Plant (Non-Fungi) Inspired Algorithm

Tugas Besar Desain Algoritma Lanjut
Muhammad Abiya Makruf (203012420034)

1. Nature Inspired Optimization Algorithm

1.1. Penjelasan

Nature-Inspired Optimization Algorithms (NIOA) atau Algoritma Optimasi Terinspirasi Alam adalah sekumpulan teknik pemecahan masalah yang meniru strategi dan mekanisme yang ditemukan di alam untuk menyelesaikan masalah optimasi yang kompleks [1]. Algoritma ini seringkali bersifat metaheuristik, yang berarti mereka tidak menjamin solusi optimal global tetapi bertujuan untuk menemukan solusi yang cukup baik dalam waktu yang wajar, terutama untuk masalah di mana metode eksak tidak praktis atau terlalu lambat .

1.2. Pembagian Secara Umum

NIOA dapat diklasifikasikan berdasarkan sumber inspirasinya. Pembagian ini membantu dalam memahami mekanisme dasar dan area aplikasi potensial dari masing-masing algoritma. Beberapa kategori utama meliputi:

- Terinspirasi Biologi (Biology-Inspired)
- Terinspirasi Fisika dan Kimia (Physics and Chemistry-Inspired)
- Terinspirasi Perilaku Manusia (Human Behavior-Inspired)
- Lain-lain: Ada juga kategori lain seperti yang terinspirasi dari musik, seni, atau algoritma hibrida yang menggabungkan beberapa inspirasi.

2. Plant (Non-Fungi) Algorithm

2.1. Penjelasan

Algoritma yang terinspirasi dari tumbuhan (Plant-Inspired Algorithms) merupakan cabang dari Nature-Inspired Optimization Algorithms (NIOA) yang relatif baru dibandingkan dengan algoritma yang terinspirasi dari hewan atau serangga. Ide dasarnya adalah meniru berbagai aspek dari "kecerdasan" dan strategi bertahan hidup tumbuhan dalam mencari sumber daya, berkembang biak, dan beradaptasi dengan lingkungannya.

2.2. Algoritma

Berikut adalah beberapa contoh algoritma yang terinspirasi dari tumbuhan beserta inspirasi utamanya:

2.2.1. Plant Propagation Algorithm (PPA)

- Inspirasi: Algoritma ini terinspirasi dari cara tumbuhan stroberi berkembang biak dan menyebar menggunakan tunas atau stolon (runners) [2].
- Mekanisme: PPA meniru strategi tanaman dalam menyebarkan tunas, tanaman di lokasi bagus menghasilkan banyak tunas pendek (eksploitasi), sedangkan tanaman di lokasi kurang baik menghasilkan sedikit tunas dengan jangkauan lebih jauh untuk mencari area baru (eksplorasi).

2.2.2. Invansive Weed Optimization (IWO)

- Inspirasi: Algoritma ini terinspirasi dari perilaku kolonisasi gulma invasif di suatu lahan pertanian. Gulma dikenal karena kemampuannya menyebar dengan cepat, beradaptasi, dan mendominasi suatu area [3].
- Mekanisme: IWO meniru penyebaran benih oleh gulma, gulma yang lebih adaptif (fitness tinggi) menghasilkan lebih banyak benih, yang disebarkan secara acak di sekitar induknya. Jangkauan penyebaran benih menyempit seiring iterasi berjalan.

2.2.3. Root Growth Algorithm (RGO)

- Inspirasi: RGO terinspirasi dari perilaku pertumbuhan adaptif sistem akar tumbuhan dalam mencari air dan nutrisi di dalam tanah. Akar tumbuhan menunjukkan plastisitas yang tinggi dalam merespons kondisi tanah, seperti menghindari rintangan dan menjelajahi area kaya nutrisi [4].
- Mekanisme: RGO memodelkan pertumbuhan akar tanaman, setiap ujung akar adalah solusi potensial yang bisa tumbuh memanjang, bercabang, atau berhenti. Akar dikelompokkan berdasarkan fitness (akar utama, lateral, dan menua). Akar utama tumbuh ke posisi terbaik lalu bercabang (monopodial), sementara akar lateral bisa menggantikan ujung asli (sympodial).

2.2.4. Tree Seed Algorithm (TSA)

- Inspirasi: TSA didasarkan pada hubungan antara pohon dan biji mereka dalam proses reproduksi dan penyebaran. Pohon menyebar ke permukaan melalui biji-biji mereka, yang kemudian tumbuh menjadi pohon baru [5].
- Mekanisme: TSA memodelkan solusi sebagai pohon dan biji dalam ruang pencarian, setiap pohon menghasilkan biji baru dengan dua strategi, yaitu bergerak ke arah pohon terbaik atau ke arah pohon lain secara acak. Pemilihan strategi diatur oleh Search Tendency (ST) untuk menyeimbangkan eksplorasi dan eksploitasi.

