Terakreditasi SINTA Peringkat 2
Surat Keputusan Dirjen Penguatan Riset dan Pengembangan Ristek Dikti No. 10/E/KPT/2019
masa berlaku mulai Vol. 1 No. 1 tahun 2017 s.d Vol. 5 No. 3 tahun 2021

Terbit online pada laman web jurnal: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 4 No. 2 (2020) 219 - 227 ISSN Media Elektronik: 2580-0760

Implementasi Algoritma Firefly pada Kasus N-Queens Problem

Dedy A. Nggego[1], Arief Setyanto[2], Sukoco[3]
[1]Magister Teknik Informatika Universitas Amikom Yogyakarta
[2,3]Universitas Amikom Yogyakarta
[1]dedynggego@gmail.com, [2]arief_s@amikom.ac.id, [3]maskoco@gmail.com

Abstract

N-Queen problem is a form of puzzle game that uses chess rules for the queen on the standard chessboard with modified size. The challenge of the n-queen problem is finding the N (N is positive integer) queens position on the chessboard, so that no queen can attack another queen on the board in a single move. Implementation of firefly algorithm in n-queens problem in this study aims to find n-queen problem solutions and count the number of iterations to achieve the optimal solution of each queen which will then be compared with the results of Sarkar and Nag's research (2017). This study uses an experimental method with a number of N between 10 to 20 and uses a population of 15 and 1000 firefly. The results showed that the firefly algorithm is able to find all the optimal solutions for the queen's position on a chessboard with dimensions 10 to 20 in a population of 1000 firefly. The firefly algorithm can find the optimal solution fewer iterations compared to the genetic algorithm. According to the experiment, firefly algorithm shows better performance in finding the optimal solution compared to genetic algorithm.

Keywords: *firefly algorithm, genetic algorithm, n-queens problem, artificial intelligence.*

Abstrak

N-Queen problem adalah bentuk permainan menggunakan aturan permain catur pada papan catur standard namun ukurannya dapat dimodifikasi menjadi lebih besar. Tantangan dari n-queen problem adalah bagaimana menempatkan n (n adalah bilangan bulat positif) buah queen pada papan catur berukuran nxn, dimana setiap queen tidak boleh saling memakan dengan hanya 1 langkah. Queens dapat memakan dengan arah vertical horizontal ataupun diagonal baik maju ataupun mundur satu langkah. Implementasi algoritma firefly pada n-queens problem dalam penelitian ini bertujuan untuk menemukan solusi n-queen problem berdasarkan aturan n-queen lalu menghitung jumlah iterasi yang diperoleh dari setiap solusi optimal masing-masing n ratu tersebut yang kemudian akan dikomparasikan dengan hasil penelitian Sarkar dan Nag (2017). Penelitian ini menggunakan metode eksperimen dengan jumlah N antara 10 sampai dengan 20 dan menggunakan populasi 15 dan 1000 firefly. Hasil yang diperoleh menunjukkan bahwa algoritma firefly mampu menemukan semua solusi optimal posisi ratu pada papan catur dengan dimensi 10 sampai dengan 20 pada populasi 1000 firefly. Algoritma firefly dapat menemukan solusi optimal dengan jumlah iterasi lebih sedikit dibanding dengan algoritma genetik, dengan kata lain bahwa algoritma firefly lebih baik dibanding dengan algoritma genetik dilihat dari jumlah iterasi yang dibutuhkan pada kasus yang sama.

Kata kunci: algoritma firefly, algoritma genetik, *n-queens problem, artificial intelligence.*

1. Pendahuluan

Dalam dunia komputasi, permasalahan dipandang berdasarkan jumlah solusi yang dapat diperoleh dan usaha untuk memperoleh solusi tersebut. Untuk mendapatkan solusi, sebuah prosedur dapat saja menebak setiap kemungkinan secara acak (*brute force*). Untuk kasus dengan ruang solusi yang besar teknik ini akan membutuhkan iterasi yang sangat banyak. Misalkan jika diberikan 10 buah angka, lalu diminta untuk mencari yang maksimal, sebuah algoritma dapat diminta mengulang 10 kali dan setelah 10 kali dipastikan bahwa nilai maksimum telah diperoleh. Namun ada kalanya sebuah persoalan memiliki banyak solusi dan jumlah solusinya tidak diketahui. Masalah komputasi yang tidak bisa diselesaikan secara komputasional dikenal juga sebagai *NP-Complete/NP-hard*: waktu eksekusi sebuah algoritma untuk masalah tersebut diyakini bertumbuh secara eksponensial dengan ukuran input [1]. Untuk itulah diperlukan

Diterima Redaksi : 19-12-2019 | Selesai Revisi : 13-03-2020 | Diterbitkan Online : 20-04-2020

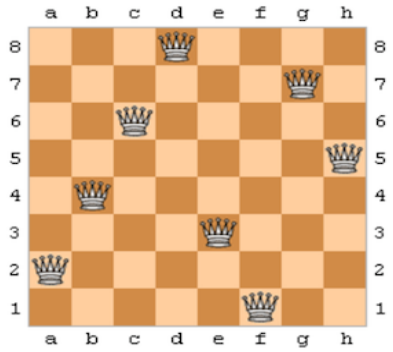
sebuah metode optimasi untuk menyelesaikan masalah-masalah seperti NP-complete dan NP-Hard.

Dalam dunia komputer dikenal sebuah istilah metode metaheuristik. Metaheuristik adalah *framework* algoritma yang terinspirasi dari alam, yang dimaksudkan untuk menyelesaikan masalah optimisasi yang kompleks [2]. Beberapa algoritma yang termasuk dalam metode metaheuristik antara lain *Genetic Algorithm* (GA), *Particle Swarm Optimization* (PSO), *Ant Colony Optimization* (ACO), *Firefly Algorithm* dan masih banyak lagi [3]. [4] mengatakan bahwa salah satu topik yang paling penting di antara berbagai topik yang berhubungan dengan perhitungan adalah estimasi kompleksitas dan optimisasi. Metaheuristik dapat melakukan optimisasi dan menemukan kandidat solusi yang sangat besar untuk masalah yang dihadapi [4].

Ada beberapa permasalahan yang umum diselesaikan menggunakan metode metaheuristik, misalnya *NP-Hard problem*, yaitu masalah yang memiliki solusi - *non-deterministic polynomial time*, solusi ini berjalan dalam waktu polinomial [5]. Salah satu kasus yang termasuk dalam *np-hard* adalah *n-queens problem* [6]. *N-queens problem* secara umum menggunakan dasar permainan catur yaitu masalah bagaimana meletakkan bidak menteri (*queen*) sebanyak n pada papan catur berukuran $n_x n$ sehingga tidak ada bidak ratu yang saling menangkap atau memakan hanya dengan satu langkah.

