

SKRIPSI

**PERBANDINGAN ALGORITMA BACKTRACKING
DENGAN ALGORITMA HYBRID GENETIC UNTUK
MENYELESAIKAN PERMAINAN CALCUDOKU**



MICHAEL ADRIAN

NPM: 2013730039

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2017**

UNDERGRADUATE THESIS

**COMPARISON OF THE BACKTRACKING ALGORITHM
AND THE HYBRID GENETIC ALGORITHM TO SOLVE THE
CALCUDOKU PUZZLE**



MICHAEL ADRIAN

NPM: 2013730039

**DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND
SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2017**

ABSTRAK

Calcudoku adalah sebuah permainan teka-teki angka. Tujuan dari teka-teki ini adalah mengisi setiap sel dalam *grid* dengan angka 1 sampai n tanpa pengulangan angka dalam setiap kolomnya dan barisnya untuk *grid* berukuran $n \times n$. *Grid* ini dibagi menjadi sejumlah *cage* dengan setiap *cage* yang jumlah selnya bervariasi. Setiap *cage* dibatasi oleh garis yang lebih tebal daripada garis pembatas antar sel. Angka-angka dalam satu *cage* yang sama harus menghasilkan angka tujuan yang telah ditentukan jika dihitung menggunakan operasi matematika yang ditentukan. Angka-angka dalam satu *cage* juga boleh berulang, selama pengulangan tidak terjadi dalam satu kolom atau baris yang sama.

Dua algoritma telah terbukti berhasil dalam menyelesaikan Calcudoku, yaitu algoritma *backtracking* dan algoritma *hybrid genetic*.

Algoritma *backtracking* adalah sebuah algoritma umum yang mencari solusi dengan mencoba salah satu dari beberapa pilihan, jika pilihan yang dipilih ternyata salah, komputasi dimulai lagi pada titik pilihan dan mencoba pilihan lainnya.

Algoritma *hybrid genetic* dalam kasus ini adalah gabungan dari algoritma *rule based* dan algoritma genetik. Algoritma *rule based* adalah sebuah algoritma berbasis aturan logika untuk menyelesaikan Calcudoku. Algoritma genetik adalah salah satu teknik heuristik *Generate and Test* yang terinspirasi oleh sistem seleksi alam yang mencari solusi dengan menggunakan operator-operator genetik seperti mutasi, kawin silang, dan *elitism*.

Hasil dari penelitian ini adalah perangkat lunak algoritma *backtracking* dapat menyelesaikan semua permainan yang diujikan, tetapi pada ukuran *grid* yang besar, algoritma *backtracking* sangat lambat dalam menyelesaikan permainan, ada kemungkinan algoritma *hybrid genetic* gagal dalam menyelesaikan permainan karena sifat acak dari algoritma *hybrid genetic* ini, semakin besar ukuran *grid*, maka kemungkinan algoritma *hybrid genetic* gagal dalam menyelesaikan permainan semakin besar, pada ukuran *grid* yang kecil, algoritma *hybrid genetic* cenderung menyelesaikan permainan lebih lambat daripada algoritma *backtracking*, tetapi pada ukuran *grid* yang besar, algoritma *hybrid genetic* mungkin mampu menyelesaikan permainan lebih cepat daripada algoritma *backtracking*, tetapi hal ini tidak dapat dibuktikan karena algoritma *hybrid genetic* gagal dalam menyelesaikan permainan dengan ukuran *grid* yang besar, dan nilai untuk parameter-parameter algoritma genetik mempengaruhi kecepatan dan tingkat keberhasilan algoritma *hybrid genetic* dalam menyelesaikan permainan.

Kata-kata kunci: Calcudoku, algoritma *backtracking*, algoritma *hybrid genetic*, algoritma *rule based*, algoritma genetik

ABSTRACT

Calcudoku is a number puzzle game. The goal of this puzzle is to fill each cell in the grid with the numbers from 1 to n without repetitions of the numbers in each column and row for grid with the size of $n \times n$. The grid is divided into a number of cages, with each cage contains a variable number of cells. Each cage is bordered with a thicker line than cell border line. Numbers in the same cage must produced the predetermined target number if calculated using the predetermined mathematical operatin. Numbers in a cage can be repeated, as long as the repetitions do not occur in the same column or row.

Two algorithms have been proven to successfully solve Calcudoku. The two algorithms are the backtracking algorithm and the hybrid genetic algorithm.

The backtracking algorithm is a general algorithm with finds a solution by trying one of several choices, if the choice proves to be incorrect, the computation restarts at the point of choice and tries another choice.

In this case, the hybrid genetic algorithm is a combination of the rule based algorithm and the genetic algorithm. Rule based algorithm uses logical rules to solve Calcudoku. Genetic algorithm is a heuristic technique inspired by the process of natural selection, which tries to find a solution by relying on genetic operators such as mutation, crossover, and elitism.

The result of this research is that the backtracking algorithm solved all puzzles, but on large grids, the algorithm is very slow in solving the puzzle, there is a chance that the hybrid genetic algorithm failed in solving the puzzle due to the random nature of the algorithm, the larger the grid, the higher the chance that the algorithm will fail in solving the puzzle, on small grids, the hybrid genetic algorithm tends to solve the puzzle slower than the backtracking algorithm, but on larger grids, the hybrid genetic algorithm may be able to solve the puzzle faster than the backtracking algorithm, but this cannot be proven because the hybrid genetic algorithm failed in solving puzzles with large grids, and the values of the genetic algorithm parameters influences the speed and the success rate of the hybrid genetic algorithm in solving the puzzle.

Keywords: Calcudoku, backtracking algorithm, hybrid genetic algorithm, rule based algorithm, genetic algorithm

DAFTAR ISI

DAFTAR ISI	ix
DAFTAR GAMBAR	xii
DAFTAR TABEL	xvi
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Tujuan	4
1.4 Batasan Masalah	5
1.5 Metodologi Penelitian	5
1.6 Sistematika Pembahasan	5
2 LANDASAN TEORI	7
2.1 Calcudoku [1] [2]	7
2.2 Algoritma <i>Backtracking</i> [1]	10
2.3 Algoritma <i>Hybrid Genetic</i> [2]	16
2.3.1 Algoritma <i>Rule Based</i>	16
2.3.2 Algoritma Genetik	19
2.3.3 Algoritma <i>Hybrid Genetic</i>	20
3 ANALISIS	25
3.1 Analisis Algoritma <i>Backtracking</i>	25
3.2 Analisis Algoritma <i>Hybrid Genetic</i>	30
3.2.1 Algoritma <i>Rule Based</i>	30
3.2.2 Algoritma Genetik	30
3.3 Perangkat Lunak	43
3.3.1 Diagram <i>Use Case</i> dan Skenario	44
3.3.2 Diagram Kelas	48
3.3.3 Diagram <i>Sequence</i>	49
4 PERANCANGAN	61
4.1 Perancangan Masukan	61
4.2 Perancangan Keluaran	62
4.3 Perancangan Antarmuka	62
4.4 Diagram Kelas	64
4.4.1 Kelas Grid	66
4.4.2 Kelas Cage	70
4.4.3 Kelas Cell	73
4.4.4 Kelas SolverBacktracking	74
4.4.5 Kelas SolverHybridGenetic	75
4.4.6 Kelas SolverRuleBased	76

4.4.7	Kelas SolverGenetic	82
4.4.8	Kelas Chromosome	84
4.4.9	Kelas ChromosomeComparator	84
4.4.10	Kelas Controller	85
4.4.11	Kelas Calcudoku	87
4.4.12	Kelas WindowListener	89
4.4.13	Kelas PuzzleFileFilter	91
4.4.14	Kelas GUI	91
4.4.15	Kelas CellKeyListener	94
4.4.16	Kelas PopupMenuItemListener	94
4.4.17	Kelas CellTextFieldListener	95
4.4.18	Kelas GeneticParameters	96
5	IMPLEMENTASI DAN PENGUJIAN	99
5.1	Lingkungan untuk Pengujian	99
5.2	Implementasi	99
5.2.1	Kode Program	100
5.2.2	Antarmuka Perangkat Lunak	100
5.3	Pengujian Fungsional	103
5.4	Pengujian Keakuratan	113
5.5	Pengujian Algoritma <i>Backtracking</i>	116
5.6	Pengujian dan Eksperimen Algoritma <i>Hybrid Genetic</i>	116
5.6.1	Skenario 1	117
5.6.2	Skenario 2	118
5.6.3	Skenario 3	118
5.6.4	Skenario 4	118
5.6.5	Skenario 5	119
5.6.6	Skenario 6	119
5.6.7	Skenario 7	119
5.6.8	Skenario 8	120
5.6.9	Skenario 9	120
5.6.10	Skenario 10	120
5.6.11	Skenario 11	121
5.6.12	Skenario 12	121
5.6.13	Skenario 13	121
5.6.14	Skenario 14	122
5.6.15	Skenario 15	122
5.6.16	Skenario 16	122
5.6.17	Hasil Eksperimen	123
6	SIMPULAN DAN SARAN	125
6.1	Simpulan	125
6.2	Saran	127
DAFTAR REFERENSI		129
A	ANALISIS ALGORITMA <i>Backtracking</i>	131
B	HASIL PENGUJIAN	145
B.1	Algoritma <i>Backtracking</i>	145
B.2	Algoritma <i>Hybrid Genetic</i>	150
C	File TEKS SOAL-SOAL PERMAINAN CALCUDOKU UNTUK PENGUJIAN	159

C.1	<i>File</i> Teks Soal-Soal Permainan Calcudoku dengan <i>Grid</i> Berukuran 4×4	159
C.2	<i>File</i> Teks Soal-Soal Permainan Calcudoku dengan <i>Grid</i> Berukuran 5×5	166
C.3	<i>File</i> Teks Soal-Soal Permainan Calcudoku dengan <i>Grid</i> Berukuran 6×6	172
C.4	<i>File</i> Teks Soal-Soal Permainan Calcudoku dengan <i>Grid</i> Berukuran 7×7	183
C.5	<i>File</i> Teks Soal-Soal Permainan Calcudoku dengan <i>Grid</i> Berukuran 8×8	188
C.6	<i>File</i> Teks Soal-Soal Permainan Calcudoku dengan <i>Grid</i> Berukuran 9×9	194
D	KODE PROGRAM	197
D.1	Grid.java	197
D.2	Cell.java	202
D.3	Cage.java	202
D.4	SolverBacktracking.java	204
D.5	SolverHybridGenetic.java	205
D.6	SolverRuleBased.java	206
D.7	SolverGenetic.java	221
D.8	Chromosome.java	224
D.9	Controller.java	225
D.10	Calcudoku.java	225
D.11	GUI.java	230
D.12	GeneticParameters.java	236

DAFTAR GAMBAR

1.1	Contoh permainan teka-teki Sudoku dengan solusinya	1
1.2	Contoh permainan teka-teki Calcudoku dengan penjelasan tentang elemen-elemen dari teka-teki ini [2]	2
2.1	Contoh permainan teka-teki Calcudoku dengan ukuran <i>grid</i> 4 x 4 yang belum diselesaikan. [1]	8
2.2	Solusi untuk permainan teka-teki Calcudoku yang diberikan pada Gambar 2.1 [1]	9
2.3	Ilustrasi <i>State space tree</i> yang digunakan dalam algoritma <i>backtracking</i> [1]	11
2.4	Contoh permainan teka-teki Calcudoku dengan ukuran <i>grid</i> 3 x 3 [1]	13
2.5	Ilustrasi <i>state</i> 3, 4, dan 5 pada sebuah <i>grid</i> teka-teki Calcudoku [1]	14
2.6	Ilustrasi <i>state</i> 19 pada sebuah <i>grid</i> teka-teki Calcudoku [1]	15
2.7	<i>State</i> 25, simpul tujuan, sebagai hasil yang dicapai [1]	16
2.8	<i>State space tree</i> yang dikembangkan dalam proses menyelesaikan teka-teki Calcudoku yang digambarkan pada Gambar 2.4 [1]	17
2.9	Contoh bagaimana cara mendeteksi aturan <i>naked pair</i> [2]	18
2.10	Contoh aturan <i>evil twin</i> [2]	18
2.11	Contoh aturan <i>hidden single</i> [2]	18
2.12	Contoh aturan <i>killer combination</i> untuk <i>cage</i> dengan ukuran 2 sel dengan operasi matematika penjumlahan [2]	19
2.13	Contoh aturan <i>X-wing</i> [2]	19
2.14	Contoh permainan teka-teki Calcudoku dengan ukuran <i>grid</i> 6 x 6 [2]	21
2.15	Contoh proses kawin silang antara dua kromosom [2]	22
2.16	Contoh proses mutasi [2]	22
2.17	Alur penyelesaian permainan teka-teki Calcudoku dengan menggunakan algoritma <i>hybrid genetic</i> [2]	23
3.1	Contoh permainan teka-teki Calcudoku dengan ukuran <i>grid</i> 4 x 4 yang belum diselesaikan, seperti yang digambarkan pada Gambar 2.1. [1]	25
3.2	<i>State</i> 4	26
3.3	<i>State</i> 11	26
3.4	<i>State</i> 12	26
3.5	<i>State</i> 17	27
3.6	<i>State</i> 18	27
3.7	<i>State</i> 19	28
3.8	<i>State</i> 23	28
3.9	<i>State</i> 93	28
3.10	<i>State space tree</i> yang dikembangkan dalam proses menyelesaikan teka-teki Calcudoku yang digambarkan pada Gambar 3.1	29
3.11	Contoh permainan teka-teki Calcudoku dengan ukuran <i>grid</i> 6 x 6 yang belum diselesaikan, seperti yang digambarkan pada Gambar 1.2. [2]	30
3.12	Permainan teka-teki Calcudoku setelah diselesaikan dengan algoritma <i>rule based</i>	31
3.13	Kromosom 1 dalam Generasi ke-1	32
3.14	Kromosom 2 dalam Generasi ke-1	32

3.15 Kromosom 3 dalam Generasi ke-1	32
3.16 Kromosom 4 dalam Generasi ke-1	32
3.17 Kromosom 5 dalam Generasi ke-1	33
3.18 Kromosom 6 dalam Generasi ke-1	33
3.19 Kromosom 7 dalam Generasi ke-1	33
3.20 Kromosom 8 dalam Generasi ke-1	33
3.21 Kromosom 9 dalam Generasi ke-1	34
3.22 Kromosom 10 dalam Generasi ke-1	34
3.23 Kromosom 11 dalam Generasi ke-1	34
3.24 Kromosom 12 dalam Generasi ke-1	34
3.25 Kromosom 1 dalam Generasi ke-2	36
3.26 Kromosom 2 dalam Generasi ke-2	36
3.27 Kromosom 3 dalam Generasi ke-2	36
3.28 Kromosom 4 dalam Generasi ke-2	36
3.29 Kromosom 5 dalam Generasi ke-2	37
3.30 Kromosom 6 dalam Generasi ke-2	37
3.31 Kromosom 7 dalam Generasi ke-2	37
3.32 Kromosom 8 dalam Generasi ke-2	37
3.33 Kromosom 9 dalam Generasi ke-2	38
3.34 Kromosom 10 dalam Generasi ke-2	38
3.35 Kromosom 11 dalam Generasi ke-2	38
3.36 Kromosom 12 dalam Generasi ke-2	38
3.37 Kromosom 1 dalam Generasi ke-3	40
3.38 Kromosom 2 dalam Generasi ke-3	40
3.39 Kromosom 3 dalam Generasi ke-3	40
3.40 Kromosom 4 dalam Generasi ke-3	40
3.41 Kromosom 5 dalam Generasi ke-3	41
3.42 Kromosom 6 dalam Generasi ke-3	41
3.43 Kromosom 7 dalam Generasi ke-3	41
3.44 Kromosom 8 dalam Generasi ke-3	41
3.45 Kromosom 9 dalam Generasi ke-3	42
3.46 Kromosom 10 dalam Generasi ke-3	42
3.47 Kromosom 11 dalam Generasi ke-3	42
3.48 Kromosom 12 dalam Generasi ke-3	42
3.49 Diagram <i>use case</i> untuk perangkat lunak permainan teka-teki Calcudoku	45
3.50 Diagram kelas untuk perangkat lunak permainan teka-teki Calcudoku	48
3.51 Diagram <i>sequence</i> saat perangkat lunak dibuka	49
3.52 Diagram <i>sequence</i> saat menu item "Load Puzzle File" dalam menu "File" dipilih	50
3.53 Diagram <i>sequence</i> saat file yang ingin dibuka dalam <i>file chooser</i> dipilih	51
3.54 Diagram <i>sequence</i> saat file permainan yang sudah dipilih dibuka	52
3.55 Diagram <i>sequence</i> saat menu item "Reset Puzzle" dalam menu "File" dipilih	54
3.56 Diagram <i>sequence</i> saat menu item "Close Puzzle File" dalam menu "File" dipilih	55
3.57 Diagram <i>sequence</i> saat menu item "Check Puzzle File" dalam menu "File" dipilih	56
3.58 Diagram <i>sequence</i> saat menu item "Backtracking" dalam menu "Solve" dipilih	57
3.59 Diagram <i>sequence</i> saat menu item "Hybrid Genetic" dalam menu "Solve" dipilih	58
3.60 Diagram <i>sequence</i> saat menu item "Set Genetic Algorithm Parameters" dalam menu "Solve" dipilih	59
3.61 Diagram <i>sequence</i> saat button "OK" dalam form "Set Genetic Algorithm Parameters" dalam menu "Solve" dipilih	59
3.62 Diagram <i>sequence</i> saat button "Cancel" dalam form "Set Genetic Algorithm Parameters" dalam menu "Solve" dipilih	59

3.63 Diagram <i>sequence</i> saat perangkat lunak ditutup	60
4.1 Contoh <i>file</i> masukan	61
4.2 Contoh keluaran	62
4.3 Perancangan GUI sebelum <i>file</i> permainan dibuka	62
4.4 Perancangan GUI sesudah <i>file</i> permainan dibuka	63
4.5 Perancangan GUI sesudah permainan berdasarkan <i>file</i> permainan yang dibuka diselesaikan	63
4.6 Menu <i>File</i>	64
4.7 Menu <i>Solve</i>	64
4.8 Diagram kelas untuk perangkat lunak Calcudoku	66
4.9 Diagram kelas Grid	71
4.10 Diagram kelas Cage	73
4.11 Diagram kelas Cell	74
4.12 Diagram kelas SolverBacktracking	75
4.13 Diagram kelas SolverHybridGenetic	76
4.14 Diagram kelas SolverRuleBased	81
4.15 Diagram kelas SolverGenetic	84
4.16 Diagram kelas Chromosome	85
4.17 Diagram kelas ChromosomeComparator	85
4.18 Diagram kelas Controller	86
4.19 Diagram kelas Calcudoku	90
4.20 Diagram kelas WindowListener	91
4.21 Diagram kelas PuzzleFileFilter	91
4.22 Diagram kelas GUI	94
4.23 Diagram kelas CellKeyListener	95
4.24 Diagram kelas PopupMenuItemListener	95
4.25 Diagram kelas CellTextFieldListener	96
4.26 Diagram kelas GeneticParameters	98
5.1 Antarmuka perangkat lunak saat pertama kali dibuka	100
5.2 Kotak dialog untuk memilih <i>file</i> permainan yang akan dibuka	101
5.3 Antarmuka perangkat lunak sesudah membuka <i>file</i> permainan yang dipilih	101
5.4 Kotak dialog untuk mengatur nilai dari parameter-parameter algoritma genetik	102
5.5 Antarmuka perangkat lunak setelah permainan berdasarkan <i>file</i> permainan yang telah dibuka diselesaikan	102
5.6 Kotak pesan error " <i>Puzzle file not loaded</i> "	103
5.7 Kotak pemilihan <i>file</i> permainan	104
5.8 Kotak dialog "Are you sure you want to load another puzzle file?"	104
5.9 Pesan error " <i>Invalid puzzle file</i> "	104
5.10 Pesan error " <i>Invalid cages</i> "	105
5.11 Pesan error " <i>Error in loading puzzle file</i> "	105
5.12 Pesan informasi " <i>Congratulations, you have successfully solved the puzzle</i> "	106
5.13 Kotak dialog "Are you sure you want to reset this puzzle?"	106
5.14 Pesan informasi " <i>Row</i> (nomor baris) has duplicate numbers"	107
5.15 Pesan informasi " <i>Column</i> (nomor kolom) has duplicate numbers"	107
5.16 Pesan informasi " <i>Values of cells in the cage do not reach the target number</i> "	107
5.17 Pesan informasi " <i>There are cells with incorrect values in the grid</i> "	108
5.18 Pesan informasi " <i>There are empty cells in the grid</i> "	108
5.19 Pesan informasi " <i>The backtracking algorithm has successfully solved the puzzle</i> "	108
5.20 Pesan informasi " <i>The backtracking algorithm has failed to solve the puzzle</i> "	109
5.21 Pesan informasi " <i>Genetic algorithm parameters have not been set</i> "	110

5.22 Pesan informasi "The hybrid genetic algorithm has successfully solved the puzzle" . . .	110
5.23 Pesan informasi "The hybrid genetic algorithm has failed to solve the puzzle"	111
5.24 Form untuk mengatur nilai untuk parameter-parameter algoritma genetik	111
5.25 Pesan error "Invalid number format"	112
5.26 Kotak dialog "Are you sure you want to close this puzzle file?"	112
5.27 Kotak dialog "Are you sure you want to exit the application?"	113
5.28 File masukan untuk pengujian keakuratan	113
5.29 GUI permainan berdasarkan file masukan yang dapat dilihat pada Gambar 5.28 . .	114
5.30 Solusi untuk permainan berdasarkan file masukan yang dapat dilihat pada Gambar 5.28	116
A.1 Contoh permainan teka-teki Calcudoku dengan ukuran grid 4 x 4 yang belum diselesaikan, seperti yang digambarkan pada Gambar 2.1. [1]	131
A.2 State 4	132
A.3 State 11	132
A.4 State 12	132
A.5 State 17	133
A.6 State 18	133
A.7 State 19	133
A.8 State 23	134
A.9 State 24	134
A.10 State 31	135
A.11 State 32	135
A.12 State 34	135
A.13 State 37	136
A.14 State 47	137
A.15 State 48	137
A.16 State 52	137
A.17 State 53	138
A.18 State 68	139
A.19 State 69	139
A.20 State 71	139
A.21 State 72	140
A.22 State 74	140
A.23 State 75	140
A.24 State 76	141
A.25 State 77	141
A.26 State 78	141
A.27 State 81	142
A.28 State 83	142
A.29 State 85	143
A.30 State 88	143
A.31 State 92	143
A.32 State 93	144

DAFTAR TABEL

3.1	Tabel parameter untuk algoritma genetik yang akan digunakan untuk menyelesaikan teka-teki Calcudoku yang digambarkan pada Gambar 3.12	31
3.2	Tabel nilai kelayakan untuk kromosom-kromosom pada Generasi ke-1	35
3.3	Tabel nilai kelayakan untuk kromosom-kromosom pada Generasi ke-1	39
3.4	Tabel nilai kelayakan untuk kromosom-kromosom pada Generasi ke-3	43
3.5	Skenario me- <i>load file</i>	45
3.6	Skenario memilih salah satu dari dua <i>solver</i> yang disediakan	46
3.7	Skenario me- <i>reset</i> permainan	46
3.8	Skenario meminta perangkat lunak untuk memeriksa permainan	46
3.9	Skenario menutup <i>file</i> masukan	47
3.10	Skenario menyelesaikan permainan dengan usahanya sendiri	47
3.11	Skenario mengatur nilai dari parameter-parameter untuk algoritma genetik	47
5.1	Lingkungan perangkat keras untuk pengujian perangkat lunak	99
5.2	Lingkungan perangkat lunak untuk pengujian perangkat lunak	100
5.3	Hasil pengujian algoritma <i>backtracking</i> untuk Calcudoku	116
5.4	Nilai untuk parameter-parameter algoritma genetik untuk setiap percobaan yang dilakukan	117
5.5	Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku (Skenario 1)	117
5.6	Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku (Skenario 2)	118
5.7	Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku (Skenario 3)	118
5.8	Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku (Skenario 4)	118
5.9	Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku (Skenario 5)	119
5.10	Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku (Skenario 6)	119
5.11	Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku (Skenario 7)	119
5.12	Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku (Skenario 8)	120
5.13	Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku (Skenario 9)	120
5.14	Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku (Skenario 10)	120
5.15	Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku (Skenario 11)	121
5.16	Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku (Skenario 12)	121
5.17	Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku (Skenario 13)	121
5.18	Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku (Skenario 14)	122
5.19	Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku (Skenario 15)	122
5.20	Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku (Skenario 16)	122
B.1	Hasil pengujian algoritma <i>backtracking</i> untuk Calcudoku dengan ukuran <i>grid</i> 4×4	146
B.2	Hasil pengujian algoritma <i>backtracking</i> untuk Calcudoku dengan ukuran <i>grid</i> 5×5	147
B.3	Hasil pengujian algoritma <i>backtracking</i> untuk Calcudoku dengan ukuran <i>grid</i> 6×6	148
B.4	Hasil pengujian algoritma <i>backtracking</i> untuk Calcudoku dengan ukuran <i>grid</i> 7×7	149
B.5	Hasil pengujian algoritma <i>backtracking</i> untuk Calcudoku dengan ukuran <i>grid</i> 8×8	149
B.6	Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku dengan ukuran <i>grid</i> 4×4 (Skenario 1-4)	151

B.7 Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku dengan ukuran <i>grid</i> 4×4 (Skenario 5-8)	152
B.8 Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku dengan ukuran <i>grid</i> 4×4 (Skenario 9-12)	153
B.9 Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku dengan ukuran <i>grid</i> 4×4 (Skenario 13-16)	154
B.10 Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku dengan ukuran <i>grid</i> 5×5 (Skenario 1-4)	155
B.11 Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku dengan ukuran <i>grid</i> 5×5 (Skenario 5-8)	156
B.12 Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku dengan ukuran <i>grid</i> 5×5 (Skenario 9-12)	157
B.13 Hasil pengujian algoritma <i>hybrid genetic</i> untuk Calcudoku dengan ukuran <i>grid</i> 5×5 (Skenario 13-16)	158

¹

BAB 1

²

PENDAHULUAN

- ³ Bab ini membahas tentang latar belakang, rumusan masalah, tujuan, batasan masalah,
⁴ metodologi penelitian, dan sistematika pembahasan dari skripsi ini.

⁵ **1.1 Latar Belakang**

⁶ Calcudoku, atau dikenal juga sebagai KenKen, atau Mathdoku, adalah sebuah permainan
⁷ teka-teki (*puzzle*) angka yang untuk menyelesaiakannya memerlukan perpaduan dari logika
⁸ dan kemampuan aritmatika yang sederhana. Permainan ini adalah sebuah permainan teka-
⁹ teki logika yang sederhana, namun, untuk menemukan solusinya cukup rumit, terutama
¹⁰ untuk masalah yang lebih susah.

¹¹ Teka-teki ini mirip dengan Sudoku. Sudoku adalah sebuah permainan teka-teki angka
¹² dengan *grid* berukuran n^2 , di mana dalam setiap baris, kolom, dan n^2 area yang berukuran
¹³ $n \times n$ tidak boleh ada angka yang berulang, dengan n adalah ukuran area. Biasanya, $n = 3$,
¹⁴ sehingga *grid* berukuran 9×9 , dan ada 9 area yang berukuran 3×3 . Contoh permainan
¹⁵ teka-teki Sudoku dapat dilihat pada Gambar 1.1.

¹⁶ Persamaannya, tujuan dari teka-teki ini adalah mengisi setiap sel (*cell*) dalam (*grid*)
¹⁷ dengan angka 1 sampai n tanpa pengulangan angka dalam setiap kolomnya dan barisnya
¹⁸ untuk *grid* berukuran $n \times n$, dengan n adalah ukuran *grid*. Tidak ada angka yang boleh
¹⁹ muncul lebih dari sekali dalam setiap baris atau kolom dalam *grid*.

²⁰ Perbedaannya, jika pada Sudoku *grid* berukuran $n \times n$ dibagi menjadi n (*cage*) dengan
²¹ setiap *cage* terdiri atas n sel, pada Calcudoku *grid* dibagi menjadi sejumlah *cage* yang jumlah
²² selnya bervariasi. Setiap *cage* dibatasi oleh garis yang lebih tebal daripada garis pembatas
²³ antar sel. Angka-angka dalam satu *cage* yang sama harus menghasilkan angka tujuan yang
²⁴ telah ditentukan jika dihitung menggunakan operasi matematika yang telah ditentukan

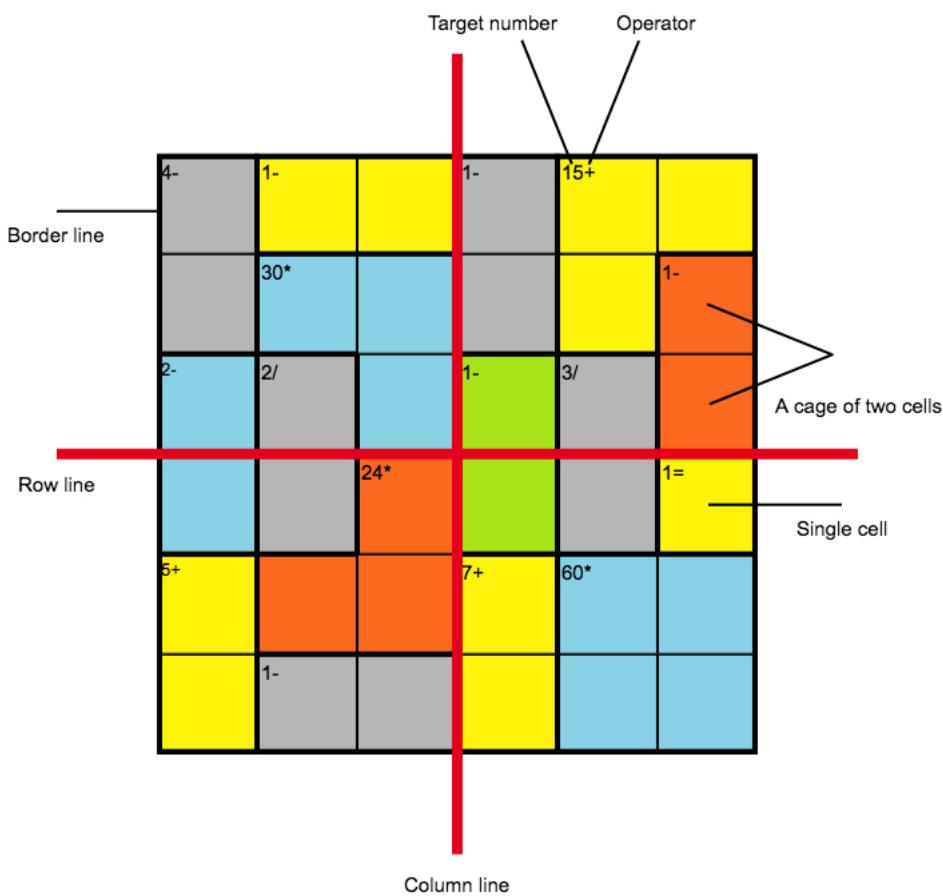
	2		5	1		9		
8			2	3			6	
	3		6		7			
		1			6			
5	4					1	9	
		2			7			
	9		3		8			
2			8	4			7	
1		9	7			6		

Unsolved Sudoku

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

Solved Sudoku

Gambar 1.1: Contoh permainan teka-teki Sudoku dengan solusinya



Gambar 1.2: Contoh permainan teka-teki Calcudoku dengan penjelasan elemen-elemen dari teka-teki ini [2]

¹ (penjumlahan, pengurangan, perkalian, atau pembagian). Angka-angka dalam satu *cage* ² juga boleh berulang, selama pengulangan tidak terjadi dalam satu kolom atau baris yang ³ sama. Jika *cage* hanya berisi satu sel, maka satu-satunya kemungkinan jawaban untuk sel ⁴ tersebut adalah angka tujuan dari *cage* tersebut. Angka tujuan dan operasi matematika ⁵ dituliskan di sudut kiri atas *cage*. Pada awalnya, setiap sel dalam setiap *cage* dalam teka- ⁶ teki ini kosong, belum terisi oleh angka-angka. *Border line* adalah garis pembatas terluar, ⁷ *row line* adalah garis pembatas antar baris, dan *column line* adalah garis pembatas antar ⁸ kolom. Gambar 1.2 menggambarkan contoh sebuah permainan teka-teki Calcudoku [1] ⁹ [2].

¹⁰ Calcudoku dapat diselesaikan menggunakan beberapa algoritma. Skripsi ini membahas ¹¹ tentang penyelesaian Calcudoku menggunakan algoritma *backtracking* dan algoritma *hybrid* ¹² *genetic*, dan perbandingan performansi *performance* antara kedua algoritma tersebut dalam ¹³ hal kecepatan dan kesuksesan dalam menyelesaikan Calcudoku.

¹⁴ Algoritma *backtracking* adalah sebuah algoritma umum yang mencari solusi dengan men- ¹⁵ coba salah satu dari beberapa pilihan, jika pilihan yang dipilih ternyata salah, komputasi ¹⁶ dimulai lagi pada titik pilihan dan mencoba pilihan lainnya. Untuk bisa melacak kembali ¹⁷ langkah-langkah yang telah dipilih, maka algoritma harus secara eksplisit menyimpan je- ¹⁸ jak dari setiap langkah yang sudah pernah dipilih, atau menggunakan rekursi (*recursion*). ¹⁹ Rekursi dipilih karena jauh lebih mudah daripada harus menyimpan jejak setiap langkah ²⁰ yang pernah dipilih, hal ini menyebabkan algoritma ini biasanya berbasis DFS (*Depth First*

1 *Search)* [1].

2 Algoritma *rule based* adalah sebuah algoritma berbasis aturan logika untuk menyelesa-
3 ikan permainan teka-teki Sudoku dan variasinya, termasuk Calcudoku. Beberapa aturan
4 logika yang digunakan dalam algoritma ini adalah *single square rule*, *naked subset rule*,
5 *hidden single rule*, *evil twin rule*, *killer combination*, dan *X-wing*.

6 Pencarian heuristik adalah sebuah teknik pencarian kecerdasan buatan (*artificial intelli-
7 gence*) yang menggunakan heuristik dalam langkah-langkahnya. Heuristik adalah semacam
8 aturan tidak tertulis yang mungkin menghasilkan solusi. Heuristik kadang-kadang efektif,
9 tetapi tidak dijamin akan berhasil. dalam setiap kasus. Heuristik memerlukan peran pen-
10 ting dalam strategi pencarian karena sifat eksponensial dari kebanyakan masalah. Heuristik
11 membantu mengurangi jumlah alternatif solusi dari angka yang bersifat eksponensial men-
12 jadi angka yang bersifat polinomial. Contoh teknik pencarian heuristik adalah *Generate
13 and Test*, *Hill Climbing*, dan *Best First Search*.

14 Algoritma genetik adalah salah satu teknik heuristik *Generate and Test* yang terinspirasi
15 oleh sistem seleksi alam. Algoritma ini adalah perpaduan dari bidang biologi dan ilmu
16 komputer. Algoritma ini adalah salah satu dari teknik pencarian heuristik.

17 Algoritma ini memanipulasi informasi, biasanya disebut sebagai kromosom. Kromosom
18 ini mengkodekan kemungkinan jawaban untuk sebuah masalah yang diberikan. Kromosom
19 dievaluasi dan diberi *fitness value* berdasarkan seberapa baikkah kromosom dalam menyele-
20 saikan masalah yang diberikan berdasarkan kriteria yang ditentukan oleh pembuat program.
21 Nilai kelayakan ini digunakan sebagai probabilitas kebertahanan hidup kromosom dalam
22 satu siklus reproduksi. Kromosom baru (kromosom anak, *child chromosome*) diproduksi
23 dengan menggabungkan dua (atau lebih) kromosom orang tua (*parent chromosome*). Pro-
24 ses ini dirancang untuk menghasilkan kromosom-kromosom keturunan yang lebih layak,
25 kromosom-kromosom ini menyandikan jawaban yang lebih baik, sampai solusi yang baik
26 dan yang bisa diterima ditemukan.

27 Algoritma *hybrid genetic* adalah gabungan antara algoritma genetik dan algoritma-
28 algoritma lainnya. Dalam kasus ini, algoritma genetik digabungkan dengan algoritma *rule
29 based*. Algoritma *rule based* akan dijalankan sampai pada titik dimana algoritma tidak bisa
30 menyelesaikan permainan teka-teki Calcudoku. Jika algoritma sudah tidak bisa menyele-
31 saikan permainan, maka algoritma genetik akan mulai dijalankan [2].

32 **1.2 Rumusan Masalah**

33 Berdasarkan latar belakang yang telah diuraikan di atas, dapat dirumuskan permasalahan
34 sebagai berikut:

- 35 1. Bagaimana cara mengimplementasikan perangkat lunak (*software*) permainan teka-
36 teki Calcudoku?
- 37 2. Bagaimana cara mengimplementasikan algoritma *backtracking* untuk menyelesaikan
38 Calcudoku?
- 39 3. Bagaimana cara mengimplementasikan algoritma *hybrid genetic* untuk menyelesaikan
40 Calcudoku?

- 1 4. Bagaimana perbandingan performansi algoritma *backtracking* dengan algoritma *hybrid genetic* dalam menyelesaikan Calcudoku?
- 2

3 1.3 Tujuan

4 Berdasarkan rumusan masalah yang telah dirumuskan, maka tujuan dari pembuatan skripsi
5 ini adalah:

- 6 1. Membuat perangkat lunak permainan teka-teki Calcudoku yang menerima input ber-
7 upa soal teka-teki dan mampu menyelesaikan soal teka-teki tersebut menggunakan
8 algoritma *backtracking* dan *hybrid genetic*.
- 9 2. Membandingkan performansi algoritma *backtracking* dengan algoritma *hybrid genetic*
10 dalam hal kecepatan dan kesuksesan dalam menyelesaikan Calcudoku.

¹ 1.4 Batasan Masalah

² Ruang lingkup dari skripsi ini dibatasi oleh batasan-batasan masalah sebagai berikut:

- ³ 1. Ukuran *grid* untuk permainan teka-teki Calcudoku adalah antara 4×4 sampai dengan
⁴ 8×8 . Pada awalnya, ukuran *grid* direncanakan akan dibatasi dari 3×3 sampai dengan
⁵ 9×9 , tetapi karena kurangnya contoh soal teka-teki Calcudoku dengan ukuran 3×3 ,
⁶ dan ada masalah saat pengujian (keluar pesan error "*Memory full*" saat menguji *solver*
⁷ dengan algoritma *backtracking* pada *grid* yang berukuran 9×9 , maka ukuran *grid*
⁸ dibatasi dari 4×4 sampai dengan 8×8 .
- ⁹ 2. Pada algoritma *rule based*, yang merupakan bagian dari algoritma *hybrid genetic*,
¹⁰ aturan-aturan logika yang digunakan dibatasi hanya pada aturan *single square*, *naked*
¹¹ *single*, *naked double*, *hidden single*, dan *killer combination*.

¹² 1.5 Metodologi Penelitian

¹³ Langkah-langkah yang akan dilakukan dalam pembuatan skripsi ini adalah:

- ¹⁴ 1. Studi literatur
 - ¹⁵ (a) Melakukan studi literatur tentang permainan teka-teki Calcudoku.
 - ¹⁶ (b) Melakukan studi literatur tentang algoritma *backtracking*.
 - ¹⁷ (c) Melakukan studi literatur tentang algoritma *rule based* dan algoritma genetik.
- ¹⁸ 2. Analisis, perancangan, dan pengembangan perangkat lunak
 - ¹⁹ (a) Melakukan analisis dan menentukan fitur-fitur yang diperlukan dalam perangkat
²⁰ lunak permainan teka-teki Calcudoku.
 - ²¹ (b) Membuat perangkat lunak Calcudoku dengan fitur-fitur yang telah ditentukan.
 - ²² (c) Mengimplementasikan algoritma *backtracking* untuk Calcudoku.
 - ²³ (d) Mengimplementasikan algoritma *hybrid genetic* untuk Calcudoku.
 - ²⁴ (e) Membandingkan performansi algoritma *backtracking* dengan algoritma *hybrid*
²⁵ *genetic* dalam menyelesaikan Calcudoku.
- ²⁶ 3. Melakukan pengujian dan eksperimen terhadap perangkat lunak Calcudoku yang te-
²⁷ lah dibuat.
- ²⁸ 4. Membuat kesimpulan berdasarkan hasil pengujian perangkat lunak yang telah dibuat.

²⁹ 1.6 Sistematika Pembahasan

³⁰ Sistematika pembahasan skripsi ini adalah sebagai berikut:

- ³¹ 1. Bab 1 berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi
³² penelitian, dan sistematika pembahasan dari skripsi ini.

- 1 2. Bab 2 membahas tentang landasan teori yang digunakan dalam skripsi ini, yaitu
2 tentang permainan teka-teki Calcudoku, algoritma *backtracking* dan algoritma *hybrid*
3 *genetic*.
- 4 3. Bab 3 membahas tentang analisis perangkat lunak Calcudoku dan analisis algoritma
5 *backtracking* dan algoritma *hybrid genetic*.
- 6 4. Bab 4 membahas tentang perancangan dan pembuatan perangkat lunak Calcudoku
7 dan algoritma *backtracking* dan algoritma *hybrid genetic* untuk menyelesaikan per-
8 mainan, perancangan antarmuka (*interface*), input dan output, diagram kelas (*class*
9 *diagram*), dan diagram aktivitas (*activity diagram*).
- 10 5. Bab 5 membahas tentang implementasi dari perangkat lunak Calcudoku dan algo-
11 ritma *backtracking* dan algoritma *hybrid genetic* yang telah dirancang, implementasi
12 antarmuka, input dan output yang telah dirancang, dan pengujian perangkat lunak
13 Calcudoku dalam hal perbandingan performansi algoritma *backtracking* dan algoritma
14 *hybrid genetic* dalam menyelesaikan permainan.
- 15 6. Bab 6 berisi simpulan dari pembuatan perangkat lunak Calcudoku dan hasil pengu-
16 jiannya, dan saran untuk penelitian pengembangan perangkat lunak selanjutnya.

¹ **BAB 2**

² **LANDASAN TEORI**

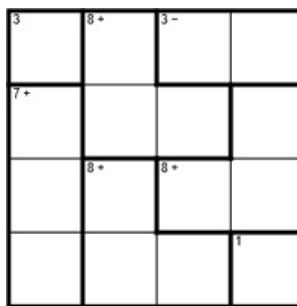
³ Bab ini membahas tentang landasan teori yang akan digunakan dalam skripsi ini yang
⁴ diambil dari dua sumber, yaitu "KenKen Puzzle Solver using Backtracking Algorithm"
⁵ karya Asanilta Fahda [1] dan "Solving and Modeling Ken-ken Puzzle by Using Hybrid
⁶ Genetics Algorithm" karya Olivia Johanna, Samuel Lukas, dan Kie Van Ivanký Saputra
⁷ [2].

⁸ **2.1 Calcudoku [1] [2]**

⁹ Sebagai salah satu jenis permainan teka-teki aritmatika dan *grid*, Calcudoku, atau dikenal
¹⁰ juga sebagai KenKen, KenDoku, atau Mathdoku, diciptakan pada tahun 2004 oleh seorang
¹¹ guru matematika dari Jepang yang bernama Tetsuya Miyamoto untuk memenuhi tujuan-
¹² nya untuk melatih kemampuan matematika dan logika siswa-siswinya dengan cara yang
¹³ menyenangkan. Nama KenKen diambil dari kata bahasa Jepang yang berarti kepanda-
¹⁴ ian. Permainan yang mengasah otak ini dengan cepat menyebar ke seluruh Jepang dan
¹⁵ Amerika Serikat, menggantikan permainan teka-teki silang di banyak koran. Permainan
¹⁶ ini kemudian menjadi sensasi di seluruh dunia setelah munculnya versi *online* dan *mobile*
¹⁷ dari permainan teka-teki ini, khususnya menarik untuk pecinta permainan teka-teki angka
¹⁸ seperti Sudoku.

¹⁹ Seperti dalam Sudoku, dalam teka-teki ini, pemain diberikan sebuah *grid* dengan ukur-
²⁰ an $n \times n$, dengan n biasanya $3 \leq n \leq 9$. *Grid* ini harus diisi dengan angka 1 sampai dengan
²¹ n sehingga dalam setiap baris setiap angka hanya muncul sekali, dalam setiap kolom setiap
²² angka hanya muncul sekali. Perbedaannya dengan Sudoku adalah, Calcudoku dibagi ke da-
²³ lam *cage* (sekelompok sel yang dibatasi oleh garis yang lebih tebal daripada garis pembatas
²⁴ antar sel, setiap *cage* mempunyai angka tujuan dan operator yang telah ditentukan), dan
²⁵ angka-angka dalam setiap *cage* harus mencapai angka tujuan jika dihitung menggunakan
²⁶ operator yang telah ditentukan. Angka tujuan dan operasi yang telah ditentukan ditulis di
²⁷ sudut kiri atas *cage*. Ada lima kemungkinan operator:

- ²⁸ 1. $+$, sebuah operator n -ary yang menandakan penjumlahan.
²⁹ 2. $-$, sebuah operator biner yang menandakan pengurangan.
³⁰ 3. \times , sebuah operator n -ary yang menandakan perkalian.
³¹ 4. \div sebuah operator biner yang menandakan pembagian.



Gambar 2.1: Contoh permainan teka-teki dengan ukuran *grid* 4×4 yang belum diselesaikan. [1]

1 5. $=$, (simbol ini biasanya dihilangkan), sebuah operator uner yang menandakan persamaan.

3 Jika operasi matematika yang ditentukan adalah pengurangan atau pembagian, maka ukuran *cage* harus berukuran dua sel. Pada beberapa versi dari teka-teki ini, hanya angka tujuan yang diberikan, dan pemain harus menebak operator dari setiap *cage* untuk menyelesaikan teka-tekinya [1] [2].

7 Untuk menyelesaikan sebuah teka-teki Calcudoku, pemain pertama-tama harus memiliki dua permasalahan utama dari teka-teki ini, yaitu:

- 9 1. Angka-angka mana yang harus dimasukkan ke dalam sebuah *cage*
10 2. Dalam urutan apa angka-angka tersebut harus dimasukkan ke dalam sebuah *cage*

11 Seperti kebanyakan permainan teka-teki angka, cara yang paling mudah untuk menyelesaikan teka-teki ini adalah dengan mengeliminasi angka-angka yang sudah digunakan dan mencoba satu per satu angka yang mungkin (*trial and error*).

14 Dalam pengisian teka-teki ini ada dua tahapan, yaitu:

15 1. Mencari *cage* yang hanya berukuran 1 sel, karena *cage* ini tidak menghasilkan pertanyaan angka apa dan urutan apa. Tahap ini adalah tahap yang paling jelas. Contoh, pada Gambar 2.1, *cage* pada sudut kiri atas dan *cage* pada sudut kanan bawah hanya berukuran 1 sel, dan dapat langsung diisi dengan angka tujuannya.

19 2. Mencari *cage* yang hanya mempunyai satu kemungkinan kombinasi angka, sehingga masalah angka-angka apa yang harus diisi dalam *cage* tersebut terjawab. Contoh, *cage* pada sudut kanan atas mempunyai aturan "3-", artinya angka tujuannya adalah 3 dengan menggunakan operasi pengurangan. Satu-satunya pasangan angka dari himpunan $\{1,2,3,4\}$ yang akan menghasilkan angka 3 saat satu angka dikurangkan dari angka yang lainnya adalah $\{1,4\}$. Namun masalahnya adalah urutan angka-angka yang harus dimasukkan. Dalam kasus ini, untungnya, sel pada sudut kanan bawah sudah diisi dengan angka 1, maka angka 1 tidak bisa digunakan lagi pada kolom yang paling kanan. Jadi, dengan menggunakan cara eliminasi, sel pada sudut kanan atas harus diisi dengan angka 4 dan sel di sebelah kirinya, yaitu sel pada baris yang paling atas dan kolom ketiga dari kiri, harus diisi dengan angka 1. Hal ini memberikan solusi untuk sel pada baris yang paling atas dan kolom kedua dari kiri, yaitu angka 2, karena angka 2 adalah angka yang belum pernah dipakai dalam baris tersebut.

3	8+	3-	
1	4	2	3
4	1	3	2
2	3	4	1

Gambar 2.2: Solusi untuk permainan teka-teki Calcudoku yang diberikan pada Gambar 2.1.

1 Proses ini berlanjut sampai semua sel dalam *grid* terisi dan menghasilkan solusi pada
 2 Gambar 2.2 [1].

3 Seiring dengan meningkatnya tingkat kesulitan, langkah berikutnya tidak akan langsung
 4 muncul dengan jelas. Kadang-kadang, pemain mencapai titik dimana langkah berikutnya
 5 tidak pasti. Pemain harus menebak langkah-langkah berikutnya dan melihat apakah lang-
 6 kah ini akan menghasilkan solusinya. Jika tidak, pemain harus mundur kembali ke titik
 7 ketidakpastian tersebut.

8 Sebuah teka-teki Calcudoku dengan ukuran $n \times n$, dengan n melambangkan jumlah
 9 sel dalam satu baris atau kolom, mempunyai n^2 sel dalam sebuah *grid*. Sel yang terletak
 10 dalam baris b dan kolom k diberi label $C_{b,k} = bn + k$ dan nilai dari sel tersebut adalah
 11 $V(C_{b,k}) \in \{1, 2, \dots, n\}$. Nomor baris b memiliki *range* $0 \leq b \leq n - 1$. Nomor kolom k
 12 memiliki *range* $0 \leq k \leq n - 1$. Nomor sel C memiliki *range* $0 \leq C \leq n^2 - 1$. Nomor
 13 sel adalah hasil perkalian dari nomor baris tempat sebuah sel berada dikalikan dengan
 14 banyaknya sel dalam sebuah baris, lalu dijumlahkan dengan nomor kolom tempat sebuah
 15 sel berada. Sebuah *cage*, yang diberi label A_i adalah sebuah himpunan dari sel, yaitu
 16 $A_i = \{C_{b,k}\}$. Setiap *cage* terhubung dengan satu operator aritmatika $O_i \in \{+, -, \times, \div, =\}$,
 17 artinya operator aritmatika adalah salah satu dari penjumlahan, pengurangan, perkalian,
 18 pembagian, dan sama dengan, dan satu angka tujuan $H_i \in N$, artinya angka tujuan adalah
 19 sebuah bilangan asli. Tiga aturan dalam mendefinisikan masalah dalam Calcudoku adalah
 20 sebagai berikut [2]:

- 21 1. $|A_i| = 1 \rightarrow O_i = \phi$, artinya setiap *cage* yang jumlah selnya 1 dengan operasi mate-
 22 matika yang terkait dengan *cage* tersebut memiliki hubungan korespondensi satu ke
 23 satu.
- 24 2. $O_i \in \{-, \div\} \rightarrow |A_i| = 2$, artinya jika operasi yang digunakan dalam sebuah *cage*
 25 adalah pengurangan atau pembagian, maka jumlah sel dalam *cage* tersebut harus 2.
- 26 3. $\forall C_{b,k} \rightarrow C_{b,k} \in \exists! A_i$, artinya setiap sel hanya boleh menjadi anggota dari satu dan
 27 hanya satu *cage*.

28 Tujuan dari teka-teki ini adalah untuk mencari nilai $V(C_{b,k})$ dan memenuhi persyaratan
 29 berikut [2]:

- 30 1. $|A_i| = 1 \wedge C_{b,k} \in A_i \rightarrow V(C_{b,k}) = H_i$, artinya jika sel adalah bagian dari sebuah
 31 *cage* yang jumlah selnya 1, maka nilai dari sel tersebut adalah angka tujuan dari *cage*
 32 tersebut.

- ¹ 2. $O_i \in \{-\} \wedge A_i = \{C_{a,b}, C_{p,q}\} \rightarrow |V(C_{a,b}) - V(C_{p,q})| = H_i$, artinya jika sebuah *cage* yang operasi matematikanya adalah pengurangan, maka nilai absolut dari hasil pengurangan nilai kedua sel di dalam *cage* tersebut adalah angka tujuan dari *cage* tersebut.
- ⁵ 3. $O_i \in \{\div\} \wedge A_i = \{C_{a,b}, C_{p,q}\} \rightarrow V(C_{a,b})/V(C_{p,q}) = H_i$, artinya jika sebuah *cage* yang operasi matematikanya adalah pembagian, maka nilai dari hasil pembagian nilai kedua sel di dalam *cage* tersebut adalah angka tujuan dari *cage* tersebut.
- ⁸ 4. $O_i \in \{+\} \rightarrow \sum_{C_{b,k} \in A_i} V(C_{b,k}) = H_i$, artinya jika sebuah *cage* yang operasi matematikanya adalah penjumlahan, maka nilai dari hasil penjumlahan dari nilai semua sel di dalam *cage* tersebut adalah angka tujuan dari *cage* tersebut.
- ¹¹ 5. $O_i \in \{\times\} \rightarrow \prod_{C_{b,k} \in A_i} V(C_{b,k}) = H_i$, artinya jika sebuah *cage* yang operasi matematikanya adalah perkalian, maka nilai dari hasil perkalian dari nilai semua sel di dalam *cage* tersebut adalah angka tujuan dari *cage* tersebut.

¹⁴ 2.2 Algoritma *Backtracking* [1]

¹⁵ Algoritma *backtracking* adalah sebuah algoritma umum yang mencari solusi dengan men-
¹⁶ coba salah satu dari beberapa pilihan, jika pilihan yang dipilih ternyata salah, komputasi
¹⁷ dimulai lagi pada titik pilihan dan mencoba pilihan lainnya. Untuk bisa melacak kembali
¹⁸ langkah-langkah yang telah dipilih, maka algoritma harus secara eksplisit menyimpan je-
¹⁹ jak dari setiap langkah yang sudah pernah dipilih, atau menggunakan rekursi (*recursion*).
²⁰ Rekursi dipilih karena jauh lebih mudah daripada harus menyimpan jejak setiap langkah
²¹ yang pernah dipilih. Hal ini menyebabkan algoritma ini biasanya berbasis DFS (*Depth First*
²² *Search*).

²³ Algoritma *backtracking* pertama kali diperkenalkan pada tahun 1950 oleh D.H. Lehmer
²⁴ sebagai perbaikan algoritma *brute force*. Algoritma ini lalu dikembangkan lebih lanjut
²⁵ oleh R.J. Walker, S.W. Golomb, dan L.D. Baumert. Algoritma ini terbukti efektif untuk
²⁶ menyelesaikan banyak permainan logika (misalnya *tic tac toe*, *maze*, catur, dan lain-lain)
²⁷ karena algoritma itu terutama berguna untuk menyelesaikan masalah-masalah *constraint*
²⁸ *satisfaction*, di mana sekumpulan objek harus memenuhi sejumlah batasan.

²⁹ Implementasi algoritma *backtracking* memiliki tiga sifat umum, yaitu [1]:

³⁰ 1. *Solution space*

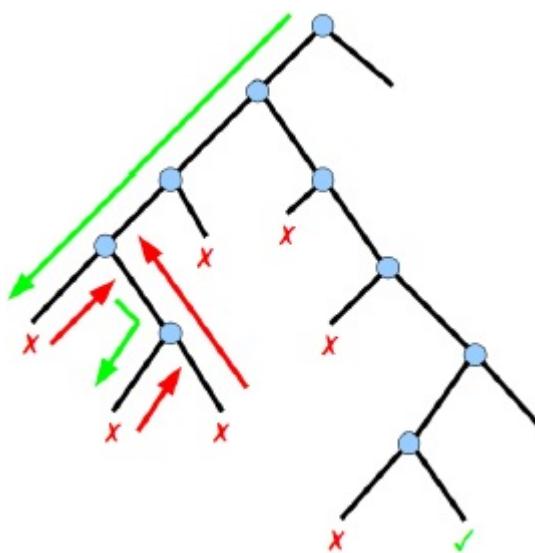
³¹ Solusi untuk masalah ini dinyatakan sebagai sebuah vektor X dengan n -tuple:

$$X = (x_1, x_2, \dots, x_n), x_i \in S_i$$

³² di mana adalah mungkin bahwa:

$$S_1 = S_2 = \dots = S_n$$

³³ 2. Fungsi pembangkit X_k (*generating function*)



Gambar 2.3: Ilustrasi *State space tree* yang digunakan dalam algoritma *backtracking* [1]

1 Fungsi pembangkit X_k dinyatakan sebagai:

$$T(k)$$

2 di mana $T(k)$ membangkitkan nilai X_k , dari 1 sampai n , yang merupakan komponen
3 dari vektor solusi.

4 3. Fungsi pembatas (*bounding function*)

5 Fungsi pembatas dinyatakan sebagai:

$$B(x_1, x_2, \dots, x_k)$$

6 di mana B bernilai *true* jika (x_1, x_2, \dots, x_k) mengarah ke solusi. Jika B bernilai
7 *true*, maka nilai $x_k + 1$ akan terus dibangkitkan, dan jika B bernilai *false*, maka
8 (x_1, x_2, \dots, x_k) akan dibuang.

9 Ruang solusi untuk algoritma *backtracking* disusun dalam sebuah struktur berbentuk
10 pohon (*tree*), di mana setiap simpul (*node*) merepresentasikan keadaan masalah dan sisi
11 (*edge*) diberi label x_i . Jalur dari akar (*root*) ke daun (*leaf*) merepresentasikan sebuah ja-
12 waban yang mungkin, dan semua jalur yang dikumpulkan bersama-sama membentuk ruang
13 solusi. Struktur pohon ini disebut sebagai *state space tree*. Gambar 2.3 menggambarkan
14 contoh sebuah *state space tree*.

15 Langkah-langkah dalam menggunakan *state space tree* untuk mencari solusi adalah [1]:

16 1. Solusi dicari dengan membangun jalur dari akar ke daun menggunakan algoritma
17 DFS.

18 2. Simpul yang terbentuk disebut sebagai simpul hidup (*live nodes*).

19 3. Simpul yang sedang diperluas disebut sebagai *expand nodes* atau *E-nodes*.

20 4. Setiap kali sebuah *E-node* sedang diperluas, jalur yang dikembangkannya menjadi
21 lebih panjang.

- 1 5. Jika jalur yang sedang dikembangkan tidak mengarah ke solusi, maka *E-node* dimatikan dan menjadi simpul mati (*dead node*).
- 3 6. Fungsi yang digunakan untuk mematikan *E-node* adalah implementasi dari fungsi pembatas.
- 5 7. Simpul mati tidak akan diperluas.
- 6 8. Jika jalur yang sedang dibangun berakhir dengan simpul mati, proses akan mundur ke simpul sebelumnya.
- 8 9. Simpul sebelumnya terus membangkitkan simpul anak (*child node*) lainnya, yang kemudian menjadi *E-node* baru.
- 10 10. Pencarian selesai jika simpul tujuan tercapai.

11 Setiap simpul di dalam *state space tree* terkait dengan panggilan rekursif. Jika jumlah
 12 simpul di dalam pohon $2n$ atau $n!$, maka pada kasus terburuk untuk algoritma *backtracking*
 13 ini memiliki kompleksitas waktu $O(p(n)2n)$ atau $O(q(n)n!)$, dengan $p(n)$ dan $q(n)$ sebagai
 14 polinomial dengan n -derajat menyatakan waktu komputasi untuk setiap simpul.

15 Ruang solusi untuk sebuah permainan teka-teki Calcudoku dengan *grid* yang berukuran
 16 $n \times n$ adalah $X = (x_1, x_2, \dots, x_m)$, $x_i \in \{1, 2, \dots, n\}$, dengan $m = n^2$. Jadi, n adalah jumlah
 17 sel dalam satu baris atau kolom, X adalah sebuah himpunan yang merepresentasikan isi
 18 dari setiap sel dalam *grid*, dimulai pada sel pada sudut kiri atas, lalu bergerak ke sel di
 19 sebelah kanannya dalam baris yang sama, jika sudah mencapai sel yang paling kanan maka
 20 bergerak ke sel yang paling kiri pada baris dibawahnya, hingga berakhir pada sel pada sudut
 21 kanan bawah, dan S_i adalah sebuah himpunan yang berisi angka-angka dari 1 sampai n .

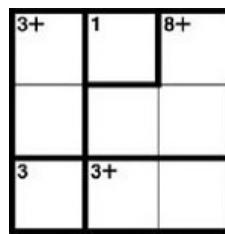
22 Fungsi pembangkit membangkitkan sebuah integer secara berurutan dari 1 sampai n
 23 sebagai x_k . Fungsi pembatas menggabungkan tiga fungsi pemeriksa pembatas (*constraint*
 24 *checking*), yaitu fungsi pemeriksa kolom (*column checking*), fungsi pemeriksa baris (*row*
 25 *checking*), dan fungsi pemeriksa *grid* (*grid checking*).

26 Fungsi pemeriksa kolom menghasilkan nilai *true* jika x_k belum ada di dalam kolom dan
 27 menghasilkan nilai *false* jika x_k sudah ada di dalam kolom.

28 Fungsi pemeriksa baris menghasilkan nilai *true* jika x_k belum ada di dalam baris dan
 29 menghasilkan nilai *false* jika x_k sudah ada di dalam baris.

30 Fungsi pemeriksa *grid* memeriksa operator pada *grid* dan memeriksa berdasarkan ope-
 31 rator yang telah ditentukan. Ada 5 operator yang digunakan dalam fungsi ini, yaitu:

- 32 1. Operator penjumlahan (+), fungsi menghasilkan nilai *true* jika hasil penjumlahan
 33 semua nilai yang ada pada *grid* ditambah dengan x_k kurang dari atau sama dengan
 34 nilai tujuan, dan menghasilkan nilai *false* jika jumlah semua nilai yang ada pada *grid*
 35 ditambah x_k lebih dari nilai tujuan.
- 36 2. Operator pengurangan (-), fungsi menghasilkan nilai *true* jika kedua sel dalam *grid*
 37 kosong, atau jika ada satu sel yang kosong dan hasil dari x_k dikurangi dengan nilai
 38 dari sel yang lainnya atau hasil dari nilai dari sel yang lainnya dikurangi dengan x_k
 39 menghasilkan nilai tujuan, dan menghasilkan nilai *false* jika ada satu sel kosong dan
 40 hasil dari x_k dikurangi dengan nilai dari sel yang lainnya atau hasil dari nilai dari sel
 41 yang lainnya dikurangi dengan x_k tidak menghasilkan nilai tujuan.

