

Universidad de
Burgos

CE Aplicada al Transporte Físico



Vadim Budagov:

vbv1002.alu.ubu.es

Willow Maui García:

wgm1001@alu.ubu.es

Indice

1. Introducción.....	2
2. Population Based Training (PBT)	2
2.1 Descripción técnica del problema	2
2.2 Descripción del algoritmo.....	3
2.3 Resultados obtenidos y conclusiones.....	7
3. Algoritmo genético para la optimización del control del tráfico.....	7
3.1 Introducción	7
3.2 Descripción de los problemas	8
3.3 Descripción de los algoritmos	9
3.4 Resultados.....	11
3.5 Conclusiones	12
4. Bibliografía	13

Lista de Ilustraciones

Ilustración 1: Sequential Optimisation.....	3
Ilustración 2: Parallel Random/Grid Search	4
Ilustración 3: Deep Learning Lifecycle	4
Ilustración 4: PBT	5
Ilustración 5: Binary Tournament (Exploit)	5
Ilustración 6: Binary tournament (Explore).....	6
Ilustración 7: PBT Algoritmo	6
Ilustración 8: Formulación problema de tráfico	8
Ilustración 9: Ecuación de fitness dl algoritmo de gestión de tráfico	9
Ilustración 10:Proceso del algoritmo para optimización del tráfico	9
Ilustración 11: Datos pruebas algoritmo de gestión del trafico	11
Ilustración 12: Simulación de tráfico con incidente	12

1. Introducción

En el marco de la asignatura Computación Neuronal y Evolutiva vamos a dedicarnos con el tema de Computación Evolutiva aplicada al transporte físico. Para realizar la investigación hemos elegido dos artículos actuales en inglés que describen no solo los problemas en el sector conducción autónoma sino algunos algoritmos desarrollados para resolverlos.

Al primer trataremos el algoritmo PBT y luego veremos el algoritmo genético para la optimización del control del tráfico.

2. Population Based Training (PBT)

Los ingenieros de los departamentos de Alphabet, Waymo y DeepMind se han unido para encontrar un proceso más eficiente para la capacitación y la optimización de los algoritmos de autoaprendizaje de la empresa para la conducción autónoma. Utilizaban una técnica llamada entrenamiento basado en la población (PBT), desarrollado previamente por DeepMind para mejorar los videojuegos. Waymo es una empresa que desarrolla tecnologías para vehículos autónomos. Continúa el trabajo del proyecto Google Driverless Car de Alphabet y se fundó en diciembre de 2016 como subsidiaria de Alphabet.

Un método de entrenamiento más eficiente para redes neuronales podría proporcionar una ventaja decisiva en el mundo altamente competitivo de la conducción automatizada. El PBT ofrece actualmente una solución efectiva.

Idea principal: el entrenamiento basado en la población es una técnica de optimización de hiperparámetros similar a los algoritmos genéticos que aprende un programa de hiperparámetros en lugar de valores fijos. Optimiza de forma conjunta parámetros e hiperparámetros.

¿Pero qué significa programa de hiperparámetros? Por ejemplo: programa de factor o tasa de aprendizaje (*ingl. learning rate*) o de conexiones celulares de arquitectura neuronal (*ingl. Neural Architecture Cell Connections*). A continuación, se describen los problemas técnicos para que se implementaba PBT.

2.1 Descripción técnica del problema

Encontrar el mejor régimen de entrenamiento o ("scheduling de hiperparámetros") se logra comúnmente a través de la experiencia e intuición de un ingeniero, o mediante una búsqueda exhaustiva (*ingl. brute-force*). Actualmente existen dos modelos comunes para la sintonización de hiperparámetros: *búsqueda aleatoria* (*ingl. Random/Grid Search Optimisation*) y *ajuste manual* (*ingl. Sequential Optimisation*).