Pada penelitian [7], n *queens* problem dipecahkan menggunakan algoritma genetika dengan jumlah n yang diujikan yaitu n=10-25, dan mendapatkan 3 solusi optimal masing-masing n dengan hasil iterasi yang bervariasi. Penelitian lain [8] menerapkan GA untuk menemukan jumlah maksimal solusi yang mungkin diperoleh pada papan catur 8x8, dan hasilnya ditemukan 92 kemungkinan solusi optimal untuk papan catur 8x8. Menurut penelitian [9] algoritma *Firefly* dapat diterapkan pada problem serupa dalam teka-teki catur yaitu *Knight Tour Problem* (KTP) yang merupakan urutan langkah kuda pada papan catur tepat satu kali. Pada penelitian lainnya [10] mengimplementasikan algoritma *Firefly* pada kasus *Travelling Salesman Problem* dan mendapatkan hasil bahwa algoritma tersebut lebih baik dibandingkan dengan 3 algoritma lainnya yaitu *Genetic Algorithm* (*GA*), *Simulated Annealing* (*SA*), dan *Ant Colony Optimation* (*ACO*) hal ini dilihat dari panjang tour terbaik yang diperoleh. Penelitian [11] memecahkan *n-queens problems* yang berfokus pada perhitungan *cost* dan *execution time*, menggunakan algoritma Biogeography-based Optimation (BBO) yang kemudian dibandingkan dengan algoritma PSO mengatakan bahwa algoritma BBO lebih akurat dan cepat daripada PSO, namun ketika BBO dibandingkan dengan GA, diperoleh bahwa GA mempunyai performa lebih baik. GA mempunyai waktu eksekusi yang lebih baik. Pada penelitian [12] digunakan pendekatan algoritma *genetic* untuk memecahkan kasus *minimum dominating set of queens problem*, dengan jumlah n=8-11. Hasil ekseprimen menunjukkan bahwa secara sistematis rata-rata nilai *fitness* mendekati maksimum pada iterasi-iterasi selanjutnya. [13] menggunakan *parallel* DNA *algorithm* untuk memecahkan kasus *n-queens problem* dengan jumlah n=4 memperoleh hasil bahwa algoritma yang diusulkan mempunyai tingkat kesalahan yang lebih rendah untuk hibridasi. Pada studi komparatif [14] menggunakan algoritma berbeda pada kasus *n-queens problem* dikatakan bahwa GA dihibrid dengan SA mempunyai kinerja yang lebih baik terkait waktu eksekusi dibanding dengan GA standar dan BF serta BT. Selain daripada itu, algoritma *firefly* standar maupun hibrid telah banyak diterapkan pada beberapa kasus seperti *job schedulling problem, mechanical design optimization problem, predict evaporation, color image segmentation, clustering, numerical optimization problem, global solar radiation prediction* masing-masing pada penelitian [15], [16], [17], [18], [19], [20], [21] dan memperoleh hasil yang baik pada masing-masing kasus. Pada kasus *rigid image registration* menggunakan pendekatan *chaotic firefly algorithm* [22] dikatakan bahwa CFA sangat efektif untuk kasus tersebut karena mempunyai rata-rata tingkat kesalahan yang rendah.

Pada penelitian ini hasil jumlah iterasi yang diperoleh nantinya akan dikomparasikan dengan hasil yang diperoleh dari penelitian [7] dengan jumlah n=10-20 untuk melihat berapa iterasi yang dibutuhkan kedua algoritma FA dan GA untuk menemukan solusi optimal *n-queens problem*. Contoh solusi *n-queens problem* dengan jumlah ratu 8 dapat dilihat pada gambar 1.



Gambar 1. Solusi *n-queens* 8x8

Pada tTbel 1 di bawah ini beberapa penelitian [23], [24], [11] yang sudah dilakukan pada kasus *n-queens problem* menggunakan algoritma yang berbeda dengan penelitian ini, lihat Tabel 1.

Pada tabel 1 di atas dapat dilihat bahwa pada penelitian [7] diperoleh 3 solusi optimal masing-masing n dengan jumlah iterasi misalnya pada n=25 yaitu 1413 iterasi, 382 iterasi, dan 281 iterasi. [23] memodifikasi GA dengan menggunakan teknik *greedy initialization* dan *best break-point* dan hasilnya efisien dalam

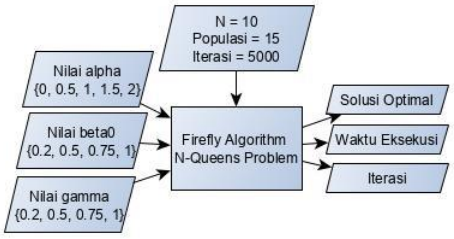
memecahkan *n-queens problem.* [11] dilihat dari waktu eksekusinya, GA lebih baik dibanding BBO dan PSO. Selanjutnya [24] hibrid BA-GA mempunyai nilai *penalty* lebih sedikit dibandingkan BA dan GA standar. Selain itu ada juga penelitian lain yang juga membahas tentang *n-queens problem* yaitu penelitian [25] yang mengkombinasikan algoritma *Cuckoo Search* dan *L'evy Flights*, lalu dibandingkan dengan algoritma *Backtracking*, dan diperoleh bahwa kombinasi CS dan LF lebih efektif dan efisien dibanding dengan BT. [26] mengusulkan *parallel genetic algorithm*, untuk memecahkan kasus yang sama. Hasilnya menunjukkan bahwa algoritma mempunyai kemampuan untuk menemukan solusi terkait masalah ini, tidak hanya cepat namun juga mengarah pada kinerja yang lebih baik.

