SKRIPSI

IMPLEMENTASI ALGORITMA *GREY WOLF OPTIMIZER* PADA PERMAINAN *LIGHT UP*

Vincent Kurniawan

NPM: 6181901024

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2023

UNDERGRADUATE THESIS

IMPLEMENTATION OF GREY WOLF OPTIMIZER ALGORITHM IN THE LIGHT UP GAME

Vincent Kurniawan

NPM: 6181901024

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2023

LEMBAR PENGESAHAN

IMPLEMENTASI ALGORITMA *GREY WOLF OPTIMIZER* PADA PERMAINAN *LIGHT UP*

Vincent Kurniawan

NPM: 6181901024

Bandung, 6 Juli 2023

Menyetujui,

Pembimbing

Digitally signed by Lionov

Lionov, Ph.D.

Ketua Tim Penguji

Digitally signed by Raymond Chandra Putra

Raymond Chandra Putra, M.T.

Anggota Tim Penguji

Digitally signed by Husnul Hakim

Husnul Hakim, M.T.

Mengetahui,

Ketua Program Studi

Digitally signed by Mariskha Tri Adithia

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

IMPLEMENTASI ALGORITMA *GREY WOLF OPTIMIZER* PADA PERMAINAN *LIGHT UP*

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal 6 Juli 2023

Vincent Kurniawan
NPM: 6181901024

ABSTRAK

Light Up merupakan permainan papan dua dimensi yang dimainkan oleh satu pemain. Setiap petak pada papan terdiri dari warna hitam atau putih. Petak hitam dapat terdiri dari angka 0–4 atau tidak terdapat angka sama sekali. Angka pada petak hitam merujuk pada jumlah lampu yang perlu diletakkan di atas, bawah, kiri, atau kanan petak hitam tersebut. Pemain hanya dapat meletakkan lampu pada petak putih. Lampu yang diletakkan pemain menyinari setiap petak putih di posisi horizontal dan vertikal dari petak yang diletakkan lampu tetapi tidak terhalang oleh petak hitam. Permainan dinyatakan selesai hanya jika seluruh petak putih tersinari pada papan, jumlah lampu yang diletakkan di sekeliling petak hitam berangka sesuai dengan angka yang tertera pada petak hitam tersebut, dan lampu yang diletakkan tidak menyinari lampu lainnya pada papan. Algoritma *Grey Wolf Optimizer* (GWO) diimplementasikan dalam pencarian solusi permainan *Light Up* pada skripsi ini.

Grey Wolf Optimizer (GWO) merupakan algoritma *population based meta-heuristics* yang menyimulasikan hierarki kepemimpinan dan perilaku serigala abu-abu dalam berburu di alam liar. Setiap serigala yang berada di dalam populasi memiliki posisinya masing-masing, sehingga perpindahan posisi setiap serigala memungkinkan penjelajahan ruang pencarian. Pada skripsi ini, Algoritma GWO diimplementasikan dalam dua tahap. Pada tahap pertama Algoritma GWO digunakan untuk mencari posisi peletakkan lampu di sekeliling petak hitam berangka 1 sampai 4, kemudian pada tahap kedua Algoritma GWO digunakan untuk mencari posisi peletakkan lampu di petak putih yang belum tersinari oleh lampu dari tahap pertama. Kondisi peletakkan lampu pada papan dihitung menggunakan fungsi evaluasi yang menghasilkan nilai *fitness*. Hukuman diberikan hanya jika kondisi peletakkan lampu pada papan melanggar aturan atau heuristik permainan. Hukuman diberikan dengan menambahkan nilai *fitness*. Nilai hukuman dihitung menggunakan bobot hukuman yang dapat diatur. *Preprocessing* sebelum tahap pertama, setelah tahap pertama, dan setelah tahap kedua dilakukan dengan memanfaatkan heuristik permainan yang ditemukan sehingga ruang pencarian solusi diperkecil.

Input papan diambil melalui situs penyedia papan *Light Up*. Selenium digunakan untuk mengambil *input* papan secara otomatis dengan menggunakan *web scraping*. Selain digunakan untuk mengambil *input* papan, Selenium juga digunakan untuk menyimulasikan model dengan melakukan interaksi pada situs permainan, sehingga model dapat bermain secara otomatis pada situs permainan. Berbagai eksperimen dilakukan pada *dataset input* papan untuk melihat pengaruh parameter yang digunakan dan mengukur kemampuan model dalam menyelesaikan seluruh jenis papan. Hasil eksperimen menunjukkan bahwa model yang dirancang mampu menyelesaikan papan hingga ukuran 25×25 dengan tingkat kesulitan *easy* hanya jika *preprocessing* digunakan, sedangkan apabila *preprocessing* tidak digunakan, maka model yang dirancang mampu menyelesaikan papan hingga ukuran 14×14 dengan tingkat kesulitan *easy*.

Kata-kata kunci: *Light Up*, *Grey Wolf Optimizer*

ABSTRACT

Light Up is a two-dimensional board game played by a single player. Each square on the board is either black or white, with black squares containing numbers from 0 to 4 or no number at all. The numbers on black squares indicate the number of lights that need to be placed above, below, to the left, or to the right of that black square. Players can only place lights on white squares. Placed lights illuminate every white square in horizontal and vertical positions from the placed light, as long as they are not blocked by black squares. The game is considered complete only if all white squares are illuminated on the board, the number of lights placed around black squares matches the numbers on those black squares, and placed lights do not illuminate other lights on the board. In this thesis, the Grey Wolf Optimizer (GWO) algorithm is implemented to find solutions for the Light Up game.

The Grey Wolf Optimizer (GWO) is a population-based metaheuristic algorithm that simulates the hierarchical leadership and hunting behavior of grey wolves in the wild. Each wolf in the population has its own position, allowing them to explore the search space by changing their positions. In this thesis, the GWO algorithm is implemented in two stages. In the first stage, it is used to find the positions to place lights around black squares with numbers 1 to 4. Then, in the second stage, it is used to find positions to place lights on white squares that are not already illuminated by lights from the first stage. The placement of lights on the board is evaluated using a fitness function, and penalties are applied only if the light placement violates the rules or heuristics of the game, with penalty values determined by adjustable penalty weights. Preprocessing is performed before the first stage, after the first stage, and after the second stage, utilizing discovered game heuristics to reduce the solution search space.

