Esta información se exporta en formato png.

A continuación, se muestra un ejemplo de activación de figura:

```
figureTotalEvacuadosVsTiempo: yes
```

Impresión de tabla de Evacuados vs Tiempo para diferentes simulaciones

Imprime una tabla con los valores de total de evacuados en todo el tiempo de evacuación para ciertas simulaciones. Se agrega el número deseado de simulación con el comando totalEvacuadosVsSimulacionAt, mostrando cómo evoluciona el aprendizaje del algoritmo. Esta información se exporta en formato csv.

A continuación, se muestra un ejemplo de activación de tabla:

```
tableTotalEvacuadosVsTiempo: yes
```

Evacuación vs Tiempo para simulaciones específicas

La impresión de la tabla y la figura del total de evacuados vs tiempo utiliza los números de simulaciones 1, 2 y 3 por defecto. Para cambiar estos valores se puede hacer uso del comando totalEvacuadosVsTiempoAt, en los cuales se debe asignar valores enteros separados por comas.

A continuación, se muestra un ejemplo de la asignación de simulaciones:

```
  evacuadosVsTiempoAt 1, 5, 8;
```

Impresión de imagen del Total de evacuados Vs Simulación

Muestra una gráfica de una línea con la información del total de personas evacuadas a lo largo de las simulaciones. Mostrando cómo evoluciona el aprendizaje del algoritmo. Esta información se exporta en formato png.

A continuación, se muestra un ejemplo de activación de figura:

```
  figureTotalEvacuadosVsSimulacion: yes
```

Impresión de tabla del Total evacuados Vs Simulación

Muestra una tabla con los valores de total de evacuados a lo largo de las simulaciones. Mostrando cómo evoluciona el aprendizaje del algoritmo. Esta información se exporta en formato csv.

A continuación, se muestra un ejemplo de activación de tabla:

```
tableTotalEvacuadosVsSimulacion: yes
```

Periodo de simulaciones del Total de evacuación vs Simulación

La impresión de la tabla y la figura del total de evacuados vs Simulación se realiza cada simulación, el cual es un valor por defecto. Para cambiar este valor se puede hacer uso del comando totalEvacuadosVsTSimulacionPerido, en los cuales se debe asignar el valor de cada cuanto se desean valores de simulación.

A continuación, se muestra un ejemplo de la asignación del periodo de simulaciones:

```
totalEvacuadosVsSimulacionPeriod 5;
```

3.3.4. Módulo Tiempo

El módulo Tiempo presenta los siguientes miembros:

- `valorTiempo`: Valor del tiempo simulado que aumenta según el `deltaT`.
- `deltaT`: Valor de paso del tiempo. Por defecto, es 1 segundo.
- `startNumberSimulation`: Es el valor de la simulación inicial. Por defecto, es 0.
- `endNumberSimulation`: Es el valor de la última simulación. Este debe ser ingresado en el archivo `control.l`.
- `graphicPrintoutPeriod`: Valor que indica cada cuánto debería exportarse las variables de posición, velocidad y cantidad de personas evacuadas.
- `pedestrianCountPeriod`: Valor que indica cada cuánto debería calcularse la densidad.

Tiempo de evacuación

Este valor sirve para asignar de manera obligatoria el tiempo de evacuación en segundos.

A continuación, se muestra un ejemplo:

```
endTime 800;
```

Colocación del número de simulaciones

Este valor sirve para asignar de manera obligatoria el número de simulaciones. Es decir, cada simulación realiza una evacuación. Solo se aceptan valores enteros y mayor a 0.

A continuación, se muestra un ejemplo:

```
endNumberSimulation 7000;
```

Periodo de impresión de variables

Este comando permite imprimir las variables cada cierto tiempo, el cual es asignado en segundos. Por ejemplo, si el comando tiene el valor de 5 y el comando endTime 50, entonces se imprimirán las variables en la carpeta data en los siguientes tiempos: 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 por default.

A continuación, se muestra un ejemplo:

```
graphicPrintoutPeriod 5;
```

Periodo de cálculo de densidad

Este comando permite realizar el conteo de las personas en cada subdivisión de la calle para luego poder calcular la densidad de cada subdivisión y obtener el máximo valor para asignarlo a la calle con el nivel de densidad. Por ejemplo, si el comando tiene el valor de 6 y el comando endTime 50, entonces se calculará la densidad en los tiempos 0, 6, 18, 24, 30, 36, 42 y 48.

A continuación, se muestra un ejemplo:

```
pedestrianCountPeriod 6;
```

Capítulo 4

Instalación

Esta guía detalla en proceso de instalación, simulación y post-procesamiento para algunos casos de aplicación del código `Sarsa` . El directorio `run` contiene casos demostrativos del uso del algoritmo y sus utilidades en diferentes escenarios. Antes interatar correr los tutoriales, el usuario debe asegurarse que `Sarsa` este correctamente instalado.

