Avrupa Bilim ve Teknoloji Dergisi
Sayı 51, S. 162-172, Ağustos 2023

Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
No. 51, pp. 162-172, August 2023

Research Article

Comparison of Optimization Techniques for Delay Minimization in Signalized Intersections: PSO vs GA

Abdullah Karadag [1*], Murat Ergün[2]

[1*] Istanbul Technical University, Faculty of Civil Engineering, Departmant of Transportation, İstanbul, Turkey, (ORCID: 0000-0002-3886-5961), karadaga@itu.edu.tr
[2] Istanbul Technical University, Faculty of Civil Engineering, Departmant of Transportation, İstanbul, Turkey, (ORCID: 0000-0002-2489-7858), ergunmur@itu.edu.tr

(İlk Geliş Tarihi 27 Mart 2023 ve Kabul Tarihi 23 Mayıs 2023)

(**DOI:** 10.31590/ejosat.1270905)

ATIF/REFERENCE: Karadağ, A., Ergün, M. (2023). Comparison of Optimization Techniques for Delay Minimization in Signalized Intersections: PSO vs GA. *Avrupa Bilim ve Teknoloji Dergisi*, (51), 162-172.

Öz

Kavşak gecikmelerini en aza indirmek günümüzün akıllı şehirlerinde önemli bir zorluktur. Gecikme minimizasyonu için farklı yaklaşımlar olsa da bunların çoğu Karayolu Kapasite El Kitabı (ABD) tarafından tanımlanan aynı doğrusal olmayan gecikme formülünü kullanır. Sonuç olarak, gecikme çıktısını en aza indiren optimum faz sürelerini bulmak için hızlı ve hassas bir algoritma seçmek kritik bir karardır. Bu makalede, kavşaklardaki kişi gecikmelerini en aza indirmek için yenilikçi bir sistem geliştirme çalışmamızın bir parçası olarak en iyi optimizasyon algoritmasının seçimindeki deneyimimizi paylaşıyoruz. En iyi bilinen iki algoritmayı karşılaştırdık: Genetik Algoritmalar (GA) ve Parçacık Sürü Optimizasyonu (PSO). Aynı popülasyon boyutu ve iterasyon sayısı kullanıldığında PSO'nun GA'dan 7 kat daha hızlı ve 17 kat daha hassas olduğu gösterilmiştir.

Anahtar Kelimeler: Uyarlanabilir trafik sinyal kontrolü, Genetik algoritmalar, Parçacık sürü optimizasyonu, Kavşak gecikme minimizasyonu, Gerçek zamanlı optimizasyon.

Comparison of Optimization Techniques for Delay Minimization in Signalized Intersections: PSO vs GA

Abstract

Minimizing intersection delays is an important challenge in today's smart cities. Even there are different approaches for delay minimization most of them uses the same nonlinear delay formula defined by Highway Capacity Manual (US). As a result choosing a fast and precise algorithm for finding the optimal phase times minimizing the delay output is a critical decision. In this paper we share our experience in selection of best optimization algorithm as a part of our work of developing an innovative system to minimize person delays in intersections. We compared two best known algorithms: Genetic Algorithms (GA) and Particle Swarm Optimization (PSO). It is shown that using the same population size and number of iterations PSO is 7x faster and 17x more precise than GA.

Keywords: Adaptive traffic signal control, Genetic algorithms, Particle swarm optimization, Intersection delay minimization, Real-time optimization.

[*] Corresponding Author: karadaga@itu.edu.tr

1. Giriş

Heuristic optimization algorithms play an important role in finding optimum solutions to global optimization problems. When genetic algorithms (GA) were first developed, they quickly gained popularity for optimizing control system parameters that are complex and challenging to solve using traditional optimization techniques. Recently another method, Particle Swarm Optimization (PSO, which was originally used for modeling a social phenomenon, behavior of fish and bird flocks (Kennedy & Eberhart, 1995)) have become a popular alternative to GA and attracted many researchers among various other optimization techniques. Both techniques are intended to discover a solution to a given aim. Each has its own concepts and approaches but most of the time are used for similar purposes, such as minimizing non-linear functions.