Tabel 1. Penelitian pada kasus *n-queens problem*

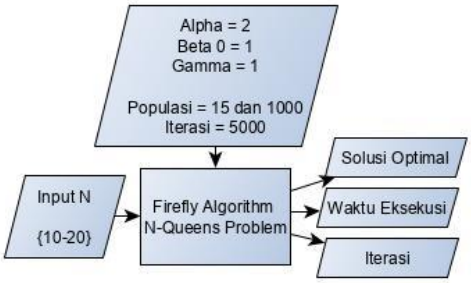
Peneliti	Metode Pengukuran	Algo.	N	Hasil
Sarkar dan Nag	Jumlah Iterasi 3 solusi optimal N.	GA	10	-2383 Iterasi -125 Iterasi -251 Iterasi
			25	-1431 Iterasi -382 Iterasi -281 Iterasi
Heris dan Oskoei	Waktu rata-rata evaluasi fitness function setiap N.	HGA	8	554.48 s
			512	1152110.00 s
		GA	8	48522.00 s
			512	*
Habiboghli dan Jalali	Compare execution time.	BBO	5	255299/0 s
			200	9828/4 s
		GA	5	0175/0 s
			200	4107/0 s
		PSO	5	036943/0 s
			200	8218/1 s
Al-Gburi, dkk	Compare penalty cost.	BA	8	0 penalty
			500	330 penalty
		GA	8	0 penalty
			500	384 penalty
		BA-GA	8	0 penalty
			500	272 penalty

2. Metode Penelitian

Penelitian ini menggunakan metode eksperimen dan bersifat kuantitatif. Hasil penelitian ini diperoleh dari percobaan pemecahan *n-queens problem* menggunakan *Firefly Algorithm* (FA) dan data yang akan bandingkan dengan penelitian [7]. Penelitian ini dibagi menjadi 2 tahap, tahap pertama bertujian menentukan parameter optimum dari algoritma *firefly*, sedngkan tahap kedua bertujuan untuk menguji algoritma *firefly* dengan parameter optimumnya untuk memecahkan masalah *N-Queen problem* dengan variasi N (jumlah *queen*). Pada gambar 2 disajikan desain percobaan pertama untuk menentukan parameter *alfa beta* dan *gamma* optimum, sedangkan gambar 3 menggambarkan percobaan kedua untuk mencari solusi optimal *n-queens problem* menggunakan parameter yang diperoleh dari percobaan pertama.

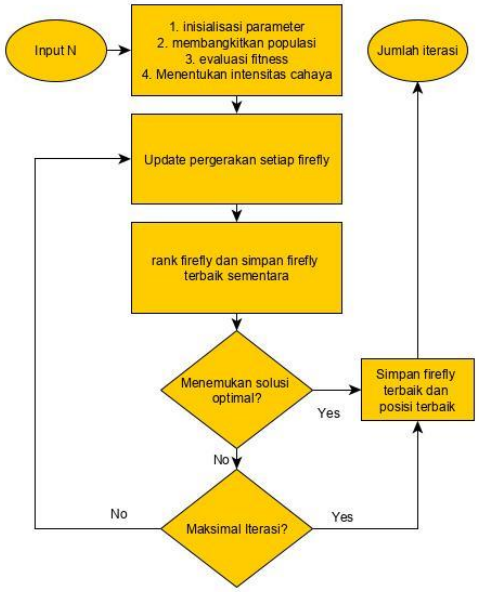


Gambar 2. Desain Percobaan 1 mencari nilai α, β0, dan γ



Gambar 3. Desain Percobaan 2 mencari solusi optimal n-queens problem

Cara kerja algoritma *firefly* disajikan dalam gambar 4.



Gambar 4. Desain penelitian

Algoritma *firefly* bekerja dengan prinsip metaheuristic dimana pada awal akan dibangkitkan beberapa kandidat solusi. Populasi kandidat solusi ditentukan di awal, pada masing masing iterasi, setiap kandidat solusi akan diuji kedekatannya dengan solusi terbaik. Setiap iterasi, kandidat solusi akan di-*update* dengan aturan *update*

yang ditentukan oleh parameter *alfa*, *beta* dan *gamma*. Selanjutnya akan memasuki iterasi berikutnya, dan pengujian kedekatan solusi dengan target (*fitness*). Iterasi akan berhenti jika jumlah iterasi maksimum tercapai atau solusi optimal telah ditemukan.

3. Hasil dan Pembahasan

Ada 2 percobaan yang dilakukan pada penelitian ini, yaitu percobaan pertama untuk mencari parameter nilai α, β0, dan γ yang akan digunakan untuk parameter pada percobaan kedua. Penentuan parameter tersebut berdasarkan nilai *fitness* (*f(x)*) dan nilai *execution time* paling kecil yang diperoleh seperti pada Tabel 2.

Tabel 2. Parameter hasil percobaan pertama

N-Ratu	Pop	Param	Nilai	Iterasi	f(x)	waktu
				4	0	
		A	2	17	0	4.13145 s ±
				2	0	
				4	0	
10	15	β0	1	3	0	3.44529 s ±
				12	0	
				4	0	
		γ	1	3	0	3.39955 s ±
				12	0	

Populasi yang digunakan pada percobaan pertama adalah 15 *firefly*. Menurut [27] pada kebanyakan kasus populasi yang digunakan oleh FA adalah 15-50

3.1. Prosedur *Firefly Algorithm* pada Penyelesaian *N-Queens Problem*

Di bawah ini dapat dilihat *pseudocode firefly algorithm* sebagai berikut.

```
Procedure FA
Begin
  input data;
  inisialisasi parameter;
  bangkitkan populasi awal firefly;
  hitung fungsi fitness;
  hitung intensitas cahaya setiap firefly;
  while (t < maksimal_iter)
    For i <- 1 to banyak firefly
      For j <- 1 to banyak firefly
        If (I(xi) < I(xj))
          Movement;
          Hitung fungsi tujuan;
          Update intensitas cahaya;
        End If
      End For j
    End For i
    Tentukan firefly terbaik sementara;
  End While
  Tentukan firefly terbaik dan posisi terbaik;
End
```

Langkah pertama yang dilakukan adalah menginputkan jumlah N yang sekaligus akan menjadi dimensi papan catur N*x*N. Kemudian dilanjutkan dengan inisialisasi parameter yang akan digunakan. Selanjutnya membangkitkan populasi awal *firefly* x_i ($i = 1,2, ..n$), untuk menentukan posisi awal *firefly* secara acak. Sebagai contoh dengan jumlah *n=10* dapat dilihat populasi awal *firefly* yang dibangkitkan secara acak pada Tabel 3.

Langkah selanjutnya yaitu menghitung nilai *fitness* dari setiap posisi awal *firefly* dengan cara menghitung jumlah baris, kolom, dan diagonal yang berisi lebih dari 1 ratu, sehingga diperoleh nilai *fitness* masing-masing *firefly* seperti pada Tabel 4.