4.1. Clonando el repositorio

Para clonar el repositorio, ejecutar los siguiente:

```
> git clone
   https://github.com/juliocesar-ramirez/cplus-version.git
```

4.2. Cargando las variables

Las variables del entorno de `Sarsa` estan definidas en los archivos del directory `Sarsa` /etc.

```
> cd cplus-version
> source bashrc
```

4.3. Compilando Sarsa

Con el entorno cargado, se empieza a compilar `Sarsa` ejecutando el script en makefile.

```
> make
```

4.4. Actualización del código

El código fuente es actualizado regularmente. Por tanto, puede ser descagando estas modificaciones y archivos nuevos utilizando git.

```
> git pull
```

Capítulo 5

Simulación de la evacuación

5.1. Prescripción de condiciones iniciales

Existen dos condiciones iniciales para iniciar la simulación de la evacuación:

- `Posiciones de las personas al iniciar la evacuación`: Las posiciones donde están ubicadas las personas no son modificables mediante el archivo de nodos.
- `Tiempo de salida de personas al inciar la evacuación`: El tiempo en el que las personas comenzarán a evacuar es una condición inicial basada en la distribución de Rayleigh, cuyo parámetro puede ser ajustado por el usuario.

5.2. Inputs y outputs

Una lista de archivos son usados para el programa ESCAPE®, algunos de ellos son de entrada y otros de salida. Asimismo, algunos de ellos son opcionales. Los archivos de entrada son los siguientes:

- El archivo de calles (obligatorio), contiene la lista de calles de la cuidad y su ubicación.
- El archivo de interseciones (obligatorio), contiene información de las intersecciones de las calles.
- El archivo de personas (obligatorio), contiene información de los peatones y su posición de inicio.

5.2.1. Archivo de intersecciones

El usuario puede cambiar el nombre del archivo modificando el comando nodesFile. El archivo contiene lo siguiente:

- Id de la intersección, número único y no puede ser igual a otra intersección.
- Coordenada x
- Coordenada y

```
# Ejemplo del archivo nodes.csv
# id x y
0,0,0,0,1
1,20,0,0,1
2,40,0,0,1
3,60,0,0,1
4,80,0,0,1
```

El orden no es importante. Lineas que empiezan con # son tomados como comentarios. La cantidad de puntos es libre y no necesariamente deben ser las misma al ejemplo.

5.2.2. Archivo de calles

El usuario puede cambiar el nombre del archivo modificando el comando linksFile. El archivo contiene lo siguiente:

- Id de la calle, número único y no puede ser igual a otra calle.
- Id de la intersección de inicio.
- Id de la intersección de salida.

```
# Ejemplo del archivo nodes.csv
# id,node1,node2,widht,
0,0,1,20,3
1,1,2,20,3
2,2,3,20,3
3,3,4,20,3
4,4,5,20,3
```

El orden no es importante. Lineas que empiezan con # son tomados como comentarios. La cantidad de calles es libre y no necesariamente deben ser las misma al ejemplo.

5.2.3. Archivo de personas

El usuario puede cambiar el nombre del archivo modificando el comando populations-File. El archivo contiene lo siguiente:

- Edad de la persona
- Genero de la persona, si hombre es 1.
- Categoria HHType
- Categoria HHID
- Interseción de inicio de la persona.

```
# Ejemplo del archivo population.csv
# age,gender,HHType,HHID,closeNode
18,1,0,0,26
18,1,0,0,22
18,1,0,0,20
18,1,0,0,6
```

El orden no es importante. Lineas que empiezan con # son tomados como comentarios. La cantidad de personas es libre y no necesariamente deben ser las misma al ejemplo.

5.2.4. Archivo de control

El usuario de ver este archivo llamado controlDict, ubicado en el directorio system. Importación de datos de entrada obligatorios y opcionales, control de tiempo, exportación.

El archivo representa un panel de control del código ESCAPE®. Contiene un número de comandos donde los valores pueden ser asignados o usados con sus valores con default. Todos los comandos están definidos en manual de referencias.

```
1  nodesFile         nodes.csv;
2
3  linksFile         links.csv;
4
5  populationsFile   population.csv;
6
7  stopAt           endTime;
8
9  endTime          800;
10
11 deltaT           1;
12
13 graphicPrintoutPeriod 1;
14
15 listingPrintoutPeriod 1;
16
17 computationContinued    no;
18
19 previousComputationFile sim_000006000.csv;
20
21 # stopSimulationAt          endNumberSimulation;
22
23 # endNumberSimulation 6003;
24
25 readPedestrianMassState no;
26
27 stopSimulationAt          addNumberSimulation;
```

Lineas que empiezan con # son tomados como comentarios.