Board input is obtained from a Light Up board provider website, and Selenium is used for automated web scraping to collect board inputs. Selenium is also employed to simulate model interactions with the game website, enabling the model to play the game automatically on the website. Various experiments are conducted on input board datasets to assess the impact of different parameters and measure the model's ability to solve various types of boards. Experimental results indicate that the designed model can solve boards up to a size of 25x25 with an easy difficulty level when preprocessing is applied, while without preprocessing, the model can solve boards up to 14x14 with an easy difficulty level.

Keywords: Light Up, Grey Wolf Optimizer

Skripsi ini dipersembahkan untuk penulis, keluarga, dan teman.

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Allah Yesus Kristus atas berkat dan rahmat-Nya yang tak terhingga sehingga penulis dapat menyelesaikan penulisan skripsi ini yang berjudul "Implementasi Algoritma *Grey Wolf Optimizer* pada Permainan *Light Up*". Skripsi ini tidak dapat diselesaikan tanpa bantuan dari berbagai pihak, sehingga penulis hendak memberikan terima kasih yang sebesar-besarnya kepada:

1. Iwan Kurniawan Wahyudi selaku ayah dari penulis, Linna Mandarin selaku ibu dari penulis, Kimberly Kurniawan selaku adik perempuan dari penulis, Ferdinand Kurniawan selaku adik laki-laki dari penulis, dan Yeyen Wahyudi selaku nenek dari penulis yang selalu memberikan dukungan secara material, moral, dan mental sehingga skripsi ini dapat terselesaikan.
2. Bapak Lionov, Ph.D. selaku dosen pembimbing utama dari penulis yang telah membimbing serta memberikan kritik dan saran sehingga skripsi ini dapat terselesaikan.
3. Bapak Raymond Chandra Putra, M.T. dan Bapak Husnul Hakim, M.T. selaku dosen penguji yang telah memberikan kritik dan saran sehingga skripsi ini dapat terselesaikan.
4. Bapak Keenan Adiwijaya Leman, M.T. dan Ibu Maria Veronica, M.T. selaku teman penulis yang telah memberikan kritik dan saran sehingga skripsi ini dapat terselesaikan.
5. Nadia Clarissa Hermawan selaku mantan pacar penulis yang telah menyemangati dan memberikan perhatiannya sehingga skripsi ini dapat terselesaikan.
6. Vincentius Daryl Kurniawan dan Clementheo Chanson selaku teman baik penulis yang telah memberikan dukungan secara mental dan moral ketika penulis sedang terpuruk sehingga skripsi ini dapat terselesaikan.
7. Yalvi Hidayat selaku teman baik penulis yang telah memberikan kritik dan saran.
8. Frans Hiemawan, Jovannie Lastelinia, Ardy Alexander Heryanto, Michelle Kurniawan, Alessandro Hans Trisna Putra, Thorino Johansen, Ivan, Geraldo Adrian Stanis, Kevin Valentino, Evelyn Jonathan, Gheraldy Pranada, dan Alfian Fenardi selaku teman penulis yang telah memberikan inspirasi sehingga skripsi ini dapat terselesaikan.
9. Dimas Kurniawan, Geraldi Akira Surya, Lucyus Matthew Ardivan, Alexander Bleuvito Fevrier, dan Dearen Hippy selaku teman kerja penulis sebagai Admin Laboratorium FTIS UNPAR yang telah memberikan kritik dan saran sehingga skripsi ini dapat terselesaikan.

Bandung, Juli 2023

Penulis

DAFTAR ISI

Kata Pengantar xv

Daftar Isi xvii

Daftar Gambar xix

Daftar Tabel xxi

Daftar Kode Program xxiii

1 Pendahuluan 1
- 1.1 Latar Belakang . . . 1
- 1.2 Rumusan Masalah . . . 3
- 1.3 Tujuan . . . 5
- 1.4 Batasan Masalah . . . 5
- 1.5 Metodologi . . . 5
- 1.6 Sistematika Pembahasan . . . 5

2 Landasan Teori 7
- 2.1 Permainan *Light Up* . . . 7
- 2.2 Algoritma *Grey Wolf Optimizer* . . . 10
- 2.3 Implementasi Algoritma *Nested Two Step Evolutionary* . . . 15
- 2.4 Algoritma *Roulette Wheel Selection* . . . 18
- 2.5 Pickle . . . 20
- 2.6 Selenium . . . 21

3 Analisis 23
- 3.1 Heuristik Permainan . . . 23
- 3.2 Pengambilan Input Papan *Light Up* . . . 27
  - 3.2.1 Analisis Alamat Situs . . . 27
  - 3.2.2 Analisis Struktur HTML Situs Permainan . . . 27
  - 3.2.3 Proses *Web Scraping* . . . 29
- 3.3 Pembuatan dan Penggunaan *Dataset Input* Papan *Light Up* . . . 31
  - 3.3.1 Pembuatan *Dataset Input* Papan *Light Up* . . . 31
  - 3.3.2 Penggunaan *Dataset Input* Papan *Light Up* . . . 31
- 3.4 Pemodelan Papan *Light Up* . . . 31
- 3.5 Rancangan Implementasi Algoritma GWO . . . 32
  - 3.5.1 Dua Tahap Implementasi Algoritma GWO . . . 33
  - 3.5.2 Vektor Posisi . . . 34
  - 3.5.3 *Preprocessing* Papan . . . 38
  - 3.5.4 Perhitungan Nilai *Fitness* . . . 40
  - 3.5.5 Pemilihan Kandidat . . . 43
- 3.6 Tingkat Kesulitan Papan . . . 45