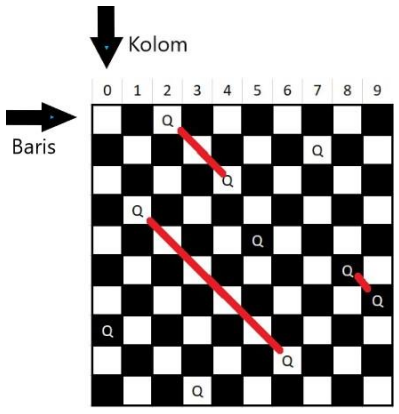
Tabel 3. Populasi awal firefly

Firefly	Firefly Sesudah Dibangkitkan Secara Acak
x_1	[7, 4, 2, 6, 5, 0, 9, 8, 1, 3]
x_2	[2, 0, 8, 1, 5, 3, 7, 4, 9, 6]
x_3	[2, 4, 9, 7, 1, 5, 6, 0, 3, 8]
x_4	[1, 9, 8, 0, 7, 6, 3, 5, 4, 2]
x_5	[2, 1, 4, 7, 5, 0, 9, 8, 6, 3]
x_6	[7, 5, 3, 0, 2, 1, 9, 8, 4, 6]
x_7	[2, 4, 8, 9, 1, 0, 6, 5, 3, 7]
x_8	[2, 0, 8, 9, 1, 7, 6, 5, 4, 3]
x_9	[5, 0, 9, 2, 4, 6, 3, 7, 8, 1]
x_{10}	[2, 3, 1, 4, 6, 5, 9, 0, 7, 8]
x_{11}	[2, 7, 4, 1, 5, 8, 9, 0, 6, 3]
x_{12}	[1, 2, 4, 3, 5, 7, 9, 6, 8, 0]
x_{13}	[7, 9, 3, 0, 8, 5, 2, 4, 1, 6]
x_{14}	[7, 5, 6, 1, 2, 8, 4, 3, 0, 9]
x_{15}	[3, 7, 6, 8, 5, 4, 9, 2, 0, 1]

Tabel 4. Nilai *fitness* masing-masing *firefly*

Firefly	Firefly Sesudah Dibangkitkan Secara Acak	Fitness
x_1	[7, 4, 2, 6, 5, 0, 9, 8, 1, 3]	9
x_2	[2, 0, 8, 1, 5, 3, 7, 4, 9, 6]	5
x_3	[2, 4, 9, 7, 1, 5, 6, 0, 3, 8]	4
x_4	[1, 9, 8, 0, 7, 6, 3, 5, 4, 2]	7
x_5	[2, 1, 4, 7, 5, 0, 9, 8, 6, 3]	4
x_6	[7, 5, 3, 0, 2, 1, 9, 8, 4, 6]	9
x_7	[2, 4, 8, 9, 1, 0, 6, 5, 3, 7]	9
x_8	[2, 0, 8, 9, 1, 7, 6, 5, 4, 3]	17
x_9	[5, 0, 9, 2, 4, 6, 3, 7, 8, 1]	6
x_{10}	[2, 3, 1, 4, 6, 5, 9, 0, 7, 8]	9
x_{11}	[2, 7, 4, 1, 5, 8, 9, 0, 6, 3]	3
x_{12}	[1, 2, 4, 3, 5, 7, 9, 6, 8, 0]	7
x_{13}	[7, 9, 3, 0, 8, 5, 2, 4, 1, 6]	4
x_{14}	[7, 5, 6, 1, 2, 8, 4, 3, 0, 9]	7
x_{15}	[3, 7, 6, 8, 5, 4, 9, 2, 0, 1]	8

Firefly ke-11 mempunyai nilai *fitness* terbaik yaitu dengan nilai 3. Sehingga *firefly* tersebut akan disimpan sebagai *firefly* terbaik sementara. *Firefly* tersebut jika divisualisasikan pada papan catur akan tampak seperti Gambar 5.



Gambar 5. Visualisasi posisi bidak pada papan catur

Selanjutnya menentukan intensitas cahaya masing-masing *firefly* dengan persamaan 1 berikut ini.

$$I(x) = 1/fitness \quad (1)$$

Sehingga diperoleh intensitas cahaya masing-masing *firefly* seperti pada Tabel 5..

Tabel 5. Intensitas cahaya *firefly*

Firefly	Fitness	Intensitas
x_1	9	0.11
x_2	5	0.2
x_3	4	0.25
x_4	7	0.14
x_5	4	0.25
x_6	9	0.11
x_7	9	0.11
x_8	17	0.05
x_9	6	0.16
x_{10}	9	0.11
x_{11}	3	0.33
x_{12}	7	0.14
x_{13}	4	0.25
x_{14}	7	0.14
x_{15}	8	0.12

Semua *firefly* akan dibandingkan berdasaarkan intensitasnya. Karena $I(x_1) < I(x_2)$ maka algoritma akan melakukan perhitungan jarak antara kedua *firefly* dengan persamaan 2 sebagai berikut.

$$r_{ij} = || x_i - x_j || = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2} \quad (2)$$

Dimana *d* adalah dimensi dan *k* adalah elemen posisi dari dimensi *d*, sehingga:

$$r_{ij} = \sqrt{\sum_{k=1}^{10} (x_{1,k} - x_{2,k})^2}$$

$$r_{ij} = \sqrt{(7-2)^2 + (4-0)^2 + (2-8)^2 + \cdots + (3-6)^2}$$

$$r_{ij} = 14.2828568570857$$

Kemudian akan dilanjutkan dengan menghitung *attractiveness* dengan menggunakan persamaan 3 sebagai berikut.

$$\beta = \beta_0 e^{-\gamma r_{ij}^{\wedge 2}} \quad (3)$$

Sehingga:

$$\beta = 1 \text{ x } e^{-1 \text{ x } 14.28285685708572}$$

$$\beta = 2.534694904308283e\text{-}89$$

Selanjutnya *firefly* akan melakukan perpindahan ke posisi yang baru menggunakan persamaan 4 sebagai berikut.

$$x_{i,k}\_new = x_{i,k} + \beta(x_{j,k} - x_{i,k}) + \alpha (\text{rand} - \tfrac{1}{2}), \quad (4)$$

Dimana nilai rand adalah elemen dari 0-1, sehingga:

$$x_{1,1}\_new = 7 + 2.534694904308283e\text{-}89 (2-7) + 2 (0.9528906111216019 - \tfrac{1}{2})$$

$x_{1,1}\_new$ = 5.905781222243204, dibulatkan menjadi 6.

Demikian juga halnya dengan posisi *k=2...10* akan dilakukan perhitungan yang sama sehingga diperoleh *firefly* 1 yang baru seperti pada Tabel 6.

Tabel 6. *Firefly* 1 pada posisi yang baru

Posisi (k)	Nilai Random (0-1)	Hasil Perhitungan Posisi Yang Baru	Pembulatan
k=1	0.9528906111216019	5.905781222243204	6
k=2	0.9942861337171172	2.9885722674342343	3
k=3	0.10252325695869124	-0.7949534860826173	-1
k=4	0.594866795451052	4.1897335909021045	4
k=5	0.680585344215862	3.361170688431724	3
k=6	0.44977145591265943	-2.1004570881746814	-2
k=7	0.610637648192606	7.221275296385212	7
k=8	0.8240252507020702	6.648050501404141	7
k=9	0.24640341417576384	-1.5071931716484723	-2
k=10	0.9427624408417746	1.8855248816835493	2

Pada posisi *k=3*, *k=6* dan *k=9* terjadi perhitungan yang melebihi batas minimal dimensi papan catur. Oleh karena itu untuk menangani hal serupa, maka dilakukan pengecekan terhadap setiap posisi *k* yang baru seperti berikut.

