Autonomous Car Parking

CS7IS2 Project (2021/2022)

Mayuresh Shelke, Ming Jun Lim, Lalu Prasad Lenka and Samridh James

shelkem@tcd.ie, limm5@tcd.ie, lplenka@tcd.ie, jamessa@tcd.ie

Abstract. In the last two decades, autonomous driving has sparked a lot of research interest since it has a lot of potential benefits, such as relieving drivers of tiresome driving and reducing traffic congestion, to name a few. Efficient parking by driverless vehicles remains to be a difficult challenge and an active area of research. In this paper, we survey some of the state-of-the-art algorithms used to build AI agents for autonomous driving tasks and implement them to build an artificial agent that can park a car in a simulated environment. This paper focuses on analyzing the algorithms and discussing their differences. We have examined three different domains of algorithms and thoroughly compared their efficiency and performance robustness in various environment settings. From our experiments, we have found that the reinforcement algorithm like Soft Actor-Critic (SAC) which combines Q-learning and Policy optimization outperforms Neuroevolution and Imitation learning-based algorithms.

Keywords: Autonomous Driving, Autonomous Parking, Reinforcement Learning, Neuroevolution, Imitation Learning, Soft Actor-Critic, Genetic Algorithm, Behavioural Cloning

1 Introduction

The Autonomous Vehicle (AV) industry is one of the most active and promising area in the recent years with many ongoing research to make it fully autonomous [1]. The International Society of Automotive Engineers (SAE) proposed a six-degree autonomy scale with level 0 as no automation and level 5 as full automation without any human interventions.

Car parking is arduous for most of the drivers as it is tedious and often leads to vehicle and pedestrian accidents. It requires the driver to estimate the space required for the car and manoeuvre it into available space by controlling the steering angle and accelerator. While in fact the traffic report statistics survey results found in 12 million traffic accidents, there are about 10,000 traffic accidents occurring in the parking lots with many more number of accidents not reported [2].

Most vehicles are parked 95% throughout the lifetime [3]. Parking is required by every driver and applying AV will greatly improve the quality of life and ease of drivers. It increases driving safety by utilising various sensors to understand the environment surrounding vehicle and park at the designated space avoid collisions due to lack of human experiences [2]. Other potential application for autonomous car parking is saving spaces in car parking lots by parking in an optimal grid resulting in a much efficient parking lot capable of more car [3].

The highway-env github repository[1] contains a collection of autonomous driving and tactical decision-making tasks environment [4]. The parking environment[2] is selected for the project which is a goal-conditioned continuous control task in a given space with a vehicle aiming to reach the destination point. The parking environment uses a grid like layout similar to actual car parks. the vehicle requires input of the steering angle and accelerator. In this paper we have investigated Soft-Actor-Critic (SAC) from the reinforcement learning family, behaviour cloning from the imitation family and genetic algorithm from the neuroevolution family.

The links for presentation is here video and presentation slides.

2 Related Work

The autonomous vehicle control system consists of different steps like navigation system detecting environment, perception module to represent the environment, trajectory planning system generating path for vehicle to reach target, trajectory tracking controller to navigate the vehicle according to the plan [5].

Path planning problem is the main subject of most autonomous vehicle studies. Prior to path planning, the algorithm needs to know the state of the vehicle such as its position and direction. GPS is commonly used to deduce the state by using signals Line Of Sight (LOS) from positioning satellite but not possible in the context of indoor parking lots and high-rise buildings [6]. Instead other data such as vision-based sensors and LiDAR sensors are used to monitor the state of vehicle [7]. Our paper focuses only on path planning algorithm. In last decade there has been a number of algorithms which have been used for path planning in autonomous vehicles like search based algorithms (Dijkstra, A*, D*), Bio inspired algorithms (evolutionary and genetic algorithm), Artificial neural network based algorithms (like Reinforcement learning, Bayesian neural network) etc[8].

Deep Learning (DL) combined with Reinforcement Learning (RL) enables us to build machine learning models which does not require labelled data to learn about a given task. Reinforcement Learning enables an artificial agent to learn optimal strategy from the environment through trial and error [9]. Deep Reinforcement Learning (DRL) is useful when the agent needs to make a decision based on more than one contributing element or high-dimensional input data. Autonomous driving systems constitute of multiple tasks where classical supervised learning methods are no more applicable due its complexity and absence of labelled data. Use of RL or Deep RL applied to autonomous driving or the self driving domain is an emergent field [10].