3.7 Peletakkan Dua atau Lebih Lampu pada Sebuah Petak Putih . . . 46
3.8 Contoh Proses Pencarian Solusi . . . 46
3.9 Simulasi Model pada Situs Permainan *Light Up* . . . 48

4 Rancangan Perangkat Lunak 51
4.1 Diagram *Use Case* . . . 51
4.2 Rancangan *Input* dan *Output* . . . 51
4.3 Diagram Kelas . . . 52
4.4 Proses Pencarian Solusi . . . 57
4.5 Proses Menyimulasikan Model pada Situs Permainan *Light Up* . . . 57
4.6 Modifikasi Algoritma *Roulette Wheel Selection* . . . 60
4.7 Diagram *Sequence* . . . 60

5 Implementasi 67
5.1 Lingkungan Implementasi . . . 67
5.2 Implementasi Pemodelan Papan . . . 67
5.3 Implementasi Penetapan Kondisi Sekeliling Petak . . . 68
5.4 Implementasi Pemberian Tanda Silang pada Papan . . . 69
5.5 Implementasi Peletakkan Lampu . . . 70
5.6 Implementasi Mencari Kemungkinan Posisi Peletakkan Lampu . . . 70
5.7 Implementasi Peletakkan Lampu Sesuai Vektor Posisi pada Tahap Pertama . . . 71
5.8 Implementasi Peletakkan Lampu Sesuai Vektor Posisi pada Tahap Kedua . . . 73
5.9 Implementasi *Preprocessing* Papan . . . 74
5.10 Implementasi Perhitungan Nilai *Fitness* . . . 76
5.11 Implementasi Proses Pencarian Solusi . . . 77
5.12 Implementasi Proses Interaksi dengan Situs Permainan . . . 81

6 Eksperimen 83
6.1 Lingkungan Eksperimen . . . 83
6.2 Hasil Eksperimen . . . 83
6.2.1 Eksperimen Pengaruh Jumlah Populasi . . . 83
6.2.2 Eksperimen Pengaruh *Epoch* . . . 84
6.2.3 Eksperimen Pengaruh *Rank* . . . 85
6.2.4 Eksperimen Pengaruh Menghilangkan Bobot Hukuman Heuristik . . . 85
6.2.5 Eksperimen Mengukur Kemampuan Model . . . 87
6.2.6 Analisis Eksperimen . . . 87
6.3 Kesimpulan Eksperimen . . . 89
6.4 Hasil Simulasi Model pada Situs Permainan . . . 91

7 Kesimpulan dan Saran 93
7.1 Kesimpulan . . . 93
7.2 Saran . . . 94

Daftar Referensi 95

A Kode Program 97

DAFTAR GAMBAR

1.1 Perbandingan antara kondisi papan sebelum diletakkan lampu, kondisi papan setelah diletakkan lampu, dan kondisi papan yang telah mengikuti seluruh aturan permainan. 2
1.2 Papan yang dibuat secara acak dan tidak memiliki solusi.[1] 3
1.3 Halaman situs penyedia papan. 4

2.1 Contoh peletakkan lampu pada papan. 7
2.2 Contoh peletakkan lebih dari satu lampu pada papan. 8
2.3 Perbandingan antara peletakkan lampu yang tidak saling menyinari dan peletakkan lampu yang saling menyinari. 9
2.4 Perbandingan antara jumlah lampu di sekeliling petak `f2` yang sesuai dan tidak sesuai dengan angka pada petak `f2`. 9
2.5 Perbandingan antara kondisi papan yang telah mengikuti aturan permainan *Light Up* dan kondisi papan yang melanggar aturan permainan *Light Up*. 10
2.6 Tingkat hierarki serigala abu-abu. 10
2.7 Tahapan berburu serigala abu-abu 11
2.8 Perbandingan antara vektor posisi dua dimensi dan tiga dimensi, serta kemungkinan perpindahan posisi berikutnya. 13
2.9 Perbandingan antara serigala abu-abu ketika menyerang mangsa dan mencari mangsa. 14
2.10 Contoh *preprocessing* dan *encoding* setelah tahap *preprocessing*; (a) Kondisi awal papan; (b) Tahap pertama dari *preprocessing*; (c) Tahap kedua dari *preprocessing*; (d) Kondisi papan setelah dilakukan *preprocessing*; (e) *Encoding* dari petak putih yang belum tersinari lampu; (f) Solusi akhir permainan 15
2.11 Ukuran ruang pencarian dari penggunaan *binary encoding*; (a) Kondisi awal papan; (b) Kondisi papan setelah dilakukan *preprocessing*; (c) *Encoding* dari ruang pencarian yang telah diperkecil dengan encoding pada petak putih di sekeliling petak hitam berangka pada papan. 16
2.12 Skema penggunaan Algoritma *Nested Two Steps Evolutionary*. 17
2.13 Kondisi tidak seluruh kombinasi peletakkan lampu di sekeliling petak hitam berangka adalah valid; (a) Kombinasi yang valid; (b) Kombinasi yang valid; (c) Kombinasi yang tidak valid; (d) Kombinasi yang tidak valid 17
2.14 Contoh dua solusi akhir papan yang ditemukan melalui tahap kedua pada penggunaan Algoritma *Nested Two Steps*. Solusi (c) didapatkan dari solusi pada tahap pertama (a) yang tidak dapat dijadikan solusi akhir papan. Solusi (d) didapatkan dari solusi pada tahap pertama (b) yang dapat dijadikan solusi akhir papan. 18
2.15 Contoh *Roulette wheel*. 19

3.1 Pemberian tanda silang pada papan. 24
3.2 Peletakkan lampu di sekeliling petak hitam berangka 2 pada sudut papan dan petak hitam berangka 3 pada tepi papan. 24
3.3 Proses peletakkan lampu pada sekeliling petak hitam berangka yang kemungkinan posisi peletakannya hanya satu. 25
3.4 Proses penyinaran petak putih yang bertanda silang yang hanya dapat tersinari melalui sebuah petak putih. 25

