Hyper Sudoku Solver dengan Menggunakan Harris Hawks Optimization Algorithm

Eric Dinata, *Teknologi Informasi Institut Sains dan Teknologi Terpadu Surabaya*, Herman Budianto, *Teknik Informatika Institut Sains dan Teknologi Terpadu Surabaya*, dan Hendrawan Armanto, *Teknik Informatika Institut Sains dan Teknologi Terpadu Surabaya*

***Abstrak*—Sudoku merupakan salah satu permainan klasik yang digemari banyak orang. Sebagai salah satu permainan papan, Sudoku mempunyai banyak varian, salah satunya Hyper Sudoku. Hyper Sudoku mempunyai tingkat kesulitan yang lebih tinggi daripada Sudoku biasa. Tingkat kompleksitas yang tinggi membuat pemainan ini menjadi *brain teaser* yang baik dan sangat cocok diambil sebagai media untuk menguji algoritma metaheuristik. Algoritma yang populer pada dekade terakhir ini adalah algoritma metaheuristik berbasis populasi, yang mengadaptasi perilaku binatang dalam memecahkan permasalahan optimasi, salah satunya adalah Harris Hawks Optimization (HHO). Seperti kebanyakan metode *swarm intelligence* (SI) lainnya, algoritma ini mengandalkan proses *diversification* dan *intensification*. Selain itu, HHO mempunyai empat strategi khusus untuk mencari solusi dengan kondisi yang berbeda. HHO mampu mencakup solusi multi dimensi, sehingga sangat cocok diimplementasikan pada persoalan Hyper Sudoku. Pada proses pengujian, digunakan dua *setting parameter* yang berbeda, tiga macam persoalan Hyper Sudoku, dan tiga puluh *independent run* untuk mencapai hasil yang diinginkan. Berdasarkan hasil pengujian, dapat disimpulkan bahwa tingkat keberhasilan untuk mencari solusi pada persoalan Hyper Sudoku dengan menggunakan HHO berkisar antara 86 hingga 88%, dilihat dari *fitness value*-nya.**

***Kata Kunci*—HHO, Hyper Sudoku, metaheuristik, optimasi.**

I. PENDAHULUAN

Permasalahan optimasi yang ada pada kehidupan sehari-hari, sering kali tidak dapat diselesaikan dengan persamaan matematika biasa. Hal ini membuat banyak orang mencoba melakukan pendekatan dengan algoritma metaheuristik. Algoritma metaheuristik secara umum dapat dibagi menjadi tiga kategori, yaitu *physical-based*, *evolutionary-based*, dan *swarm-based*. Beberapa metode yang populer meliputi Genetic Algorithms (GA), Particle Swarm Optimization (PSO), dan Ant Colony Optimization (ACO).

Dalam dekade terakhir, muncul banyak metode *swarm-based* dengan menggunakan nama-nama hewan, karena memang metode ini mengimplementasikan perilaku natural dari jenis-jenis hewan tertentu pada algoritmanya, yang dianggap efektif dalam mencari solusi dalam masalah optimasi. Penelitian ini membahas sebuah metode baru yang dipublikasikan pada tahun 2019, yaitu Harris Hawks Optimization. Algoritma ini menggunakan pendekatan dari sekumpulan elang yang memantau dan mengepung mangsanya. Algoritma ini juga menggunakan dua fase, yakni *exploration* (*diversification*) dan *exploitation (intensification)*. Kedua fase ini umum dipakai pada metode-metode yang menggunakan *Swarm Intelligence* (SI).

Untuk permasalahan optimasi yang dipakai, penelitian ini menggunakan permainan Hyper Sudoku sebagai pokok masalah. Di mana permainan ini merupakan varian dari permainan Sudoku yang populer sejak tahun 1979, baik secara kertas maupun *online*. Hyper Sudoku juga sering dikenal sebagai Windoku, Four-square Sudoku, dan NRC Sudoku (pertama kali muncul pada koran Belanda NRC Handelsblad). Seperti yang kita ketahui, Sudoku standar memakai grid 9x9 dan memiliki aturan pokok, yaitu tidak boleh ada angka yang sama pada setiap baris, setiap kolom, dan setiap area yang ditentukan, dalam hal ini, sembilan area 3x3. Sedangkan Hyper Sudoku memiliki aturan tambahan, yaitu adanya area tambahan sebanyak empat kotak 3x3 (Four-square Sudoku) yang menyerupai jendela (Windoku).

II. HARRIS HAWKS OPTIMIZATION

Harris Hawks Optimization [1]merupakan *population-based metaheuristic algorithm*, dengan memimik pergerakan natural dari sekumpulan burung elang jenis Harris dalam mencari mangsa.

Gambar. 1. Elang Harris

Kunci utama dari HHO adalah perilaku kooperatif dan gaya pengejaran para elang yang disebut sebagai *surprise*

Eric Dinata, Teknologi Informasi Institut Sains dan Teknologi Terpadu Surabaya, Surabaya, Jawa Timur, Indonesia (e-mail: eric.dinata@gmail.com)

Herman Budianto, Teknik Informatika Institut Sains dan Teknologi Terpadu Surabaya, Surabaya, Jawa Timur, Indonesia (e-mail: herman.budianto@gmail.com)

Hendrawan Armanto, Teknik Informatika Institut Sains dan Teknologi Terpadu Surabaya, Surabaya, Jawa Timur, Indonesia (e-mail: hendrawan@stts.edu)

pounce (terkaman kejutan). Strategi ini dilakukan bersama-sama dengan cara mengepung sang mangsa dari berbagai arah. Ada beberapa pola pengejaran yang bisa dilakukan, bergantung pada berbagai skenario dan pola sang mangsa untuk melarikan diri. Dari berbagai pola tersebut, maka dibentuklah algoritma optimasinya.