En la *búsqueda aleatoria*, los investigadores aplican muchos programas de hiperparámetros aleatorios sobre múltiples tipos de hiperparámetros para entrenar diferentes redes de forma independiente y en paralelo, después de lo cual es posible establecer el modelo de mejor rendimiento.

Con el *ajuste manual*, los investigadores deben adivinar los mejores hiperparámetros, entrenar a sus modelos con ellos y luego evaluar el rendimiento. Esto se hace una y otra vez, hasta que el investigador esté contento con el rendimiento de la red. Aunque esto puede resultar en un mejor rendimiento, la desventaja es que esto lleva mucho tiempo, a veces toma semanas o incluso meses para encontrar la configuración perfecta. Y si bien hay formas de

automatizar este proceso, como la optimización bayesiana, todavía lleva mucho tiempo y requiere muchas sesiones de entrenamiento secuenciales para encontrar los mejores hiperparámetros.

PBT se puede ver como un híbrido de ambos modelos que es un método de ajuste de hiperparámetros que, si bien es muy simple, resulta en un aprendizaje más rápido, menores recursos de cómputo y, a menudo, una mejor solución.

2.2 Descripción del algoritmo

Antes de llegar al algoritmo real, nos gustaría explicar la idea de cómo el PBT-algoritmo estaba desarrollada.

En la ilustración 1 se puede observar como la optimización secuencial funciona. Métodos como el ajuste manual y la optimización bayesiana realizan cambios en los hiperparámetros al observar muchas sesiones de entrenamiento secuencialmente, lo que hace que estos métodos sean lentos. Con el ajuste manual, los investigadores deben adivinar los mejores hiperparámetros, entrenar a sus modelos con ellos y luego evaluar el rendimiento. Es decir, después de cada entrenamiento los resultados se evalúan y de esta manera los nuevos hiperparámetros se eligen y se ajustan. La optimización secuencial requiere que se completen múltiples ejecuciones de entrenamiento (potencialmente con una detención temprana), después de lo cual se seleccionan nuevos hiperparámetros y el modelo se vuelve a entrenar desde cero con los nuevos hiperparámetros. Este es un proceso inherentemente secuencial y conduce a tiempos de optimización de parámetros largos, aunque utiliza recursos computacionales mínimos.

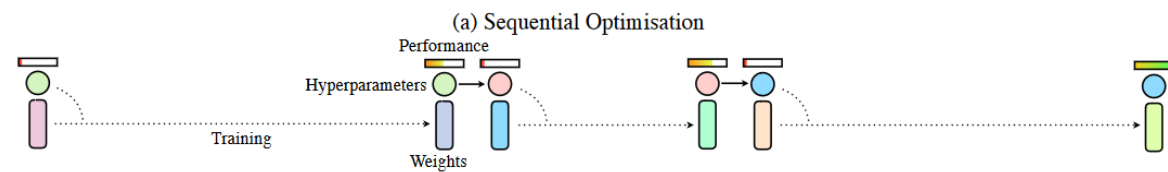


Ilustración 1: Sequential Optimisation

En comparación con optimización secuencial la búsqueda aleatoria se ejecutan varios modelos donde se prueban muchos hiperparámetros en paralelo, pero de forma independiente con la vista de que uno de los modelos se optimizará mejor. En otras palabras, se define diferentes modelos con diferentes hiperparámetros y los ejecuta en paralelo y se selecciona el modelo que dio los mejores resultados. Esto solo requiere el tiempo para un entrenamiento, pero requiere el uso de más recursos computacionales para entrenar muchos modelos en paralelo (Véase la Ilustración 2).