Soft actor-critic (SAC) is a class of off-policy deep reinforcement learning algorithms, which optimizes the stochastic policy based on the maximum entropy framework. The agent aims to simultaneously maximize expected return and

[1] https://github.com/eleurent/highway-env
[2] id of ”parking-v0”

entropy; that is, to succeed at the task while acting as randomly as possible. Soft actor critic (SAC) [11] is a variation that stabilizes learning by using stochastic policies, entropy regularization, and a few more methods. After reaching the steady-state phase, the maximum entropy framework in SAC may diminish the efficiency of learning outcomes [12]. Hindsight experience replay (HER) is a sample-efficient replay approach for off-policy DRL algorithms that allows the agent to learn from both failures and successes, comparable to humans. HER gives the agent a supplemental reward based on the idea of aim, which increases the efficiency of the learning outcomes even if the objective is not met [12].

Behaviour Cloning (BC) is from the Imitation learning family which uses observations from the expert to learn [13]. In paper [6] trains a Recurrent Convolutional Network for autonomous driving in lane changing context by using BC to learn from human experts driving vehicles. The paper noted the ease of changing architecture when using BC technique and found that spatio-temporal properties worked well in dynamic environments. The huge advantage of BC algorithm is its end-to-end trainable architecture which uses supervised learning to learn policies from an expert by inferring expert's actions [14][15][16][6]. However the BC approach suffers from several limitations such as over-fitting and generalisation issue [14]. BC algorithm is a simple yet effective approach in the project and it can be used to learn the policies created from other algorithms selected in the project.

Evolutionary algorithms are algorithms with ability to evolve in order to learn a task. EA's are based on the concept of survival of the fittest principal of the modern genetic [17]. Genetic algorithm is a population based metaheuristic algorithm belonging to the family of evolutionary algorithms [17]. Genetic algorithm is inspired by Darwin's theory of evolution. Genetic algorithm is used to solve complex real world problems, one of which is autonomous cars modelling. In the paper [18] implements the genetic algorithm with the combination of artificial neural network which is also referred as Neuro-evolution to model vehicle dynamics and train autonomous cars. The paper states that evolutionary algorithms evolve via trial and error method over the iterations and hence provide advantages over the data dependent legacy approaches. It is also observed that the use of neural network with genetic algorithm can reduces the initial phase of generalising features and works without any prerequisite data for model training [18]. One of the notable advantages of genetic algorithm is parallelism. The genomes in the population acts as independent agents, the algorithm is able to explore the search space in all the direction at a time [19].

3 Problem Definition and Algorithm

Deep Reinforcement Learning algorithms like SAC-HER have come a long way to solve arduous problems like parking, and have made it safer and efficient. Thus this approach is used to achieve best outcome, State is a system which has all the information required to know the outcomes of any action. Here parking is a continuous action space in which agent decides and dictates the throttle and the steering angle of the vehicle based on various aspects and its observation like its current position along the 2-D coordinates(x and y), its current velocity (v_x and v_y) and the angle of its wheels. And after each step absolute distance is calculated between the vehicle's current position and the goal state.

The distance is then procured on the weights which then tells how much each element has to be weighed in order to reach or reduce the distance between the two points. This signifies that the agents gains its reward from its absolute distance from the goal while moving in the current direction. Velocity of the vehicle makes sure that the agent is not gaining reward just by standing in a fixed position. Ultimately reward is decided based on velocity and its proximity from the goal state.

Thus vehicle starts with a high negative reward as its away from the goal and static, and then gradually it moves towards the goal and gaining reward by maintaining its velocity and reducing distance obtaining rewards and eventually inching close to 0. The goal state has a reward of 0.12, algorithms tries to get to the long term reward for the agent.

3.1 Soft-Actor-Critic Algorithm

The two types of deep reinforcement learning algorithm widely used these days is Policy Optimization and Deep Q Learning. Deep Q learning algorithms try to learn the optimal Q function $Q(s, a)$ with a function approximator $Q_\theta(s, a)$. It generally uses an objective function based on Bellman equation. To the contrary, Policy optimization algorithms try to directly learn the policy $\pi_\theta(a|s)$ by optimizing parameters theta by using gradient ascent on performance objective $J(\pi_\theta)$.