Seperti kebanyakan metode *swarm intelligence* (SI) lainnya, HHO dibagi menjadi dua fase, yaitu fase *exploration* (eksplorasi) dan fase *exploitation* (exploitasi). Yang unik di HHO adalah adanya *perching strategies* (strategi hinggap para elang) pada fase *exploration* dan *escaping behavior* (perilaku melarikan diri mangsa) pada fase *exploitation*.

Ada beberapa pengembangan algoritma HHO, seperti *Harris Hawks optimization with information exchange* [2](IEHHO), yang menambahkan adanya pertukaran informasi antar elang, dan *Horizontal and vertical crossover of Harris hawk optimizer with Nelder-Mead simplex for parameter estimation of photovoltaic models* [3] (CCNMHHO), yang menambahkan *crisscross optimizer* dan *Nelder-Mead simplex algorithm*, yang dipakai dalam perhitungan *solar power system* (sistem tenaga surya).

A. *Fase pada HHO*

Variabel E adalah energi sang mangsa untuk melarikan diri. Jika nilai $|E| \geq 1$, berarti energi sang mangsa masih sangat banyak, bisa berpindah tempat dengan cepat, sehingga pada saat ini para elang memasuki fase pencarian, atau fase *exploration*. Sedangkan jika nilai $|E| < 1$, maka dimulailah fase *exploitation*, dengan catatan apabila $|E| \geq 0.5$, maka akan memasuki fase *soft besiege* dan apabila $|E| < 0.5$, maka akan memasuki fase *hard besiege*.

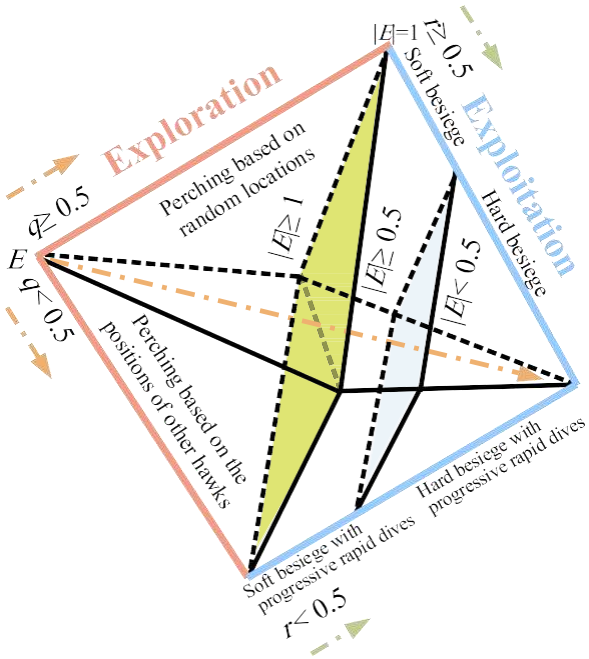
Variabel q adalah strategi hinggap, jika nilai $q < 0.5$, maka elang akan hinggap pada suatu posisi, dengan memperhitungkan posisi elang-elang lainnya. Jika nilai $q \geq 0.5$, maka elang akan hinggap pada pohon random atau bisa dibilang posisi random.

Variabel r adalah kemungkinan sang mangsa untuk melarikan diri. Jika nilai $r < 0.5$, maka kemungkinan sang mangsa untuk melarikan diri adalah besar. Sebaliknya, jika nilai $r \geq 0.5$, maka kemungkinan mangsa untuk melarikan diri adalah kecil.

B. *Fase exploration*

Pada fase ini, para elang akan melacak dan mendeteksi sang mangsa dengan penglihatan tajam mereka. Namun, hal ini tidak selalu mudah, dikarenakan sang mangsa sering kali tidak begitu saja muncul di area penglihatan para elang. Oleh karena itu, para elang akan menunggu, mengamati, dan memantau untuk mencari mangsa, bahkan sampai berjam-jam.

Pada HHO, para elang ibarat kandidat solusi dan kandidat terbaik pada setiap iterasi adalah yang mempunyai posisi yang sama dengan sang mangsa, atau yang paling mendekati.



Gambar. 2. Fase pada algoritma Harris Hawks Optimization

Psedocode dari HHO

Input: jumlah populasi (N) dan maksimum iterasi (T)
Output: posisi kelinci dan fitness valuenya
Inisialisasi populasi random $X_i(i = 1,2,\ldots,N)$
while (kondisi berhenti belum tercapai) **do**
  Hitung fitness value elang
  Set X_{rabbit} sebagai posisi kelinci (best location)
  for (setiap elang (X_i)) **do**
    Update energi awal (E_0) dan kekuatan lompat (J)
    `E₀=2rand()-1`
    `J=2(1-rand())`
    Update energi dengan persamaan (3)
    Fase eksplorasi
    if ($|E| \geq 1$) **then**
      Update lokasi dengan persamaan (1)
    Fase eksploitasi
    if ($|E| < 1$) **then**
      Soft besiege
      if ($r \geq 0.5$ and $|E| \geq 0.5$) **then**
        Update lokasi dengan persamaan (4)
      Hard besiege
      else if ($r \geq 0.5$ and $|E| < 0.5$) **then**
        Update lokasi dengan persamaan (6)
      Soft besiege with progressive rapid dives
      else if ($r < 0.5$ and $|E| \geq 0.5$) **then**
        Update lokasi dengan persamaan (10)
      Hard besiege with progressive rapid dives
      else if ($r < 0.5$ and $|E| < 0.5$) **then**
        Update lokasi dengan persamaan (11)
Return X_{rabbit}