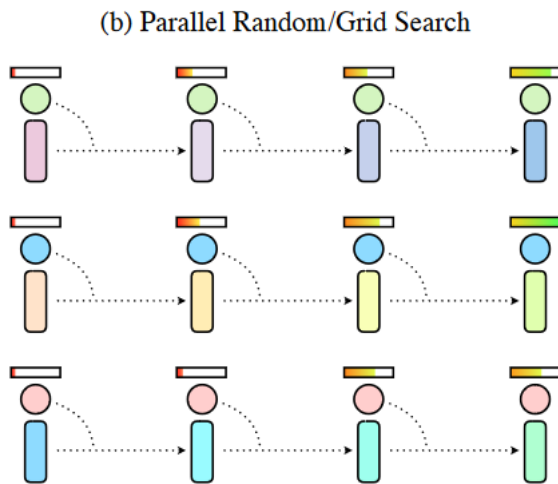


Ilustración 2: Parallel Random/Grid Search

Como hemos visto anterior, hay dos procesos separados básicos en el aprendizaje profundo: el entrenamiento de modelo y el tuneo de hiperparámetros (Ilustración 3). Los investigadores se preguntaban si sería posible combinar estos dos procesos separados para lograr un rendimiento mejor y más rápido. De este modo se formaba la idea del entrenamiento basado en la población que puede hacerlo.

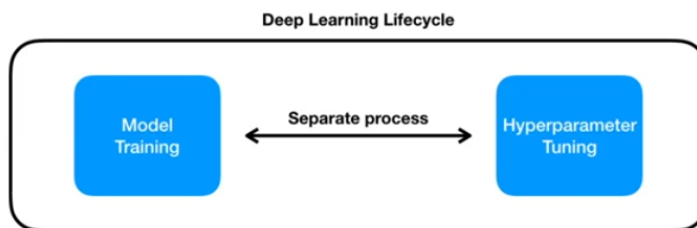


Ilustración 3: Deep Learning Lifecycle

El entrenamiento basado en la población PBT de redes neuronales comienza como una búsqueda aleatoria, pero permite a los trabajadores (*ingl. Workers*) explotar los resultados parciales de otros trabajadores y explorar nuevos hiperparámetros a medida que avanza el entrenamiento (Véase la Ilustración 4). Durante la capacitación de la población de redes neuronales avanza, este proceso de **explotación** y **exploración** se realiza periódicamente, asegurando que todos los trabajadores de la población tengan un buen nivel base de rendimiento y también que se exploren constantemente nuevos hiperparámetros. Esto significa que PBT puede explotar rápidamente buenos hiperparámetros, puede dedicar más tiempo de entrenamiento a modelos prometedores y, crucialmente, puede adaptar los valores de hiperparámetros durante todo el entrenamiento, lo que lleva al aprendizaje automático de las mejores configuraciones.

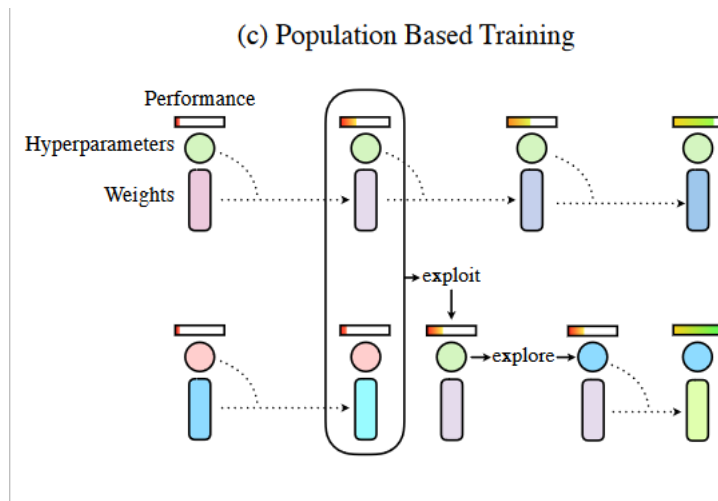


Ilustración 4: PBT

El entrenamiento se divide en generaciones. Después de cada generación, hay N-modelos diferentes de N-trabajadores diferentes. Cada trabajador se confronta con otro para competir. Este proceso se llama torneo binario (*ingl. Binary Tournament*) y se realiza en proceso de explotación.