The primary advantage of policy optimization approaches is that they are principled, in that we explicitly optimize for what we desire. This makes them more stable and dependable. Q-learning approaches, on the other hand, only indirectly improve agent performance by training Q theta to meet a self-consistency equation. Since there are so many failure scenarios in this type of learning, it is less stable. When they do work, however, Q-learning approaches have the benefit of being far more sample efficient than policy optimization techniques since they can reuse data more efficiently [20].

There are a number of algorithms that exist on this spectrum and are capable of carefully balancing the strengths and limitations of each side. Sample efficiency is a problem for some of the most effective RL algorithms in recent years, such as trust region policy optimization (TRPO), proximal policy optimization (PPO), and asynchronous actor-critic agents (A3C) [11].

This is because these strategies allow for on-policy learning and necessitate new samples following each policy modification. Using experience replay buffers, Q-learning based off-policy algorithms like DDPG learn efficiently from past samples. However, such approaches are hyper-parameter sensitive and require a great deal of adjustment to converge. SAC is an alternate strategy for accelerating convergence.

HER is a sample-efficient replay method that improves off-policy DRL algorithm performance by allowing the agent to learn from both successes and failures. HER is applied to SAC and propose to help SAC learn more effectively.

3.2 Behaviour Cloning Algorithm

Behaviour Cloning (BC) generates trajectory by learning policy from direct mapping of states to actions without recovering reward function of the expert [21].

$$\tau^d = \pi(s) \tag{1}$$

$$\upsilon_t = \pi(x_t) \tag{2}$$

Equation (1) is the general aim of BC algorithm to generate trajectory of agent by learning policy for a given state. The goal of BC for the project car parking in an action-state space can be formulated as equation (2) to generate control input υ_t using policy π given the current state x_t. The driving task for BC can be treated as a supervised learning problem [16][21][6].

The abstract of BC algorithm described in [21] requires a dataset of trajectories from the expert $\mathcal{D}$, policy representation π_θ and objective function $\mathcal{L}$. The result is an optimised policy parameters θ trained using the dataset $\mathcal{D}$ by optimising the objective function $\mathcal{L}$.

$$\mathcal{D} = \{(\upsilon_i^*, s_i^*)\}_{i=1}^{M} = d^{\pi^*} \tag{3}$$

Equation (3) describes the dataset represented by a set of control input (υ_i) and state (s_i) pair from the trajectories of the expert. The policy of the expert is used as the ground truth reward and assumed to be near the optimal policy π^* .

$$\ell(x^L, x^{demo}) = (x^L - x^{demo})^T (x^L - x^{demo}) \tag{4}$$

The quadratic loss function also known as ℓ_2-loss or least squares (LS) in equation (4) is commonly used as an objective function and most optimiser libraries include additional regularisation term. The x terms are vectors and loss calculates the action from agent's policy x^L against expert's action x^{demo}. Other loss functions can be used as the object function such as ℓ_1-loss, cross entropy loss, hinge loss, etc.

Algorithm 1 Behaviour Cloning algorithm

```
1: expert_trajectory $\mathcal{D}$ ← sample from expert's policy from environment
2: agent_policy $\pi_\theta$ ← initialise a random policy parameters $\theta$
3: train agent_policy $\theta$ dataset expert_trajectory $\mathcal{D}$
4: save optimised agent_policy $\theta$
```

The algorithm (1) is a high-level pseudo-code of the behaviour cloning approach. Line *3* uses supervised learning algorithms with hyper-parameters such as objective function $\mathcal{L}$, optimiser, network architecture, etc.

3.3 Evolutionary Algorithm

EAs are a family of algorithms which are inspired by biological process of natural selection and Darwin's evolution theory. Neuro-evolutionary algorithms are the combination of neural network and genetic algorithm []. Each genome in this algorithm is a neural network. Weights and biases of the neural network are initialized randomly. Each takes the observation from environment as an input and return required actions. Based on these actions fitness score is calculated and returned by the network. The goal of neural network is to optimize the fitness score. Genetic algorithm works based on five stages: Population, Fitness evaluation, Selection, Crossover, Mutation [22][23][24].