Pada fase ini terdapat dua strategi, yang pertama, seekor elang akan hinggap pada suatu posisi dimana posisi ini dihitung atau ditentukan dari posisi kawan-kawannya dan juga posisi mangsa (supaya mereka berada pada posisi yang cukup dekat untuk menerkam mangsa), ini digambarkan pada persamaan 1, di mana kondisi $q < 0.5$, dan yang ke-dua, seekor elang akan hinggap pada posisi random, di mana

pada dunia nyata posisi ini adalah letak pepohonan yang dihinggapi oleh para elang tersebut, ini digambarkan pada persamaan 1, dengan kondisi $q \geq 0.5$.

$$X(t+1) = \begin{cases} X_{rand}(t) - r_1|X_{rand}(t) - 2r_2X(t)| & q \geq 0.5 \\ \left(X_{rabbit}(t) - X_m(t)\right) - r_3(LB + r_4(UB - LB)) & q < 0.5 \end{cases} \quad (1)$$

Variabel q merupakan *perching strategies* (strategi hinggap), di mana $X(t+1)$ adalah posisi *vector* dari para elang pada iterasi selanjutnya, $X_{rabbit}(t)$ adalah posisi dari sang mangsa, kelinci, kemudian $X(t)$ adalah posisi *vector* para elang yang sekarang. Variabel r_1, r_2, r_3, r_4, dan q adalah angka random di antara 0 dan 1, yang akan berganti terus pada setiap iterasi, kemudian LB dan UB adalah batas bawah *(lower bound)* dan atas *(upper bound)* untuk setiap variabel. $X_{rand}(t)$ adalah elang yang dipilih secara acak dari populasi saat ini, dan X_m adalah rata-rata posisi elang dari populasi saat ini.

Untuk menghitung rata-rata posisi elang dari populasi saat ini, digunakan persamaan berikut:

$$X_m(t) = \frac{1}{N}\sum_{i=1}^{N} X_i(t) \quad (2)$$

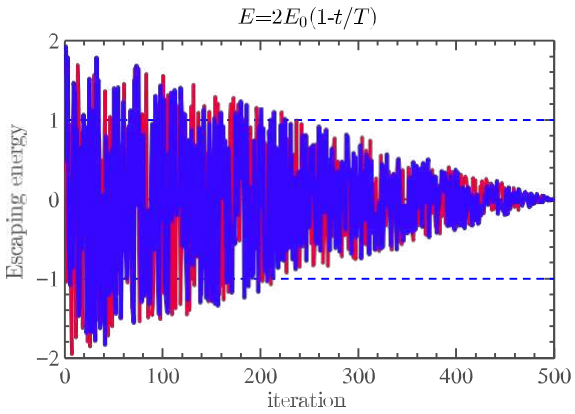
Di mana $X_i(t)$ merupakan lokasi dari setiap elang pada iterasi t dan N merupakan jumlah total elang.

C. *Transisi dari exploration ke exploitation*

Algoritma HHO akan mengalami transisi dari fase *exploration* ke fase *exploitation*. Kemudian, akan terjadi perubahan *exploitative behavior* (perilaku eksploitasi), yang bergantung pada *escaping energy* (tenaga untuk melarikan diri) dari sang mangsa dan juga *chance of prey escaping* (kemungkinan sang mangsa untuk berhasil melarikan diri). Untuk memodelkan kondisi ini, maka digunakan persamaan sebagai berikut:

$$E = 2E_0\left(1 - \frac{t}{T}\right) \quad (3)$$

Di mana E merupakan tenaga untuk melarikan diri sang mangsa, T adalah jumlah maksimum iterasi, dan E_0 adalah energi awal sang mangsa. Pada HHO, E_0 akan berganti secara acak pada setiap iterasi, dengan interval antara -1 dan 1. Jika nilai dari E_0 menurun dari 0 ke -1, maka itu berarti si kelinci melemah secara fisik. Sedangkan, sebaliknya, jika nilai E_0 naik dari 0 ke 1, maka itu berarti si kelinci mengalami penguatan secara fisik. Dinamika energi untuk melarikan diri ini cenderung mengalami penurunan pada proses iterasi. Pada saat $|E| \geq 1$, para elang akan mencari pada area yang berbeda untuk menjelajahi lokasi sang mangsa, yaitu dengan dilakukannya fase *exploration*. Sedangkan pada saat $|E| < 1$, algoritma ini akan memanfaatkan solusi-solusi yang berdekatan (*neighborhood*) pada fase *exploitation*. Singkatnya, fase *exploration* terjadi saat $|E| \geq 1$, sedangkan fase *exploitation* akan terjadi setelahnya, pada saat $|E| < 1$.



Gambar. 3. Hasil simulasi *E*

Simulasi E ini dilakukan dua kali, dengan jumlah iterasi masing-masing 500. Terdapat garis berwarna biru dan merah, yang mewakili simulasi pertama dan ke-dua. Seperti yang terlihat di gambar 3, *escaping energy* atau tenaga yang dimiliki sang mangsa untuk melarikan diri semakin lama akan semakin berkurang, atau mendekati 0.