5.3. Iniciar Programa

Los calculos se mandan via el terminal mediante el comando sarsa. El comando activa la ejecución de un script principal de todos los módulos, funciones y variables del codigo. La sintaxis del comando es la siguiente:

El programa se invoca escribiendo el comando `Sarsa` en la terminal. El comando ejecuta un archivo binario guardado en la carpeta `bin`, que es un binario de todos los scripts de la carpeta `src`.

La sintaxis del comando es la siguiente:

```
1 > sarsa
```

5.4. Casos ejemplos

ESCAPE® incluye tres casos tutoriales para mostrar las funcionalidades del código. Estos se encuentran disponibles en la carpeta `tutorials`.

5.4.1. Caso Grilla

Este tutorial va a describir cómo pre-procesar y post-procesar un caso donde se tiene una ciudad ideal en forma de un rectangulo. La geometria es mostrada en la figura, donde las líneas son las calles. Las personas se encuentran posicionadas inicialmente en los puntos de intersección de las calles. La velocidad iniciales de movimiento de las personas será de 1 m/s.

Este ejemplo mostrará la ruta de evacuación más óptima cuando las personas salen de un mismo punto inicial hasta su punto de evacuación. La configuración de la ciudad presenta calles con un patrón característico que son uniones de 4 calles que forman un rectángulo, con un total de 25 bloques con la misma forma, como se muestra en la siguiente imagen.

Asimismo, una calle tiene una longitud de 20 metros y un ancho 3 metros. Esta configuración presenta un único punto de evacuación ubicado en la posición (100, 100).

El usuario debe colocar cierta información en el controlDict:

- `endTime`: Tiempo final de evacuación con el comando `endTime`.
- `endNumberSimulation`: El número final de simulaciones con el comando `endNumberSimulation`.
- `process`: Tipo de proceso con el comando `process`.
- `meanRayleigh`: El valor medio para la distribución de Rayleigh.
- `popultationsFile`: El nombre del archivo de población.

Los datos de entrada acerca de control del tiempo, lectura, impresión y valores de la solución serán leídos del archivos controlDict. El usuario deberá modificarlo previamente.

Para el presente caso se muestra en la siguiente imagen el archivo mencionado anteriormente, así como los valores de los comandos obligatorios.

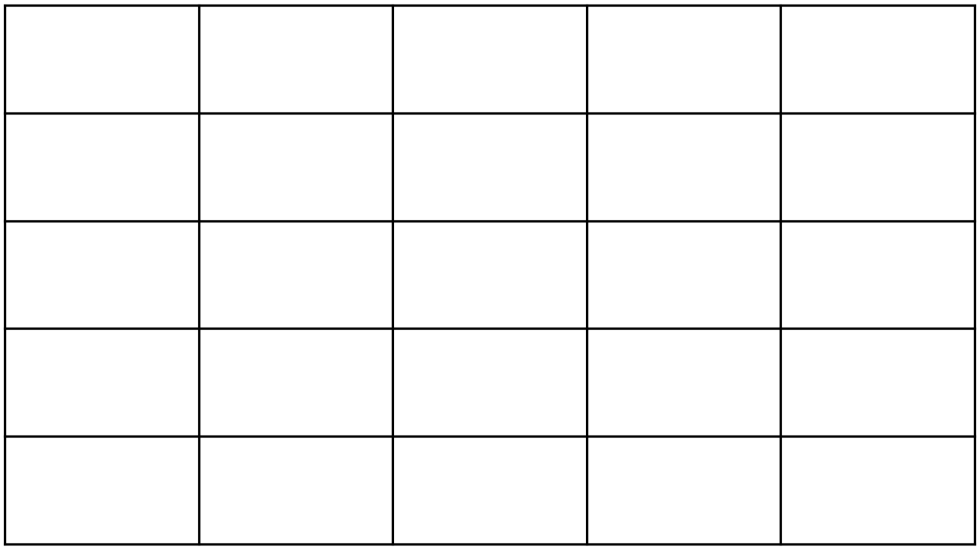

Figura 5.1: Esquema en planta de la cuidad del caso grilla.

```
populationsFile  populationAtPoint.csv;

process calibration;

endTime          1800;

meanRayleigh    420;

endNumberSimulation 1000;
```

Existen comandos que son necesarios especificar su valor debido a que estará utilizando su valor por defecto, el cual está listado en la sección de comandos.

Una vez definido el controlDict, se recomienda realizar una limpieza de la carpeta de trabajo, para ello existe un script externo con nombre de archivo Allclean.