3.5 Pengaruh memprioritaskan peletakkan lampu di sekeliling petak hitam berangka. . 26
3.6 Peletakkan lampu pada petak putih yang dikelilingi oleh petak hitam. 26
3.7 Perbedaan antara papan yang dapat dilakukan *preprocessing* dengan papan yang tidak dapat dilakukan *preprocessing*. 27
3.8 Tampilan halaman utama situs penyedia papan *Light Up* 28
3.9 Tampilan dan struktur HTML papan.[2] 29
3.10 Identifikasi papan sesuai dengan Tabel 3.3. 32
3.11 Kemungkinan posisi peletakkan lampu di sekeliling petak hitam berangka 1. 34
3.12 Kemungkinan posisi peletakkan lampu di sekeliling petak hitam berangka 2. 34
3.13 Kemungkinan posisi peletakkan lampu di sekeliling petak hitam berangka 3. 35
3.14 Kemungkinan posisi peletakkan lampu di sekeliling petak hitam berangka 4. 35
3.15 Kondisi papan yang telah diletakkan lampu di sekeliling petak hitam berangka 1–4. 35
3.16 Kondisi papan yang memperlihatkan bahwa tidak seluruh angka kemungkinan posisi peletakkan lampu dapat digunakan. 36
3.17 Papan yang telah dilakukan *preprocessing*. 37
3.18 Hasil peletakkan lampu dengan vektor posisi `[2]` dari papan pada Gambar 3.17. . 38
3.19 Hasil akhir papan dari peletakkan lampu pada Gambar 3.17. 39
3.20 Proses pengolahan vektor posisi tahap pertama. 39
3.21 Proses pengolahan vektor posisi tahap kedua. 40
3.22 Papan yang dapat diselesaikan hanya dengan *preprocessing*. 41
3.23 Contoh *preprocessing* sebelum tahap kedua implementasi Algoritma GWO. 41
3.24 Kondisi papan yang melakukan pelanggaran. 43
3.25 Perbedaan posisi peletakkan lampu pada tahap pertama implementasi Algoritma GWO menghasilkan hasil akhir yang berbeda. 44
3.26 Papan dengan ukuran 7×7 pada tingkat kesulitan (a) *easy*, (b) *normal*, dan (c) *hard*. 46
3.27 Contoh peletakkan dua atau lebih lampu pada petak putih yang sama. 47
3.28 Contoh proses pencarian solusi pada sebuah papan. 48
3.29 Tampilan Permainan *Light Up*. 49

4.1 Diagram *use case* yang digunakan pada skripsi ini. 51
4.2 Diagram kelas program. 56
4.3 Diagram *sequence* melakukan eksperimen pada *dataset input* papan. 62
4.4 Diagram *sequence* simulasi model pada situs permainan *Light Up*. 63
4.5 Diagram *sequence* fungsi `firstGWO()`. 64
4.6 Diagram *sequence* fungsi `secondGWO()`. 65

6.1 Grafik garis perbandingan antara rata-rata nilai *fitness* dan jumlah papan solusi untuk setiap jumlah populasi. 84
6.2 Grafik garis perbandingan antara rata-rata nilai *fitness* dan jumlah papan solusi untuk setiap *epoch*. 85
6.3 Grafik garis perbandingan antara jumlah papan selesai dan rata-rata nilai *fitness* dengan pengaruh *rank*. 86
6.4 Rata-rata nilai *fitness* pada pengaruh menghilangkan bobot hukuman heuristik dalam hubungannya dengan jumlah populasi. 90
6.5 *Screenshot* hasil simulasi model pada situs permainan yang menemukan solusi untuk jenis papan 25×25 *easy* dengan menggunakan *preprocessing*. 91

DAFTAR TABEL

2.1 Tabel hasil persentase *convergence* penggunaan Algoritma *Nested Two Steps Evolutionary* 19

3.1 Tabel alamat situs pada setiap ukuran dan tingkat kesulitan papan 28
3.2 Tabel atribut kelas dan isi elemen `div` untuk setiap jenis petak 29
3.3 Tabel jenis petak dengan angka identifikasinya. 32
3.4 Tabel perbedaan nilai komponen perhitungan hukuman total. 43

6.1 Spesifikasi *laptop* 83
6.2 Hasil eksperimen pengaruh menghilangkan bobot hukuman heuristik dengan menggunakan *preprocessing* pada papan 86
6.3 Hasil eksperimen pengaruh menghilangkan bobot hukuman heuristik tanpa menggunakan *preprocessing* pada papan 86
6.4 Hasil eksperimen mengukur kemampuan model dengan menggunakan *preprocessing* 88
6.5 Hasil eksperimen mengukur kemampuan model tanpa *preprocessing* 88

DAFTAR KODE PROGRAM

2.1 Proses penyimpanan variabel ke dalam sebuah *file* . . . 21
2.2 Proses membaca variabel yang tersimpan pada *file* . . . 21
2.3 Proses penekanan tombol pada sebuah situs . . . 21

3.1 Kode penggunaan Selenium dalam mengambil informasi papan. . . . 30
3.2 Fungsi pengolah informasi papan. . . . 30
3.3 Nilai identifikasi petak. . . . 30
3.4 Penyimpanan variabel kumpulan papan pada sebuah *file*. . . . 31
3.5 Penggunaan modul Pickle dalam membaca *dataset input* papan. . . . 31