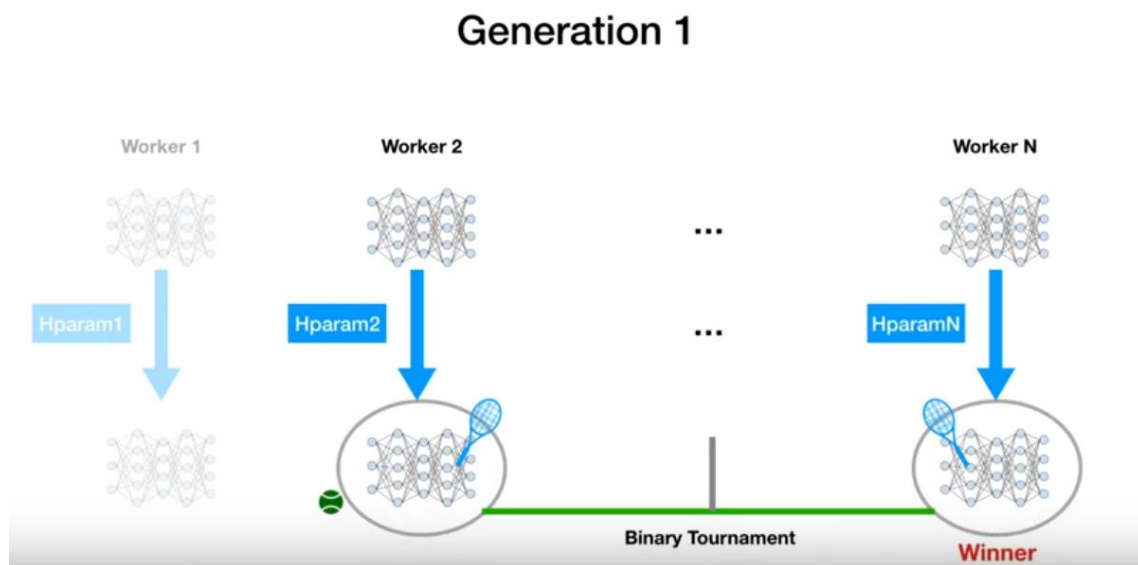


Ilustración 5: Binary Tournament (Exploit)

En este ejemplo, el Trabajador 2 pierde ante el Trabajador-N. Como siguiente paso, en generación 2 el perdedor copia de los pesos y los hiperparámetros del ganador que se realiza en el proceso de exploración (Véase la ilustración 6).

Generation 2

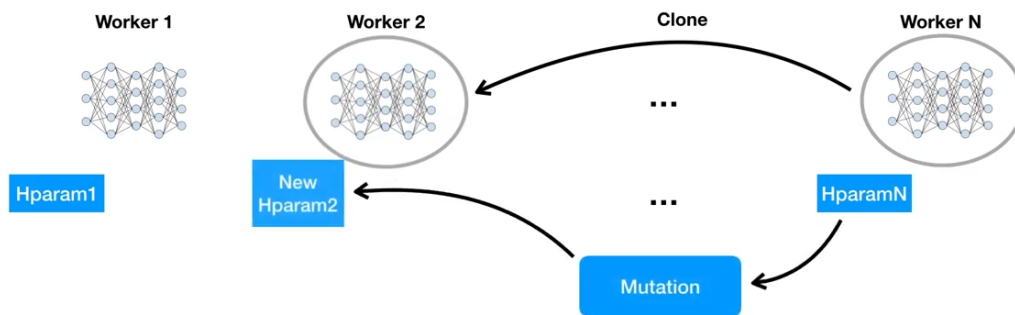


Ilustración 6: Binary tournament (Explore)