D. *Fase exploitation*

Pada fase ini, para elang Harris akan melakukan *surprise pounce* (terkaman kejutan), atau disebut juga sebagai *seven kills*, dengan cara menyerang mangsa yang telah dideteksi pada fase sebelumnya. Akan tetapi, secara natural, sang mangsa akan sering kali mencoba untuk melarikan diri. Oleh karena itu, diperlukan strategi pengejaran yang berbeda pada kondisi sebenarnya. Berdasarkan dari perilaku melarikan diri dari sang mangsa dan strategi pengejaran para elang, maka ada empat macam strategi yang bisa dilakukan, yaitu *soft besiege*, *hard besiege*, *soft besiege with progressive rapid dives*, dan *hard besiege with progressive rapid dives.*

Sang mangsa akan selalu mencoba untuk melarikan diri dari situasi yang mengancam dirinya. Variabel r adalah besarnya peluang dari sang mangsa untuk berhasil melarikan diri (jika $r < 0.5$) dan gagal untuk melarikan diri (jika $r \geq 0.5$), sebelum *surprise pounce*. Apapun yang dilakukan sang mangsa, para elang akan melakukan *hard besiege* atau *soft besiege* untuk menangkap mangsa tersebut. Itu berarti para elang akan mengitari sang mangsa dari arah yang berbeda, secara *hard* atau *soft*, tergantung dari energi yang tersisa dari sang mangsa. Pada situasi sebenarnya, para elang akan mendekat perlahan-lahan menuju sang mangsa untuk memperbesar kesempatan mereka dalam bekerja sama membunuh sang mangsa dengan melakukan *surprise pounce*. Setelah beberapa menit, sang mangsa akan kehilangan banyak tenaga. Kemudian, para elang akan memperkuat proses pengepungan untuk menangkap sang mangsa yang sudah kehilangan banyak tenaga tersebut. Untuk memodelkan strategi ini dan menentukan apakah HHO akan menggunakan strategi *soft besiege* atau *hard besiege*, maka parameter E akan digunakan.

Dalam hal ini, jika $|E| \geq 0.5$, maka strategi *soft besiege* akan digunakan, dan jika $|E| < 0.5$, maka strategi *hard besiege* yang akan digunakan.

E. *Soft besiege*

Pada saat sang mangsa masih mempunyai tenaga yang cukup untuk melarikan diri dan kemudian mencoba untuk melompat ke sana kemari, para elang akan mengelilinginya secara perlahan, dengan tujuan untuk membuat sang mangsa kehabisan tenaga dan pada akhirnya akan melakukan *surprise pounce* (terkaman kejutan). Strategi ini terjadi jika $r \geq 0.5$ dan $|E| \geq 0.5$. Perilaku ini digambarkan dengan persamaan berikut:

$$X(t+1) = \Delta X(t) - E|JX_{rabbit}(t) - X(t)| \quad (4)$$

$$\Delta X(t) = X_{rabbit}(t) - X(t) \quad (5)$$

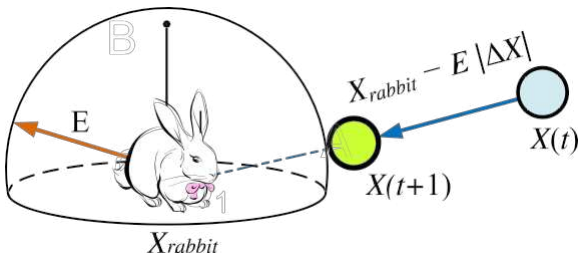
Di mana $\Delta X(t)$ adalah selisih antara posisi vector sang mangsa dan lokasi saat ini, pada iterasi t. Variabel r_5 merupakan angka random di antara 0 dan 1, yang akan digunakan untuk menghitung $J = 2(1 - r_5)$, yang merupakan kekuatan melompat kelinci sepanjang proses melarikan diri. Nilai J akan berganti dengan nilai acak pada setiap iterasi, untuk men-simulasi pergerakan natural kelinci.

F. *Hard besiege*

Pada saat sang mangsa sudah kehabisan tenaga dan kemungkinan untuk melarikan diri semakin kecil, maka para elang tidak lagi mengelilinginya secara perlahan, namun sudah bersiap untuk melakukan *surprise pounce*. Strategi ini terjadi jika $r \geq 0.5$ dan $|E| < 0.5$. Pada situasi ini, posisi elang pada iterasi selanjutnya akan dihitung dengan menggunakan persamaan berikut:

$$X(t+1) = X_{rabbit}(t) - E|\Delta X(t)| \quad (6)$$

Model sederhana untuk menggambarkan langkah ini, dapat dilihat dari gambar berikut, dengan menggunakan satu ekor elang sebagai contoh:



Gambar. 4. Contoh vector pada hard besiege

G. *Soft besiege with progressive rapid dives*

Pada saat sang mangsa masih mempunyai tenaga yang cukup untuk melarikan diri, para elang tetap akan melakukan soft besiege sebelum melakukan *surprise pounce*. Strategi ini terjadi jika $|E| \geq 0.5$, namun $r < 0.5$.

Untuk memodelkan pola pergerakan melarikan diri dari sang mangsa dan juga *leapfrog movement* (pergerakan melompat), maka digunakanlah fungsi *levy flight* (*LF*). Fungsi *LF* ini digunakan untuk meniru pergerakan *zigzag* (berkelok-kelok) dari sang mangsa pada saat melarikan diri, serta pergerakan tidak teratur dari para elang pada saat melakukan *rapid dives* (pergerakan menukik secara cepat) untuk menangkap sang mangsa.

Untuk melakukan *soft besiege*, para elang akan mengevaluasi untuk menentukan pergerakan mereka selanjutnya melalui persamaan berikut:

$$Y = X_{rabbit}(t) - E|JX_{rabbit}(t) - X(t)| \quad (7)$$

Kemudian, mereka akan membandingkan pergerakan ini dengan pergerakan sebelumnya untuk menentukan apakah itu merupakan sebuah pergerakan yang bagus atau tidak. Jika tidak, maka mereka akan mulai melakukan pergerakan yang tidak teratur dan mendadak untuk mendekati sang mangsa. Pergerakan ini akan digambarkan melalui persamaan berikut:

$$Z = Y + S \times LF(D) \quad (8)$$

Di mana D adalah dimensi dari masalah, S adalah vector random dengan ukuran $1 \times D$, dan LF adalah fungsi *levy flight*, yang dapat dihitung dengan menggunakan persamaan berikut:

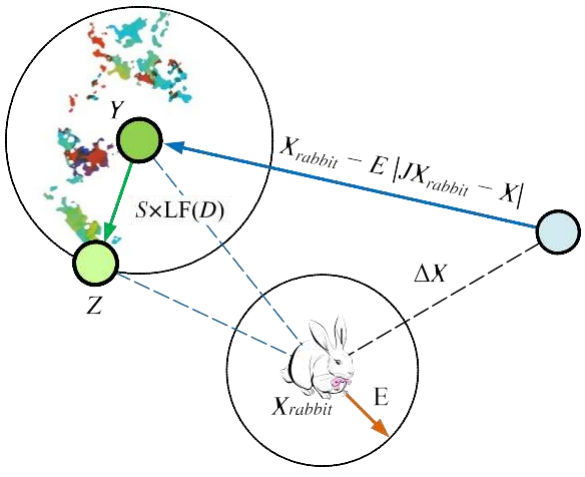
$$LF(x) = 0.01 \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}}, \sigma = \left(\frac{\Gamma(1+\beta) \times \sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \times \beta \times 2^{(\frac{\beta-1}{2})}} \right)^{\frac{1}{\beta}} \quad (9)$$

Di mana u dan v merupakan bilangan random di antara 0 dan 1, kemudian β adalah konstanta yang ditetapkan pada nilai 1.5.

Pada akhirnya, untuk meng-*update* posisi para elang pada fase *soft besiege* ini, digunakanlah persamaan sebagai berikut:

$$X(t+1) = \begin{cases} Y & if\ F(Y) < F(X(t)) \\ Z & if\ F(Z) < F(X(t)) \end{cases} \quad (10)$$

Di mana nilai Y dan Z didapatkan dari persamaan (7) dan (8). Ilustrasi sederhana langkah ini dapat dilihat pada gambar berikut, dengan menggunakan seekor elang sebagai contoh:



Gambar. 5. Contoh vector pada soft besiege with progressive rapid dives

Pada gambar 5 ini, terdapat titik-titik berwarna-warni, yang merupakan jejak lokasi dari pola perhitungan *LF* sebelumnya, yang kemudian mencapai posisi Z. Pada setiap langkah, hanya posisi Y atau Z yang lebih baik saja yang akan dipilih sebagai posisi selanjutnya. Strategi ini akan diterapkan pada setiap elang yang ada pada populasi.

H. Hard besiege with progressive rapid dives

Pada saat sang mangsa sudah tidak mempunyai tenaga yang cukup untuk melarikan diri, maka para elang akan melakukan *hard besiege* sebelum melakukan *surprise pounce*. Situasi pada langkah ini mirip dengan pada langkah *soft besiege*, hanya saja, pada langkah ini para elang mencoba memperkecil jarak rata-rata mereka dengan sang mangsa. Strategi ini terjadi jika $|E| < 0.5$ dan $r < 0.5$. Oleh karena itu, digunakanlah persamaan berikut:

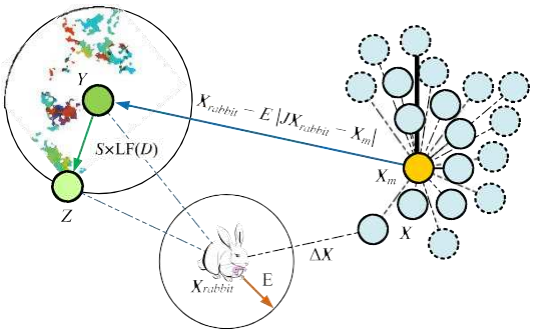
$$X(t+1) = \begin{cases} Y & if\ F(Y) < F(X(t)) \\ Z & if\ F(Z) < F(X(t)) \end{cases} \quad (11)$$

Di mana nilai Y dan Z didapatkan dari persamaan berikut:

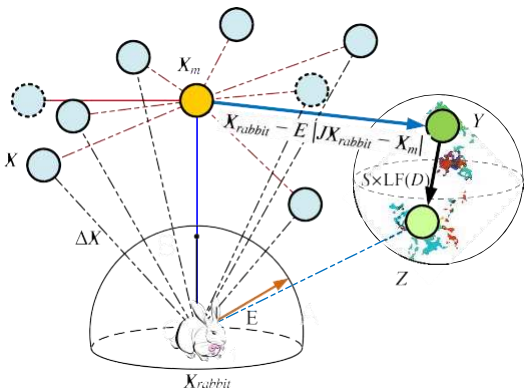
$$Y = X_{rabbit}(t) - E|JX_{rabbit}(t) - X_m(t)| \quad (12)$$

$$Z = Y + S \times LF(D) \quad (13)$$

Variabel $X_m(t)$ dapat dihitung dengan menggunakan persamaan (2.2). Ilustrasi sederhana dari langkah ini dapat dilihat dari dua gambar berikut:



Gambar. 6. Contoh vector pada hard besiege with progressive rapid dives (2D)



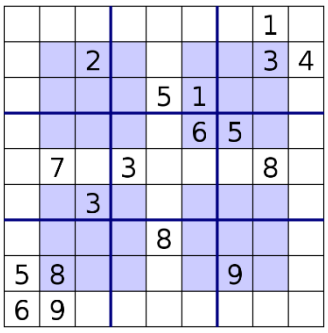
Gambar. 6. Contoh vector pada hard besiege with progressive rapid dives (3D)

Seperti halnya pada *soft besiege with progressive rapid dives*, titik-titik berwarna-warni merupakan jejak lokasi dari pola perhitungan *LF* sebelumnya dan hanya nilai Y atau Z saja yang akan dipakai sebagai posisi baru pada iterasi selanjutnya.