2.3. Alur Umum

Meskipun setiap algoritma memiliki inspirasi dan mekanisme detail yang unik, terdapat beberapa kemiripan alur umum dalam cara kerja keempat algoritma yang terinspirasi tumbuhan ini:

- Inisialisasi Populasi
- Evaluasi Fitness
- Mekanisme Reproduksi/Propagasi (Generasi Solusi Baru)
- Seleksi dan/atau Penggantian
- Iterasi hingga memenuhi kriterita berhenti

3. Permasalahan yang Diangkat

3.1. Vertex Cover

Permasalahan Vertex Cover adalah salah satu masalah optimasi klasik dalam teori graf. Diberikan sebuah graf tidak berarah $G = (V, E)$, di mana V adalah himpunan simpul dan E adalah himpunan sisi. Tujuannya adalah untuk himpunan V yang memiliki jumlah simpul paling sedikit namun tetap menutupi semua sisi dalam graf. Permasalahan ini diketahui NP-hard untuk graf umum, yang berarti tidak ada algoritma

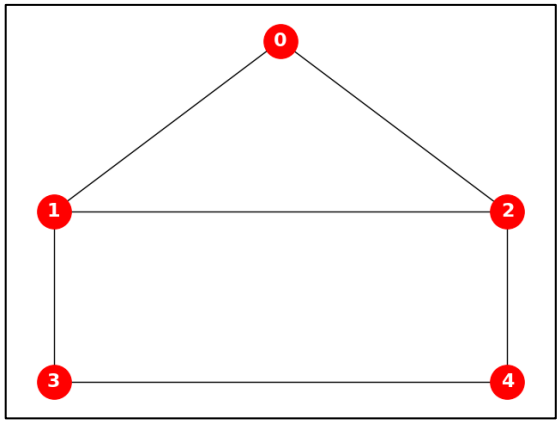
waktu polinomial yang diketahui dapat menemukan solusi optimal secara pasti untuk semua kasus.

3.2. Contoh Input

Berikut adalah contoh format isi file input.dat untuk graf dengan 5 node:

Input.dat
5
0 1
0 2
1 2
1 3
2 4
3 4

- Baris pertama (5) adalah jumlah node dalam graf (node diindeks dari 0 hingga 4).
- Baris-baris berikutnya masing-masing mendefinisikan sebuah sisi, dengan dua angka yang merepresentasikan node yang dihubungkan oleh sisi tersebut.
- Gambar 1 menunjukan contoh visualisasi dari input.dat



Gambar 1. Visualisasi Graf.

3.3. Contoh Solusi

Untuk graf contoh di atas, mari kita analisis beberapa kemungkinan Vertex Cover dan ukurannya:

- Graf: Node={0,1,2,3,4}, Edge={(0,1), (0,2), (1,2), (1,3), (2,4), (3,4)}
- Solusi Kandidat 1: {0, 1, 2, 3, 4} (Semua node)
  - Ukuran: 5
  - Valid? Ya, semua edge pasti tertutup.
- Solusi Kandidat 2: {0, 1, 4}
  - Ukuran: 3
  - Cek edge:
    - (0,1): Tertutup (oleh 0 atau 1)
    - (0,2): Tertutup (oleh 0)
    - (1,2): Tertutup (oleh 1)
    - (1,3): Tertutup (oleh 1)
    - (2,4): Tertutup (oleh 4)

        - (3,4): Tertutup (oleh 4)
    - Valid? Ya.
- Solusi Kandidat 3: {1, 2, 3}
- Solusi Kandidat 4 (Optimal): {1, 2, 4}
- Dan seterusnya.

Dalam contoh ini, ukuran Vertex Cover minimum adalah 3. Ada beberapa himpunan node berukuran 3 yang merupakan Vertex Cover yang valid (misalnya, {1,2,4}, {0,2,3}).

4. Detail Algoritma PPA dan IWO Untuk Menyelesaikan Vertex Cover

Implementasi pseudocode menggunakan bahasa c++ dapat dilihat di lampiran.