Vehicle delay in signalized intersections is one of that non-linear problems. Many studies on signalized intersections and vehicle delays have used heuristic algorithms. When these studies are examined, it is seen that heuristic algorithms offer effective solutions for such studies (Akgüngör vd., 2019; Çakici & Murat, 2021). Traffic control systems are required to find optimum green times for estimated traffic volumes for each direction of an intersection. As traffic volumes vary every cycle the control system need to estimate next traffic volumes and calculate new optimum green times and cycle length. Such a system if it has a central architecture then maybe thousands of intersections (for example in Istanbul city there are more than 2000 intersections) are served by one or a few computers. So it must use as less as possible computational power and should find an optimum solution as fast as possible. If the system is going to be embedded at intersection controller than its computing capacity, speed and power consumption is requested to be kept as low as possible. As a result selection of fastest and reliable optimization technique matters.

Because of the similarities between the two algorithms, and because both of them are used for similar purposes, some researchers have compared these algorithms. Eberhart at al. (Eberhart & Shi, 1998), Eberhart is one of the researchers who first proposed the PSO, tried to clarify the differences between the two methods in general terms. According to Eberhart, analogies can be established in terms of some concepts between two algorithms. In GA, the solution candidates are called individuals and the whole individuals are called populations. In PSO, the solution candidates are called particles, while the whole set of particles is called swarm. The most important difference emphasized by Eberhart is that the best chromosomes in the GA are carried to the next generation, while in PSO all the particles are carried to the next iteration. These similarities and differences do not lead us to a definite judgment on the superiority of the two algorithms. It is natural to have different results in different conditions and in the comparisons made for different problem spaces. For example, Panda et al. (Panda & Padhy, 2007) made a comparison of their problem spaces in the field of TCSC-based Controller Design.

Studies in the literature were reviewed and the results of similar studies using heuristic algorithms are shown in Table I. The results of the design made in these studies for all vehicles were examined. Since all of the studies were conducted for different cities and different intersections, the comparison between them will not be very healthy. For this reason, each study should be evaluated with the benefits it brings for its own application area(Shaikh vd., 2022).

Table I. Average Successes Achieved with Different Heuristic Algorithms

Heuristic Algorithm	Avg. Vehicle Delay Reduction Rate	Reference
PSO	41%	(Dong vd., 2010)
GA	34,57%	(Li vd., 2017)
DEA	28-30%	(Cakici & Murat, 2019)
ACO	25%	(Peñabaena-Niebles vd., 2017)
HS	25%	(Zhang vd., 2019)

In this paper we share our experience in selection of best optimization algorithm as a part of our work of developing an innovative system to minimize person delays in intersections. We compared two best known algorithms GA and PSO in problem space of intersection vehicle delay optimization.

2. Test Method

To compare the two algorithms, a simple virtual intersection with four arms and two movements is used, shown in Figure 1. At virtual intersection initial hourly traffic (traffic volume) for each movement is 400 veh/h. As a reference value the optimum green times for each movement, and optimum cycle length is calculated by a brut-force trial-error method which tries all integer cycle length values between minimum and maximum cycle lengths (The trial-error method limitations are given in Table II.). Then GA and PSO optimization algorithms are run to find optimum green times. Inputs of both GA and PSO algorithms are movements' hourly traffic volumes. Fitness function of both GA and PSO algorithms is HCM delay function. The green times, error values, and processing times

for varying number of iterations (50, 100, 1000, 2500, 5000, 10000) and population sizes (10, 20, 50, 100) are recorded. The traffic volumes are varied from 400 veh/h to 8000 veh/h in one arm and from 400 veh/h to 5200 veh/h in the other arm with increase of 400 vehicles in each step. A sample run result is given in Table III.