III. HYPER SUDOKU

Hyper Sudoku[4], [5] merupakan varian dari Sudoku standar, yang memiliki besar *grid* yang sama (9x9), yang terdiri dari 9 *region* 3x3. Yang membedakan Hyper Sudoku dari Sudoku standar adalah adanya 4 tambahan *region*[6][7], yang mengarah kepada adanya *constraint* tambahan, tidak boleh ada angka kembar pada *region* tambahan ini.

Angka-angka awal yang diberi dinamakan sebagai *givens*[6], [7]. Jumlah *givens* tidak mempengaruhi tingkat kesulitan. Untuk mencapai Sudoku dengan solusi unik, jumlah *givens* minimal untuk Sudoku standar adalah 17, sedangkan untuk Hyper Sudoku belum diketahui pasti, namun telah ditemukan persoalan Hyper Sudoku dengan 11 *givens*. Sudoku dengan solusi unik dinamakan sebagai *proper*[7] Sudoku.



Gambar. 7. Hyper Sudoku

Constraint pada Sudoku [8]:

1) Angka harus berada pada *range* 1 sampai dengan 9
2) Tidak ada angka yang sama pada setiap baris
3) Tidak ada angka yang sama pada setiap kolom
4) Tidak ada angka yang sama pada setiap *region*

Constraint pada Hyper Sudoku:

5) Tidak ada angka yang sama pada setiap *region* tambahan

Setiap *constraint* pada Hyper Sudoku ini akan dipakai untuk menghitung *fitness value* dalam satu solusi.

IV. IMPLEMENTASI

Implementasi algoritma HHO diawali dengan inisialisasi populasi elang, dengan angka random dalam batasan 0 (batas bawah) sampai dengan 1 (batas atas). Seekor elang mewakili satu solusi, di mana satu solusi mempunyai 81 dimensi (81 angka) yang di-*sort* secara per baris dan diberi indeks. Sembilan angka pertama mewakili baris pertama, sembilan angka ke-dua mewakili baris ke-dua, dan seterusnya. Dengan diberi indeks dan pengurutan per baris, maka dijamin tidak ada angka yang sama pada setiap baris.

TABEL I
INISIALISASI SOLUSI HYPER SUDOKU DENGAN HHO

X	unsorted	u-index	sorted	s-index
1	0.0185268	1	0.0185268	1
2	0.0751726	2	0.0751726	2
3	0.1938518	5	0.1418404	3
4	0.2032932	3	0.1938518	4
5	0.1418404	4	0.2032932	5
6	0.8858696	9	0.2326893	6
7	0.44012	7	0.44012	7
8	0.6946542	8	0.6946542	8
9	0.2326893	6	0.8858696	9

Angka yang ditampilkan mewakili baris pertama saja (sembilan angka pertama).

Angka-angka awal (*unsorted*) akan diurutkan (*sorted*), kemudian diberi indeks (*s-index*), setelah itu akan dilakukan *reverse indexing* untuk mengetahui indeks pada angka-angka awal (*u-index*).

Setelah itu, *fitness value* akan dihitung untuk setiap solusi, dengan cara menghitung banyaknya pelanggaran *constraint* terlebih dahulu:

1) Menghitung jumlah kotak kosong (a)
2) Menghitung jumlah kotak yang melanggar *constraint* baris (b)
3) Menghitung jumlah kotak yang melanggar *constraint* kolom (c)
4) Menghitung jumlah kotak yang melanggar *constraint region* awal pada Sudoku (d)
5) Menghitung jumlah kotak yang melanggar *constraint region* tambahan pada Hyper Sudoku (e)

$$Error = \frac{1}{5}\left(\frac{(a+b+c+d)}{81} + \frac{e}{36}\right) \quad (14)$$

Variabel a, b, c, dan d dibagi 81 karena angka maksimum pada *constraint-constraint* tersebut bernilai 81 (dari kotak 9x9). Sedangkan untuk variabel e dibagi 36 dikarenakan angka maksimum untuk *constraint* ini bernilai 36 (dari 4 kotak 3x3).

Berikutnya, karena $Error$ adalah besarnya pelanggaran *constraint* dalam bentuk persentase (antara 0 dan 1), maka untuk menghitung *fitness value* (FV), digunakan persamaan sebagai berikut:

$$FV = 1 - Error \quad (15)$$

Nilai *fitness value* berkisar antara 0 dan 1, di mana nilai 0 merupakan yang paling jelek dan nilai 1 merupakan yang paling baik. Jika *fitness value* bernilai 1, berarti solusi untuk soal Hyper Sudoku telah berhasil ditemukan.

Setelah proses inisialisasi, solusi dengan *fitness value* terbaik akan dipakai sebagai patokan awal, atau pada HHO dipakai istilah kelinci (X_{rabbit}) dan pada setiap iterasi dan fase yang dilalui setiap elang, elang terbaik akan dibandingkan dengan solusi terbaik tersebut. Apabila elang baru mempunyai *fitness value* yang lebih baik, maka X_{rabbit} akan di-*update*, apabila tidak, maka X_{rabbit} yang lama-lah yang akan dipakai untuk iterasi berikutnya.

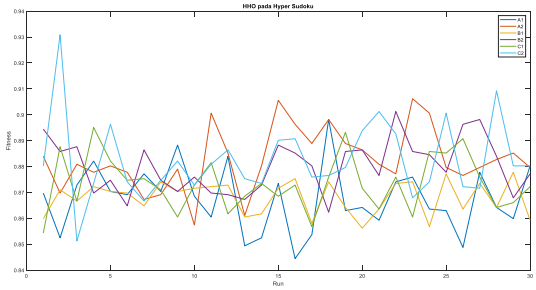
V. UJI COBA

Uji coba yang dilakukan adalah uji coba parameter, yaitu jumlah populasi, jumlah maksimum iterasi, dan juga jumlah *independent run*. Ada dua *setting parameter* yang diuji-cobakan, yang pertama adalah jumlah populasi 20 dengan jumlah maksimum iterasi 100. Kemudian yang ke-dua adalah jumlah populasi 30 dengan jumlah maksimum iterasi 500. Ke-dua *setting parameter* ini diuji-cobakan pada tiga soal Hyper Sudoku yang memiliki jumlah *givens* yang berbeda, masing-masing dilakukan 30 *independent run*.