Population: Population is set of solution candidates to the problem we are dealing with. Here each genome is a neural network whose weights and biases are randomly initialized.

Fitness evaluation: Fitness function calculates a fitness score for each individual or genome which is the measure of difference between desired and current state. Each neural network returns a fitness score which is used in selection to generate future individuals.

Selection: Selection is the process of selecting neural networks based on their fitness score as parents for future generations. One of the most common approaches is ranked selection where individuals are order in descending order of their fitness score and those with higher fitness score are selected as parents.

Crossover: Crossover is the process where weights and biases of two neural networks are exchanged with each other. Percentage of genes to be exchanged depends on crossover rate. New individuals which are referred as offspring are created until the crossover point is reached. These new individuals are further added to the population.

Mutation: Mutation is the process where some of the weights and biases of the neural network are flipped based on low probability. Mutation ensures that the population has diversity in population and further increases the search space for the algorithm.

Although, both genetic algorithm and neural networks largely depend on the choice of parameters such as population size, cross-over and mutation rate, number of network layers. Hence wrong choices can lead to a situation where algorithm may not converge.

4 Experimental Results

4.1 Methodology

SAC-HER and imitation algorithms were tested for three episodes and genetic algorithm for one episode. Initial and goal state for SAC-HER and imitation algorithms were different for each episode where as these two were kept same for genetic algorithm. The performance of each algorithm is evaluated based on below evaluation matrices:

- Cumulative reward across all the episodes
- Average velocity across all the episodes

Algorithm 2 Genetic Algorithm pseudocode [25]

```
1: Random population initialization
2: Fitness evaluation of the population individuals
3: Repeat
    4: Select neural networks with high fitness score as parents
    5: Generate new individuals by crossover process
    6: Perform mutation on the newly created individuals
    7: Fitness evaluation of the newly created individuals
    8: Replace old worst individuals with new best individuals
9: Stop until termination criteria
```

4.2 Results

Figures 1,2 shows the rewards and velocity with respect to number of steps taken to reach the goal state during first episode. Reward is a distance metric which is calculated as distance difference between current location of the vehicle and the goal state [parking spot] location. Hence the reward starts with high negative value and decreases towards zero as the vehicle moves closer to the parking spot. Table 1 below shows the average velocity achieved by the vehicle

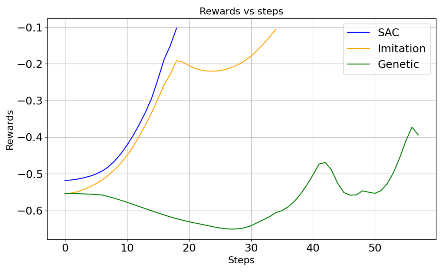


Fig. 1. Rewards vs no. of steps

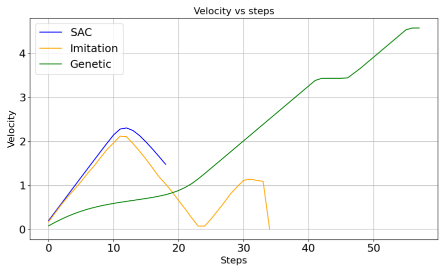
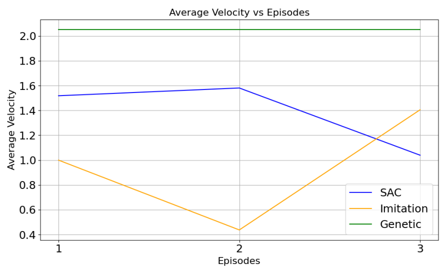


Fig. 2. Velocity vs no. of steps

per episode. The environment returns velocities in x (v_x) and y (v_y) direction. Average velocity is calculated using formula $v_x^2 + v_y^2$.