La ejecución del programa se realiza con el comando sarsa. Para el presente caso, se muestra en la siguiente imagen el archivo mencionado anteriormente, así como los valores de los comandos obligatorios.

```
***** Simu: 410 *****
epsilon greedy - exploration: 0.379363
survived pedestrian: 6195
Duracion: 0 min / 3 s / 3018 ms

***** Simu: 411 *****
epsilon greedy - exploration: 0.378788
survived pedestrian: 6216
Duracion: 0 min / 3 s / 3040 ms

***** Simu: 412 *****
epsilon greedy - exploration: 0.378215
survived pedestrian: 6225
```

```
Duracion: 0 min / 2 s / 2899 ms

***** Simu: 413 *****
epsilon greedy - exploration: 0.377644
survived pedestrian: 6231
Duracion: 0 min / 2 s / 2972 ms

***** Simu: 414 *****
epsilon greedy - exploration: 0.377074
survived pedestrian: 6220
Duracion: 0 min / 3 s / 3005 ms
```

A su término, se podrá obtener los resultados de cada variable, tablas con información de número de evacuados e imágenes de validación.

Por último, la exportación de las variables permite al usuario post-procesar la información en cualquier otro programa. A continuación, se muestra el post- procesamiento de la información con el lenguaje python.

5.4.2. Caso Camana

Este ejemplo mostrará la evacuación de personas en el distrito de Camaná con varias opciones de puntos de evacuación, todas alejadas de la playa, como se muestra en la siguiente figura. La configuración es una idealización con base en la distribución de calles del distrito de Camaná. Con longitud máxima de calle de 600 metros y un ancho de 3 metros.

La configuración de este caso es el siguiente:

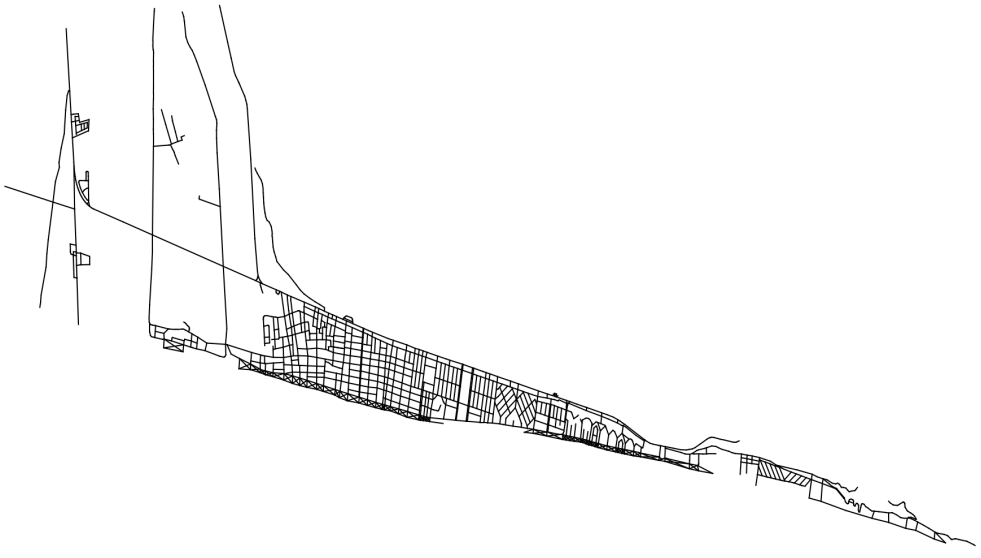

Figura 5.2: Esquema en planta de la cuidad del caso grilla.

- `endTime`: Tiempo final de evacuación: 800 seg.
- `endNumberSimulation`: Número final de simulaciones: 7000.

- `meanRayleigh`: El valor medio para la distribución de Rayleigh. Tiempo promedio de salida de persona de 420 seg.

A continuación, se muestra el archivo controlDict.

```
populationsFile  populationAtPoint.csv;

process calibration;

endTime          1800;

meanRayleigh    420;

endNumberSimulation 1000;
```

Por último, en la carpeta post-procesamiento, existen archivos para realizar visualizaciones de los datos de manera externa al programa principal. El archivo snapshot.py, proporciona imágenes en cada tiempo de paso. El archivo zoom.py permite crear imágenes con una zona amplia para observar a detalle el movimiento de evacuación, como se muestra en la siguiente imagen.

5.4.3. Caso Python

En caso se quiera usar datos de stateMatrix de simulaciones anteriores, pero de la versión en python del mismo código, se debe activar el comando pythonVersion. A continuación, se muestra el controlDict del caso grilla utilizando datos exportados del código en python. Se debe activar esta opción debido a que el orden de las conexiones de las calles en una intersección es diferente entre los diferentes lenguajes.