4.1. **Pseudocode** PPA

```
Algorithm PPA_VertexCover(Graph G)
    Initialize population with random solutions
    Set best_solution with very high fitness value
    For generation = 1 to MAX_GENERATIONS do
        For each plant in population do
            Evaluate fitness
        End for
        Sort population by ascending fitness
        If best plant in population is better than best_solution then
            Update best_solution
        End if
        Normalize and map fitness for each plant
        Initialize combined_population with all current plants
        For each plant in population do
            Determine number of runners using mapped fitness and randomness
            For each runner do
                Copy solution from parent
                Modify runner using mapped distance and sigmoid
                Evaluate fitness of runner
                Add runner to combined_population
            End for
        End for
        Sort combined_population by fitness
        Select top POPULATION_SIZE individuals as new population
    End for
    If best_solution fitness not valid then
        Evaluate fitness of best_solution
    End if
    Return best_solution
End
```

Gambar 2. Pseudocode PPA.

4.2. Pseudocode IWO

```
Algorithm IWO_VertexCover(Graph G)
    Initialize population with random solutions
    Set best_solution with very high fitness
    For iteration = 1 to MAX_ITERATIONS do
        For each weed in population do
            Evaluate fitness
        End for
        If population is empty, continue
        Find min and max fitness in current population
        Initialize combined_population with all current weeds
        For each weed in population do
            Determine number of seeds to produce based on fitness
            Compute current sigma value (σ_iter)
            For each seed to produce do
                Copy solution from parent weed
                For each bit in solution:
                    Apply Gaussian perturbation and sigmoid
                End for
                Evaluate fitness of new seed
                Add seed to combined_population
            End for
        End for
        Sort combined_population by fitness
        Select best individuals up to max population size
        Replace current population with selected individuals
        Update best_solution if better individual found
    End for
    If best_solution not updated, assign from population or compute fitness manually
    Return best_solution
End
```

Gambar 3. Pseudocode IWO.

5. Struktur Data

Berdasarkan implementasi algoritma PPA dan IWO yang ditulis menggunakan bahasa C++ yang disajikan pada lampiran, berikut adalah detail struktur data yang digunakan untuk PPA dan IWO:

5.1. Algoritma PPA

- Graph array of edge pairs (edges[])
- Solution/Plant = array (boolean, satu array untuk setiap solusi/plant)
- Population = array of Plant
- Combined Population = array of Plant (untuk seleksi dan runner)
- Fitness = double (satu per Plant)

5.2. Algoritma IWO

- Graph array of edge pairs (edges[])
- Solution/Weed array (boolean, satu array untuk setiap solusi/weed)
- Population array of weed
- Combined Population array of Weed (untuk benih/seed dan seleksi)
- Fitness = double (satu per Weed)

6. Kompleksitas

Analisis kompleksitas berikut didasarkan pada implementasi C++ menggunakan array C-style, untuk menyelesaikan masalah Vertex Cover.

Notasi yang Digunakan:

G_{max}: Jumlah generasi maksimum (untuk PPA) atau iterasi maksimum (untuk IWO).

N_{pop}: Ukuran populasi tetap untuk PPA.

N_{init}: Ukuran populasi awal untuk IWO.

$N_{\max\_pop}$: Ukuran populasi maksimum untuk IWO.

R_{max}: Jumlah *runner* atau benih maksimum per individu.

M: Jumlah node dalam graf.

E: Jumlah edge aktual dalam graf.

6.1. Kompleksitas Fungsi Fitness (Untuk PPA dan IWO)

- Menghitung cover_size : $O(M)$
- Menghitung uncovered_edges: $O(E)$
- Total: $O(M + E)$

6.2. PPA

Kompleksitas per Generasi

6.2.1. Evaluasi Fitness

- Ada N_{pop} tanaman
- Kompleksitas: $N_{pop} \times O(M + E)$

6.2.2. Pengurutan Populasi

- Mengurutkan N_{pop} tanaman
- Kompleksitas: $O(N_{pop} \log N_{pop})$

6.2.3. Normalisasi

- Iterasi sebanyak N_{pop} tanaman
- Kompleksitas: $O(N_{pop})$

6.2.4. Produksi Runner

- Outer loop untuk N_{pop} tanaman
- Inner loop untuk maksimal R_{max}runner per tanaman
- Setiap runner dihitung fitness nya: $O(M + E)$
- Total runner yang mungkin dihasilkan adalah $O(N_{pop} \times R_{max})$
- Kompleksitas produksi dan evaluasi: $N_{pop} \times R_{max} \times O(M + E)$

6.2.5. Pengurutan

- Ukuran combined_population maksimal adalah $N_{pop} + N_{pop} \times R_{max}$. Sebuh $S_{comb=} N_{pop}(1 + R_{max})$