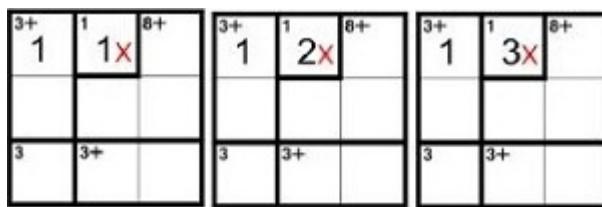


Gambar 2.4: Contoh permainan teka-teki Calcudoku dengan ukuran $grid$ 3×3 [1]

- 1 3. Operator perkalian (\times), fungsi menghasilkan nilai *true* jika hasil perkalian dari semua nilai yang ada pada $grid$ dikali dengan x_k kurang dari atau sama dengan nilai tujuan, dan menghasilkan nilai *false* jika hasil perkalian dari semua nilai yang ada pada $grid$ dikali dengan x_k lebih dari nilai tujuan.
- 5 4. Operator pembagian (\div), fungsi menghasilkan nilai *true* jika kedua sel dalam $grid$ kosong, atau jika ada satu sel yang kosong dan hasil dari x_k dibagi dengan nilai dari sel yang lainnya atau hasil dari nilai dari sel yang lainnya dibagi dengan x_k menghasilkan nilai tujuan, dan menghasilkan nilai *false* jika ada satu sel yang kosong dan hasil dari x_k dibagi dengan nilai dari sel yang lainnya atau hasil dari nilai dari sel yang lainnya dibagi dengan x_k tidak menghasilkan nilai tujuan.
- 11 5. Operator $=$, fungsi akan menghasilkan nilai *true* jika x_k sama dengan nilai tujuan, dan menghasilkan nilai *false* jika x_k tidak sama dengan nilai tujuan.

13 *State space tree* bersifat dinamis, berkembang secara terus-menerus sampai solusi ditemukan. Untuk mengilustrasikan berkembangnya *state space tree*, teka-teki Calcudoku yang digambarkan pada Gambar 2.4 akan digunakan. Berikut ini adalah tahap-tahap berkembangnya *state space tree* untuk teka-teki tersebut.

- 17 1. *State space tree* dimulai dengan *state 1* yang merepresentasikan sebuah $grid$ yang kosong.
- 19 2. Fungsi pembangkit pertama-tama akan membangkitkan angka 1 sebagai x_1 , yang akan diisikan pada sel pertama yang kosong, yaitu sel yang terletak di sudut kiri atas $grid$, atau sel pada kolom ke-1 dan baris ke-1 (*state 2*). Fungsi pembatas akan memeriksa jika langkah ini adalah langkah yang berlaku, dan ternyata langkah ini berlaku.
- 24 3. Untuk sel yang kosong berikutnya, yaitu x_2 , atau sel pada kolom ke-2 dan baris ke-1, fungsi pembangkit akan membangkitkan angka 1 (*state 3*), tetapi langkah ini gagal dalam pemeriksaan baris dalam fungsi pembatas karena angka 1 sudah pernah digunakan pada baris tersebut, ini membentuk sebuah simpul mati.
- 28 4. Fungsi pembangkit akan mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state 4*), tetapi langkah ini gagal dalam pemeriksaan $grid$ dalam fungsi pembatas karena angka 2 tidak sama dengan angka tujuan, yaitu angka 1.
- 31 5. Fungsi pembangkit akan mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state 5*), tetapi langkah ini juga gagal dalam pemeriksaan $grid$ dalam fungsi pembatas



Gambar 2.5: Ilustrasi *state* 3, 4, dan 5 pada sebuah *grid* teka-teki Calcudoku [1]

1 karena angka 3 tidak sama dengan angka tujuan, yaitu angka 1. Gambar 2.5 meng-
2 gambarkan *state* 3, *state* 4, dan *state* 5 dalam penyelesaian teka-teki Calcudoku ini.

- 3 6. Karena tidak ada solusi yang mungkin, maka mundur ke *state* 1. Fungsi pembangkit
4 akan membangkitkan kemungkinan angka berikutnya sebagai x_1 , yaitu 2, dan ternyata
5 angka 2 berlaku sebagai x_1 (*state* 6), sehingga bisa maju ke x_2 , yaitu sel pada kolom
6 ke-2 dan baris ke-1.
- 7 7. Fungsi pembangkit akan membangkitkan angka 1 (*state* 7), dan ini memenuhi syarat
8 yang ditentukan dalam fungsi pembatas, karena angka 1 sama dengan angka tujuan,
9 yaitu angka 1, sehingga bisa maju ke x_3 , yaitu sel pada kolom ke-3 dan baris ke-1.
- 10 8. Angka 1 (*state* 8) gagal dalam pemeriksaan baris karena angka 1 sudah pernah digu-
11 nakan pada baris tersebut.
- 12 9. Angka 2 (*state* 9) juga gagal dalam pemeriksaan baris karena angka 2 sudah pernah
13 digunakan pada baris tersebut.
- 14 10. Hal ini menyebabkan hanya tersisa angka 3 sebagai angka yang bisa dimasukkan ke
15 dalam x_3 (*state* 10). Karena *state* 10 ternyata berlaku, baris ke-1 telah selesai diisi,
16 dan bisa maju ke baris ke-2.
- 17 11. Langkah berikutnya adalah membuat *state* baru dengan mengisikan angka 1 pada
18 x_4 , yaitu sel pada kolom ke-1 dan baris ke-2 (*state* 11). Ini memenuhi pemeriksaan
19 pembatas, karena $2 + 1 = 3$, sehingga akan maju ke sel berikutnya, yaitu x_5 , atau sel
20 pada kolom ke-2 dan baris ke-2.
- 21 12. Angka 1 (*state* 12) jelas tidak bisa digunakan karena gagal dalam pemeriksaan ko-
22 lom dan pemeriksaan baris; angka 1 sudah pernah digunakan pada kolom dan baris
23 tersebut.
- 24 13. Angka 2 (*state* 13) adalah langkah yang berlaku, sehingga bisa maju ke sel berikutnya,
25 yaitu x_6 , atau sel pada kolom ke-3 dan baris ke-2.
- 26 14. Langkah berikutnya adalah mengisikan x_6 dengan angka 1 (*state* 14), tetapi gagal
27 dalam pemeriksaan baris karena angka 1 sudah pernah digunakan pada baris tersebut.
- 28 15. Langkah berikutnya adalah mencoba kemungkinan angka berikutnya, yaitu angka 2
29 (*state* 15), tetapi juga gagal dalam pemeriksaan baris karena angka 2 sudah pernah
30 digunakan pada baris tersebut.

$3+$	1	$8+$
2	1	3
1	3	2
3	$3+$	

Gambar 2.6: Ilustrasi *state* 19 pada sebuah *grid* teka-teki Calcudoku [1]

- 1 16. Langkah berikutnya adalah mencoba kemungkinan angka berikutnya, yaitu angka 3
2 (*state* 16), tetapi juga gagal, kali ini angka 3 gagal dalam pemeriksaan kolom karena
3 angka 3 sudah pernah digunakan pada kolom tersebut.
- 4 17. Karena semua kemungkinan angka gagal dalam pemeriksaan baris dan kolom, maka
5 akan mundur ke *state* 11 dan mencoba kemungkinan angka berikutnya, yaitu angka
6 3 (*state* 17), dan ternyata angka 3 berlaku sebagai x_5 , sehingga bisa maju ke sel
7 berikutnya, yaitu x_6 .
- 8 18. Langkah berikutnya adalah mencoba angka 1 (*state* 18) sebagai x_6 , tetapi gagal dalam
9 pemeriksaan baris karena angka 1 sudah pernah digunakan dalam baris tersebut.
- 10 19. Langkah berikutnya adalah mencoba kemungkinan angka berikutnya, yaitu angka 2
11 (*state* 19), dan ternyata angka 2 berlaku. Baris ke-2 telah selesai diisi. Gambar 2.6
12 menggambarkan *state* 19 dalam penyelesaian teka-teki Calcudoku ini.
- 13 20. Langkah berikutnya adalah mulai mengisikan sel-sel yang terletak pada baris ke-3,
14 dari kolom yang paling kiri ke kolom yang paling kanan, dimulai dengan mengisikan
15 x_7 , yaitu sel pada kolom ke-1 dan baris ke-3 dengan angka 1 (*state* 20), tetapi gagal
16 dalam pemeriksaan kolom, karena angka 1 sudah pernah digunakan dalam kolom
17 tersebut.
- 18 21. Langkah berikutnya adalah mencoba kemungkinan angka berikutnya, yaitu angka 2
19 (*state* 21), tetapi juga gagal dalam pemeriksaan kolom, karena angka 2 sudah pernah
20 digunakan dalam kolom tersebut.
- 21 22. Langkah berikutnya adalah mencoba kemungkinan angka berikutnya, yaitu angka 3
22 (*state* 22), dan ternyata berhasil, sehingga bisa maju ke sel berikutnya, yaitu x_8 , atau
23 sel pada kolom ke-2 dan baris ke-3.
- 24 23. Langkah berikutnya adalah mencoba mengisikan angka 1 pada x_8 (*state* 23), teta-
25 pi gagal dalam pemeriksaan kolom, karena angka 1 sudah pernah digunakan dalam
26 kolom tersebut.
- 27 24. Langkah berikutnya adalah mencoba kemungkinan angka berikutnya, yaitu angka 2
28 (*state* 24), dan ternyata berhasil, sehingga bisa maju ke sel berikutnya, yaitu x_9 , atau
29 sel pada kolom ke-3 dan baris ke-3.
- 30 25. x_9 adalah sel terakhir, terletak pada sudut kanan bawah *grid*. Langkah berikutnya
31 adalah mencoba mengisikan x_9 dengan angka 1 (*state* 25), dan ternyata berhasil.
32 Algoritma *backtracking* telah selesai mengisikan seluruh sel dalam *grid* dengan benar.

³⁺ 2	1	⁸⁺ 3
1	3	2
³ 3	³⁺ 2	1

Gambar 2.7: *State* 25, simpul tujuan, sebagai hasil yang dicapai [1]

1 Gambar 2.7 menggambarkan *state* 25 dalam penyelesaian teka-teki Calcudoku ini.
 2 Algoritma *backtracking* ini mencapai solusinya pada *state* 25, seperti pada *state space*
 3 *tree* yang digambarkan dalam Gambar 2.8. *State space tree* ini telah mencapai simpul
 4 tujuannya, yaitu simpul 25, dengan jalur 2-1-3-1-3-2-3-2-1.

5 Tinggi pohon yang dikembangkan untuk menyelesaikan sebuah teka-teki dengan ukuran
 6 $n \times n$ akan memiliki tinggi $n^2 + 1$ saat mencapai simpul tujuannya, dengan jalur dari simpul
 7 akar ke simpul tujuan merepresentasikan semua angka yang digunakan untuk mengisi *grid*
 8 dari sel pada sudut kiri atas ke sel pada sudut kanan bawah.

9 Singkatnya, langkah-langkah dasar dari implementasi algoritma *backtracking* dapat di-
 10 jelaskan sebagai berikut [1]:

- 11 1. Carilah sel pertama atau sel yang kosong di dalam *grid*.
- 12 2. Isilah sel dengan sebuah angka dimulai dari 1 sampai n sampai sebuah angka yang
 berlaku (*valid*) ditemukan atau sampai angka sudah melebihi n .
- 13 3. Jika angka untuk sel berlaku, ulangi langkah 1 dan 2.
- 14 4. Jika angka untuk sel sudah melebihi n dan tidak ada angka dari 1 sampai n yang
 berlaku untuk sel tersebut, mundur ke sel sebelumnya dan cobalah kemungkinan angka
 berikutnya yang berlaku untuk sel tersebut.
- 15 5. Jika tidak ada lagi sel yang kosong, solusi sudah ditemukan.

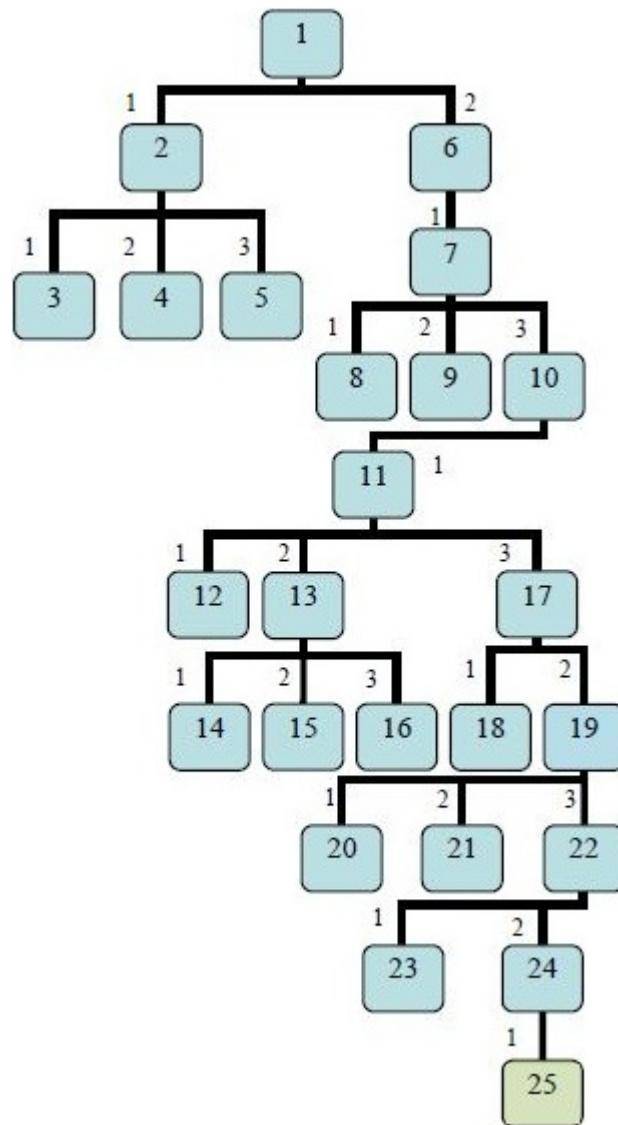
19 2.3 Algoritma *Hybrid Genetic* [2]

20 Dalam kasus ini, algoritma *hybrid genetic* adalah gabungan dari algoritma *rule based* dan
 21 algoritma genetik. Algoritma *rule based* akan dijalankan sampai pada titik dimana algorit-
 22 ma tidak bisa menyelesaikan permainan teka-teki Calcudoku. Jika algoritma sudah tidak
 23 bisa menyelesaikan permainan, maka algoritma genetik akan mulai dijalankan.

24 2.3.1 Algoritma *Rule Based*

25 Algoritma *rule based* adalah sebuah algoritma berbasis aturan logika untuk menyelesaikan
 26 teka-teki Sudoku dan variasinya, termasuk Calcudoku. Beberapa aturan logika yang digu-
 27 nakan dalam algoritma ini adalah *single square rule*, *naked subset rule*, *hidden single rule*,
 28 *evil twin rule*, *killer combination*, dan *X-wing* [2].

29 Aturan *single square* digunakan jika sebuah *cage* hanya berisi satu sel. Hal ini berarti
 30 nilai dari sel tersebut sama dengan angka tujuan yang telah ditentukan.



Gambar 2.8: *State space tree* yang dikembangkan dalam proses menyelesaikan teka-teki Calcudoku yang digambarkan pada Gambar 2.4 [1]

3	9	4	17	6	17	18	27	5
---	---	---	----	---	----	----	----	---

Gambar 2.9: Contoh bagaimana cara mendeteksi aturan *naked pair* [2]Gambar 2.10: Contoh aturan *evil twin* [2]

Aturan *naked subset* digunakan jika ada n sel dalam kolom atau baris yang sama yang mempunyai n kemungkinan nilai yang sama persis untuk mengisikannya, dengan $n \geq 2$. Hal ini berarti sel-sel lainnya dalam baris dan kolom tersebut tidak mungkin diisi dengan nilai yang sama dengan nilai milik n sel tersebut. Gambar 2.9 menunjukkan bagaimana cara kerja aturan ini. Sel-sel pada kolom ke-4 dan ke-6 mempunyai tepat dua kemungkinan nilai (1 atau 7). Ini disebut sebagai *naked pair*. Karena angka 1 dan 7 harus diisi pada sel-sel pada kolom ke-4 dan ke-6, maka angka 1 dan 7 bisa dieliminasi dari sel-sel pada kolom ke-7 dan ke-8.

Aturan *evil twin* digunakan jika sebuah *cage* berisikan dua sel, dan salah satu dari kedua sel sudah terisi, maka sel yang satunya lagi diisi dengan angka yang jika kedua angka dihitung dengan operasi matematika yang ditentukan maka akan menghasilkan angka tujuan yang ditentukan. Aturan ini adalah aturan yang paling mudah. Kenyataannya, aturan ini bisa digeneralisasikan untuk *cage* yang berukuran lebih dari 2 sel. Sel yang belum terisi yang terakhir dalam sebuah area diisi oleh sebuah nilai yang diperlukan untuk mencapai nilai tujuan menggunakan operasi matematika yang telah ditentukan. Contohnya, pada Gambar 2.10, begitu sel di sudut kiri bawah diisi oleh angka 4, maka sel diatasnya harus diisi oleh angka 9.

Aturan *hidden single* digunakan jika sebuah angka hanya bisa diisikan dalam satu sel dalam sebuah baris atau kolom. Aturan ini secara konsep cukup mudah, tetapi kadang-kadang sulit untuk diamati. Pada Gambar 2.11, nilai-nilai yang mungkin untuk sel yang paling kiri adalah 3, 5, dan 7, tetapi dalam baris ini, angka 7 harus muncul dalam salah satu selnya, dan hanya sel yang paling kiri tersebut yang memiliki kemungkinan nilai 7. Ini disebut sebagai *hidden single*. Sel tersebut harus diisi dengan angka 7.

Aturan *killer combination* adalah aturan yang paling krusial. Aturan ini digunakan jika sebuah *cage* berisikan sel-sel yang berada dalam baris atau kolom yang sama dan operasi yang ditentukan adalah penjumlahan. Kemungkinan angka yang unik untuk aturan *killer combination* berhubungan dengan ukuran *cage*. Contoh, jika sebuah *cage* memiliki dua sel dan angka tujuannya adalah 3, maka kemungkinan angka yang bisa diisikan ke dalam kedua sel tersebut adalah 1 atau 2. Hal ini berarti semua angka lainnya tidak mungkin diisikan ke dalam kedua sel tersebut. Contoh lain, jika sebuah *cage* memiliki tiga sel dan

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

Gambar 2.11: Contoh aturan *hidden single* [2]

Cage size	Cage value	Combination
2	3	1/2
2	4	1/3
2	17	8/9
2	16	7/9

Gambar 2.12: Contoh aturan *killer combination* untuk *cage* dengan ukuran 2 sel dengan operasi matematika penjumlahan [2]



Gambar 2.13: Contoh aturan *X-wing* [2]

angka tujuannya adalah 24, maka kemungkinan angka yang bisa diisi ke dalam ketiga sel tersebut adalah 7, 8, atau 9. Gambar 2.12 menampilkan contoh penerapan aturan *killer combination* untuk *cage* dengan ukuran 2 sel. Tabel ini juga bisa diperluas untuk ukuran *cage* lainnya.

Aturan *X-wing* digunakan jika hanya ada dua kemungkinan angka yang bisa diisi ke dalam dua sel yang berada di dalam dua baris yang berbeda, dan dua kemungkinan angka tersebut juga berada di dalam kolom yang sama maka sel-sel lainnya dalam kolom tersebut tidak mungkin diisi oleh dua kemungkinan angka tersebut, atau jika hanya ada dua kemungkinan angka yang bisa diisi ke dalam dua sel yang berada di dalam dua kolom yang berbeda, dan dua kemungkinan angka tersebut juga berada di dalam baris yang sama maka sel-sel lainnya dalam baris tersebut tidak mungkin diisi oleh dua kemungkinan angka tersebut. Gambar 2.13 menampilkan contoh penggunaan aturan *X-wing*. Misalnya, jika sel A diisi oleh angka 7, maka angka 7 akan dieliminasi dari sel B dan sel C. Karena sel A dengan sel C dan sel D 'terkunci', maka sel D harus diisi oleh angka 7. Jadi, angka 7 harus diisi pada sel A dan sel D atau pada sel B dan sel C. Angka 7 bisa dieliminasi dari sel-sel yang berwarna hijau.

2.3.2 Algoritma Genetik

Pencarian heuristik adalah sebuah teknik pencarian kecerdasan buatan (*artifical intelligence*) yang menggunakan heuristik dalam langkah-langkahnya. Heuristik adalah semacam aturan tidak tertulis yang mungkin menghasilkan solusi. Heuristik kadang-kadang efektif, tetapi tidak dijamin akan berhasil. dalam setiap kasus. Heuristik memerlukan peran penting dalam strategi pencarian karena sifat eksponensial dari kebanyakan masalah. Heuristik membantu mengurangi jumlah alternatif solusi dari angka yang bersifat eksponensial men-

¹ jadi angka yang bersifat polinomial. Contoh teknik pencarian heuristik adalah *Generate*
² *and Test*, *Hill Climbing*, dan *Best First Search*.

³ Algoritma genetik adalah salah satu teknik heuristik *Generate and Test* yang terinspirasi
⁴ oleh sistem seleksi alam. Algoritma ini adalah perpaduan dari bidang biologi dan ilmu
⁵ komputer. Algoritma ini memanipulasi informasi, biasanya disebut sebagai kromosom.
⁶ Kromosom ini meng-*encode* kemungkinan jawaban untuk sebuah masalah yang diberikan.
⁷ Kromosom dievaluasi dan diberi *fitness value* berdasarkan seberapa baik kromosom dalam
⁸ menyelesaikan masalah yang diberikan berdasarkan kriteria yang ditentukan oleh pembuat
⁹ program. Nilai kelayakan ini digunakan sebagai probabilitas kebertahanan hidup kromosom
¹⁰ dalam satu siklus reproduksi. Kromosom baru (kromosom anak, *child chromosome*) diproduksi
¹¹ dengan menggabungkan dua (atau lebih) kromosom orang tua (*parent chromosome*).
¹² Proses ini dirancang untuk menghasilkan kromosom-kromosom keturunan yang lebih layak,
¹³ kromosom-kromosom ini meng-*encode* jawaban yang lebih baik, sampai solusi yang baik dan
¹⁴ yang bisa diterima ditemukan.

¹⁵ Cara kerja algoritma genetik adalah sebagai berikut [2]:

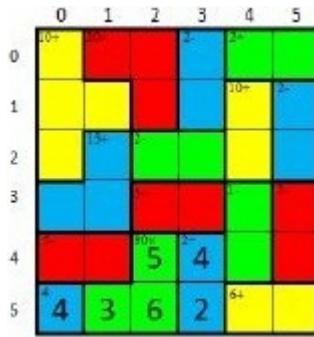
- ¹⁶ 1. Menentukan populasi kromosom kemungkinan jawaban awal.
- ¹⁷ 2. Membangkitkan populasi kemungkinan jawaban awal secara acak.
- ¹⁸ 3. Mengevaluasi fungsi objektif.
- ¹⁹ 4. Melakukan operasi terhadap kromosom menggunakan operator genetik (reproduksi,
kawin silang, dan mutasi).
- ²¹ 5. Ulangi langkah 3 dan 4 sampai mencapai kriteria untuk menghentikan algoritma.

²² Langkah-langkah utama dalam penggunaan algoritma genetik adalah membangkitkan populasi
²³ kemungkinan jawaban, mencari fungsi objektif dan fungsi kelayakan, dan penggunaan
²⁴ operator genetik.

²⁵ 2.3.3 Algoritma *Hybrid Genetic*

²⁶ Pencarian *rule based* dimulai dengan mengasumsikan semua nilai sel yang tidak diketahui
²⁷ dengan semua kemungkinan nilai untuk mengisi sel tersebut tanpa melanggar batasan,
²⁸ dengan $P(C_{b,k}) = 1, 2, \dots, n$. Setelah nilai dari satu sel sudah ditentukan, kemungkinan nilai
²⁹ untuk beberapa sel tertentu diperbarui. Misalnya, penggunaan aturan *naked single* yang
³⁰ dinyatakan dalam persamaan 1 di bawah ini, akan mengakibatkan semua kemungkinan nilai
³¹ untuk semua sel lain dalam baris yang sama dan dalam kolom yang sama harus diperbarui,
³² seperti dinyatakan dalam persamaan 2 dan 3 di bawah ini. Aturan *naked pair*, salah satu
³³ dari aturan jenis *naked subset*, dinyatakan dalam persamaan 4 untuk baris dan persamaan
³⁴ 5 untuk kolom. [2]

- ³⁵ 1. $|P(C_{b,k})| = 1 \wedge x \in P(C_{b,k}) \rightarrow V(C_{b,k}) = x$, artinya jika sebuah *cage* berukuran 1 sel,
dan x adalah nilai tujuan dari *cage* tersebut, maka nilai dari sel tersebut adalah x .
- ³⁷ 2. $(V(C_{b,k}) = x) \wedge (\forall a \in \{1, 2, \dots, n\}) \rightarrow P(C_{a,k}) = P(C_{a,k}) - \{x\}$, artinya jika nilai suatu
sel pada baris b dan kolom k adalah x , maka x dihapus dari kemungkinan angka-angka
yang bisa digunakan untuk mengisi sel-sel lain pada baris b .



Gambar 2.14: Contoh permainan teka-teki Calcudoku dengan ukuran *grid* 6 x 6 [2]

1 3. $(V(C_{b,k}) = x) \wedge (\forall q \in \{1, 2, \dots, n\}) \rightarrow P(C_{b,q}) = P(C_{b,q}) - \{x\}$ artinya jika nilai suatu
2 sel pada baris b dan kolom k adalah x , maka x dihapus dari kemungkinan angka-angka
3 yang bisa digunakan untuk mengisi sel-sel lain pada kolom k .

4 4. $|P(C_{b,k1})| = |P(C_{b,k2})| = 2 \wedge P(C_{b,k1}) = P(C_{b,k2}) \rightarrow P(C_{b,q}) = P(C_{b,q}) - P(C_{b,k1})$,
5 artinya jika ada dua sel dalam satu baris yang hanya bisa diisi oleh dua kemungkinan
6 angka, maka kedua angka tersebut dihapus dari kemungkinan angka-angka yang bisa
7 digunakan untuk mengisi sel-sel lain pada baris tersebut.

8 5. $|P(C_{b1,k})| = |P(C_{b2,k})| = 2 \wedge P(C_{b1,k}) = P(C_{b2,k}) \rightarrow P(C_{p,k}) = P(C_{p,k}) - P(C_{b1,k})$,
9 artinya jika ada dua sel dalam satu kolom yang hanya bisa diisi oleh dua kemungkinan
10 angka, maka kedua angka tersebut dihapus dari kemungkinan angka-angka yang bisa
11 digunakan untuk mengisi sel-sel lain pada kolom tersebut.

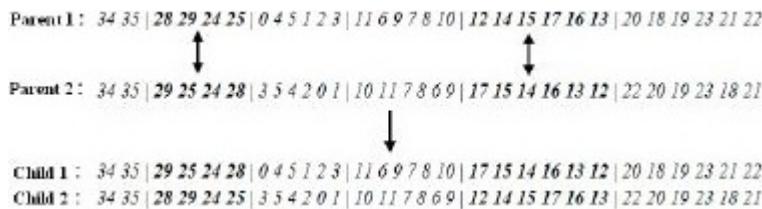
12 Algoritma genetik digunakan saat teka-teki masih tidak bisa diselesaikan setelah meng-
13 erjakan semua aturan logika secara berulang-ulang. Algoritma ini dimulai dengan meng-
14 *encode* kromosom. Satu kromosom terdiri dari k segmen, dengan $m \leq n$. Satu segmen ber-
15 isikan sekumpulan gen yang belum diselesaikan yang berada di dalam segmen tersebut. Se-
16 buah segmen merepresentasikan sebuah baris atau kolom. Dalam sebuah kromosom, segmen
17 diurutkan dari baris yang paling atas ke baris yang paling bawah atau dari kolom yang pa-
18 ling kiri ke kolom yang paling kanan. Contoh, salah satu kromosom dari permainan teka-teki
19 Calcudoku pada Gambar 2.14 adalah 34 35 | 28 29 24 25 | 0 4 5 1 2 3 | 11 6 9 7 8 10 | 12 14 15 17 16 13 | 20 18 19 2
20 Setiap segmen dalam contoh kromosom ini merepresentasikan sebuah baris yang belum ter-
21 selesaikan.

22 Fungsi objektif, yang direpresentasikan dengan x_j , akan dihitung setelah pembangkitan
23 nilai dari gen pada kromosom sudah dilakukan. Nilai untuk gen ke- j pada sebuah kromosom
24 direpresentasikan dengan w_j . x_j akan bernilai 0 jika belum diselesaikan ($w_j = 0$), dan
25 bernilai 1 jika sudah diselesaikan ($w_j \neq 0$). Untuk kromosom dengan jumlah gen k , fungsi
26 kelayakan, yaitu hasil penjumlahan dari hasil fungsi objektif untuk setiap gen dibagi dengan
27 jumlah gen, dinyatakan dalam persamaan di bawah ini [2]:

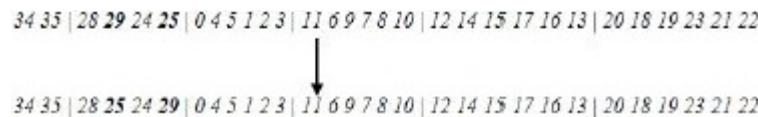
$$x_j = \begin{cases} 0 & w_j = 0 \\ 1 & w_j \neq 0 \end{cases}$$

$$\text{fitness} = \frac{\sum_{j=0}^k x_j}{k}$$

28 Jadi, solusi dari teka-teki ini adalah mencari kromosom yang nilai kelayakannya 1.



Gambar 2.15: Contoh proses kawin silang antara dua kromosom [2]



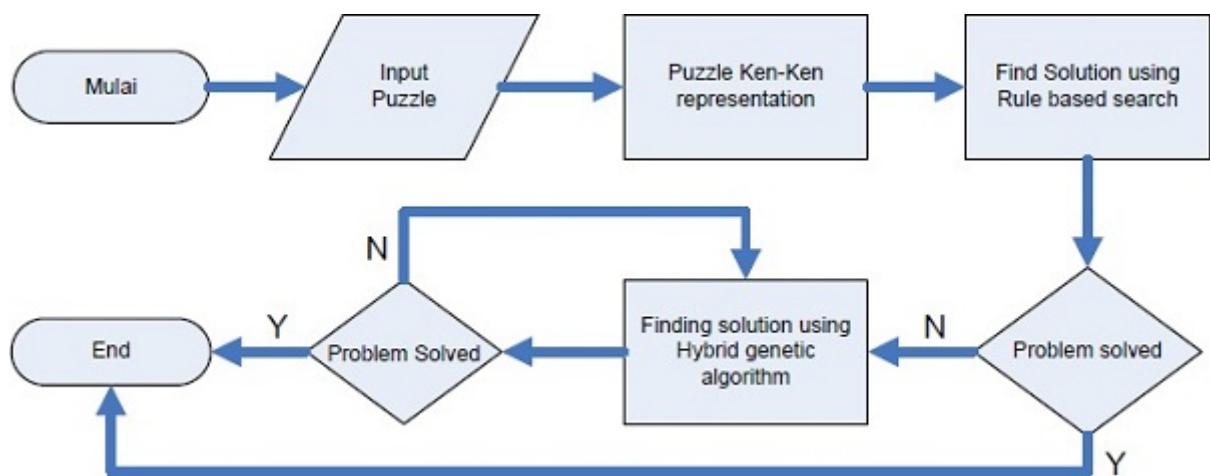
Gambar 2.16: Contoh proses mutasi [2]

1 Dalam proses reproduksi kawin silang, dua kromosom, yaitu kromosom orang tua, di-
 2 silangkan untuk membuat dua kromosom yang baru, yaitu kromosom anak, dengan meto-
 3 dologi kawin silang *N-segments*. Gambar 2.15 menggambarkan contoh proses kawin silang
 4 antara dua kromosom.

5 Pertukaran mutasi digunakan untuk mendapatkan kemungkinan kromosom yang lain.
 6 Mutasi dilakukan di antara gen yang berada dalam segmen yang sama. Gambar 2.16 adalah
 7 contoh proses mutasi antara dua gen dalam segmen yang sama.

8 Cara kerja algoritma *hybrid genetic* adalah sebagai berikut [2]:

- 9 1. Masukkan teka-teki yang akan diselesaikan sebagai input. Teka-teki Calcudoku diin-
 10 putkan oleh pemain dalam bentuk *file*.
- 11 2. Representasikan input yang dimasukkan ke dalam format teka-teki Calcudoku. File
 12 teka-teki Calcudoku yang telah diinputkan oleh pemain ditampilkan ke layar sebagai
 13 teka-teki Calcudoku.
- 14 3. Algoritma *hybrid genetic* akan mencoba menyelesaikan teka-teki tersebut dengan
 15 menggunakan algoritma *rule based* terlebih dahulu.
- 16 4. Jika berhasil menyelesaikan teka-teki tersebut dengan menggunakan algoritma *rule
 17 based*, maka algoritma selesai.
- 18 5. Jika gagal dengan menggunakan algoritma *rule based*, maka algoritma *hybrid gene-
 19 tic* akan mencoba menyelesaikan teka-teki tersebut dengan menggunakan algoritma
 20 genetik.
- 21 6. Jika berhasil menyelesaikan teka-teki tersebut dengan menggunakan algoritma gene-
 22 tik, maka algoritma selesai.
- 23 7. Jika gagal dalam menyelesaikan teka-teki tersebut setelah menggunakan algoritma
 24 genetik, artinya algoritma *hybrid genetic* gagal dalam menyelesaikan teka-teki terse-
 25 but.
- 26 26. Alur (*flow chart*) penyelesaian permainan teka-teki Calcudoku dengan menggunakan algo-
 27 ritma *hybrid genetic* dapat dilihat di Gambar 2.17.



Gambar 2.17: Alur penyelesaian permainan teka-teki Calcudoku dengan menggunakan algoritma *hybrid genetic* [2]

1

BAB 3

2

ANALISIS

3 Bab ini membahas tentang analisis cara kerja algoritma *backtracking* dan algoritma *hybrid genetic* untuk menyelesaikan permainan teka-teki Calcudoku, dan analisis kebutuhan
4 perangkat lunak Calcudoku.
5

6 **3.1 Analisis Algoritma *Backtracking***

7 Untuk mengilustrasikan cara kerja algoritma *backtracking*, akan digunakan permainan teka-
8 teki Calcudoku yang digambarkan pada Gambar 3.1 sebagai contoh.

- 9 1. Algoritma *backtracking* dimulai dengan teka-teki yang belum diselesaikan, seperti
10 yang digambarkan pada Gambar 3.1 (*state 1*).
11 2. Algoritma mengisikan sel pada baris ke-1 dan kolom ke-1 dengan angka 1 (*state 2*),
12 tetapi angka 1 tidak sesuai dengan angka tujuan dari *cage* tersebut.
13 3. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state 3*),
14 tetapi angka 2 juga tidak sesuai dengan angka tujuan dari *cage* tersebut.
15 4. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state 4*),
16 seperti dapat dilihat pada Gambar 3.2, dan ternyata angka 3 sesuai dengan angka
17 tujuan dari *cage* tersebut, sehingga algoritma dapat maju ke sel berikutnya.
18 5. Algoritma lalu mengisikan sel pada baris ke-1 dan kolom ke-2 dengan angka 1 (*state*
19 5). Algoritma lalu maju ke sel berikutnya.
20 6. Algoritma lalu mengisikan sel pada baris ke-1 dan kolom ke-3 dengan angka 1 (*state*
21 6), tetapi angka 1 sudah pernah digunakan dalam baris tersebut.

3	8+	3-	
7+			
	8+	8+	
			1

Gambar 3.1: Contoh permainan teka-teki dengan ukuran *grid* 4 x 4 yang belum diselesaikan, seperti yang digambarkan pada Gambar 2.1. [1]

³ 3	⁸⁺	³⁻	
⁷⁺			
	⁸⁺	⁸⁺	
			¹

Gambar 3.2: *State 4*

³ 3	⁸⁺ 1	³⁻ 2	4	
⁷⁺				
	⁸⁺	⁸⁺		
			¹	

Gambar 3.3: *State 11*

- 1 7. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state 7*).
 2 Algoritma lalu maju ke sel berikutnya.
- 3 8. Algoritma lalu mengisikan sel pada baris ke-1 dan kolom ke-4 dengan angka 1 (*state 8*), tetapi angka 1 sudah pernah digunakan dalam baris tersebut.
- 5 9. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state 9*), tetapi angka 2 sudah pernah digunakan dalam baris tersebut.
- 7 10. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state 10*), tetapi angka 3 sudah pernah digunakan dalam baris tersebut.
- 9 11. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 4 (*state 11*), seperti dapat dilihat pada Gambar 3.3, tetapi hasilnya tidak sesuai dengan angka tujuan dari *cage* tersebut.
- 12 12. Karena semua kemungkinan angka untuk baris ke-1 dan kolom ke-4 telah dicoba dan gagal, maka algoritma harus mundur kembali ke (*state 7*). Algoritma mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state 12*), seperti dapat dilihat pada Gambar 3.4, tetapi angka 3 sudah pernah digunakan dalam baris tersebut.

³ 3	⁸⁺ 1	³⁻ 3	
⁷⁺			
	⁸⁺	⁸⁺	
			¹

Gambar 3.4: *State 12*

³ 3	⁸⁺ 1	³⁻ 4	4
7+			
	⁸⁺	⁸⁺	
			1

Gambar 3.5: *State 17*

³ 3	⁸⁺ 2	³⁻	
7+			
	⁸⁺	⁸⁺	
			1

Gambar 3.6: *State 18*

- 1 13. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 4 (*state 13*).
 2 Algoritma lalu maju ke sel berikutnya.
- 3 14. Algoritma lalu mengisikan sel pada baris ke-1 dan kolom ke-4 dengan angka 1 (*state 14*), tetapi angka 1 sudah pernah digunakan dalam baris tersebut.
- 5 15. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state 15*),
 6 tetapi hasilnya tidak sesuai dengan angka tujuan dari *cage* tersebut.
- 7 16. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state 16*),
 8 tetapi angka 3 sudah pernah digunakan dalam baris tersebut.
- 9 17. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 4 (*state 17*),
 10 seperti dapat dilihat pada Gambar 3.5, tetapi angka 4 sudah pernah digunakan dalam
 11 baris tersebut.
- 12 18. Karena semua kemungkinan angka untuk baris ke-1 dan kolom ke-3 dan ke-4 telah
 13 dicoba dan gagal, maka algoritma harus mundur kembali ke (*state 5*). Algoritma men-
 14 coba kemungkinan angka berikutnya, yaitu angka 2 (*state 18*), seperti dapat dilihat
 15 pada Gambar 3.6. Algoritma lalu maju ke sel berikutnya.
- 16 19. Algoritma lalu mengisikan sel pada baris ke-1 dan kolom ke-3 dengan angka 1 (*state 19*), seperti dapat dilihat pada Gambar 3.7. Algoritma lalu maju ke sel berikutnya.
- 18 20. Algoritma lalu mengisikan sel pada baris ke-1 dan kolom ke-4 dengan angka 1 (*state 20*), tetapi angka 1 sudah pernah digunakan dalam baris tersebut.
- 20 21. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state 21*),
 21 tetapi angka 2 sudah pernah digunakan dalam baris tersebut.
- 22 22. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state 22*),
 23 tetapi angka 3 sudah pernah digunakan dalam baris tersebut.

³ 3	⁸⁺ 2	³⁻ 1	
7+			
	⁸⁺	⁸⁺	
			1

Gambar 3.7: *State 19*

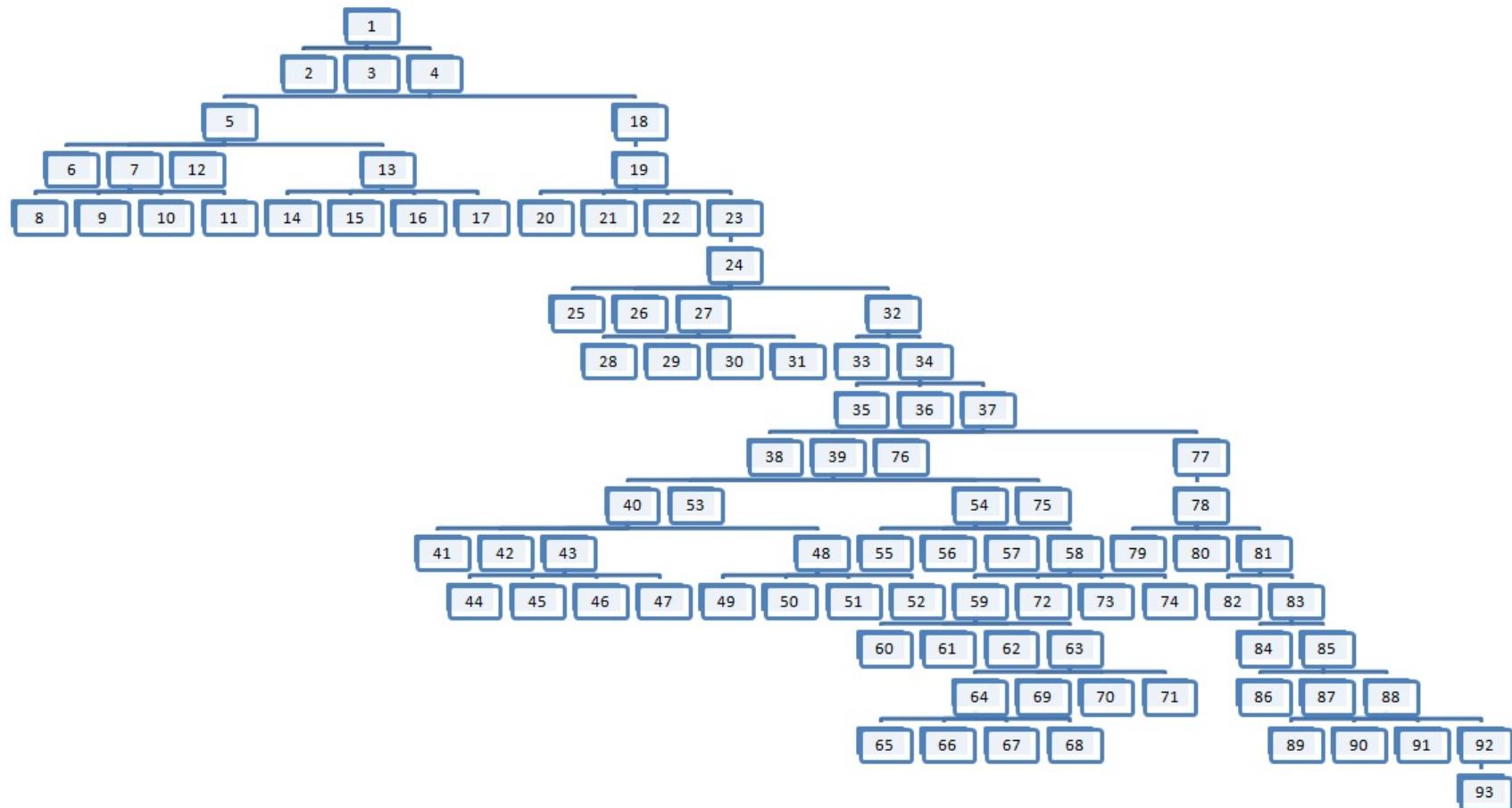
³ 3	⁸⁺ 2	³⁻ 1	4
7+			
	⁸⁺	⁸⁺	
			1

Gambar 3.8: *State 23*

- 1 23. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 4 (*state 23*), dan
 2 ternyata hasilnya sesuai dengan angka tujuan dari *cage* tersebut, seperti dapat dilihat
 3 pada Gambar 3.8. Algoritma telah selesai mengisikan baris ke-1, sehingga bisa maju
 4 ke baris berikutnya.
- 5 24. Langkah-langkah di atas diulang untuk mengisi sel-sel pada baris-baris selanjutnya.
 6 Algoritma *backtracking* berhasil mengisi semua sel dalam permainan teka-teki Calcu-
 7 doku ini dengan benar pada *state 93*, seperti dapat dilihat pada Gambar 3.9.
- 8 Algoritma ini mencapai solusinya pada state 93, seperti pada *state space tree* yang
 9 digambarkan dalam Gambar 3.10. *State space tree* ini telah mencapai simpul tujuannya,
 10 yaitu simpul 93, dengan jalur 3-2-1-4-1-4-2-3-4-1-3-2-2-3-4-1. Penjelasan tentang analisis
 11 algoritma *backtracking* secara lengkap dapat dilihat di Lampiran A.

³ 3	⁸⁺ 2	³⁻ 1	4
7+ 1	4	2	3
4	⁸⁺ 1	⁸⁺ 3	2
2	3	4	¹ 1

Gambar 3.9: *State 93*



Gambar 3.10: *State space tree* yang dikembangkan dalam proses menyelesaikan teka-teki Calcudoku yang digambarkan pada Gambar 3.1

4-	1-		1-	15+	
	30*				1-
2-	2/		1-	3/	
		24*			1
5+			7+	60*	
	1-				

Gambar 3.11: Contoh permainan teka-teki Calcudoku dengan ukuran *grid* 6 x 6 yang belum diselesaikan, seperti yang digambarkan pada Gambar 1.2. [2]

3.2 Analisis Algoritma *Hybrid Genetic*

Untuk mengilustrasikan cara kerja algoritma *hybrid genetic*, akan digunakan permainan teka-teki Calcudoku yang digambarkan pada Gambar 3.11 sebagai contoh. Algoritma *hybrid genetic* dimulai dengan mencoba menyelesaikan permainan teka-teki Calcudoku dengan algoritma *rule based*.

3.2.1 Algoritma *Rule Based*

Sel pada baris ke-4 dan kolom ke-6 adalah bagian dari sebuah *cage* yang berukuran hanya 1 sel, dan oleh karena itu, angka tujuan dari sel tersebut adalah angka tujuan dari *cage* tersebut (aturan *single square*). Angka tujuan dari *cage* tersebut adalah 1, dan oleh karena itu sel tersebut dapat langsung diisi dengan angka 1, seperti dapat dilihat pada Gambar 3.12.

Sayangnya, algoritma *rule based* gagal dalam mengisi sel-sel lainnya berdasarkan aturan-aturan yang telah didefinisikan setelah beberapa kali percobaan, sehingga algoritma *hybrid genetic* akan mencoba menyelesaikan teka-teki Calcudoku dengan algoritma genetik.

3.2.2 Algoritma Genetik

Dalam contoh ini, parameter-parameter untuk algoritma genetik yang akan digunakan untuk teka-teki Calcudoku ini ditunjukkan pada Tabel 3.1. Dalam kasus ini, parameter ditentukan oleh pembuat program (penulis). Setiap generasi terdiri dari 12 kromosom. $40\% \times 12 \approx 5$ kromosom diambil dari generasi sebelumnya (*elitism*). $50\% \times 12 \approx 6$ kromosom adalah hasil dari pembentukan kromosom-kromosom baru dengan operasi kawin silang, dan $10\% \times 12 \approx 1$ kromosom adalah hasil dari pembentukan kromosom-kromosom baru dengan operasi mutasi. Untuk mengilustrasikan cara kerja algoritma genetik, hanya 3 generasi pertama yang akan dibahas.

Setiap sel mempunyai nilai kelayakan. Nilai kelayakan dari sebuah sel akan bernilai 1 jika nilai dari semua sel yang merupakan bagian dari *cage* yang salah satu selnya adalah sel tersebut menghasilkan nilai tujuan setelah dihitung menggunakan operator yang telah ditentukan dan tidak ada pengulangan angka di dalam baris tersebut maupun kolom tersebut, dan bernilai 0 jika nilai dari semua sel yang merupakan bagian dari *cage* yang salah satu selnya adalah sel tersebut tidak menghasilkan nilai tujuan setelah dihitung menggunakan

4-	1-		1-	15+	
	30*				1-
2-	2/		1-	3/	
		24*			1
5+			7+	60*	
	1-				

Gambar 3.12: Permainan teka-teki Calcudoku setelah diselesaikan dengan algoritma *rule based*

Tabel 3.1: Tabel parameter untuk algoritma genetik yang akan digunakan untuk menyelesaikan teka-teki Calcudoku yang digambarkan pada Gambar 3.12

Parameter	Nilai
Ukuran Populasi	12
Probabilitas <i>Elitism</i>	40%
Probabilitas Kawin Silang	50%
Probabilitas Mutasi	10%

operator yang telah ditentukan atau ada pengulangan angka di dalam baris tersebut maupun kolom tersebut. Nilai kelayakan sel untuk setiap sel dalam sebuah baris dijumlahkan, lalu dibagi dengan jumlah kolom dalam baris tersebut, dan hasilnya adalah nilai kelayakan baris. Nilai kelayakan baris untuk setiap baris dalam sebuah teka-teki dijumlahkan, lalu dibagi dengan jumlah baris dalam teka-teki tersebut, dan hasilnya adalah nilai kelayakan teka-teki.

Algoritma genetik dimulai dengan membangkitkan kromosom-kromosom baru sebanyak ukuran populasi yang telah ditentukan. Dalam contoh ini, ukuran populasi adalah 12, maka algoritma akan membangkitkan 12 kromosom baru. Ke-12 kromosom awal ini adalah bagian dari generasi pertama. Gambar 3.13 menggambarkan Kromosom 1, gambar 3.14 menggambarkan Kromosom 2, gambar 3.15 menggambarkan Kromosom 3, gambar 3.16 menggambarkan Kromosom 4, gambar 3.17 menggambarkan Kromosom 5, gambar 3.18 menggambarkan Kromosom 6, gambar 3.19 menggambarkan Kromosom 7, gambar 3.20 menggambarkan Kromosom 8, gambar 3.21 menggambarkan Kromosom 9, gambar 3.22 menggambarkan Kromosom 10, gambar 3.23 menggambarkan Kromosom 11, dan gambar 3.24 menggambarkan Kromosom 12.

⁴⁻ 1	¹⁻ 3	5	¹⁻ 6	¹⁵⁺ 4	2
3	^{30*} 4	2	5	1	¹⁻ 6
²⁻ 2	^{2/} 1	6	¹⁻ 4	^{3/} 3	5
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 4	6	1	⁷⁺ 2	^{60*} 5	3
5	¹ 2	3	1	6	4

Gambar 3.13: Kromosom 1 dalam Generasi ke-1

⁴⁻ 3	¹⁻ 2	1	¹⁻ 6	¹⁵⁺ 5	4
1	^{30*} 6	2	5	4	¹⁻ 3
²⁻ 5	^{2/} 3	6	¹⁻ 4	^{3/} 1	2
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 4	1	3	⁷⁺ 2	^{60*} 6	5
2	¹ 4	5	1	3	6

Gambar 3.14: Kromosom 2 dalam Generasi ke-1

⁴⁻ 4	¹⁻ 3	6	¹⁻ 2	¹⁵⁺ 1	5
6	^{30*} 5	1	3	2	¹⁻ 4
²⁻ 5	^{2/} 1	4	¹⁻ 6	^{3/} 3	2
3	6	^{24*} 2	5	4	¹ 1
⁵⁺ 1	2	3	⁷⁺ 4	^{60*} 5	6
2	¹ 4	5	1	6	3

Gambar 3.15: Kromosom 3 dalam Generasi ke-1

⁴⁻ 5	¹⁻ 1	4	¹⁻ 6	¹⁵⁺ 2	3
1	^{30*} 2	3	4	5	¹⁻ 6
²⁻ 6	^{2/} 3	5	¹⁻ 2	^{3/} 1	4
2	4	^{24*} 6	5	3	¹ 1
⁵⁺ 4	5	1	⁷⁺ 6	^{60*} 3	2
3	¹ 6	2	1	4	5

Gambar 3.16: Kromosom 4 dalam Generasi ke-1

⁴⁻ 5	¹⁻ 3	2	¹⁻ 6	¹⁵⁺ 1	4
1	^{30*} 6	3	5	4	¹⁻ 2
²⁻ 3	^{2/} 2	1	¹⁻ 4	^{3/} 6	5
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 4	1	6	⁷⁺ 2	^{60*} 5	3
2	¹⁻ 4	5	1	3	6

Gambar 3.17: Kromosom 5 dalam Generasi ke-1

⁴⁻ 6	¹⁻ 3	5	¹⁻ 2	¹⁵⁺ 1	4
5	^{30*} 1	4	6	2	¹⁻ 3
²⁻ 2	^{2/} 4	6	¹⁻ 1	^{3/} 3	5
3	6	^{24*} 2	5	4	¹ 1
⁵⁺ 1	2	3	⁷⁺ 4	^{60*} 5	6
4	¹⁻ 5	1	3	6	2

Gambar 3.18: Kromosom 6 dalam Generasi ke-1

⁴⁻ 1	¹⁻ 3	2	¹⁻ 6	¹⁵⁺ 4	5
3	^{30*} 4	1	5	6	¹⁻ 2
²⁻ 5	^{2/} 2	6	¹⁻ 4	^{3/} 1	3
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 4	3	1	⁷⁺ 2	^{60*} 5	6
2	¹⁻ 6	5	1	3	4

Gambar 3.19: Kromosom 7 dalam Generasi ke-1

⁴⁻ 3	¹⁻ 1	5	¹⁻ 6	¹⁵⁺ 4	2
2	^{30*} 6	3	5	1	¹⁻ 4
²⁻ 1	^{2/} 2	6	¹⁻ 4	^{3/} 3	5
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 5	4	1	⁷⁺ 2	^{60*} 6	3
4	¹⁻ 3	2	1	5	6

Gambar 3.20: Kromosom 8 dalam Generasi ke-1

⁴⁻ 4	¹⁻ 6	5	¹⁻ 3	¹⁵⁺ 1	2
3	^{30*} 5	1	6	2	¹⁻ 4
²⁻ 6	^{2/} 1	4	¹⁻ 2	^{3/} 3	5
2	3	^{24*} 6	5	4	¹ 1
⁵⁺ 1	2	3	⁷⁺ 4	^{60*} 5	6
5	¹⁻ 4	2	1	6	3

Gambar 3.21: Kromosom 9 dalam Generasi ke-1

⁴⁻ 5	¹⁻ 1	3	¹⁻ 6	¹⁵⁺ 4	2
3	^{30*} 6	2	5	1	¹⁻ 4
²⁻ 2	^{2/} 3	1	¹⁻ 4	^{3/} 6	5
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 1	4	6	⁷⁺ 2	^{60*} 5	3
4	¹⁻ 2	5	1	3	6

Gambar 3.22: Kromosom 10 dalam Generasi ke-1

⁴⁻ 5	¹⁻ 1	6	¹⁻ 2	¹⁵⁺ 4	3
1	^{30*} 2	3	4	5	¹⁻ 6
²⁻ 4	^{2/} 3	5	¹⁻ 6	^{3/} 1	2
6	4	^{24*} 2	5	3	¹ 1
⁵⁺ 2	5	1	⁷⁺ 3	^{60*} 6	4
3	¹⁻ 6	4	1	2	5

Gambar 3.23: Kromosom 11 dalam Generasi ke-1

⁴⁻ 1	¹⁻ 5	6	¹⁻ 4	¹⁵⁺ 3	2
3	^{30*} 6	1	2	4	¹⁻ 5
²⁻ 4	^{2/} 1	5	¹⁻ 3	^{3/} 2	6
2	3	^{24*} 4	5	6	¹ 1
⁵⁺ 5	4	2	⁷⁺ 6	^{60*} 1	3
6	¹⁻ 2	3	1	5	4

Gambar 3.24: Kromosom 12 dalam Generasi ke-1

Tabel 3.2: Tabel nilai kelayakan untuk kromosom-kromosom pada Generasi ke-1

Nomor Kromosom	Nilai Kelayakan
1	0,3333
2	0,3056
3	0,25
4	0,2222
5	0,4444
6	0,1389
7	0,3889
8	0,25
9	0,1389
10	0,3056
11	0,3889
12	0,5556

1 Berdasarkan nilai kelayakan untuk kromosom-kromosom pada Generasi ke-1 yang di-
 2 tampilkan pada Tabel 3.2, 5 kromosom terbaik akan diambil untuk menjadi bagian dari
 3 Generasi ke-2. Ke-5 kromosom yang terpilih adalah Kromosom 12, Kromosom 5, Kromo-
 4 som 7, Kromosom 11, dan Kromosom 1.

5 Untuk Generasi ke-2, 5 kromosom adalah 5 kromosom terbaik dari Generasi ke-1, 6
 6 kromosom adalah hasil kawin silang dari 2 kromosom dari Generasi ke-1, dan 1 kromosom
 7 adalah hasil mutasi dari 1 kromosom dari Generasi ke-1.

8 Gambar 3.25 menggambarkan Kromosom 1, yaitu Kromosom 12 dari Generasi ke-1,
 9 gambar 3.26 menggambarkan Kromosom 2, yaitu Kromosom 5 dari Generasi ke-1, gam-
 10 bar 3.27 menggambarkan Kromosom 3, yaitu Kromosom 7 dari Generasi ke-1, gambar 3.28
 11 menggambarkan Kromosom 4, yaitu Kromosom 11 dari Generasi ke-1, gambar 3.29 meng-
 12 gambarkan Kromosom 5, yaitu Kromosom 1 dari Generasi ke-1, gambar 3.30 menggambark-
 13 an Kromosom 6, yaitu hasil kawin silang dari Kromosom 5 dan Kromosom 12 dari Generasi
 14 ke-1, gambar 3.31 menggambarkan Kromosom 7, yaitu hasil kawin silang dari Kromosom
 15 3 dan Kromosom 8 dari Generasi ke-1, gambar 3.32 menggambarkan Kromosom 8, yaitu
 16 hasil kawin silang dari Kromosom 7 dan Kromosom 10 dari Generasi ke-1, gambar 3.33
 17 menggambarkan Kromosom 9, yaitu hasil kawin silang dari Kromosom 7 dan Kromosom
 18 10 dari Generasi ke-1, gambar 3.34 menggambarkan Kromosom 10, yaitu hasil kawin silang
 19 dari Kromosom 2 dan Kromosom 5 dari Generasi ke-1, gambar 3.35 menggambarkan Kro-
 20 mosom 11, yaitu hasil kawin silang dari Kromosom 2 dan Kromosom 5 dari Generasi ke-1,
 21 dan gambar 3.36 menggambarkan Kromosom 12, yaitu hasil mutasi dari Kromosom 12 dari
 22 Generasi ke-1.

1	¹⁻ 5	6	4	¹⁵⁺ 3	2
3	^{30*} 6	1	2	4	¹⁻ 5
²⁻ 4	^{2/} 1	5	¹⁻ 3	^{3/} 2	6
2	3	^{24*} 4	5	6	¹ 1
⁵⁺ 5	4	2	⁷⁺ 6	^{60*} 1	3
6	¹⁻ 2	3	1	5	4

Gambar 3.25: Kromosom 1 dalam Generasi ke-2

⁴⁻ 5	¹⁻ 3	2	¹⁻ 6	¹⁵⁺ 1	4
1	^{30*} 6	3	5	4	¹⁻ 2
²⁻ 3	^{2/} 2	1	¹⁻ 4	^{3/} 6	5
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 4	1	6	⁷⁺ 2	^{60*} 5	3
2	¹⁻ 4	5	1	3	6

Gambar 3.26: Kromosom 2 dalam Generasi ke-2

⁴⁻ 1	¹⁻ 3	2	¹⁻ 6	¹⁵⁺ 4	5
3	^{30*} 4	1	5	6	¹⁻ 2
²⁻ 5	^{2/} 2	6	¹⁻ 4	^{3/} 1	3
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 4	1	3	⁷⁺ 2	^{60*} 5	6
2	¹⁻ 6	5	1	3	4

Gambar 3.27: Kromosom 3 dalam Generasi ke-2

⁴⁻ 5	¹⁻ 1	6	¹⁻ 2	¹⁵⁺ 4	3
1	^{30*} 2	3	4	5	¹⁻ 6
²⁻ 4	^{2/} 3	5	¹⁻ 6	^{3/} 1	2
6	4	^{24*} 2	5	3	¹ 1
⁵⁺ 2	5	1	⁷⁺ 3	^{60*} 6	4
3	¹⁻ 6	4	1	2	5

Gambar 3.28: Kromosom 4 dalam Generasi ke-2

⁴⁻ 1	¹⁻ 3	5	¹⁻ 6	¹⁵⁺ 4	2
3	^{30*} 4	2	5	1	¹⁻ 6
²⁻ 2	^{2/} 1	6	¹⁻ 4	^{3/} 3	5
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 4	6	1	⁷⁺ 2	^{60*} 5	3
5	¹⁻ 2	3	1	6	4

Gambar 3.29: Kromosom 5 dalam Generasi ke-2

⁴⁻ 3	¹⁻ 2	1	¹⁻ 6	¹⁵⁺ 5	4
1	^{30*} 6	2	5	4	¹⁻ 3
²⁻ 2	^{2/} 3	1	¹⁻ 4	^{3/} 6	5
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 1	4	6	⁷⁺ 2	^{60*} 5	3
2	¹⁻ 4	5	1	3	6

Gambar 3.30: Kromosom 6 dalam Generasi ke-2

⁴⁻ 5	¹⁻ 1	3	¹⁻ 6	¹⁵⁺ 4	2
3	^{30*} 6	2	5	1	¹⁻ 4
²⁻ 5	^{2/} 3	6	¹⁻ 4	^{3/} 1	2
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 4	1	3	⁷⁺ 2	^{60*} 6	5
4	¹⁻ 2	5	1	3	6

Gambar 3.31: Kromosom 7 dalam Generasi ke-2

⁴⁻ 1	¹⁻ 3	2	¹⁻ 6	¹⁵⁺ 4	5
3	^{30*} 4	1	5	6	¹⁻ 2
²⁻ 2	^{2/} 3	1	¹⁻ 4	^{3/} 6	5
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 1	4	6	⁷⁺ 2	^{60*} 5	3
2	¹⁻ 6	1	5	3	4

Gambar 3.32: Kromosom 8 dalam Generasi ke-2

⁴⁻ 5	¹⁻ 1	3	¹⁻ 6	¹⁵⁺ 4	2
3	^{30*} 6	2	5	1	¹⁻ 4
²⁻ 5	^{2/} 2	6	¹⁻ 4	^{3/} 1	3
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 4	3	1	⁷⁺ 2	^{60*} 5	6
4	¹⁻ 2	5	1	3	6

Gambar 3.33: Kromosom 9 dalam Generasi ke-2

⁴⁻ 3	¹⁻ 2	1	¹⁻ 6	¹⁵⁺ 5	4
1	^{30*} 6	3	5	4	¹⁻ 2
²⁻ 3	^{2/} 2	1	¹⁻ 4	^{3/} 6	5
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 4	1	3	⁷⁺ 2	^{60*} 6	5
2	¹⁻ 4	5	1	3	6

Gambar 3.34: Kromosom 10 dalam Generasi ke-2

⁴⁻ 5	¹⁻ 3	2	¹⁻ 6	¹⁵⁺ 1	4
1	^{30*} 6	2	5	4	¹⁻ 3
²⁻ 5	^{2/} 3	6	¹⁻ 4	^{3/} 1	2
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 4	1	6	⁷⁺ 2	^{60*} 5	3
2	¹⁻ 4	5	1	3	6

Gambar 3.35: Kromosom 11 dalam Generasi ke-2

⁴⁻ 1	¹⁻ 5	6	¹⁻ 4	¹⁵⁺ 3	2
3	^{30*} 6	1	2	4	¹⁻ 5
²⁻ 4	^{2/} 1	5	¹⁻ 3	^{3/} 2	6
2	3	^{24*} 4	5	6	¹ 1
⁵⁺ 5	4	2	⁷⁺ 6	^{60*} 1	3
1	¹⁻ 2	3	6	5	4

Gambar 3.36: Kromosom 12 dalam Generasi ke-2

Tabel 3.3: Tabel nilai kelayakan untuk kromosom-kromosom pada Generasi ke-2

Nomor Kromosom	Nilai Kelayakan
1	0,5556
2	0,4444
3	0,3889
4	0,3889
5	0,3333
6	0,1944
7	0,1389
8	0,0833
9	0,25
10	0,1944
11	0,1944
12	0,5

Berdasarkan nilai kelayakan untuk kromosom-kromosom pada Generasi ke-2 yang ditampilkan pada Tabel 3.3, 5 kromosom terbaik akan diambil untuk menjadi bagian dari Generasi ke-3. Ke-5 kromosom yang terpilih adalah Kromosom 1, Kromosom 12, Kromosom 2, Kromosom 3, dan Kromosom 4.