5.1 Kelas `Cell` . . . 67
5.2 *Constructor* kelas `Board` dan fungsi `makeBoard` . . . 68
5.3 Fungsi `setAvailability` . . . 68
5.4 Fungsi `setAvailable` dan `checkAvailable` . . . 69
5.5 Fungsi `setForbidden` dan `setType` . . . 69
5.6 Fungsi `putSingleLamp` . . . 70
5.7 Fungsi `getRandomBlack` . . . 71
5.8 Fungsi `findBlackPosition` . . . 71
5.9 Fungsi `updateBlackPosition` dan `putBlackLamps` . . . 72
5.10 Fungsi `saveBoard` . . . 73
5.11 Fungsi `findWhitePosition` . . . 73
5.12 Fungsi `updateWhitePosition` dan `putWhiteLamps` . . . 74
5.13 Fungsi `preproc` . . . 74
5.14 Fungsi `secondPreproc` . . . 75
5.15 Fungsi `lastPreproc` . . . 76
5.16 Fungsi `updateFitness` pada kelas `Wolf` dan kelas `Board` . . . 77
5.17 Fungsi `firstGWO` . . . 78
5.18 Fungsi `roulette` . . . 78
5.19 Fungsi `secondGWO` . . . 79
5.20 Fungsi `experiment` . . . 80
5.21 Fungsi `play` . . . 80
5.22 Fungsi `clickBoard` . . . 81
5.23 Proses menekan tombol `Done` dan New Puzzle dengan menggunakan Selenium . . . 81
5.24 Proses mengambil *screenshot* dengan menggunakan Selenium . . . 81

A.1 InputGathering.py . . . 97
A.2 Cell.py . . . 99
A.3 Board.py . . . 99
A.4 Wolf.py . . . 103
A.5 Game.py . . . 104
A.6 Main.py . . . 106
A.7 Play.py . . . 107

BAB 1

PENDAHULUAN

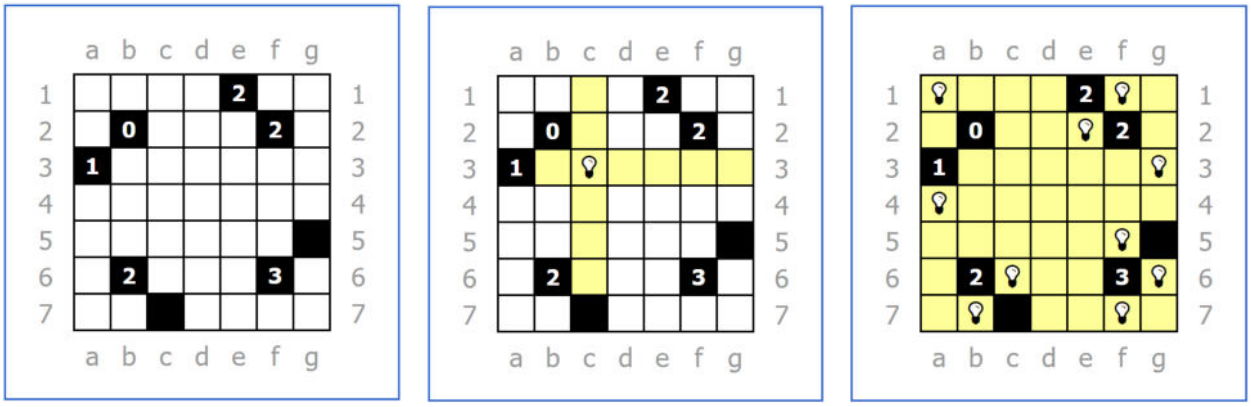
1.1 Latar Belakang

Permainan diciptakan oleh manusia dengan beberapa tujuan seperti melatih logika berpikir, melatih keterampilan dalam memecahkan masalah, kegiatan pengisi waktu luang, serta kegiatan yang dapat dijadikan sebagai hobi baru. Salah satu jenis permainan yang dapat melatih logika berpikir adalah permainan logika. Terdapat beberapa macam permainan logika yang cukup populer, seperti *Sudoku*, *Minesweeper*, dan *Light Up*. Permainan *Light Up* digunakan pada skripsi ini. Permainan *Light Up* disediakan oleh beberapa situs permainan seperti https://www.interactive.onlinemathlearning.com/fun_lightup.php, https://www.brainbashers.com/, dan https://www.puzzle-light-up.com/. Situs permainan https://www.puzzle-light-up.com/ digunakan pada skripsi ini, serta seluruh gambar papan *Light Up* yang ditampilkan pada skripsi ini diambil melalui situs permainan tersebut.

Light Up merupakan permainan papan dua dimensi dengan tampilan berupa *grid of cells* yang berukuran $N \times M$, di mana N merupakan jumlah baris dan M merupakan jumlah kolom dari *grid*. Setiap petak pada papan terdiri dari warna putih atau hitam. *Light Up* dimainkan oleh satu pemain. di mana pemain hanya dapat meletakkan lampu pada petak putih. Peletakkan lampu pada sebuah petak putih mengakibatkan seluruh petak putih yang berada pada baris atau kolom yang sama dengan petak putih tersebut namun tidak terhalangi oleh petak hitam tersinari. Permainan *Light Up* dinyatakan selesai hanya jika pemain telah mengikuti beberapa aturan dari permainan *Light Up*, yaitu setiap lampu yang diletakkan pada petak putih tidak diperbolehkan menyinari lampu lainnya pada papan, jumlah lampu di atas, bawah, kiri, dan kanan dari petak hitam berangka diharuskan sesuai dengan angka pada petak hitam tersebut, dan seluruh petak putih pada papan diharuskan tersinari oleh lampu. Sebagai contoh, Gambar 1.1a menunjukkan kondisi papan sebelum diletakkan lampu, Gambar 1.1b menunjukkan kondisi papan setelah diletakkan lampu, dan Gambar 1.1c menunjukkan kondisi papan yang telah mengikuti seluruh aturan permainan.

Tujuan dari skripsi ini adalah merancang model yang dapat mencari solusi peletakkan lampu pada papan *Light Up* yang berukuran $N \times M$. Model dapat dirancang dengan memanfaatkan implementasi algoritma tertentu, namun tidak seluruh algoritma dapat digunakan dalam merancang model tersebut, hal ini disebabkan ketika ukuran papan dan jumlah lampu yang perlu diletakkan meningkat, implementasi dari algoritma eksak seperti *brute force* membutuhkan waktu pencarian yang cukup lama dan bahkan tidak memungkinkan. Mengetahui bahwa kondisi petak putih berupa diletakkan lampu atau tidak diletakkan lampu, sehingga algoritma *brute force* memiliki ruang pencarian berupa 2^n, di mana n merupakan jumlah petak putih pada papan. Sebagai contoh, pada papan yang memiliki 43 petak putih, maka dibutuhkan sekitar 2^{43} atau 8.796093×12 kemungkinan.