La siguiente figura muestra el pseudocódigo del algoritmo PBT.

```

1: procedure TRAIN( $\mathcal{P}$ ) ▷ initial population  $\mathcal{P}$ 
2:   for  $(\theta, h, p, t) \in \mathcal{P}$  (asynchronously in parallel) do
3:     while not end of training do
4:        $\theta \leftarrow \text{step}(\theta|h)$  ▷ one step of optimisation using hyperparameters  $h$ 
5:        $p \leftarrow \text{eval}(\theta)$  ▷ current model evaluation
6:       if  $\text{ready}(p, t, \mathcal{P})$  then
7:          $h', \theta' \leftarrow \text{exploit}(h, \theta, p, \mathcal{P})$  ▷ use the rest of population to find better solution
8:         if  $\theta \neq \theta'$  then
9:            $h, \theta \leftarrow \text{explore}(h', \theta', \mathcal{P})$  ▷ produce new hyperparameters  $h$ 
10:           $p \leftarrow \text{eval}(\theta)$  ▷ new model evaluation
11:        end if
12:      end if
13:      update  $\mathcal{P}$  with new  $(\theta, h, p, t + 1)$  ▷ update population
14:    end while
15:  end for
16:  return  $\theta$  with the highest  $p$  in  $\mathcal{P}$ 
17: end procedure

```

Ilustración 7: PBT Algoritmo

El entrenamiento dura T pasos en total (T está depende del dominio). La función **step()** realiza un paso de optimización, y la función de evaluación evalúa el modelo en el conjunto de validación y devuelve la métrica de evaluación. Aquí hay tres funciones específicas de PBT: **ready()**, **exploit()** y **explore()**, y esto es lo que hacen:

El método **ready()** toma el paso de tiempo actual t , y devuelve un bool si la ejecución del entrenamiento debe compararse con los otros agentes (Workers) en este paso de tiempo o no.

Si **ready()** devuelve True, el rendimiento del modelo se compara con el resto de la población, y si se cumple un criterio, se reemplaza con una copia de un modelo con mejor rendimiento y sus hiperparámetros con la ayuda de **exploit()**-función: en Python-> "`copied_hparams, copied_model, copied_performance = exploit(hparams, model, performance, population)`"

El criterio exacto varía según el método que utilice; **Truncation Selection**, **T-test section o Binary Tournament**.

La función `explore()` se llama si se reemplaza el modelo y el hiperparámetro con uno diferente de la población. Reemplaza los hiperparámetros y el modelo copiados por una versión cambiada.

```
If model  $j$  = copied_model:  
    model = copied_model  
    hparams = explore(copied_hparams)  
    performace = copied_performance
```

Al final se actualizará la población por ejemplo de esta manera:

`population[i] = (model, hparams, performance, t)` y después devolvemos el mejor modelo.

`Best_model = get_best_model(population)`

2.3 Resultados obtenidos y conclusiones

Una cita que describe el comportamiento de PBT: “Population Based Training is a class of students, who are always communicating with each other, sharing their most recent insights with the entire class and constantly attempting to improve on them. Since they’re always in the loop, new ideas spread rapidly across the class, enabling faster discovery of even newer ideas.” (Author: Irhum Shafkat)

Entonces, el PBT ahorra tiempo y recursos. La programación de hiperparámetros descubierto con redes capacitadas en PBT superó a la red anterior de Waymo con la mitad del tiempo y los recursos de capacitación. En general, PBT usa la mitad de los recursos computacionales utilizados por la búsqueda paralela aleatoria para descubrir eficientemente mejores programas de hiperparámetros. También les ahorra tiempo a los investigadores: al incorporar PBT directamente en la infraestructura técnica de Waymo. Los investigadores de toda la empresa pueden aplicar este método con solo hacer clic en un botón y dedicar menos tiempo a ajustar sus tasas de aprendizaje. Desde la finalización de estos experimentos, PBT se aplicaba a muchos modelos diferentes de Waymo y es muy prometedor para ayudar a crear vehículos más capaces para la carretera. En futuro esta podría ser definitivamente una de las mejores formas de desarrollar automóviles autopropulsados seguros.