TABEL II
UJI COBA HHO PADA HYPER SUDOKU DENGAN 30 RUN

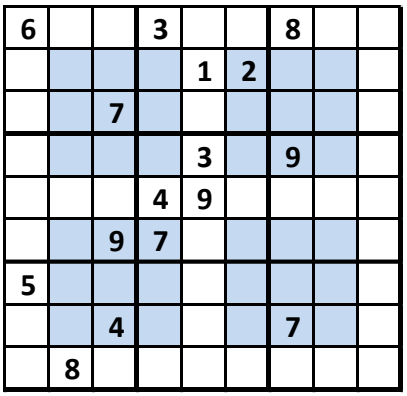
HHO	A1	A2	B1	B2	C1	C2
Givens	15	15	18	18	19	19
Populasi	20	30	20	30	20	30
Iterasi	100	500	100	500	100	500
Run	30	30	30	30	30	30
Avg Time (s)	3.95	22.14	4.09	27.56	4.03	27.88
Avg Fitness	0.868	0.883	0.868	0.88	0.873	0.883
Run-1	0.87	0.884	0.86	0.894	0.854	0.88
Run-2	0.852	0.87	0.871	0.886	0.888	0.931
Run-3	0.873	0.881	0.867	0.888	0.867	0.851
Run-4	0.882	0.878	0.872	0.87	0.895	0.873
Run-5	0.87	0.88	0.87	0.875	0.882	0.896
Run-6	0.869	0.878	0.87	0.865	0.875	0.874
Run-7	0.877	0.867	0.865	0.886	0.875	0.867
Run-8	0.87	0.869	0.874	0.875	0.871	0.875
Run-9	0.888	0.879	0.87	0.87	0.86	0.882
Run-10	0.869	0.857	0.872	0.876	0.873	0.873
Run-11	0.86	0.901	0.872	0.87	0.881	0.881
Run-12	0.884	0.885	0.873	0.869	0.862	0.886
Run-13	0.849	0.861	0.86	0.867	0.869	0.875
Run-14	0.852	0.88	0.862	0.873	0.873	0.873
Run-15	0.873	0.906	0.872	0.888	0.869	0.89
Run-16	0.844	0.896	0.875	0.885	0.873	0.891
Run-17	0.854	0.889	0.858	0.88	0.857	0.876
Run-18	0.898	0.898	0.874	0.862	0.876	0.877
Run-19	0.863	0.889	0.864	0.886	0.893	0.88
Run-20	0.864	0.886	0.856	0.886	0.872	0.894
Run-21	0.859	0.881	0.864	0.877	0.864	0.901
Run-22	0.874	0.877	0.873	0.901	0.876	0.893
Run-23	0.876	0.906	0.874	0.886	0.86	0.868
Run-24	0.864	0.901	0.857	0.885	0.886	0.874

Huruf A mewakili soal A, begitu juga dengan huruf B dan C.
Sedangkan angka pada A1 merupakan *setting parameter* 1 dan angka pada A2 merupakan setting parameter 2.

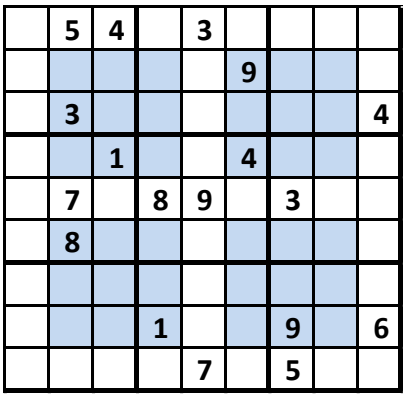


Gambar. 8. Grafik uji coba HHO pada Hyper Sudoku dengan 3 jenis persoalan, 2 macam *setting parameter*, dan 30 *run*.

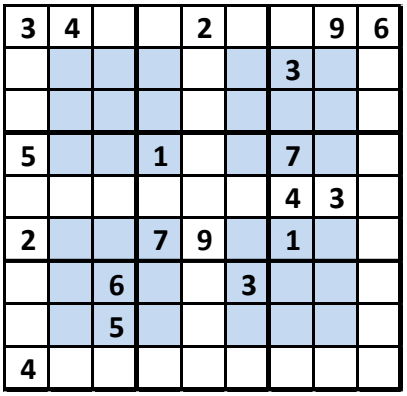
Ada tiga jenis soal Hyper Sudoku dengan jumlah *givens* yang berbeda-beda yang digunakan pada uji coba ini, yakni soal A dengan jumlah *givens* sebanyak 15, soal B dengan 18, dan soal C dengan 19.



Gambar. 9. Soal Hyper Sudoku A



Gambar. 10. Soal Hyper Sudoku B



Gambar. 11. Soal Hyper Sudoku C

Ada dua jenis setting *parameter* yang digunakan pada uji coba ini. Berikut perbandingan hasil kedua *setting parameter* tersebut pada masing-masing persoalan.