While :
  Posisi_baru < 0 :
    Posisi_baru + dimensi
While :
  Posisi_baru ⩽ dimensi :
    Posisi_baru - dimensi

Dengan demikian posisi *firefly* yang baru akan menjadi seperti pada Tabel 7.

Tabel 7. *Firefly* 1 posisi baru setelah perhitungan

Posisi (k)	Nilai Random (0-1)	Hasil Perhitungan Posisi Yang Baru	Pembulatan
k=1	0.9528906111216019	5.905781222243204	6
k=2	0.9942861337171172	2.9885722674342343	3
k=3	0.10252325695869124	-0.7949534860826173	9
k=4	0.594866795451052	4.1897335909021045	4
k=5	0.680585344215862	3.361170688431724	3
k=6	0.44977145591265943	-2.1004570881746814	8
k=7	0.610637648192606	7.221275296385212	7
k=8	0.8240252507020702	6.648050501404141	7
k=9	0.24640341417576384	-1.5071931716484723	8
k=10	0.9427624408417746	1.8855248816835493	2

Oleh karena *firefly* x_1 yang baru akan menjadi *array* [6, 3, 9, 4, 3, 8, 7, 7, 8, 2]. Kemudian menghitung nilai *fitness* dari posisi *firefly* x_1 yang baru dengan cara melakukan pengecekan pada baris, kolom dan diagonal sehingga diperoleh nilai *fitness firefly* x_1 yaitu 8. Dari proses ini dilakukan perhitungan nilai intensitas cahaya *firefly* x_1 seperti persamaan (1) sehingga diperoleh $I(x_1)$ adalah 0.125.

Hasil perbandingan *firefly* x_1 selengkapnya disajikan pada Tabel 8.

Setelah proses perpindahan maka akan diperoleh populasi *firefly* baru dimana posisi *firefly* x_1 yang baru adalah [0, 1, 8, 1, 7, 5, 5, 8, 6, 1] dengan nilai *fitness* 13 dan intensitas cahaya 0.076. demikian juga untuk *firefly* x_2 sampai dengan x_{15} akan dilakukan proses yang sama sehingga diperoleh populasi baru *firefly*.

3.2. Menentukan *Firefly* Terbaik

Firefly terbaik ditentukan berdasarkan nilai *fitness* terendah pada setiap perhitungan. *Firefly* terbaik merupakan calon solusi yang akan diambil untuk memecahkan kasus *n-queens problem* yang berarti mempunyai resiko tabrakan paling rendah bidak catur.

Tabel 8. Hasil perbandingan cahaya *firefly* x_1

Firefly	*k 1*	*k 2*	*k 3*	*k 4*	*k 5*	*k 6*	*k 7*	*k 8*	*k 9*	*k 10*
x_1_*lama*	7	4	2	6	5	0	9	8	1	3
x_2	2	0	8	1	5	3	7	4	9	6
r	14.2828568570857									
β	2.534694904308283e-89									
x_1_*baru*	6	3	9	4	3	8	7	7	8	2
$f(x_1)$	8									
$I(x_1)$	0.125									
x_1_*baru*	6	3	9	4	3	8	7	7	8	2
x_3	2	4	9	7	1	5	6	0	3	8
r	12.24744871391589									
β	7.175095973164411e-66									
x_1_*baru*	4	2	7	2	0	5	6	5	6	0
$f(x_1)$	9									
$I(x_1)$	0.11									
x_1_*baru*	4	2	7	2	0	5	6	5	6	0
x_4	1	9	8	0	7	6	3	5	4	2
r	11.40175425099138									
β	3.4811068399043105e-57									
x_1_*baru*	3	9	5	0	9	4	4	3	3	8
$f(x_1)$	11									
$I(x_1)$	0.090									
x_1_*baru*	3	9	5	0	9	4	4	3	3	8
x_5	2	1	4	7	5	0	9	8	6	3
r	15.198684153570664									
β	4.764032113782328e-101									
x_1_*baru*	2	8	2	9	6	2	1	2	1	7
$f(x_1)$	13									
$I(x_1)$	0.076									
.....										
..... x_{15}										
x_1_*baru*	0	1	8	1	7	5	5	8	6	1
$f(x_1)$	13									
	0.076									

3.3. Menentukan *Global Best* Sementara

Pada setiap iterasi akan ditentukan sebuah *firefly* terbaik untuk menjadi *global best.* Penentuan ini berdasarkan nilai *fitness* yang dimiliki oleh setiap *firefly. Firefly* yang mempunyai nilai *fitness* terendah akan dibandingkan dengan *firefly* yang menjadi *global best* sementara sehingga diperoleh *global best* yang baru. Hal tersebut akan terus berulang hingga ditemukan *firefly* dengan nilai *fitness* 0 atau hingga batas maksimal *firefly.*

3.4 Hasil Eksekusi Algoritma *Firefly* pada Kasus *N-Queens Problem*

Parameter yang digunakan dalam mengeksekusi algoritma *firefly* pada kasus *n-queens problem* dapat dilihat pada Tabel 9.

Tabel 9. Parameter algoritma

α	γ	β0	Populasi	Max Iter	N
2	1	1	15	5000	10-20
2	1	1	1000	5000	10-20

Adapun 3 solusi optimal masing-masing n hasil eksekusi FA dengan populasi 15 dapat dilihat pada Tabel 10.