Table II. Limitations for Methods

Key	Value
Minimum green time	6 s
Maximum green time	134 s
Lost time during green transation	5+5=10 s
Maximum cycle time	150 s

Table III. Sample Comparasion

Algorithm	V1	V2	Error	T1	T2	Delay
Trial-Error Reference	400	800		8	12	15.67
GA	400	800	0.00	7.99	12.00	15.66
PSO	400	800	0.06	7.56	12.44	15.68

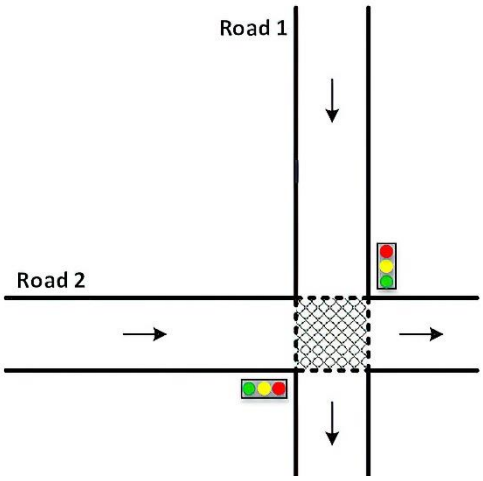

Figure I. Intersection model used in the study

For the same counts, the GA converged to (7.99, 12.00) s phase times and 15.66 s delay and the PSO converged to (7.56, 12.44) s phase times and 15.68 s delay. As the delay from the GA is less than the delay with trial and error, the error of GA is assumed to be zero. Because the trial error method uses only integer values, it is possible for algorithms to obtain better delay results with floating point numbers. However, these differences do not have a significant effect in practice. Because the integer green times are usually applied in real world intersection controllers. While determining the alternatives to be tried, the limitations in Table II. are used.

GA default parameters are shown Table IV. In Table III. a sample comparison is shown. In Table III; V1 and V2 are hourly traffic volumes for 1st and 2nd arms of intersection respectively. T1 and T2 are green times for 1st and 2nd movements of intersection respectively, in seconds. Error for an algorithm is the percentage of the positive shifts of delays from reference delay values. Delay is the average vehicle delay for an movements as formulated in Section 5.

Table IV. Parameters of GA

Elitism	True
Mutation rate	0.215
Uniform rate	0.5

3. Genetic Algorithms

Genetic algorithms are one of the well-known general purpose heuristic optimization techniques inspired by biological evolution. Different individuals (candidate solutions) constitutes a population. Chromosomes of individuals are modified at each step to converge the final fittest solution. The fittest individuals in every step are selected to be parents of next step individuals (next generation). The best individuals are selected based on a fitness function. The fitness score itself is not the only parameter for selecting parents but it

increases probability of selection of an individual. Selection of parents is a stochastic process. Crossover and mutation operators of a biological process are also valid for genetic algorithms. In Figure II, the computational flowchart for GA is shown (Panda & Padhy, 2007).