Untuk Generasi ke-3, 5 kromosom adalah 5 kromosom terbaik dari Generasi ke-2, 6 kromosom adalah hasil kawin silang dari 2 kromosom dari Generasi ke-2, dan 1 kromosom adalah hasil mutasi dari 1 kromosom dari Generasi ke-2.

Gambar 3.37 menggambarkan Kromosom 1, yaitu Kromosom 1 dari Generasi ke-2, Gambar 3.38 menggambarkan Kromosom 2, yaitu Kromosom 12 dari Generasi ke-2, Gambar 3.39 menggambarkan Kromosom 3, yaitu Kromosom 2 dari Generasi ke-2, Gambar 3.40 menggambarkan Kromosom 4, yaitu Kromosom 3 dari Generasi ke-2, Gambar 3.41 menggambarkan Kromosom 5, yaitu Kromosom 4 dari Generasi ke-2, Gambar 3.42 menggambarkan Kromosom 6, yaitu hasil kawin silang dari Kromosom 2 dan Kromosom 12 dari Generasi ke-2, Gambar 3.43 menggambarkan Kromosom 7, yaitu hasil kawin silang dari Kromosom 2 dan Kromosom 12 dari Generasi ke-2, Gambar 3.44 menggambarkan Kromosom 8, yaitu hasil kawin silang dari Kromosom 5 dan Kromosom 9 dari Generasi ke-2, Gambar 3.45 menggambarkan Kromosom 9, yaitu hasil kawin silang dari Kromosom 5 dan Kromosom 9 dari Generasi ke-2, Gambar 3.46 menggambarkan Kromosom 10, yaitu hasil kawin silang dari Kromosom 5 dan Kromosom 6 dari Generasi ke-2, Gambar 3.47 menggambarkan Kromosom 11, yaitu hasil kawin silang dari Kromosom 5 dan Kromosom 6 dari Generasi ke-2, dan Gambar 3.48 menggambarkan Kromosom 12, yaitu hasil mutasi dari Kromosom 2 dari Generasi ke-2.

⁴⁻ 1	¹⁻ 5	6	¹⁻ 4	¹⁵⁺ 3	2
3	^{30*} 6	1	2	4	¹⁻ 5
²⁻ 4	^{2/} 1	5	¹⁻ 3	^{3/} 2	6
2	3	^{24*} 4	5	6	¹ 1
⁵⁺ 5	4	2	⁷⁺ 6	^{60*} 1	3
6	¹⁻ 2	3	1	5	4

Gambar 3.37: Kromosom 1 dalam Generasi ke-3

⁴⁻ 1	¹⁻ 5	6	4	¹⁵⁺ 3	2
3	^{30*} 6	1	2	4	¹⁻ 5
²⁻ 4	^{2/} 1	5	¹⁻ 3	^{3/} 2	6
2	3	^{24*} 4	5	6	¹ 1
⁵⁺ 5	4	2	⁷⁺ 6	^{60*} 1	3
1	¹⁻ 2	3	6	5	4

Gambar 3.38: Kromosom 2 dalam Generasi ke-3

⁴⁻ 5	¹⁻ 3	2	¹⁻ 6	¹⁵⁺ 1	4
1	^{30*} 6	3	5	4	¹⁻ 2
²⁻ 3	^{2/} 2	1	¹⁻ 4	^{3/} 6	5
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 4	1	6	⁷⁺ 2	^{60*} 5	3
2	¹⁻ 4	5	1	3	6

Gambar 3.39: Kromosom 3 dalam Generasi ke-3

⁴⁻ 1	¹⁻ 3	2	¹⁻ 6	¹⁵⁺ 4	5
3	^{30*} 4	1	5	6	¹⁻ 2
²⁻ 5	^{2/} 2	6	¹⁻ 4	^{3/} 1	3
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 4	1	3	⁷⁺ 2	^{60*} 5	6
2	¹⁻ 6	5	1	3	4

Gambar 3.40: Kromosom 4 dalam Generasi ke-3

⁴⁻ 5	¹⁻ 1	6	¹⁻ 2	¹⁵⁺ 4	3
1	^{30*} 2	3	4	5	¹⁻ 6
²⁻ 4	^{2/} 3	5	¹⁻ 6	^{3/} 1	2
6	4	^{24*} 2	5	3	¹ 1
⁵⁺ 2	5	1	⁷⁺ 3	^{60*} 6	4
3	¹⁻ 6	4	1	2	5

Gambar 3.41: Kromosom 5 dalam Generasi ke-3

⁴⁻ 5	¹⁻ 3	2	¹⁻ 6	¹⁵⁺ 1	4
3	^{30*} 6	1	2	4	¹⁻ 5
²⁻ 4	^{2/} 1	5	¹⁻ 3	^{3/} 2	6
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 4	1	6	⁷⁺ 2	^{60*} 5	3
1	¹⁻ 2	3	6	5	4

Gambar 3.42: Kromosom 6 dalam Generasi ke-3

⁴⁻ 1	¹⁻ 5	6	¹⁻ 4	¹⁵⁺ 3	2
1	^{30*} 6	3	5	4	¹⁻ 2
²⁻ 3	^{2/} 2	1	¹⁻ 4	^{3/} 6	5
2	3	^{24*} 4	5	6	¹ 1
⁵⁺ 5	4	2	⁷⁺ 6	^{60*} 3	1
2	¹⁻ 4	5	1	3	6

Gambar 3.43: Kromosom 7 dalam Generasi ke-3

⁴⁻ 1	¹⁻ 3	5	¹⁻ 6	¹⁵⁺ 4	2
3	^{30*} 4	2	5	1	¹⁻ 6
²⁻ 2	^{2/} 1	6	¹⁻ 4	^{3/} 3	5
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 4	3	1	⁷⁺ 2	^{60*} 5	6
4	¹⁻ 2	5	1	3	6

Gambar 3.44: Kromosom 8 dalam Generasi ke-3

⁴⁻ 5	¹⁻ 1	3	¹⁻ 6	¹⁵⁺ 4	2
3	^{30*} 6	2	5	1	¹⁻ 4
²⁻ 5	^{2/} 2	6	¹⁻ 4	^{3/} 1	3
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 4	6	1	⁷⁺ 2	^{60*} 5	3
5	¹⁻ 2	3	1	6	4

Gambar 3.45: Kromosom 9 dalam Generasi ke-3

⁴⁻ 1	¹⁻ 3	5	¹⁻ 6	¹⁵⁺ 4	2
1	^{30*} 6	2	5	4	¹⁻ 3
²⁻ 2	^{2/} 1	6	¹⁻ 4	^{3/} 3	5
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 1	4	6	⁷⁺ 2	^{60*} 5	3
2	¹⁻ 4	5	1	3	6

Gambar 3.46: Kromosom 10 dalam Generasi ke-3

⁴⁻ 3	¹⁻ 2	1	¹⁻ 6	¹⁵⁺ 5	4
3	^{30*} 4	2	5	1	¹⁻ 6
²⁻ 2	^{2/} 3	1	¹⁻ 4	^{3/} 6	5
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 4	6	1	⁷⁺ 2	^{60*} 5	3
5	¹⁻ 2	3	1	6	4

Gambar 3.47: Kromosom 11 dalam Generasi ke-3

⁴⁻ 5	¹⁻ 3	2	¹⁻ 6	¹⁵⁺ 1	4
1	^{30*} 6	3	5	4	¹⁻ 2
²⁻ 3	^{2/} 2	1	¹⁻ 4	^{3/} 6	5
6	5	^{24*} 4	3	2	¹ 1
⁵⁺ 4	1	6	⁷⁺ 2	^{60*} 5	3
1	¹⁻ 4	5	2	3	6

Gambar 3.48: Kromosom 12 dalam Generasi ke-3

Tabel 3.4: Tabel nilai kelayakan untuk kromosom-kromosom pada Generasi ke-3

Nomor Kromosom	Nilai Kelayakan
1	0,5556
2	0,5
3	0,4444
4	0,3889
5	0,3889
6	0,2778
7	0,1389
8	0,1389
9	0,1389
10	0,1389
11	0,1944
12	0,3889

1 Berdasarkan nilai kelayakan untuk kromosom-kromosom pada Generasi ke-3 yang di-
 2 tampilkan pada Tabel 3.4, 5 kromosom terbaik akan diambil untuk menjadi bagian dari
 3 Generasi ke-4. Ke-5 kromosom yang terpilih adalah Kromosom 1, Kromosom 2, Kromosom
 4 3, Kromosom 4, dan Kromosom 12.

5 Proses ini diulang untuk menghasilkan generasi-generasi berikutnya, sampai algoritma
 6 genetik dapat menemukan solusi dari teka-teki Calcudoku tersebut.

7 3.3 Perangkat Lunak

8 Berdasarkan landasan teori dan analisis algoritma *backtracking* dan *hybrid genetic* untuk
 9 menyelesaikan permainan teka-teki Calcudoku yang telah dilakukan, perangkat lunak Cal-
 10 cudoku akan dibuat. Perangkat lunak ini akan menerima masukan dalam bentuk *file* yang
 11 berisi:

- 12 1. Ukuran *grid*.
- 13 2. Jumlah *cage*.
- 14 3. Matriks *cage assignment*, yang merepresentasikan posisi dari setiap *cage* dalam *grid*.
- 15 4. Matriks *cage objectives*, yang berisikan angka tujuan dan operasi matematika yang
 16 telah ditentukan untuk setiap *cage*.

17 Perangkat lunak ini akan menghasilkan keluaran berupa antarmuka grafis permainan
 18 teka-teki Calcudoku berdasarkan isi *file* yang di-*load* oleh pengguna. Permainan ini dapat
 19 diselesaikan oleh pengguna dengan usahanya sendiri, atau menggunakan salah satu dari
 20 dua *solver* yang disediakan. Kedua *solver* tersebut yaitu:

- 21 1. Algoritma *backtracking*, dan
- 22 2. Algoritma *hybrid genetic*.

23 Pengguna dapat me-*load* file masukan untuk memulai permainan, me-*reset* permainan
 24 untuk mengulang permainan berdasarkan *file* masukan yang sudah di-*load* dari awal, dan

menutup *file* masukan untuk mengakhiri permainan, atau jika ingin me-*load* *file* masukan yang lain. Pengguna juga dapat meminta perangkat lunak untuk memeriksa permainan jika ada masukan yang salah di dalam *grid*, misalnya ada angka yang berulang dalam sebuah baris atau kolom, atau angka-angka dalam sebuah *cage* tidak mencapai angka tujuan yang ditentukan setelah dihitung dengan operasi matematika yang ditentukan. Pemain juga dapat mengatur nilai dari parameter-parameter untuk algoritma genetik.

Kebutuhan-kebutuhan yang diperlukan oleh perangkat lunak ini akan dijelaskan menggunakan diagram *use case*, dan skenario.

3.3.1 Diagram *Use Case* dan Skenario

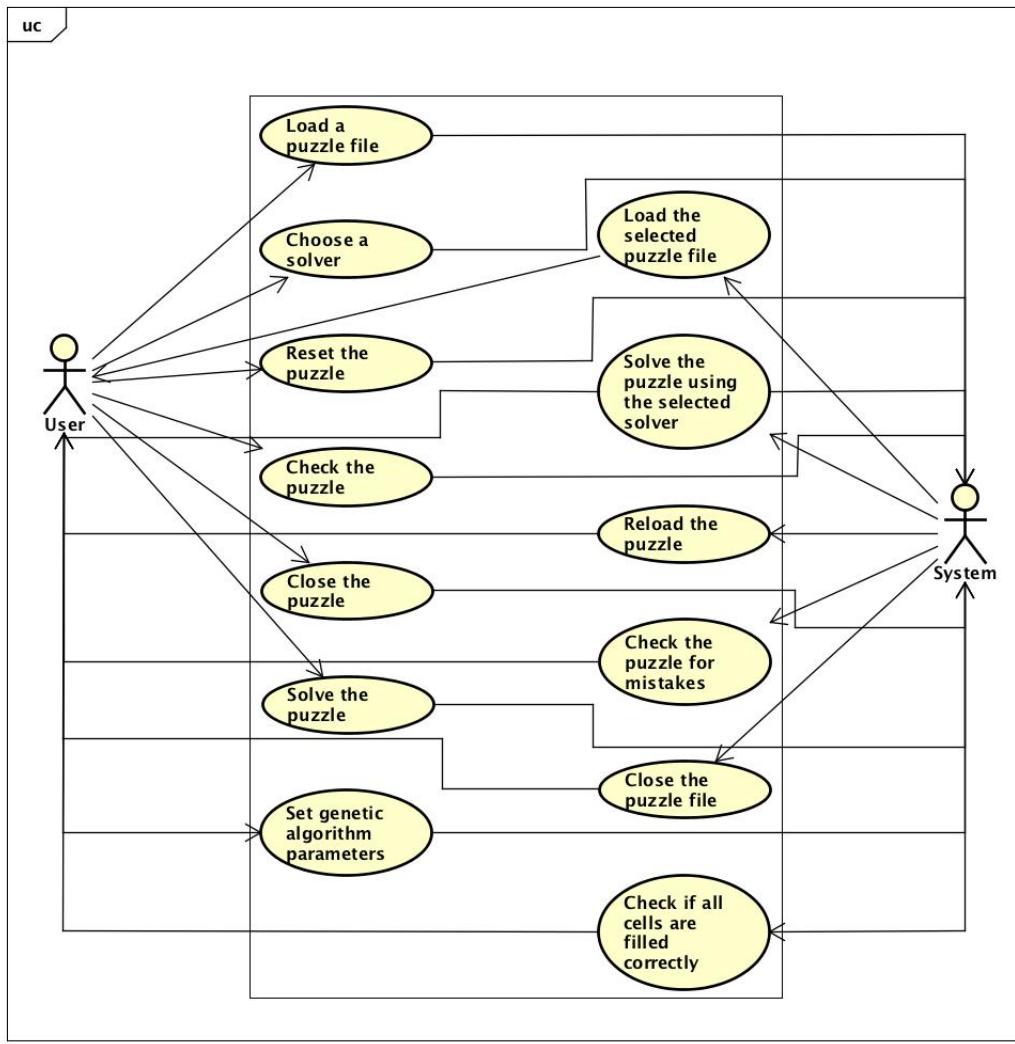
Diagram *use case* adalah diagram yang menggambarkan interaksi antara sistem (perangkat lunak) dengan pengguna. Berdasarkan analisis perangkat lunak yang telah dilakukan, maka pengguna dapat:

1. Membuka file masukan untuk memulai permainan.
2. Memilih salah satu dari dua *solver* yang disediakan untuk menyelesaikan permainan berdasarkan *file* yang sudah di-*load*.
3. Me-*reset* permainan untuk mengulang permainan berdasarkan *file* masukan yang sudah di-*load* dari awal.
4. Meminta perangkat lunak untuk memeriksa permainan jika ada masukan yang salah di dalam *grid*.
5. Menutup *file* masukan untuk mengakhiri permainan.
6. Menyelesaikan permainan dengan usahanya sendiri.
7. Mengatur nilai dari parameter-parameter untuk algoritma genetik.

Diagram *use case* untuk perangkat lunak permainan teka-teki Calcudoku dapat dilihat pada Gambar 3.49.

Berdasarkan diagram *use case* yang dapat dilihat pada Gambar 3.49, skenario-skenario yang dapat dilakukan oleh pengguna adalah:

1. Membuka file masukan. Penjelasan untuk skenario ini dapat dilihat pada Tabel 3.5.
2. Memilih salah satu dari dua *solver* yang disediakan. Penjelasan untuk skenario ini dapat dilihat pada Tabel 3.6
3. Me-*reset* permainan. Penjelasan untuk skenario ini dapat dilihat pada Tabel 3.7.
4. Meminta perangkat lunak untuk memeriksa permainan. Penjelasan untuk skenario ini dapat dilihat pada Tabel 3.8.
5. Menutup *file* masukan. Penjelasan untuk skenario ini dapat dilihat pada Tabel 3.9.
6. Menyelesaikan permainan dengan usahanya sendiri. Penjelasan untuk skenario ini dapat dilihat pada Tabel 3.10.
7. Mengatur nilai dari parameter-parameter untuk algoritma genetik. Penjelasan untuk skenario ini dapat dilihat pada Tabel 3.11.

Gambar 3.49: Diagram *use case* untuk perangkat lunak permainan teka-teki CalcudokuTabel 3.5: Skenario *me-load file*

Nama	Membuka file masukan
Aktor	Pengguna
Deskripsi	Memembuka file masukan untuk memulai permainan.
Kondisi Awal	Perangkat lunak belum membuka file masukan.
Kondisi Akhir	Perangkat lunak sudah membuka file masukan .
Skenario Utama	Pengguna masuk ke dalam menu "File", lalu memilih menu item "Load Puzzle File", lalu memilih file masukan yang akan dibuka, dan mengklik tombol "OK". Jika perangkat lunak sudah membuka file masukan, dan ingin membuka file masukan yang baru, akan keluar kotak dialog "Are you sure you want to load another puzzle file?", klik tombol "Yes" untuk membuka file masukan baru, atau klik tombol "No" untuk membatalkan.

Tabel 3.6: Skenario memilih salah satu dari dua *solver* yang disediakan

Nama	Memilih salah satu dari dua <i>solver</i> yang disediakan
Aktor	Pengguna
Deskripsi	Memilih salah satu dari dua <i>solver</i> yang disediakan untuk menyelesaikan permainan berdasarkan <i>file</i> yang sudah di-load.
Kondisi Awal	<i>Solver</i> belum menyelesaikan permainan.
Kondisi Akhir	<i>Solver</i> berhasil atau gagal dalam menyelesaikan permainan.
Skenario Utama	Pengguna masuk ke dalam menu " <i>Solve</i> ", lalu memilih salah satu dari dua <i>solver</i> yang disediakan. Pemain memilih menu item " <i>Backtracking</i> " untuk memilih <i>solver</i> dengan algoritma <i>backtracking</i> , atau menu item " <i>Hybrid Genetic</i> " untuk memilih <i>solver</i> dengan algoritma <i>hybrid genetic</i> .

Tabel 3.7: Skenario me-reset permainan

Nama	Me-reset permainan
Aktor	Pengguna
Deskripsi	Me-reset permainan untuk mengulang permainan berdasarkan <i>file</i> masukan yang sudah di-load dari awal.
Kondisi Awal	Permainan belum di-reset, sel-sel dalam <i>grid</i> mungkin masih berisi angka-angka.
Kondisi Akhir	Permainan sudah di-reset, semua sel-sel dalam <i>grid</i> sudah dalam keadaan kosong.
Skenario Utama	Pengguna masuk ke dalam menu " <i>File</i> ", lalu memilih menu item " <i>Reset Puzzle File</i> ". Akan keluar kotak dialog " <i>Are you sure you want to reset this puzzle?</i> ", klik tombol " <i>Yes</i> " untuk me-reset permainan, atau klik tombol " <i>No</i> " untuk membatalkan. Jika perangkat lunak belum me-load <i>file</i> masukan, maka akan keluar pesan <i>error</i> " <i>Puzzle file not loaded</i> ".

Tabel 3.8: Skenario meminta perangkat lunak untuk memeriksa permainan

Nama	Meminta perangkat lunak untuk memeriksa permainan
Aktor	Pengguna
Deskripsi	Meminta perangkat lunak untuk memeriksa permainan jika ada masukan yang salah di dalam <i>grid</i> .
Kondisi Awal	Permainan belum diperiksa oleh perangkat lunak.
Kondisi Akhir	Permainan sudah diperiksa oleh perangkat lunak. Pengguna akan diberitahu oleh perangkat lunak jika ada masukan yang salah di dalam <i>grid</i> , misalnya ada angka yang berulang dalam sebuah baris atau kolom, atau angka-angka dalam sebuah <i>cage</i> tidak mencapai angka tujuan yang ditentukan setelah dihitung dengan operasi matematika yang ditentukan.
Skenario Utama	Pengguna masuk ke dalam menu " <i>File</i> ", lalu memilih menu item " <i>Check Puzzle File</i> ". Jika perangkat lunak belum me-load <i>file</i> masukan, maka akan keluar pesan <i>error</i> " <i>Puzzle file not loaded</i> ".

Tabel 3.9: Skenario menutup *file* masukan

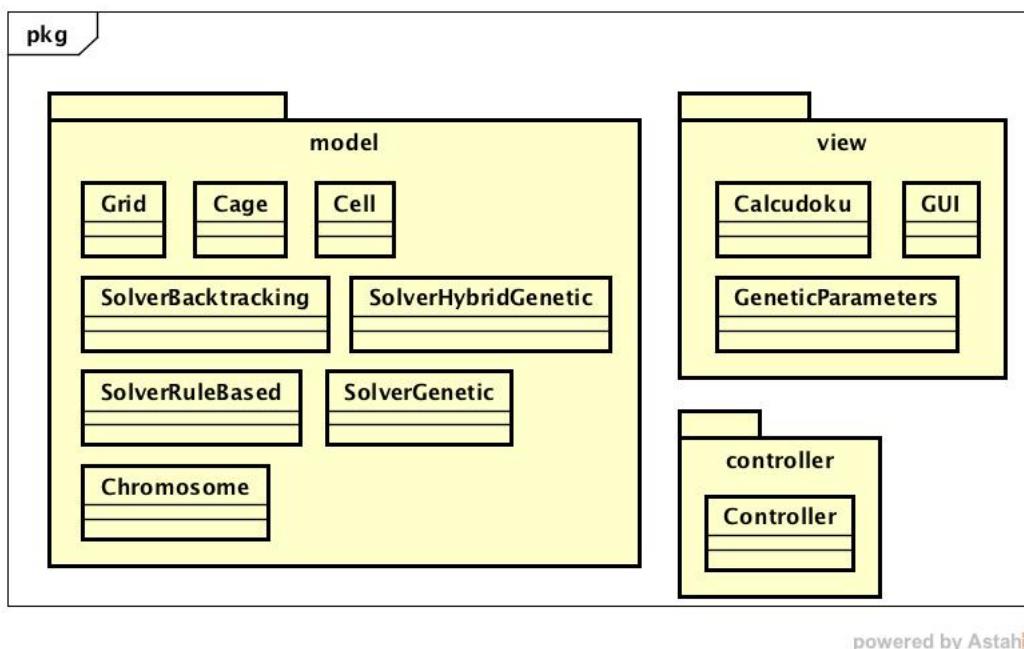
Nama	Menutup <i>file</i> masukan
Aktor	Pengguna
Deskripsi	Menutup <i>file</i> masukan untuk mengakhiri permainan.
Kondisi Awal	Perangkat lunak belum menutup <i>file</i> masukan.
Kondisi Akhir	Perangkat lunak sudah menutup <i>file</i> masukan.
Skenario Utama	Pengguna masuk ke dalam menu "File", lalu memilih menu item "Close Puzzle File". Jika perangkat lunak belum me-load <i>file</i> masukan, maka akan keluar pesan <i>error</i> "Puzzle file not loaded".

Tabel 3.10: Skenario menyelesaikan permainan dengan usahanya sendiri

Nama	Menyelesaikan permainan dengan usahanya sendiri.
Aktor	Pengguna
Deskripsi	Pemain menyelesaikan permainan dengan usahanya sendiri. Pemain mengisikan sel-sel dalam <i>grid</i> dengan angka 1 sampai <i>n</i> , dengan <i>n</i> merupakan ukuran dari <i>grid</i> . Perangkat lunak dapat secara otomatis memeriksa <i>grid</i> jika ada masukan yang salah di dalam <i>grid</i> , misalnya ada angka yang berulang dalam sebuah baris atau kolom, atau angka-angka dalam sebuah <i>cage</i> tidak mencapai angka tujuan yang ditentukan setelah dihitung dengan operasi matematika yang ditentukan.
Kondisi Awal	Semua sel-sel dalam <i>grid</i> dalam keadaan kosong.
Kondisi Akhir	Semua sel-sel dalam <i>grid</i> sudah terisi dengan angka-angka, dengan rincian tidak ada angka yang berulang dalam sebuah baris atau kolom, dan angka-angka dalam setiap <i>cage</i> mencapai angka tujuan yang ditentukan setelah dihitung dengan operasi matematika yang ditentukan.
Skenario Utama	Pemain mengisikan sel-sel dalam <i>grid</i> dengan angka 1 sampai <i>n</i> , dengan <i>n</i> merupakan ukuran dari <i>grid</i> .

Tabel 3.11: Skenario mengatur nilai dari parameter-parameter untuk algoritma genetik

Nama	Mengatur nilai dari parameter-parameter untuk algoritma genetik
Aktor	Pengguna
Deskripsi	Pemain mengatur atau mengubah nilai dari parameter-parameter untuk algoritma genetik dengan mengisi <i>form</i> yang telah disediakan. Pemain menekan tombol "OK" untuk mengatur atau mengubah nilai dari parameter-parameter untuk algoritma genetik.
Kondisi Awal	Parameter-parameter untuk algoritma genetik belum diatur, atau sudah diatur tetapi belum diubah.
Kondisi Akhir	Parameter-parameter untuk algoritma genetik sudah diatur jika belum diatur sebelumnya, atau sudah diubah jadi sudah diatur sebelumnya.
Skenario Utama	Pemain mengisikan <i>form</i> yang telah disediakan, lalu menekan tombol "OK".



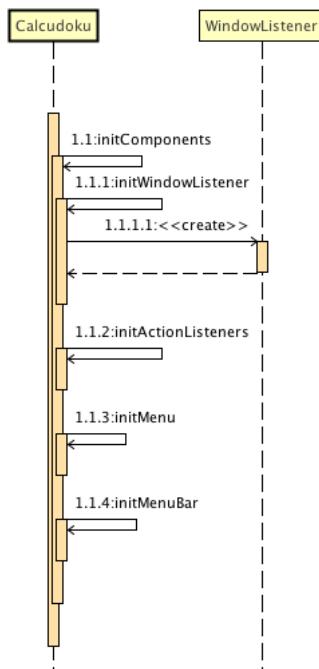
Gambar 3.50: Diagram kelas untuk perangkat lunak permainan teka-teki Calcudoku

¹ 3.3.2 Diagram Kelas

² Berdasarkan diagram *use case* yang telah dibuat, maka diagram kelas dapat dibuat. ³ Diagram kelas untuk perangkat lunak permainan teka-teki Calcudoku dapat dilihat pada ⁴ Gambar 3.50.

⁵ Berdasarkan diagram kelas tersebut, kelas-kelas yang digunakan dalam perangkat lunak ⁶ Calcudoku adalah:

- ⁷ 1. *Package* model, yaitu *package* yang berisi kelas-kelas yang merepresentasikan permainan teka-teki Calcudoku. *Package* ini terdiri dari 8 kelas, yaitu:
- ⁹ (a) Kelas Grid, yaitu kelas yang merepresentasikan sebuah *grid* dalam permainan Calcudoku.
- ¹¹ (b) Kelas Cell, yaitu kelas yang merepresentasikan sebuah sel dalam *grid*.
- ¹² (c) Kelas Cage, yaitu kelas yang merepresentasikan sebuah *cage* dalam *grid*.
- ¹³ (d) Kelas SolverBacktracking, yaitu kelas yang merepresentasikan *solver* untuk permainan Calcudoku dengan algoritma *backtracking*.
- ¹⁵ (e) Kelas SolverHybridGenetic, yaitu kelas yang merepresentasikan *solver* untuk permainan Calcudoku dengan algoritma *hybrid genetic*.
- ¹⁷ (f) Kelas SolverRuleBased, yaitu kelas yang merepresentasikan *solver* untuk permainan Calcudoku dengan algoritma *rule based*, bagian pertama dari algoritma *hybrid genetic*.
- ¹⁹ (g) Kelas SolverGenetic, yaitu kelas yang merepresentasikan *solver* untuk permainan Calcudoku dengan algoritma genetik, bagian kedua dari algoritma *hybrid genetic*.
- ²¹ (h) Kelas Chromosome, yaitu kelas yang merepresentasikan sebuah kromosom dalam algoritma genetik.



Gambar 3.51: Diagram *sequence* saat perangkat lunak dibuka

1 2. *Package view*, yaitu *package* yang merepresentasikan GUI untuk permainan Calcudoku. *Package* ini terdiri dari 2 kelas, yaitu:

- 3 (a) Kelas Calcudoku, yaitu kelas yang merepresentasikan *frame* untuk GUI permainan Calcudoku. Kelas ini berisi menu *bar*, dan panel yang merepresentasikan *grid* untuk permainan Calcudoku (kelas GUI).
- 4 (b) Kelas GUI, yaitu kelas yang merepresentasikan *grid* untuk permainan Calcudoku.
- 5 (c) Kelas GeneticParameters, yaitu kelas yang berisi *form* untuk mengatur nilai dari parameter-parameter untuk algoritma genetik.

6 3. *Package controller*, yaitu penghubung antara kelas-kelas yang ada di dalam *package* model dengan kelas-kelas yang ada di dalam *package view*. *Package* ini terdiri dari 1 kelas, yaitu kelas Controller. Kelas ini menghubungkan kelas-kelas yang ada di dalam *package* model dengan kelas-kelas yang ada di dalam *package view*.

7 13 Penjelasan tentang variabel-variabel dan *method-method* yang ada di dalam kelas-kelas
8 14 di atas akan dijelaskan di dalam bab Perancangan.

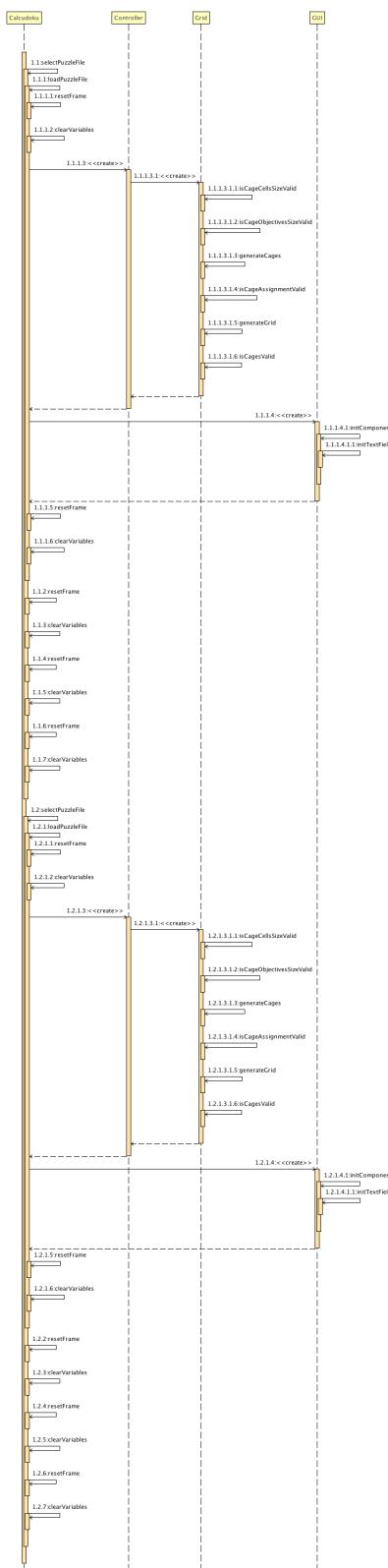
15 3.3.3 Diagram *Sequence*

16 Diagram *sequence* adalah diagram yang menggambarkan interaksi antar kelas dalam suatu
17 skenario.

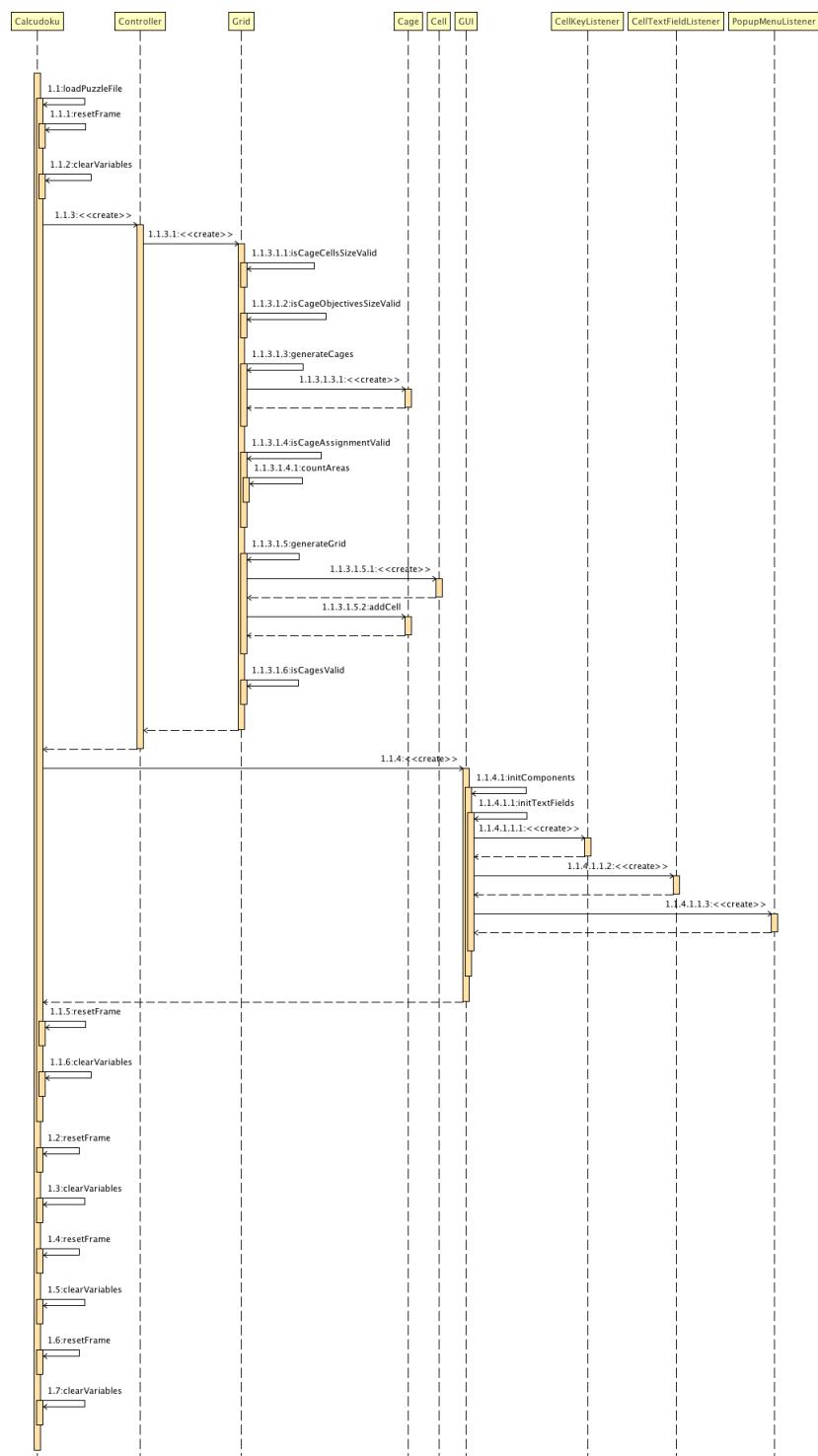
18 Gambar 3.51 menunjukkan diagram *sequence* saat perangkat lunak dibuka.

19 Gambar 3.52 menunjukkan diagram *sequence* saat menu item "Load Puzzle File" dalam
20 menu "File" dipilih.

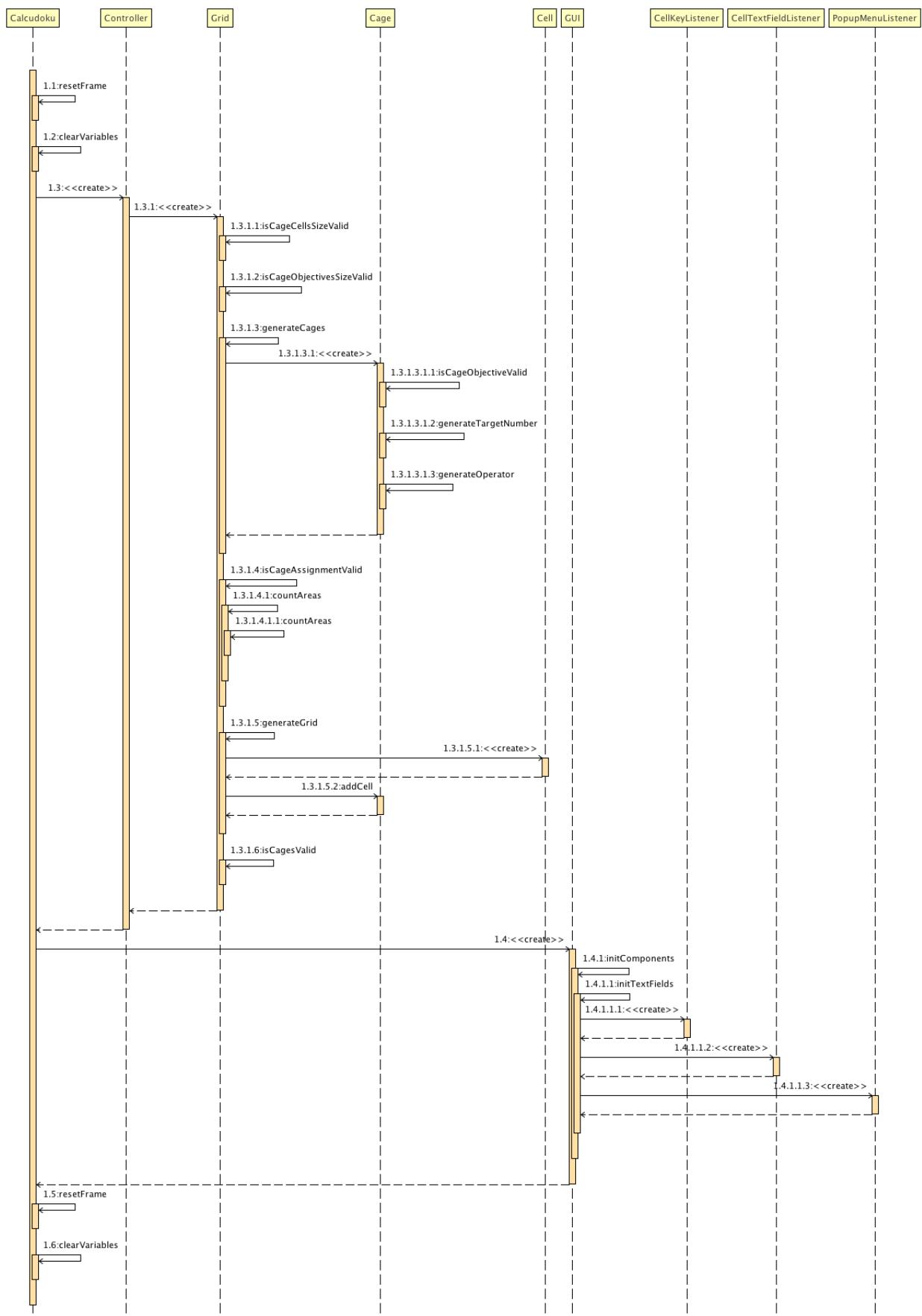
21 Gambar 3.53 menunjukkan diagram *sequence* saat file permainan yang ingin dibuka
22 dalam *file chooser* dipilih.



Gambar 3.52: Diagram *sequence* saat menu item "Load Puzzle File" dalam menu "File" dipilih

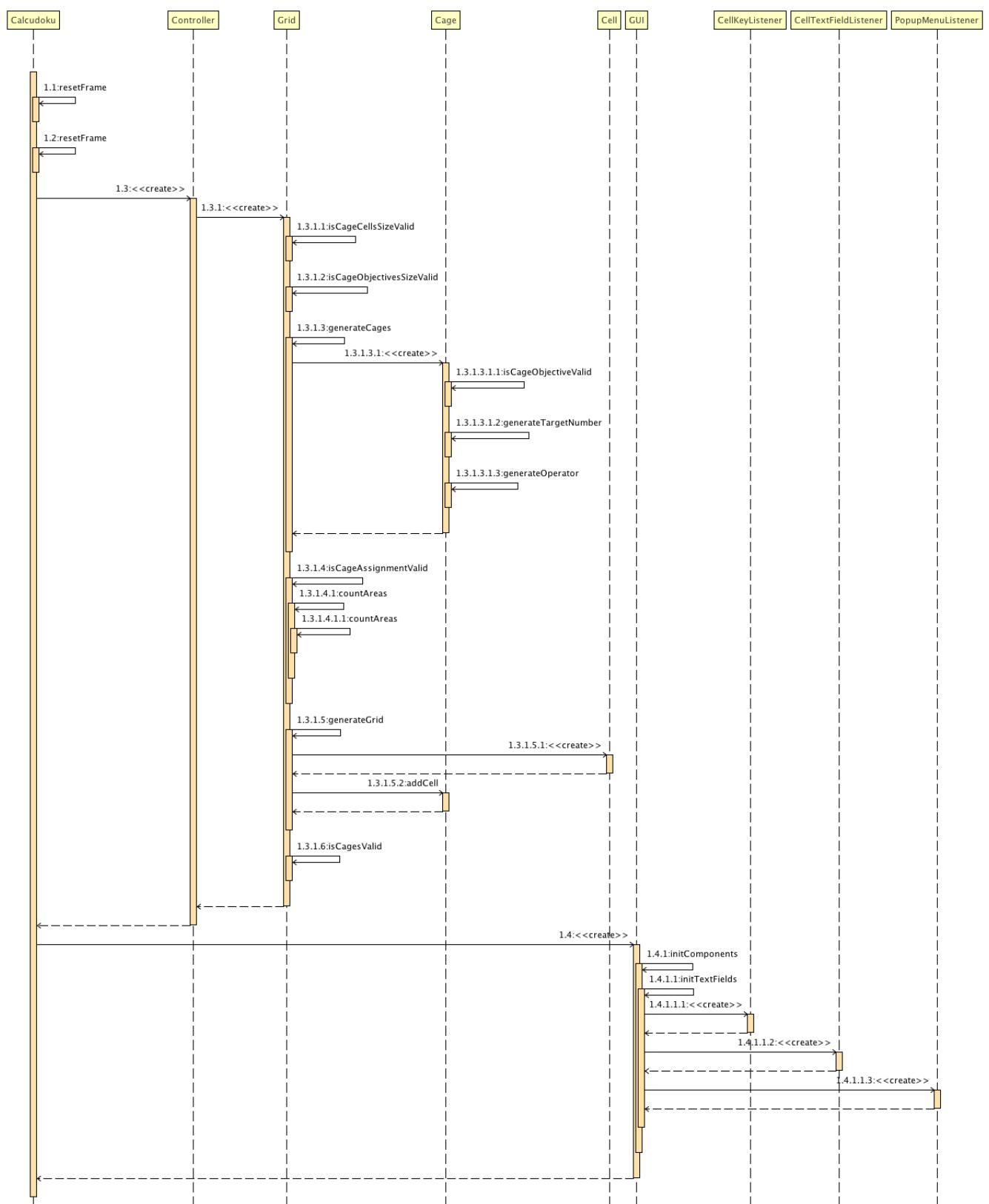


Gambar 3.53: Diagram *sequence* saat file yang ingin dibuka dalam file *chooser* dipilih

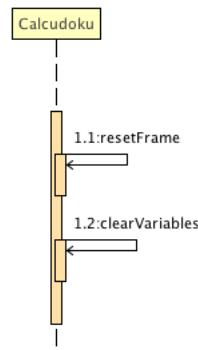


Gambar 3.54: Diagram *sequence* saat file permainan yang sudah dipilih dibuka

- ¹ Gambar 3.54 menunjukkan diagram *sequence* saat *file* permainan yang sudah dipilih dibuka.
- ³ Gambar 3.55 menunjukkan diagram *sequence* saat menu *item "Reset Puzzle"* dalam menu *"File"* dipilih.

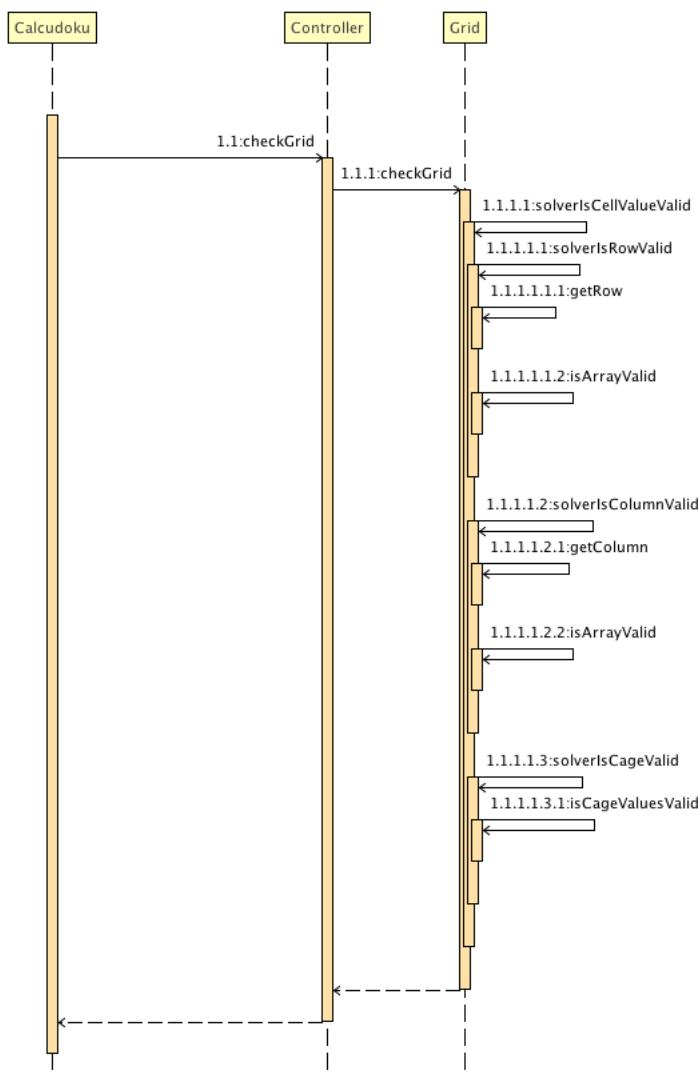


Gambar 3.55: Diagram *sequence* saat menu item "Reset Puzzle" dalam menu "File" dipilih

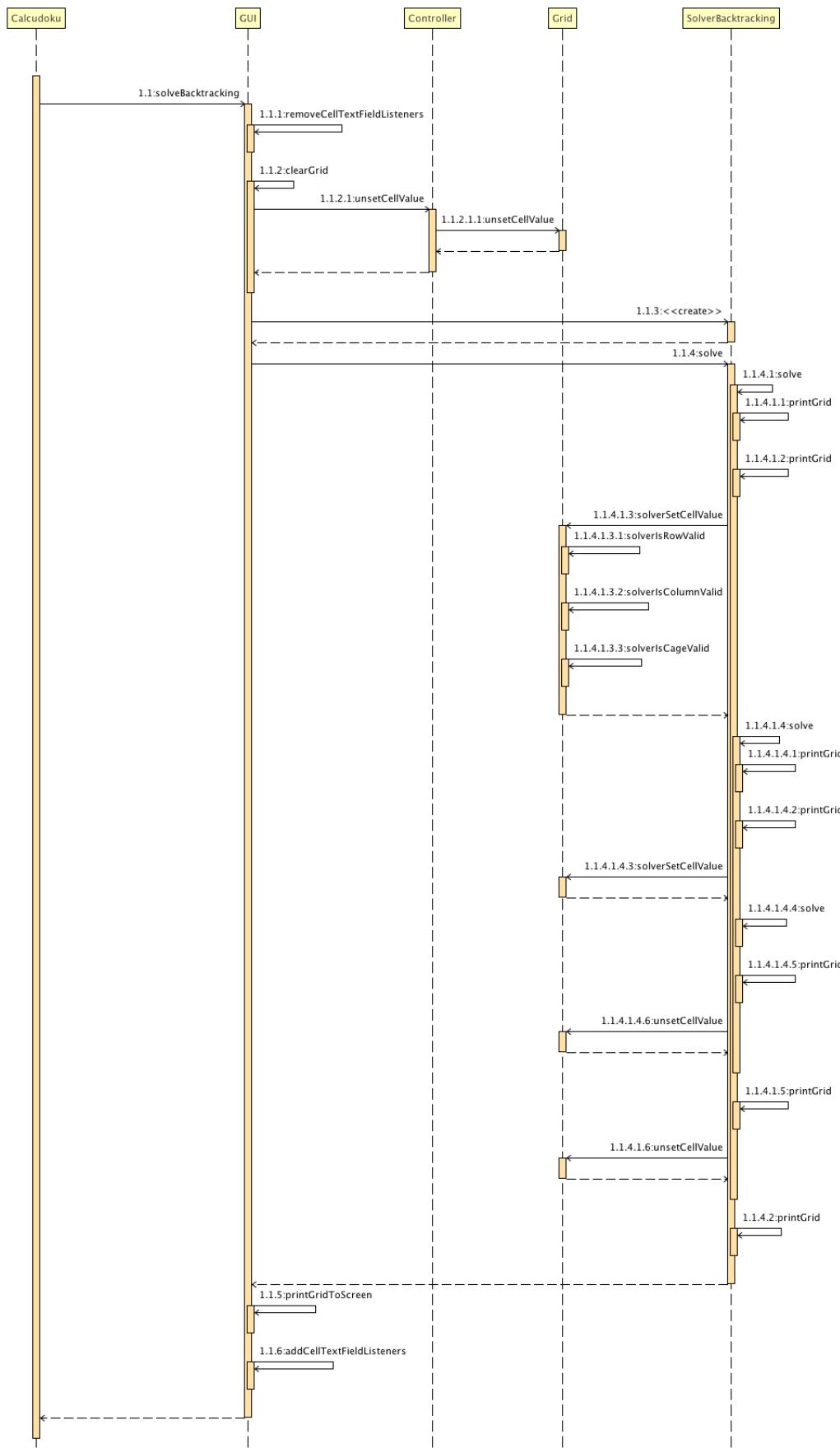


Gambar 3.56: Diagram *sequence* saat menu item "Close Puzzle File" dalam menu "File" dipilih

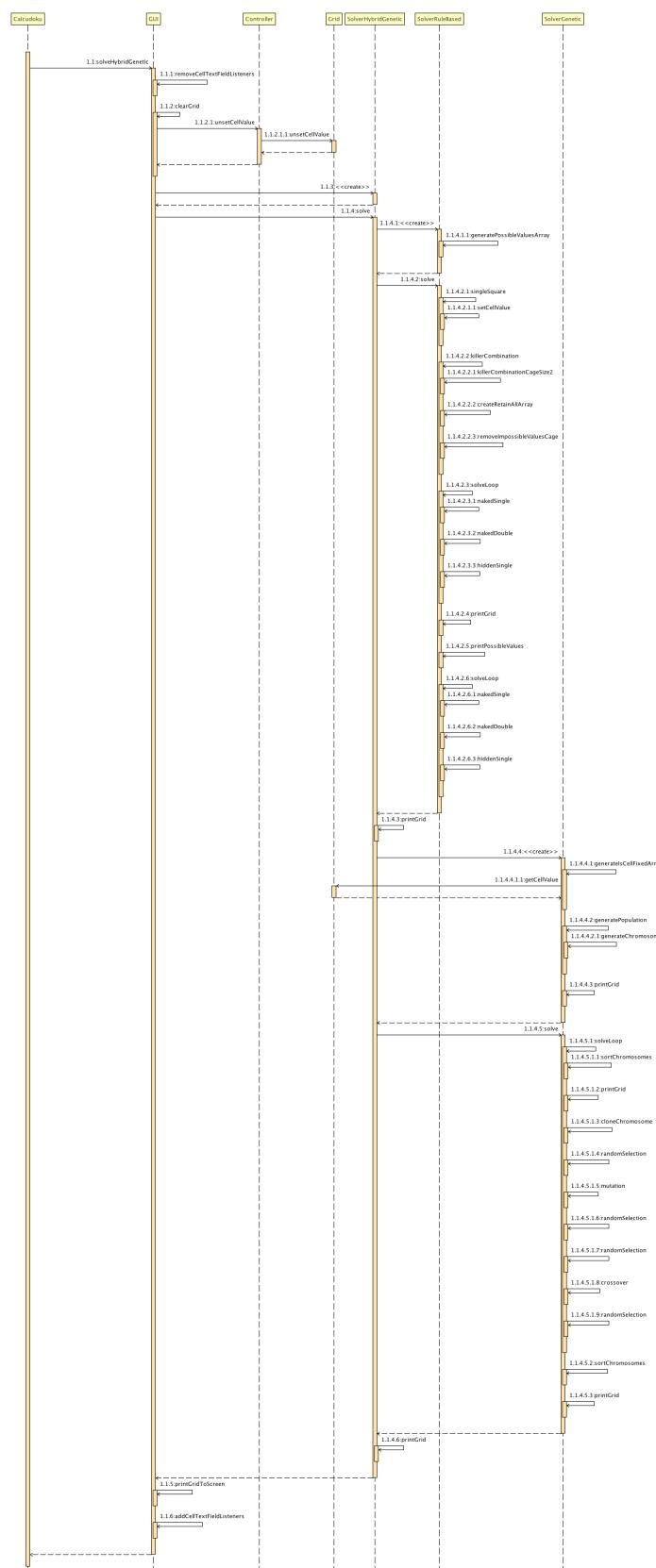
- 1 Gambar 3.56 menunjukkan diagram *sequence* saat menu item "Close Puzzle File" dalam menu "File" dipilih.
- 2 Gambar 3.57 menunjukkan diagram *sequence* saat menu item "Check Puzzle File" dalam menu "File" dipilih.
- 3 Gambar 3.58 menunjukkan diagram *sequence* saat menu item "Backtracking" dalam menu "Solve" dipilih.
- 4 Gambar 3.59 menunjukkan diagram *sequence* saat menu item "Hybrid Genetic" dalam menu "Solve" dipilih.



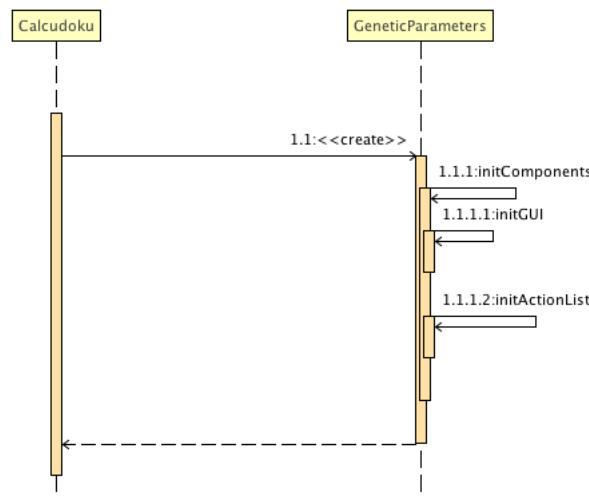
Gambar 3.57: Diagram *sequence* saat menu item "Check Puzzle File" dalam menu "File" dipilih



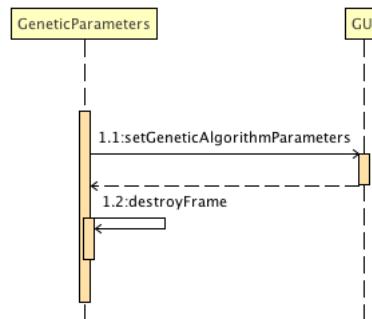
Gambar 3.58: Diagram *sequence* saat menu *item* "Backtracking" dalam menu "Solve" dipilih



Gambar 3.59: Diagram *sequence* saat menu item "Hybrid Genetic" dalam menu "Solve" dipilih

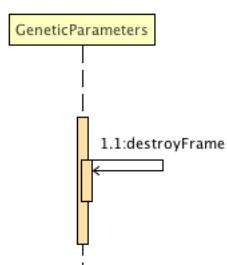


Gambar 3.60: Diagram *sequence* saat menu item "Set Genetic Algorithm Parameters" dalam menu "Solve" dipilih

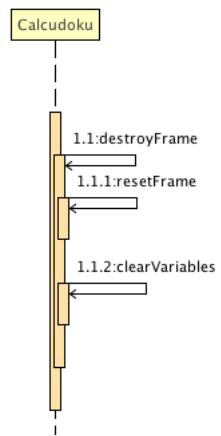


Gambar 3.61: Diagram *sequence* saat button "OK" dalam form "Set Genetic Algorithm Parameters" dalam menu "Solve" dipilih

- 1 Gambar 3.60 menunjukkan diagram *sequence* saat menu item "Set Genetic Algorithm Parameters" dalam menu "Solve" dipilih.
- 2 Gambar 3.61 menunjukkan diagram *sequence* saat button "OK" dalam form "Set Genetic Algorithm Parameters" dalam menu "Solve" dipilih.
- 3 Gambar 3.62 menunjukkan diagram *sequence* saat button "Cancel" dalam form "Set Genetic Algorithm Parameters" dalam menu "Solve" dipilih.
- 4 Gambar 3.63 menunjukkan diagram *sequence* saat perangkat lunak ditutup.



Gambar 3.62: Diagram *sequence* saat button "Cancel" dalam form "Set Genetic Algorithm Parameters" dalam menu "Solve" dipilih



Gambar 3.63: Diagram *sequence* saat perangkat lunak ditutup

¹ **BAB 4**

² **PERANCANGAN**

³ Bab ini membahas tentang perancangan perangkat lunak yang dibuat. Bab ini juga akan
⁴ membahas tentang perancangan masukan, perancangan keluaran, diagram kelas, diagram
⁵ *use case*, diagram aktivitas, dan diagram *sequence* untuk perangkat lunak tersebut.

⁶ **4.1 Perancangan Masukan**

⁷ Masukan untuk perangkat lunak permainan teka-teki Calcudoku ini berupa sebuah *file* teks,
⁸ seperti yang ditunjukkan pada Gambar 4.1.

⁹ Adapun rincian dari *file* teks masukan tersebut adalah sebagai berikut:

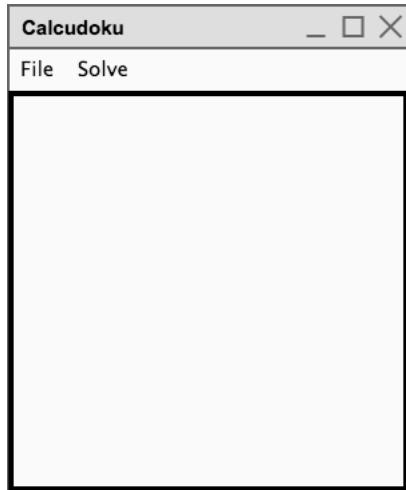
- ¹⁰ 1. Baris pertama berisi ukuran *grid* dan banyaknya *cage* dari teka-teki Calcudoku ter-
¹¹ sebut. Angka pertama adalah ukuran *grid*, dan angka kedua adalah banyaknya *cage*.
- ¹² 2. Baris kedua sampai ke baris $2 + (n - 1)$, dengan n adalah ukuran *grid*, berisi
¹³ matriks *cage assignment*. Matriks ini merepresentasikan posisi dari setiap *cage* dalam
¹⁴ *grid*. Setiap *cage* direpresentasikan dengan angka yang berbeda. Setiap *cage* dapat
¹⁵ mempunyai ukuran (jumlah sel yang terdapat dalam *cage*) yang bervariasi. Setiap sel
¹⁶ dalam sebuah *cage* harus berhubungan secara horizontal atau vertikal dengan sel lain
¹⁷ dalam *cage* yang sama.
- ¹⁸ 3. Baris $2 + n$ dan seterusnya berisi *cage objectives* untuk setiap *cage*. *Cage objectives*

4
9
1 2 3 3
1 4 4 5
6 7 7 5
8 8 9 9
7+
2=
2-
3-
2/
1=
6*
3+
7+

Gambar 4.1: Contoh *file* masukan.

4	2	3	1
3	4	1	2
1	3	2	4
2	1	4	3

Gambar 4.2: Contoh keluaran.



Gambar 4.3: Perancangan GUI sebelum membuka *file* permainan

berisikan angka tujuan dan operasi matematika yang telah ditentukan. Angka-angka dalam sebuah *cage* harus mencapai angka tujuan jika dihitung menggunakan operasi matematika yang telah ditentukan.

4.2 Perancangan Keluaran

Keluaran untuk perangkat lunak permainan teka-teki Calcudoku ini berupa sebuah matriks yang berisi solusi dari teka-teki Calcudoku yang sudah diselesaikan oleh program, seperti dapat dilihat pada Gambar 4.2.