Mengetahui bahwa tidak seluruh algoritma dapat diimplementasikan dalam merancang model pencari solusi, maka algoritma *meta-heuristic* digunakan dalam merancang model pencari solusi. Meta-heuristics merupakan jenis algoritma yang digunakan untuk mencari solusi perkiraan pada masalah optimisasi. Algoritma *meta-heuristic* umumnya dapat memberikan solusi yang optimal dalam waktu yang wajar. Terdapat beberapa algoritma *meta-heuristic* yang telah digunakan dalam penelitian permainan *Light Up*, seperti *Ant Colony Optimization* [1], Algoritma Genetik [2],



(a) Kondisi papan sebelum diletakkan lampu.

(b) Kondisi papan setelah diletakkan lampu.

(c) Kondisi papan yang telah mengikuti seluruh aturan permainan.

Gambar 1.1: Perbandingan antara kondisi papan sebelum diletakkan lampu, kondisi papan setelah diletakkan lampu, dan kondisi papan yang telah mengikuti seluruh aturan permainan.

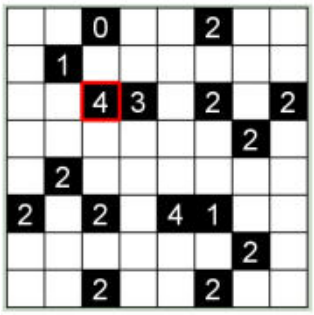
dan *Particle Swarm Optimization* [3]. Dari hasil studi literatur yang sudah dilakukan, algoritma *meta-heuristic* lain seperti *Grey Wolf Optimizer* belum pernah diteliti dalam implementasinya pada permainan *Light Up*, sehingga pada skripsi ini Algoritma *Grey Wolf Optimizer* diimplementasikan dalam pencarian solusi permainan *Light Up*.

Menurut Seyedali [4] *Grey Wolf Optimizer* (GWO) merupakan algoritma *population based meta-heuristics* yang menyimulasikan hierarki kepemimpinan dan perilaku serigala abu-abu dalam berburu di alam liar. Algoritma GWO bertujuan untuk mencari solusi optimal pada permasalahan pengoptimalan dengan mengadopsi perilaku serigala abu-abu yang memiliki tingkatan hierarki dalam kelompoknya. Terdapat 4 tingkatan hierarki serigala abu-abu dalam kelompoknya, yaitu serigala *alpha* yang merupakan kandidat solusi terbaik, diikuti oleh serigala *beta* yang merupakan kandidat solusi kedua terbaik, dan serigala *delta* yang merupakan kandidat solusi ketiga terbaik. Melalui interaksi dan kolaborasi antarindividu dalam populasi, Algoritma GWO mampu menjelajahi ruang pencarian secara efisien dan mendekati solusi terbaik. Setiap serigala yang berada di dalam populasi memiliki posisinya masing-masing, sehingga perpindahan posisi setiap serigala memungkinkan penjelajahan ruang pencarian. Penjelajahan ruang pencarian merupakan tahap serigala abu-abu berburu mangsa. Perpindahan posisi setiap serigala di dalam populasi dipengaruhi oleh posisi dari serigala *alpha*, serigala *beta* dan serigala *delta*.

Pada implementasi Algoritma GWO, posisi setiap serigala merupakan kandidat solusi, sehingga dalam menentukan tingkatan solusi terbaik hingga terburuk dibutuhkan sebuah fungsi evaluasi yang dapat mengkuantifikasi keadaan. Fungsi evaluasi yang dirancang perlu menghasilkan sebuah nilai yang disebut sebagai nilai *fitness*. Setiap serigala memiliki nilai *fitness*-nya masing–masing, sehingga nilai *fitness* dapat digunakan sebagai ukuran terkait baik-buruknya posisi sebuah serigala. Fungsi evaluasi yang dirancang pada skripsi ini memanfaatkan bobot hukuman, sehingga apabila posisi sebuah serigala melanggar aturan atau heuristik dari permainan *Light Up*, maka serigala tersebut diberikan hukuman sesuai bobot hukuman yang ditentukan. Pemberian hukuman untuk setiap jenis pelanggaran yang dilakukan oleh serigala meningkatkan nilai *fitness* serigala tersebut, sehingga solusi dicari dengan meminimalkan nilai *fitness*.

Pengukuran kemampuan model yang dirancang untuk mencari solusi papan memerlukan *dataset input* papan yang masih belum diletakkan oleh lampu. Pembuatan sebuah papan *Light Up* tidak dapat dilakukan secara acak, hal ini disebabkan karena pembuatan papan perlu mempertimbangkan aturan permainan sehingga terdapat solusi. Sebagai contoh, Gambar 1.2 merupakan papan yang dibuat secara acak sehingga tidak dapat ditemukan solusi, hal ini dapat dibuktikan dengan melihat petak yang ditandai dengan garis berwarna merah. Petak tersebut merupakan petak hitam berangka 4, sehingga diperlukan 4 buah lampu di sekelilingnya, namun pada sisi kanan petak tersebut terdapat

sebuah petak hitam, sehingga jumlah lampu yang dapat diletakkan di sekeliling petak tersebut hanyalah 3 buah. Mengetahui bahwa pembuatan *dataset input* papan tidak dapat dilakukan secara acak, maka diperlukan sumber yang menyediakan papan *Light Up* dalam berbagai ukuran, serta papan yang disediakan oleh sumber tersebut perlu dipastikan memiliki solusi.