3. Algoritmo genético para la optimización del control del tráfico

3.1 Introducción

El algoritmo desarrollado en el artículo consiste en un algoritmo genético diseñado para el control del tráfico a través de la señalización, pero lo que le hace realmente especial es que está implementado con algoritmos genéticos y es capaz de en situaciones de incidentes de tráfico, como pueden ser accidentes, es capaz de reorganizar las señales para obtener un flujo hasta un 40% más eficiente que otros algoritmos que no tienen implementada esta característica.

3.2 Descripción de los problemas

El problema que plantean los autores de este artículo es que la mayoría de los algoritmos actuales de optimización del tráfico no tienen en cuenta los incidentes sobre el tráfico, lo cual hace que cuando estos suceden cueste más trabajo el reorganizar el tráfico además de que lo hace menos eficiente al, las señalizaciones, no estar adaptadas a el nuevo flujo del tráfico.

En un incidente en el tráfico hay cuatro fases: la identificación, la verificación, la respuesta y el despeje. Este artículo se centra en la gestión del tráfico en las dos últimas fases. Además, se asume que el incidente ha sido verificado, se ha predicho la duración de la cuarta fase y el número de vías afectadas. El impacto del accidente en el tráfico se calcula usando trabajos previos. El área afectada no solo es la calle del accidente, sino también zonas alrededor de esta que sufren mayor congestión del tráfico a causa de dicho incidente. Para intentar minimizar el impacto de estos incidentes formularon el problema como:

A is the set of links in the network,
 W is the set of origin-destination pairs of the network,
 R_w is the set of routes between origin-destination pair $w \in W$,
 d_a is the queuing delay at link $a \in A$,
 f_r^w is the flow on route $r \in R_w$,
 v_a is the link flow on link $a \in A$,
 λ_a is the "link green split" λ_a which is determined by traffic signals at the end of the link (the definition will be discussed in the next section),
 $t_a(v_a, \lambda_a)$ is the travel time on link $a \in A$ described as a function of link flow v_a and "link green split" λ_a ,
 S_a is the capacity of link $a \in A$,
 σ_{ar}^w is 1 if route r between O-D pair w uses link a , and 0 otherwise,
 D_w is the demand between O-D pair $w \in W$,
 The target is to minimize the total travel time of the network. The target objective function is as follow:

$$\text{minimize } \sum_{a \in A} \int_0^{v_a} t_a(v_a, \lambda_a) dx \quad (1)$$

Subject to

$$\sum_{w \in W} \sum_{r \in R_w} f_r^w \sigma_{ar}^w = v_a, a \in A \quad (2)$$

$$\sum_{r \in R_w} f_r^w = D_w, w \in W \quad (3)$$

$$v_a \leq \lambda_a S_a, a \in A \quad (4)$$

$$f_r^w \geq 0, r \in R_w, w \in W \quad (5)$$

La ecuación 2 representa la relación entre el flujo de ruta y el flujo de enlaces.

La ecuación 3 muestra la conservación del flujo.

La ecuación 5 restringe el límite de flujo en los enlaces.

La ecuación 5 hace que el flujo de los enlaces no sea 0.

La definición de green split es la cantidad de tiempo que un semáforo está en verde para un enlace en una intersección señalizada.

Ilustración 8: Formulación problema de tráfico

3.3 Descripción de los algoritmos

Los inputs del problema serán:

- La configuración de orígenes y destinos
- La tabla de demandas para cada origen y destino
- La configuración de la red
- La tabla de detalles de los enlaces
- La configuración de las señales de tráfico.

Para optimizar computacionalmente el proceso, una solución del algoritmo genético se usa para reducir la carga computacional; el algoritmo toma muestras al azar del espacio de factibilidad total de las combinaciones y elige las más representativas que minimizarían el tiempo de viaje.