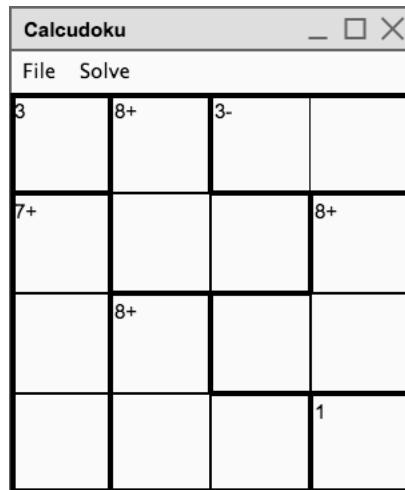
4.3 Perancangan Antarmuka

Antarmuka untuk perangkat lunak ini terdiri dari sebuah *frame* yang berisi sebuah menu *bar* dan GUI dari permainan teka-teki Calcudoku. GUI hanya akan ditampilkan jika perangkat lunak sudah membuka *file* permainan. Jika *file* permainan ditutup, maka GUI juga akan ditutup.

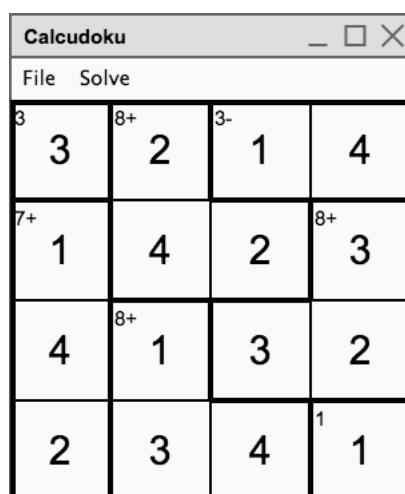
Gambar 4.3 menunjukkan perancangan GUI sebelum *file* permainan dibuka. Gambar 4.4 menunjukkan perancangan GUI sesudah *file* permainan dibuka. Gambar 4.5 menunjukkan perancangan GUI sesudah permainan berdasarkan *file* permainan yang dibuka diselesaikan.

Menu *bar* untuk perangkat lunak ini terdiri dari dua menu, yaitu:

1. *File*, yaitu menu yang berisi *item-item* menu yang terkait dengan *file* permainan.
2. *Solve*, yaitu menu yang berisi *item-item* menu yang terkait dengan *solver*.



Gambar 4.4: Perancangan GUI sesudah membuka file permainan



Gambar 4.5: Perancangan GUI sesudah permainan berdasarkan file permainan yang dibuka diselesaikan.

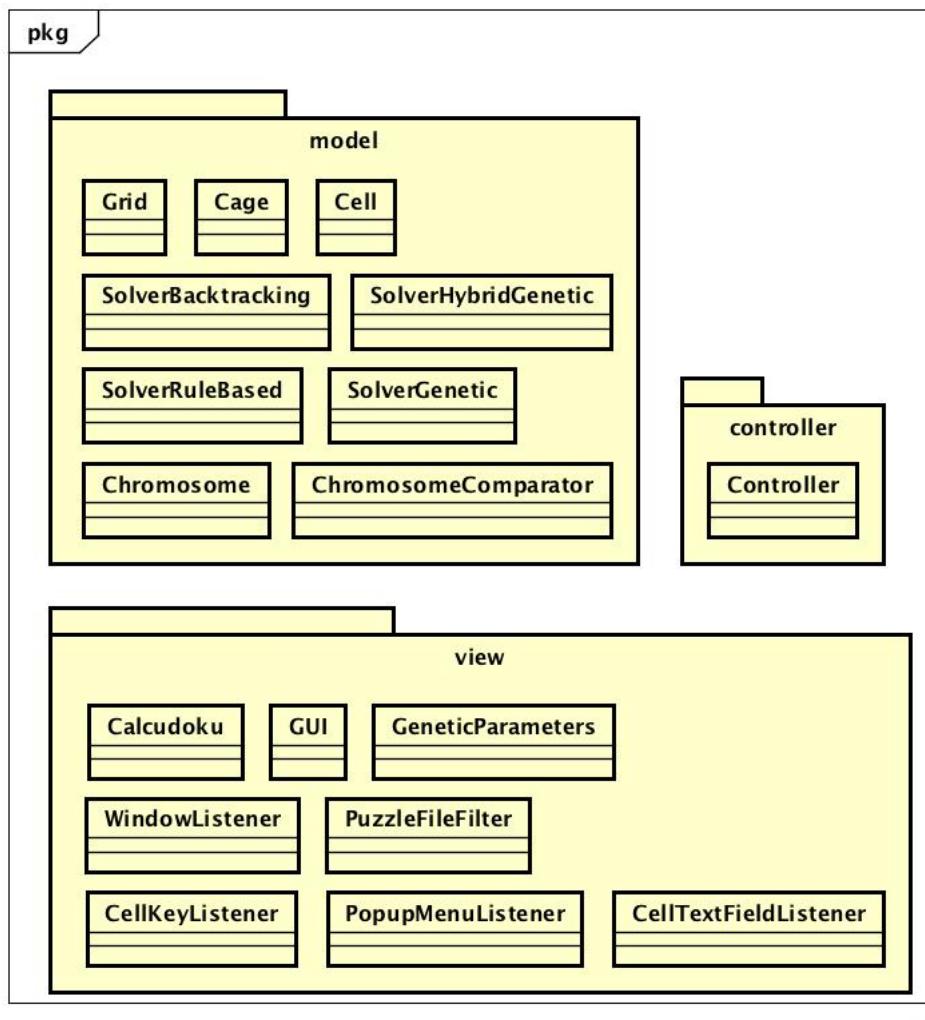
Gambar 4.6: Menu *File*Gambar 4.7: Menu *Solve*

- 1 Menu *File* mempunyai beberapa menu *item*, yaitu:
- 2 1. *Load Puzzle File*, yaitu menu *item* untuk membuka *file* permainan.
 - 3 2. *Reset Puzzle*, yaitu menu *item* untuk me-reset permainan.
 - 4 3. *Close Puzzle File*, yaitu menu *item* untuk menutup *file* permainan.
 - 5 4. *Check Puzzle*, yaitu menu *item* untuk memeriksa permainan jika ada masukan yang salah di dalam *grid*.
 - 6 5. *Exit*, yaitu menu *item* untuk menutup perangkat lunak.
- 7 Gambar 4.6 menunjukkan isi dari menu *File*.
- 8 Menu *Solve* mempunyai beberapa menu *item*, yaitu:
- 9 1. *Backtracking*, yaitu menu *item* untuk menyelesaikan permainan berdasarkan *file* permainan yang dibuka dengan algoritma *backtracking*.
 - 10 2. *Hybrid Genetic*, yaitu menu *item* untuk menyelesaikan permainan berdasarkan *file* permainan yang dibuka dengan algoritma *hybrid genetic*.
 - 11 3. *Set Genetic Algorithm Parameters*, yaitu menu *item* untuk mengatur nilai dari parameter-parameter untuk algoritma genetik.
- 12 Gambar 4.7 menunjukkan isi dari menu *Solve*.

17 4.4 Diagram Kelas

- 18 Perangkat lunak teka-teki Calcudoku ini terdiri dari beberapa kelas, yang dikelompokkan
19 dalam tiga package, yaitu:
- 20 1. Model, yaitu *engine* dari perangkat lunak ini. Package ini memiliki beberapa kelas,
21 yaitu:
 - 22 (a) Grid, yaitu kelas yang merepresentasikan *grid* dalam teka-teki Calcudoku.
 - 23 (b) Cell, yaitu kelas yang merepresentasikan sel dalam teka-teki Calcudoku.

- 1 (c) Cage, yaitu kelas yang merepresentasikan *cage* dalam teka-teki Calcudoku.
 - 2 (d) SolverBacktracking, yaitu kelas *solver* untuk teka-teki Calcudoku menggunakan algoritma backtracking.
 - 3 (e) SolverHybridGenetic, yaitu kelas *solver* untuk teka-teki Calcudoku menggunakan algoritma *hybrid genetic*. Algoritma ini akan mencoba menyelesaikan teka-teki Calcudoku menggunakan algoritma *rule based* terlebih dahulu. Algoritma genetik baru akan dijalankan jika algoritma *rule based* gagal dalam menyelesaikan teka-teki Calcudoku.
 - 4 (f) SolverRuleBased, yaitu kelas *solver* untuk teka-teki Calcudoku menggunakan algoritma *rule based*. Dalam algoritma *hybrid genetic*, algoritma akan mencoba menyelesaikan teka-teki Calcudoku menggunakan algoritma *rule based* terlebih dahulu.
 - 5 (g) SolverGenetic, yaitu kelas *solver* untuk teka-teki Calcudoku menggunakan algoritma genetik. Dalam algoritma *hybrid genetic*, algoritma genetik baru akan dijalankan jika algoritma *rule based* gagal dalam menyelesaikan teka-teki Calcudoku.
 - 6 (h) Chromosome, yaitu kelas yang merepresentasikan sebuah kromosom untuk algoritma genetik dalam solver *hybrid genetic*.
 - 7 (i) ChromosomeComparator, yaitu kelas pembanding *custom (custom comparator)* yang berfungsi untuk mengurutkan kromosom berdasarkan nilai kelayakkannya (*fitness value*).
- 22 2. View, yaitu GUI dari perangkat lunak ini. Package ini memiliki beberapa kelas, yaitu:
- 23 (a) Calcudoku, yaitu kelas *frame* yang berisi menu *bar* dan instansiasi kelas panel GUI.
 - 24 (b) WindowListener, yaitu kelas *listener* untuk kelas Calcudoku. *Listener* ini berfungsi untuk menambahkan pesan peringatan saat akan menutup perangkat lunak.
 - 25 (c) PuzzleFileFilter, yaitu kelas filter untuk *file chooser*. Filter ini membatasi agar *file chooser* hanya bisa membuka *file* teks.
 - 26 (d) GUI, yaitu kelas panel yang merepresentasikan GUI dari permainan teka-teki Calcudoku.
 - 27 (e) CellKeyListener, yaitu kelas *listener* untuk kelas GUI. *Listener* ini berfungsi untuk menggerakkan kursor dari sebuah sel ke sel di sebelahnya menggunakan tombol-tombol panah ke kiri, ke atas, ke bawah, dan ke kanan. Listener ini juga berfungsi untuk membatasi agar sel hanya bisa diisi oleh satu angka.
 - 28 (f) PopupMenuListener, yaitu kelas *listener* untuk kelas GUI. *Listener* ini berfungsi untuk mengisi sel dengan angka menggunakan menu *pop up*, sel akan diisi dengan angka yang dipilih.
 - 29 (g) CellTextFieldListener, yaitu kelas *listener* untuk kelas GUI. *Listener* ini berfungsi untuk mengisikan sel dalam kelas Grid dengan angka yang diisikan ke dalam sel dalam GUI.



powered by Astah

Gambar 4.8: Diagram kelas untuk perangkat lunak Calcudoku.

- 1 (h) GeneticParameters, yaitu kelas yang berisi *form* untuk mengatur nilai dari
2 parameter-parameter untuk algoritma genetik.
- 3 3. Controller, yaitu penghubung antara *package* model dan *package* view. Package ini
4 hanya berisi satu kelas, yaitu kelas Controller.

5 Diagram kelas untuk perangkat lunak ini dapat dilihat pada Gambar 4.8.

6 Berikut ini adalah rincian dari setiap kelas, dengan setiap atribut dan setiap *method*
7 yang dimilikinya.

8 4.4.1 Kelas Grid

9 Kelas Grid mempunyai beberapa atribut, yaitu:

- 10 1. size, yaitu ukuran dari matriks *grid*.
- 11 2. numberCages, yaitu banyaknya *cage* yang terdapat dalam *grid*.
- 12 3. cageCells, sebuah matriks *cage assignment*. Matriks ini merepresentasikan posisi
13 dari setiap *cage* dalam *grid*.

- 1 4. cageObjectives, yaitu sebuah *array* yang berisi *cage objectives* untuk setiap *cage*. *Cage objectives* berisikan angka tujuan dan operasi matematika yang telah ditentukan.
- 2 5. grid, yaitu representasi dari *grid* dalam teka-teki Calcudoku. *grid* adalah sebuah matriks yang berisi sel-sel. Matriks ini berukuran $n \times n$.
- 3 6. cages, yaitu representasi dari sebuah *cage* dalam sebuah *grid*.

4 Kelas Grid mempunyai beberapa *method*, yaitu:

- 5 1. Grid(Integer size, Integer numberOfCages, Integer[][] cageCells, String[] cageObjectives), yaitu konstruktor dari kelas ini. Konstruktor ini menerima masukan berupa ukuran dari matriks *grid*, banyaknya *cage* yang terdapat dalam *grid*, matriks *cage assignment*, dan array *cage objectives*.
- 6 2. countAreas(Integer[][] array), yaitu *method* pembungkus dari *method* countAreas(int[][] array, boolean[][] checked). *Method* ini menerima masukan berupa array *cage assignment* untuk sebuah *cage*, dan menghasilkan keluaran berupa jumlah area dari *cage* tersebut.
- 7 3. countAreas(Integer[][] array, boolean[][] checked), yaitu *method* yang menghitung jumlah area dari sebuah *cage* secara rekursif dengan menggunakan algoritma *flood fill*. *Method* ini menerima masukan berupa array *cage assignment* untuk sebuah *cage* dan sebuah array *checked* yang berfungsi untuk menandai sel-sel yang sudah pernah dikunjungi atau belum, dan menghasilkan keluaran berupa jumlah area dari *cage* tersebut.
- 8 4. floodFill(int i, int j, Integer[][] array, boolean[][] checked), yaitu implementasi dari algoritma *flood fill* untuk menghitung jumlah area dari sebuah *cage*. *Method* ini menerima masukan berupa posisi baris dan kolom dari sebuah sel, array *cage assignment* untuk sebuah *cage* dan sebuah array *checked* yang berfungsi untuk menandai sel-sel yang sudah pernah dikunjungi atau belum.
- 9 5. isCageCellsSizeValid(Integer[][] cageCells), yaitu *method* yang memeriksa apakah ukuran matriks *cage assignment* valid atau tidak. *Method* ini menerima masukan berupa matriks *cage assignment*, dan menghasilkan keluaran apakah matriks tersebut *valid* atau tidak. Matriks tersebut *valid* jika ukuran barisnya dan kolomnya sama dengan variabel *size*.
- 10 6. isCageObjectivesSizeValid(String[] cageObjectives), yaitu *method* yang memeriksa apakah ukuran matriks *cage objectives* valid atau tidak. *Method* ini menerima masukan berupa *array cage objectives*, dan menghasilkan keluaran apakah *array* tersebut *valid* atau tidak. *Array* tersebut *valid* jika ukuran dari *array* tersebut sama dengan variabel *numberOfCages*.
- 11 7. isCageAssignmentValid(Integer[][] array), yaitu *method* yang memeriksa apakah *cage assignment* untuk sebuah *cage* *valid* atau tidak. *Method* ini menerima masukan berupa matriks *cage assignment* untuk sebuah *cage* dan menghasilkan keluaran apakah matriks tersebut *valid* atau tidak. Matriks tersebut *valid* jika jumlah area dari *cage* tersebut adalah satu.

- 1 8. isCagesValid(Cage[] cages), yaitu *method* yang memeriksa apakah setiap *cage* yang
2 ada di dalam *grid valid* atau tidak. *Method* ini menerima masukan berupa *array cage*,
3 dan menghasilkan keluaran apakah *array* tersebut *valid* atau tidak. *Array* tersebut
4 *valid* jika setiap *cage* dengan operator = hanya berukuran satu sel, setiap *cage* dengan
5 operator + atau × berukuran minimal dua sel, dan setiap *cage* dengan operator - atau
6 ÷ berukuran tepat dua sel.
- 7 9. generateCages(Cage[] cages), yaitu *method* yang membangkitkan *cage-cage* dalam se-
8 buah *grid*. *Method* ini menerima masukan berupa sebuah array *Cage* yang kosong.
- 9 10. generateGrid(Cell[][] grid, Cage[] cages), yaitu *method* yang membangkitkan *grid* dan
10 *cage assignment* dari *grid* tersebut.. *Method* ini menerima masukan berupa sebuah
11 matriks sel yang kosong dan sebuah array *cage* yang kosong.
- 12 11. getRow(int rowNumber), yaitu *method* untuk mendapatkan isi dari sebuah baris yang
13 diminta. *Method* ini menerima masukan berupa nomor baris yang diminta dan meng-
14 hasilkan keluaran berupa isi baris yang diminta.
- 15 12. getColumn(int columnNumber), yaitu *method* untuk mendapatkan isi dari sebuah ko-
16 lom yang diminta. *Method* ini menerima masukan berupa nomor kolom yang diminta
17 dan menghasilkan keluaran berupa isi kolom yang diminta dalam bentuk *ArrayList*.
- 18 13. getCageValues(int cageNumber), yaitu *method* untuk mendapatkan isi dari sebuah
19 *cage* yang diminta. *Method* ini menerima masukan berupa nomor *cage* yang diminta
20 dan menghasilkan keluaran berupa isi *cage* yang diminta dalam bentuk *ArrayList*.
- 21 14. isArrayValid(ArrayList<Integer> array), yaitu *method* untuk memeriksa apakah se-
22 buah *array valid* atau tidak. *Method* ini menerima masukan berupa *array* yang akan
23 diperiksa dan menghasilkan keluaran apakah *array* tersebut *valid* atau tidak. *Array*
24 tersebut *valid* jika tidak ada angka yang berulang dalam *array* tersebut.
- 25 15. isRowValid(int row), yaitu *method* untuk memeriksa apakah sebuah baris *valid* atau
26 tidak. *Method* ini menerima masukan berupa nomor baris yang diminta dan mengha-
27 silkan keluaran apakah baris yang diminta tersebut *valid* atau tidak. Baris tersebut
28 *valid* jika tidak ada angka yang berulang dalam baris tersebut.
- 29 16. solverIsRowValid(int column), yaitu *method* yang sama dengan *isRowValid*, tetapi
30 *method* ini hanya untuk dipanggil oleh *solver*.
- 31 17. isColumnValid(int column), yaitu *method* untuk memeriksa apakah sebuah kolom *va-*
32 *lid* atau tidak. *Method* ini menerima masukan berupa nomor kolom yang diminta dan
33 menghasilkan keluaran apakah kolom yang diminta tersebut *valid* atau tidak. Kolom
34 tersebut *valid* jika tidak ada angka yang berulang dalam kolom tersebut.
- 35 18. solverIsColumnValid(int column), yaitu *method* yang sama dengan *isColumnValid*,
36 tetapi *method* ini hanya untuk dipanggil oleh *solver*.
- 37 19. isCageValuesValid(int cageNumber), yaitu *method* untuk memeriksa apakah nilai dari
38 setiap sel yang berada dalam sebuah *cage valid* atau tidak. *Method* ini menerima

- 1 masukan berupa nomor *cage* yang diminta dan menghasilkan keluaran apakah nilai
2 dari setiap sel yang berada dalam *cage* yang diminta tersebut *valid* atau tidak. *Cage*
3 tersebut *valid* jika nilai dari setiap sel yang berada di dalam *cage* tersebut mencapai
4 angka tujuan yang telah ditentukan jika dihitung menggunakan operator yang telah
5 ditentukan.
- 6 20. `isCageValid(int row, int column)`, yaitu *method* untuk memeriksa apakah sebuah *cage*
7 *valid* atau tidak. *Method* ini menerima masukan berupa nomor baris dan nomor kolom
8 dari sebuah sel yang diminta dan menghasilkan keluaran apakah *cage* yang berisi sel
9 tersebut *valid* atau tidak. *Cage* tersebut *valid* jika angka-angka dalam *cage* tersebut
10 mencapai angka tujuan yang telah ditentukan jika dihitung menggunakan operator
11 yang telah ditentukan.
- 12 21. `solverIsCageValid(int column)`, yaitu *method* yang sama dengan `isCageValid`, tetapi
13 *method* ini hanya untuk dipanggil oleh solver.
- 14 22. `isCellValueValid(int row, int column)`, yaitu *method* untuk memeriksa apakah nilai
15 dari sel tersebut *valid* atau tidak. *Method* ini menerima masukan berupa nomor baris
16 dan nomor kolom dari sel yang akan diperiksa dan menghasilkan keluaran apakah nilai
17 dari sel tersebut *valid* atau tidak. Nilai dari sebuah sel *valid* jika nilai dari sel tersebut
18 tidak berulang dalam baris dan kolom tempat sel tersebut berada, dan angka-angka
19 dari *cage* yang berisi sel tersebut mencapai angka tujuan yang telah ditentukan jika
20 dihitung menggunakan operator yang telah ditentukan.
- 21 23. `solverIsCellValueValid(int column)`, yaitu *method* yang sama dengan `isCellValueValid`,
22 tetapi *method* ini hanya untuk dipanggil oleh solver.
- 23 24. `setCellValue(int row, int column, Integer value)`, yaitu *method* untuk mengisi sebuah
24 sel dengan nilai yang telah ditentukan. *Method* ini menerima masukan berupa
25 nomor baris dan nomor kolom dari sel yang akan diisi dan nilai dari sel tersebut,
26 dan menghasilkan keluaran apakah nilai dari sel tersebut *valid* atau tidak. Nilai dari
27 sebuah sel *valid* jika nilai dari sel tersebut tidak berulang dalam baris dan kolom tem-
28 pat sel tersebut berada, dan angka-angka dari *cage* yang berisi sel tersebut mencapai
29 angka tujuan yang telah ditentukan jika dihitung menggunakan operator yang telah
30 ditentukan.
- 31 25. `solverSetCellValue(int row, int column, Integer value)`, yaitu *method* yang sama de-
32 ngan `setCellValue`, tetapi *method* ini hanya untuk dipanggil oleh solver.
- 33 26. `unsetCellValue(int row, int column)`, yaitu *method* untuk menghapus isi dari sebuah
34 sel. *Method* ini menerima masukan berupa nomor baris dan nomor kolom dari sel
35 yang akan dihapus isinya.
- 36 27. `isWin()`, yaitu *method* yang memeriksa apakah semua sel sudah diisi dengan nilai
37 yang *valid* atau tidak. *Method* ini menghasilkan keluaran apakah semua sel sudah
38 diisi dengan nilai yang *valid* atau tidak. *Method* ini menghasilkan *null* jika ada sel
39 yang belum diisi.

- 1 28. `checkGrid()`, yaitu *method* yang memeriksa apakah ada sel yang berisi nilai yang
2 tidak *valid* atau tidak. *Method* ini akan menghasilkan apakah ada sel yang berisi nilai
3 yang tidak *valid* atau tidak. Nilai dari sebuah sel *valid* jika nilai dari sel tersebut
4 tidak berulang dalam baris dan kolom tempat sel tersebut berada, dan angka-angka
5 dari *cage* yang berisi sel tersebut mencapai angka tujuan yang telah ditentukan jika
6 dihitung menggunakan operator yang telah ditentukan. *Method* ini juga memeriksa
7 apakah ada sel yang kosong atau tidak.
- 8 29. `isFilled()`, yaitu *method* yang memeriksa apakah semua sel sudah diisi atau tidak.
9 *Method* ini menghasilkan keluaran apakah semua sel sudah diisi atau tidak.
- 10 30. `getCellValue(int row, int column)`, yaitu *method* untuk mendapatkan isi dari sebuah
11 sel. *Method* ini menerima masukan berupa nomor baris dan nomor kolom dari sel
12 yang diminta dan menghasilkan keluaran berupa isi dari sel yang diminta tersebut.
- 13 31. `getSize()`, yaitu *method* untuk mendapatkan ukuran dari *grid*. *Method* ini mengha-
14 silkan keluaran berupa ukuran dari *grid*.
- 15 32. `getNumberOfCages()`, yaitu *method* untuk mendapatkan jumlah *cage* yang ada di
16 dalam *grid*. *Method* ini menghasilkan keluaran berupa jumlah *cage* yang ada di dalam
17 *grid*.
- 18 33. `getCageCells()`, yaitu *method* untuk mendapatkan matriks *cage assignment* dari *grid*.
19 *Method* ini menghasilkan keluaran berupa matriks *cage assignment* dari *grid*.
- 20 34. `getCageObjectives()`, yaitu *method* untuk mendapatkan *cage objectives* dari setiap
21 *cage* dalam *grid*. *Method* ini menghasilkan keluaran berupa sebuah array yang berisi
22 *cage objectives* dari setiap *cage* dalam *grid*.
- 23 35. `getGridContents()`, yaitu *method* untuk mendapatkan nilai dari setiap sel *grid*. *Method*
24 ini menghasilkan keluaran berupa sebuah matriks yang berisi nilai dari setiap sel *grid*.
- 25 36. `getCages()`, yaitu *method* untuk mendapatkan semua *cage* dalam *grid*. *Method* ini
26 menghasilkan keluaran berupa array yang berisi semua *cage* dalam *grid*.
- 27 37. `getGame()`, yaitu *method* untuk mendapatkan instansiasi saat ini dari kelas *Grid*.
28 *Method* ini menghasilkan keluaran berupa instansiasi saat ini dari kelas *Grid*.

29 Diagram kelas Grid dapat dilihat pada Gambar 4.9.

30 4.4.2 Kelas Cage

31 Kelas Cage mempunyai beberapa atribut, yaitu:

- 32 1. `objective`, yaitu angka tujuan dan operator yang ditentukan untuk *cage* tersebut.
- 33 2. `targetNumber`, yaitu angka tujuan dari *cage* tersebut.
- 34 3. `operator`, yaitu operator yang ditentukan untuk *cage* tersebut.
- 35 4. `cells`, yaitu sebuah array yang berisi sel-sel yang merupakan anggota dari *cage* tersebut.

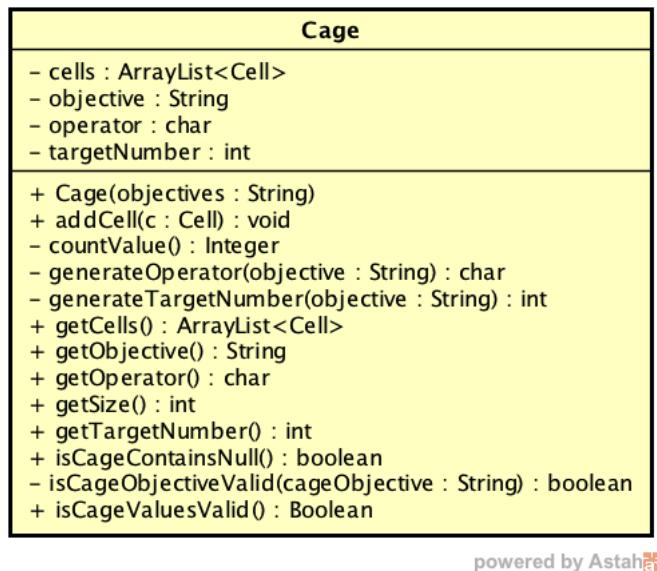


powered by Astah

Gambar 4.9: Diagram kelas Grid.

1 Kelas Cage mempunyai *method-method* berikut:

- 2 1. Cage(String objectives), yaitu konstruktor dari kelas ini. Konstruktor ini menerima
3 masukan berupa *cage objectives* dari *cage* tersebut.
- 4 2. isCageObjectiveValid(String cageObjective), yaitu *method* yang memeriksa apakah
5 *cage objective* dari *cage* tersebut *valid* atau tidak. *Method* ini menerima masukan
6 berupa String yang berisi *cage objective* dan menghasilkan keluaran apakah String
7 tersebut valid atau tidak. *Cage objective valid* jika isi dari *cage objective* tersebut
8 adalah satu angka tujuan dari *cage* tersebut dan diikuti oleh satu operator yang telah
9 ditentukan untuk *cage* tersebut.
- 10 3. generateTargetNumber(String objective), yaitu *method* yang membangkitkan angka
11 tujuan dari sebuah *cage* dari *cage objective* yang diberikan. *Method* ini menerima
12 masukan berupa String yang berisi *cage objective* dari sebuah *cage* dan menghasilkan
13 keluaran berupa angka tujuan dari *cage* tersebut.
- 14 4. generateOperator(String objective), yaitu *method* yang membangkitkan operator yang
15 telah ditentukan untuk sebuah *cage* dari *cage objective* yang diberikan. *Method* ini
16 menerima masukan berupa String yang berisi *cage objective* dari sebuah *cage* dan
17 menghasilkan keluaran berupa operator yang telah ditentukan untuk *cage* tersebut.
- 18 5. addCell(Cell c), yaitu *method* untuk menambahkan sebuah sel kedalam sebuah *cage*.
19 *Method* ini menerima masukan berupa sel yang akan dimasukkan ke dalam *cage*.
- 20 6. isCageContainsNull(), yaitu *method* yang memeriksa apakah sebuah *cage* mempunyai
21 sel yang belum diisi. *Method* ini menghasilkan keluaran apakah *cage* tersebut mem-
22 punyai sel yang belum terisi.
- 23 7. isCageValuesValid(), yaitu *method* yang memeriksa apakah angka-angka dalam sebuah
24 *cage* mencapai angka tujuan dari *cage* tersebut jika dihitung menggunakan operator
25 yang telah ditentukan untuk *cage* tersebut. *Method* ini menghasilkan keluaran apa-
26 kah angka-angka dalam *cage* tersebut mencapai angka tujuan dari *cage* tersebut jika
27 dihitung menggunakan operator yang telah ditentukan untuk *cage* tersebut. *Method*
28 ini menghasilkan *null* jika ada sel di dalam *cage* yang belum diisi.
- 29 8. countValue(), yaitu *method* yang menghitung angka-angka di dalam sebuah *cage*
30 menggunakan operator yang telah ditentukan untuk *cage* tersebut. *Method* ini meng-
31 hasilkan keluaran hasil perhitungan dari angka-angka di dalam sebuah *cage* menggu-
32 nakan operator yang telah ditentukan untuk *cage* tersebut. *Method* ini menghasilkan
33 *null* jika ada sel di dalam *cage* yang belum diisi.
- 34 9. getTargetNumber(), yaitu *method* untuk mendapatkan angka tujuan dari sebuah *cage*.
35 *Method* ini menghasilkan keluaran berupa angka tujuan dari *cage* tersebut.
- 36 10. getOperator(), yaitu *method* untuk mendapatkan operator yang telah ditentukan
37 untuk sebuah *cage*. *Method* ini menghasilkan keluaran berupa operator yang telah di-
38 tentukan untuk *cage* tersebut.



powered by Astah

Gambar 4.10: Diagram kelas Cage.

11. `getCells()`, yaitu *method* untuk mendapatkan sel-sel anggota sebuah *cage*. *Method* ini menghasilkan keluaran sebuah `ArrayList` yang berisi sel-sel anggota *cage* tersebut.
12. `getSize()`, yaitu *method* untuk mendapatkan jumlah dari sel-sel anggota sebuah *cage*. *Method* ini menghasilkan keluaran berupa jumlah dari sel-sel anggota *cage* tersebut.

5 Diagram kelas Cage dapat dilihat pada Gambar 4.10.

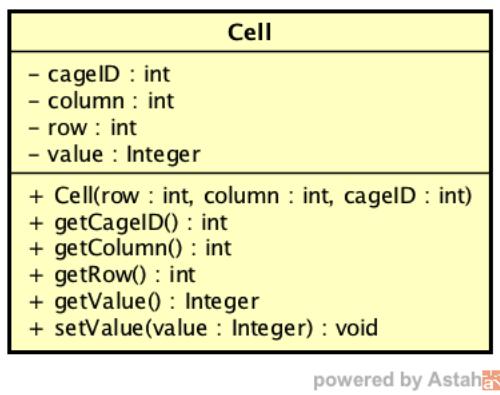
6 4.4.3 Kelas Cell

7 Kelas Cell mempunyai beberapa atribut, yaitu:

- 8 1. `row`, yaitu posisi baris dari sel tersebut.
- 9 2. `column`, yaitu posisi kolom dari sel tersebut.
- 10 3. `cageID`, yaitu nomor *cage* yang berisi sel tersebut.
- 11 4. `value`, yaitu nilai dari sel tersebut.

12 Kelas Cell mempunyai beberapa *method*, yaitu:

- 13 1. `Cell(int row, int column, int cageID)`, yaitu konstruktor dari kelas ini. Konstruktor ini menerima masukan berupa nomor baris, dan nomor kolom dari sel tersebut, dan nomor *cage* yang berisi sel tersebut.
- 14 2. `setValue(Integer value)`, yaitu *method* untuk mengisi sebuah sel tersebut dengan nilai yang telah ditentukan. *Method* ini menerima masukan berupa nilai yang akan diisikan ke dalam sel tersebut.
- 15 3. `getValue()`, yaitu *method* untuk mendapatkan nilai dari sel tersebut. *Method* ini menghasilkan keluaran berupa nilai dari sel tersebut.



Gambar 4.11: Diagram kelas Cell.

- 1 4. `getRow()`, yaitu *method* untuk mendapatkan nomor baris dari sebuah sel. *Method* ini menghasilkan keluaran berupa nomor baris dari sel tersebut.
- 2
- 3 5. `getColumn()`, yaitu *method* untuk mendapatkan nomor kolom dari sebuah sel. *Method* ini menghasilkan keluaran berupa nomor kolom dari sel tersebut.
- 4
- 5 6. `getCageID()`, yaitu *method* untuk mendapatkan nomor *cage* yang berisi sebuah sel. *Method* ini menghasilkan keluaran berupa nomor *cage* yang berisi sel tersebut.
- 6
- 7 Diagram kelas Cell dapat dilihat pada Gambar 4.11.

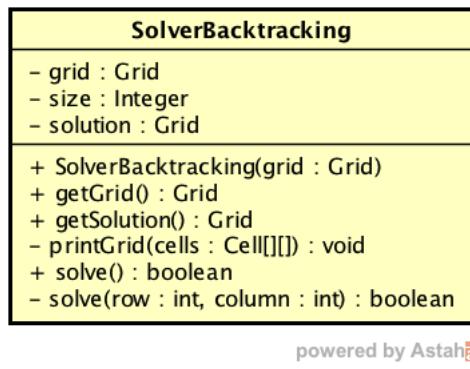
8 4.4.4 Kelas SolverBacktracking

- 9 Kelas SolverBacktracking mempunyai beberapa atribut, yaitu:

- 10 1. `grid`, yaitu *grid* yang akan diselesaikan oleh *solver* dengan algoritma *backtracking*.
- 11 2. `size`, yaitu ukuran dari *grid* yang akan diselesaikan oleh *solver* dengan algoritma *backtracking*.
- 12
- 13 3. `solution`, yaitu *grid* yang sudah diselesaikan oleh *solver* dengan algoritma *backtracking*.

- 14 Kelas SolverBacktracking mempunyai beberapa *method*, yaitu:

- 15 1. `SolverBacktracking(Grid grid)`, yaitu konstruktor dari kelas ini. Konstruktor ini menerima masukan berupa *grid* yang akan diselesaikan oleh *solver* dengan algoritma *backtracking*.
- 16
- 17
- 18 2. `solve()`, yaitu *method* pembungkus dari *method* `solve(int row, int column)`. *Method* ini menghasilkan keluaran apakah *solver* berhasil menyelesaikan teka-teki Calcudoku atau tidak. *Solver* bekerja mulai dari sel pada sudut kiri atas, lalu bergerak ke kanan sampai ke sel yang paling kanan, lalu bergerak ke baris berikutnya sampai ke baris yang paling bawah, selesai pada sel pada sudut kanan bawah.
- 19
- 20
- 21
- 22
- 23 3. `solve(int row, int column)`, yaitu *method* yang mencoba untuk menyelesaikan teka-teki Calcudoku menggunakan algoritma *backtracking*. *Method* ini menerima masukan berupa nomor baris dan nomor kolom yang akan diisi oleh *solver* dan menghasilkan
- 24
- 25



Gambar 4.12: Diagram kelas SolverBacktracking.

1 keluaran apakah nilai yang diisi oleh *solver valid* atau tidak. *Solver* akan mulai
 2 mengisi sel dari angka 1. Jika berhasil, maka *solver* akan maju ke sel berikutnya.
 3 Jika gagal, maka *solver* akan mencoba kemungkinan angka berikutnya. Jika semua
 4 kemungkinan angka gagal, maka *solver* akan mundur ke sel sebelumnya dan mencoba
 5 kemungkinan angka berikutnya.

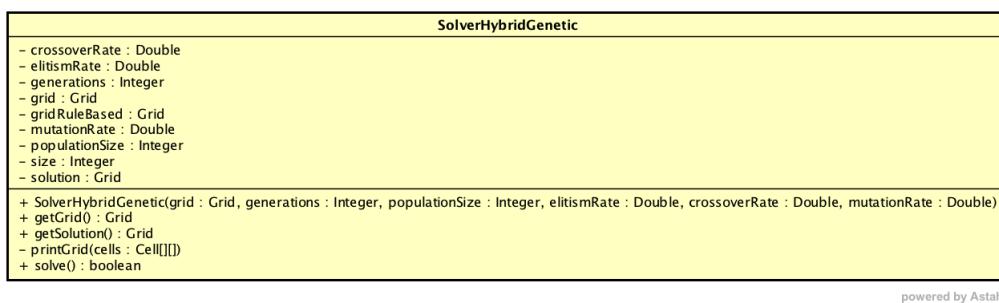
- 6 4. `getGrid()`, yaitu *method* untuk mendapatkan *grid*. *Method* ini menghasilkan keluaran
 7 berupa *grid*.
- 8 5. `getSolution()`, yaitu *method* untuk mendapatkan solusi dari *grid* yang sudah diselesa-
 9 ikan oleh *solver*. *Method* ini menghasilkan keluaran berupa solusi dari *grid* tersebut.
- 10 6. `printGrid()`, yaitu *method* untuk mencetak isi *grid* ke layar. *Method* ini menerima
 11 masukan berupa *grid* yang akan dicetak isinya ke layar.

12 Diagram kelas SolverBacktracking dapat dilihat pada Gambar 4.12.

13 4.4.5 Kelas SolverHybridGenetic

14 Kelas SolverHybridGenetic mempunyai beberapa atribut, yaitu:

- 15 1. *grid*, yaitu *grid* yang akan diselesaikan oleh *solver* dengan algoritma *hybrid genetic*.
- 16 2. *gridRuleBased*, yaitu *grid* yang telah diselesaikan oleh algoritma *rule based*.
- 17 3. *size*, yaitu ukuran dari *grid* yang akan diselesaikan oleh *solver* dengan algoritma *hybrid*
 18 *genetic*.
- 19 4. *solution*, yaitu *grid* yang sudah diselesaikan oleh *solver* dengan algoritma *hybrid ge-*
 20 *netic*.
- 21 5. *generations*, yaitu jumlah generasi maksimum yang akan dibangkitkan oleh algoritma
 22 genetik.
- 23 6. *populationSize*, yaitu jumlah kromosom yang akan dibangkitkan dalam sebuah gene-
 24 rasi.
- 25 7. *elitismRate*, yaitu parameter tingkat *elitism* dalam algoritma genetik..



Gambar 4.13: Diagram kelas SolverHybridGenetic.

- 1 8. crossoverRate, yaitu parameter tingkat kawin silang dalam algoritma genetik.
- 2 9. mutationRate, yaitu parameter tingkat mutasi dalam algoritma genetik.
- 3 Kelas SolverHybridGenetic mempunyai beberapa *method*, yaitu:
 - 4 1. SolverHybridGenetic(Grid grid, Integer generations, Integer populationSize, Double elitismRate, Double crossoverRate, Double mutationRate), yaitu konstruktor dari kelas ini. Konstruktor ini menerima masukan berupa *grid* yang akan diselesaikan oleh *solver* dengan algoritma *hybrid genetic*, dan parameter-parameter algoritma genetik, yaitu jumlah generasi, jumlah populasi dalam sebuah generasi, tingkat *elitism*, tingkat kawin silang, dan tingkat mutasi.
 - 5 2. getGrid(), yaitu *method* untuk mendapatkan *grid* yang akan diselesaikan oleh *solver* dengan algoritma *hybrid genetic*.
 - 6 3. getSolution(), yaitu *method* untuk mendapatkan *grid* yang sudah diselesaikan oleh *solver* dengan algoritma *hybrid genetic*.
 - 7 4. solve(), yaitu *method* yang mencoba untuk menyelesaikan teka-teki Calcudoku menggunakan algoritma *hybrid genetic*. *Method* ini akan memanggil solver algoritma *rule based*. Jika algoritma *rule based* gagal dalam menyelesaikan teka-teki Calcudoku, maka *method* akan memanggil solver algoritma genetik. *Method* ini menghasilkan keluaran apakah *solver* berhasil menyelesaikan teka-teki Calcudoku atau tidak.
 - 8 5. printGrid(), yaitu *method* untuk mencetak isi *grid* ke layar. *Method* ini menerima masukan berupa *grid* yang akan dicetak isinya ke layar.
- 9 Diagram kelas SolverHybridGenetic dapat dilihat pada Gambar 4.13.

22 4.4.6 Kelas SolverRuleBased

- 23 Kelas SolverRuleBased mempunyai beberapa atribut, yaitu:

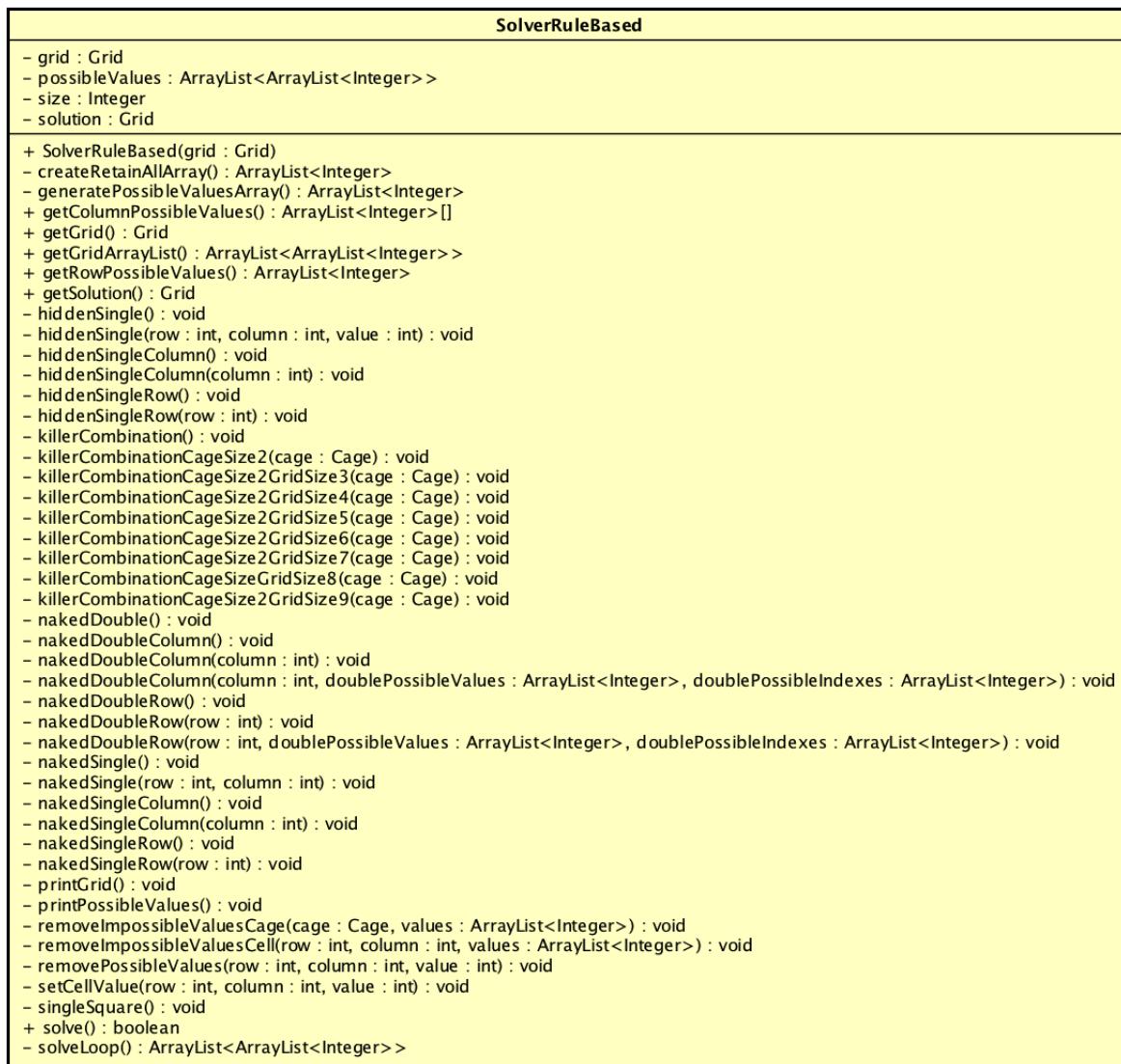
- 24 1. grid, yaitu *grid* yang akan diselesaikan oleh *solver* dengan algoritma *rule based*.
- 25 2. size, yaitu ukuran dari *grid* yang akan diselesaikan oleh *solver* dengan algoritma *rule based*.
- 26 3. solution, yaitu *grid* yang sudah diselesaikan oleh *solver* dengan algoritma *rule based*.

- 1 4. `possibleValues`, yaitu sebuah *array* yang menampung semua kemungkinan angka yang mungkin untuk setiap sel yang ada di dalam *grid*
- 2
- 3 Kelas `SolverRuleBased` mempunyai *method-method* berikut:
 - 4 1. `SolverRuleBased(Grid grid)`, yaitu konstruktor dari kelas ini. Konstruktor ini menerima masukan berupa *grid* yang akan diselesaikan oleh *solver* dengan algoritma *rule based*, dan parameter-parameter algoritma genetik, yaitu jumlah generasi, jumlah populasi dalam sebuah generasi, tingkat *elitism*, tingkat kawin silang, dan tingkat mutasi.
 - 5
 - 6
 - 7
 - 8
 - 9 2. `generatePossibleValuesArray()`, yaitu *method* yang membangkitkan kemungkinan angka-angka yang mungkin untuk setiap sel yang ada di dalam *grid*. Angka-angka yang mungkin adalah dari 1 sampai ke ukuran dari *grid*. *Method* ini menghasilkan keluaran berupa array yang menampung semua kemungkinan angka yang mungkin untuk setiap sel yang ada di dalam *grid*.
 - 10
 - 11
 - 12
 - 13
 - 14 3. `solve()`, yaitu *method* yang mencoba untuk menyelesaikan teka-teki Calcudoku menggunakan algoritma *rule based*. *Method* ini menghasilkan keluaran apakah *solver* berhasil menyelesaikan teka-teki Calcudoku atau tidak. *Solver* akan mencoba menyelesaikan teka-teki Calcudoku menggunakan aturan-aturan logika, misalnya *single square rule*, *killer combination rule*, *naked subset rule*, *hidden subset rule*, dan *evil twin rule*. Aturan *single square* dan *killer combination* hanya dipakai sekali, dan dilakukan oleh *method* ini, sedangkan aturan *naked subset*, *hidden subset*, dan *evil twin* dapat dipakai berkali-kali, dan dilakukan oleh *method* `solveLoop()`.
 - 15
 - 16
 - 17
 - 18
 - 19
 - 20
 - 21
 - 22 4. `solveLoop()`, yaitu *method* yang mengaplikasikan aturan *naked subset*, *hidden subset*, dan *evil twin* kepada *grid* yang sedang diselesaikan oleh algoritma *rule based*. *Method* ini menerima masukan berupa *array* kemungkinan angka yang mungkin untuk setiap sel yang ada di dalam *grid*. *Method* ini akan diulang sampai *method* ini tidak bisa mengisi sel-sel yang ada di dalam *grid*.
 - 23
 - 24
 - 25
 - 26
 - 27 5. `getRowPossibleValues(int rowNumber)`, yaitu *method* untuk mendapatkan kemungkinan angka-angka yang mungkin dari sel-sel yang ada di dalam baris yang diminta. *Method* ini menerima masukan berupa nomor baris yang diminta dan menghasilkan keluaran berupa kemungkinan angka-angka yang mungkin dari sel-sel yang ada di dalam baris yang diminta.
 - 28
 - 29
 - 30
 - 31
 - 32 6. `getColumnPossibleValues(int rowNumber)`, yaitu *method* untuk mendapatkan kemungkinan angka-angka yang mungkin dari sel-sel yang ada di dalam kolom yang diminta. *Method* ini menerima masukan berupa nomor kolom yang diminta dan menghasilkan keluaran berupa kemungkinan angka-angka yang mungkin dari sel-sel yang ada di dalam kolom yang diminta.
 - 33
 - 34
 - 35
 - 36
 - 37 7. `singleSquare()`, yaitu *method* yang mengaplikasikan aturan *single square* kepada *grid* yang sedang diselesaikan oleh algoritma *rule based*.
 - 38

- 1 8. killerCombination(), yaitu *method* yang mengaplikasikan aturan *killer combination* kepada *grid* yang sedang diselesaikan oleh algoritma *rule based*. Dalam perangkat lunak ini aturan ini dibatasi hanya untuk *cage* yang berukuran dua sel.
- 2
- 3
- 4 9. killerCombinationCageSize2(), yaitu *method* yang mengaplikasikan aturan *killer combination* kepada setiap *cage* yang berukuran 2 di dalam *grid* yang sedang diselesaikan oleh algoritma *rule based*.
- 5
- 6
- 7 10. killerCombinationCageSize2GridSize3(), yaitu *method* yang mengaplikasikan aturan *killer combination* kepada setiap *cage* yang berukuran 2 sel di dalam *grid* yang berukuran 3×3 yang sedang diselesaikan oleh algoritma *rule based*.
- 8
- 9
- 10 11. killerCombinationCageSize2GridSize4(), yaitu *method* yang mengaplikasikan aturan *killer combination* kepada setiap *cage* yang berukuran 2 sel di dalam *grid* yang berukuran 4×4 yang sedang diselesaikan oleh algoritma *rule based*.
- 11
- 12 12. killerCombinationCageSize2GridSize5(), yaitu *method* yang mengaplikasikan aturan *killer combination* kepada setiap *cage* yang berukuran 2 sel di dalam *grid* yang berukuran 5×5 yang sedang diselesaikan oleh algoritma *rule based*.
- 13
- 14 13. killerCombinationCageSize2GridSize6(), yaitu *method* yang mengaplikasikan aturan *killer combination* kepada setiap *cage* yang berukuran 2 sel di dalam *grid* yang berukuran 6×6 yang sedang diselesaikan oleh algoritma *rule based*.
- 15
- 16 14. killerCombinationCageSize2GridSize7(), yaitu *method* yang mengaplikasikan aturan *killer combination* kepada setiap *cage* yang berukuran 2 sel di dalam *grid* yang berukuran 7×7 yang sedang diselesaikan oleh algoritma *rule based*.
- 17
- 18 15. killerCombinationCageSize2GridSize8(), yaitu *method* yang mengaplikasikan aturan *killer combination* kepada setiap *cage* yang berukuran 2 sel di dalam *grid* yang berukuran 8×8 yang sedang diselesaikan oleh algoritma *rule based*.
- 19
- 20 16. killerCombinationCageSize2GridSize9(), yaitu *method* yang mengaplikasikan aturan *killer combination* kepada setiap *cage* yang berukuran 2 sel di dalam *grid* yang berukuran 9×9 yang sedang diselesaikan oleh algoritma *rule based*.
- 21
- 22 17. nakedSingle(), yaitu *method* yang mengaplikasikan aturan *naked single* kepada *grid* yang sedang diselesaikan oleh algoritma *rule based*. Aturan ini diaplikasikan untuk baris (*nakedSingleRow()*) dan untuk kolom (*nakedSingleColumn()*).
- 23
- 24 18. nakedSingleRow(), yaitu *method* yang mengaplikasikan aturan *naked single* kepada baris-baris yang ada di dalam *grid* yang sedang diselesaikan oleh algoritma *rule based*.
- 25
- 26 19. nakedSingleRow(int row), yaitu *method* yang mengaplikasikan aturan *naked single* kepada sebuah baris yang ada di dalam *grid* yang sedang diselesaikan oleh algoritma *rule based*. *Method* ini menerima masukan berupa nomor baris yang akan diaplikasikan dengan aturan *naked single*.
- 27
- 28 20. nakedSingleColumn(), yaitu *method* yang mengaplikasikan aturan *naked single* kepada kolom-kolom yang ada di dalam *grid* yang sedang diselesaikan oleh algoritma *rule based*.
- 29

- 1 21. nakedSingleColumn(int column), yaitu *method* yang mengaplikasikan aturan *naked single* kepada sebuah kolom yang ada di dalam *grid* yang sedang diselesaikan oleh algoritma *rule based*. *Method* ini menerima masukan berupa nomor kolom yang akan diaplikasikan dengan aturan *naked single*.
- 5 22. nakedSingle(int row, int column), yaitu *method* yang mengaplikasikan aturan *naked single* kepada sebuah sel yang sedang diselesaikan oleh algoritma *rule based*. *Method* ini menerima masukan berupa nomor baris dan nomor kolom dari sel yang akan diaplikasikan dengan aturan *naked single*.
- 9 23. nakedDouble(), yaitu *method* yang mengaplikasikan aturan *naked double* kepada *grid* yang sedang diselesaikan oleh algoritma *rule based*. Aturan ini diaplikasikan untuk baris (*nakedSingleDouble()*) dan untuk kolom (*nakedSingleDouble()*).
- 12 24. nakedDoubleRow(), yaitu *method* yang mengaplikasikan aturan *naked double* kepada baris-baris yang ada di dalam *grid* yang sedang diselesaikan oleh algoritma *rule based*.
- 14 25. nakedDoubleRow(int row), yaitu *method* yang mengaplikasikan aturan *naked double* kepada sebuah baris yang ada di dalam *grid* yang sedang diselesaikan oleh algoritma *rule based*. *Method* ini menerima masukan berupa nomor baris yang akan diaplikasikan dengan aturan *naked double*.
- 18 26. nakedDoubleRow(int row, ArrayList<Integer> doublePossibleValues, ArrayList<Integer> doublePossibleIndexes), yaitu *method* yang mengaplikasikan aturan *naked double* kepada baris-baris yang ada di dalam *grid* yang sedang diselesaikan oleh algoritma *rule based*. *Method* ini menyimpan nilai-nilai yang disimpan pada *doublePossibleValues* pada kolom-kolom yang nomor kolomnya disimpan pada *array doublePossibleIndexes* dan menghapus nilai-nilai tersebut dari kolom-kolom lainnya. *Method* ini menerima masukan berupa nomor baris yang akan diaplikasikan dengan aturan *naked double*, sebuah *array* yang berisi nilai-nilai, dan sebuah *array* yang berisi nomor-nomor kolom.
- 26 27. nakedDoubleColumn(), yaitu *method* yang mengaplikasikan aturan *naked double* kepada kolom-kolom yang ada di dalam *grid* yang sedang diselesaikan oleh algoritma *rule based*.
- 29 28. nakedDoubleColumn(int column), yaitu *method* yang mengaplikasikan aturan *naked double* kepada sebuah kolom yang ada di dalam *grid* yang sedang diselesaikan oleh algoritma *rule based*. *Method* ini menerima masukan berupa nomor kolom yang akan diaplikasikan dengan aturan *naked double*.
- 33 29. nakedDoubleColumn(int column, ArrayList<Integer> doublePossibleValues, ArrayList<Integer> doublePossibleIndexes), yaitu *method* yang mengaplikasikan aturan *naked double* kepada kolom-kolom yang ada di dalam *grid* yang sedang diselesaikan oleh algoritma *rule based*. *Method* ini menyimpan nilai-nilai yang disimpan pada *doublePossibleValues* pada baris-baris yang nomor barisnya disimpan pada *array doublePossibleIndexes* dan menghapus nilai-nilai tersebut dari baris-baris lainnya. *Method* ini menerima masukan berupa nomor kolom yang akan diaplikasikan dengan aturan *naked double*, sebuah *array* yang berisi nilai-nilai, dan sebuah *array* yang berisi nomor-nomor baris.

- 1 30. `hiddenSingle()`, yaitu *method* yang mengaplikasikan aturan *hidden single* kepada *grid*
2 yang sedang diselesaikan oleh algoritma *rule based*. Aturan ini diaplikasikan untuk
3 baris (`hiddenSingleRow()`) dan untuk kolom (`hiddenSingleColumn()`).
- 4 31. `hiddenSingleRow()`, yaitu *method* yang mengaplikasikan aturan *hidden single* kepada
5 baris-baris yang ada di dalam *grid* yang sedang diselesaikan oleh algoritma *rule based*.
- 6 32. `hiddenSingleRow(int row)`, yaitu *method* yang mengaplikasikan aturan *hidden single*
7 kepada sebuah baris yang ada di dalam *grid* yang sedang diselesaikan oleh algoritma
8 *rule based*. *Method* ini menerima masukan berupa nomor baris yang akan diaplika-
9 sikan dengan aturan *hidden single*.
- 10 33. `hiddenSingleColumn()`, yaitu *method* yang mengaplikasikan aturan *hidden single* ke-
11 pada kolom-kolom yang ada di dalam *grid* yang sedang diselesaikan oleh algoritma
12 *rule based*.
- 13 34. `hiddenSingleColumn(int column)`, yaitu *method* yang mengaplikasikan aturan *hidden*
14 *single* kepada sebuah kolom yang ada di dalam *grid* yang sedang diselesaikan oleh
15 algoritma *rule based*. *Method* ini menerima masukan berupa nomor kolom yang akan
16 diaplikasikan dengan aturan *hidden single*.
- 17 35. `hiddenSingle(int row, int column)`, yaitu *method* yang mengaplikasikan aturan *hidden*
18 *single* kepada sebuah sel yang sedang diselesaikan oleh algoritma *rule based*. *Method*
19 ini menerima masukan berupa nomor baris dan nomor kolom dari sel yang akan
20 diaplikasikan dengan aturan *hidden single*.
- 21 36. `setCellValue(int row, int column, int value)`, yaitu *method* untuk mengisi sebuah sel
22 dengan nilai yang telah ditentukan. *Method* ini menerima masukan berupa nomor
23 baris dan nomor kolom dari sel yang akan diisi dan nilai dari sel tersebut.
- 24 37. `removePossibleValues(int row, int column, int value)`, yaitu *method* untuk menghapus
25 kemungkinan nilai yang sudah digunakan dalam sebuah sel. *Method* ini menghapus
26 nilai tersebut dari baris yang sama dan kolom yang sama. *Method* ini menerima
27 masukan berupa nomor baris, nomor kolom, dan nilai yang akan dihapus dari sel-sel
28 lain dalam baris dan kolom tempat sel tersebut berada.
- 29 38. `removeImpossibleValuesCage(Cage cage, ArrayList<Integer> values)`, yaitu *method*
30 untuk menghabus kemungkinan nilai yang tidak mungkin dari sel-sel di dalam sebuah
31 *cage*. *Method* ini menerima masukan berupa sebuah *cage* dan sebuah *array* yang
32 berisi nilai-nilai yang mungkin.
- 33 39. `removeImpossibleValuesCell(int row, int column, ArrayList<Integer> values)`, yaitu
34 *method* untuk menghabus kemungkinan nilai yang tidak mungkin dari sebuah sel yang
35 diminta. *Method* ini menerima masukan berupa nomor baris dan nomor kolom dari
36 sel yang diminta, dan sebuah *array* yang berisi nilai-nilai yang mungkin.
- 37 40. `createRetainAllArray()`, yaitu *method* yang menghasilkan *array* yang berisi semua
38 angka dari 1 sampai ukuran dari *grid*. *Method* ini menghasilkan keluaran berupa
39 *array* yang berisi semua angka dari 1 sampai ukuran dari *grid*.



powered by Astah

Gambar 4.14: Diagram kelas SolverRuleBased.

- 1 41. `getGridArrayList()`, yaitu *method* untuk mendapatkan isi dari *grid* dalam bentuk *ArrayList*. Method ini menghasilkan keluaran berupa isi dari *grid* dalam bentuk *ArrayList*.
 - 2 42. `getGrid`, yaitu *method* untuk mendapatkan *grid*. Method ini menghasilkan keluaran berupa *grid*.
 - 3 43. `getSolution`, yaitu *method* untuk mendapatkan *grid* yang sudah diselesaikan oleh algoritma *rule based*. *Method* ini menghasilkan keluaran berupa *grid* yang sudah diselesaikan oleh algoritma *rule based*.
 - 4 44. `printGrid()`, yaitu *method* untuk mencetak isi *grid* ke layar.
 - 5 45. `printPossibleValues()`, yaitu *method* untuk mencetak kemungkinan angka-angka yang valid untuk setiap sel di dalam *grid* ke layar.
- 12 Diagram kelas SolverRuleBased dapat dilihat pada Gambar 4.14.

¹ **4.4.7 Kelas SolverGenetic**

² Kelas SolverGenetic mempunyai atribut-atribut berikut, yaitu:

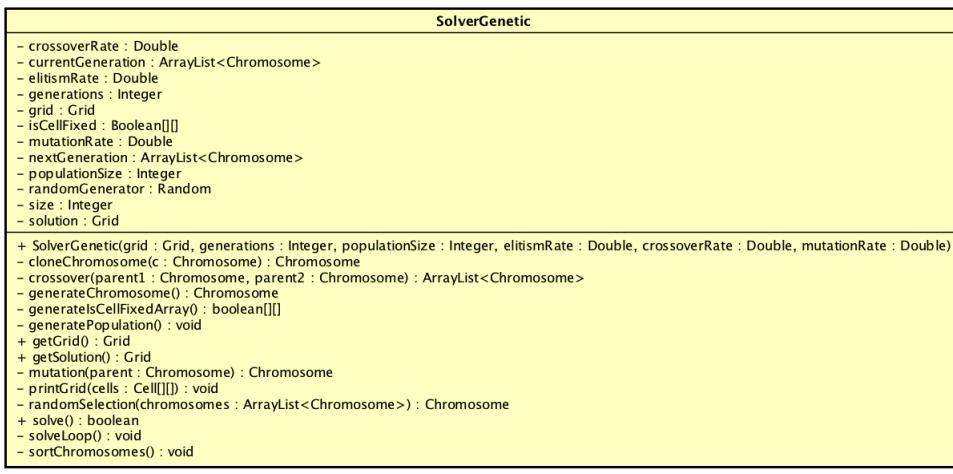
- ³ 1. grid, yaitu *grid* yang akan diselesaikan oleh *solver* dengan algoritma genetik.
- ⁴ 2. size, yaitu ukuran dari *grid* yang akan diselesaikan oleh *solver* dengan algoritma genetik.
- ⁵ 3. isGridFixed, yaitu sebuah matriks yang berisi apakah sel tersebut sudah diisi oleh algoritma *rule based* atau belum. Nilai dari sel yang sudah diisi oleh *rule based* tidak boleh diganti atau dihapus.
- ⁶ 4. randomGenerator, yaitu pembangkit angka acak.
- ⁷ 5. generations, yaitu jumlah generasi maksimum yang akan dibangkitkan oleh algoritma genetik.
- ⁸ 6. populationSize, yaitu jumlah kromosom yang akan dibangkitkan dalam sebuah generasi.
- ⁹ 7. elitismRate, yaitu parameter tingkat *elitism* dalam algoritma genetik.
- ¹⁰ 8. crossoverRate, yaitu parameter tingkat kawin silang dalam algoritma genetik.
- ¹¹ 9. mutationRate, yaitu parameter tingkat mutasi dalam algoritma genetik.
- ¹² 10. solution, yaitu *grid* yang sudah diselesaikan oleh *solver* dengan algoritma genetik.
- ¹³ 11. currentGeneration, yaitu generasi saat ini dalam algoritma genetik. Algoritma genetik akan membangkitkan generasi baru (*nextGeneration*), dan generasi baru ini akan menjadi generasi saat ini, dan algoritma akan membangkitkan generasi baru berikutnya.
- ¹⁴ 12. *nextGeneration*, yaitu generasi berikutnya dalam algoritma genetik.