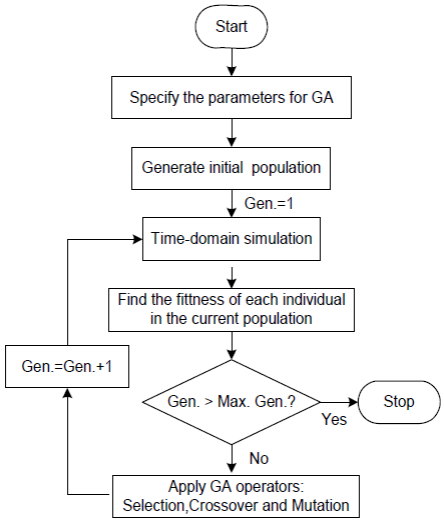

Figure II. Flowchart of genetic algorithms

4. PSO Algorithm

Particle Swarm Optimization is an optimization method draws inspiration from swarms of birds and fish. It was proposed by J. Kennedy and R.C. Eberhart in 1995 (Kennedy & Eberhart, 1995). PSO has successfully solved many optimization problems such as function optimization, neural network training, parameter adjustment in automatic control (Erdoğmuş, 2010). PSO is an improved method for optimization problems inspired by the behavior of fish, birds and some animals. When fish, birds, and other social animals were watched in groups, it was found that they interacted with one another while in search of food. It was found that when someone found food, the others changed the direction in which they found food and they updated their speed accordingly. This social interaction is modeled with PSO. PSO is an evolutionary algorithm as Genetic Algorithms (GA). In GA, all current solutions are called populations, while in PSO they are called swarm. PSO is an evolutionary algorithm similar to GA, but it is simpler because it lacks operators like crossing and mutation, and it converges to the optimal solution more quickly. Each bird in the swarm is referred to as a particle. Each particle in the PSO is a potential solution. For example, the calculation of the function of a problem with the D variables (dimension) is an initial solution of the equation with n particles for calculating the Equation I (Erdoğmuş & Yalçın, 2015).

$$\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{nD} \end{bmatrix} \qquad \text{I}$$

The function for the particle i is given in Equation II below.

$$f_{fitness} = f(x_{i1}, x_{i2}, x_{i3}, \dots, x_{iD}) \qquad \text{II}$$

Each line called a particle in the above matrix refers to a solution. n solution for n particles is obtained. In the research space, particles move based on two key variables. The best of each individual (pbest) and the best of all individuals (gbest) are obtained in each iteration. There is only one (gbest) in each iteration with a PSO. The initial value for each particle at the beginning of the algorithm is also the value of pbest. The other particles' positions are updated in succeeding iterations, and the current position is compared to the previous pbest. The new solution is now designated as pbest if it is superior to pbest.

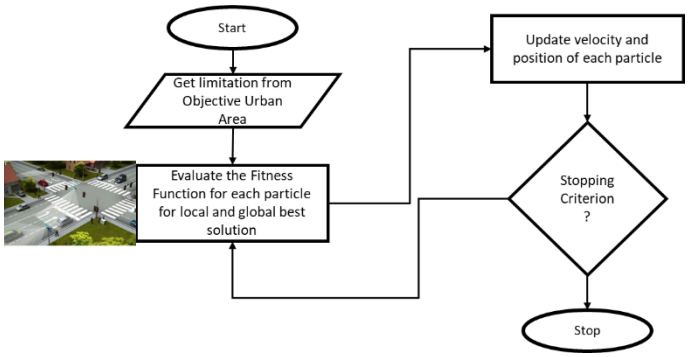

Figure III. Flowchart for PSO algorithm

In PSO, the swarm begins with a preliminary resolution. Each particle starts out with a speed and location value. The variable values of the particle positions fall within the specified range. The particles move by modifying their velocities for both the swarm and themselves. Each iteration of the particles brings them closer to the optimal solution. In Figure III, the computational flowchart for PSO is shown Equations IV and IV provide the velocity and position formulas for the ith particle (Clerc, 1999; Parrott & Li, 2006).

The impact of the limitation factor K is to dampen the oscillation in the particle's motion. As a result, over time, the particles move closer to the solution. $\varphi_1$and $\varphi_2$are social and cognitive parameters. A random number (0–1) is chosen from a random distribution as rand. Equation IV is used to determine the position after calculating the speed.

$$v_i^{k+1} = K(v_i^k + \varphi_1 rand()\left(p_{best\ i}^k - x_i^k\right) + \varphi_2 rand()\left(g_{best} - x_i^k\right)) \qquad \text{III}$$

where

$k =$ the number of iterations
$v_i^k = speed\ of\ particle\ i\ for\ iteration\ k$
$v_i^{k+1} = speed\ of\ particle\ i\ for\ iteration\ k+1$
$K, \varphi_1, \varphi_2 = dependent\ coefficients$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \qquad \text{IV}$$

Both φ_1 and φ_2 allow the particle to move (cognitively) in accordance with its own experience and the experiences of other particles in the swarm(social), respectively.

PSO's particle number, n, can range from 20 to 40. If a more accurate measurement is needed, the number of particles can be increased. Number of variables and the value range of the variables vary depending on the problem. φ_1 ve φ_2 are usually set to values close to 2. If 2 and 2.1 values are taken respectively, *K* constant is used as 0.7298 (Parrott & Li, 2006).