	After 5 runs	After 10 runs	After 20 runs
SAC	1.34	1.442	1.466
Imitation	0.95	0.89	0.94
Genetic	2.64	2.41	2.49

Table 1. Average velocities after n runs



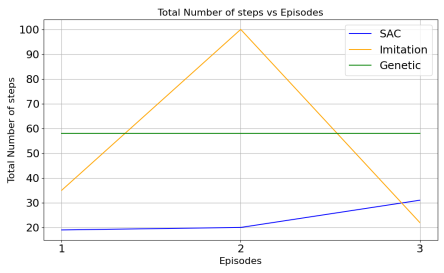


Fig. 3. Avg Velocity vs Episodes **Fig. 4.** No. of steps vs Episodes

	After 5 runs	After 10 runs	After 20 runs
SAC	-8.93	-8.375	-8.25
Imitation	-16.74	-18.269	-15.93
Genetic	-24.363	-24.042	-26.04

Table 2. Average Cumulative Rewards after n runs

4.3 Results Analysis

Reward achieved with respect to steps: In figure (1) We can see that the rewards for all three algorithms are gradually decreasing with steps taken. Both SAC and imitation algorithm achieves lowest reward of -0.1 as the run progresses whereas genetic algorithm achieves a lowest reward of -0.3 at the end of the run. SAC and imitation algorithm clearly outperforms genetic algorithm and are able to perform the task efficiently.
Velocity vs no. of steps: In figure (2) we can see that the velocity of the vehicle decreases at the end of run for SAC and imitation algorithm indicating the vehicle is coming to halt once it reaches the parking spot. The fluctuations in the velocity over steps also shows that vehicle is able to adjust its throttle during the travel path. Genetic algorithm shows a contradictory behavior where velocity is not decreasing at the end of the run.
No. of steps per episode: Figure (4) shows that SAC takes minimum number of steps to converge. Imitation algorithm takes highest number of steps in one of the episode whereas converges within average of 30 steps in other episodes. Genetic algorithm takes 65 steps on an average to converge.
Average velocity after N runs: In Table (1) and Figure (2) Average velocity

	After 5 runs	After 10 runs	After 20 runs
SAC	25.4	22.9	22.3
Imitation	54.8	61.4	52.6
Genetic	62.4	67	68.6

Table 3. Average of total no of steps to reach the goal

for SAC and imitation algorithm is ranging between 0.95 and 1.5 whereas for genetic algorithm it is higher. Average velocity gives the notion of how the vehicle adjusts the throttle based on the path followed towards parking spot. SAC and Imitation maintain their acceleration well whereas genetic completes or reaches towards the goal with a lot higher speed.
Average Cumulative Reward after N runs: In table (2) Cumulative rewards for SAC is the best, and for imitation and genetic the rewards are low. The reason behind imitation having such low rewards after n runs as it got stuck in few of the cases as the vehicle approached near the goal, and thus did not perform so well.
Average of total number of steps to reach the goal after N runs: In table (3) Again SAC reaches the goal with minimum steps whereas imitation and genetic took a lot of steps to reach the goal. For imitation, it was not consistent in reaching the goal quickly as it got stuck when its close to the goal.
SAC being reinforcement learning algorithm shows the signs of adapting very well to the environment and achieving the goal consistently over the episodes. Imitation algorithm uses SAC model as an expert resulting in similar behavior as that of SAC algorithm.

5 Conclusions

Soft actor critic with hindsight experience replay significantly outperformed the imitation learning and neuroevolution algorithm in terms of cumulative rewards and average velocity. It can be seen from the table (1) that the SAC agent paces its velocity well, it accelerates gradually towards the goal and then carefully reduces its speed and reaches the goal. From the simulation it was evident that SAC had no abrupt movements or sudden change in velocity. SAC right from the start steers in the right direction, even accelerating backwards when needed and traversed in the shortest path possible.

Imitation learning agent did well at times and performed as similar to SAC, but few times it gets stuck when its close to the goal. Its inconsistency can be observed in (seen in table (1)) which is because it gets stuck near goal with to and fro motion and takes more time to reach the goal.

Neuroevolution algorithm on the other hand often takes a longer route in reaching the goal, and it is also evident from the simulation that instead of straight shortest path it takes elliptical paths towards the goal, and once it gains velocity it keeps on accelerating which is evident from table (1). Thus it can be seen that SAC performs the best in all terms like shortest distance, velocity and time taken to reach the goal(3).

The performance of neuroevolution algorithm can be further improved by optimizing the hyperparamters of genetic algorithm and neural network. One major learning from this project is that we observed that deep reinforcement learning algorithms perform better for autonomous agents. We also observed that there is lack of literature for evolutionary algorithm in autonomous driving domain.