Gambar 1.2: Papan yang dibuat secara acak dan tidak memiliki solusi.[1]

Sumber penyedia papan *Light Up* yang digunakan pada skripsi ini adalah situs permainan *Light Up*. Situs permainan *Light Up* menyediakan berbagai ukuran papan seperti 7×7, 10×10, 14×14, 25×25, 30×30, 30×40, dan 40×50. *Screenshot* halaman utama dari situs yang digunakan dapat terlihat melalui Gambar 1.3. Pengambilan papan dari situs tersebut dapat dilakukan dengan dua cara, yaitu menyalin secara manual atau membuat aplikasi yang dapat berjalan secara otomatis dalam menyalin papan pada situs tersebut. Apabila pengambilan papan dari situs tersebut dilakukan secara manual, maka penyalinan papan memakan cukup banyak waktu dan rawan terjadi kesalahan. Mempertimbangkan papan yang diambil tidak hanya satu melainkan cukup banyak, oleh karena itu penyalinan papan dengan memanfaatkan aplikasi yang dapat berjalan secara otomatis adalah pilihan terbaik. Penyalinan papan secara otomatis dapat dilakukan dengan metode *web scraping*. *Web scraping* merupakan metode untuk mengambil informasi dari *file* HTML sebuah halaman situs.[2] *Web scraping* dapat dilakukan dengan memanfaatkan beberapa *library* seperti Selenium, Beautiful Soup, dan Scrapy.[3] Pada skripsi ini digunakan Selenium untuk menyalin papan dari situs penyedia papan secara otomatis.

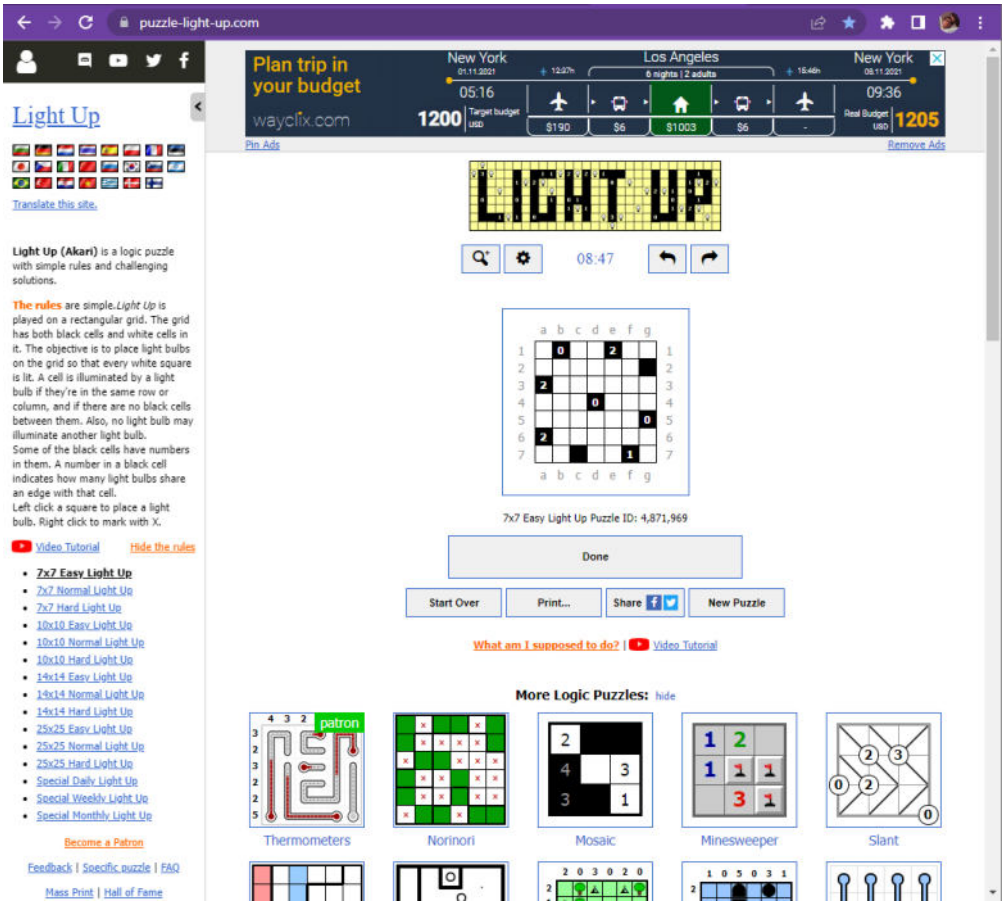
Setiap papan yang disalin menggunakan Selenium disimpan dalam *dataset* yang berbentuk *file*, sehingga *dataset input* papan dapat terbuat. Pada skripsi ini, pembuatan *dataset input* papan memanfaatkan modul Pickle yang disediakan oleh Python. Selain penggunaan Selenium dalam pengambilan *input* papan, Selenium juga dapat digunakan untuk melakukan interaksi berupa penekanan tombol kiri *mouse* dan penekanan tombol kanan *mouse* pada situs.[4] Pada skripsi ini, Selenium digunakan untuk menyimulasikan model pada situs permainan *Light Up*. Pada skripsi ini, Selenium diimplementasikan dalam bahasa pemrograman Python, serta *library* NumPy yang disediakan oleh Python digunakan pada proses pencarian solusi.

1.2 Rumusan Masalah

Berdasarkan latar belakang permasalahan yang dijelaskan pada Bab 1, berikut merupakan masalah–masalah yang hendak diselesaikan oleh skripsi ini:

1. Bagaimana pemodelan papan *Light Up* sehingga dapat digunakan dalam pencarian solusi?
2. Bagaimana implementasi Algoritma GWO dalam pencarian solusi papan *Light Up*?
3. Bagaimana merancang fungsi evaluasi yang dapat mengkuantifikasi kondisi papan saat ini?

[1] *Screenshot* diambil dari aplikasi Crossword Express—aplikasi pembuat *grid* beberapa permainan seperti *Minesweeper*, *Sudoku*, dan *Light Up*, yang dapat diunduh melalui situs http://www.crauswords.com/download.html.
[2] https://en.wikipedia.org/wiki/Web_scraping
[3] https://www.projectpro.io/article/python-libraries-for-web-scraping/625
[4] https://www.selenium.dev/documentation/webdriver/elements/interactions/



Gambar 1.3: Halaman situs penyedia papan.