²³ Kelas SolverGenetic mempunyai *method-method* berikut:

- ²⁴ 1. SolverGenetic(Grid grid, Integer generations, Integer populationSize, Double elitismRate, Double crossoverRate, Double mutationRate), yaitu konstruktor dari kelas ini. Konstruktor ini menerima masukan berupa *grid* yang akan diselesaikan oleh *solver* dengan algoritma genetik.
- ²⁵ 2. solve(), yaitu *method* yang mencoba untuk menyelesaikan teka-teki Calcudoku menggunakan algoritma genetik. *Method* ini menghasilkan keluaran apakah *solver* berhasil menyelesaikan teka-teki Calcudoku atau tidak. Algoritma genetik berhasil menyelesaikan teka-teki Calcudoku jika ada kromosom yang nilai kelayakannya 1. *Solver* akan membangkitkan generasi pertama, sedangkan generasi-generasi berikutnya akan dibangkitkan oleh *method* solveLoop().
- ²⁶ 3. solveLoop(), yaitu *method* yang membangkitkan generasi berikutnya dari generasi sebelumnya menggunakan operator algoritma genetik, yaitu *elitism*, mutasi, dan kawin silang.

- 1 4. setParameters(int generations, int populationSize, double elitismRate, double cros-
2 soverRate, double mutationRate), yaitu *method* untuk menentukan jumlah generasi
3 maksimum, jumlah kromosom dalam satu generasi, tingkat *elitism*, tingkat kawin si-
4 lang, dan tingkat mutasi untuk algoritma genetik. Method ini menerima masukan
5 berupa jumlah generasi maksimum, jumlah kromosom dalam satu generasi, tingkat
6 *elitism*, tingkat kawin silang, dan tingkat mutasi untuk algoritma genetik.
- 7 5. generateIsCellFixedArray(), yaitu *method* yang membangkitkan matriks yang berisi
8 apakah sel tersebut sudah diisi oleh algoritma *rule based* atau tidak. *Method* ini
9 menghasilkan keluaran berupa sebuah matriks yang berisi apakah sel tersebut sudah
10 diisi oleh algoritma *rule based* atau tidak.
- 11 6. generatePopulation(), yaitu *method* yang membangkitkan populasi kromosom awal.
- 12 7. generateChromosome(), yaitu *method* yang membangkitkan sebuah kromosom. *Me-*
13 *thod* ini menghasilkan keluaran berupa sebuah kromosom.
- 14 8. sortChromosomes(), yaitu *method* yang mengurutkan kromosom-kromosom dalam ge-
15 nerasi saat ini berdasarkan nilai kelayakannya.
- 16 9. randomSelection(ArrayList<Chromosome> chromosomes), yaitu *method* untuk me-
17 milih sebuah kromosom dari sebuah populasi kromosom secara acak. *Method* ini
18 menerima masukan berupa ArrayList yang berisi sekumpulan kromosom dan meng-
19 hasilkan keluaran berupa sebuah kromosom yang terpilih. Kromosom untuk proses
20 kawin silang dan proses mutasi dipilih secara acak menggunakan *method* ini.
- 21 10. cloneChromosome(Chromosome c), yaitu *method* untuk mengkopi sebuah kromosom.
22 Method ini menerima masukan berupa kromosom yang akan dikopi dan menghasilkan
23 keluaran berupa kromosom baru hasil kopian dari kromosom yang dikopi tersebut.
- 24 11. crossover(Chromosome parent1, Chromosome parent2), yaitu *method* yang mengaplikasikan
25 operator kawin silang kepada dua kromosom. *Method* ini menerima masukan
26 berupa dua kromosom yang akan dikawinsilangkan dan menghasilkan keluaran berupa
27 sebuah ArrayList yang berisi dua kromosom hasil kawin silang.
- 28 12. mutation(Chromosome parent), yaitu *method* yang mengaplikasikan operator mutasi
29 kepada sebuah kromosom. *Method* ini menerima masukan berupa kromosom yang
30 akan dimutasi dan menghasilkan keluaran berupa sebuah kromosom hasil mutasi.
- 31 13. getGrid, yaitu *method* untuk mendapatkan *grid*. Method ini menghasilkan keluaran
32 berupa *grid*.
- 33 14. getSolution, yaitu *method* untuk mendapatkan *grid* yang sudah diselesaikan oleh algo-
34 ritma genetik. *Method* ini menghasilkan keluaran berupa *grid* yang sudah diselesaikan
35 oleh algoritma genetik.
- 36 15. printGrid(), yaitu *method* untuk mencetak isi *grid* ke layar.

37 Diagram kelas SolverGenetic dapat dilihat pada Gambar 4.15.



powered by Astah

Gambar 4.15: Diagram kelas SolverGenetic.

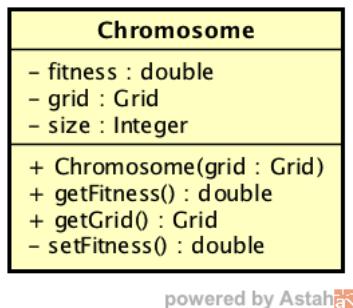
1 4.4.8 Kelas Chromosome

- 2 Kelas Chromosome mempunyai beberapa atribut, yaitu:
- 3 1. grid, yaitu sebuah *grid* yang sudah diisi dengan angka-angka secara acak.
 - 4 2. size, yaitu ukuran dari sebuah *grid*.
 - 5 3. fitness, yaitu nilai kelayakan dari sebuah *grid*.
- 6 Kelas Chromosome mempunyai beberapa *method*, yaitu:
- 7 1. Chromosome(Grid grid), yaitu konstruktor dari kelas ini. Konstruktor ini menerima masukan berupa sebuah *grid* yang sudah diisi dengan angka-angka secara acak.
 - 9 2. setFitness(), yaitu *method* yang menghitung nilai kelayakan untuk sebuah *grid*. *Method* ini menghasilkan keluaran berupa nilai kelayakan untuk *grid* tersebut.
 - 11 3. getFitness(), yaitu *method* untuk mendapatkan nilai kelayakan untuk sebuah *grid*. *Method* ini menghasilkan keluaran berupa nilai kelayakan untuk *grid* tersebut.
 - 13 4. getGrid, yaitu *method* untuk mendapatkan *grid*. Method ini menghasilkan keluaran berupa *grid*.

15 Diagram kelas Chromosome dapat dilihat pada Gambar 4.16.

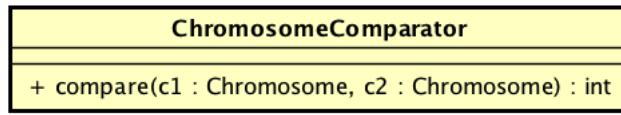
16 4.4.9 Kelas ChromosomeComparator

- 17 Kelas ChromosomeComparator tidak mempunyai variabel, tetapi kelas ini mempunyai sebuah *method*, yaitu compare(Chromosome c1, Chromosome c2). Fungsi dari *method* ini adalah membandingkan dua buah kromosom berdasarkan nilai kelayakannya. *Method* ini mengeluarkan hasil 1 jika c1 lebih besar daripada c2, -1 jika c1 lebih kecil daripada c2, atau 0 jika c1 sama dengan c2. Diagram kelas ChromosomeComparator dapat dilihat pada Gambar 4.17.



powered by Astah

Gambar 4.16: Diagram kelas Chromosome.



powered by Astah

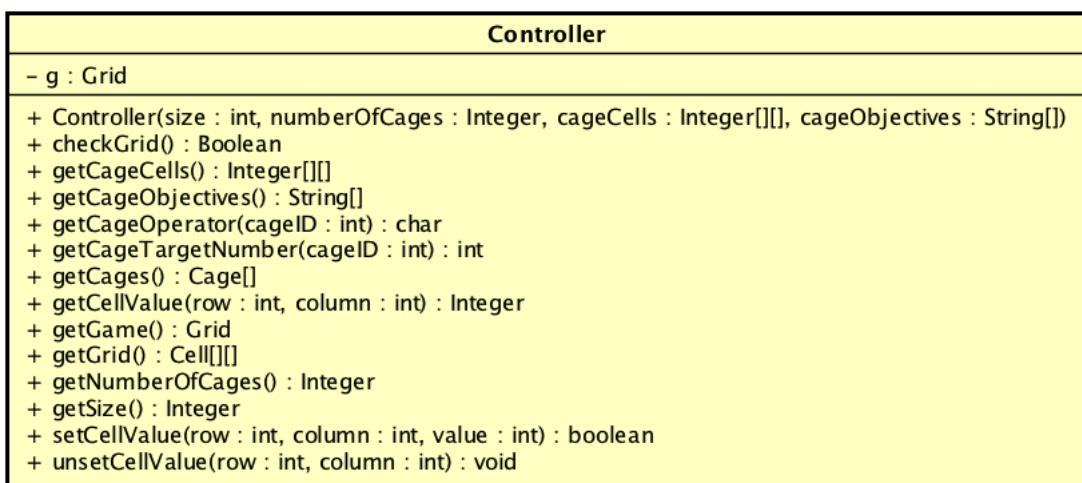
Gambar 4.17: Diagram kelas ChromosomeComparator.

¹ 4.4.10 Kelas Controller

² Kelas Controller mempunyai beberapa satut atribut, yaitu g. g adalah representasi dari *grid*
³ dalam teka-teki Calcudoku. *grid* adalah sebuah matriks yang berisi sel-sel. Matriks ini
⁴ berukuran $n \times n$.

⁵ Kelas Controller mempunyai beberapa *method*, yaitu:

- ⁶ 1. Controller(Integer size, Integer numberOfCages, Integer[][] cageCells, String[] cageObjects), yaitu konstruktor dari kelas ini. Konstruktor ini menerima masukan berupa ukuran dari matriks *grid*, banyaknya *cage* yang terdapat dalam *grid*, matriks *cage assignment*, dan array *cage objectives*.
- ¹⁰ 2. setCellValue(int row, int column, Integer value), yaitu *method* untuk mengisi sebuah sel dengan nilai yang telah ditentukan. *Method* ini menerima masukan berupa nomor baris dan nomor kolom dari sel yang akan diisi dan nilai dari sel tersebut, dan menghasilkan keluaran apakah nilai dari sel tersebut *valid* atau tidak. Nilai dari sebuah sel *valid* jika nilai dari sel tersebut tidak berulang dalam baris dan kolom tempat sel tersebut berada, dan angka-angka dari *cage* yang berisi sel tersebut mencapai angka tujuan yang telah ditentukan jika dihitung menggunakan operator yang telah ditentukan.
- ¹⁸ 3. unsetCellValue(int row, int column), yaitu *method* untuk menghapus isi dari sebuah sel. *Method* ini menerima masukan berupa nomor baris dan nomor kolom dari sel yang akan dihapus isinya.
- ²¹ 4. checkGrid(), yaitu *method* yang memeriksa apakah ada sel yang berisi nilai yang tidak *valid* atau tidak. *Method* ini akan menghasilkan apakah ada sel yang berisi nilai yang tidak *valid* atau tidak. *Method* ini hanya bekerja jika semua sel yang berada di dalam *grid* sudah diisi. Nilai dari sebuah sel *valid* jika nilai dari sel tersebut tidak berulang dalam baris dan kolom tempat sel tersebut berada, dan angka-angka



powered by Astah

Gambar 4.18: Diagram kelas Controller.

- 1 dari *cage* yang berisi sel tersebut mencapai angka tujuan yang telah ditentukan jika
 2 dihitung menggunakan operator yang telah ditentukan.
- 3 5. `getCellValue(int row, int column)`, yaitu *method* untuk mendapatkan isi dari sebuah
 4 sel. *Method* ini menerima masukan berupa nomor baris dan nomor kolom dari sel
 5 yang diminta dan menghasilkan keluaran berupa isi dari sel yang diminta tersebut.
- 6 6. `getSize()`, yaitu *method* untuk mendapatkan ukuran dari *grid*. *Method* ini mengha-
 7 silkan keluaran berupa ukuran dari *grid*.
- 8 7. `getNumberOfCages()`, yaitu *method* untuk mendapatkan jumlah *cage* yang ada di
 9 dalam *grid*. *Method* ini menghasilkan keluaran berupa jumlah *cage* yang ada di dalam
 10 *grid*.
- 11 8. `getCageCells()`, yaitu *method* untuk mendapatkan matriks *cage assignment* dari *grid*.
 12 *Method* ini menghasilkan keluaran berupa matriks *cage assignment* dari *grid*.
- 13 9. `getCageObjectives()`, yaitu *method* untuk mendapatkan *cage objectives* dari setiap
 14 *cage* dalam *grid*. *Method* ini menghasilkan keluaran berupa sebuah array yang berisi
 15 *cage objectives* dari setiap *cage* dalam *grid*.
- 16 10. `getGridContents()`, yaitu *method* untuk mendapatkan nilai dari setiap sel *grid*. *Method*
 17 ini menghasilkan keluaran berupa sebuah matriks yang berisi nilai dari setiap sel *grid*.
- 18 11. `getCages()`, yaitu *method* untuk mendapatkan semua *cage* dalam *grid*. *Method* ini
 19 menghasilkan keluaran berupa array yang berisi semua *cage* dalam *grid*.
- 20 12. `getGame()`, yaitu *method* untuk mendapatkan instansiasi saat ini dari kelas *Grid*.
 21 *Method* ini menghasilkan keluaran berupa instansiasi saat ini dari kelas *Grid*.
- 22 Diagram kelas Controller dapat dilihat pada Gambar 4.18.

¹ **4.4.11 Kelas Calcudoku**

² Kelas Calcudoku mempunyai beberapa atribut, yaitu:

- ³ 1. puzzleFile, yaitu *file* permainan yang sedang dibuka oleh perangkat lunak. *File* ini
⁴ berbentuk *file* teks.
- ⁵ 2. size, yaitu ukuran dari matriks *grid* berdasarkan *file* permainan yang sedang dibuka.
- ⁶ 3. numberOfCages, yaitu banyaknya *cage* yang terdapat dalam *grid* berdasarkan *file*
⁷ permainan yang sedang dibuka.
- ⁸ 4. cageCells, yaitu sebuah matriks *cage assignment* berdasarkan *file* permainan yang
⁹ sedang dibuka. Matriks ini merepresentasikan posisi dari setiap *cage* dalam *grid*.
- ¹⁰ 5. cageObjectives, yaitu sebuah *array* yang berisi *cage objectives* untuk setiap *cage* ber-
¹¹ dasarkan *file* permainan yang sedang dibuka. *Cage objectives* berisikan angka tujuan
¹² dan operasi matematika yang telah ditentukan.
- ¹³ 6. c, yaitu sebuah instansiasi dari kelas Controller. c berfungsi untuk menghubungkan
¹⁴ kelas-kelas yang berada di dalam *package* model dengan kelas-kelas yang berada di
¹⁵ dalam *package* view.
- ¹⁶ 7. menuBar, yaitu menu *bar* untuk perangkat lunak ini.
- ¹⁷ 8. menuFile, yaitu menu File di dalam menu *bar*.
- ¹⁸ 9. menuSolve, yaitu menu Solve di dalam menu *bar*.
- ¹⁹ 10. menuItemLoad, yaitu menu *item* Load Puzzle File di dalam menu File.
- ²⁰ 11. menuItemReset, yaitu menu *item* Reset Puzzle di dalam menu File.
- ²¹ 12. menuItemClose, yaitu menu *item* Close Puzzle File di dalam menu File.
- ²² 13. menuItemCheck, yaitu menu *item* Check Puzzle di dalam menu File.
- ²³ 14. menuItemExit, yaitu menu *item* Exit di dalam menu File.
- ²⁴ 15. menuItemBacktracking, yaitu menu *item* Backtracking di dalam menu Solve.
- ²⁵ 16. menuItemHybridGenetic, yaitu menu *item* Hybrid Genetic di dalam menu File.
- ²⁶ 17. fileChooser, yaitu *file chooser* untuk membuka *file* permainan.
- ²⁷ 18. gui, yaitu representasi GUI dari *file* permainan yang sedang dibuka.

²⁸ Kelas Calcudoku mempunyai beberapa *method*, yaitu:

- ²⁹ 1. Calcudoku(), yaitu konstruktor dari kelas ini. Konstruktor ini berfungsi untuk meng-
³⁰ inisialisasi *frame* GUI.
- ³¹ 2. main(String args[]), yaitu *method main* dari perangkat lunak ini. *Method* ini berfungsi
³² untuk menjalankan perangkat lunak ini.

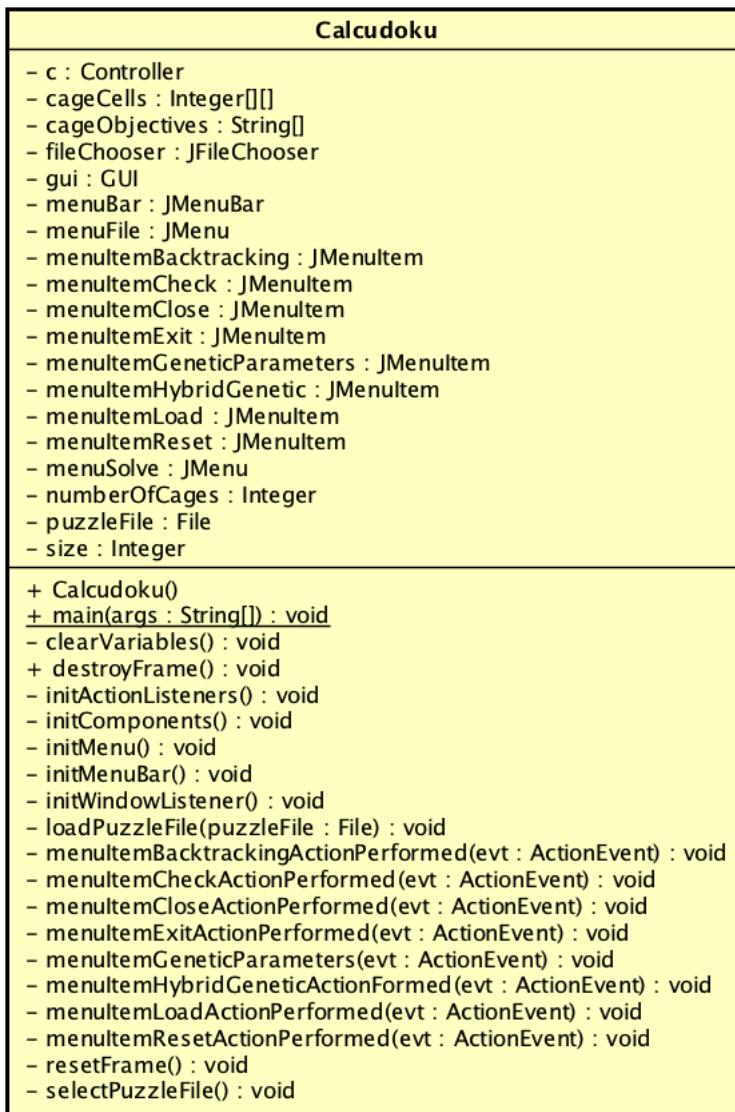
3. initComponents(), yaitu *method* untuk menginisialisasi komponen-komponen dari *frame* GUI.
4. initWindowListener(), yaitu *method* untuk menginisialisasi *window listener* untuk *frame* GUI.
5. initActionListener(), yaitu *method* untuk menginisialisasi *action listener* untuk setiap menu *item* yang ada di dalam *frame* GUI.
6. initMenu(), yaitu *method* untuk menginisialisasi menu-menu dalam menu *bar* untuk *frame* GUI.
7. initMenuBar(), yaitu *method* untuk menginisialisasi menu *bar* untuk *frame* GUI.
8. menuItemLoadActionPerformed(ActionEvent evt), yaitu *method* yang akan dijalankan jika menu "*Load Puzzle File*" dipilih. *Method* ini akan membuka *file chooser* dimana pengguna memilih *file* permainan untuk dibuka oleh perangkat lunak. Jika perangkat lunak sudah membuka sebuah *file* permainan, maka akan keluar kotak dialog "*Are you sure you want to load another puzzle file?*", jika pengguna memilih "Yes", maka *file* permainan yang baru akan dibuka oleh perangkat lunak.
9. selectPuzzleFile, yaitu *method* yang memeriksa apakah *file* permainan yang akan dibuka *valid* atau tidak. Jika perangkat lunak gagal dalam membuka *file* permainan, maka akan keluar pesan peringatan "*Error in loading puzzle file*". Jika *file* permainan yang dibuka bukan *file* teks, maka akan keluar pesan peringatan "*Invalid puzzle file*". Jika *file* permainan tidak ditemukan, maka akan keluar pesan peringatan "*Puzzle file not found*".
10. menuItemResetActionPerformed(ActionEvent evt), yaitu *method* yang akan dijalankan jika menu "*Reset Puzzle*" dipilih. *Method* ini akan mengeluarkan kotak dialog "*Are you sure you want to reset this puzzle?*", jika pengguna memilih "Yes", maka perangkat lunak akan me-*reset* permainan. Jika perangkat lunak tidak sedang membuka *file* permainan, maka akan keluar pesan peringatan "*Puzzle file not loaded*".
11. menuItemCheckActionPerformed(ActionEvent evt), yaitu *method* yang akan dijalankan jika menu "*Check Puzzle*" dipilih. *Method* ini akan memeriksa apakah ada sel yang berisi nilai yang tidak *valid* atau tidak. Nilai dari sebuah sel *valid* jika nilai dari sel tersebut tidak berulang dalam baris dan kolom tempat sel tersebut berada, dan angka-angka dari *cage* yang berisi sel tersebut mencapai angka tujuan yang telah ditentukan jika dihitung menggunakan operator yang telah ditentukan. *Method* ini juga memeriksa apakah ada sel yang kosong atau tidak. Jika perangkat lunak tidak sedang membuka *file* permainan, maka akan keluar pesan peringatan "*Puzzle file not loaded*".
12. menuItemCloseActionPerformed(ActionEvent evt), yaitu *method* yang akan dijalankan jika menu "*Close Puzzle File*" dipilih. *Method* ini akan mengeluarkan kotak dialog "*Are you sure you want to close this puzzle file?*", jika pengguna memilih "Yes", maka

- 1 perangkat lunak akan menutup *file* permainan dan instansiasi kelas GUI berdasarkan
2 *file* permainan yang ditutup. Jika perangkat lunak tidak sedang membuka *file*
3 permainan, maka akan keluar pesan peringatan "*Puzzle file not loaded*".
- 4 13. menuItemExitActionPerformed(ActionEvent evt), yaitu *method* yang akan dijalankan
5 jika menu "*Exit*" dipilih. *Method* ini akan mengeluarkan kotak dialog "*Are you sure*
6 *you want to exit this application*", jika pengguna memilih "*Yes*", maka perangkat lunak
7 akan ditutup.
- 8 14. menuItemBacktrackingActionPerformed(ActionEvent evt), yaitu *method* yang akan
9 dijalankan jika menu "*Backtracking*" dijalankan. *Method* ini akan mencoba menye-
10 lesaikan permainan berdasarkan *file* permainan yang dibuka dengan algoritma *back-
11 tracking*. Jika perangkat lunak tidak sedang membuka *file* permainan, maka akan
12 keluar pesan peringatan "*Puzzle file not loaded*".
- 13 15. menuItemHybridGeneticActionPerformed(ActionEvent evt), yaitu *method* yang akan
14 dijalankan jika menu "*Hybrid Genetic*" dijalankan. *Method* ini akan mencoba menye-
15 lesaikan permainan berdasarkan *file* permainan yang dibuka dengan algoritma *hybrid
16 genetic*. Jika perangkat lunak tidak sedang membuka *file* permainan, maka akan
17 keluar pesan peringatan "*Puzzle file not loaded*".
- 18 16. menuItemGeneticParametersActionPerformed(ActionEvent evt), yaitu *method* yang akan
19 dijalankan jika menu "*Set Genetic Algorithm Parameters*" dijalankan. *Method*
20 ini akan menampilkan *window* baru dimana pengguna bisa mengatur parameter-
21 parameter untuk algoritma genetik dengan mengisi *form* yang disediakan pada *window*
22 tersebut. Jika perangkat lunak tidak sedang membuka *file* permainan, maka akan
23 keluar pesan peringatan "*Puzzle file not loaded*".
- 24 17. loadPuzzleFile(File puzzleFile), yaitu *method* yang menginisialisasi sebuah instansiasi
25 dari kelas GUI berdasarkan *file* permainan yang dibuka. *Method* ini menerima ma-
26 sukan berupa sebuah *file* permainan. Jika perangkat lunak gagal dalam membuka *file*
27 permainan, maka akan keluar pesan peringatan "*Invalid puzzle file*".
- 28 18. resetFrame(), yaitu *method* untuk me-reset *frame* setelah menutup *file* permainan.
29 *Method* ini akan menutup instansiasi kelas GUI dan menghapus isi dari variabel c
30 dalam *frame*.
- 31 19. clearVariables(), yaitu *method* untuk menghapus nilai dari variabel-variabel puzzleFi-
32 le, size, cageCells, numberofCages, dan cageObjectives dalam *frame*.
- 33 20. destroyFrame(), yaitu *method* untuk menutup *frame* saat menutup perangkat lunak.
34 *Method* ini juga akan me-reset *frame* dan menghapus nilai dari semua variabel dalam
35 *frame*.

36 Diagram kelas Calcudoku dapat dilihat pada Gambar 4.19.

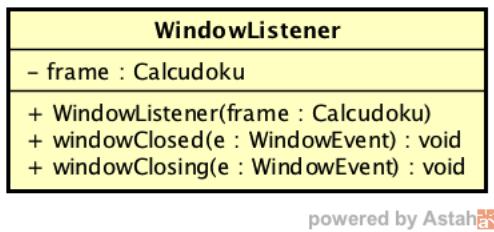
37 **4.4.12 Kelas WindowListener**

38 Kelas WindowListener hanya mempunyai satu atribut, yaitu frame. frame adalah sebuah
39 instansiasi dari kelas Calcudoku. Kelas ini mempunyai beberapa *method*, yaitu:

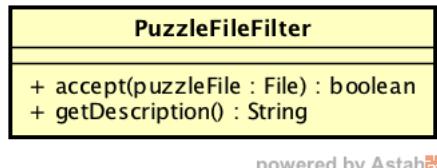


powered by Astah

Gambar 4.19: Diagram kelas Calcudoku.



Gambar 4.20: Diagram kelas WindowListener.



Gambar 4.21: Diagram kelas PuzzleFileFilter.

1. 1. WindowListener(Calcudoku frame), yaitu konstruktor dari kelas ini. Konstruktor ini menerima masukan berupa sebuah instansiasi dari kelas Calcudoku.
2. 2. windowClosing(WindowEvent e), yaitu *method* yang meng-*override method* dari kelas WindowAdapter. *Method* ini berfungsi untuk menutup semua komponen GUI saat perangkat lunak ditutup.
3. 3. windowClosed(WindowEvent e), yaitu *method* yang meng-*override method* dari kelas WindowAdapter. *Method* ini berfungsi untuk menutup perangkat lunak setelah semua komponen GUI ditutup.

9. Diagram kelas WindowListener dapat dilihat pada Gambar 4.20.

10. 4.4.13 Kelas PuzzleFileFilter

11. Kelas PuzzleFileFilter tidak mempunyai atribut, tetapi memiliki beberapa *method*, yaitu:

12. 1. accept(File puzzleFile), yaitu *method* yang meng-*override method* dari kelas FileFilter. *Method* ini berfungsi untuk memeriksa apakah file yang akan dibuka memenuhi syarat atau tidak. File yang memenuhi syarat adalah file teks.
13. 2. getDescription(), yaitu *method* yang meng-*override method* dari kelas FileFilter. *Method* ini berfungsi untuk menampilkan deskripsi file yang memenuhi syarat.

17. Diagram kelas PuzzleFileFilter dapat dilihat pada Gambar 4.21.

18. 4.4.14 Kelas GUI

19. Kelas GUI mempunyai beberapa atribut, yaitu:

20. 1. c, yaitu sebuah instansiasi dari kelas Controller. c berfungsi untuk menghubungkan kelas-kelas yang berada di dalam *package* model dengan kelas-kelas yang berada di dalam *package* view.

- 1 2. game, yaitu sebuah instansi dari kelas Grid berdasarkan *file* permainan yang sedang dibuka.
- 2 3. size, yaitu ukuran dari matriks *grid* berdasarkan *file* permainan yang sedang dibuka.
- 4 4. numberOfCages, yaitu banyaknya *cage* yang terdapat dalam *grid* berdasarkan *file* permainan yang sedang dibuka.
- 5 5. cageCells, yaitu sebuah matriks *cage assignment* berdasarkan *file* permainan yang sedang dibuka. Matriks ini merepresentasikan posisi dari setiap *cage* dalam *grid*.
- 6 6. cageObjectives, yaitu sebuah *array* yang berisi *cage objectives* untuk setiap *cage* berdasarkan *file* permainan yang sedang dibuka. *Cage objectives* berisikan angka tujuan dan operasi matematika yang telah ditentukan.
- 7 7. grid, yaitu sebuah *array* yang berisikan semua sel yang ada di dalam *grid*.
- 8 8. cages, yaitu sebuah *array* yang berisikan semua *cage* yang ada di dalam *grid*. *textFields*, yaitu sebuah *array* yang berisikan *text field*. Setiap *text field* merepresentasikan sebuah sel yang ada di dalam *grid*. *textFieldCoordinates*, yang berfungsi untuk memetakan *text field* dengan koordinat posisinya. *cellTextFieldListeners*, yaitu sebuah *array* yang berisikan *document listener* untuk semua sel yang ada di dalam *grid*. *font*, yaitu *font* yang digunakan di dalam GUI ini. *cellSize*, yaitu ukuran sebuah sel di dalam GUI ini. *cellBorderWidth*, yaitu ukuran garis pembatas antara dua sel yang berada di dalam sebuah *cage* yang sama. *cageBorderWidth*, yaitu ukuran garis pembatas antara dua sel yang berada di dalam dua *cage* yang berbeda, dan ukuran garis pembatas *grid*.
- 9 9. generations, yaitu jumlah generasi maksimum yang akan dibangkitkan oleh algoritma genetik.
- 10 10. populationSize, yaitu jumlah kromosom yang akan dibangkitkan dalam sebuah generasi.
- 11 11. elitismRate, yaitu parameter tingkat *elitism* dalam algoritma genetik..
- 12 12. crossoverRate, yaitu parameter tingkat kawin silang dalam algoritma genetik.
- 13 13. mutationRate, yaitu parameter tingkat mutasi dalam algoritma genetik.

28 Kelas GUI mempunyai beberapa *method*, yaitu:

- 29 1. GUI(Controller c), yaitu konstruktor dari kelas ini. Konstruktor ini berfungsi untuk menginisialisasi GUI berdasarkan *file* permainan yang dibuka. *Method* ini menerima masukan berupa sebuah instansiasi dari kelas Controller.
- 30 2. getController(), yaitu *method* untuk mendapatkan instansiasi dari kelas Controller. *Method* ini menghasilkan keluaran berupa instansiasi dari kelas Controller.
- 31 3. getGridSize(), yaitu *method* untuk mendapatkan ukuran dari *grid*. *Method* ini menghasilkan keluaran berupa ukuran dari *grid*.

- 1 4. `getNumberOfCages()`, yaitu *method* untuk mendapatkan jumlah *cage* yang ada di
2 dalam *grid*. *Method* ini menghasilkan keluaran berupa jumlah *cage* yang ada di dalam
3 *grid*.
- 4 5. `getCageCells()`, yaitu *method* untuk mendapatkan matriks *cage assignment* dari *grid*.
5 *Method* ini menghasilkan keluaran berupa matriks *cage assignment* dari *grid*.
- 6 6. `getCageObjectives()`, yaitu *method* untuk mendapatkan *cage objectives* dari setiap
7 *cage* dalam *grid*. *Method* ini menghasilkan keluaran berupa sebuah array yang berisi
8 *cage objectives* dari setiap *cage* dalam *grid*.
- 9 7. `getGrid()`, yaitu *method* untuk mendapatkan nilai dari setiap sel *grid*. *Method* ini
10 menghasilkan keluaran berupa sebuah matriks yang berisi nilai dari setiap sel *grid*.
- 11 8. `getCages()`, yaitu *method* untuk mendapatkan semua *cage* dalam *grid*. *Method* ini
12 menghasilkan keluaran berupa array yang berisi semua *cage* dalam *grid*.
- 13 9. `setGeneticAlgorithmParameters(Integer generations, Integer populationSize, Double
14 elitismRate, Double crossoverRate, Double mutationRate)`, yaitu *method* untuk meng-
15 atur parameter-parameter untuk algoritma genetik. *Method* ini menerima masukan
16 berupa jumlah generasi, jumlah populasi dalam sebuah generasi, tingkat *elitism*, ting-
17 kat kawin silang, dan tingkat mutasi.
- 18 10. `initTextFields`, yaitu *method* untuk menginisialisasi *text field* yang ada di dalam *grid*
19 pada GUI.
- 20 11. `solveBacktracking()`, yaitu *method* untuk memanggil solver untuk menyelesaikan per-
21 mainan Calcudoku menggunakan algoritma *backtracking*.
- 22 12. `solveHybridGenetic()`, yaitu *method* untuk memanggil solver untuk menyelesaikan per-
23 mainan Calcudoku menggunakan algoritma *hybrid genetic*.
- 24 13. `printGridToScreen()`, yaitu *method* untuk mencetak isi *grid* ke GUI.
- 25 14. `clearGrid()`, yaitu *method* untuk menghapus isi dari semua sel yang ada di dalam
26 *grid* yang akan diselesaikan oleh *solver* sebelum *solver* mencoba menyelesaikan *grid*
27 tersebut.
- 28 15. `addCellTextFieldListeners()`, yaitu *method* yang berfungsi untuk menambahkan *docu-*
29 *ment listener* untuk semua *text field* yang ada pada GUI setelah *solver* berhasil atau
30 gagal dalam menyelesaikan permainan Calcudoku.
- 31 16. `removeCellTextFieldListeners()`, yaitu *method* yang berfungsi untuk menghapus *do-*
32 *cument listener* untuk semua *text field* yang ada pada GUI sebelum *solver* mencoba
33 untuk menyelesaikan permainan Calcudoku, untuk mencegah munculnya pesan peri-
34 ngatan jika ada sel yang kosong atau jika ada angka yang berulang di dalam sebuah
35 baris atau kolom.
- 36 17. `moveCursor(JTextField textField)`, yaitu *method* yang berfungsi untuk pindah dari se-
37 buah *text field* ke *text field* di sebelahnya dengan menggunakan tombol-tombol panah.
38 *Method* ini menerima masukan berupa sebuah *text field*.



powered by Astah

Gambar 4.22: Diagram kelas GUI.

- 1 Diagram kelas GUI dapat dilihat pada Gambar 4.22.

2 4.4.15 Kelas CellKeyListener

- 3 Kelas CellKeyListener hanya mempunyai satu atribut, yaitu gui. gui adalah sebuah instansi dari kelas GUI. Kelas ini mempunya beberapa *method*, yaitu:

- 5 1. CellKeyLIstener(GUI gui), yaitu konstruktor dari kelas ini. Konstruktor ini menerima masukan berupa sebuah instansiasi dari kelas GUI.
- 7 2. keyPressed(KeyEvent e), yaitu *method* yang meng-*override* *method* dari kelas KeyListener. *Method* ini berfungsi untuk pindah dari sebuah *text field* ke *text field* di sebelahnya dengan menggunakan tombol-tombol panah.
- 10 3. keyTyped(KeyEvent e), yaitu *method* yang meng-*override* *method* dari kelas KeyListener. *Method* ini berfungsi agar *text field* hanya bisa diisi oleh sebuah angka.
- 12 4. keyReleased(KeyEvent e), yaitu *method* yang meng-*override* *method* dari kelas KeyListener. *Method* ini kosong.

- 14 Diagram kelas CellKeyListener dapat dilihat pada Gambar 4.23.

15 4.4.16 Kelas PopupMenuListener

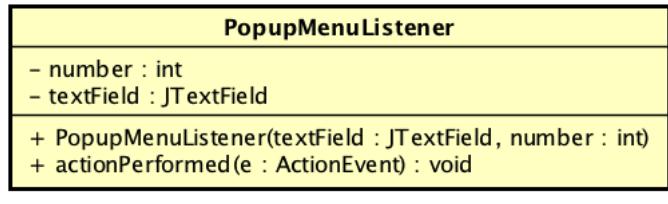
- 16 Kelas PopupMenuListener mempunyai beberapa atribut, yaitu:

- 17 1. textField, yaitu sebuah *text field* yang merepresentasikan sebuah sel yang berada di dalam *grid*.
- 19 2. number, yaitu sebuah angka.



powered by Astah

Gambar 4.23: Diagram kelas CellKeyListener.



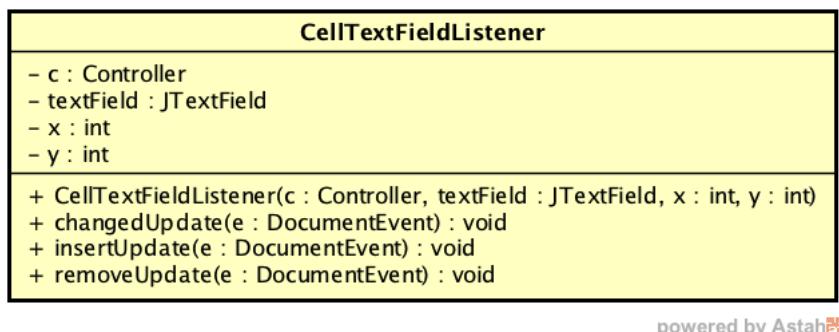
powered by Astah

Gambar 4.24: Diagram kelas PopupMenuItemListener.

- 1 Kelas PopupMenuItemListener mempunyai beberapa *method*, yaitu:
- 2 1. PopupMenuItemListener(JTextField textField, int number), yaitu konstruktor dari kelas ini. Konstruktor ini menerima masukan berupa sebuah *text field* dan sebuah angka.
 - 4 2. actionPerformed(ActionEvent e), yaitu *method* yang meng-*override* *method* dari kelas ActionListener. *Method* ini berfungsi untuk mengisi sebuah *text field* dengan angka yang dipilih.
- 7 Diagram kelas PopupMenuItemListener dapat dilihat pada Gambar 4.24.

8 4.4.17 Kelas CellTextFieldListener

- 9 Kelas CellTextFieldListener mempunyai beberapa atribut, yaitu:
- 10 1. c, yaitu sebuah instansiasi dari kelas Controller. c berfungsi untuk menghubungkan kelas-kelas yang berada di dalam *package* model dengan kelas-kelas yang berada di dalam *package* view.
 - 13 2. textField, yaitu sebuah *text field* yang merepresentasikan sebuah sel yang berada di dalam *grid*.
 - 15 3. x, yaitu koordinat x dari *text field*.
 - 16 4. y, yaitu koordinat y dari *text field*.
- 17 Kelas CellTextFieldListener mempunyai beberapa *method*, yaitu:
- 18 1. CellTextFieldListener(Controller c, JTextField textField, int x, int y), yaitu konstruktor dari kelas ini. Konstruktor ini menerima masukan berupa sebuah instansiasi dari kelas Controller, sebuah *text field*, dan koordinat dari *text field* tersebut.



powered by Astah

Gambar 4.25: Diagram kelas CellTextFieldListner.

- 1 2. insertUpdate(DocumentEvent e), yaitu *method* yang meng-*override method* dari kelas DocumentEvent. *Method* ini berfungsi untuk mengisi sebuah sel pada kelas Grid dengan angka yang diisikan ke dalam sel pada GUI.
- 4 3. removeUpdate(DocumentEvent e), yaitu *method* yang meng-*override method* dari kelas DocumentEvent. *Method* ini berfungsi untuk menghapus isi sebuah sel pada kelas Grid jika angka dalam sel pada GUI dihapus.
- 7 4. changeUpdate(DocumentEvent e), yaitu *method* yang meng-*override method* dari kelas DocumentEvent. *Method* ini berfungsi untuk menghapus isi sebuah sel pada kelas Grid dan mengisi sebuah sel pada kelas Grid dengan angka yang baru yang diisikan ke dalam sel pada GUI jika terjadi perubahan angka yang diisikan ke dalam sel pada GUI.

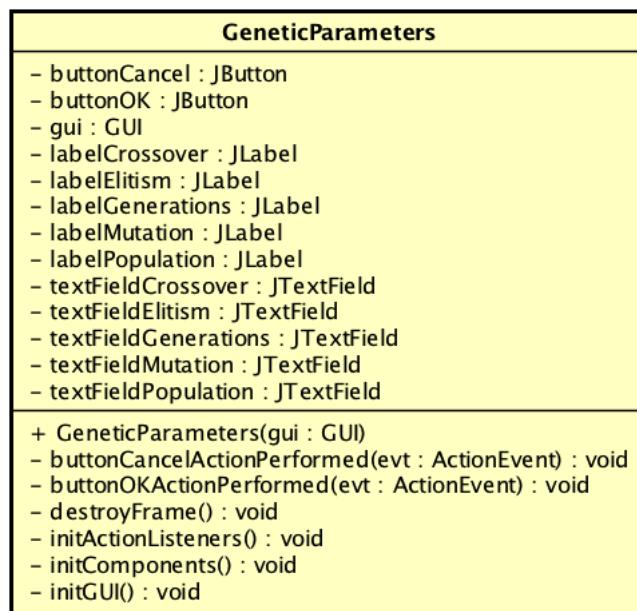
12 Diagram kelas CellTextFieldListener dapat dilihat pada Gambar 4.25.

13 4.4.18 Kelas GeneticParameters

14 Kelas GeneticParameters mempunyai beberapa atribut, yaitu:

- 15 1. gui, yaitu representasi GUI dari file permainan yang sedang dibuka.
- 16 2. labelGenerations, yaitu label yang bertuliskan "Generations".
- 17 3. labelPopulation, yaitu label yang bertuliskan "Population Size".
- 18 4. labelElitism, yaitu label yang bertuliskan "Elitism Rate".
- 19 5. labelCrossover, yaitu label yang bertuliskan "Crossover Rate".
- 20 6. labelMutation, yaitu label yang bertuliskan "Mutation Rate".
- 21 7. textFieldGenerations, yaitu *text field* untuk mengisi nilai jumlah generasi.
- 22 8. textFieldPopulation, yaitu *text field* untuk mengisi nilai jumlah populasi dalam sebuah generasi.
- 24 9. textFieldElitism, yaitu *text field* untuk mengisi nilai tingkat elitism.
- 25 10. textFieldCrossover, yaitu *text field* untuk mengisi nilai tingkat kawin silang.

- 1 11. textFieldMutation, yaitu *text field* untuk mengisi nilai tingkat mutasi.
 - 2 12. buttonOK, yaitu tombol untuk mengganti nilai-nilai parameter untuk algoritma ge-
 - 3 netik dengan nilai-nilai yang dimasukkan ke dalam *form* yang disediakan.
 - 4 13. buttonCancel, yaitu tombol untuk membatalkan perubahan nilai-nilai parameter un-
 - 5 tuk algoritma genetik.
- 6 Kelas GeneticParameters mempunyai beberapa *method*, yaitu:
- 7 1. GeneticParameters(GUI gui), yaitu konstruktor dari kelas ini. Konstruktor ini me-
 - 8 nerima masukan berupa sebuah instansiasi dari kelas GUI.
 - 9 2. initComponents(), yaitu *method* untuk menginisialisasi komponen-komponen dari *form*.
 - 10 3. initActionListener(), yaitu *method* untuk menginisialisasi *action listener* untuk setiap
 - 11 tombol yang ada di dalam *form*.
 - 12 4. initGUI(), yaitu *method* untuk menginisialisasi GUI dari *form*.
 - 13 5. initMenuBar(), yaitu *method* untuk menginisialisasi menu *bar* untuk *frame* GUI.
 - 14 6. buttonOKActionPerformed(ActionEvent evt), yaitu *method* yang akan dijalankan ji-
 - 15 ka tombol "OK" ditekan. *Method* ini akan mengganti nilai-nilai parameter untuk
 - 16 algoritma genetik dengan nilai-nilai yang telah dimasukkan ke dalam *form* yang dise-
 - 17 diakan.
 - 18 7. buttonCancelActionPerformed(ActionEvent evt), yaitu *method* yang akan dijalankan
 - 19 jika tombol "Cancel" ditekan. *Method* ini akan membatalkan perubahan nilai-nilai
 - 20 parameter untuk algoritma genetik.
 - 21 8. destroyFrame(), yaitu *method* untuk menutup *form*.
- 22 Diagram kelas GeneticParameters dapat dilihat pada Gambar 4.26.



powered by Astah

Gambar 4.26: Diagram kelas GeneticParameters.

¹ **BAB 5**

² **IMPLEMENTASI DAN PENGUJIAN**

³ Bab ini membahas tentang implementasi dan pengujian perangkat lunak berdasarkan ran-
⁴ cangan yang sudah dibuat. Ada dua jenis pengujian yang dilakukan, yaitu pengujian fung-
⁵ sional, pengujian keakuratan, dan pengujian eksperimental. Bab ini juga membahas tentang
⁶ lingkungan yang digunakan untuk pengujian perangkat lunak ini.

⁷ **5.1 Lingkungan untuk Pengujian**

⁸ Pengujian perangkat lunak ini dilakukan di Lab Komputasi FTIS Unpar ruang 9018. Semua
⁹ file permainan yang dipakai dalam pengujian ini dapat dilihat di bab Lampiran C.

¹⁰ Ada dua jenis lingkungan untuk pengujian perangkat lunak ini, yaitu:

- ¹¹ 1. Lingkungan perangkat keras, yaitu lingkungan digunakan untuk pengujian perangkat
¹² lunak ini memiliki spesifikasi berikut:
- ¹³ 2. Lingkungan perangkat lunak, yaitu lingkungan yang digunakan untuk pengujian per-
¹⁴ angkat lunak ini memiliki spesifikasi berikut:

¹⁵ **5.2 Implementasi**

¹⁶ Hasil implementasi dari rancangan perangkat lunak yang sudah dibuat ini terdiri dari dua
¹⁷ bagian, yaitu:

- ¹⁸ 1. Kode program
- ¹⁹ 2. Antarmuka perangkat lunak

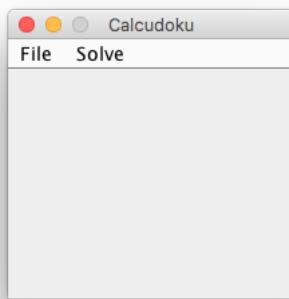
²⁰ Kedua bagian tersebut akan dijelaskan lebih lanjut di bawah ini.

Tabel 5.1: Lingkungan perangkat keras untuk pengujian perangkat lunak

Parameter	Nilai
<i>Processor</i>	Intel Core i3-4130 CPU @ 3.40 GHz
<i>RAM (Random Access Memory)</i>	6.00 GB
<i>VGA (Video Graphics Array)</i>	Intel HD Graphics 4400 dan NVIDIA GeForce GT 630

Tabel 5.2: Lingkungan perangkat lunak untuk pengujian perangkat lunak

Parameter	Nilai
Sistem Operasi	Windows 10
Bahasa Pemrograman	Java
<i>IDE (Integrated Development Environment)</i>	NetBeans IDE 8.2
<i>Library Java</i>	JDK (Java Development Kit) 1.8
<i>JVM (Java Virtual Machine)</i>	Java Version 8 Update 141



Gambar 5.1: Antarmuka perangkat lunak saat pertama kali dibuka

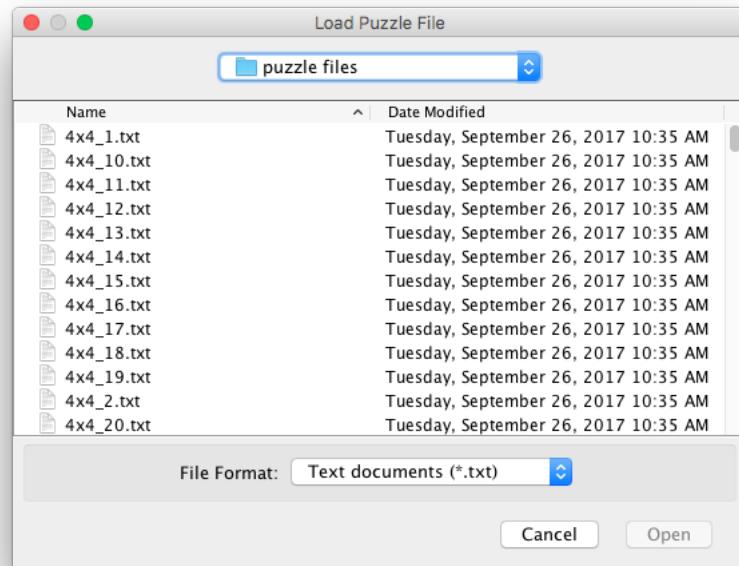
5.2.1 Kode Program

Kode program untuk perangkat lunak ini ditulis dalam bahasa pemrograman Java, berdasarkan dengan rancangan diagram kelas yang sudah dibuat, seperti dapat dilihat pada sub-bab 4.4. Seluruh kode program untuk perangkat lunak ini dapat dilihat di bab Lampiran D.

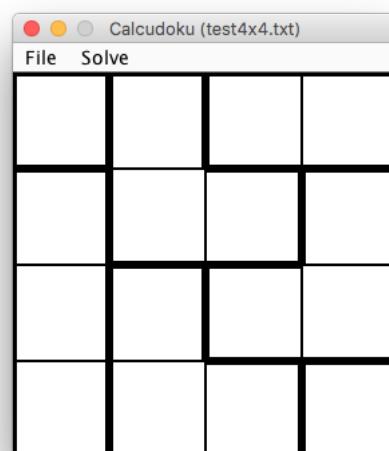
5.2.2 Antarmuka Perangkat Lunak

Antarmuka untuk perangkat lunak ini dirancang berdasarkan rancangan yang sudah dibuat, seperti dapat dilihat pada sub-bab 4.3.

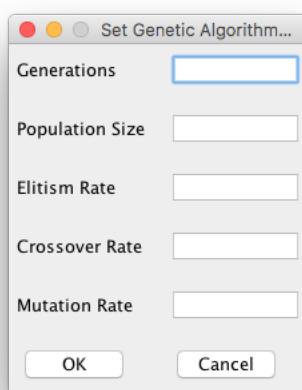
Gambar 5.1 menunjukkan antarmuka perangkat lunak saat pertama kali dibuka, belum *file* permainan dibuka. Gambar 5.2 menunjukkan kotak dialog untuk memilih *file* permainan yang akan dibuka. Gambar 5.3 menunjukkan antarmuka perangkat lunak telah membuka *file* permainan yang dipilih. Gambar 5.4 menunjukkan kotak dialog untuk mengatur nilai untuk parameter-parameter algoritma genetik. Gambar 5.5 menunjukkan antarmuka perangkat lunak setelah permainan berdasarkan *file* permainan yang telah dibuka diselesaikan.



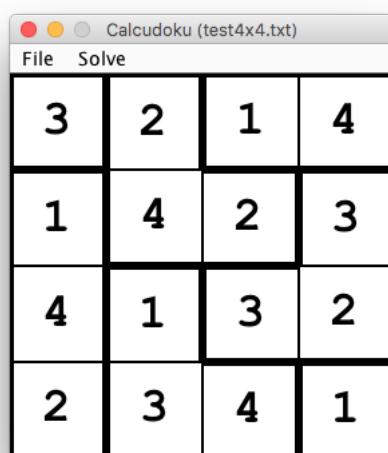
Gambar 5.2: Kotak dialog untuk memilih file permainan yang akan dibuka



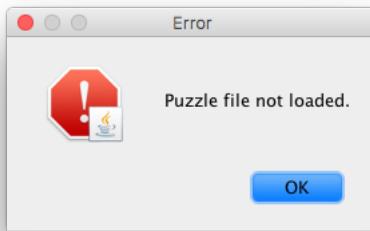
Gambar 5.3: Antarmuka perangkat lunak sesudah membuka file permainan yang dipilih



Gambar 5.4: Kotak dialog untuk mengatur nilai dari parameter-parameter algoritma genetik



Gambar 5.5: Antarmuka perangkat lunak setelah permainan berdasarkan *file* permainan yang telah dibuka diselesaikan



Gambar 5.6: Kotak pesan error "Puzzle file not loaded"

5.3 Pengujian Fungsional

Pengujian fungsional bertujuan untuk memastikan bahwa perangkat lunak dapat berfungsi sebagaimana mestinya. Skenario-skenario yang dilakukan dalam pengujian fungsional untuk perangkat lunak ini adalah:

1. Memilih menu *item* "Reset Puzzle", "Close Puzzle File", dan "Check Puzzle" dari menu "File" atau menu *item* "Backtracking", "Hybrid Genetic", dan "Set Genetic Algorithm Parameters" dari menu "Solve" sebelum membuka *file* permainan.

Jika salah satu dari menu *item* tersebut dipilih sebelum membuka *file* permainan, maka pesan error "Puzzle file not loaded" akan muncul, seperti dapat dilihat pada Gambar 5.6.

2. Membuka *file* permainan

Pemain dapat membuka *file* permainan dengan memilih menu *item* "Load Puzzle File" dalam menu "File". Jika menu tersebut dipilih maka akan keluar kotak pemilihan *file* permainan, seperti dapat dilihat pada Gambar 5.7. Pilihlah sebuah *file* permainan yang ingin dibuka. Tekan tombol "OK" untuk membuka *file* permainan tersebut, atau tekan tombol "Cancel" untuk membatalkan proses membuka *file* permainan.

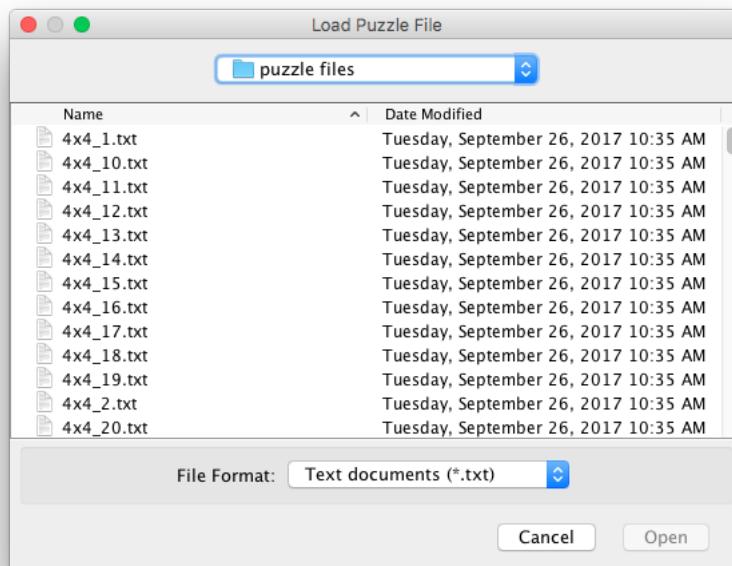
Jika perangkat lunak sudah membuka sebuah *file* permainan, maka akan keluar kotak dialog "Are you sure you want to load another puzzle file?", seperti dapat dilihat pada Gambar 5.8. Tekan tombol "Yes" untuk membuka *file* permainan tersebut, atau tekan tombol "No" untuk membatalkan proses membuka *file* permainan.

Jika *file* permainan yang dipilih berhasil dibuka, maka perangkat lunak akan menampilkan permainan berdasarkan *file* yang dipilih, seperti dapat dilihat pada Gambar 5.3.

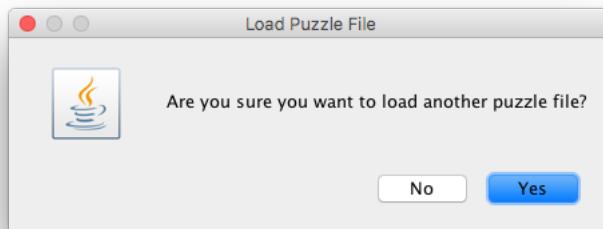
File permainan untuk perangkat lunak ini adalah *file* teks (*.txt). Format *file* permainan yang valid dapat dilihat di sub-bab 4.1.

Jika *file* permainan yang dibuka tidak *valid*, misalnya karena *file* permainan yang dibuka bukan *file* teks, atau jika *file* teks yang akan dibuka formatnya tidak *valid*, maka pesan error "Invalid puzzle file" akan muncul, seperti dapat dilihat pada Gambar 5.9.

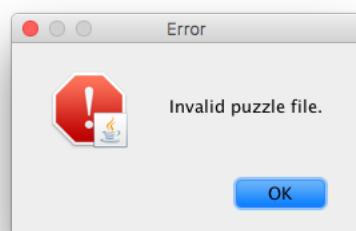
Jika ada kesalahan dalam penentuan operator untuk *cage* yang ada di dalam *grid*, misalnya operator -, atau ÷ untuk *cage* yang tidak berukuran 2 sel, operator +, atau



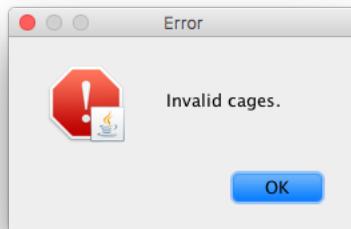
Gambar 5.7: Kotak pemilihan *file* permainan



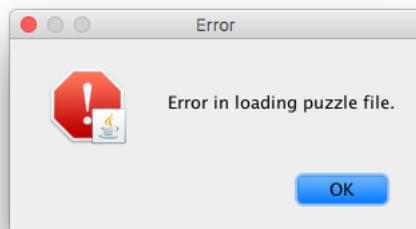
Gambar 5.8: Kotak dialog "Are you sure you want to load another puzzle file?"



Gambar 5.9: Pesan error "Invalid puzzle file"



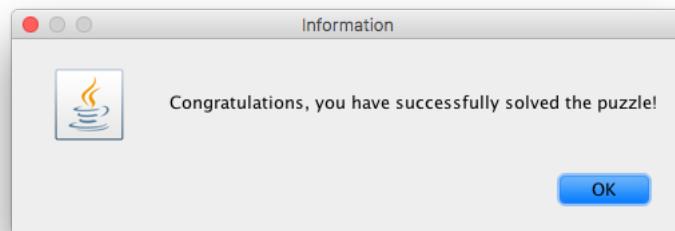
Gambar 5.10: Pesan error "*Invalid cages*"



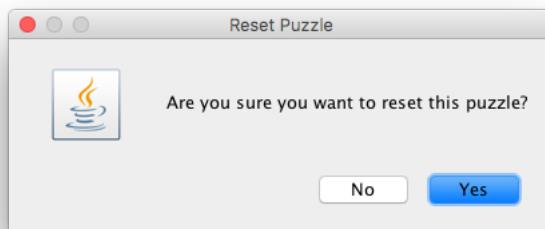
Gambar 5.11: Pesan error "*Error in loading puzzle file*"

1 ×, untuk *cage* yang berukuran 1 sel, atau operator = untuk *cage* yang berukuran
2 lebih besar dari 1 sel, maka pesan error "*Invalid cages*" akan muncul, seperti dapat
3 dilihat pada Gambar 5.10.

4 Jika salah satu dari kedua error tersebut terjadi, maka akan keluar pesan error "*Error*
5 *in loading puzzle file*", seperti dapat dilihat pada Gambar 5.11.



Gambar 5.12: Pesan informasi "*Congratulations, you have successfully solved the puzzle!*"



Gambar 5.13: Kotak dialog "*Are you sure you want to reset this puzzle?*"

3. Menyelesaikan permainan dengan usahanya sendiri

Jika pemain berhasil menyelesaikan permainan dengan usahanya sendiri, maka kotak informasi "*Congratulations, you have successfully solved the puzzle!*" akan muncul, seperti dapat dilihat pada Gambar 5.12.

4. Me-reset permainan

Pemain dapat me-reset permainan dengan memilih menu item "*Reset Puzzle*" dalam menu "*File*". Jika menu item ini dipilih, maka akan keluar kotak dialog "*Are you sure you want to reset this puzzle?*", seperti dapat dilihat pada Gambar 5.13. Tekan tombol "*Yes*" untuk me-reset permainan, atau tekan tombol "*No*" untuk membatalkan proses *reset* permainan.

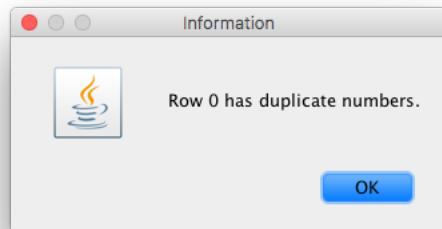
5. Memeriksa permainan jika ada nilai yang salah di dalam *grid*.

Perangkat lunak ini dapat memeriksa jika ada nilai yang salah di dalam *grid* secara otomatis.

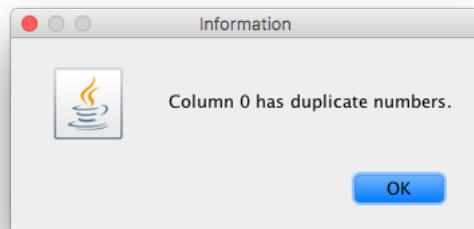
Jika ada nilai yang berulang dalam sebuah baris, maka akan keluar kotak informasi "*Row (nomor baris) has duplicate numbers*", seperti dapat dilihat pada Gambar 5.14.