- Kompleksitas: $O(S_{comb} \log S_{comb})$

6.2.6. Seleksi Generasi

- Menyalin N_{pop} elemen dari combined_population yang sudah terurut
- Kompleksitas: $O(N_{pop})$

Kompleksitas total PPA:

$G_{max} \times O(N_{pop}(M + E) + N_{pop} \log N_{pop})$

6.3. IWO

Kompleksitas per Iterasi

6.3.1. Evaluasi Fitness Populasi

- Ada N_{curr} gulma
- Kompleksitas: $N_{curr} \times O(M + E)$

6.3.2. Menemukan Fitness Min/Max

- Iterasi N_{curr} gulma
- Kompleksitas: $O(N_{curr})$

6.3.3. Reproduksi Benih dan Pengisian

- Outer loop untuk N_{curr} gulma
- Inner loop untuk maksimal R_{max} benih per gulma
- Setiap benih dihitung fitness nya: $O(M + E)$
- Total benih yang mungkin dihasilkan adalah $N_{curr} \times R_{max}$
- Kompleksitas produksi dan evaluasi fitness benih: $N_{curr} \times R_{max} \times (O(M + E)$
- Menyalin populasi awal ke combines_population: $O(N_{curr})$
- Menambahkan benih ke combined_population: $O(1)$

6.3.4. Pengurutan

- Ukuran combined_population maksimal adalah $N_{curr} + N_{curr} \times R_{max}$. Sebut $S_{comb\_iwo} = N_{curr}(1 + R_{max})$
- Kompleksitas: $O(S_{comb_{iwo}} \log S_{comb\_iwo})$

6.3.5. Eliminasi Kompetitif

- Menyalin maksimal $N_{\max\ _pop}$elemen dari combined_population yang sudah terurut.
- Kompleksitas: $O(N_{\max\ \_pop})$

Kompleksitas Total IWO:

$G_{max} \times O(N_{maxpop}(M + E) + N_{maxpop} \log N_{maxpop})$

7. Perbandingan PPA dan IWO

7.1. Representasi Solusi Individu

Keduanya menggunakan pendekatan yang sama untuk representasi solusi inti, yaitu array boolean 1 dimensi. PPA menyimpan lebih banyak jenis nilai fitness karena mekanisme internalnya yang berbeda dalam menentukan jumlah dan "jarak" runner.

7.2. Representasi Graf

Keduanya menggunakan struktur yang sama (array 1 dimensi dari pair) untuk menyimpan informasi graf. Tidak ada perbedaan signifikan dalam hal ini.

7.3. Manajemen Populasi

PPA cenderung bekerja dengan ukuran populasi yang lebih stabil, sedangkan IWO memiliki mekanisme di mana ukuran populasi bisa membengkak sementara karena produksi benih sebelum dipangkas kembali.

8. Kelebihan dan Kekurangan PPA dan IWO

Berikut adalah analisis kelebihan dan kekurangan dari Plant Propagation Algorithm (PPA) dan Invasive Weed Optimization (IWO), khususnya dalam konteks penerapan pada masalah optimasi seperti Vertex Cover.

Kelebihan	
Plant Propagation Algorithm (PPA)	**Invasive Weed Optimization (IWO)**
Manajemen Populasi Sederhana: Array solusi dan kombinasi runner ditangani secara langsung dalam satu blok memori, sehingga overhead manajemen data kecil.	**Manajemen Populasi Fleksibel:** Array sementara untuk benih (seed) memudahkan variasi jumlah individu di setiap iterasi.
Struktur Data Konsisten: Semua solusi bertipe Plant yang identik dalam array, sehingga proses copy dan sort sangat efisien di memori.	**Operator Overload pada Struct:** Weed memiliki operator < untuk sorting langsung di array, memudahkan pengurutan solusi secara efisien tanpa struktur tambahan.

Kekurangan	
Plant Propagation Algorithm (PPA)	**Invasive Weed Optimization (IWO)**
Sensitif Parameter: Kinerja sangat tergantung pada parameter populasi dan pemetaan fitness, sehingga perlu trial-error pada struktur data dan ukuran array.	**Penggunaan Array Besar:** Banyaknya benih dan populasi gabungan membuat penggunaan array sementara bisa sangat besar, berisiko boros memori pada masalah skala besar.
Stagnasi pada Populasi Mirip: Jika seluruh array solusi (plant) memiliki fitness seragam, proses sorting kurang efektif dalam menyeleksi solusi yang benar-benar berbeda, sehingga keragaman bisa turun.	**Kompleksitas Seleksi Naik Cepat:** Sorting dan seleksi di array besar (gabungan weed + seed) pada setiap iterasi meningkatkan overhead komputasi dibandingkan algoritma lain yang memakai struktur data dinamis.