Tabel 10. Hasil eksekusi algoritma *firefly*

N	Iterasi	Fitness	Solusi	Waktu Eksekusi
10	3	0	[2, 9, 1, 8, 5, 3, 0, 7, 4, 6]	4.7335 s ±
	14	0	[1, 8, 5, 7, 9, 0, 2, 4, 6, 3]	
	4	0	[3, 8, 2, 7, 9, 0, 5, 1, 4, 6]	
	28	0	[4, 2, 8, 5, 1, 10, 6, 0, 3, 7, 9]	
11	4	0	[6, 2, 7, 10, 1, 5, 9, 0, 3, 8, 4]	9.0275 s ±
	27	0	[1, 5, 7, 2, 10, 8, 4, 0, 3, 9, 6]	
	20	0	[3, 8, 4, 7, 10, 0, 11, 1, 5, 2, 6, 9]	
12	47	0	[7, 10, 1, 3, 0, 9, 4, 8, 5, 11, 2, 6]	17.7722 s ±
	28	0	[1, 3, 6, 10, 7, 2, 11, 5, 8, 0, 9, 4]	
	227	0	[8, 5, 3, 9, 11, 2, 4, 6, 1, 12, 10, 0, 7]	
13	106	0	[8, 1, 5, 7, 2, 0, 12, 4, 9, 11, 6, 10, 3]	68.6967 s ±
	39	0	[2, 7, 11, 4, 8, 0, 3, 10, 6, 1, 9, 5, 12]	
	324	0	[0, 8, 12, 4, 6, 10, 2, 13, 11, 7, 3, 1, 9, 5]	
14	194	0	[5, 7, 13, 3, 0, 9, 4, 2, 11, 8, 12, 1, 6, 10]	132.8445 s ±
	92	0	[10, 3, 6, 12, 5, 11, 0, 4, 13, 8, 2, 9, 7, 1]	
	1529	0	[5, 14, 8, 3, 0, 6, 13, 11, 1, 7, 12, 10, 2, 4, 9]	
15	150	0	[8, 3, 13, 7, 10, 2, 5, 12, 1, 9, 11, 14, 0, 4, 6]	433.1803 s ±
	209	0	[8, 2, 7, 13, 10, 5, 0, 6, 3, 11, 14, 1, 9, 4, 12]	
	300	0	[6, 2, 5, 12, 15, 3, 8, 11, 1, 14, 10, 0, 9, 4, 13, 7]	
16	109	0	[2, 13, 15, 4, 14, 5, 10, 1, 11, 6, 3, 0, 8, 12, 9, 7]	422.1155 s ±
	916	0	[7, 11, 6, 14, 5, 8, 0, 13, 3, 1, 9, 4, 2, 10, 12, 15]	
	-	1	[3, 11, 14, 7, 10, 16, 1, 4, 15, 9, 2, 13, 8, 12, 5, 0, 6]	
17	2712	0	[4, 14, 5, 8, 15, 11, 0, 6, 16, 1, 10, 12, 2, 9, 7, 3, 13]	3334.1929 s ±
	1483	0	[3, 10, 12, 9, 1, 5, 14, 6, 0, 16, 11, 15, 7, 4, 8, 13, 2]	
	-	1	[16, 14, 6, 17, 7, 1, 11, 5, 10, 4, 0, 3, 12, 9, 8, 15, 13, 2]	
18	172	0	[7, 14, 1, 13, 8, 6, 17, 15, 10, 4, 0, 3, 5, 16, 11, 9, 12, 2]	1306.0422 s ±
	3061	0	[2, 7, 3, 12, 9, 16, 13, 17, 1, 5, 10, 15, 0, 4, 8, 14, 11, 6]	

N	Iterasi	Fitness	Solusi	Waktu Eksekusi
	376	0	[10, 18, 7, 12, 1, 6, 2, 14, 8, 13, 16, 3, 11, 4, 17, 5, 9, 15, 0]	
19	-	2	[13, 0, 17, 11, 5, 12, 16, 3, 15, 6, 1, 4, 10, 8, 14, 18, 2, 9, 7]	3268.2356 s ±
	-	1	[17, 9, 5, 2, 8, 3, 15, 7, 14, 16, 1, 13, 11, 18, 6, 4, 0, 10, 12]	
	4091	0	[17, 9, 14, 3, 5, 7, 15, 11, 19, 6, 1, 18, 2, 0, 12, 8, 4, 16, 10, 13]	
20	-	1	[8, 3, 0, 11, 17, 19, 10, 12, 14, 1, 15, 6, 13, 7, 2, 18, 5, 9, 16, 4]	8656.0766 s ±
	-	1	[17, 8, 5, 15, 13, 7, 9, 6, 3, 4, 10, 19, 14, 1, 11, 0, 18, 16, 12, 2]	

Algoritma *firefly* dapat menemukan masing-masing 3 solusi optimal dengan *fitness* 0 pada dimensi 10 sampai dengan 16 dengan waktu eksekusi dibawah ± 500 detik. Namun pada dimensi 17 sampai dengan 20 hanya menemukan paling sedikit 1 solusi optimal dengan nilai *fitness* 0 dengan peningkatan waktu eksekusi yang meningkat.

Sedangkan 3 solusi optimal masing-masing n hasil eksekusi dengan menggunakan populasi 1000 dapat dilihat pada tabel 11.

Tabel 11. Hasil eksekusi FA dengan populasi 1000

N	Iterasi	Fitness	Solusi	Waktu Eksekusi
	1	0	[9, 2, 8, 3, 0, 7, 5, 1, 6, 4]	
10	-	0	[9, 2, 8, 1, 4, 7, 0, 6, 3, 5]	1.3531 s ±
	-	0	[7, 4, 2, 0, 9, 1, 5, 8, 6, 3]	
	1	0	[5, 10, 0, 4, 6, 8, 3, 1, 7, 9, 2]	
11	-	0	[4, 10, 1, 9, 6, 3, 0, 2, 8, 5, 7]	4.3730 s ±
	-	0	[6, 4, 10, 0, 3, 9, 7, 1, 8, 2, 5]	
	1	0	[6, 1, 10, 5, 0, 4, 9, 11, 2, 7, 3, 8]	
12	1	0	[8, 10, 4, 7, 0, 2, 11, 6, 1, 3, 5, 9]	12.8450 s ±
	1	0	[5, 11, 8, 1, 7, 4, 2, 0, 9, 6, 10, 3]	
	1	0	[11, 7, 1, 10, 5, 2, 8, 12, 4, 9, 0, 6, 3]	
13	1	0	[9, 6, 12, 2, 4, 1, 7, 10, 3, 11, 8, 5, 0]	87.8468 s ±
	1	0	[2, 4, 10, 8, 11, 5, 3, 0, 12, 7, 9, 6, 1]	
	1	0	[7, 10, 13, 3, 8, 0, 4, 9, 1, 5, 11, 2, 6, 12]	
14	1	0	[7, 9, 11, 0, 2, 4, 10, 8, 13, 3, 1, 6, 12, 5]	77.9390 s ±
	1	0	[12, 5, 13, 6, 0, 3, 7, 4, 10, 1, 9, 11, 2, 8]	
15	1	0	[3, 7, 10, 1, 13, 6, 8, 0, 2, 4, 14, 11, 9, 12, 5]	551.4728 s ±
	1	0	[11, 8, 4, 0, 10, 3, 6, 12, 9, 13, 5, 14, 1, 7, 2]	
	1	0	[9, 3, 1, 13, 11, 6, 4, 7, 12, 14, 2, 8, 5, 0, 10]	
16	1	0	[1, 5, 10, 8, 0, 4, 12, 14, 2, 11, 7, 3, 15, 6, 9, 13]	
	1	0	[2, 7, 3, 14, 11, 4, 10, 15, 5, 9, 13, 0, 8, 1, 12, 6]	1645.2878 s ±
	1	0	[8, 1, 5, 13, 11, 0, 15, 12, 4, 2, 14, 9, 6, 10, 3, 7]	
	1	0	[9, 14, 6, 3, 16, 12, 5, 0, 13, 4, 2, 8, 10, 1, 15, 11, 7]	
17	3	0	[10, 8, 5, 12, 16, 3, 7, 15, 13, 2, 9, 1, 6, 4, 14, 11, 0]	6712.8393 s ±
	1	0	[15, 13, 10, 8, 0, 3, 7, 16, 1, 11, 9, 14, 6, 4, 2, 12, 5]	
	1	0	[8, 16, 9, 4, 15, 5, 0, 11, 1, 12, 6, 3, 17, 2, 13, 10, 7, 14]	
18	1	0	[1, 12, 9, 2, 17, 14, 10, 5, 0, 15, 7, 4, 8, 13, 16, 3, 11, 6]	7007.0114 s ±
	2	0	[11, 0, 12, 4, 8, 5, 3, 15, 10, 7, 16, 14, 17, 2, 9, 6, 1, 13]	
	4	0	[15, 12, 6, 3, 7, 17, 4, 0, 13, 10, 16, 1, 8, 5, 2, 14, 18, 11, 9]	
19	5	0	[17, 13, 9, 4, 18, 10, 0, 11, 5, 1, 16, 14, 8, 6, 2, 15, 7, 12, 3]	55760.8527 s ±
	13	0	[15, 8, 12, 2, 9, 5, 18, 0, 14, 17, 11, 6, 16, 10, 4, 1, 3, 13, 7]	
	35	0	[11, 7, 10, 15, 19, 2, 0, 6, 8, 16, 5, 13, 17, 1, 3, 18, 14, 9, 4, 12]	
20	6	0	[16, 12, 3, 7, 0, 14, 11, 17, 1, 9, 5, 18, 2, 10, 6, 13, 15, 19, 4, 8]	188858.2223 s ±
	19	0	[5, 10, 0, 19, 15, 13, 18, 6, 8, 3, 16, 4, 9, 1, 17, 2, 7, 12, 14, 11]	