4. Bagaimana menyalin papan secara otomatis dari situs penyedia papan dan mengubahnya menjadi *input* yang dapat diolah?
5. Bagaimana menggunakan Selenium dalam menyimulasikan model pada situs permainan *Light Up*?

1.3 Tujuan

Berdasarkan rumusan masalah yang dijelaskan pada Bab 1.2, berikut tujuan dari skripsi ini:
1. Memodelkan papan *Light Up* sehingga dapat digunakan dalam proses pencarian solusi.
2. Mengimplementasikan Algoritma GWO dalam proses pencarian solusi papan *Light Up*.
3. Merancang fungsi evaluasi yang dapat mengkuantifikasi kondisi papan saat ini.
4. Menyalin papan secara otomatis dari situs penyedia papan dan mengubahnya menjadi *input* yang dapat diolah.
5. Menggunakan Selenium dalam menyimulasikan model pada situs permainan *Light Up*.

1.4 Batasan Masalah

Batasan masalah pada skripsi ini berupa papan *Light Up* hanya diambil melalui situs `https://www.puzzle-light-up.com/`, di mana papan *Light Up* yang disediakan oleh situs tersebut hanya terbatatas pada ukuran 7×7, 10×10, 14×14, 25×25, 30×30, 30×40, dan 40×50.

1.5 Metodologi

Berikut langkah-langkah yang dilakukan untuk mencapai tujuan pada skripsi ini:
1. Bermain berbagai ukuran dan tingkat kesulitan papan *Light Up*.
2. Melakukan studi literatur terkait pemodelan permainan *Light Up* yang dapat digunakan dalam pencarian solusi. Studi literatur dapat dilakukan dengan melihat contoh pemodelan pencarian solusi permainan lain yang sejenis.
3. Melakukan studi literatur terkait Algoritma *Grey Wolf Optimizer* dan penggunaannya pada pencarian solusi papan *Light Up*.
4. Melakukan studi literatur terkait heuristik permainan *Light Up* yang dapat digunakan dalam pencarian solusi.
5. Melakukan studi literatur terkait Selenium dengan penggunaannya pada pengambilan *input* papan dan menyimulasikan model pada situs permainan *Light Up*.
6. Membuat *dataset input* papan yang digunakan pada proses eksperimen.
7. Memodelkan papan *Light Up* sehingga dapat digunakan pada proses pencarian solusi.
8. Mengembangkan Algoritma *Grey Wolf Optimizer* yang dapat digunakan dalam proses pencarian solusi.
9. Memanfaatkan Selenium dalam menyimulasikan model pada situs permainan *Light Up*.
10. Melakukan eksperimen pada *dataset input* papan yang telah dibuat.
11. Melakukan analisis terhadap hasil eksperimen yang dilakukan.
12. Membuat kesimpulan dari analisis yang telah dilakukan.
13. Menulis dokumen skripsi.

1.6 Sistematika Pembahasan

Berikut susunan yang digunakan dalam memaparkan isi skripsi ini:
1. **Bab 1 Pendahuluan**
   Pada Bab 1 dijelaskan latar belakang masalah terkait permainan *Light Up*, pengambilan *dataset input* papan secara otomatis, penggunaan Algoritma GWO dalam pencarian solusi, dan

simulasi model pada situs permainan *Light Up*. Pada Bab 1 dijelaskan juga terkait rumusan masalah, tujuan penelitian, batasan masalah, metodologi, dan sistematika pembahasan pada skripsi ini.

2. **Bab 2 Landasan Teori**
   Pada Bab 2 dijelaskan landasan teori terkait permainan *Light Up*, Algoritma GWO, implementasi Algoritma *Nested Two Step Evolutionary* pada permainan *Light Up*, Pickle, NumPy, Selenium, dan Algoritma *Roulette Wheel Selection*.
3. **Bab 3 Analisis**
   Pada Bab 3 dijelaskan analisis terkait heuristik permainan yang dapat digunakan dalam proses pencarian solusi, pengambilan *input* papan *Light Up*, pembuatan dan penggunaan *dataset input* papan *Light Up*, pemodelan papan *Light Up*, rancangan implementasi Algoritma GWO, *preprocessing* papan, perhitungan nilai *fitness*, pemilihan kandidat, tingkat kesulitan papan, peletakkan dua atau lebih lampu pada sebuah petak putih, contoh proses pencarian solusi, dan simulasi model pada situs permainan *Light Up*.
4. **Bab 4 Rancangan Perangkat Lunak**
   Pada Bab 4 dijelaskan rancangan terkait diagram *use case*, rancangan *input* dan *output*, diagram kelas, proses pencarian solusi, proses menyimulasikan model pada situs permainan *Light Up*, modifikasi Algoritma *Roulette Wheel Selection*, dan diagram *sequence*.
5. **Bab 5 Implementasi**
   Pada Bab 5 dijelaskan implementasi terkait pemodelan papan, penetapan kondisi sekeliling petak, pemberian tanda silang pada papan, peletakkan lampu, mencari kemungkinan posisi peletakkan lampu, peletakkan lampu sesuai vektor posisi tahap pertama, peletakkan lampu sesuai vektor posisi tahap kedua, tahap *preprocessing*, perhitungan nilai *fitness*, proses pencarian solusi, dan proses interaksi dengan situs permainan.
6. **Bab 6 Eksperimen**
   Pada Bab 6 dijelaskan eksperimen terkait lingkungan eksperimen, hasil eksperimen, analisis eksperimen, dan kesimpulan dari eksperimen yang dilakukan. Hasil simulasi model diberikan pada bagian ini.
7. **Bab 7 Kesimpulan dan Saran**
   Pada Bab 7 dijelaskan kesimpulan berupa tujuan yang tercapai melalui skripsi ini dan saran yang dapat menjadi panduan apabila skripsi ini digunakan untuk penelitian lebih lanjut.