Jika ada nilai yang berulang dalam sebuah kolom, maka akan keluar kotak informasi "*Column (nomor kolom) has duplicate numbers*", seperti dapat dilihat pada Gambar 5.15.

Jika nilai-nilai dalam sebuah *cage* tidak mencapai nilai tujuan yang telah ditentukan jika dihitung dengan operasi matematika yang telah ditentukan, maka akan keluar



Gambar 5.14: Pesan informasi "*Row* (nomor baris) *has duplicate numbers*"



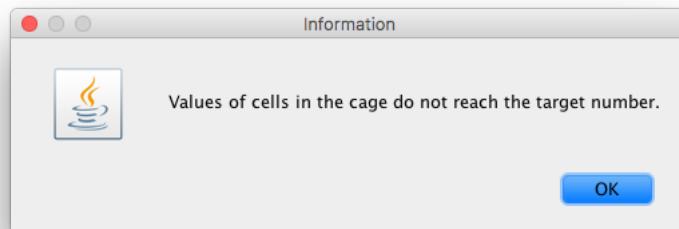
Gambar 5.15: Pesan informasi "*Column* (nomor kolom) *has duplicate numbers*"

1 kotak informasi "*Values of cells in the cage do not reach the target number*", seperti
2 dapat dilihat pada Gambar 5.16.

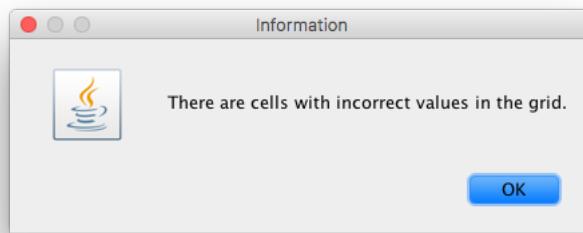
3 Pemain juga dapat meminta perangkat lunak untuk memeriksa permainan jika ada
4 nilai yang salah di dalam *grid* secara manual, dengan memilih menu item "*Check*
5 *Puzzle*" dalam menu "*File*".

6 Jika menu item ini dijalankan, dan ternyata ada nilai yang salah di dalam *grid*, maka
7 akan muncul kotak informasi "*There are cells with incorrect values in the grid*", seperti
8 dapat dilihat pada Gambar 5.17.

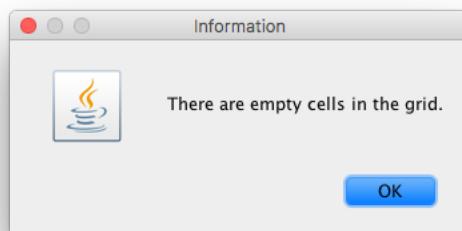
9 Jika menu item ini dijalankan dan ternyata ada sel yang masih kosong, maka akan
10 muncul kotak informasi "*There are empty cells in the grid*", seperti dapat dilihat pada
11 Gambar 5.18.



Gambar 5.16: Pesan informasi "*Values of cells in the cage do not reach the target number*"



Gambar 5.17: Pesan informasi "*There are cells with incorrect values in the grid*"



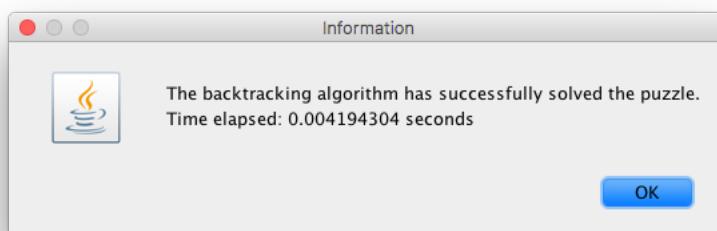
Gambar 5.18: Pesan informasi "*There are empty cells in the grid*"

1 6. Menyelesaikan permainan dengan menggunakan algoritma *backtracking*

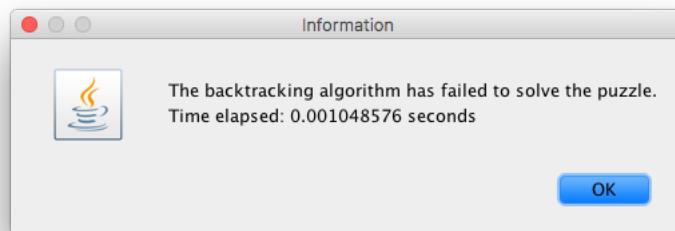
2 Pemain dapat meminta perangkat lunak untuk menyelesaikan permainan dengan sa-
3 larah satu dari dua *solver* yang disediakan. Salah satu dari kedua *solver* tersebut adalah
4 *solver* dengan algoritma *backtracking*. Untuk menggunakan *solver* ini, pemain memi-
5 lih menu item "*Backtracking*" dalam menu "*Solve*".

6 Jika *solver* ini berhasil dalam menyelesaikan permainan, maka akan muncul kotak
7 informasi "*The backtracking algorithm has successfully solved the puzzle*", dan waktu
8 yang dibutuhkan oleh *solver* untuk menyelesaikan permainan tersebut, seperti dapat
9 dilihat pada Gambar 5.19.

10 Jika gagal, maka akan muncul kotak informasi "*The backtracking algorithm has failed*

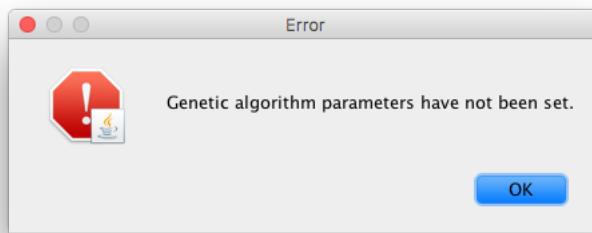


Gambar 5.19: Pesan informasi "*The backtracking algorithm has successfully solved the puzzle*"

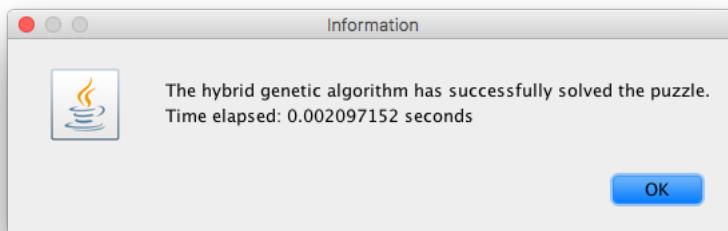


Gambar 5.20: Pesan informasi "*The backtracking algorithm has failed to solve the puzzle*"

- 1 *to solve the puzzle*", dan waktu yang dibutuhkan oleh *solver* untuk menemukan bahwa
2 tidak ada solusi untuk permainan tersebut, seperti dapat dilihat pada Gambar 5.20



Gambar 5.21: Pesan informasi "*Genetic algorithm parameters have not been set*"



Gambar 5.22: Pesan informasi "*The hybrid genetic algorithm has successfully solved the puzzle*"

1 7. Menyelesaikan permainan dengan menggunakan algoritma *hybrid genetic*

2 Pemain dapat meminta perangkat lunak untuk menyelesaikan permainan dengan sa-
3 lah satu dari dua *solver* yang disediakan. Salah satu dari kedua *solver* tersebut
4 adalah *solver* dengan algoritma *hybrid genetic*. Untuk menggunakan *solver* ini, pe-
5 main memilih menu item "*Hybrid Genetic*" dalam menu "*Solve*". Jika nilai untuk
6 parameter-parameter algoritma genetik belum ditentukan, maka akan keluar pesan
7 error "*Genetic algorithm parameters have not been set*", seperti dapat dilihat pada
8 Gambar 5.21.

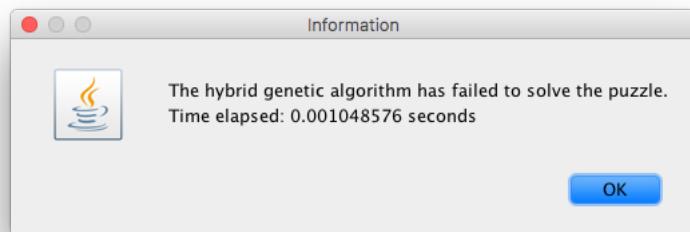
9 Jika *solver* ini berhasil dalam menyelesaikan permainan, maka akan muncul kotak
10 informasi "*The hybrid genetic algorithm has successfully solved the puzzle*", dan waktu
11 yang dibutuhkan oleh *solver* untuk menyelesaikan permainan tersebut, seperti dapat
12 dilihat pada Gambar 5.22.

13 Jika gagal, maka akan muncul kotak informasi "*The hybrid genetic algorithm has failed*
14 *to solve the puzzle*", dan waktu yang dibutuhkan oleh *solver* untuk menemukan bahwa
15 tidak ada solusi untuk permainan tersebut, seperti dapat dilihat pada Gambar 5.23

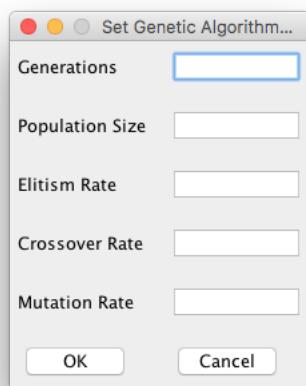
16 8. Mengatur nilai untuk parameter-parameter algoritma genetik

17 Pemain dapat mengatur sendiri nilai untuk parameter-parameter algoritma genetik
18 dengan memilih menu item "*Set Genetic Algorithm Parameters*" dalam menu "*Solve*".
19 Akan muncul sebuah *form* seperti dapat dilihat pada Gambar 5.24.

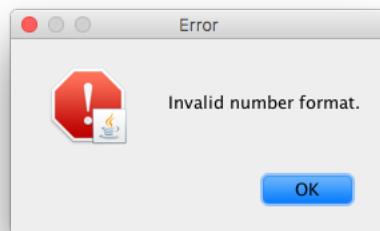
20 Isilah *form* tersebut dengan nilai yang diinginkan untuk setiap parameter algoritma



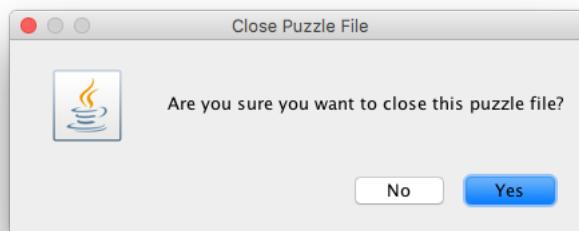
Gambar 5.23: Pesan informasi "*The hybrid genetic algorithm has failed to solve the puzzle*"



Gambar 5.24: *Form* untuk mengatur nilai untuk parameter-parameter algoritma genetik



Gambar 5.25: Pesan error "Invalid number format"



Gambar 5.26: Kotak dialog "Are you sure you want to close this puzzle file?"

genetik. Tekan tombol "OK" untuk menyimpan nilai-nilai telah diisikan ke dalam *form*, atau tekan tombol "Cancel" untuk membatalkan proses mengatur nilai untuk parameter-parameter genetik. Jika angka yang dimasukkan ke dalam *form* tidak *valid*, maka akan muncul pesan error "Invalid number format", seperti dapat dilihat pada Gambar 5.25.

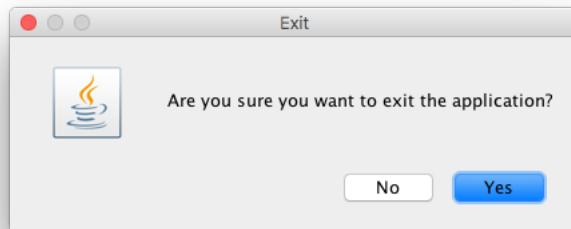
Jika jumlah dari tingkat *elitism*, tingkat mutasi, dan tingkat kawin silang kurang dari 100%, maka beberapa kromosom dari generasi sebelumnya akan diambil secara acak dan dimasukkan ke dalam generasi berikutnya sehingga jumlah kromosom dalam generasi berikutnya tetap sama. Jika jumlah dari tingkat *elitism*, tingkat mutasi, dan tingkat kawin silang lebih dari 100%, maka beberapa kromosom dari generasi berikutnya akan dibuang secara acak sehingga jumlah kromosom dalam generasi berikutnya tetap sama.

9. Menutup *file* permainan

Pemain dapat menutup *file* permainan dengan memilih menu *item* "Close Puzzle File" dalam menu "File". Jika menu *item* ini dipilih, maka akan keluar kotak dialog "Are you sure you want to close this puzzle file?", seperti dapat dilihat pada Gambar 5.26. Tekan tombol "Yes" untuk menutup *file* permainan, atau tekan tombol "No" untuk membatalkan proses menutup *file* permainan.

10. Menutup perangkat lunak

Pemain dapat menutup perangkat lunak dengan memilih menu *item* "Exit" dalam menu "File". Jika menu *item* ini dipilih, maka akan keluar kotak dialog "Are you sure you



Gambar 5.27: Kotak dialog "Are you sure you want to exit the application?"

```

3 5
1 2 3
1 3 3
4 5 5
3+
1=
8+
3=
3+

```

Gambar 5.28: File masukan untuk pengujian keakuratan

1 *want to exit the application?*, seperti dapat dilihat pada Gambar 5.27. Tekan tombol
 2 "Yes" untuk menutup perangkat lunak, atau tekan tombol "No" untuk membatalkan
 3 proses menutup perangkat lunak.

4 5.4 Pengujian Keakuratan

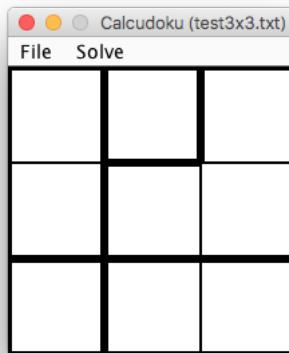
5 Pengujian keakuratan dilakukan untuk memastikan bahwa permainan yang ditampilkan
 6 oleh perangkat lunak sesuai dengan *file* permainan yang dibuka. Skenario-skenario yang
 7 akan dilakukan dalam pengujian keakuratan untuk perangkat lunak ini adalah:

8 1. Keakuratan dalam menerjemahkan isi *file* masukan menjadi keluaran berupa GUI
 9 Untuk menguji keakuratan perangkat lunak dalam menerjemahkan isi *file* masukan
 10 menjadi keluaran berupa GUI, maka perangkat lunak akan diuji dengan membuka
 11 sebuah *file* masukan, dan melihat apakah keluarannya, yaitu GUI-nya sesuai dengan
 12 *file* masukan yang dibuka.

13 Dalam kasus ini, perangkat lunak akan membuka *file* masukan yang isinya dapat
 14 dilihat pada Gambar 5.28.

15 Perangkat lunak ini lalu menerjemahkan isi *file* masukan menjadi keluaran berupa
 16 GUI yang dapat dilihat pada Gambar 5.29.

17 Seharusnya, petunjuk, yaitu angka tujuan dan operasi matematika yang ditentukan
 18 untuk sebuah *cage*, ditampilkan pada sudut kiri atas dari sel yang paling atas atau
 19 yang paling kiri dalam *cage* tersebut. Tetapi, karena dalam perangkat lunak ini
 20 sebuah sel adalah sebuah *text field*, dan sebuah *text field* tidak dapat diisi dengan



Gambar 5.29: GUI permainan berdasarkan *file* masukan yang dapat dilihat pada Gambar 5.28

1 dua teks yang berbeda (petunjuk untuk sel tersebut dan isi dari sel tersebut), maka
 2 petunjuk ditampilkan sebagai *tooltip* yang akan muncul saat sel-sel yang berada di
 3 dalam sebuah *cage* di-hover. Setiap sel memiliki *tooltip* yang berisi petunjuk sesuai
 4 dengan *cage* tempat sel tersebut berada.

5 Dalam perangkat lunak ini, setiap sel sudah memiliki *tooltip* sesuai dengan *cage* tem-
 6 pat sel tersebut berada.

7 Sel-sel dalam *cage* 1, yaitu sel pada baris ke-1 dan kolom ke-1, dan sel pada baris ke-2
 8 dan kolom ke-1, memiliki *tooltip* "3+". Sel dalam *cage* 2, yaitu sel pada baris ke-1 dan
 9 kolom ke-2, memiliki *tooltip* "1=". Sel-sel dalam *cage* 3, yaitu sel pada baris ke-1 dan
 10 kolom ke-3, sel pada baris ke-2 dan kolom ke-2, dan sel pada baris ke-2 dan kolom
 11 ke-3, memiliki *tooltip* "8+". Sel dalam *cage* 4, yaitu sel pada baris ke-3 dan kolom
 12 ke-1, memiliki *tooltip* "3=". Sel-sel dalam *cage* 5, yaitu sel pada baris ke-3 dan kolom
 13 ke-2, dan sel pada baris ke-3 dan kolom ke-3, memiliki *tooltip* "3+".

14 *Grid* dibatasi oleh garis tebal, setiap *cage* dibatasi oleh garis tebal, dua sel yang berada
 15 di dalam dua *cage* yang berbeda dibatasi oleh garis tebal, dan dua sel yang berada di
 16 dalam *cage* yang sama dibatasi oleh garis tipis.

17 Dalam perangkat lunak ini, *grid* dan setiap sel sudah memiliki garis pembatas yang
 18 tepat. *Grid* dibatasi oleh garis tebal.

19 Pada baris ke-1, sel pada kolom ke-1 dan kolom ke-2 dibatasi oleh garis tebal karena
 20 kedua sel tersebut berada dalam dua *cage* yang berbeda, sel pada kolom ke-2 dan
 21 kolom ke-3 dibatasi oleh garis tebal karena kedua sel tersebut berada dalam dua *cage*
 22 yang berbeda. Pada baris ke-2, sel pada kolom ke-1 dan kolom ke-2 dibatasi oleh garis
 23 tebal karena kedua sel tersebut berada dalam dua *cage* yang berbeda, sel pada kolom
 24 ke-2 dan kolom ke-3 dibatasi oleh garis tipis karena kedua sel tersebut berada dalam
 25 dua *cage* yang sama. Pada baris ke-3, sel pada kolom ke-1 dan kolom ke-2 dibatasi
 26 oleh garis tebal karena kedua sel tersebut berada dalam dua *cage* yang berbeda, sel
 27 pada kolom ke-2 dan kolom ke-3 dibatasi oleh garis tipis karena kedua sel tersebut
 28 berada dalam dua *cage* yang sama.

1 Pada kolom ke-1, sel pada baris ke-1 dan baris ke-2 dibatasi oleh garis tips karena
2 kedua sel tersebut berada dalam dua *cage* yang sama, sel pada baris ke-2 dan baris
3 ke-3 dibatasi oleh garis tebal karena kedua sel tersebut berada dalam dua *cage* yang
4 berbeda. Pada kolom ke-2, sel pada baris ke-1 dan baris ke-2 dibatasi oleh garis tebal
5 karena kedua sel tersebut berada dalam dua *cage* yang berbeda, sel pada baris ke-2
6 dan baris ke-3 dibatasi oleh garis tebal karena kedua sel tersebut berada dalam dua
7 *cage* yang berbeda. Pada kolom ke-3, sel pada baris ke-1 dan baris ke-2 dibatasi oleh
8 garis tipis karena kedua sel tersebut berada dalam dua *cage* yang sama, sel pada baris
9 ke-2 dan baris ke-3 dibatasi oleh garis tebal karena kedua sel tersebut berada dalam
10 dua *cage* yang berbeda.

11 2. Keakuratan dalam menampilkan pesan informasi pada waktunya.

12 Untuk menguji keakuratan perangkat lunak dalam menampilkan pesan informasi pada
13 waktunya, maka perangkat lunak harus diuji dengan menyelesaikan sebuah permainan,
14 dan melihat apa kotak informasi ditampilkan pada waktunya atau tidak.

15 Dalam kasus ini, permainan berdasarkan *file* yang isinya ditampilkan pada Gambar
16 [5.28](#) akan diselesaikan.

17 *Cage* 2, yang hanya berukuran satu sel yang terletak pada baris ke-1 dan kolom
18 ke-2, dan memiliki petunjuk "1=". Isilah sel tersebut dengan angka 2. Karena 2
19 bukan angka tujuan dari *cage* tersebut, maka pesan informasi "*Values of cells in the cage do not reach the target number*" seperti yang dapat dilihat pada Gambar [5.16](#)
20 akan muncul. Hapuslah angka 2 dari sel tersebut, dan isilah dengan angka 1, yang
21 merupakan angka tujuan dari *cage* tersebut.

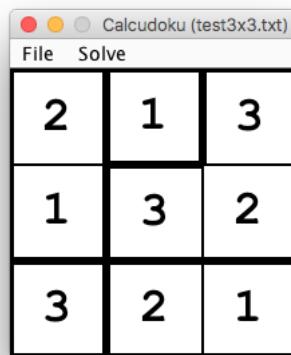
22 Isilah sel pada baris ke-1 dan kolom ke-1 dengan angka 1. Karena ada angka yang
23 berulang dalam satu baris, maka pesan informasi "*Row has duplicate numbers*" seperti
24 yang dapat dilihat pada Gambar [5.14](#) akan muncul. Hapuslah angka 1 dari sel tersebut.
25 Sekarang isilah sel pada baris ke-1 dan kolom ke-1 dengan angka 1. Karena
26 ada angka yang berulang dalam satu baris, maka pesan informasi "*Row has duplicate numbers*" seperti yang dapat dilihat pada Gambar [5.14](#) akan muncul. Hapuslah angka
27 1 dari sel tersebut.

28 Isilah sel pada baris ke-2 dan kolom ke-2 dengan angka 1. Karena ada angka yang
29 berulang dalam satu kolom, maka pesan informasi "*Column has duplicate numbers*"
30 seperti yang dapat dilihat pada Gambar [5.15](#) akan muncul. Hapuslah angka 1 dari
31 sel tersebut.

32 Selesaikan permainan ini dengan mengisi sel-sel lainnya dengan angka yang benar.
33 Solusi dari permainan ini dapat dilihat pada Gambar [5.30](#).

34 Setelah mengisi seluruh sel di dalam *grid* sesuai dengan solusi tersebut, maka pesan
35 informasi "*Congratulations, you have successfully solved the puzzle*" seperti yang dapat
36 dilihat pada Gambar [5.12](#) akan muncul.

37 Perangkat lunak ini telah berhasil memunculkan pesan informasi pada waktunya.



Gambar 5.30: Solusi untuk permainan berdasarkan *file* masukan yang dapat dilihat pada Gambar 5.28

Tabel 5.3: Hasil pengujian algoritma *backtracking* untuk Calcudoku

Ukuran Grid	Tingkat Keberhasilan	Kecepatan
4×4	100%	0.067 detik
5×5	100%	0.701 detik
6×6	100%	13.84 detik
7×7	100%	482.653 detik
8×8	100%	2134.655 detik

1 5.5 Pengujian Algoritma *Backtracking*

2 Pengujian algoritma *backtracking* untuk Calcudoku dilakukan untuk mengetahui keberha-
3 silan dan kecepatan algoritma *backtracking* dalam menyelesaikan permainan Calcudoku.

4 Berdasarkan hasil pengujian yang dapat dilihat pada Tabel 5.3, algoritma *backtracking*
5 dapat menyelesaikan semua permainan yang diujikan. Semakin besar ukuran *grid*, maka
6 waktu yang dibutuhkan oleh algoritma *backtracking* untuk menyelesaikan permainan se-
7 makin lama. Pada ukuran *grid* yang besar, algoritma *backtracking* sangat lambat dalam
8 menyelesaikan permainan.

9 Hasil pengujian selengkapnya dapat dilihat pada Lampiran B.

10 5.6 Pengujian dan Eksperimen Algoritma *Hybrid Genetic*

11 Pengujian algoritma *hybrid genetic* untuk Calcudoku dilakukan untuk mengetahui keberha-
12 silan dan kecepatan algoritma *hybrid genetic* dalam menyelesaikan permainan Calcudoku.

13 Pada algoritma *backtracking* tidak ada parameter yang nilainya dapat diubah, sedangkan
14 pada algoritma *hybrid genetic*, nilai dari parameter-parameter untuk algoritma genetik
15 dapat diubah. Tidak ada nilai *default* untuk parameter-parameter dari algoritma genetik
16 ini. Nilai dari parameter-parameter tersebut harus diisi sendiri oleh pemain.

17 Dalam kasus ini, pengujian eksperimental dilakukan dengan melakukan pengujian ke-
18 berhasilan dan kecepatan dari *solver* dengan algoritma *hybrid genetic* dengan mengatur
19 nilai dari parameter-parameter dari algoritma genetik dengan nilai yang berbeda-beda un-

Tabel 5.4: Nilai untuk parameter-parameter algoritma genetik untuk setiap percobaan yang dilakukan

Skenario	Populasi	Generasi	<i>Elitism</i>	Mutasi	Kawin Silang
1	1000	100	10%	10%	80%
2	1000	100	5%	10%	85%
3	1000	100	10%	5%	85%
4	1000	100	5%	5%	90%
5	100	100	10%	10%	80%
6	100	100	5%	10%	85%
7	100	100	10%	5%	85%
8	100	100	5%	5%	90%
9	1000	10	10%	10%	80%
10	1000	10	5%	10%	85%
11	1000	10	10%	5%	85%
12	1000	10	5%	5%	90%
13	100	10	10%	10%	80%
14	100	10	5%	10%	85%
15	100	10	10%	5%	85%
16	100	10	5%	5%	90%

Tabel 5.5: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku (Skenario 1)

Ukuran Grid	Rata-Rata Tingkat Keberhasilan	Rata-Rata Kecepatan	Rata-Rata Jumlah Sel Diisi Algoritma Rule Based
4×4	100%	3.579 detik	3
5×5	42.308%	8.389 detik	9
6×6	0%	-	-
7×7	0%	-	-
8×8	0%	-	-

1 tuk setiap percobaan.

2 Terdapat 16 skenario yang dilakukan untuk percobaan ini. Nilai untuk parameter-
3 parameter untuk algoritma genetik untuk setiap percobaan yang dilakukan dapat dilihat
4 pada Tabel 5.4.

5 5.6.1 Skenario 1

6 Hasil pengujian Skenario 1 dapat dilihat pada Tabel 5.5.

Tabel 5.6: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku (Skenario 2)

Ukuran Grid	Rata-Rata Tingkat Keberhasilan	Rata-Rata Kecepatan	Rata-Rata Jumlah Sel Diisi Algoritma Rule Based
4×4	100%	4.002 detik	3
5×5	42.308%	9.258 detik	9
6×6	0%	-	-
7×7	0%	-	-
8×8	0%	-	-

Tabel 5.7: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku (Skenario 3)

Ukuran Grid	Rata-Rata Tingkat Keberhasilan	Rata-Rata Kecepatan	Rata-Rata Jumlah Sel Diisi Algoritma Rule Based
4×4	100%	3.751 detik	3
5×5	42.308%	8.806 detik	9
6×6	0%	-	-
7×7	0%	-	-
8×8	0%	-	-

¹ 5.6.2 Skenario 2

² Hasil pengujian Skenario 2 dapat dilihat pada Tabel 5.6.

³ 5.6.3 Skenario 3

⁴ Hasil pengujian Skenario 3 dapat dilihat pada Tabel 5.7.

⁵ 5.6.4 Skenario 4

⁶ Hasil pengujian Skenario 4 dapat dilihat pada Tabel 5.8.

Tabel 5.8: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku (Skenario 4)

Ukuran Grid	Rata-Rata Tingkat Keberhasilan	Rata-Rata Kecepatan	Rata-Rata Jumlah Sel Diisi Algoritma Rule Based
4×4	100%	4.175 detik	3
5×5	42.308%	9.676 detik	9
6×6	0%	-	-
7×7	0%	-	-
8×8	0%	-	-

Tabel 5.9: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku (Skenario 5)

Ukuran Grid	Rata-Rata Tingkat Keberhasilan	Rata-Rata Kecepatan	Rata-Rata Jumlah Sel Diisi Algoritma Rule Based
4×4	61.538%	0.498 detik	5
5×5	19.231%	0.311 detik	14
6×6	0%	-	-
7×7	0%	-	-
8×8	0%	-	-

Tabel 5.10: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku (Skenario 6)

Ukuran Grid	Rata-Rata Tingkat Keberhasilan	Rata-Rata Kecepatan	Rata-Rata Jumlah Sel Diisi Algoritma Rule Based
4×4	61.538%	0.553 detik	5
5×5	19.231%	0.339 detik	14
6×6	0%	-	-
7×7	0%	-	-
8×8	0%	-	-

¹ **5.6.5 Skenario 5**

² Hasil pengujian Skenario 5 dapat dilihat pada Tabel 5.9.

³ **5.6.6 Skenario 6**

⁴ Hasil pengujian Skenario 6 dapat dilihat pada Tabel 5.9.

⁵ **5.6.7 Skenario 7**

⁶ Hasil pengujian Skenario 7 dapat dilihat pada Tabel 5.11.

Tabel 5.11: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku (Skenario 7)

Ukuran Grid	Rata-Rata Tingkat Keberhasilan	Rata-Rata Kecepatan	Rata-Rata Jumlah Sel Diisi Algoritma Rule Based
4×4	61.538%	0.524 detik	5
5×5	19.231%	0.325 detik	14
6×6	0%	-	-
7×7	0%	-	-
8×8	0%	-	-

Tabel 5.12: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku (Skenario 8)

Ukuran Grid	Rata-Rata Tingkat Keberhasilan	Rata-Rata Kecepatan	Rata-Rata Jumlah Sel Diisi Algoritma Rule Based
4×4	61.538%	0.579 detik	5
5×5	19.231%	0.352 detik	14
6×6	0%	-	-
7×7	0%	-	-
8×8	0%	-	-

Tabel 5.13: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku (Skenario 9)

Ukuran Grid	Rata-Rata Tingkat Keberhasilan	Rata-Rata Kecepatan	Rata-Rata Jumlah Sel Diisi Algoritma Rule Based
4×4	35.897%	0.511 detik	7
5×5	15.385%	0.487 detik	15
6×6	0%	-	-
7×7	0%	-	-
8×8	0%	-	-

1 5.6.8 Skenario 8

2 Hasil pengujian Skenario 8 dapat dilihat pada Tabel 5.12.

3 5.6.9 Skenario 9

4 Hasil pengujian Skenario 9 dapat dilihat pada Tabel 5.13.

5 5.6.10 Skenario 10

6 Hasil pengujian Skenario 10 dapat dilihat pada Tabel 5.14.

Tabel 5.14: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku (Skenario 10)

Ukuran Grid	Rata-Rata Tingkat Keberhasilan	Rata-Rata Kecepatan	Rata-Rata Jumlah Sel Diisi Algoritma Rule Based
4×4	35.897%	0.511 detik	7
5×5	15.385%	0.487 detik	15
6×6	0%	-	-
7×7	0%	-	-
8×8	0%	-	-

Tabel 5.15: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku (Skenario 11)

Ukuran Grid	Rata-Rata Tingkat Keberhasilan	Rata-Rata Kecepatan	Rata-Rata Jumlah Sel Diisi Algoritma Rule Based
4×4	35.897%	0.511 detik	7
5×5	15.385%	0.487 detik	15
6×6	0%	-	-
7×7	0%	-	-
8×8	0%	-	-

Tabel 5.16: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku (Skenario 12)

Ukuran Grid	Rata-Rata Tingkat Keberhasilan	Rata-Rata Kecepatan	Rata-Rata Jumlah Sel Diisi Algoritma Rule Based
4×4	35.897%	0.511 detik	7
5×5	15.385%	0.487 detik	15
6×6	0%	-	-
7×7	0%	-	-
8×8	0%	-	-

¹ 5.6.11 Skenario 11

² Hasil pengujian Skenario 11 dapat dilihat pada Tabel 5.15.

³ 5.6.12 Skenario 12

⁴ Hasil pengujian Skenario 12 dapat dilihat pada Tabel 5.16.

⁵ 5.6.13 Skenario 13

⁶ Hasil pengujian Skenario 13 dapat dilihat pada Tabel 5.17.

Tabel 5.17: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku (Skenario 13)

Ukuran Grid	Rata-Rata Tingkat Keberhasilan	Rata-Rata Kecepatan	Rata-Rata Jumlah Sel Diisi Algoritma Rule Based
4×4	23.077%	0.054 detik	9
5×5	15.385%	0.077 detik	15
6×6	0%	-	-
7×7	0%	-	-
8×8	0%	-	-

Tabel 5.18: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku (Skenario 14)

Ukuran Grid	Rata-Rata Tingkat Keberhasilan	Rata-Rata Kecepatan	Rata-Rata Jumlah Sel Diisi Algoritma Rule Based
4×4	23.077%	0.054 detik	9
5×5	15.385%	0.077 detik	15
6×6	0%	-	-
7×7	0%	-	-
8×8	0%	-	-

Tabel 5.19: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku (Skenario 15)

Ukuran Grid	Rata-Rata Tingkat Keberhasilan	Rata-Rata Kecepatan	Rata-Rata Jumlah Sel Diisi Algoritma Rule Based
4×4	23.077%	0.054 detik	9
5×5	15.385%	0.077 detik	15
6×6	0%	-	-
7×7	0%	-	-
8×8	0%	-	-

1 5.6.14 Skenario 14

2 Hasil pengujian Skenario 14 dapat dilihat pada Tabel 5.18.

3 5.6.15 Skenario 15

4 Hasil pengujian Skenario 15 dapat dilihat pada Tabel 5.19.

5 5.6.16 Skenario 16

6 Hasil pengujian Skenario 16 dapat dilihat pada Tabel 5.20.

Tabel 5.20: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku (Skenario 16)

Ukuran Grid	Rata-Rata Tingkat Keberhasilan	Rata-Rata Kecepatan	Rata-Rata Jumlah Sel Diisi Algoritma Rule Based
4×4	23.077%	0.054 detik	9
5×5	15.385%	0.077 detik	15
6×6	0%	-	-
7×7	0%	-	-
8×8	0%	-	-

¹ 5.6.17 Hasil Eksperimen

² Berdasarkan hasil pengujian yang telah dilakukan, ada beberapa hal yang dapat dilihat,
³ yaitu:

- ⁴ 1. Ada kemungkinan algoritma *hybrid genetic* gagal dalam menyelesaikan permainan
⁵ karena sifat acak dari algoritma *hybrid genetic* ini. Semakin besar ukuran *grid*, maka
⁶ kemungkinan algoritma *hybrid genetic* gagal dalam menyelesaikan permainan semakin
⁷ besar. Algoritma *hybrid genetic* ini gagal dalam menyelesaikan permainan Calcudoku
⁸ dengan ukuran *grid* 6×6 ke atas.
- ⁹ 2. Semakin besar ukuran *grid*, maka waktu yang dibutuhkan oleh algoritma *hybrid ge-*
¹⁰ *netic* untuk menyelesaikan semakin lama. Pada ukuran *grid* yang kecil, algoritma
¹¹ *hybrid genetic* cenderung menyelesaikan permainan lebih lambat daripada algoritma
¹² *backtracking*. Tetapi, pada ukuran *grid* yang besar, algoritma *hybrid genetic* mung-
¹³ kin mampu menyelesaikan permainan lebih cepat daripada algoritma *backtracking*,
¹⁴ tetapi hal ini tidak dapat dibuktikan karena algoritma *hybrid genetic* gagal dalam
¹⁵ menyelesaikan permainan dengan ukuran *grid* yang besar.
- ¹⁶ 3. Semakin banyak sel yang diisi dalam tahap algoritma *rule based*, semakin besar juga
¹⁷ kemungkinan algoritma genetik untuk berhasil dalam menyelesaikan permainan dan
¹⁸ semakin cepat algoritma genetik dalam menyelesaikan permainan.
- ¹⁹ 4. Nilai untuk parameter-parameter algoritma genetik mempengaruhi kecepatan dan
²⁰ tingkat keberhasilan algoritma *hybrid genetic* dalam menyelesaikan permainan. Se-
²¹ makin besar populasi dalam sebuah generasi sampai ke titik tertentu, dan semakin
²² banyak generasi sampai ke titik tertentu, maka semakin besar juga kemungkinan
²³ algoritma *hybrid genetic* berhasil dalam menyelesaikan permainan. Semakin besar
²⁴ tingkat *elitism* dan tingkat mutasi sampai ke titik tertentu, maka semakin cepat juga
²⁵ algoritma *hybrid genetic* dalam menyelesaikan permainan.

²⁶ Hasil pengujian selengkapnya dapat dilihat pada Lampiran B.

¹ **BAB 6**

² **SIMPULAN DAN SARAN**

³ Bab ini membahas tentang simpulan berdasarkan hasil dari analisis, implementasi, dan
⁴ pengujian perangkat lunak yang telah dibuat, dan saran-saran untuk penelitian dan pe-
⁵ ngembangan selanjutnya.

⁶ **6.1 Simpulan**

⁷ Berdasarkan hasil dari analisis, implementasi, dan pengujian perangkat lunak Calcudoku
⁸ yang telah dibuat, maka dapat diambil simpulan sebagai berikut:

⁹ 1. Perangkat lunak permainan teka-teki Calcudoku dengan dua *solver*, yaitu *solver* de-
¹⁰ ngan algoritma *backtracking* dan *solver* dengan algoritma *hybrid genetic*, berhasil
¹¹ dibuat.

¹² Perangkat lunak ini menerima input berupa soal teka-teki dan mampu menyelesaikan
¹³ soal teka-teki tersebut menggunakan algoritma *backtracking* dan *hybrid genetic*.

¹⁴ 2. Algoritma *backtracking* dapat menyelesaikan semua permainan yang diujikan. Pada
¹⁵ ukuran *grid* yang besar, algoritma *backtracking* sangat lambat dalam menyelesaikan
¹⁶ permainan.

¹⁷ Ada kemungkinan algoritma *hybrid genetic* gagal dalam menyelesaikan permainan
¹⁸ karena sifat acak dari algoritma *hybrid genetic* ini. Semakin besar ukuran *grid*, maka
¹⁹ kemungkinan algoritma *hybrid genetic* gagal dalam menyelesaikan permainan semakin
²⁰ besar.

²¹ Pada ukuran *grid* yang kecil, algoritma *hybrid genetic* cenderung menyelesaikan per-
²² mainan lebih lambat daripada algoritma *backtracking*. Tetapi, pada ukuran *grid* yang
²³ besar, algoritma *hybrid genetic* mungkin mampu menyelesaikan permainan lebih ce-
²⁴ pat daripada algoritma *backtracking*, tetapi hal ini tidak dapat dibuktikan karena
²⁵ algoritma *hybrid genetic* gagal dalam menyelesaikan permainan dengan ukuran *grid*
²⁶ yang besar.

²⁷ Semakin banyak sel yang diisi dalam tahap algoritma *rule based*, semakin besar juga
²⁸ kemungkinan algoritma genetik untuk berhasil dalam menyelesaikan permainan dan
²⁹ semakin cepat algoritma genetik dalam menyelesaikan permainan.

³⁰ Nilai untuk parameter-parameter algoritma genetik mempengaruhi kecepatan dan
³¹ tingkat keberhasilan algoritma *hybrid genetic* dalam menyelesaikan permainan. Se-
³² makin besar populasi dalam sebuah generasi sampai ke titik tertentu, dan semakin

1 banyak generasi sampai ke titik tertentu, maka semakin besar juga kemungkinan
2 algoritma *hybrid genetic* berhasil dalam menyelesaikan permainan. Semakin besar
3 tingkat *elitism* dan tingkat mutasi sampai ke titik tertentu, maka semakin cepat juga
4 algoritma *hybrid genetic* dalam menyelesaikan permainan.

¹ 6.2 Saran

² Saran-saran yang dapat diberikan untuk mengembangkan penelitian ini adalah:

- ³ 1. Memperbaiki GUI dari perangkat lunak ini agar petunjuk, yaitu angka tujuan dan
⁴ operasi matematika yang ditentukan untuk sebuah *cage*, dapat ditampilkan sebagai
⁵ mana mestinya, yaitu pada di sudut kiri atas sel yang paling atas dan yang paling kiri
⁶ dalam *cage* tersebut.
- ⁷ 2. Menambah aturan-aturan logika untuk algoritma *rule based*, misalnya aturan *naked*
⁸ *subset* untuk *cage* yang berukuran lebih besar dari 3 sel, aturan *hidden subset* untuk
⁹ *cage* yang berukuran lebih besar dari 2 sel, aturan *killer combination* untuk *cage*
¹⁰ yang berukuran lebih besar dari 2 sel, dan aturan *evil twin* untuk *cage* yang beru-
¹¹ kuran minimal 2 sel. Dengan menambah aturan-aturan logika untuk algoritma *rule*
¹² *based*. Diharapkan, dengan menambah aturan-aturan logika untuk algoritma *rule*
¹³ *based*, maka tingkat kesuksesan algoritma *hybrid genetic* dalam menyelesaikan per-
¹⁴ mainan Calcudoku dapat meningkat.
- ¹⁵ 3. Memperbaiki algoritma genetik, misalnya proses pemberian nilai kelayakan untuk
¹⁶ kromosom, proses pemilihan kromosom untuk kawin silang dan mutasi, proses *elitism*,
¹⁷ proses kawin silang, dan proses mutasi, sehingga tingkat kesuksesan algoritma *hybrid*
¹⁸ *genetic* dalam menyelesaikan permainan Calcudoku dapat meningkat.

1

DAFTAR REFERENSI

- 2 [1] Fahda, A. (2015) Kenken puzzle solver using backtracking algorithm. Makalah
3 IF2211 Strategi Algoritma - Semester II Tahun 2014/2015, Program Studi Tek-
4 nik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Ban-
5 dung. [http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2014-2015/
6 Makalah2015/Makalah_IF221_Strategi_Algoritma_2015_016.pdf](http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2014-2015/Makalah2015/Makalah_IF221_Strategi_Algoritma_2015_016.pdf).
- 7 [2] Johanna, O., Lukas, S., dan Saputra, K. V. I. (2012) Solving and modeling ken-ken pu-
8 zzle by using hybrid genetics algorithm. *1st International Conference on Engineering
9 and Technology Development (ICETD 2012)*, Bandar Lampung, Lampung, Indone-
10 sia, 20-21 Juni, pp. 98–102. Faculty of Engineering and Faculty of Computer Science,
11 Bandar Lampung University.

¹

LAMPIRAN A

²

ANALISIS ALGORITMA BACKTRACKING

³ Dalam lampiran ini dijelaskan tentang analisis algoritma *backtracking* seperti dijelaskan
⁴ pada sub-bab 3.1 secara lengkap.

⁵ Untuk mengilustrasikan cara kerja algoritma *backtracking*, akan digunakan permainan
⁶ teka-teki Calcudoku yang digambarkan pada Gambar A.1 sebagai contoh.

- ⁷ 1. Algoritma *backtracking* dimulai dengan teka-teki yang belum diselesaikan, seperti
⁸ yang digambarkan pada Gambar A.1 (*state 1*).
- ⁹ 2. Algoritma mengisikan sel pada baris ke-1 dan kolom ke-1 dengan angka 1 (*state 2*),
¹⁰ tetapi angka 1 tidak sesuai dengan angka tujuan dari *cage* tersebut.
- ¹¹ 3. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state 3*),
¹² tetapi angka 2 juga tidak sesuai dengan angka tujuan dari *cage* tersebut.
- ¹³ 4. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state 4*),
¹⁴ seperti dapat dilihat pada Gambar A.2, dan ternyata angka 3 sesuai dengan angka
¹⁵ tujuan dari *cage* tersebut, sehingga algoritma dapat maju ke sel berikutnya.
- ¹⁶ 5. Algoritma lalu mengisikan sel pada baris ke-1 dan kolom ke-2 dengan angka 1 (*state*
¹⁷ 5). Algoritma lalu maju ke sel berikutnya.
- ¹⁸ 6. Algoritma lalu mengisikan sel pada baris ke-1 dan kolom ke-3 dengan angka 1 (*state*
¹⁹ 6), tetapi angka 1 sudah pernah digunakan dalam baris tersebut.
- ²⁰ 7. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state 7*).
²¹ Algoritma lalu maju ke sel berikutnya.
- ²² 8. Algoritma lalu mengisikan sel pada baris ke-1 dan kolom ke-4 dengan angka 1 (*state*
²³ 8), tetapi angka 1 sudah pernah digunakan dalam baris tersebut.

3	8+	3-	
7+			
	8+	8+	
			1

Gambar A.1: Contoh permainan teka-teki dengan ukuran *grid* 4 x 4 yang belum
diselesaikan, seperti yang digambarkan pada Gambar 2.1. [1]

³ 3	⁸⁺	³⁻	
⁷⁺			
	⁸⁺	⁸⁺	
			¹

Gambar A.2: *State 4*

³ 3	⁸⁺ 1	³⁻ 2	⁴
⁷⁺			
	⁸⁺	⁸⁺	
			¹

Gambar A.3: *State 11*

- 1 9. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state 9*),
2 tetapi angka 2 sudah pernah digunakan dalam baris tersebut.
- 3 10. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state 10*),
4 tetapi angka 3 sudah pernah digunakan dalam baris tersebut.
- 5 11. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 4 (*state 11*),
6 seperti dapat dilihat pada Gambar A.3, tetapi hasilnya tidak sesuai dengan angka
7 tujuan dari *cage* tersebut.
- 8 12. Karena semua kemungkinan angka untuk baris ke-1 dan kolom ke-4 telah dicoba
9 dan gagal, maka algoritma harus mundur kembali ke (*state 7*). Algoritma mencoba
10 kemungkinan angka berikutnya, yaitu angka 3 (*state 12*), seperti dapat dilihat pada
11 Gambar A.4, tetapi angka 3 sudah pernah digunakan dalam baris tersebut.
- 12 13. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 4 (*state 13*).
13 Algoritma lalu maju ke sel berikutnya.
- 14 14. Algoritma lalu mengisikan sel pada baris ke-1 dan kolom ke-4 dengan angka 1 (*state*
15 14), tetapi angka 1 sudah pernah digunakan dalam baris tersebut.

³ 3	⁸⁺ 1	³⁻ 3	
⁷⁺			
	⁸⁺	⁸⁺	
			¹

Gambar A.4: *State 12*

³ 3	⁸⁺ 1	³⁻ 4	4
7+			
	⁸⁺	⁸⁺	
			1

Gambar A.5: *State 17*

³ 3	⁸⁺ 2	³⁻	
7+			
	⁸⁺	⁸⁺	
			1

Gambar A.6: *State 18*

- 1 15. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state 15*),
 2 tetapi hasilnya tidak sesuai dengan angka tujuan dari *cage* tersebut.
- 3 16. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state 16*),
 4 tetapi angka 3 sudah pernah digunakan dalam baris tersebut.
- 5 17. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 4 (*state 17*),
 6 seperti dapat dilihat pada Gambar A.5, tetapi angka 4 sudah pernah digunakan dalam
 7 baris tersebut.
- 8 18. Karena semua kemungkinan angka untuk baris ke-1 dan kolom ke-3 dan ke-4 telah
 9 dicoba dan gagal, maka algoritma harus mundur kembali ke (*state 5*). Algoritma men-
 10 coba kemungkinan angka berikutnya, yaitu angka 2 (*state 18*), seperti dapat dilihat
 11 pada Gambar A.6. Algoritma lalu maju ke sel berikutnya.
- 12 19. Algoritma lalu mengisikan sel pada baris ke-1 dan kolom ke-3 dengan angka 1 (*state*
 13 19), seperti dapat dilihat pada Gambar A.7. Algoritma lalu maju ke sel berikutnya.
- 14 20. Algoritma lalu mengisikan sel pada baris ke-1 dan kolom ke-4 dengan angka 1 (*state*
 15 20), tetapi angka 1 sudah pernah digunakan dalam baris tersebut.

³ 3	⁸⁺ 2	³⁻ 1	
7+			
	⁸⁺	⁸⁺	
			1

Gambar A.7: *State 19*

³ 3	⁸⁺ 2	³⁻ 1	4	
⁷⁺				
	⁸⁺	⁸⁺		
				1

Gambar A.8: *State 23*

³ 3	⁸⁺ 2	³⁻ 1	4	
⁷⁺ 1				
	⁸⁺	⁸⁺		
				1

Gambar A.9: *State 24*

- 1 21. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state 21*), tetapi angka 2 sudah pernah digunakan dalam baris tersebut.
- 3 22. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state 22*), tetapi angka 3 sudah pernah digunakan dalam baris tersebut.
- 5 23. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 4 (*state 23*), dan ternyata hasilnya sesuai dengan angka tujuan dari *cage* tersebut, seperti dapat dilihat pada Gambar A.8. Algoritma telah selesai mengisikan baris ke-1, sehingga bisa maju ke baris berikutnya.
- 9 24. Algoritma lalu mengisikan sel pada baris ke-2 dan kolom ke-1 dengan angka 1 (*state 24*), seperti dapat dilihat pada Gambar A.9. Algoritma lalu maju ke sel berikutnya.
- 11 25. Algoritma lalu mengisikan sel pada baris ke-2 dan kolom ke-2 dengan angka 1 (*state 25*), tetapi angka 1 sudah pernah digunakan dalam baris tersebut.
- 13 26. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state 26*), tetapi angka 2 sudah pernah digunakan dalam kolom tersebut.
- 15 27. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state 27*). Algoritma lalu maju ke sel berikutnya.
- 17 28. Algoritma lalu mengisikan sel pada baris ke-2 dan kolom ke-3 dengan angka 1 (*state 28*), tetapi angka 1 sudah pernah digunakan dalam baris dan kolom tersebut.
- 19 29. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state 29*), tetapi hasilnya tidak sesuai dengan angka tujuan dari *cage* tersebut.
- 21 30. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state 30*), tetapi angka 3 sudah pernah digunakan dalam baris tersebut.

³ 3	⁸⁺ 2	³⁻ 1	4
⁷⁺ 1	3	4	
	⁸⁺	⁸⁺	
			1

Gambar A.10: *State 31*

³ 3	⁸⁺ 2	³⁻ 1	4
⁷⁺ 1	4		
	⁸⁺	⁸⁺	
			1

Gambar A.11: *State 32*

- 1 31. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 4 (*state 31*),
 2 seperti dapat dilihat pada Gambar A.10, tetapi hasilnya tidak sesuai dengan angka
 3 tujuan dari *cage* tersebut.
- 4 32. Karena semua kemungkinan angka untuk baris ke-2 dan kolom ke-3 telah dicoba
 5 dan gagal, maka algoritma harus mundur kembali ke (*state 27*). Algoritma mencoba
 6 kemungkinan angka berikutnya, yaitu angka 4 (*state 32*), seperti dapat dilihat pada
 7 Gambar A.11. Algoritma lalu maju ke sel berikutnya.
- 8 33. Algoritma lalu mengisikan sel pada baris ke-2 dan kolom ke-3 dengan angka 1 (*state*
 9 33), tetapi angka 1 sudah pernah digunakan dalam baris tersebut.
- 10 34. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state 34*),
 11 dan ternyata hasilnya sesuai dengan angka tujuan dari *cage* tersebut, seperti dapat
 12 dilihat pada Gambar A.12. Algoritma lalu maju ke sel berikutnya.
- 13 35. Algoritma lalu mengisikan sel pada baris ke-2 dan kolom ke-4 dengan angka 1 (*state*
 14 35), tetapi angka 1 sudah pernah digunakan dalam baris tersebut.
- 15 36. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state 36*),
 16 tetapi angka 2 sudah pernah digunakan dalam baris tersebut.

³ 3	⁸⁺ 2	³⁻ 1	4
⁷⁺ 1	4	2	
	⁸⁺	⁸⁺	
			1

Gambar A.12: *State 34*

³ 3	⁸⁺ 2	³⁻ 1	4
⁷⁺ 1	4	2	3
	⁸⁺	⁸⁺	
			1

Gambar A.13: *State 37*

- 1 37. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state 37*),
2 seperti dapat dilihat pada Gambar A.13. Algoritma telah selesai mengisikan baris
3 ke-2, sehingga bisa maju ke baris berikutnya.
- 4 38. Algoritma lalu mengisikan sel pada baris ke-3 dan kolom ke-1 dengan angka 1 (*state*
5 38), tetapi angka 1 sudah pernah digunakan dalam kolom tersebut.
- 6 39. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state 39*).
7 Algoritma lalu maju ke sel berikutnya.
- 8 40. Algoritma lalu mengisikan sel pada baris ke-3 dan kolom ke-2 dengan angka 1 (*state*
9 40). Algoritma lalu maju ke sel berikutnya.
- 10 41. Algoritma lalu mengisikan sel pada baris ke-3 dan kolom ke-3 dengan angka 1 (*state*
11 41), tetapi angka 1 sudah pernah digunakan dalam baris dan kolom tersebut.
- 12 42. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state 42*),
13 tetapi angka 2 sudah pernah digunakan dalam baris dan kolom tersebut.
- 14 43. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state 43*).
15 Algoritma lalu maju ke sel berikutnya.
- 16 44. Algoritma lalu mengisikan sel pada baris ke-3 dan kolom ke-4 dengan angka 1 (*state*
17 44), tetapi angka 1 sudah pernah digunakan dalam baris tersebut.
- 18 45. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state 45*),
19 tetapi angka 2 sudah pernah digunakan dalam baris tersebut.
- 20 46. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state 46*),
21 tetapi angka 3 sudah pernah digunakan dalam baris dan kolom tersebut.
- 22 47. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 4 (*state 47*),
23 seperti dapat dilihat pada Gambar A.14, tetapi hasilnya tidak sesuai dengan angka
24 tujuan dari *cage* tersebut.
- 25 48. Karena semua kemungkinan angka untuk baris ke-3 dan kolom ke-4 telah dicoba dan
26 gagal, maka algoritma harus mundur kembali ke (*state 43*). Algoritma lalu mencoba
27 kemungkinan angka berikutnya, yaitu angka 4 (*state 48*), seperti dapat dilihat pada
28 Gambar A.15. Algoritma lalu maju ke sel berikutnya.
- 29 49. Algoritma lalu mengisikan sel pada baris ke-3 dan kolom ke-4 dengan angka 1 (*state*
30 49), tetapi angka 1 sudah pernah digunakan dalam baris tersebut.

³ 3	⁸⁺ 2	³⁻ 1	4
⁷⁺ 1	4	2	3
2	⁸⁺ 1	⁸⁺ 3	4
			1

Gambar A.14: *State 47*

³ 3	⁸⁺ 2	³⁻ 1	4
⁷⁺ 1	4	2	3
2	⁸⁺ 1	⁸⁺ 4	
			1

Gambar A.15: *State 48*

- 1 50. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state 50*),
 2 tetapi angka 2 sudah pernah digunakan dalam baris tersebut.
- 3 51. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state 51*),
 4 tetapi angka 3 sudah pernah digunakan dalam kolom tersebut.
- 5 52. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 4 (*state 52*),
 6 seperti dapat dilihat pada Gambar A.16, tetapi angka 3 sudah pernah digunakan
 7 dalam baris dan kolom tersebut.
- 8 53. Karena semua kemungkinan angka untuk baris ke-3 dan kolom ke-4 telah dicoba dan
 9 gagal, maka algoritma harus mundur kembali ke (*state 48*). Algoritma lalu mencoba
 10 kemungkinan angka berikutnya, yaitu angka 2 (*state 53*) seperti dapat dilihat pa-
 11 da Gambar A.17, tetapi angka 2 sudah pernah digunakan dalam baris dan kolom
 12 tersebut.
- 13 54. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state 54*).
 14 Algoritma lalu maju ke sel berikutnya.
- 15 55. Algoritma lalu mengisikan sel pada baris ke-3 dan kolom ke-3 dengan angka 1 (*state*
 16 55), tetapi angka 1 sudah pernah digunakan dalam kolom tersebut.

³ 3	⁸⁺ 2	³⁻ 1	4
⁷⁺ 1	4	2	3
2	⁸⁺ 1	⁸⁺ 4	4
			1

Gambar A.16: *State 52*

3	2	1	4
⁷⁺ 1	4	2	3
2	⁸⁺ 2	⁸⁺	
			1

Gambar A.17: *State* 53

- 1 56. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state* 56),
 2 tetapi angka 2 sudah pernah digunakan dalam baris dan kolom tersebut.
- 3 57. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state* 57),
 4 tetapi angka 3 sudah pernah digunakan dalam baris tersebut.
- 5 58. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 4 (*state* 58).
 6 Algoritma lalu maju ke sel berikutnya.
- 7 59. Algoritma lalu mengisikan sel pada baris ke-3 dan kolom ke-4 dengan angka 1 (*sta-*
 8 *te* 59). Algoritma telah selesai mengisikan baris ke-2, sehingga bisa maju ke baris
 9 berikutnya.
- 10 60. Algoritma lalu mengisikan sel pada baris ke-4 dan kolom ke-1 dengan angka 1 (*state*
 11 60), tetapi angka 1 sudah pernah digunakan dalam kolom tersebut.
- 12 61. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state* 61),
 13 tetapi angka 2 sudah pernah digunakan dalam kolom tersebut.
- 14 62. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state* 62),
 15 tetapi angka 3 sudah pernah digunakan dalam kolom tersebut.
- 16 63. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 4 (*state* 63).
 17 Algoritma lalu maju ke sel berikutnya.
- 18 64. Algoritma lalu mengisikan sel pada baris ke-4 dan kolom ke-2 dengan angka 1 (*state*
 19 64). Algoritma lalu maju ke sel berikutnya.
- 20 65. Algoritma lalu mengisikan sel pada baris ke-4 dan kolom ke-3 dengan angka 1 (*state*
 21 65), tetapi angka 1 sudah pernah digunakan dalam baris dan kolom tersebut.
- 22 66. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state* 66),
 23 tetapi angka 2 sudah pernah digunakan dalam kolom tersebut.
- 24 67. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state* 67),
 25 tetapi hasilnya tidak sesuai dengan angka tujuan dari *cage* tersebut.
- 26 68. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 4 (*state* 68),
 27 seperti dapat dilihat di Gambar A.18. tetapi angka 4 sudah pernah digunakan dalam
 28 baris dan kolom tersebut.

³ 3	⁸⁺ 2	³⁻ 1	4
⁷⁺ 1	4	2	3
2	⁸⁺ 3	⁸⁺ 4	1
4	1	4	¹

Gambar A.18: *State 68*

³ 3	⁸⁺ 2	³⁻ 1	4
⁷⁺ 1	4	2	3
2	⁸⁺ 3	⁸⁺ 4	1
4	2		¹

Gambar A.19: *State 69*

- 1 69. Karena semua kemungkinan angka untuk baris ke-4 dan kolom ke-3 telah dicoba dan
 2 gagal, maka algoritma harus mundur kembali ke (*state 64*). Algoritma lalu mencoba
 3 kemungkinan angka berikutnya, yaitu angka 2 (*state 69*), seperti dapat dilihat pada
 4 Gambar A.19, tetapi angka 2 sudah pernah digunakan dalam kolom tersebut.
- 5 70. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state 70*),
 6 tetapi angka 3 sudah pernah digunakan dalam kolom tersebut.
- 7 71. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 4 (*state 71*),
 8 seperti dapat dilihat di Gambar A.20, tetapi angka 4 sudah pernah digunakan dalam
 9 kolom tersebut.
- 10 72. Karena semua kemungkinan angka untuk baris ke-4 dan kolom ke-1 dan ke-2 telah
 11 dicoba dan gagal, maka algoritma harus mundur kembali ke (*state 59*). Algoritma
 12 lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state 72*), seperti da-
 13 pat dilihat pada Gambar A.21, tetapi angka 2 sudah pernah digunakan dalam baris
 14 tersebut.
- 15 73. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state 73*),
 16 tetapi angka 3 sudah pernah digunakan dalam baris dan kolom tersebut.

³ 3	⁸⁺ 2	³⁻ 1	4
⁷⁺ 1	4	2	3
2	⁸⁺ 3	⁸⁺ 4	1
4	4		¹

Gambar A.20: *State 71*

³ 3	⁸⁺ 2	³⁻ 1	4
⁷⁺ 1	4	2	3
2	⁸⁺ 3	⁸⁺ 4	2
			1

Gambar A.21: *State 72*

³ 3	⁸⁺ 2	³⁻ 1	4
⁷⁺ 1	4	2	3
2	⁸⁺ 3	⁸⁺ 4	4
			1

Gambar A.22: *State 74*

- 1 74. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 4 (*state 74*),
 2 seperti dapat dilihat di Gambar A.22, tetapi angka 4 sudah pernah digunakan dalam
 3 baris dan kolom tersebut.
- 4 75. Karena semua kemungkinan angka untuk baris ke-3 dan kolom ke-3 dan ke-4 telah
 5 dicoba dan gagal, maka algoritma harus mundur kembali ke (*state 54*). Algoritma
 6 lalu mencoba kemungkinan angka berikutnya, yaitu angka 4 (*state 75*), seperti dapat
 7 dilihat pada Gambar A.23, tetapi angka 4 sudah pernah digunakan dalam kolom
 8 tersebut.
- 9 76. Karena semua kemungkinan angka untuk baris ke-3 dan kolom ke-2 telah dicoba dan
 10 gagal, maka algoritma harus mundur kembali ke (*state 54*). Algoritma lalu mencoba
 11 kemungkinan angka berikutnya, yaitu angka 3 (*state 76*), seperti dapat dilihat pada
 12 Gambar A.24, tetapi angka 3 sudah pernah digunakan dalam kolom tersebut.
- 13 77. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 4 (*state 77*),
 14 seperti dapat dilihat di Gambar A.25. Algoritma lalu maju ke sel berikutnya.
- 15 78. Algoritma lalu mengisikan sel pada baris ke-3 dan kolom ke-2 dengan angka 1 (*state*
 16 78), seperti dapat dilihat di Gambar A.26. Algoritma lalu maju ke sel berikutnya.