La función de fitness, como hemos detallado antes es:

$$Fitness = - \sum_{a \in A} \int_0^{v_a} t_a(v_a, \lambda_a) dx \quad (8)$$

Ilustración 9: Ecuación de fitness dl algoritmo de gestión de tráfico

La cual está en negativo porque se busca maximizar el fitness.

La variable de decisión es un vector con la duración de las distintas fases de los semáforos, para cada una de las intersecciones.

El proceso que sigue la red es el descrito en la siguiente imagen:

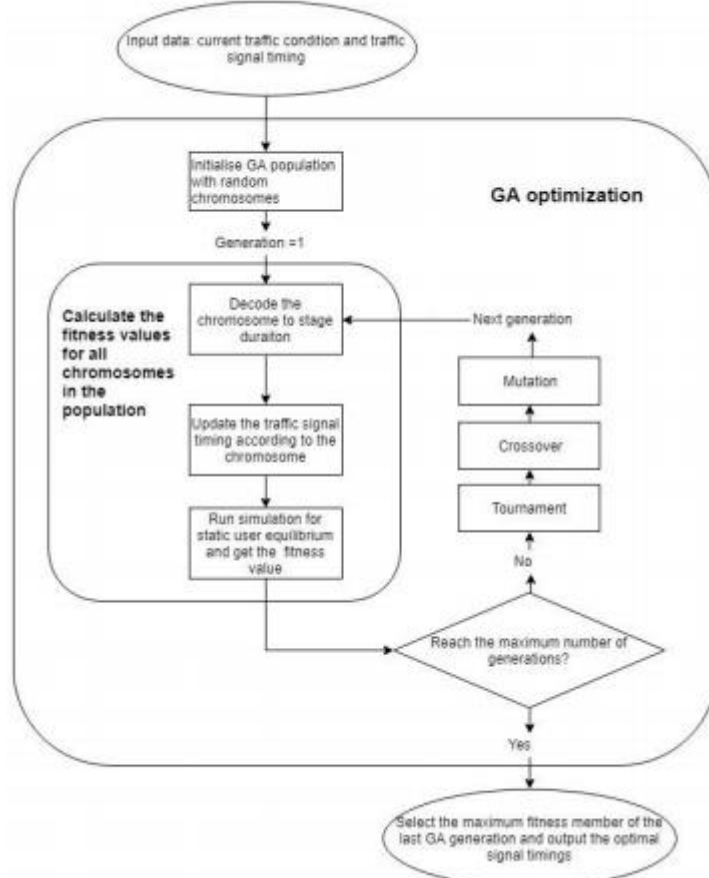


Ilustración 10: Proceso del algoritmo para optimización del tráfico

Descrito por fases es:

1 Preparar los inputs:

Se determinan los distintos parámetros de la red para tener una convergencia rápida con un bajo coste computacional.

2 Inicialización

Se inicializa la población con individuos aleatorios

3 Cálculo del fitness:

Para cada individuo se calcula el fitness. Para ello se usa AIMSUM y en la primera llamada se le inicializa.

4 Se comprueba si se han realizado el máximo de iteraciones:

Si no es así se continua con los siguientes pasos.

5 Selección por torneo:

Con una selección por torne se escogen dos individuos, que han sido seleccionados como el de mayor fitness tras seleccionar a dos aleatorios.

6 Cruce:

Entre los dos individuos del paso anterior, con una probabilidad prefijada, se seleccionan valores del cromosoma de uno u otro individuo.

7 Mutación:

Con una probabilidad prefijada se cambia una de las fases de una intersección del individuo saliente del cruce

8 Optimización:

Se continua con la siguiente generación volviendo al paso 2.