Algoritma *firefly* dapat menemukan 3 solusi optimal dengan nilai *fitness* 0 pada setiap N dengan jumlah iterasi paling banyak yaitu 35 iterasi. Pada N=10 dan N=11 solusi ke-2 dan ke-3 telah menemukan solusi optimal pada pembangkitan awal populasi *firefly* yaitu dengain nilai *fitness* 0, sehingga secara otomatis *firefly* tersebut akan menjadi *g-best* sementara, dan seberapapun iterasi dan perhitungan yang dilakukan tidak akan merubah *g-best* yang dibangkitkan pada populasi awal tersebut. Hasil eksekusi pada N < 17 rata-rata hanya membutuhkan 1 iterasi dengan waktu eksekusi bervariasi. Peningkatan iterasi terjadi pada N ≥ 17 dan membutuhkan waktu eksekusi di atas 6000 detik.

3.5 Komparasi GA dan FA

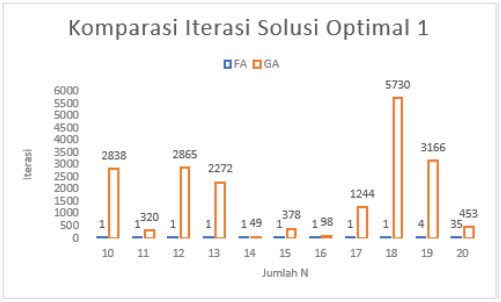
Data yang digunakan untuk komparasi ini adalah data hasil eksekusi FA dengan populasi 1000 untuk

menyesuaikan dengan penelitian [7] yang menggunakan populasi yang sama. Pada tabel 12 dibawah ini disajikan perbandingan iterasi kedua algoritma dalam menemukan 3 solusi optimal masing-masing n pada *n-queens problem* pada Tabel 12.

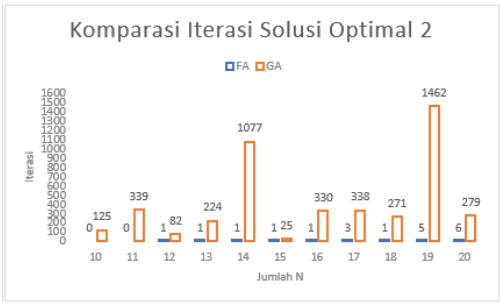
Diagram hasil komparasi iterasi GA dan FA dapat dilihat pada gambar 6, 7, 8.

Tabel 12. Perbandingan iterasi GA dan FA dalam menemukan 3 solusi optimal

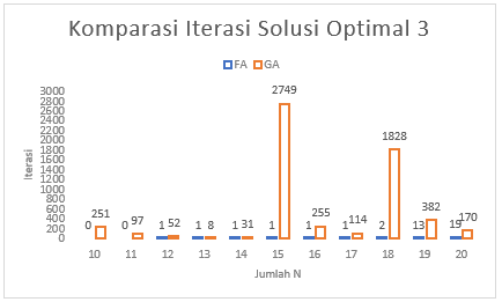
N	GA	FA	N	GA	FA
	2383	1		98	1
10	125	0	16	330	1
	251	0		255	1
	320	1		1244	1
11	339	0	17	338	3
	97	0		114	1
	2865	1		5730	1
12	82	1	18	271	1
	52	1		1828	2
	2272	1		3166	4
13	224	1	19	1462	5
	8	1		382	13
	49	1		453	35
14	1077	1	20	279	6
	31	1		170	19
	378	1			
15	25	1			
	2749	1			



Gambar 6. Diagram komparasi iterasi GA dan FA solusi optimal 1



Gambar 7. Diagram komparasi iterasi GA dan FA solusi optimal 2



Gambar 8. Diagram komparasi iterasi GA dan FA solusi optimal 3

Algoritma *firefly* mempunyai iterasi paling sedikit dibanding dengan algoritma *genetic* dalam menemukan solusi optimal kasus *n-queens problem* dengan jumlah n=10-20, dengan kata lain bahwa algoritma *firefly* lebih baik dibanding algoritma *genetic* pada kasus ini.