References

1. Talavera E, Díaz-Álvarez A, Naranjo JE, Olaverri-Monreal C. Autonomous Vehicles Technological Trends. Multidisciplinary Digital Publishing Institute; 2021.
2. Wang W, Song Y, Zhang J, Deng H. Automatic parking of vehicles: A review of literatures. International Journal of Automotive Technology. 2014;15(6):967-78.
3. Choi CQ. How Self-Driving Cars Might Transform City Parking. IEEE Spectrum Febrero. 2019;20.
4. Leurent E. An Environment for Autonomous Driving Decision-Making. GitHub; 2018. https://github.com/eleurent/highway-env.
5. Lin L, Zhu JJ. Path planning for autonomous car parking. In: Dynamic Systems and Control Conference. vol. 51913. American Society of Mechanical Engineers; 2018. p. V003T32A017.
6. Saksena SK, Navaneethkrishnan B, Hegde S, Raja P, Vishwanath RM. Towards behavioural cloning for autonomous driving. In: 2019 Third IEEE International Conference on Robotic Computing (IRC). IEEE; 2019. p. 560-7.
7. Chan TK, Chin CS. Review of autonomous intelligent vehicles for urban driving and parking. Electronics. 2021;10(9):1021.
8. Abdallaoui S, Aglzim EH, Chaibet A, Kribèche A. Thorough Review Analysis of Safe Control of Autonomous Vehicles: Path Planning and Navigation Techniques. Energies. 2022;15(4). Available from: https://www.mdpi.com/1996-1073/15/4/1358.
9. Grigorescu S, Trasnea B, Cocias T, Macesanu G. A survey of Deep Learning techniques for autonomous driving. Journal of Field Robotics. 2020;37(3):362–386.
10. Kiran BR, Sobh I, Talpaert V, Mannion P, Al Sallab AA, Yogamani S, et al. Deep reinforcement learning for autonomous driving: A survey. IEEE Transactions on Intelligent Transportation Systems. 2021.
11. Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International conference on machine learning. PMLR; 2018. p. 1861-70.
12. Lee MH, Moon J. Deep Reinforcement Learning-based UAV Navigation and Control: A Soft Actor-Critic with Hindsight Experience Replay Approach. arXiv; 2021. Available from: https://arxiv.org/abs/2106.01016.
13. Torabi F, Warnell G, Stone P. Behavioral cloning from observation. arXiv preprint arXiv:180501954. 2018.
14. Codevilla F, Santana E, López AM, Gaidon A. Exploring the limitations of behavior cloning for autonomous driving. In: Proceedings of the IEEE/CVF International Conference on Computer Vision; 2019. p. 9329-38.
15. Farag W, Saleh Z. Behavior cloning for autonomous driving using convolutional neural networks. In: 2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT). IEEE; 2018. p. 1-7.
16. Ly AO, Akhloufi M. Learning to drive by imitation: An overview of deep behavior cloning methods. IEEE Transactions on Intelligent Vehicles. 2020;6(2):195-209.
17. Yu X, Gen M. Introduction to evolutionary algorithms. Springer London; 2013.
18. Katoch S, Chauhan SS, Kumar V. A review on Genetic Algorithm: Past, present, and future. Multimedia Tools and Applications. 2020;80(5):8091–8126.
19. G S, S V, S S, Suganya G. Application of Neuroevolution in Autonomous Cars. arXiv; 2020. Available from: https://arxiv.org/abs/2006.15175.
20. Russell SJ, Norvig P. Artificial intelligence: a modern approach. Malaysia. Pearson Education Limited London, UK:; 2016.

21. Osa T, Pajarinen J, Neumann G, Bagnell JA, Abbeel P, Peters J. An algorithmic perspective on imitation learning. arXiv preprint arXiv:181106711. 2018.
22. Xin-She Y. Nature-Inspired Optimization Algorithms (Second Edition). Academic Press; 2021.
23. Sloss AN, Gustafson S. 2019 Evolutionary Algorithms Review. arXiv; 2019. Available from: https://arxiv.org/abs/1906.08870.
24. Simon D. Evolutionary optimization algorithms : biologically-Inspired and population-based approaches to computer intelligence; 2013. .
25. ENSI MZ, Zouita M, Ensi, ENSI SB, Bouamama S, CNAM KB, et al.. Improving genetic algorithm using arc consistency technic; 1970. Available from: https://dl.acm.org/doi/10.1016/j.procs.2019.09.309.