9. Manfaat Algoritma PPA dan IWO dalam Menyelesaikan Permasalahan Optimasi

Algoritma PPA dan IWO, sebagai bagian dari keluarga besar Nature-Inspired Optimization Algorithms (NIOA), menawarkan sejumlah manfaat dalam menyelesaikan berbagai permasalahan optimisasi, termasuk masalah kompleks seperti Vertex Cover:

9.1. Kemampuan Mengatasi Masalah Kompleks (NP-hard)

Baik PPA maupun IWO dirancang sebagai metaheuristik yang dapat memberikan solusi berkualitas baik (mendekati optimal) untuk masalah-masalah di mana pencarian solusi optimal secara eksak tidak praktis karena kompleksitas waktu yang sangat tinggi (misalnya, NP-hard seperti Vertex Cover).

9.2. Pendekatan Berbasis Populasi

Kedua algoritma memelihara sekumpulan solusi (populasi) secara bersamaan. Ini memungkinkan eksplorasi yang lebih luas terhadap ruang pencarian dibandingkan

dengan metode pencarian titik tunggal, sehingga mengurangi risiko terjebak dalam optimum lokal yang buruk.

9.3. Keseimbangan Bawaan antara Eksplorasi dan Eksploitasi

- PPA: Secara eksplisit menyeimbangkan melalui mekanisme *runner* pendek (eksploitasi solusi yang baik) dan *runner* panjang (eksplorasi dari solusi yang kurang baik).
- IWO: Mencapai keseimbangan ini secara dinamis melalui penurunan standar deviasi penyebaran benih (dari eksplorasi ke eksploitasi) dan melalui reproduksi diferensial (solusi yang lebih baik lebih banyak bereproduksi).

9.4. Robustness terhadap Kondisi Awal

Karena sifat stokastik dan pencarian berbasis populasi, algoritma ini umumnya kurang sensitif terhadap pilihan solusi awal dibandingkan dengan metode optimasi lokal berbasis gradien. Mereka memiliki kemampuan untuk melepaskan diri dari solusi awal yang buruk.

10. Kesimpulan

Laporan ini membahas berbagai Algoritma Optimasi Terinspirasi Tumbuhan, dengan fokus pada Plant Propagation Algorithm (PPA) dan Invasive Weed Optimization (IWO). Kedua algoritma ini diadaptasi untuk menyelesaikan masalah Vertex Cover dengan menggunakan representasi solusi berbasis array boolean di C++. Studi kasus dan analisis menunjukkan PPA dan IWO sama-sama mampu menyeimbangkan eksplorasi dan eksploitasi, serta memiliki kompleksitas per iterasi yang sebanding dan efisien untuk ukuran populasi kecil. Secara keseluruhan, algoritma yang terinspirasi tumbuhan seperti PPA dan IWO terbukti efektif dan adaptif untuk optimasi diskrit, serta menawarkan potensi pengembangan lebih lanjut untuk berbagai jenis masalah optimasi.

Daftar Pustaka

[1] A. Kumar, M. Nadeem, and H. Banka, "Nature inspired optimization algorithms: a comprehensive overview," Feb. 01, 2023, *Institute for Ionics*. doi: 10.1007/s12530-022-09432-6.

[2] A. Salhi, A. Salhi, and E. S. Fraga, "Nature-Inspired Optimisation Approaches and the New Plant Propagation Algorithm," 2011. [Online]. Available: https://www.researchgate.net/publication/252321319

[3] D. Kumar, B. G. R. Gandhi, and R. K. Bhattacharjya, "Introduction to Invasive Weed Optimization Method," in *Modeling and Optimization in Science and Technologies*, vol. 16, Springer, 2020, pp. 203–214. doi: 10.1007/978-3-030-26458-1_12.

[4] X. He, S. Zhang, and J. Wang, "A Novel Algorithm Inspired by Plant Root Growth with Self-similarity Propagation," 2015.

[5] M. S. Kiran, "TSA: Tree-seed algorithm for continuous optimization," *Expert Syst Appl*, vol. 42, no. 19, pp. 6686–6698, May 2015, doi: 10.1016/j.eswa.2015.04.055.

Lampiran

No	Nama	Tautan
	Implementasi C++ PPA	
	Implementasi C++ PPA	
	Contoh Input	
	Slide Presentasi	