5. Highway Capacity Manual (HCM-2000) Delay Calculation

The calculations in this section (Section 5) are taken from Highway Capacity Manual (HCM-2000). Equation V is used to compute the control delay. To calculate uniform delay and incremental delay, Equation VII and Equation VIII can be used respectively.

$$d = d_1(PF) + d_2 + d_3 \qquad \text{V}$$

where

d = Control delay (s/veh),

d1 = Uniform delay (s/veh),

PF =progression adjustment factor,

d2 =incremental delay(s/veh),

d3 = initial queue delay(s/veh).

5.1. Progression Adjustment Factor

Progression primarily affects uniform delay; for this reason, the adjustment is applied only to d1. Equation VI can be used to determine the value of PF.

$$PF = \frac{(1-P)f_{PA}}{1-\frac{g}{c}} \qquad \text{VI}$$

where

PF = progression adjustment factor,

P = percentage of all vehicles that arrive during green,

g/c = effective green-time ratio, and

fPA = additional adjusting coefficient for platoon arrival during the green.

For a lane group, uniform delay is represented by the v/c ratio (x). Equation VII estimates the control delay under the conditions of perfectly uniform arrivals and a steady flow. When calculating d_1, x values greater than 1.0 are not taken into account.

$$d_1 = \frac{0{,}5c(1-g/c)^2}{1-[\min(1,\ x)\frac{g}{c}]} \qquad \text{VII}$$

where

d1 = uniform delay (s/veh);

x= the lane group's volume to capacity (V/C) ratio;

c = cycle length (s);

g = effective green time for lane group (s);

5.2. Incremental Delay

Equation VIII calculates the incremental delay brought on by irregular arrivals and individual cycle failures as well as the delay brought on by extended oversaturation periods. The equation presumes that no initial queue exists and that the whole demand flow has been serviced during the prior analysis period. As a result d_3 is assumed to be zero.

$$d_2 = 900T[(x-1)+\sqrt{(x-1)^2+\frac{8kIx}{CT}}] \qquad \text{VIII}$$

where

C = lane group's capacity (veh/h);

k = adjustment of the actuated control's incremental delay;

I = incremental delay modification for upstream signals' filtering or metering;

x= the lane group's volume to capacity (v/c) ratio; and

T = the length of the analysis period (h).

Since the k coefficient in Equation VIII is related to the effect of the controller on the delay, it is taken as 0.5 for pretimed intersections.

The impacts of filtered arrivals from upstream signals are taken into account by the incremental delay adjustment factor I in Equation 8. An I-value of 1.0 is used for an isolated intersection.

6. Results & Findings

6.1. PSO Results

In Table V. the results for PSO algorithm are shown. Population (swarm) size is varied from 10 to 100. For every swarm size number of iterations is varied from 50 to 10000. The change of error depending on the number of particles and number of iterations is shown graphically in Figure IV. A swarm size of 20 is more than enough for achieving a reasonable error rate. It should be noted that most intersection controllers most of the time uses integer values for green times. So a float-integer conversion already causes an error change at level of 1%. So 1% error means practically zero error.

Table V. PSO Results

No	Population Size	Number Of Iterations	Error	Proccessing Time
1	10	50	2.98	0.02
2	10	100	0.92	0.04
3	10	1000	0.19	0.37
4	10	2500	0.03	0.92
5	10	5000	0.05	1.66
6	10	7500	0.04	2.47
7	10	10000	0.03	3.30
8	20	50	0.34	0.04
9	20	100	0.06	0.10
10	20	1000	0.02	0.73
11	20	2500	0.01	1.84
12	20	5000	0.00	3.31
13	20	7500	0.01	5.06
14	20	10000	0.00	6.44
15	50	50	0.07	0.09
16	50	100	0.03	0.18
17	50	1000	0.00	1.76
18	50	2500	0.00	4.36
19	50	5000	0.00	8.41
20	50	7500	0.00	12.51
21	50	10000	0.00	16.44
22	100	50	0.02	0.19
23	100	100	0.02	0.36
24	100	1000	0.00	3.40
25	100	2500	0.00	8.75
26	100	5000	0.00	17.96
27	100	7500	0.00	25.11
28	100	10000	0.00	33.16

As seen from Figure V PSO processing time increases as either swarm size or number of iterations increases. The relation is nearly linear.