3.4 Resultados

Para ver los resultados se realizan tres pruebas sobre el mismo conjunto de datos, los cuales son los siguientes:

Table 1 Configuration of traffic signals for each intersection

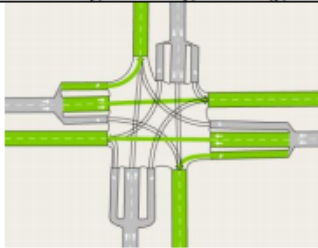
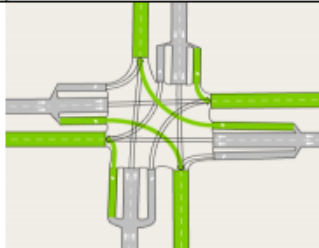

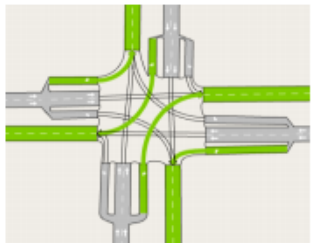
Phase ID	Traffic signal configuration (green movements highlighted)		
1		2	
3		4	

Table 2 Traffic demand

From\To	1	2	3	4	5	6	7	8	Total
1	0	150	150	150	150	100	100	150	950
2	150	0	100	100	100	150	150	100	850
3	150	100	0	150	100	100	100	150	850
4	100	150	100	0	150	100	150	150	900
5	150	100	100	150	0	150	150	100	900
6	100	100	100	100	0	0	150	100	650
7	100	150	750	150	150	100	0	150	1550
8	100	150	150	100	150	100	100	0	850
Total	850	900	1450	900	800	800	900	900	7500

Ilustración 11: Datos pruebas algoritmo de gestión del tráfico

En el primer escenario se ejecuta el algoritmo genético para determinar la combinación de tiempos óptima para esas intersecciones sin estar bajo las condiciones de un accidente de tráfico. Esta encontró un valor de fitness de 22.41, lo cual significa 22,41 vehículos * hora.

El segundo escenario consiste en usar la misma configuración que se ha determinado en la fase anterior, pero habiendo un incidente en la ruta 2 como se muestra en la Ilustración 12. En este caso el fitness se ha empeorado hasta en un 111%, resultando en 47.37 vehículos * hora.

El tercer escenario, es bajo el mismo accidente y configuración que en el escenario 2 aplicar el algoritmo genético, lo cual de un fitness de 28.24 vehículos*hora, lo cual supone una mejora de un 40.76% sobre el escenario 2 demostrando así que este algoritmo es capaz de adaptarse a problemas de tráfico y optimizarlos.

4. Bibliografía

Autores: Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, Koray Kavukcuoglu. (2019) Deep Mind/ London, UK. **Population Based Training of Neural Networks** [Online]. Available: <https://arxiv.org/pdf/1711.09846.pdf>

Autores: Yu-hsin Chen y Matthieu Devin de Waymo y Ali Razavi, Ang Li, Sibon Li, Ola Spyra, Pramod Gupta and Oriol Vinyals de DeepMind. (2019) **How evolutionary selection can train more capable self-driving cars** [Online]. Available: <https://deepmind.com/blog/article/how-evolutionary-selection-can-train-more-capable-self-driving-cars>

Autores: Ang Li, Ola Spyra, Sagi Perel, Valentin Dalibard, Max Jaderberg, Chenjie Gu, David Budden, Tim Harley and Pramod Gupta. (2019) **A generalized framework for population-based training** [Online]. Available: <https://www.youtube.com/watch?v=DIBiW7aasWg>

Autor: Henry AI Lab. (2019) **Population Based Training** [Online]. Available: <https://www.youtube.com/watch?v=pEANQ8uau88>

Autor: Irhum Shafkat. (2019) **So what is Population Based Training?** [Online]. Available: <https://irhum.github.io/2019/08/04/population-based-training.html>

Traffic signal control optimization under severe incident conditions using Genetic Algorithm Autores: Tuo Mao, Adriana-Simona Mihaita, Chen Cai: arXiv:1906.05356 [eess.SY] [Online]. Available: <https://techxplore.com/news/2019-07-genetic-algorithm-traffic-optimization.html>