³ 3	⁸⁺ 2	³⁻ 1	4
⁷⁺ 1	4	2	3
2	⁸⁺ 4	⁸⁺	
			1

Gambar A.23: *State 75*

³ 3	⁸⁺ 2	³⁻ 1	4
⁷⁺ 1	4	2	3
3	⁸⁺	⁸⁺	
			1

Gambar A.24: *State 76*

³ 3	⁸⁺ 2	³⁻ 1	4
⁷⁺ 1	4	2	3
4	⁸⁺	⁸⁺	
			1

Gambar A.25: *State 77*

³ 3	⁸⁺ 2	³⁻ 1	4
⁷⁺ 1	4	2	3
4	⁸⁺ 1	⁸⁺	
			1

Gambar A.26: *State 78*

³ 3	⁸⁺ 2	³⁻ 1	4
⁷⁺ 1	4	2	3
4	⁸⁺ 1	⁸⁺ 3	
			1

Gambar A.27: *State 81*

³ 3	⁸⁺ 2	³⁻ 1	4
⁷⁺ 1	4	2	3
4	⁸⁺ 1	⁸⁺ 3	2
			1

Gambar A.28: *State 83*

- 1 79. Algoritma lalu mengisikan sel pada baris ke-3 dan kolom ke-3 dengan angka 1 (*state*
2 79), tetapi angka 1 sudah pernah digunakan dalam baris dan kolom tersebut.
- 3 80. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state* 80),
4 tetapi angka 2 sudah pernah digunakan dalam kolom tersebut.
- 5 81. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state* 81),
6 seperti dapat dilihat pada Gambar A.27. Algoritma lalu maju ke sel berikutnya.
- 7 82. Algoritma lalu mengisikan sel pada baris ke-3 dan kolom ke-3 dengan angka 1 (*state*
8 82), tetapi angka 1 sudah pernah digunakan dalam baris tersebut.
- 9 83. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state* 83),
10 dan ternyata hasilnya sesuai dengan angka tujuan dari *cage* tersebut, seperti dapat
11 dilihat pada Gambar A.28. Algoritma telah selesai mengisikan baris ke-3, sehingga
12 bisa maju ke baris berikutnya.
- 13 84. Algoritma lalu mengisikan sel pada baris ke-4 dan kolom ke-1 dengan angka 1 (*state*
14 84), tetapi angka 1 sudah pernah digunakan dalam kolom tersebut.
- 15 85. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state* 85),
16 dan ternyata hasilnya sesuai dengan angka tujuan dari *cage* tersebut, seperti dapat
17 dilihat pada Gambar A.29. Algoritma lalu maju ke sel berikutnya.
- 18 86. Algoritma lalu mengisikan sel pada baris ke-4 dan kolom ke-2 dengan angka 1 (*state*
19 86), tetapi angka 1 sudah pernah digunakan dalam kolom tersebut.
- 20 87. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state* 87),
21 tetapi angka 2 sudah pernah digunakan dalam baris dan kolom tersebut.
- 22 88. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state* 88),
23 seperti dapat dilihat pada Gambar A.30. Algoritma lalu maju ke sel berikutnya.

³ 3	⁸⁺ 2	³⁻ 1	4
⁷⁺ 1	4	2	3
4	⁸⁺ 1	⁸⁺ 3	2
2			1

Gambar A.29: *State 85*

³ 3	⁸⁺ 2	³⁻ 1	4
⁷⁺ 1	4	2	3
4	⁸⁺ 1	⁸⁺ 3	2
2	3		1

Gambar A.30: *State 88*

- 1 89. Algoritma lalu mengisikan sel pada baris ke-4 dan kolom ke-3 dengan angka 1 (*state*
2 89), tetapi angka 1 sudah pernah digunakan dalam baris dan kolom tersebut.
- 3 90. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 2 (*state* 90),
4 tetapi angka 2 sudah pernah digunakan dalam baris dan kolom tersebut.
- 5 91. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 3 (*state* 91),
6 tetapi angka 3 sudah pernah digunakan dalam kolom tersebut.
- 7 92. Algoritma lalu mencoba kemungkinan angka berikutnya, yaitu angka 4 (*state* 92),
8 dan ternyata hasilnya sesuai dengan angka tujuan dari *cage* tersebut, seperti dapat
9 dilihat pada Gambar A.31. Algoritma lalu maju ke sel berikutnya.
- 10 93. Algoritma lalu mengisikan sel pada baris ke-4 dan kolom ke-4 dengan angka 1 (*state*
11 93), dan ternyata hasilnya sesuai dengan angka tujuan dari *cage* tersebut, seperti dapat
12 dilihat pada Gambar A.32. Algoritma *backtracking* telah selesai mengisi semua
13 sel dalam permainan teka-teki Calcudoku ini dengan benar.

³ 3	⁸⁺ 2	³⁻ 1	4
⁷⁺ 1	4	2	3
4	⁸⁺ 1	⁸⁺ 3	2
2	3	4	1

Gambar A.31: *State 92*

³ 3	⁸⁺ 2	³⁻ 1	4
⁷⁺ 1	4	2	3
4	⁸⁺ 1	⁸⁺ 3	2
2	3	4	¹ 1

Gambar A.32: *State 93*

¹

LAMPIRAN B

²

HASIL PENGUJIAN

³ Lampiran ini berisi hasil pengujian yang telah dilakukan.

⁴ **B.1 Algoritma *Backtracking***

⁵ Berikut adalah hasil pengujian algoritma *backtracking* untuk Calcudoku.

⁶ Hasil pengujian algoritma *backtracking* untuk Calcudoku dengan ukuran $grid\ 4\times 4$ dapat
⁷ dilihat pada Tabel [B.1](#).

⁸ Hasil pengujian algoritma *backtracking* untuk Calcudoku dengan ukuran $grid\ 5\times 5$ dapat
⁹ dilihat pada Tabel [B.2](#).

¹⁰ Hasil pengujian algoritma *backtracking* untuk Calcudoku dengan ukuran $grid\ 6\times 6$ dapat
¹¹ dilihat pada Tabel [B.3](#).

¹² Hasil pengujian algoritma *backtracking* untuk Calcudoku dengan ukuran $grid\ 7\times 7$ dapat
¹³ dilihat pada Tabel [B.4](#).

¹⁴ Hasil pengujian algoritma *backtracking* untuk Calcudoku dengan ukuran $grid\ 8\times 8$ dapat
¹⁵ dilihat pada Tabel [B.5](#).

Tabel B.1: Hasil pengujian algoritma *backtracking* untuk Calcudoku dengan ukuran *grid* 4×4

Nomor Soal	Waktu
1	0.104 detik
2	0.008 detik
3	0.134 detik
4	0.082 detik
5	0.092 detik
6	0.008 detik
7	0.074 detik
8	0.185 detik
9	0.095 detik
10	0.106 detik
11	0.048 detik
12	0.122 detik
13	0.036 detik
14	0.043 detik
15	0.058 detik
16	0.013 detik
17	0.117 detik
18	0.076 detik
19	0.079 detik
20	0.043 detik
21	0.039 detik
22	0.055 detik
23	0.063 detik
24	0.048 detik
25	0.11 detik
26	0.036 detik
27	0.083 detik
28	0.059 detik
29	0.079 detik
30	0.058 detik
31	0.167 detik
32	0.046 detik
33	0.058 detik
34	0.021 detik
35	0.069 detik
36	0.024 detik
37	0.026 detik
38	0.017 detik
39	0.036 detik

Tabel B.2: Hasil pengujian algoritma *backtracking* untuk Calcudoku dengan ukuran *grid* 5×5

Nomor Soal	Waktu
1	0.257 detik
2	1.836 detik
3	0.958 detik
4	0.068 detik
5	0.816 detik
6	0.426 detik
7	1.17 detik
8	0.931 detik
9	1.017 detik
10	0.184 detik
11	0.716 detik
12	0.524 detik
13	0.15 detik
14	0.494 detik
15	0.438 detik
16	3.224 detik
17	0.276 detik
18	0.627 detik
19	1.755 detik
20	0.264 detik
21	0.446 detik
22	0.326 detik
23	0.092 detik
24	0.944 detik
25	0.137 detik
26	0.144 detik

Tabel B.3: Hasil pengujian algoritma *backtracking* untuk Calcudoku dengan ukuran *grid* 6×6

Nomor Soal	Waktu
1	6.315 detik
2	3.072 detik
3	2.306 detik
4	1.232 detik
5	0.775 detik
6	4.233 detik
7	2.498 detik
8	0.592 detik
9	5.529 detik
10	2.498 detik
11	0.62 detik
12	3.768 detik
13	22.784 detik
14	19.724 detik
15	0.866 detik
16	5.21 detik
17	2.327 detik
18	2.958 detik
19	5.97 detik
20	6.457 detik
21	4.011 detik
22	3.128 detik
23	243.767 detik
24	0.988 detik
25	0.172 detik
26	3.628 detik
27	8.873 detik
28	5.596 detik
29	1.1 detik
30	4.112 detik
31	1.328 detik
32	2.172 detik
33	5.381 detik
34	1.018 detik
35	72.546 detik
36	14.35 detik
37	14.662 detik
38	2.638 detik
39	50.561 detik

Tabel B.4: Hasil pengujian algoritma *backtracking* untuk Calcudoku dengan ukuran *grid* 7×7

Nomor Soal	Waktu
1	5924.967 detik
2	24.596 detik
3	40.597 detik
4	26.073 detik
5	75.227 detik
6	29.977 detik
7	338.317 detik
8	43.976 detik
9	109.051 detik
10	48.554 detik
11	43.503 detik
12	8.538 detik
13	1990.996 detik
14	64.485 detik
15	270.934 detik

Tabel B.5: Hasil pengujian algoritma *backtracking* untuk Calcudoku dengan ukuran *grid* 8×8

Nomor Soal	Waktu
1	1417.117 detik
2	4249.97 detik
3	2699.821 detik
4	180.779 detik
5	563.79 detik
6	6068.212 detik
7	1923.112 detik
8	727.159 detik
9	2817.854 detik
10	65.25 detik
11	4800.963 detik
12	1691.002 detik
13	545.492 detik

¹ B.2 Algoritma *Hybrid Genetic*

² Berikut adalah hasil pengujian algoritma *hybrid genetic* untuk Calcudoku.

³ Pada semua skenario, algoritma *hybrid genetic* gagal dalam menyelesaikan permainan
⁴ dengan ukuran *grid* 6×6 ke atas.

⁵ Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku dengan ukuran *grid* 4×4
⁶ untuk Skenario 1 sampai dengan Skenario 4 dapat dilihat pada Tabel [B.6](#).

⁷ Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku dengan ukuran *grid* 4×4
⁸ untuk Skenario 5 sampai dengan Skenario 8 dapat dilihat pada Tabel [B.7](#).

⁹ Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku dengan ukuran *grid* 4×4
¹⁰ untuk Skenario 9 sampai dengan Skenario 12 dapat dilihat pada Tabel [B.8](#).

¹¹ Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku dengan ukuran *grid* 4×4
¹² untuk Skenario 13 sampai dengan Skenario 16 dapat dilihat pada Tabel [B.9](#).

¹³ Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku dengan ukuran *grid* 5×5
¹⁴ untuk Skenario 1 sampai dengan Skenario 4 dapat dilihat pada Tabel [B.10](#).

¹⁵ Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku dengan ukuran *grid* 5×5
¹⁶ untuk Skenario 5 sampai dengan Skenario 8 dapat dilihat pada Tabel [B.11](#).

¹⁷ Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku dengan ukuran *grid* 5×5
¹⁸ untuk Skenario 9 sampai dengan Skenario 12 dapat dilihat pada Tabel [B.12](#).

¹⁹ Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku dengan ukuran *grid* 5×5
²⁰ untuk Skenario 13 sampai dengan Skenario 16 dapat dilihat pada Tabel [B.13](#).

Tabel B.6: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku dengan ukuran *grid* 4×4 (Skenario 1-4)

Nomor Soal	Waktu Skenario 1	Waktu Skenario 2	Waktu Skenario 3	Waktu Skenario 4
1	9.801 detik	11.025 detik	10.413 detik	11.637 detik
2	2.457 detik	2.763 detik	2.457 detik	2.763 detik
3	9.801 detik	11.025 detik	10.413 detik	11.637 detik
4	4.905 detik	5.517 detik	5.211 detik	5.823 detik
5	0.62 detik	0.62 detik	0.62 detik	0.62 detik
6	2.457 detik	2.763 detik	2.457 detik	2.763 detik
7	2.457 detik	2.763 detik	2.457 detik	2.763 detik
8	9.801 detik	11.025 detik	10.413 detik	11.637 detik
9	0.314 detik	0.314 detik	0.314 detik	0.314 detik
10	4.905 detik	5.517 detik	5.211 detik	5.823 detik
11	4.905 detik	5.517 detik	5.211 detik	5.823 detik
12	2.457 detik	2.763 detik	2.457 detik	2.763 detik
13	1.232 detik	1.232 detik	1.232 detik	1.232 detik
14	0.314 detik	0.314 detik	0.314 detik	0.314 detik
15	9.801 detik	11.025 detik	10.413 detik	11.637 detik
16	1.232 detik	1.232 detik	1.232 detik	1.232 detik
17	1.232 detik	1.232 detik	1.232 detik	1.232 detik
18	0.62 detik	0.62 detik	0.62 detik	0.62 detik
19	4.905 detik	5.517 detik	5.211 detik	5.823 detik
20	2.457 detik	2.763 detik	2.457 detik	2.763 detik
21	2.457 detik	2.763 detik	2.457 detik	2.763 detik
22	0.314 detik	0.314 detik	0.314 detik	0.314 detik
23	9.801 detik	11.025 detik	10.413 detik	11.637 detik
24	2.457 detik	2.763 detik	2.457 detik	2.763 detik
25	0.008 detik	0.008 detik	0.008 detik	0.008 detik
26	2.457 detik	2.763 detik	2.457 detik	2.763 detik
27	4.905 detik	5.517 detik	5.211 detik	5.823 detik
28	0.314 detik	0.314 detik	0.314 detik	0.314 detik
29	2.457 detik	2.763 detik	2.457 detik	2.763 detik
30	4.905 detik	5.517 detik	5.211 detik	5.823 detik
31	9.801 detik	11.025 detik	10.413 detik	11.637 detik
32	0.008 detik	0.008 detik	0.008 detik	0.008 detik
33	9.801 detik	11.025 detik	10.413 detik	11.637 detik
34	0.314 detik	0.314 detik	0.314 detik	0.314 detik
35	0.314 detik	0.314 detik	0.314 detik	0.314 detik
36	4.905 detik	5.517 detik	5.211 detik	5.823 detik
37	2.457 detik	2.763 detik	2.457 detik	2.763 detik
38	0.314 detik	0.314 detik	0.314 detik	0.314 detik
39	4.905 detik	5.517 detik	5.211 detik	5.823 detik

Tabel B.7: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku dengan ukuran *grid* 4×4 (Skenario 5-8)

Nomor Soal	Waktu Skenario 5	Waktu Skenario 6	Waktu Skenario 7	Waktu Skenario 8
1	Gagal	Gagal	Gagal	Gagal
2	0.999 detik	1.109 detik	1.054 detik	1.164 detik
3	Gagal	Gagal	Gagal	Gagal
4	Gagal	Gagal	Gagal	Gagal
5	0.229 detik	0.256 detik	0.229 detik	0.256 detik
6	0.999 detik	1.109 detik	1.054 detik	1.164 detik
7	0.999 detik	1.109 detik	1.054 detik	1.164 detik
8	Gagal	Gagal	Gagal	Gagal
9	0.036 detik	0.036 detik	0.036 detik	0.036 detik
10	Gagal	Gagal	Gagal	Gagal
11	Gagal	Gagal	Gagal	Gagal
12	0.999 detik	1.109 detik	1.054 detik	1.164 detik
13	0.339 detik	0.394 detik	0.366 detik	0.421 detik
14	0.036 detik	0.036 detik	0.036 detik	0.036 detik
15	Gagal	Gagal	Gagal	Gagal
16	0.339 detik	0.394 detik	0.366 detik	0.421 detik
17	0.339 detik	0.394 detik	0.366 detik	0.421 detik
18	0.229 detik	0.256 detik	0.229 detik	0.256 detik
19	Gagal	Gagal	Gagal	Gagal
20	0.999 detik	1.109 detik	1.054 detik	1.164 detik
21	0.999 detik	1.109 detik	1.054 detik	1.164 detik
22	0.063 detik	0.063 detik	0.063 detik	0.063 detik
23	Gagal	Gagal	Gagal	Gagal
24	0.999 detik	1.109 detik	1.054 detik	1.164 detik
25	0.008 detik	0.008 detik	0.008 detik	0.008 detik
26	0.999 detik	1.109 detik	1.054 detik	1.164 detik
27	Gagal	Gagal	Gagal	Gagal
28	0.063 detik	0.063 detik	0.063 detik	0.063 detik
29	0.999 detik	1.109 detik	1.054 detik	1.164 detik
30	Gagal	Gagal	Gagal	Gagal
31	Gagal	Gagal	Gagal	Gagal
32	0.008 detik	0.008 detik	0.008 detik	0.008 detik
33	Gagal	Gagal	Gagal	Gagal
34	0.036 detik	0.036 detik	0.036 detik	0.036 detik
35	0.118 detik	0.118 detik	0.118 detik	0.118 detik
36	Gagal	Gagal	Gagal	Gagal
37	0.999 detik	1.109 detik	1.054 detik	1.164 detik
38	0.118 detik	0.118 detik	0.118 detik	0.118 detik
39	Gagal	Gagal	Gagal	Gagal

Tabel B.8: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku dengan ukuran *grid* 4×4 (Skenario 9-12)

Nomor Soal	Waktu Skenario 9	Waktu Skenario 10	Waktu Skenario 11	Waktu Skenario 12
1	Gagal	Gagal	Gagal	Gagal
2	Gagal	Gagal	Gagal	Gagal
3	Gagal	Gagal	Gagal	Gagal
4	Gagal	Gagal	Gagal	Gagal
5	0.62 detik	0.62 detik	0.62 detik	0.62 detik
6	Gagal	Gagal	Gagal	Gagal
7	Gagal	Gagal	Gagal	Gagal
8	Gagal	Gagal	Gagal	Gagal
9	0.314 detik	0.314 detik	0.314 detik	0.314 detik
10	Gagal	Gagal	Gagal	Gagal
11	Gagal	Gagal	Gagal	Gagal
12	Gagal	Gagal	Gagal	Gagal
13	1.232 detik	1.232 detik	1.232 detik	1.232 detik
14	0.314 detik	0.314 detik	0.314 detik	0.314 detik
15	Gagal	Gagal	Gagal	Gagal
16	1.232 detik	1.232 detik	1.232 detik	1.232 detik
17	1.232 detik	1.232 detik	1.232 detik	1.232 detik
18	0.62 detik	0.62 detik	0.62 detik	0.62 detik
19	Gagal	Gagal	Gagal	Gagal
20	Gagal	Gagal	Gagal	Gagal
21	Gagal	Gagal	Gagal	Gagal
22	0.314 detik	0.314 detik	0.314 detik	0.314 detik
23	Gagal	Gagal	Gagal	Gagal
24	Gagal	Gagal	Gagal	Gagal
25	0.008 detik	0.008 detik	0.008 detik	0.008 detik
26	Gagal	Gagal	Gagal	Gagal
27	Gagal	Gagal	Gagal	Gagal
28	0.314 detik	0.314 detik	0.314 detik	0.314 detik
29	Gagal	Gagal	Gagal	Gagal
30	Gagal	Gagal	Gagal	Gagal
31	Gagal	Gagal	Gagal	Gagal
32	0.008 detik	0.008 detik	0.008 detik	0.008 detik
33	Gagal	Gagal	Gagal	Gagal
34	0.314 detik	0.314 detik	0.314 detik	0.314 detik
35	0.314 detik	0.314 detik	0.314 detik	0.314 detik
36	Gagal	Gagal	Gagal	Gagal
37	Gagal	Gagal	Gagal	Gagal
38	0.314 detik	0.314 detik	0.314 detik	0.314 detik
39	Gagal	Gagal	Gagal	Gagal

Tabel B.9: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku dengan ukuran *grid* 4×4 (Skenario 5-8)

Nomor Soal	Waktu Skenario 13	Waktu Skenario 14	Waktu Skenario 15	Waktu Skenario 16
1	Gagal	Gagal	Gagal	Gagal
2	Gagal	Gagal	Gagal	Gagal
3	Gagal	Gagal	Gagal	Gagal
4	Gagal	Gagal	Gagal	Gagal
5	Gagal	Gagal	Gagal	Gagal
6	Gagal	Gagal	Gagal	Gagal
7	Gagal	Gagal	Gagal	Gagal
8	Gagal	Gagal	Gagal	Gagal
9	0.036 detik	0.036 detik	0.036 detik	0.036 detik
10	Gagal	Gagal	Gagal	Gagal
11	Gagal	Gagal	Gagal	Gagal
12	Gagal	Gagal	Gagal	Gagal
13	Gagal	Gagal	Gagal	Gagal
14	0.036 detik	0.036 detik	0.036 detik	0.036 detik
15	Gagal	Gagal	Gagal	Gagal
16	Gagal	Gagal	Gagal	Gagal
17	Gagal	Gagal	Gagal	Gagal
18	Gagal	Gagal	Gagal	Gagal
19	Gagal	Gagal	Gagal	Gagal
20	Gagal	Gagal	Gagal	Gagal
21	Gagal	Gagal	Gagal	Gagal
22	0.063 detik	0.063 detik	0.063 detik	0.063 detik
23	Gagal	Gagal	Gagal	Gagal
24	Gagal	Gagal	Gagal	Gagal
25	0.008 detik	0.008 detik	0.008 detik	0.008 detik
26	Gagal	Gagal	Gagal	Gagal
27	Gagal	Gagal	Gagal	Gagal
28	0.063 detik	0.063 detik	0.063 detik	0.063 detik
29	Gagal	Gagal	Gagal	Gagal
30	Gagal	Gagal	Gagal	Gagal
31	Gagal	Gagal	Gagal	Gagal
32	0.008 detik	0.008 detik	0.008 detik	0.008 detik
33	Gagal	Gagal	Gagal	Gagal
34	0.036 detik	0.036 detik	0.036 detik	0.036 detik
35	0.118 detik	0.118 detik	0.118 detik	0.118 detik
36	Gagal	Gagal	Gagal	Gagal
37	Gagal	Gagal	Gagal	Gagal
38	0.118 detik	0.118 detik	0.118 detik	0.118 detik
39	Gagal	Gagal	Gagal	Gagal

Tabel B.10: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku dengan ukuran $grid 5 \times 5$ (Skenario 1-4)

Nomor Soal	Waktu Skenario 1	Waktu Skenario 2	Waktu Skenario 3	Waktu Skenario 4
1	18.369 detik	19.899 detik	19.134 detik	20.664 detik
2	Gagal	Gagal	Gagal	Gagal
3	0.391 detik	0.391 detik	0.391 detik	0.391 detik
4	Gagal	Gagal	Gagal	Gagal
5	Gagal	Gagal	Gagal	Gagal
6	Gagal	Gagal	Gagal	Gagal
7	12.249 detik	13.779 detik	13.014 detik	14.544 detik
8	Gagal	Gagal	Gagal	Gagal
9	12.249 detik	13.779 detik	13.014 detik	14.544 detik
10	Gagal	Gagal	Gagal	Gagal
11	Gagal	Gagal	Gagal	Gagal
12	Gagal	Gagal	Gagal	Gagal
13	0.391 detik	0.391 detik	0.391 detik	0.391 detik
14	Gagal	Gagal	Gagal	Gagal
15	Gagal	Gagal	Gagal	Gagal
16	Gagal	Gagal	Gagal	Gagal
17	Gagal	Gagal	Gagal	Gagal
18	0.391 detik	0.391 detik	0.391 detik	0.391 detik
19	18.369 detik	19.899 detik	19.134 detik	20.664 detik
20	Gagal	Gagal	Gagal	Gagal
21	4.599 detik	4.981 detik	4.599 detik	4.981 detik
22	0.773 detik	0.773 detik	0.773 detik	0.773 detik
23	Gagal	Gagal	Gagal	Gagal
24	12.249 detik	13.779 detik	13.014 detik	14.544 detik
25	12.249 detik	13.779 detik	13.014 detik	14.544 detik
26	Gagal	Gagal	Gagal	Gagal

Tabel B.11: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku dengan ukuran *grid* 5×5 (Skenario 5-8)

Nomor Soal	Waktu Skenario 5	Waktu Skenario 6	Waktu Skenario 7	Waktu Skenario 8
1	Gagal	Gagal	Gagal	Gagal
2	Gagal	Gagal	Gagal	Gagal
3	0.043 detik	0.043 detik	0.043 detik	0.043 detik
4	Gagal	Gagal	Gagal	Gagal
5	Gagal	Gagal	Gagal	Gagal
6	Gagal	Gagal	Gagal	Gagal
7	Gagal	Gagal	Gagal	Gagal
8	Gagal	Gagal	Gagal	Gagal
9	Gagal	Gagal	Gagal	Gagal
10	Gagal	Gagal	Gagal	Gagal
11	Gagal	Gagal	Gagal	Gagal
12	Gagal	Gagal	Gagal	Gagal
13	0.077 detik	0.077 detik	0.077 detik	0.077 detik
14	Gagal	Gagal	Gagal	Gagal
15	Gagal	Gagal	Gagal	Gagal
16	Gagal	Gagal	Gagal	Gagal
17	Gagal	Gagal	Gagal	Gagal
18	0.043 detik	0.043 detik	0.043 detik	0.043 detik
19	Gagal	Gagal	Gagal	Gagal
20	Gagal	Gagal	Gagal	Gagal
21	1.247 detik	1.384 detik	1.315 detik	1.453 detik
22	0.146 detik	0.146 detik	0.146 detik	0.146 detik
23	Gagal	Gagal	Gagal	Gagal
24	Gagal	Gagal	Gagal	Gagal
25	Gagal	Gagal	Gagal	Gagal
26	Gagal	Gagal	Gagal	Gagal

Tabel B.12: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku dengan ukuran *grid* 5×5 (Skenario 9-12)

Nomor Soal	Waktu Skenario 9	Waktu Skenario 10	Waktu Skenario 11	Waktu Skenario 12
1	Gagal	Gagal	Gagal	Gagal
2	Gagal	Gagal	Gagal	Gagal
3	0.391 detik	0.391 detik	0.391 detik	0.391 detik
4	Gagal	Gagal	Gagal	Gagal
5	Gagal	Gagal	Gagal	Gagal
6	Gagal	Gagal	Gagal	Gagal
7	Gagal	Gagal	Gagal	Gagal
8	Gagal	Gagal	Gagal	Gagal
9	Gagal	Gagal	Gagal	Gagal
10	Gagal	Gagal	Gagal	Gagal
11	Gagal	Gagal	Gagal	Gagal
12	Gagal	Gagal	Gagal	Gagal
13	0.391 detik	0.391 detik	0.391 detik	0.391 detik
14	Gagal	Gagal	Gagal	Gagal
15	Gagal	Gagal	Gagal	Gagal
16	Gagal	Gagal	Gagal	Gagal
17	Gagal	Gagal	Gagal	Gagal
18	0.391 detik	0.391 detik	0.391 detik	0.391 detik
19	Gagal	Gagal	Gagal	Gagal
20	Gagal	Gagal	Gagal	Gagal
21	Gagal	Gagal	Gagal	Gagal
22	0.773 detik	0.773 detik	0.773 detik	0.773 detik
23	Gagal	Gagal	Gagal	Gagal
24	Gagal	Gagal	Gagal	Gagal
25	Gagal	Gagal	Gagal	Gagal
26	Gagal	Gagal	Gagal	Gagal

Tabel B.13: Hasil pengujian algoritma *hybrid genetic* untuk Calcudoku dengan ukuran *grid* 5×5 (Skenario 13-16)

Nomor Soal	Waktu Skenario 13	Waktu Skenario 14	Waktu Skenario 15	Waktu Skenario 16
1	Gagal	Gagal	Gagal	Gagal
2	Gagal	Gagal	Gagal	Gagal
3	0.043 detik	0.043 detik	0.043 detik	0.043 detik
4	Gagal	Gagal	Gagal	Gagal
5	Gagal	Gagal	Gagal	Gagal
6	Gagal	Gagal	Gagal	Gagal
7	Gagal	Gagal	Gagal	Gagal
8	Gagal	Gagal	Gagal	Gagal
9	Gagal	Gagal	Gagal	Gagal
10	Gagal	Gagal	Gagal	Gagal
11	Gagal	Gagal	Gagal	Gagal
12	Gagal	Gagal	Gagal	Gagal
13	0.077 detik	0.077 detik	0.077 detik	0.077 detik
14	Gagal	Gagal	Gagal	Gagal
15	Gagal	Gagal	Gagal	Gagal
16	Gagal	Gagal	Gagal	Gagal
17	Gagal	Gagal	Gagal	Gagal
18	0.043 detik	0.043 detik	0.043 detik	0.043 detik
19	Gagal	Gagal	Gagal	Gagal
20	Gagal	Gagal	Gagal	Gagal
21	Gagal	Gagal	Gagal	Gagal
22	0.146 detik	0.146 detik	0.146 detik	0.146 detik
23	Gagal	Gagal	Gagal	Gagal
24	Gagal	Gagal	Gagal	Gagal
25	Gagal	Gagal	Gagal	Gagal
26	Gagal	Gagal	Gagal	Gagal

1 **LAMPIRAN C**

2 **FILE TEKS SOAL-SOAL PERMAINAN CALCUDOKU**
3 **UNTUK PENGUJIAN**

4 Lampiran ini berisi *file* teks dari soal-soal permainan Calcudoku yang dipakai dalam pe-
5 ngujian ini. Soal-soal ini diambil dari sumber-sumber berikut:

- 6 1. <https://iota.math.msu.edu/k12-outreach/kenken-puzzles/>
7 2. <http://thinkmath.edc.org/resource/kenken-puzzles>

8 **C.1 File Teks Soal-Soal Permainan Calcudoku dengan Grid
9 Berukuran 4×4**

Listing C.1: 4x4_1.txt

```
10 | 4 7
11 | 1 2 2 3
12 | 1 4 5 3
13 | 6 4 5 5
14 | 6 7 7 7
15 | 1-
16 | 1-
17 | 3-
18 | 2/
19 | 24*
20 | 3-
21 | 6+
```

Listing C.2: 4x4_2.txt

```
22 | 4 8
23 | 1 2 2 3
24 | 4 4 5 3
25 | 6 4 5 5
26 | 6 7 7 8
27 | 4=
28 | 2/
29 | 1-
30 | 6+
31 | 12*
32 | 1-
33 | 5+
34 | 2=
```

Listing C.3: 4x4_3.txt

```
35 | 4 6
36 | 1 1 2 2
37 | 1 3 3 3
38 | 1 4 4 5
39 | 6 6 5 5
40 | 24*
41 | 3-
42 | 9+
43 | 2-
44 | 6+
45 | 2/
```

Listing C.4: 4x4_4.txt

```

1 | 4 7
2 | 1 1 2 3
3 | 4 4 2 3
4 | 5 5 6 3
5 | 7 6 6 3
6 | 1-
7 | 2-
8 | 10+
9 | 2/
10 | 6*
11 | 7+
12 | 4=

```

Listing C.5: 4x4_5.txt

```

13 | 4 7
14 | 1 2 3 3
15 | 1 2 4 4
16 | 5 5 4 6
17 | 5 7 7 6
18 | 6*
19 | 3-
20 | 2/
21 | 48*
22 | 7+
23 | 1-
24 | 4+

```

Listing C.6: 4x4_6.txt

```

25 | 4 8
26 | 1 2 2 2
27 | 1 3 4 5
28 | 6 3 4 7
29 | 6 8 7 7
30 | 2/
31 | 24*
32 | 2-
33 | 2/
34 | 3=
35 | 1-
36 | 4+
37 | 4=

```

Listing C.7: 4x4_7.txt

```

38 | 4 7
39 | 1 1 2 2
40 | 1 3 3 4
41 | 5 6 6 7
42 | 5 5 7 7
43 | 8+
44 | 2-
45 | 3-
46 | 3=
47 | 9*
48 | 2/
49 | 10+

```

Listing C.8: 4x4_8.txt

```

50 | 4 7
51 | 1 1 2 2
52 | 1 3 4 5
53 | 6 3 4 5
54 | 6 7 7 7
55 | 8*
56 | 1-
57 | 7+
58 | 2/
59 | 2-
60 | 2-
61 | 8*

```

Listing C.9: 4x4_9.txt

```

62 | 4 8
63 | 1 2 2 3
64 | 1 4 4 3
65 | 1 5 6 7
66 | 8 5 7 7
67 | 24*
68 | 1-
69 | 3-
70 | 7+
71 | 2/
72 | 1=
73 | 18*
74 | 1=

```

Listing C.10: 4x4_10.txt

```
1 | 4 7  
2 | 1 2 2 3  
3 | 1 4 3 3  
4 | 5 4 6 6  
5 | 5 5 7 6  
6 | 2/  
7 | 1-  
8 | 4*  
9 | 2-  
10 | 8+  
11 | 9+  
12 | 2=
```

Listing C.11: 4x4_11.txt

```
13 | 4 7  
14 | 1 1 2 2  
15 | 3 4 2 5  
16 | 3 4 6 5  
17 | 7 7 7 7  
18 | 2-  
19 | 24*  
20 | 1-  
21 | 2/  
22 | 5+  
23 | 2=  
24 | 10+
```

Listing C.12: 4x4_12.txt

```
25 | 4 8  
26 | 1 1 2 2  
27 | 3 3 4 4  
28 | 5 4 4 6  
29 | 5 7 8 8  
30 | 2/  
31 | 4+  
32 | 1-  
33 | 12*  
34 | 1-  
35 | 4=  
36 | 1=  
37 | 1-
```

Listing C.13: 4x4_13.txt

```
38 | 4 8  
39 | 1 1 2 2  
40 | 3 3 4 5  
41 | 6 6 4 7  
42 | 8 6 4 7  
43 | 2-  
44 | 2-  
45 | 3+  
46 | 9+  
47 | 4=  
48 | 9*  
49 | 2/  
50 | 4=
```

Listing C.14: 4x4_14.txt

```
51 | 4 7  
52 | 1 2 2 3  
53 | 1 1 4 3  
54 | 5 5 4 6  
55 | 7 7 6 6  
56 | 24*  
57 | 2/  
58 | 2-  
59 | 3+  
60 | 4+  
61 | 9+  
62 | 3-
```

Listing C.15: 4x4_15.txt

```
63 | 4 7  
64 | 1 2 2 3  
65 | 1 4 5 3  
66 | 1 4 5 6  
67 | 7 7 7 6  
68 | 6+  
69 | 3-  
70 | 2/  
71 | 2/  
72 | 2-  
73 | 2-  
74 | 24*
```

Listing C.16: 4x4_16.txt

```

1 | 4 8
2 | 1 2 2 3
3 | 1 1 4 4
4 | 5 5 6 6
5 | 7 7 8 8
6 | 12*
7 | 4+
8 | 4=
9 | 5+
10 | 3-
11 | 5+
12 | 7+
13 | 2/

```

Listing C.17: 4x4_17.txt

```

14 | 4 7
15 | 1 1 2 3
16 | 4 4 2 3
17 | 5 6 7 3
18 | 5 6 7 7
19 | 7+
20 | 5+
21 | 6*
22 | 1-
23 | 2/
24 | 3-
25 | 9+

```

Listing C.18: 4x4_18.txt

```

26 | 4 8
27 | 1 1 2 3
28 | 1 4 5 3
29 | 6 4 5 7
30 | 6 8 8 7
31 | 8*
32 | 3=
33 | 2-
34 | 3-
35 | 2/
36 | 5+
37 | 2-
38 | 4+

```

Listing C.19: 4x4_19.txt

```

39 | 4 7
40 | 1 1 1 2
41 | 3 3 4 2
42 | 5 6 6 2
43 | 5 5 7 7
44 | 6+
45 | 12*
46 | 1-
47 | 2=
48 | 4*
49 | 2/
50 | 1-

```

Listing C.20: 4x4_20.txt

```

51 | 4 8
52 | 1 2 2 3
53 | 4 4 5 3
54 | 6 6 5 7
55 | 6 8 8 7
56 | 2=
57 | 12*
58 | 2-
59 | 3-
60 | 2/
61 | 7+
62 | 2/
63 | 3+

```

Listing C.21: 4x4_21.txt

```

64 | 4 8
65 | 1 2 3 3
66 | 1 2 4 5
67 | 1 6 5 5
68 | 7 7 8 8
69 | 8+
70 | 2/
71 | 1-
72 | 3=
73 | 8*
74 | 1=
75 | 1-
76 | 3-

```

Listing C.22: 4x4_22.txt

```
1 | 4 8
2 | 1 2 3 3
3 | 1 2 4 4
4 | 5 5 6 6
5 | 7 7 8 6
6 | 6*
7 | 7+
8 | 2/
9 | 3-
10 | 5+
11 | 9+
12 | 2/
13 | 3=
```

Listing C.23: 4x4_23.txt

```
14 | 4 7
15 | 1 1 2 2
16 | 3 3 4 5
17 | 6 6 4 5
18 | 6 7 7 7
19 | 7+
20 | 2/
21 | 2-
22 | 2/
23 | 7+
24 | 4*
25 | 12*
```

Listing C.24: 4x4_24.txt

```
26 | 4 8
27 | 1 1 2 2
28 | 1 3 4 5
29 | 6 3 7 5
30 | 6 8 7 5
31 | 6*
32 | 1-
33 | 1-
34 | 1=
35 | 6+
36 | 5+
37 | 2/
38 | 2=
```

Listing C.25: 4x4_25.txt

```
39 | 4 8
40 | 1 2 2 2
41 | 1 3 4 5
42 | 6 3 4 5
43 | 6 7 8 8
44 | 3-
45 | 24*
46 | 4+
47 | 2/
48 | 7+
49 | 1-
50 | 2=
51 | 5+
```

Listing C.26: 4x4_26.txt

```
52 | 4 7
53 | 1 1 2 2
54 | 3 1 4 4
55 | 3 5 4 4
56 | 6 6 6 7
57 | 6*
58 | 2/
59 | 1-
60 | 9+
61 | 4=
62 | 8*
63 | 3=
```

Listing C.27: 4x4_27.txt

```
64 | 4 8
65 | 1 2 3 3
66 | 1 2 2 4
67 | 5 5 6 7
68 | 8 8 6 7
69 | 5+
70 | 12*
71 | 2-
72 | 4=
73 | 2/
74 | 3-
75 | 1-
76 | 2-
```

Listing C.28: 4x4_28.txt

```

1 | 4 9
2 | 1 2 3 4
3 | 1 2 5 4
4 | 6 7 5 8
5 | 6 7 9 9
6 | 7+
7 | 1-
8 | 1=
9 | 2/
10 | 2/
11 | 2/
12 | 5+
13 | 3=
14 | 12*

```

Listing C.29: 4x4_29.txt

```

15 | 4 7
16 | 1 1 2 3
17 | 4 5 2 3
18 | 4 4 6 3
19 | 7 4 6 6
20 | 1-
21 | 2/
22 | 6+
23 | 11+
24 | 1=
25 | 12*
26 | 3=

```

Listing C.30: 4x4_30.txt

```

27 | 4 8
28 | 1 2 2 3
29 | 1 4 4 5
30 | 6 6 5 5
31 | 7 7 8 8
32 | 2-
33 | 1-
34 | 1=
35 | 4+
36 | 48*
37 | 3+
38 | 1-
39 | 2/

```

Listing C.31: 4x4_31.txt

```

40 | 4 7
41 | 1 1 1 2
42 | 3 4 4 2
43 | 3 5 5 6
44 | 7 7 6 6
45 | 12*
46 | 2/
47 | 1-
48 | 2/
49 | 3-
50 | 10+
51 | 1-

```

Listing C.32: 4x4_32.txt

```

52 | 4 8
53 | 1 2 2 3
54 | 1 4 4 3
55 | 5 6 6 7
56 | 5 8 6 7
57 | 3-
58 | 2/
59 | 7+
60 | 6*
61 | 1-
62 | 9+
63 | 1-
64 | 3=

```

Listing C.33: 4x4_33.txt

```

65 | 4 7
66 | 1 1 2 2
67 | 3 4 4 5
68 | 3 6 6 5
69 | 3 6 7 7
70 | 2/
71 | 1-
72 | 24*
73 | 2/
74 | 3-
75 | 7+
76 | 2/

```

Listing C.34: 4x4_34.txt

```
1 | 4 8  
2 | 1 1 2 3  
3 | 4 4 2 3  
4 | 5 6 6 7  
5 | 5 5 8 8  
6 | 3+  
7 | 3-  
8 | 1-  
9 | 1-  
10 | 48*  
11 | 2-  
12 | 2=  
13 | 2/
```

Listing C.35: 4x4_35.txt

```
14 | 4 9  
15 | 1 2 3 3  
16 | 1 4 5 5  
17 | 1 4 6 7  
18 | 8 9 9 7  
19 | 12*  
20 | 1=  
21 | 1-  
22 | 1-  
23 | 2/  
24 | 1=  
25 | 5+  
26 | 2=  
27 | 7+
```

Listing C.36: 4x4_36.txt

```
28 | 4 7  
29 | 1 2 2 3  
30 | 1 1 3 3  
31 | 4 5 3 6  
32 | 4 5 7 7  
33 | 2*  
34 | 7+  
35 | 10+  
36 | 1-  
37 | 5+  
38 | 4=  
39 | 2/
```

Listing C.37: 4x4_37.txt

```
40 | 4 8  
41 | 1 1 2 3  
42 | 4 2 2 3  
43 | 5 5 6 3  
44 | 7 7 6 8  
45 | 2/  
46 | 6*  
47 | 9+  
48 | 1=  
49 | 2-  
50 | 2/  
51 | 1-  
52 | 1=
```

Listing C.38: 4x4_38.txt

```
53 | 4 8  
54 | 1 2 3 3  
55 | 1 2 4 5  
56 | 6 6 4 5  
57 | 7 7 8 5  
58 | 2-  
59 | 1-  
60 | 2/  
61 | 3-  
62 | 6*  
63 | 3+  
64 | 2/  
65 | 3=
```

Listing C.39: 4x4_39.txt

```
66 | 4 7  
67 | 1 1 1 2  
68 | 3 3 4 2  
69 | 5 3 6 6  
70 | 5 7 7 7  
71 | 6*  
72 | 2/  
73 | 9+  
74 | 1=  
75 | 2/  
76 | 2-  
77 | 8+
```

C.2 File Teks Soal-Soal Permainan Calcudoku dengan Grid Berukuran 5×5

Listing C.40: 5x5_1.txt

```

3 | 5 12
4 | 1 1 2 2 3
5 | 4 4 5 2 3
6 | 4 6 6 7 8
7 | 4 9 9 7 10
8 | 11 12 12 10 10
9 | 4-
10 | 10+
11 | 2/
12 | 120*
13 | 5=
14 | 2-
15 | 3+
16 | 5=
17 | 3-
18 | 12+
19 | 2=
20 | 2-

```

Listing C.41: 5x5_2.txt

```

21 | 5 10
22 | 1 1 1 2 2
23 | 3 3 4 2 5
24 | 6 6 4 2 5
25 | 7 8 4 9 9
26 | 7 8 8 10 10
27 | 6+
28 | 40*
29 | 3-
30 | 60*
31 | 1-
32 | 2-
33 | 2/
34 | 10*
35 | 4-
36 | 1-

```

Listing C.42: 5x5_3.txt

```

37 | 5 12
38 | 1 1 1 2 3
39 | 4 5 5 5 3
40 | 6 6 7 7 8
41 | 6 9 9 10 11
42 | 6 12 12 10 11
43 | 30*
44 | 1=
45 | 9+
46 | 4=
47 | 6+
48 | 24*
49 | 2-
50 | 2=
51 | 4-
52 | 20*
53 | 2-
54 | 2/

```

Listing C.43: 5x5_4.txt

```

55 | 5 13
56 | 1 1 2 2 3
57 | 4 4 5 6 3
58 | 7 7 5 6 8
59 | 9 10 11 11 8
60 | 9 10 12 12 13
61 | 12*
62 | 2/
63 | 1-
64 | 4-
65 | 2/
66 | 2-
67 | 2-
68 | 2/
69 | 2/
70 | 7+
71 | 1-
72 | 4-
73 | 3=

```

Listing C.44: 5x5_5.txt

```

74 | 5 12
75 | 1 2 2 3 3
76 | 1 2 4 4 5

```

```

1 | 6 7 7 8 9
2 | 6 10 11 8 9
3 | 10 10 11 12 12
4 | 2/
5 | 60*
6 | 3-
7 | 3-
8 | 3=
9 | 1-
10 | 3+
11 | 1-
12 | 4-
13 | 15*
14 | 2/
15 | 1-

```

Listing C.45: 5x5_6.txt

```

16 | 5 11
17 | 1 1 2 2 3
18 | 4 4 5 6 3
19 | 7 8 5 6 6
20 | 7 8 9 9 10
21 | 11 9 9 10 10
22 | 2-
23 | 1-
24 | 7+
25 | 3-
26 | 4-
27 | 40*
28 | 2/
29 | 2-
30 | 120*
31 | 6*
32 | 5=

```

Listing C.46: 5x5_7.txt

```

33 | 5 10
34 | 1 1 2 2 3
35 | 4 5 5 3 3
36 | 4 5 6 6 6
37 | 7 7 6 8 8
38 | 9 9 10 10 8
39 | 2/
40 | 2-
41 | 5*
42 | 9+
43 | 8+
44 | 11+
45 | 4-
46 | 24*
47 | 4+
48 | 3-

```

Listing C.47: 5x5_8.txt

```

49 | 5 11
50 | 1 1 2 3 4
51 | 5 6 7 3 4
52 | 5 6 7 8 8
53 | 5 9 9 8 10
54 | 5 11 11 11 10
55 | 2/
56 | 5=
57 | 2-
58 | 3-
59 | 30*
60 | 4-
61 | 7+
62 | 40*
63 | 2-
64 | 5+
65 | 11+

```

Listing C.48: 5x5_9.txt

```

66 | 5 12
67 | 1 2 3 3 4
68 | 1 2 2 3 4
69 | 5 5 6 7 8
70 | 9 10 10 7 8
71 | 9 11 11 12 12
72 | 4-
73 | 12+
74 | 6*
75 | 2-
76 | 2-
77 | 2=
78 | 1-
79 | 4-
80 | 2/
81 | 3-
82 | 3+
83 | 2-

```

Listing C.49: 5x5_10.txt

```

1 | 5 11
2 | 1 1 2 2 3
3 | 1 4 4 4 3
4 | 5 5 6 7 3
5 | 8 9 6 7 10
6 | 8 9 11 11 10
7 | 100*
8 | 3+
9 | 30*
10 | 12*
11 | 1-
12 | 3-
13 | 2-
14 | 2/
15 | 1-
16 | 3-
17 | 9+

```

Listing C.50: 5x5_11.txt

```

18 | 5 10
19 | 1 1 2 3 4
20 | 5 1 2 3 4
21 | 5 6 7 7 7
22 | 5 6 6 8 9
23 | 10 10 8 8 9
24 | 45*
25 | 4-
26 | 2/
27 | 3+
28 | 20*
29 | 9+
30 | 30*
31 | 8+
32 | 1-
33 | 3+

```

Listing C.51: 5x5_12.txt

```

34 | 5 12
35 | 1 1 2 2 3
36 | 4 4 5 5 3
37 | 6 7 7 8 8
38 | 6 9 7 10 11
39 | 9 9 12 12 11
40 | 6+
41 | 1-
42 | 2/
43 | 3-
44 | 2-
45 | 2/
46 | 12*
47 | 4-
48 | 10+
49 | 5=
50 | 4+
51 | 2/

```

Listing C.52: 5x5_13.txt

```

52 | 5 14
53 | 1 2 2 3 4
54 | 1 5 6 6 4
55 | 7 7 8 8 9
56 | 10 10 11 11 9
57 | 12 12 13 14 14
58 | 2-
59 | 1-
60 | 4=
61 | 6+
62 | 2=
63 | 3-
64 | 1-
65 | 4+
66 | 2/
67 | 3+
68 | 15*
69 | 3-
70 | 5=
71 | 1-

```

Listing C.53: 5x5_14.txt

```

72 | 5 12
73 | 1 1 2 2 3
74 | 1 4 5 6 3
75 | 7 4 5 6 6
76 | 7 8 9 9 9
77 | 10 10 11 12 12
78 | 15*
79 | 2/
80 | 1-
81 | 2/
82 | 2/

```

1	45*
2	3+
3	5=
4	8+
5	7+
6	5=
7	1-

Listing C.54: 5x5_15.txt

8	5 11
9	1 2 3 3 4
10	1 2 5 5 4
11	6 7 7 5 4
12	6 8 9 9 10
13	8 8 11 11 10
14	4-
15	3-
16	2/
17	30*
18	9+
19	1-
20	6*
21	20*
22	4-
23	5+
24	1-

Listing C.55: 5x5_16.txt

25	5 10
26	1 1 2 2 3
27	4 4 2 5 3
28	6 4 4 5 3
29	6 7 8 8 9
30	7 7 10 9 9
31	1-
32	24*
33	6+
34	75*
35	5+
36	2/
37	24*
38	3-
39	60*
40	1=

Listing C.56: 5x5_17.txt

41	5 11
42	1 1 2 3 4
43	5 6 2 3 4
44	5 6 2 7 8
45	9 6 10 7 8
46	9 9 10 11 11
47	5+
48	60*
49	6*
50	6+
51	2/
52	6+
53	4-
54	1-
55	75*
56	2/
57	2/

Listing C.57: 5x5_18.txt

58	5 13
59	1 1 2 2 3
60	4 4 5 5 3
61	6 4 7 8 9
62	10 11 7 8 9
63	10 12 12 13 13
64	1-
65	2/
66	6*
67	8+
68	12*
69	1=
70	4-
71	1-
72	1-
73	1-
74	4=
75	1-
76	4-

Listing C.58: 5x5_19.txt

77	5 10
78	1 1 2 3 3
79	4 5 2 2 6

1	4	7	7	2	6
2	4	7	8	9	9
3	10	10	10	9	9
4	2/				
5	15*				
6	2-				
7	12+				
8	2=				
9	3+				
10	12*				
11	2=				
12	15+				
13	15*				

Listing C.59: 5x5_20.txt

14	5	12			
15	1	1	2	2	3
16	4	5	5	6	3
17	4	7	7	6	8
18	4	9	8	8	8
19	10	10	11	11	12
20	3-				
21	5+				
22	1-				
23	12+				
24	15*				
25	2-				
26	2/				
27	40*				
28	1=				
29	5+				
30	1-				
31	5=				

Listing C.60: 5x5_21.txt

32	5	13			
33	1	2	2	3	4
34	5	5	6	3	4
35	5	7	6	8	9
36	10	7	11	8	9
37	10	12	12	13	9
38	4=				
39	4-				
40	1-				
41	2/				
42	12*				
43	2-				
44	3-				
45	3+				
46	12+				
47	4-				
48	4=				
49	1-				
50	5=				

Listing C.61: 5x5_22.txt

51	5	13			
52	1	2	3	4	4
53	1	5	3	6	6
54	7	5	3	8	8
55	9	10	10	11	12
56	9	13	13	11	12
57	20*				
58	3=				
59	8*				
60	2/				
61	10*				
62	4+				
63	3=				
64	1-				
65	1-				
66	9+				
67	10*				
68	1-				
69	2-				

Listing C.62: 5x5_23.txt

70	5	11			
71	1	2	3	3	4
72	1	2	2	5	5
73	1	2	6	6	7
74	8	8	9	9	7
75	10	10	10	11	11
76	15*				
77	96*				
78	7+				
79	3=				
80	4-				
81	2/				
82	2/				
83	3-				
84	2-				
85	10+				
86	5+				

Listing C.63: 5x5_24.txt

```

1 | 5 13
2 | 1 1 2 2 3
3 | 4 5 5 2 3
4 | 4 6 6 7 8
5 | 9 10 10 7 8
6 | 11 11 12 12 13
7 | 2-
8 | 24*
9 | 5+
10 | 4-
11 | 2/
12 | 2-
13 | 9+
14 | 1-
15 | 2=
16 | 3-
17 | 2/
18 | 4+
19 | 5=

```

Listing C.64: 5x5_25.txt

```

20 | 5 11
21 | 1 1 1 2 3
22 | 4 5 5 3 3
23 | 4 4 6 7 7
24 | 8 8 6 9 7
25 | 8 10 10 9 11
26 | 8*
27 | 5-
28 | 11+
29 | 40*
30 | 3+
31 | 2-
32 | 8*
33 | 9+
34 | 2/
35 | 12*
36 | 1=

```

Listing C.65: 5x5_26.txt

```

37 | 5 10
38 | 1 2 2 2 3
39 | 1 4 4 3 3
40 | 5 5 6 7 3
41 | 8 9 6 6 10
42 | 8 9 6 10 10
43 | 2/
44 | 20*
45 | 9+
46 | 7+
47 | 2-
48 | 6*
49 | 4=
50 | 4-
51 | 7+
52 | 11+

```

¹ **C.3 File Teks Soal-Soal Permainan Calcudoku dengan Grid**
² **Berukuran 6×6**

Listing C.66: 6x6_1.txt

```

3 | 6 15
4 | 1 1 2 3 4 4
5 | 1 5 2 3 6 4
6 | 5 5 7 7 6 8
7 | 9 9 10 10 6 11
8 | 12 12 13 10 11 11
9 | 12 13 13 14 15 15
10 | 30*
11 | 3+
12 | 11+
13 | 16*
14 | 13+
15 | 36*
16 | 1-
17 | 3=
18 | 2/
19 | 9*
20 | 50*
21 | 8+
22 | 15+
23 | 2=
24 | 5-

```

Listing C.67: 6x6_2.txt

```

25 | 6 19
26 | 1 1 2 3 4 4
27 | 5 6 6 3 7 4
28 | 5 8 9 9 10 11
29 | 12 12 13 14 10 11
30 | 15 15 13 14 14 16
31 | 17 18 18 19 19 16
32 | 2/
33 | 5=
34 | 2-
35 | 6*
36 | 8+
37 | 3/
38 | 4=
39 | 4=
40 | 3/
41 | 3/
42 | 3-
43 | 2/
44 | 5-
45 | 24*
46 | 4-
47 | 12*
48 | 2=
49 | 5+
50 | 11+

```

Listing C.68: 6x6_3.txt

```

51 | 6 19
52 | 1 1 2 3 4 5
53 | 6 6 2 3 4 7
54 | 8 9 10 10 10 7
55 | 8 11 11 12 12 13
56 | 14 15 16 17 17 13
57 | 15 15 16 18 19 19
58 | 2/
59 | 2-
60 | 1-
61 | 1-
62 | 5=
63 | 2/
64 | 3+
65 | 5-
66 | 4=
67 | 15*
68 | 3-
69 | 2/
70 | 1-
71 | 4=
72 | 75*
73 | 5-
74 | 2/
75 | 4=
76 | 3/

```

Listing C.69: 6x6_4.txt

```

77 | 6 18
78 | 1 2 3 3 4 5
79 | 1 6 7 3 4 5
80 | 8 6 7 9 10 11
81 | 8 12 12 9 11 11

```

```

1 | 13 14 14 15 15 15
2 | 16 16 14 17 17 18
3 | 11+
4 | 6=
5 | 4*
6 | 12*
7 | 8+
8 | 1-
9 | 3-
10 | 1-
11 | 11+
12 | 2=
13 | 4*
14 | 2/
15 | 4=
16 | 9+
17 | 36*
18 | 2/
19 | 1-
20 | 6=

```

Listing C.70: 6x6_5.txt

```

21 | 6 18
22 | 1 2 2 3 3 3
23 | 4 2 5 6 7 3
24 | 4 8 9 6 7 10
25 | 11 8 9 12 13 10
26 | 11 14 15 12 13 13
27 | 16 16 15 17 17 18
28 | 5=
29 | 4*
30 | 216*
31 | 3+
32 | 4=
33 | 6+
34 | 2-
35 | 1-
36 | 15*
37 | 3/
38 | 2/
39 | 7+
40 | 8+
41 | 6=
42 | 5-
43 | 1-
44 | 7+
45 | 1=

```

Listing C.71: 6x6_6.txt

```

46 | 6 18
47 | 1 1 2 2 3 4
48 | 5 1 6 2 7 4
49 | 5 8 6 9 7 10
50 | 11 8 12 12 13 10
51 | 14 14 14 12 13 15
52 | 16 16 17 18 18 15
53 | 24*
54 | 2*
55 | 5=
56 | 11+
57 | 3/
58 | 7+
59 | 2-
60 | 5-
61 | 5=
62 | 2/
63 | 6=
64 | 12+
65 | 1-
66 | 13+
67 | 3/
68 | 20*
69 | 2=
70 | 5-

```

Listing C.72: 6x6_7.txt

```

71 | 6 19
72 | 1 2 2 2 3 3
73 | 4 5 5 6 7 3
74 | 4 8 9 9 7 10
75 | 11 11 12 13 14 10
76 | 15 15 12 13 14 16
77 | 17 17 18 19 19 16
78 | 1=
79 | 72*
80 | 11+
81 | 20*
82 | 1-
83 | 1=
84 | 2/
85 | 6=
86 | 7+
87 | 3/
88 | 7+
89 | 6+

```

1	2/
2	9+
3	6*
4	3/
5	2-
6	4=
7	5-

Listing C.73: 6x6_8.txt

8	6 16
9	1 1 2 3 4 5
10	6 6 2 3 4 5
11	7 8 8 3 9 9
12	7 7 8 10 10 9
13	11 12 13 14 14 15
14	11 12 13 16 16 15
15	3/
16	9+
17	15+
18	7+
19	2/
20	1-
21	12+
22	3*
23	16+
24	1-
25	1-
26	30*
27	3/
28	1-
29	2-
30	2/

Listing C.74: 6x6_9.txt

31	6 16
32	1 1 2 2 3 3
33	1 1 4 5 6 7
34	8 9 4 5 6 10
35	8 9 9 11 12 10
36	13 14 14 11 12 10
37	13 13 15 15 16 16
38	14+
39	1-
40	2-
41	2-
42	1-
43	3/
44	1=
45	2/
46	4*
47	40*
48	3/
49	11+
50	80*
51	2-
52	5-
53	1-

Listing C.75: 6x6_10.txt

54	6 17
55	1 1 2 2 3 3
56	1 1 4 5 6 3
57	7 4 4 8 6 6
58	9 9 10 8 11 11
59	12 13 10 14 15 16
60	12 13 17 14 15 16
61	14+
62	30*
63	3*
64	60*
65	6=
66	12+
67	2=
68	2/
69	30*
70	3+
71	1-
72	5-
73	5+
74	1-
75	1-
76	3-
77	3=

Listing C.76: 6x6_11.txt

78	6 15
79	1 1 1 2 3 3
80	4 5 5 6 7 3
81	4 4 6 6 7 7
82	4 8 8 9 9 10
83	11 12 13 9 9 10
84	12 12 13 14 14 15

```

1 | 8+
2 | 3=
3 | 48*
4 | 13+
5 | 3/
6 | 36*
7 | 12+
8 | 5+
9 | 60*
10 | 1-
11 | 5=
12 | 17+
13 | 6*
14 | 7+
15 | 1=

```

Listing C.77: 6x6_12.txt

```

16 | 6 15
17 | 1 2 3 3 3 4
18 | 1 2 5 6 6 4
19 | 1 7 5 5 8 8
20 | 9 7 10 11 11 11
21 | 9 12 10 13 14 15
22 | 12 12 12 13 14 15
23 | 9+
24 | 2/
25 | 36*
26 | 1-
27 | 12+
28 | 3-
29 | 11+
30 | 6+
31 | 5-
32 | 3/
33 | 24*
34 | 18+
35 | 6+
36 | 3/
37 | 2/

```

Listing C.78: 6x6_13.txt

```

38 | 6 15
39 | 1 1 2 2 3 3
40 | 1 4 4 4 5 5
41 | 6 6 6 7 8 8
42 | 9 6 10 7 11 8
43 | 9 12 13 14 11 15
44 | 9 12 13 11 11 15
45 | 24*
46 | 2-
47 | 1-
48 | 15+
49 | 2-
50 | 13+
51 | 5-
52 | 48*
53 | 60*
54 | 2=
55 | 60*
56 | 3/
57 | 2-
58 | 5=
59 | 1-

```

Listing C.79: 6x6_14.txt

```

60 | 6 18
61 | 1 1 2 2 3 4
62 | 5 6 7 8 3 4
63 | 5 6 7 8 9 10
64 | 11 12 12 13 9 10
65 | 11 14 15 15 16 17
66 | 11 14 18 18 16 17
67 | 2-
68 | 3/
69 | 2/
70 | 2-
71 | 5-
72 | 2/
73 | 15*
74 | 6+
75 | 2-
76 | 2-
77 | 24*
78 | 3-
79 | 3=
80 | 11+
81 | 5-
82 | 2-
83 | 2/
84 | 2-

```

Listing C.80: 6x6_15.txt

1	1	1	2	3	4	5
2	1	6	2	3	4	5
3	7	6	8	8	9	10
4	7	7	11	11	9	10
5	12	13	14	11	9	10
6	12	13	14	15	15	15
7	36*					
8	7+					
9	2-					
10	1-					
11	1-					
12	1-					
13	60*					
14	3/					
15	10*					
16	11+					
17	96*					
18	1-					
19	1-					
20	2-					
21	30*					

Listing C.81: 6x6_16.txt

22	6	15				
23	1	1	2	3	3	4
24	5	6	2	7	4	4
25	5	6	6	7	8	9
26	10	11	11	12	8	13
27	10	14	12	12	8	13
28	10	14	15	15	15	15
29	5+					
30	1-					
31	2/					
32	20*					
33	4-					
34	10+					
35	1-					
36	15+					
37	4=					
38	11+					
39	3-					
40	8+					
41	5-					
42	2-					
43	15+					

Listing C.82: 6x6_17.txt

44	6	16				
45	1	2	2	3	4	5
46	6	7	2	3	4	5
47	6	7	8	9	4	5
48	10	11	8	9	9	12
49	10	11	13	14	15	12
50	16	16	13	14	14	12
51	2=					
52	6*					
53	2/					
54	7+					
55	60*					
56	1-					
57	3-					
58	2/					
59	13+					
60	3/					
61	3-					
62	12*					
63	15*					
64	20*					
65	3=					
66	2-					

Listing C.83: 6x6_18.txt

67	6	16				
68	1	1	2	2	3	3
69	1	4	5	5	5	6
70	7	4	8	9	10	6
71	7	7	8	9	10	6
72	11	12	13	14	14	14
73	11	12	15	15	16	16
74	30*					
75	5-					
76	2/					
77	5-					
78	15+					
79	12+					
80	13+					
81	7+					
82	1-					
83	3/					
84	3-					
85	1-					
86	6=					
87	8+					
88	1-					
89	11+					

Listing C.84: 6x6_19.txt

```

1 | 6 15
2 | 1 2 3 3 4 4
3 | 1 2 5 5 6 7
4 | 1 5 5 6 6 7
5 | 8 8 8 9 9 9
6 | 10 11 11 12 12 13
7 | 10 10 14 15 15 13
8 | 12*
9 | 1-
10 | 2/
11 | 5-
12 | 17+
13 | 6*
14 | 4-
15 | 9+
16 | 60*
17 | 20*
18 | 2-
19 | 2-
20 | 1-
21 | 6=
22 | 3/