TABEL III
UJI COBA HHO PADA HYPER SUDOKU DENGAN SETTING PARAMETER 1

HHO	Soal A1	Soal B1	Soal C1
Givens	15	18	19
Populasi	20	20	20
Iterasi	100	100	100
Run	30	30	30
Avg Time (s)	3.952196	4.0927832	4.0303001
Avg Fitness	0.8676955	0.8680247	0.8734774

Uji coba dengan *setting parameter* 1 menggunakan jumlah populasi 20 dan jumlah maksimum iterasi 100.

TABEL IV
UJI COBA HHO PADA HYPER SUDOKU DENGAN SETTING PARAMETER 2

HHO	Soal A2	Soal B2	Soal C2
Givens	15	18	19
Populasi	30	30	30
Iterasi	500	500	500
Run	30	30	30
Avg Time (s)	22.141024	27.555999	27.884046
Avg Fitness	0.8827572	0.8800412	0.8825103

Uji coba dengan *setting parameter* 1 menggunakan jumlah populasi 30 dan jumlah maksimum iterasi 500.

Tabel III dan IV digunakan untuk membandingkan hasil antar persoalan Hyper Sudoku, dengan jumlah *givens* yang berbeda. Pada tabel III, terlihat peningkatan *average fitness*, namun tidak cukup signifikan. Pada tabel IV agak sedikit berbeda, terlihat *average fitness* pada soal A2 lebih tinggi dari *average fitness* pada soal yang lain, meskipun tidak banyak.

TABEL V
UJI COBA HHO PADA HYPER SUDOKU SOAL A

HHO	Soal A1	Soal A2
Givens	15	15
Populasi	20	30
Iterasi	100	500
Run	30	30
Avg Time (s)	3.952196	22.1410238
Avg Fitness	0.867695473	0.882757202

Uji coba dengan *setting parameter* 1 dan 2 pada Hyper Sudoku soal A.

TABEL VI
UJI COBA HHO PADA HYPER SUDOKU SOAL B

HHO	Soal B1	Soal B2
Givens	18	18
Populasi	20	30
Iterasi	100	500
Run	30	30
Avg Time (s)	4.0927832	27.5559993
Avg Fitness	0.868024691	0.880041152

Uji coba dengan *setting parameter* 1 dan 2 pada Hyper Sudoku soal B.

TABEL VII
UJI COBA HHO PADA HYPER SUDOKU SOAL C

HHO	Soal C1	Soal C2
Givens	19	19
Populasi	20	30
Iterasi	100	500
Run	30	30
Avg Time (s)	4.0303001	27.8840455
Avg Fitness	0.873477366	0.882510288

Uji coba dengan *setting parameter* 1 dan 2 pada Hyper Sudoku soal C.

Tabel V, VI, dan VII membandingkan hasil antar soal dengan *setting parameter* yang berbeda, apakah *setting parameter* 1 lebih baik atau *setting parameter* 2. Dari ketiga tabel, dapat dilihat bahwa *average fitness* dengan parameter ke-2 selalu lebih tinggi dari *average fitness* dengan parameter yang pertama.

Dari tabel II, dapat dilihat bahwa nilai *fitness* tertinggi ada pada *run* ke-2 soal C, dengan menggunakan *setting parameter* 2, yaitu bernilai sebesar 0.930864198.

3	4	1	5	2	7	8	9	6
7	6	5	1	8	9	3	4	2
4	8	9	2	3	5	1	6	7
5	3	4	1	6	8	7	2	9
1	7	8	5	9	2	4	3	6
2	3	5	7	9	4	1	8	6
4	8	6	9	5	3	2	7	1
9	1	5	4	2	3	6	7	8
4	8	2	1	7	6	9	3	5

Gambar. 12. Angka *fitness* tertinggi yang dicapai

Pada gambar 12, angka berwarna hitam berarti angka soal, angka berwarna hijau berarti angka yang benar, dan angka yang merah berarti angka yang salah.

VI. KESIMPULAN

Dari sini hasil uji coba, dapat ditarik beberapa kesimpulan, yang pertama adalah, *jumlah givens* yang lebih banyak tidak menjamin *fitness value* yang lebih tinggi. Yang kedua, jumlah populasi dan jumlah maksimum iterasi yang lebih banyak akan meningkatkan *fitness value*. Yang ketiga, tingkat keberhasilan untuk menemukan solusi pada semua parameter dan persoalan berkisar antara 86-88%.

Selain itu, dapat disimpulkan bahwa waktu komputasi tidak bergantung pada banyaknya jumlah *givens* pada persoalan. Namun, untuk perbedaan parameter, jumlah populasi dan jumlah maksimum iterasi, jelas mempengaruhi waktu komputasi secara signifikan.

Daftar Pustaka

[1] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Futur. Gener. Comput. Syst.*, vol. 97, 2019, doi: 10.1016/j.future.2019.02.028.

[2] C. Qu, W. He, X. Peng, and X. Peng, "Harris hawks optimization with information exchange," *Appl. Math. Model.*, vol. 84, pp. 52–75, 2020.

[3] Y. Liu *et al.*, "Horizontal and vertical crossover of Harris hawk optimizer with Nelder-Mead simplex for parameter estimation of photovoltaic models," *Energy Convers. Manag.*, vol. 223, 2020, doi: 10.1016/j.enconman.2020.113211.

[4] J. M. Weiss, "Genetic algorithms and sudoku," in *Midwest Instruction and Computing Symposium (MICS 2009)*, 2009, pp. 1–9.

[5] B. Michel, "Mathematics of NRC-Sudoku," 2007.

[6] M. Weller, "Counting, generating, and solving Sudoku," *Ph. D. Diss.*, 2008.

[7] A. Moraglio and J. Togelius, "Geometric particle swarm optimization for the sudoku puzzle," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, 2007, pp. 118–125.

[8] S. McGerty, "Solving Sudoku puzzles with particle swarm optimisation," *Final Report, Macquarie Univ.*, 2009.