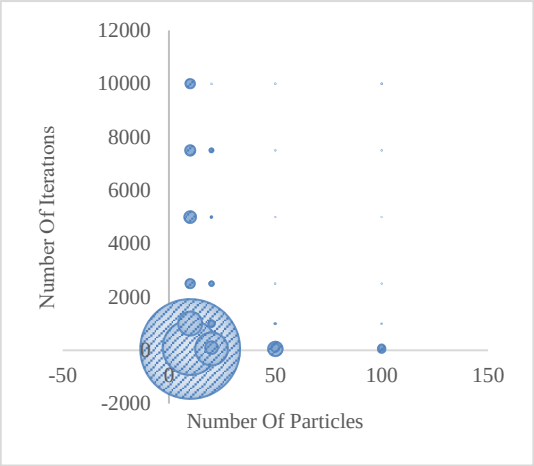

Figure IV. PSO Error Change

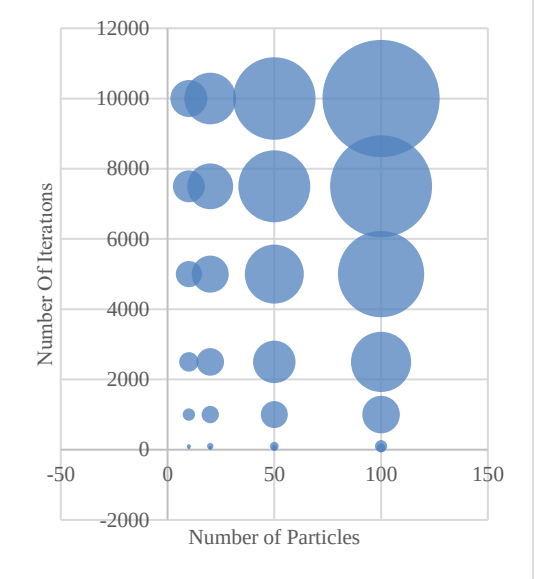

Figure V. PSO Proccessing Times

6.2. GA Results

In Table VI the results for GA algorithm are shown. Population size (swarm size) is varied from 10 to 100. For every swarm size number of iterations is varied from 50 to 1000, unlike PSO where upper value was 10000. That is because GA network becomes too slow when number of iterations are more than 1000. So the run has to be halted after 1000 iterations. The change of error depending on the number of particles is shown graphically in Figure VI. At least a population size of 1000 is required for achieving a reasonable error rate.

As seen from Figure VII GA processing time increases dramatically as either swarm size or number of iterations increases. The processing times after 1000 iterations are very large to even test.

Table VI. GA Results

No	Population Size	Number Of Iterations	Error	Processing Time
1	10	50	3.4748	0.1576
2	10	100	1.7959	0.3195
3	10	1000	0.2418	3.1942
4	20	50	1.5358	0.3357
5	20	100	1.0595	0.6758
6	20	1000	0.1056	8.1324
7	50	50	0.4573	0.9452
8	50	100	0.2829	1.7073
9	50	1000	0.0454	18.1872