4. Kesimpulan

Dengan menggunakan populasi sebanyak 15 *firefly*, algoritma *firefly* dapat menemukan solusi optimal dengan *fitness* 0 pada dimensi 10 sampai 16 dengan waktu eksekusi dibawah ± 500 detik. Namun pada dimensi 17 sampai dengan 20 hanya menemukan paling sedikit 1 solusi optimal dengan nilai *fitness* 0. Sedangkan Dengan menggunakan populasi sebanyak 1000, algoritma *firefly* dapat menemukan semua solusi optimal dengan *fitness* 0 yaitu N sebanyak 10-20 dengan waktu eksekusi paling cepat yaitu ± 1.3 detik dan paling lama yaitu 188858.2 detik. Jumlah iterasi yang dibutuhkan untuk memecahkan kasus *n-queens problem* menggunakan algoritma *firefly* lebih sedikit dibanding dengan algoritma *genetic*, yaitu FA paling sedikit 0 dan paling banyak 35 iterasi, artinya algoritma *firefly* lebih baik dibanding dengan algoritma *genetic* pada kasus ini. Adapun saran untuk penelitian kedepan agar supaya menguji coba algoritma *firefly* pada kasus yang sama namun dengan jumlah dimensi papan catur yang lebih besar serta menambah jumlah maksimal iterasi. Adapun saran lain yaitu mengimplementasikan algoritma *firefly* untuk kasus optimasi di dunia nyata.

Daftar Rujukan

[1] Y. Huang, "Computing quantum discord is NP-complete Computing quantum discord is NP-complete," 2014.

[2] L. Bianchi, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr, "A survey on metaheuristics for stochastic combinatorial optimization," *Nat. Comput.*, vol. 8, no. 2, pp. 239–287, 2009.

[3] E.-G. Talbi, *METAHEURISTICS - From Design To Implementation*, vol. 66. 2012.

[4] S. Desale, A. Rasool, S. Andhale, and P. Rane, "Heuristic and Meta-Heuristic Algorithms and Their Relevance to the Real World: A Survey," *Int. J. Comput. Eng. Res. Trends*, vol. 351, no. 5, pp. 2349–7084, 2015.

[5] C. A. Cusack, "Online Games as Social-Computational Systems for Solving NP-complete Problems Running Head : SOCIAL-COMPUTATIONAL GAMES Online Games as Social-Computational Systems for Solving NP-complete Problems Charles A . Cusack , Jeff Largent , Ryan Alfuth , Kimberly Klask Hope College," no. January, 2015.

[6] S. Sathyapriya, R. Stephen, and V. S. J. Irudayaraj, "Survey on N-Queen Problem with Genetic Algorithm," no. 2, pp. 54–59, 2018.
[7] U. Sarkar and S. Nag, "An Adaptive Genetic Algorithm for Solving N-Queens Problem," 2017.
[8] A. S.Farhan, W. Z. Tareq, and F. H. Awad, "Solving N Queen Problem using Genetic Algorithm," *Int. J. Comput. Appl.*, vol. 122, no. 12, pp. 11–14, 2015.
[9] W. N. Mahfud, "Penerapan Algoritma Kunang-kunang (Firefly Algorithm) Dalam Knight Tour Problem Pada Papan Catur," *Biomass Chem Eng*, vol. 49, no. 23–6, pp. 22–23, 2015.
[10] A. Hatamlou, "Solving Travelling Salesman Problem Using Heart Algorithm," *Int. J. Appl. Evol. Comput.*, vol. 8, no. 4, pp. 32–42, 2018.
[11] A. Habiboghli and T. Jalali, "A Solution to the N-Queens Problem Using Biogeography-Based Optimization," *Int. J. Interact. Multimed. Artif. Intell.*, vol. 4, no. 4, p. 20, 2017.
[12] S. Alharbi and I. Venkat, "A Genetic Algorithm Based Approach for Solving the Minimum Dominating Set of Queens Problem," vol. 2017, 2017.
[13] Z. Wang, D. Huang, J. Tan, T. Liu, K. Zhao, and L. Li, "BioSystems A parallel algorithm for solving the n -queens problem based on inspired computational model," *BioSystems*, vol. 131, pp. 22–29, 2015.
[14] S. Mukherjee, S. Datta, P. B. Chanda, and P. Pathak, "Comparative Study of Different Algorithms to Solve N Queens Problem," *Int. J. Found. Comput. Sci. Technol.*, vol. 5, no. 2, pp. 15–27, 2015.
[15] S. Nickolas, "Institutional Repository A hybrid discrete firefly algorithm for solving multi-objective flexible job shop scheduling problems," pp. 0–26, 2015.
[16] A. Baykaso and F. B. Ozsoydan, "Adaptive firefly algorithm with chaos for mechanical design optimization problems," vol. 36, pp. 152–164, 2015.
[17] R. Moazenzadeh, B. Mohammadi, S. Shamshirband, and K. Chau, "Mechanics Coupling a firefly algorithm with support vector regression to predict evaporation in northern Iran," vol. 2060, 2018.
[18] V. Rajinikanth and M. S. Couceiro, "RGB Histogram based Color Image Segmentation Using Firefly Algorithm," *Procedia - Procedia Comput. Sci.*, vol. 46, no. Icict 2014, pp. 1449–1457, 2015.
[19] J. Mathew and D. . Vijayakumar, "Scalable parallel clustering using modified Firefly algorithm," *IOSR J. Comput. Eng.*, vol. 16, no. 6, pp. 14–24, 2014.
[20] S. Yu, S. Zhu, Y. Ma, and D. Mao, "A variable step size firefly algorithm for numerical optimization," *Appl. Math. Comput.*, vol. 263, pp. 214–220, 2015.
[21] L. Olatomiwa, S. Mekhilef, S. Shamshirband, and K. Mohammadi, "ScienceDirect A support vector machine – firefly algorithm-based model for global solar radiation prediction," vol. 115, pp. 632–644, 2015.
[22] Y. Zhang and L. Wu, "Rigid image registration based on normalized cross correlation and chaotic firefly algorithm," *Int. J. Digit. Content Technol. its Appl.*, vol. 6, no. 22, pp. 129–138, 2012.
[23] J. E. Aghazadeh Heris and M. A. Oskoei, "Modified genetic algorithm for solving n-queens problem," *2014 Iran. Conf. Intell. Syst. ICIS 2014*, 2014.
[24] A. F. J. Al-gburi, S. Naim, and A. N. Boraik, "Hybridization of Bat and Genetic Algorithm to Solve N-Queens Problem," vol. 7, no. 4, pp. 626–632, 2018.
[25] R. G. Sharma, "Implementation of n-Queens Puzzle using Meta-heuristic algorithm (Cuckoo Search)," vol. 2, no. 3, 2013.
[26] M. T. Sarvetamin, A. Khatibi, and M. H. Zahedi, "A New Approach to Solve N-Queen Problem with," vol. 4, no. 2, 2018.
[27] X. S. Yang, "Firefly algorithms for multimodal optimization," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5792 LNCS, pp. 169–178, 2009.