10	100	50	0.2044	1.8087
11	100	100	0.0856	3.7515
12	100	1000	0.0154	35.7794

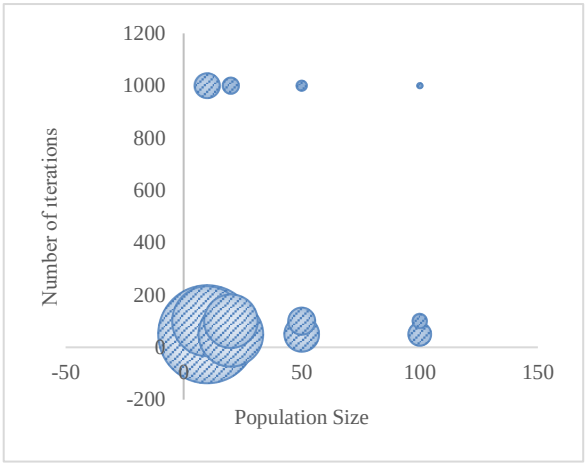

Figure VI. GA Error Change

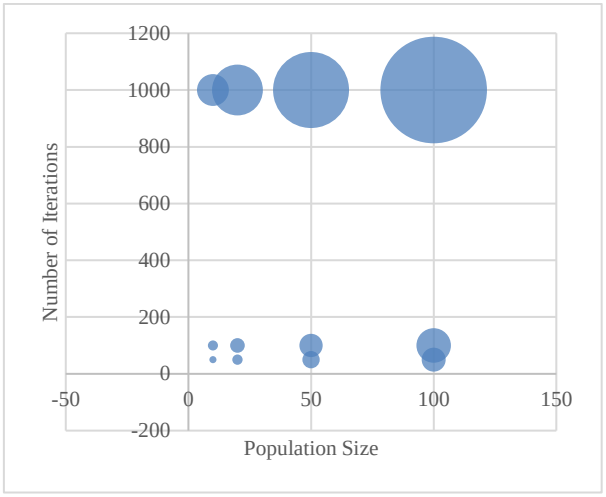

Figure VII. GA Proccessing Times

In Figure VIII and Figure IX, processing times and the error are compared with the bubble method to better understand the difference between PSO and GA, respectively. In this notation, the x-axis represents the number of iterations and the y-axis represents the population size. The bubbles' radiuses show the magnitude of the error and processing times, respectively.

Figure VIII. PSO and GA processing time comparison

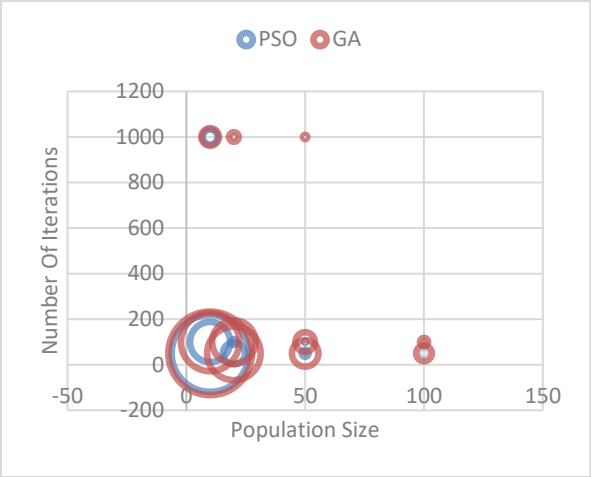

Figure IX. PSO and GA error comparison

7. Conclusion

Depending on the results it is possible to make the following determinations:

- Performance differences are achieved, but both PSO and GA can find optimal green times to achieve minimum delay values.
- For the same number of iterations (100) the PSO algorithm runs 6.8 times faster than GA.
- While the population size (20) and the number of iterations (100) are fixed, the PSO algorithm reaches an error value 17.6 times smaller than GA.
- GA can reach an acceptable error rate (1.5%) only when the population size is 100 and the number of iterations is 1000. However, to achieve this 1000th iteration 35.7 s is required. Considering that the tests are done using a Python programming language on a laptop with an i7 processor, it is understood that the codes for a real-time system should be rewritten with more system-level languages such as C / C ++.
- When the population size is 50 and the number of iterations is 1000, the PSO reaches almost zero error. Moreover, with the same hardware, only 1.76 s is sufficient for the PSO to reach the 1000th iteration. A real-time system can be managed with the PSO almost without the need for additional hardware or software enhancement.

Finally it can easily be seen that increasing the number of iterations in genetic algorithms extends the processing time considerably. Therefore, the use of the PSO algorithm for the delay in traffic optimization is more suitable for both the processing time and the error performance.

References

Akgüngör, A., Yılmaz, Ö., Korkmaz, E., & Doğan, E. (2019). Meta-Sezgisel Yöntemlerle Sabit Zamanlı Sinyalize Kavşaklar için Optimum Devre Süresi Modeli. *El-Cezeri*, *6*(2), 259–269. https://doi.org/10.31202/ECJSE.496257

Cakici, Z., & Murat, Y. S. (2019). A Differential Evolution Algorithm-Based Traffic Control Model for Signalized Intersections. *Advances in Civil Engineering*, *2019*, 7360939. https://doi.org/10.1155/2019/7360939

Çakici, Z., & Murat, Y. Ş. (2021). Sinyalize Dönel Kavşaklarda Diferansiyel Gelişim Algoritması ile Sinyal Süre Optimizasyonu. *El-Cezeri*, *8*(2), 635–651. https://doi.org/10.31202/ECJSE.861429

Clerc, M. (1999). The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, *3*, 1951–1957. https://doi.org/10.1109/CEC.1999.785513

Dong, C., Huang, S., & Liu, X. (2010). Urban Area Traffic Signal Timing Optimization Based on Sa-PSO. *2010 International Conference on Artificial Intelligence and Computational Intelligence*, *3*, 80–84. https://doi.org/10.1109/AICI.2010.257

Eberhart, R. C., & Shi, Y. (1998). Comparison between genetic algorithms and particle swarm optimization. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *1447*, 611–616. https://doi.org/10.1007/BFB0040812/COVER

Erdoğmuş, P. (2010). (1) (PDF) Particle swarm optimization performance on special linear programming problems. *Scientific Research and Essays*, *5*(12), 1506–1518. https://www.researchgate.net/publication/229041601_Particle_swarm_optimization_performance_on_special_linear_programming_problems

Erdoğmuş, P., & Yalçın, E. (2015). Parçacık Sürü Optimizasyonu ile Kısıtsız Optimizasyon Test Problemlerinin Çözümü. *İleri Teknoloji Bilimleri Dergisi*, *4*(1), 14–22. https://dergipark.org.tr/tr/pub/duzceitbd/issue/4817/66451

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, *4*, 1942--1948 c.4. https://doi.org/10.1109/ICNN.1995.488968

Li, X., Zhao, Z., Liu, L., Liu, Y., & Li, P. (2017). An Optimization Model of Multi-Intersection Signal Control for Trunk Road under Collaborative Information. *Journal of Control Science and Engineering*, *2017*, 2846987. https://doi.org/10.1155/2017/2846987

Panda, S., & Padhy, N. P. (2007). Comparison of Particle Swarm Optimization and Genetic Algorithm for TCSC-based Controller Design. *International Journal of Electrical and Computer Engineering*, *1*(3), 551–559. https://doi.org/10.5281/ZENODO.1082007

Parrott, D., & Li, X. (2006). Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *IEEE Transactions on Evolutionary Computation*, *10*(4), 440–458. https://doi.org/10.1109/TEVC.2005.859468

Peñabaena-Niebles, R., Cantillo, V., Moura, J. L., & Ibeas, A. (2017). Design and Evaluation of a Mathematical Optimization Model for Traffic Signal Plan Transition Based on Social Cost Function. *Journal of Advanced Transportation*, *2017*, 1943846. https://doi.org/10.1155/2017/1943846

Shaikh, P. W., El-Abd, M., Khanafer, M., & Gao, K. (2022). A Review on Swarm Intelligence and Evolutionary Algorithms for Solving the Traffic Signal Control Problem. *IEEE Transactions on Intelligent Transportation Systems*, *23*(1), 48–63. https://doi.org/10.1109/TITS.2020.3014296

Zhang, Y., Gao, K., Zhang, Y., & Su, R. (2019). Traffic Light Scheduling for Pedestrian-Vehicle Mixed-Flow Networks. *IEEE Transactions on Intelligent Transportation Systems*, *20*(4), 1468–1483. https://doi.org/10.1109/TITS.2018.2852646