```

Listing C.85: 6x6_20.txt

```

23 | 6 16
24 | 1 1 2 2 3 4
25 | 5 6 7 8 4 4
26 | 5 6 7 8 9 9
27 | 5 10 11 11 11 12
28 | 13 10 14 14 12 12
29 | 13 15 15 16 16 16
30 | 11+
31 | 1-
32 | 4=
33 | 36*
34 | 10+
35 | 2/
36 | 10*
37 | 2/
38 | 7+
39 | 5-
40 | 10+
41 | 13+
42 | 1-
43 | 5+
44 | 2/
45 | 20*

```

Listing C.86: 6x6_21.txt

```

46 | 6 16
47 | 1 1 2 2 3 4
48 | 5 5 6 6 3 4
49 | 7 7 8 9 9 10
50 | 8 8 8 11 11 10
51 | 12 13 13 14 15 16
52 | 12 12 14 14 16 16
53 | 5-
54 | 1-
55 | 1-
56 | 2/
57 | 2/
58 | 3-
59 | 1-
60 | 48*
61 | 3+
62 | 2-
63 | 2/
64 | 15*
65 | 2/
66 | 10+
67 | 5=
68 | 48*

```

Listing C.87: 6x6_22.txt

```

69 | 6 17
70 | 1 1 2 2 3 3
71 | 4 5 5 2 6 6
72 | 4 7 8 9 9 10
73 | 7 7 8 11 11 10
74 | 12 12 13 13 14 14
75 | 15 16 16 17 17 17
76 | 2-
77 | 6*
78 | 8+
79 | 1-
80 | 3/
81 | 3-
82 | 12*
83 | 8+
84 | 10*
85 | 5-
86 | 2-
87 | 7+

```

1	5-
2	1-
3	1=
4	1-
5	60*

Listing C.88: 6x6_23.txt

6	6 13
7	1 2 3 3 3 3
8	1 2 4 4 5 5
9	6 7 7 4 5 5
10	6 7 8 8 8 9
11	10 11 11 12 8 9
12	10 13 11 12 8 9
13	2-
14	1-
15	48*
16	11+
17	72*
18	5-
19	15+
20	13+
21	13+
22	2/
23	10*
24	1-
25	6=

Listing C.89: 6x6_24.txt

26	6 17
27	1 2 2 3 4 5
28	1 6 6 3 4 7
29	8 9 10 10 10 7
30	8 11 12 13 13 7
31	14 11 12 15 15 16
32	14 11 17 17 15 16
33	6+
34	3-
35	3-
36	24*
37	2=
38	6*
39	15+
40	4-
41	5=
42	12*
43	11+
44	10*
45	6*
46	1-
47	13+
48	3/
49	4-

Listing C.90: 6x6_25.txt

50	6 17
51	1 1 2 3 4 5
52	6 6 2 3 4 5
53	6 7 8 9 9 9
54	7 7 8 10 11 11
55	12 13 14 10 15 15
56	12 13 14 16 16 17
57	20*
58	3-
59	3/
60	3/
61	1-
62	11+
63	10+
64	10*
65	60*
66	7+
67	1-
68	4-
69	2/
70	2/
71	1-
72	5-
73	2=

Listing C.91: 6x6_26.txt

74	6 16
75	1 1 2 2 3 3
76	1 1 4 4 5 5
77	6 7 4 8 8 9
78	6 7 10 8 11 9
79	12 12 10 11 11 13
80	14 15 15 11 16 16
81	72*
82	4-
83	3/
84	8+

1	1-
2	1-
3	6+
4	10+
5	1-
6	1-
7	36*
8	4-
9	1=
10	1=
11	3/
12	1-

Listing C.92: 6x6_27.txt

13	6 17
14	1 2 3 3 4 5
15	1 2 6 3 4 7
16	8 8 6 9 4 7
17	10 11 11 9 12 12
18	13 14 14 9 12 15
19	13 16 16 16 17 15
20	2/
21	1-
22	24*
23	9+
24	5=
25	15*
26	5+
27	1-
28	13+
29	1=
30	1-
31	24*
32	2/
33	5-
34	2/
35	6+
36	5=

Listing C.93: 6x6_28.txt

37	6 14
38	1 1 2 2 3 3
39	4 5 5 5 3 6
40	4 4 7 7 6 6
41	4 8 8 9 10 10
42	4 11 11 9 12 12
43	11 11 13 13 14 14
44	4-
45	1-
46	24*
47	21+
48	14+
49	7+
50	2/
51	2/
52	3/
53	9+
54	200*
55	2/
56	5-
57	1-

Listing C.94: 6x6_29.txt

58	6 16
59	1 2 3 4 4 5
60	1 1 3 6 6 5
61	7 8 8 6 9 10
62	7 11 12 12 9 13
63	11 11 14 14 9 13
64	11 15 15 16 16 16
65	6*
66	5=
67	1-
68	3/
69	1-
70	12+
71	2-
72	5-
73	6+
74	2=
75	15+
76	1-
77	2/
78	3-
79	3/
80	20*

Listing C.95: 6x6_30.txt

81	6 17
82	1 1 2 2 3 3
83	4 4 5 6 6 7
84	8 8 5 9 9 7

```

1 | 10 11 11 11 12 12
2 | 10 13 14 14 15 16
3 | 13 13 17 14 15 16
4 | 1-
5 | 1-
6 | 5-
7 | 5-
8 | 1-
9 | 15*
10 | 3-
11 | 3-
12 | 3/
13 | 6*
14 | 15+
15 | 2/
16 | 10+
17 | 6*
18 | 9+
19 | 1-
20 | 6=

```

Listing C.96: 6x6_31.txt

```

21 | 6 16
22 | 1 1 2 3 4 4
23 | 5 1 3 3 6 6
24 | 5 7 7 8 8 9
25 | 10 10 11 11 11 9
26 | 12 13 14 15 16 9
27 | 12 13 14 15 16 16
28 | 60*
29 | 4=
30 | 12+
31 | 1-
32 | 5-
33 | 12*
34 | 1-
35 | 10*
36 | 24*
37 | 3-
38 | 18*
39 | 1-
40 | 3/
41 | 3/
42 | 1-
43 | 10*

```

Listing C.97: 6x6_32.txt

```

44 | 6 16
45 | 1 2 3 4 5 5
46 | 1 2 3 3 5 6
47 | 7 8 8 9 10 6
48 | 7 11 11 9 10 6
49 | 12 12 13 13 14 14
50 | 15 15 13 16 16 14
51 | 2/
52 | 5-
53 | 13+
54 | 1=
55 | 30*
56 | 60*
57 | 5-
58 | 2/
59 | 1-
60 | 2/
61 | 8+
62 | 2-
63 | 4*
64 | 24*
65 | 1-
66 | 2/

```

Listing C.98: 6x6_33.txt

```

67 | 6 17
68 | 1 2 3 4 5 6
69 | 1 2 3 4 5 5
70 | 1 7 7 8 9 9
71 | 10 11 11 11 12 13
72 | 14 15 15 16 12 13
73 | 14 17 15 16 12 13
74 | 60*
75 | 5-
76 | 1-
77 | 3-
78 | 10+
79 | 2=
80 | 6*
81 | 6=
82 | 3-
83 | 6=
84 | 6+
85 | 30*
86 | 13+
87 | 2/
88 | 11+
89 | 2-
90 | 4=

```

Listing C.99: 6x6_34.txt

```

1 | 6 14
2 | 1 1 2 2 2 3
3 | 1 4 4 5 5 3
4 | 6 7 7 5 8 8
5 | 6 6 9 9 10 8
6 | 11 11 9 9 10 10
7 | 11 12 13 13 14 14
8 | 4*
9 | 15+
10 | 2-
11 | 7+
12 | 36*
13 | 90*
14 | 3-
15 | 12*
16 | 12+
17 | 11+
18 | 15+
19 | 3=
20 | 4-
21 | 2/

```

Listing C.100: 6x6_35.txt

```

22 | 6 15
23 | 1 2 2 3 3 4
24 | 1 5 5 5 4 4
25 | 6 6 6 7 8 8
26 | 9 9 7 7 10 8
27 | 11 12 13 13 10 14
28 | 11 15 15 15 14 14
29 | 7+
30 | 1-
31 | 1-
32 | 12*
33 | 60*
34 | 10+
35 | 14+
36 | 40*
37 | 2-
38 | 2/
39 | 7+
40 | 1=
41 | 1-
42 | 108*
43 | 20*

```

Listing C.101: 6x6_36.txt

```

44 | 6 15
45 | 1 2 2 3 4 4
46 | 1 5 6 3 7 8
47 | 1 5 6 9 7 8
48 | 10 5 6 9 9 9
49 | 10 11 12 13 14 15
50 | 10 11 12 13 14 14
51 | 18*
52 | 3/
53 | 1-
54 | 12*
55 | 6+
56 | 12+
57 | 5+
58 | 1-
59 | 12+
60 | 40*
61 | 20*
62 | 1-
63 | 2-
64 | 12*
65 | 6=

```

Listing C.102: 6x6_37.txt

```

66 | 6 15
67 | 1 2 3 3 4 4
68 | 1 2 3 5 6 6
69 | 1 7 7 5 6 8
70 | 9 9 9 10 10 8
71 | 11 11 12 12 10 13
72 | 14 14 15 15 13 13
73 | 20*
74 | 2/
75 | 12+
76 | 2/
77 | 1-
78 | 14+
79 | 5-
80 | 2/
81 | 30*
82 | 11+
83 | 1-
84 | 4-
85 | 14+
86 | 5-
87 | 2/

```

Listing C.103: 6x6_38.txt

```

1 | 6 14
2 | 1 1 2 2 3 4
3 | 5 1 6 2 4 4
4 | 5 7 6 8 8 8
5 | 7 7 6 9 9 9
6 | 10 10 11 11 12 9
7 | 13 13 14 14 12 12
8 | 72*
9 | 5+
10 | 2=
11 | 12+
12 | 2/
13 | 12+
14 | 11+
15 | 20*
16 | 16+
17 | 1-
18 | 1-
19 | 13+
20 | 4-
21 | 1-

```

Listing C.104: 6x6_39.txt

```

22 | 6 14
23 | 1 1 2 2 3 3
24 | 1 1 4 4 4 5
25 | 6 6 7 7 7 5
26 | 8 8 8 9 10 5
27 | 11 8 12 9 13 13
28 | 11 11 12 9 14 14
29 | 6*
30 | 3/
31 | 1-
32 | 15+
33 | 18*
34 | 1-
35 | 6+
36 | 240*
37 | 10+
38 | 3=
39 | 36*
40 | 1-
41 | 11+
42 | 2/

```

¹ C.4 **File Teks Soal-Soal Permainan Calcudoku dengan Grid**
² **Berukuran 7×7**

Listing C.105: 7x7_1.txt

```

3 | 7 21
4 | 1 1 1 2 3 3 3
5 | 4 5 6 2 7 7 7
6 | 4 5 6 8 9 9 7
7 | 4 10 10 8 9 11 12
8 | 13 14 10 15 16 11 12
9 | 13 14 10 15 17 17 18
10 | 19 19 20 20 17 21 18
11 | 210*
12 | 1-
13 | 7+
14 | 12+
15 | 3-
16 | 3/
17 | 17+
18 | 13+
19 | 11+
20 | 19+
21 | 1-
22 | 1-
23 | 15*
24 | 3+
25 | 5+
26 | 6-
27 | 15+
28 | 11+
29 | 5-
30 | 3-
31 | 3=

```

Listing C.106: 7x7_2.txt

```

32 | 7 21
33 | 1 1 2 2 3 4 4
34 | 5 1 1 6 3 7 7
35 | 5 8 8 6 6 7 9
36 | 10 11 12 12 13 13 9
37 | 10 14 14 15 15 16 17
38 | 10 18 19 19 19 16 17
39 | 18 18 20 20 21 21 21
40 | 45*
41 | 2/
42 | 1-
43 | 3-
44 | 3/
45 | 10+
46 | 13+
47 | 1-
48 | 1-
49 | 28*
50 | 6=
51 | 3-
52 | 2-
53 | 3-
54 | 6-
55 | 2/
56 | 5-
57 | 14+
58 | 30*
59 | 2-
60 | 8+

```

Listing C.107: 7x7_3.txt

```

61 | 7 22
62 | 1 1 2 2 3 4 4
63 | 5 6 6 3 3 7 7
64 | 5 8 9 10 10 10 7
65 | 11 8 9 12 13 14 14
66 | 11 15 15 12 13 16 14
67 | 17 18 19 19 20 16 21
68 | 17 17 22 22 20 21 21
69 | 15*
70 | 5-
71 | 9+
72 | 7+
73 | 2/
74 | 2-
75 | 210*
76 | 6-
77 | 7+
78 | 15+
79 | 6+
80 | 1-
81 | 10+
82 | 6+
83 | 3/
84 | 1-
85 | 15+
86 | 4=

```

1	2-
2	3-
3	14+
4	5-

Listing C.108: 7x7_4.txt

```

5 7 21
6 1 2 2 2 3 3 4
7 1 5 2 6 6 4 4
8 7 5 8 9 9 10 10
9 7 8 8 11 12 10 13
10 14 14 14 11 12 15 13
11 16 16 17 11 18 19 19
12 20 20 17 17 18 21 21
13 1-
14 98*
15 2/
16 15+
17 1-
18 2/
19 8+
20 8+
21 9+
22 30*
23 60*
24 10+
25 1-
26 16+
27 6=
28 1-
29 14+
30 3-
31 3-
32 3/
33 6-

```

Listing C.109: 7x7_5.txt

```

34 7 20
35 1 2 2 3 4 4 4
36 1 2 3 3 5 6 7
37 8 8 9 3 5 6 7
38 10 10 9 11 12 13 13
39 14 14 9 11 12 15 15
40 16 17 18 18 19 19 15
41 17 17 17 18 20 20 20
42 1-
43 45*
44 12+
45 9+
46 1-
47 3-
48 9+
49 8+
50 12*
51 6+
52 2-
53 2/
54 2-
55 2/
56 11+
57 2=
58 20+
59 60*
60 21*
61 20*

```

Listing C.110: 7x7_6.txt

```

62 7 22
63 1 2 3 4 4 4 5
64 1 2 3 6 6 5
65 7 7 3 8 9 10 11
66 7 12 12 8 9 10 11
67 13 14 15 15 16 17 17
68 13 18 18 16 16 17 19
69 20 20 21 21 22 22 19
70 6*
71 1-
72 18+
73 12+
74 3-
75 9+
76 11+
77 1-
78 6-
79 9+
80 6+
81 3-
82 2-
83 7=
84 4-
85 60*
86 18*
87 3-
88 3/
89 2-

```

1	28*
2	3-

Listing C.111: 7x7_7.txt

```

3 | 7 21
4 | 1 2 2 3 3 4 4
5 | 1 5 6 6 3 4 4
6 | 7 5 8 9 9 10 11
7 | 7 12 8 13 13 10 11
8 | 7 12 14 13 15 16 16
9 | 17 17 14 15 15 18 16
10 | 19 19 14 20 21 21 21
11 | 6-
12 | 3-
13 | 14+
14 | 420*
15 | 1-
16 | 1-
17 | 12+
18 | 3-
19 | 42*
20 | 4-
21 | 3-
22 | 6-
23 | 10*
24 | 168*
25 | 10+
26 | 84*
27 | 1-
28 | 3=
29 | 3/
30 | 3=
31 | 10+

```

Listing C.112: 7x7_8.txt

```

32 | 7 21
33 | 1 2 2 3 4 5 5
34 | 1 1 3 3 4 4 6
35 | 7 7 8 8 9 4 6
36 | 10 11 11 12 9 13 13
37 | 10 14 15 12 12 13 16
38 | 17 14 15 18 18 16 16
39 | 17 19 19 18 20 20 21
40 | 14+
41 | 2/
42 | 105*
43 | 30*
44 | 1-
45 | 6-
46 | 30*
47 | 1-
48 | 3-
49 | 1-
50 | 11+
51 | 8+
52 | 12+
53 | 4-
54 | 3-
55 | 9+
56 | 6-
57 | 15+
58 | 5-
59 | 2/
60 | 4=

```

Listing C.113: 7x7_9.txt

```

61 | 7 20
62 | 1 2 3 3 4 5 5
63 | 1 2 6 6 4 7 7
64 | 8 2 9 10 10 7 11
65 | 8 8 9 9 12 11 11
66 | 13 13 13 9 12 14 15
67 | 16 17 17 18 18 14 14
68 | 16 19 19 20 20 20 14
69 | 6+
70 | 84*
71 | 4-
72 | 1-
73 | 3-
74 | 2/
75 | 17+
76 | 210*
77 | 13+
78 | 6-
79 | 24*
80 | 11+
81 | 9+
82 | 378*
83 | 4=
84 | 1-
85 | 6-
86 | 3-
87 | 3-
88 | 24*

```

Listing C.114: 7x7_10.txt

```

1 | 7 21
2 | 1 2 3 3 4 4 5
3 | 1 6 6 7 7 7 5
4 | 8 8 6 6 7 9 9
5 | 10 10 11 11 11 12 13
6 | 14 14 15 16 12 12 13
7 | 17 17 15 16 18 18 19
8 | 17 17 20 20 21 21 19
9 | 3/
10 | 7=
11 | 2/
12 | 1-
13 | 30*
14 | 1008*
15 | 14+
16 | 5+
17 | 6-
18 | 11+
19 | 11+
20 | 9+
21 | 1-
22 | 5-
23 | 1-
24 | 1-
25 | 14+
26 | 9+
27 | 1-
28 | 2-
29 | 2/

```

Listing C.115: 7x7_11.txt

```

30 | 7 21
31 | 1 2 2 3 4 4 5
32 | 1 6 7 3 4 4 5
33 | 8 6 7 9 9 10 10
34 | 8 11 12 12 13 13 10
35 | 8 11 14 12 12 15 16
36 | 17 17 14 18 12 15 16
37 | 19 19 20 18 21 21 16
38 | 6-
39 | 3-
40 | 1-
41 | 240*
42 | 6+
43 | 1-
44 | 1-
45 | 60*
46 | 6-
47 | 12+
48 | 4-
49 | 23+
50 | 3/
51 | 5+
52 | 9+
53 | 84*
54 | 3/
55 | 24*
56 | 2/
57 | 5=
58 | 4-

```

Listing C.116: 7x7_12.txt

```

59 | 7 22
60 | 1 2 3 3 4 4 5
61 | 1 2 6 6 4 7 5
62 | 8 8 9 9 10 7 11
63 | 12 13 14 10 10 10 11
64 | 12 13 14 15 16 17 18
65 | 19 19 15 15 16 17 18
66 | 19 20 20 21 21 22 22
67 | 5+
68 | 4-
69 | 13+
70 | 84*
71 | 3/
72 | 2-
73 | 3/
74 | 2-
75 | 3+
76 | 17+
77 | 8+
78 | 1-
79 | 3-
80 | 3/
81 | 11+
82 | 4-
83 | 4-
84 | 1-
85 | 36*
86 | 9+
87 | 10+
88 | 4-

```

Listing C.117: 7x7_13.txt

```

1 | 7 20
2 | 1 2 2 2 3 3 4
3 | 1 5 5 6 6 7 4
4 | 1 8 5 9 9 7 7
5 | 8 8 10 11 12 12 13
6 | 14 15 10 11 16 16 13
7 | 14 15 17 17 17 18 13
8 | 19 19 20 20 17 18 18
9 | 105*
10 | 18*
11 | 9+
12 | 3-
13 | 12+
14 | 2-
15 | 6+
16 | 15+
17 | 1-
18 | 6-
19 | 1-
20 | 3-
21 | 70*
22 | 2/
23 | 6-
24 | 2/
25 | 120*
26 | 17+
27 | 2/
28 | 4-

```

Listing C.118: 7x7_14.txt

```

29 | 7 21
30 | 1 2 3 4 4 4 5
31 | 1 2 2 2 6 5 5
32 | 7 8 9 6 6 6 10
33 | 7 8 9 9 11 12 10
34 | 13 8 14 14 11 11 15
35 | 13 13 16 17 17 18 19
36 | 20 20 16 21 18 18 19
37 | 2-
38 | 9+
39 | 2=
40 | 18+
41 | 63*
42 | 17+
43 | 5-
44 | 18+
45 | 90*
46 | 3-
47 | 14*
48 | 1=
49 | 60*
50 | 1-
51 | 2=
52 | 3-
53 | 6+
54 | 15+
55 | 1-
56 | 3/
57 | 2=

```

Listing C.119: 7x7_15.txt

```

58 | 7 18
59 | 1 1 2 2 3 3 3
60 | 1 1 4 2 3 5 6
61 | 7 7 8 8 5 5 6
62 | 9 7 7 10 10 10 10
63 | 9 11 12 12 13 13 10
64 | 11 11 12 14 15 16 17
65 | 11 18 18 14 15 16 17
66 | 16+
67 | 40*
68 | 126*
69 | 1=
70 | 28*
71 | 1-
72 | 12+
73 | 13+
74 | 6-
75 | 4200*
76 | 96*
77 | 36*
78 | 3/
79 | 8+
80 | 2-
81 | 3-
82 | 2/
83 | 42*

```

C.5 File Teks Soal-Soal Permainan Calcudoku dengan Grid Berukuran 8×8

Listing C.120: 8x8_1.txt

```

3 | 8 27
4 | 1 2 2 3 3 3 4 4
5 | 1 2 5 5 6 6 7 8
6 | 9 9 5 10 11 11 7 8
7 | 12 12 13 10 10 14 15 15
8 | 16 17 13 18 18 14 15 15
9 | 16 17 13 19 20 21 21 21
10 | 22 23 23 19 20 24 25 26
11 | 22 22 22 27 27 24 25 26
12 | 3/
13 | 20*
14 | 21+
15 | 1-
16 | 28*
17 | 1-
18 | 7-
19 | 2/
20 | 2-
21 | 15*
22 | 14*
23 | 28*
24 | 144*
25 | 2/
26 | 150*
27 | 2/
28 | 8+
29 | 2-
30 | 2-
31 | 4-
32 | 168*
33 | 26+
34 | 2-
35 | 7-
36 | 2/
37 | 5-
38 | 3-

```

Listing C.121: 8x8_2.txt

```

39 | 8 26
40 | 1 1 2 3 3 3 4 4
41 | 5 2 2 6 7 8 9 9
42 | 5 10 6 6 7 8 11 11
43 | 5 10 12 13 14 15 15 15
44 | 16 12 12 13 14 17 17 18
45 | 16 19 20 21 21 22 22 18
46 | 23 19 20 24 24 24 24 18
47 | 23 23 25 25 25 26 26 18
48 | 5-
49 | 60*
50 | 14+
51 | 1-
52 | 20*
53 | 13+
54 | 24*
55 | 2-
56 | 2/
57 | 6-
58 | 1-
59 | 17+
60 | 2/
61 | 14*
62 | 144*
63 | 5+
64 | 24*
65 | 280*
66 | 10+
67 | 5-
68 | 48*
69 | 2-
70 | 15+
71 | 10+
72 | 16+
73 | 3/

```

Listing C.122: 8x8_3.txt

```

74 | 8 25
75 | 1 1 1 2 3 4 5 6
76 | 7 7 8 2 3 4 5 6
77 | 9 9 8 10 10 11 11 12
78 | 13 9 14 14 14 14 15 12
79 | 13 16 16 14 17 17 15 12
80 | 13 18 19 19 17 20 20 12
81 | 21 18 22 22 22 20 23 23
82 | 21 21 22 24 24 24 24 25
83 | 210*
84 | 1-
85 | 3-
86 | 3/

```

```

1 8*
2 1-
3 1-
4 12*
5 42*
6 2/
7 6+
8 26+
9 13+
10 504*
11 2-
12 3-
13 112*
14 7-
15 3/
16 45*
17 11+
18 12+
19 6-
20 26+
21 4=

```

Listing C.123: 8x8_4.txt

```

22 8 30
23 1 1 2 3 3 4 4 5
24 6 7 2 8 9 9 10 5
25 6 7 8 8 11 11 10 12
26 13 13 14 14 15 15 16 12
27 17 18 18 19 20 16 16 12
28 17 21 22 19 20 23 23 24
29 25 21 22 19 26 26 27 24
30 25 28 28 29 29 30 27 27
31 12*
32 4-
33 7-
34 3-
35 3/
36 3-
37 5-
38 20+
39 3-
40 1-
41 11+
42 19+
43 1-
44 1-
45 7+
46 32*
47 11+
48 5-
49 13+
50 1-
51 13+
52 4/
53 9+
54 2-
55 1-
56 4-
57 3*
58 2/
59 14*
60 5=

```

Listing C.124: 8x8_5.txt

```

61 8 26
62 1 1 1 2 2 3 4 5
63 1 6 7 7 8 3 4 5
64 9 9 9 8 10 10 10 10
65 11 11 12 13 13 13 10 14
66 11 11 15 15 16 13 17 14
67 18 18 19 20 16 21 17 22
68 23 23 19 20 21 21 22 22
69 23 24 24 25 25 26 26 26
70 210*
71 7+
72 10*
73 2/
74 4/
75 1=
76 1-
77 10+
78 10+
79 1680*
80 25+
81 5=
82 216*
83 6-
84 2-
85 9+
86 1-
87 2/
88 9+
89 7-
90 20*
91 14+
92 56*
93 5-
94 1-
95 12+

```

Listing C.125: 8x8_6.txt

```

1 | 8 27
2 | 1 2 2 2 3 3 4 4
3 | 1 2 5 5 6 4 7
4 | 8 9 10 11 12 7 7 7
5 | 8 9 10 11 12 12 13 13
6 | 14 15 16 16 17 17 18 18
7 | 14 15 19 19 20 20 21 22
8 | 23 23 24 25 26 26 21 22
9 | 23 24 24 25 25 27 27 22
10 | 1-
11 | 480*
12 | 3+
13 | 56*
14 | 168*
15 | 6=
16 | 17+
17 | 6+
18 | 4/
19 | 2-
20 | 5+
21 | 20+
22 | 2/
23 | 15+
24 | 3/
25 | 2-
26 | 2/
27 | 2/
28 | 6*
29 | 2-
30 | 2-
31 | 96*
32 | 120*
33 | 42*
34 | 8*
35 | 1-
36 | 20*

```

Listing C.126: 8x8_7.txt

```

37 | 8 27
38 | 1 1 2 2 3 3 4 4
39 | 5 6 7 8 3 9 10 10
40 | 5 6 7 8 8 9 10 11
41 | 12 12 13 13 13 14 14 11
42 | 15 15 16 16 17 18 18 18
43 | 19 19 16 17 17 20 20 20
44 | 19 21 21 22 23 24 25 26
45 | 27 27 27 22 23 24 25 26
46 | 2/
47 | 2/
48 | 90*
49 | 4-
50 | 3+
51 | 1-
52 | 5-
53 | 280*
54 | 2-
55 | 96*
56 | 4-
57 | 8+
58 | 64*
59 | 1-
60 | 8+
61 | 160*
62 | 14+
63 | 11+
64 | 90*
65 | 16*
66 | 1-
67 | 3/
68 | 5-
69 | 40*
70 | 4-
71 | 11+
72 | 24*

```

Listing C.127: 8x8_8.txt

```

73 | 8 27
74 | 1 1 2 2 3 3 3 4
75 | 5 6 7 8 8 9 9 4
76 | 5 6 7 10 10 11 9 12
77 | 5 13 14 15 10 11 11 12
78 | 16 13 14 15 17 17 18 12
79 | 16 19 19 20 20 18 18 18
80 | 16 21 22 22 23 23 24 24
81 | 25 25 26 26 23 27 27 27
82 | 3/
83 | 1-
84 | 168*
85 | 7-
86 | 112*
87 | 14+
88 | 3/
89 | 8+
90 | 60*
91 | 168*
92 | 10*

```

```

1 | 15+
2 | 4-
3 | 4-
4 | 2/
5 | 15*
6 | 1-
7 | 25+
8 | 20*
9 | 2/
10 | 1=
11 | 5-
12 | 56*
13 | 3/
14 | 1-
15 | 56*
16 | 18*

```

Listing C.128: 8x8_9.txt

```

17 | 8 28
18 | 1 2 2 3 3 4 5 6
19 | 1 7 7 8 4 4 5 6
20 | 9 10 10 8 4 11 11 11
21 | 9 9 12 13 14 14 15 15
22 | 16 9 12 13 17 18 19 20
23 | 16 21 21 22 17 18 19 20
24 | 23 23 24 22 22 25 25 20
25 | 23 23 24 26 26 27 28 28
26 | 2-
27 | 2-
28 | 7-
29 | 120*
30 | 3/
31 | 3-
32 | 6-
33 | 5-
34 | 17+
35 | 3-
36 | 10+
37 | 4/
38 | 5-
39 | 4-
40 | 5-
41 | 2-
42 | 2/
43 | 42*
44 | 15*
45 | 14+
46 | 2-
47 | 14+
48 | 6*
49 | 1-
50 | 6-
51 | 30*
52 | 8=
53 | 12*

```

Listing C.129: 8x8_10.txt

```

54 | 8 28
55 | 1 1 2 3 3 4 5 6
56 | 7 1 2 2 4 4 5 8
57 | 7 9 2 10 11 11 12 8
58 | 13 14 10 10 15 15 12 8
59 | 13 14 16 17 17 18 12 19
60 | 20 21 16 16 22 18 23 19
61 | 20 21 24 25 22 22 23 19
62 | 26 26 24 25 27 27 28 28
63 | 144*
64 | 1960*
65 | 3/
66 | 10+
67 | 4/
68 | 4=
69 | 1-
70 | 48*
71 | 5=
72 | 16+
73 | 10+
74 | 21+
75 | 1-
76 | 1-
77 | 2/
78 | 14+
79 | 4-
80 | 11+
81 | 105*
82 | 3-
83 | 3-
84 | 17+
85 | 1-
86 | 1-
87 | 3-
88 | 1-
89 | 10+
90 | 7+

```

Listing C.130: 8x8_11.txt

```

1 | 8 27
2 | 1 2 2 3 3 4 4 5
3 | 1 6 2 7 7 8 8 5
4 | 1 6 6 9 7 10 10 11
5 | 12 12 9 9 13 14 15 11
6 | 16 17 18 19 13 14 15 11
7 | 16 17 18 19 19 20 21 22
8 | 23 17 24 25 20 20 21 22
9 | 23 23 24 25 26 26 27 27
10 | 30*
11 | 20+
12 | 2/
13 | 2/
14 | 6-
15 | 11+
16 | 12*
17 | 2-
18 | 16*
19 | 6-
20 | 18+
21 | 42*
22 | 1-
23 | 24*
24 | 2/
25 | 7-
26 | 11+
27 | 3/
28 | 8+
29 | 16+
30 | 5-
31 | 8+
32 | 96*
33 | 5-
34 | 1-
35 | 7-
36 | 7+

```

Listing C.131: 8x8_12.txt

```

37 | 8 28
38 | 1 1 1 2 3 4 4 5
39 | 6 7 7 2 3 8 5 5
40 | 6 9 10 2 11 11 12 13
41 | 6 9 10 14 14 12 12 13
42 | 15 16 17 14 18 19 19 20
43 | 15 16 16 21 18 19 22 20
44 | 23 24 25 21 18 26 22 27
45 | 23 24 25 26 26 26 28 27
46 | 28*
47 | 17+
48 | 2-
49 | 1-
50 | 14+
51 | 24*
52 | 4/
53 | 6-
54 | 1-
55 | 2/
56 | 7-
57 | 20+
58 | 1-
59 | 24*
60 | 5-
61 | 14+
62 | 7=
63 | 6+
64 | 12+
65 | 56*
66 | 7+
67 | 7-
68 | 10*
69 | 5-
70 | 2/
71 | 25+
72 | 3-
73 | 4=

```

Listing C.132: 8x8_13.txt

```

74 | 8 27
75 | 1 2 3 4 5 6 6 7
76 | 1 3 3 4 5 5 8 7
77 | 1 9 10 10 11 12 8 8
78 | 13 9 9 11 11 12 14 14
79 | 13 15 15 16 17 18 18 18
80 | 19 20 16 16 17 21 22 22
81 | 19 20 20 23 23 21 22 24
82 | 25 25 25 26 27 27 27 24
83 | 210*
84 | 3=
85 | 17+
86 | 3-
87 | 32*
88 | 5-
89 | 4/
90 | 9+
91 | 192*
92 | 1-
93 | 42*
94 | 10*

```

1	2/
2	3/
3	7-
4	6*
5	2-
6	210*
7	5-
8	11+
9	4-
10	14+
11	3-
12	1-
13	6+
14	6=
15	17+

¹ **C.6 File Teks Soal-Soal Permainan Calcudoku dengan Grid**
² **Berukuran 9×9**

Listing C.133: 9x9_1.txt

```

3 9 34
4 1 2 2 3 4 5 5 6 6
5 1 7 2 3 4 8 8 8 9
6 7 7 10 3 11 12 12 13 9
7 14 14 10 3 11 15 13 13 9
8 16 16 17 3 15 15 18 18 18
9 19 16 17 20 20 20 21 22 23
10 24 24 25 25 26 27 21 22 23
11 24 28 28 29 30 27 31 31 23
12 32 32 28 29 30 33 33 34 34
13 6-
14 128*
15 15+
16 4/
17 3/
18 1-
19 14+
20 19+
21 19+
22 1-
23 2-
24 8-
25 56*
26 24*
27 40*
28 378*
29 3/
30 56*
31 4=
32 432*
33 40*
34 2/
35 14+
36 15+
37 2-
38 6=
39 21*
40 72*
41 1-
42 5-
43 6+
44 6+
45 3/
46 7+
```

Listing C.134: 9x9_2.txt

```

47 9 36
48 1 2 3 3 4 5 5 6 7
49 1 2 2 4 4 8 8 6 7
50 9 10 10 11 11 11 12 12 12
51 9 13 10 14 14 14 15 16 17 17
52 9 13 18 18 14 19 16 20 20
53 21 21 21 21 19 19 22 23 23
54 24 24 25 25 26 27 27 28 28
55 29 29 30 26 26 31 32 32 33
56 34 34 30 35 35 31 36 36 33
57 28*
58 18+
59 1-
60 576*
61 2-
62 2/
63 5-
64 2/
65 6+
66 180*
67 63*
68 14+
69 1-
70 12+
71 8=
72 7-
73 4-
74 5-
75 40*
76 15+
77 23+
78 3=
79 13+
80 4-
81 4/
82 140*
83 1-
84 3/
85 2-
86 4-
87 7-
88 8-
89 9+
90 4/
91 3/
92 2-
```

Listing C.135: 9x9_3.txt

```

1 | 9 34
2 | 1 1 1 1 2 2 3 3 4
3 | 5 6 7 7 8 8 3 9 4
4 | 5 6 10 10 11 12 13 9 9
5 | 5 14 10 15 11 11 13 13 16
6 | 17 14 18 15 19 19 20 20 16
7 | 17 18 18 21 19 22 22 23 23
8 | 24 24 25 21 26 26 27 27 28
9 | 29 25 25 30 30 31 32 27 28
10 | 29 29 33 33 33 31 32 34 28
11 | 30+
12 | 1-
13 | 30*
14 | 5-
15 | 90*
16 | 3-
17 | 5-
18 | 4/
19 | 13+
20 | 14+
21 | 21*
22 | 6=
23 | 16+
24 | 2-
25 | 2-
26 | 13+
27 | 10+
28 | 4+
29 | 60*
30 | 1-
31 | 1-
32 | 11+
33 | 11+
34 | 1-
35 | 24*
36 | 8-
37 | 120*
38 | 14+
39 | 24*
40 | 5-
41 | 14+
42 | 8-
43 | 12+
44 | 7=

```

Listing C.136: 9x9_4.txt

```

45 | 9 32
46 | 1 1 1 2 2 2 3
47 | 4 5 5 1 6 6 7 7 3
48 | 4 4 8 8 9 10 11 12 12
49 | 4 13 13 8 9 10 11 14 15
50 | 4 16 17 18 10 10 11 15 15
51 | 19 16 17 18 20 20 20 21 22
52 | 19 23 23 18 24 25 25 21 22
53 | 26 27 28 18 24 25 25 29 29
54 | 26 27 28 30 30 31 31 32 32
55 | 3024*
56 | 19+
57 | 28*
58 | 720*
59 | 2/
60 | 3-
61 | 1-
62 | 245*
63 | 5-
64 | 24+
65 | 16*
66 | 24*
67 | 20*
68 | 9=
69 | 17+
70 | 1-
71 | 8-
72 | 10+
73 | 63*
74 | 120*
75 | 3/
76 | 2/
77 | 3-
78 | 16+
79 | 336*
80 | 6-
81 | 3-
82 | 1-
83 | 4-
84 | 14+
85 | 4-
86 | 16+

```


1

LAMPIRAN D

2

KODE PROGRAM

3 Lampiran ini berisi kode program dari perangkat lunak Calcudoku ini.

4 **D.1 Grid.java**

Listing D.1: Grid.java

```

5  package model;
6
7  import java.util.ArrayList;
8  import java.util.Objects;
9  import javax.swing.JOptionPane;
10
11 public class Grid
12 {
13
14     private final Integer size;
15     private final Integer numberOfCages;
16     private final Integer[][] cageCells;
17     private final String[] cageObjectives;
18     private final Cell[][] grid;
19     private final Cage[] cages;
20
21     public Grid(Integer size, Integer numberOfCages, Integer[][] cageCells,
22                 String[] cageObjectives)
23     {
24         this.size = size;
25         this.numberOfCages = numberOfCages;
26         if (isCageCellsSizeValid(cageCells))
27         {
28             this.cageCells = cageCells;
29         }
30         else
31         {
32             JOptionPane.showMessageDialog(null, "Invalid array size.", "Error",
33                                         JOptionPane.ERROR_MESSAGE);
34             throw new IllegalStateException("Invalid array size.");
35         }
36         if (isCageObjectivesSizeValid(cageObjectives))
37         {
38             this.cageObjectives = cageObjectives;
39         }
39         else
40         {
41             JOptionPane.showMessageDialog(null, "Invalid array size.", "Error",
42                                         JOptionPane.ERROR_MESSAGE);
43             throw new IllegalStateException("Invalid array size.");
44         }
45         this.grid = new Cell[size][size];
46         this.cages = new Cage[numberOfCages];
47         generateCages(cages);
48         for (int i = 0; i < cages.length; i++)
49         {
50             Integer[][] array = new Integer[size][size];
51             for (int j = 0; j < cageCells.length; j++)
52             {
53                 for (int k = 0; k < cageCells[j].length; k++)
54                 {
55                     if (cageCells[j][k] == i + 1)
56                     {
57                         array[j][k] = 1;
58                     }
59                     else
60                     {
61                         array[j][k] = 0;
62                     }
63                 }
64             }
65             if (!isCageAssignmentValid(array))
66             {
67                 JOptionPane.showMessageDialog(null, "Invalid cage assignment.",
68                                         "Error", JOptionPane.ERROR_MESSAGE);
69                 throw new IllegalStateException("Invalid cage assignment.");
70             }
71         }
72     }
73     generateGrid(grid, cages);
74     if (!isCagesValid(cages))

```

```

1      {
2          JOptionPane.showMessageDialog(null, "Invalid cages.", "Error",
3              JOptionPane.ERROR_MESSAGE);
4          throw new IllegalStateException("Invalid cages.");
5      }
6  }
7
8  private int countAreas(Integer [][] array)
9  {
10     boolean [][] checked = new boolean [size] [size];
11     for (int i = 0; i < size; i++)
12     {
13         for (int j = 0; j < size; j++)
14         {
15             checked [i] [j] = false;
16         }
17     }
18     return countAreas (array, checked);
19 }
20
21 private int countAreas (Integer [][] array, boolean [][] checked)
22 {
23     int areas = 0;
24     for (int i = 0; i < array.length; i++)
25     {
26         for (int j = 0; j < array.length; j++)
27         {
28             if (checked [i] [j])
29             {
30                 continue;
31             }
32             if (array [i] [j] == 0)
33             {
34                 checked [i] [j] = true;
35                 continue;
36             }
37             areas++;
38             floodFill (i, j, array, checked);
39         }
40     }
41     return areas;
42 }
43
44 private void floodFill (int i, int j, Integer [][] array,
45                         boolean [][] checked)
46 {
47     if (array [i] [j] == 0 || checked [i] [j])
48     {
49         return;
50     }
51     checked [i] [j] = true;
52     if (j < array.length - 1)
53     {
54         floodFill (i, j + 1, array, checked);
55     }
56     if (i < array.length - 1)
57     {
58         floodFill (i + 1, j, array, checked);
59     }
60     if (j > 0)
61     {
62         floodFill (i, j - 1, array, checked);
63     }
64     if (i > 0)
65     {
66         floodFill (i - 1, j, array, checked);
67     }
68 }
69
70 private boolean isCageCellsSizeValid (Integer [][] cageCells)
71 {
72     if (cageCells.length == size)
73     {
74         for (int i = 0; i < size; i++)
75         {
76             if (cageCells [i].length != size)
77             {
78                 return false;
79             }
80         }
81     }
82     else
83     {
84         return false;
85     }
86     return true;
87 }
88
89 private boolean isCageObjectivesSizeValid (String [] cageObjectives)
90 {
91     return cageCells.length == size;
92 }
93
94 private boolean isCageAssignmentValid (Integer [][] array)
95 {
96     return countAreas (array) == 1;
97 }
98
99 private boolean isCagesValid (Cage [] cages)
100 {
101     for (Cage c : cages)
102     {
103         if (c.getOperator () == '=' && c.getSize () != 1)

```

```

1      {
2          return false;
3      }
4      if ((c.getOperator() == '-' || c.getOperator() == '/')
5          && c.getSize() != 2)
6      {
7          return false;
8      }
9      if ((c.getOperator() == '+' || c.getOperator() == '*')
10         && c.getSize() < 2)
11     {
12         return false;
13     }
14   }
15   return true;
16 }
17
18 private void generateCages(Cage[] cages)
19 {
20     for (int i = 0; i < cages.length; i++)
21     {
22         cages[i] = new Cage(cageObjectives[i]);
23     }
24 }
25
26 private void generateGrid(Cell[][] grid, Cage[] cages)
27 {
28     for (int i = 0; i < cageCells.length; i++)
29     {
30         for (int j = 0; j < cageCells[i].length; j++)
31         {
32             grid[i][j] = new Cell(i, j, (cageCells[i][j] - 1));
33             cages[cageCells[i][j] - 1].addCell(grid[i][j]);
34         }
35     }
36 }
37
38 public ArrayList<Integer> getRow(int rowNumber)
39 {
40     ArrayList<Integer> row = new ArrayList<>();
41     for (int i = 0; i < grid.length; i++)
42     {
43         if (grid[rowNumber][i].getValue() != null)
44         {
45             row.add(grid[rowNumber][i].getValue());
46         }
47     }
48     return row;
49 }
50
51 public ArrayList<Integer> getColumn(int columnNumber)
52 {
53     ArrayList<Integer> column = new ArrayList<>();
54     for (Cell[] row : grid) {
55         if (row[columnNumber].getValue() != null) {
56             column.add(row[columnNumber].getValue());
57         }
58     }
59     return column;
60 }
61
62 public ArrayList<Integer> getCageValues(int cageNumber)
63 {
64     ArrayList<Integer> cage = new ArrayList<>();
65     for (int i = 0; i < cages[cageNumber].getSize(); i++)
66     {
67         if (cages[cageNumber].getCells().get(i).getValue() != null)
68         {
69             cage.add(cages[cageNumber].getCells().get(i).getValue());
70         }
71     }
72     return cage;
73 }
74
75 private boolean isArrayValid(ArrayList<Integer> array)
76 {
77     for (int i = 0; i < array.size(); i++)
78     {
79         for (int j = 0; j < array.size(); j++)
80         {
81             if (Objects.equals(array.get(i), array.get(j)) && (i != j))
82             {
83                 return false;
84             }
85         }
86     }
87     return true;
88 }
89
90 private boolean isRowValid(int row)
91 {
92     ArrayList<Integer> array = getRow(row);
93     if (!isArrayValid(array))
94     {
95         JOptionPane.showMessageDialog(null,
96             "Row " + row + " has duplicate numbers.",
97             "Information", JOptionPane.INFORMATION_MESSAGE);
98         return false;
99     }
100    else
101    {
102        return true;
103    }

```

```

1      }
2
3      private boolean solverIsRowValid(int row)
4      {
5          ArrayList<Integer> array = getRow(row);
6          return isArrayValid(array);
7      }
8
9      private boolean isColumnValid(int column)
10     {
11         ArrayList<Integer> array = getColumn(column);
12         if (!isArrayValid(array))
13         {
14             JOptionPane.showMessageDialog(null,
15                 "Column " + column + " has duplicate numbers. ",
16                 "Information", JOptionPane.INFORMATION_MESSAGE);
17             return false;
18         }
19         else
20         {
21             return true;
22         }
23     }
24
25     private boolean solverIsColumnValid(int column)
26     {
27         ArrayList<Integer> array = getColumn(column);
28         return isArrayValid(array);
29     }
30
31     private Boolean isCageValuesValid(int cageNumber)
32     {
33         if (cages[cageNumber].isCageValuesValid() == null)
34         {
35             return true;
36         }
37         else
38         {
39             return cages[cageNumber].isCageValuesValid();
40         }
41     }
42
43     private boolean isCageValid(int row, int column)
44     {
45         if (isCageValuesValid(
46             getGridContents()[row][column].get CageID()) == false)
47         {
48             JOptionPane.showMessageDialog(null,
49                 "Values of cells in the cage do not reach the target number. ",
50                 "Information", JOptionPane.INFORMATION_MESSAGE);
51             return false;
52         }
53         else
54         {
55             return true;
56         }
57     }
58
59     private boolean solverIsCageValid(int row, int column)
60     {
61         return isCageValuesValid(
62             getGridContents()[row][column].get CageID()) == true;
63     }
64
65     public boolean isCellValueValid(int row, int column)
66     {
67         return (isRowValid(row) && isColumnValid(column)
68             && isCageValid(row, column));
69     }
70
71     public boolean solverIsCellValueValid(int row, int column)
72     {
73         return (solverIsRowValid(row) && solverIsColumnValid(column)
74             && solverIsCageValid(row, column));
75     }
76
77     public boolean setCellValue(int row, int column, Integer value)
78     {
79         if (value > 0 && value <= size)
80         {
81             getGridContents()[row][column].setValue(value);
82             isWin();
83             return isCellValueValid(row, column);
84         }
85         else
86         {
87             JOptionPane.showMessageDialog(null,
88                 "Cell value must be between 1 and " + size + ".",
89                 "Information", JOptionPane.INFORMATION_MESSAGE);
90             return false;
91         }
92     }
93
94     public boolean solverSetCellValue(int row, int column, Integer value)
95     {
96         if (value > 0 && value <= size)
97         {
98             getGridContents()[row][column].setValue(value);
99             return (solverIsRowValid(row) && solverIsColumnValid(column)
100                && solverIsCageValid(row, column));
101        }
102        else
103        {

```

```

1         return false;
2     }
3 }
4
5 public void unsetCellValue(int row, int column)
6 {
7     getGridContents() [row] [column].setValue(null);
8 }
9
10 public Boolean isWin()
11 {
12     for (int i = 0; i < size; i++)
13     {
14         for (int j = 0; j < size; j++)
15         {
16             if (getGridContents () [i] [j].getValue () == null)
17             {
18                 return null;
19             }
20             if (isCellValueValid (i, j) == false)
21             {
22                 return false;
23             }
24         }
25     }
26     JOptionPane.showMessageDialog (null,
27         "Congratulations, you have successfully solved the puzzle!",
28         "Information", JOptionPane.INFORMATION_MESSAGE);
29     return true;
30 }
31
32 public Boolean checkGrid()
33 {
34     for (int i = 0; i < size; i++)
35     {
36         for (int j = 0; j < size; j++)
37         {
38             if (getGridContents () [i] [j].getValue () == null)
39             {
40                 JOptionPane.showMessageDialog (null,
41                     "There are empty cells in the grid.",
42                     "Information", JOptionPane.INFORMATION_MESSAGE);
43                 return null;
44             }
45             if (solverIsCellValueValid (i, j) == false)
46             {
47                 JOptionPane.showMessageDialog (null,
48                     "There are cells with incorrect values in the"
49                     + "grid.", "Information",
50                     JOptionPane.INFORMATION_MESSAGE);
51                 return false;
52             }
53         }
54     }
55     JOptionPane.showMessageDialog (null,
56         "Congratulations, you have successfully solved the puzzle!",
57         "Information", JOptionPane.INFORMATION_MESSAGE);
58     return true;
59 }
60
61 public boolean isFilled()
62 {
63     for (int i = 0; i < size; i++)
64     {
65         for (int j = 0; j < size; j++)
66         {
67             if (getGridContents () [i] [j].getValue () == null)
68             {
69                 return false;
70             }
71         }
72     }
73     return true;
74 }
75
76 public Integer getCellValue(int row, int column)
77 {
78     return getGridContents () [row] [column].getValue ();
79 }
80
81 public Integer getSize()
82 {
83     return size;
84 }
85
86 public Integer getNumberOfCages()
87 {
88     return numberOfCages;
89 }
90
91 public Integer [][] getCageCells()
92 {
93     return cageCells;
94 }
95
96 public String [] getCageObjectives()
97 {
98     return cageObjectives;
99 }
100
101 public Cell [][] getGridContents()
102 {
103     return grid;

```

```

1     }
2
3     public Cage[] getCages()
4     {
5         return cages;
6     }
7
8     public Grid getGame()
9     {
10    return this;
11 }
12
13 }
```

14 D.2 Cell.java

Listing D.2: Cell.java

```

15 package model;
16
17 public class Cell
18 {
19
20     private final int row;
21     private final int column;
22     private final int cageID;
23     private Integer value;
24
25     public Cell(int row, int column, int cageID)
26     {
27         this.row = row;
28         this.column = column;
29         this.cageID = cageID;
30     }
31
32     public void setValue(Integer value)
33     {
34         this.value = value;
35     }
36
37     public Integer getValue()
38     {
39         return value;
40     }
41
42     public int getRow()
43     {
44         return row;
45     }
46
47     public int getColumn()
48     {
49         return column;
50     }
51
52     public int getCageID()
53     {
54         return cageID;
55     }
56 }
57 }
```

58 D.3 Cage.java

Listing D.3: Cage.java

```

59 package model;
60
61 import java.util.ArrayList;
62 import javax.swing.JOptionPane;
63
64 public class Cage
65 {
66
67     private final String objective;
68     private final int targetNumber;
69     private final char operator;
70     private final ArrayList<Cell> cells;
71
72     public Cage(String objective)
73     {
74         if (isCageObjectiveValid(objective))
75         {
76             this.objective = objective;
77         }
78         else
79         {
80             JOptionPane.showMessageDialog(null, "Invalid cage objectives.", "Error", JOptionPane.ERROR_MESSAGE);
81             throw new IllegalStateException("Invalid cage objectives.");
82         }
83         this.targetNumber = generateTargetNumber(this.objective);
84         this.operator = generateOperator(this.objective);
85         this.cells = new ArrayList<>();
86     }
87 }
```

```

1  private boolean isCageObjectiveValid(String cageObjective)
2  {
3      return cageObjective.matches("\d+[*+-/=]");
4  }
5
6  private int generateTargetNumber(String objective)
7  {
8      return Integer.parseInt(objective.substring(0,
9          objective.length() - 1));
10 }
11
12 private char generateOperator(String objective)
13 {
14     return objective.charAt(objective.length() - 1);
15 }
16
17 public void addCell(Cell c)
18 {
19     cells.add(c);
20 }
21
22
23 public boolean isCageContainsNull()
24 {
25     for (int i = 0; i < cells.size(); i++)
26     {
27         if (cells.get(i).getValue() == null)
28         {
29             return true;
30         }
31     }
32     return false;
33 }
34
35 public Boolean isCageValuesValid()
36 {
37     if (countValue() == null)
38     {
39         return null;
40     }
41     else
42     {
43         return (countValue() == getTargetNumber());
44     }
45 }
46
47 private Integer countValue()
48 {
49     Integer value = null;
50     if (isCageContainsNull() == true)
51     {
52         value = null;
53     }
54     else
55     {
56         switch (getOperator())
57         {
58             case '+':
59                 value = 0;
60                 for (int i = 0; i < cells.size(); i++)
61                 {
62                     value += cells.get(i).getValue();
63                 }
64                 break;
65             case '-':
66                 if (cells.get(0).getValue() > cells.get(1).getValue())
67                 {
68                     value = cells.get(0).getValue()
69                         - cells.get(1).getValue();
70                 }
71                 else if (cells.get(1).getValue() > cells.get(0).getValue())
72                 {
73                     value = cells.get(1).getValue()
74                         - cells.get(0).getValue();
75                 }
76                 else
77                 {
78                     value = 0;
79                 }
80                 break;
81             case '*':
82                 value = 1;
83                 for (int i = 0; i < cells.size(); i++)
84                 {
85                     value *= cells.get(i).getValue();
86                 }
87                 break;
88             case '/':
89                 if (cells.get(0).getValue() > cells.get(1).getValue())
90                 {
91                     if (cells.get(0).getValue()
92                         % cells.get(1).getValue() == 0)
93                     {
94                         value = cells.get(0).getValue()
95                             / cells.get(1).getValue();
96                     }
97                     else
98                     {
99                         value = 0;
100                    }
101                }
102            }
103        }

```

```

1      if (cells.get(1).getValue() == 0)
2          value = cells.get(0).getValue();
3      else
4          value = 0;
5  }
6  else
7  {
8      value = 0;
9  }
10 }
11 else
12 {
13     value = 0;
14 }
15 break;
16 case '=':
17     value = cells.get(0).getValue();
18     break;
19 default :
20     value = null;
21 }
22 }
23 return value;
24 }
25
26 public String getObjective()
27 {
28     return objective;
29 }
30
31 public int getTargetNumber()
32 {
33     return targetNumber;
34 }
35
36 public char getOperator()
37 {
38     return operator;
39 }
40
41 public ArrayList<Cell> getCells()
42 {
43     return cells;
44 }
45
46 public int getSize()
47 {
48     return cells.size();
49 }
50
51 }
52 }
```

53 D.4 SolverBacktracking.java

Listing D.4: SolverBacktracking.java

```

54 package model;
55
56 public class SolverBacktracking
57 {
58
59     private final Grid grid;
60     private final Integer size;
61     private Grid solution;
62
63     public SolverBacktracking(Grid grid)
64     {
65         this.grid = grid;
66         this.size = grid.getSize();
67     }
68
69     public boolean solve()
70     {
71         if (solve(0, 0) == true)
72         {
73             this.solution = grid;
74             printGrid(solution.getGridContents());
75             return true;
76         }
77         else
78         {
79             return false;
80         }
81     }
82
83     private boolean solve(int row, int column)
84     {
85         if (column >= size)
86         {
87             column = 0;
88             row++;
89             if (row >= size)
90             {
91                 printGrid(grid.getGridContents());
92                 return true;
93             }
94         }
95         for (int value = 1; value <= size; value++)
96         {
97             if (isSafe(row, column, value))
98             {
99                 grid.setCell(row, column, value);
100                if (solve(row, column + 1) == true)
101                {
102                    return true;
103                }
104            }
105        }
106        grid.setCell(row, column, 0);
107        return false;
108    }
109
110    private boolean isSafe(int row, int column, int value)
111    {
112        for (int i = 0; i < size; i++)
113        {
114            if (grid.getCell(row, i).getValue() == value)
115            {
116                return false;
117            }
118        }
119        for (int i = 0; i < size; i++)
120        {
121            if (grid.getCell(i, column).getValue() == value)
122            {
123                return false;
124            }
125        }
126        int startRow = Math.max(0, row - 3);
127        int startColumn = Math.max(0, column - 3);
128        for (int i = startRow; i <= row; i++)
129        {
130            for (int j = startColumn; j <= column; j++)
131            {
132                if (grid.getCell(i, j).getValue() == value)
133                {
134                    return false;
135                }
136            }
137        }
138        return true;
139    }
140
141    private void printGrid(List<List<Integer>> grid)
142    {
143        for (List<Integer> row : grid)
144        {
145            for (Integer cell : row)
146            {
147                System.out.print(cell + " ");
148            }
149            System.out.println();
150        }
151    }
152 }
```

```

1   {
2       printGrid(grid.getGridContents());
3       if (grid.solverSetValue(row, column, value))
4       {
5           if (solve(row, column + 1))
6           {
7               return true;
8           }
9       }
10      }
11      printGrid(grid.getGridContents());
12      grid.unsetCellValue(row, column);
13      return false;
14  }
15
16  public Grid getGrid()
17  {
18      return grid;
19  }
20
21  public Grid getSolution()
22  {
23      return solution;
24  }
25
26  private void printGrid(Cell [][] cells)
27  {
28      for (int i = 0; i < size; i++)
29      {
30          for (int j = 0; j < size; j++)
31          {
32              System.out.print(cells[i][j].getValue() + " ");
33          }
34          System.out.println("");
35      }
36      System.out.println("");
37  }
38 }
39 }
```

40 D.5 SolverHybridGenetic.java

Listing D.5: SolverHybridGenetic.java

```

41 package model;
42
43 public class SolverHybridGenetic
44 {
45
46     private final Grid grid;
47     private Grid gridRuleBased;
48     private final Integer size;
49     private Grid solution;
50     private final Integer generations;
51     private final Integer populationSize;
52     private final Double elitismRate;
53     private final Double mutationRate;
54     private final Double crossoverRate;
55
56     public SolverHybridGenetic(Grid grid, Integer generations,
57         Integer populationSize, Double elitismRate, Double crossoverRate,
58         Double mutationRate)
59     {
60         this.grid = grid;
61         this.size = grid.getSize();
62         this.generations = generations;
63         this.populationSize = populationSize;
64         this.elitismRate = elitismRate;
65         this.crossoverRate = crossoverRate;
66         this.mutationRate = mutationRate;
67     }
68
69     public boolean solve()
70     {
71         SolverRuleBased srb = new SolverRuleBased(grid);
72         boolean isFilled = srb.solve();
73         this.gridRuleBased = srb.getGrid();
74         if (isFilled)
75         {
76             this.solution = srb.getSolution();
77             printGrid(solution.getGridContents());
78             return true;
79         }
80         else
81         {
82             SolverGenetic sg = new SolverGenetic(gridRuleBased, generations,
83                 populationSize, elitismRate, crossoverRate, mutationRate);
84             if (sg.solve() == true)
85             {
86                 this.solution = sg.getSolution();
87                 printGrid(solution.getGridContents());
88                 return true;
89             }
90         }
91         return false;
92     }
93
94     public Grid getGrid()
95     {
```

```

1     return grid;
2 }
3
4 public Grid getSolution()
5 {
6     return solution;
7 }
8
9 private void printGrid(Cell[][] cells)
10 {
11     for (int i = 0; i < size; i++)
12     {
13         for (int j = 0; j < size; j++)
14         {
15             System.out.print(cells[i][j].getValue() + " ");
16         }
17         System.out.println("");
18     }
19     System.out.println("");
20 }
21 }
22 }
```

23 D.6 SolverRuleBased.java

Listing D.6: SolverRuleBased.java

```

24 package model;
25
26 import java.util.ArrayList;
27 import java.util.Collections;
28 import javax.swing.JOptionPane;
29
30 public class SolverRuleBased
31 {
32
33     private final Grid grid;
34     private final Integer size;
35     private Grid solution;
36     private ArrayList<Integer>[][] possibleValues;
37
38     public SolverRuleBased(Grid grid)
39     {
40         this.grid = grid;
41         this.size = grid.getSize();
42         this.possibleValues = generatePossibleValuesArray();
43     }
44
45     @SuppressWarnings("unchecked")
46     private ArrayList<Integer>[][] generatePossibleValuesArray()
47     {
48         possibleValues = new ArrayList[size][size];
49         for (int i = 0; i < size; i++)
50         {
51             for (int j = 0; j < size; j++)
52             {
53                 possibleValues[i][j] = new ArrayList<Integer>();
54             }
55         }
56         for (int i = 0; i < size; i++)
57         {
58             for (int j = 0; j < size; j++)
59             {
60                 for (int k = 1; k <= size; k++)
61                 {
62                     possibleValues[i][j].add(k);
63                 }
64             }
65         }
66     }
67     return possibleValues;
68 }
69
70     public boolean solve()
71     {
72         singleSquare();
73         killerCombination();
74         ArrayList<ArrayList<Integer>> currentGridArrayList
75         = getGridArrayList();
76         ArrayList<ArrayList<Integer>> newGridArrayList = solveLoop();
77         while (!currentGridArrayList.equals(newGridArrayList))
78         {
79             printGrid();
80             printPossibleValues();
81             currentGridArrayList = newGridArrayList;
82             newGridArrayList = solveLoop();
83         }
84         if (grid.isFilled())
85         {
86             this.solution = grid;
87             return true;
88         }
89         else
90         {
91             return false;
92         }
93     }
94     private ArrayList<ArrayList<Integer>> solveLoop()
95     {
```

```

1     nakedSingle();
2     nakedDouble();
3     hiddenSingle();
4     return getGridArrayList();
5 }
6
7 @SuppressWarnings("unchecked")
8 public ArrayList<Integer>[] getRowPossibleValues(int row)
9 {
10    ArrayList<Integer>[] array;
11   array = new ArrayList[size];
12   for (int i = 0; i < size; i++)
13   {
14      array[i] = possibleValues[row][i];
15   }
16   return array;
17 }
18
19 @SuppressWarnings("unchecked")
20 public ArrayList<Integer>[] getColumnPossibleValues(int column)
21 {
22    ArrayList<Integer>[] array;
23   array = new ArrayList[size];
24   for (int i = 0; i < size; i++)
25   {
26      array[i] = possibleValues[i][column];
27   }
28   return array;
29 }
30
31 private void singleSquare()
32 {
33   for (Cage c : grid.getCages())
34   {
35     if (c.getSize() == 1)
36     {
37       setCellValue(c.getCells().get(0).getRow(),
38                   c.getCells().get(0).getColumn(), c.getTargetNumber());
39     }
40   }
41 }
42
43 private void killerCombination()
44 {
45   int cageSize;
46   ArrayList<Integer> array = new ArrayList<Integer>();
47   for (Cage c : grid.getCages())
48   {
49     cageSize = c.getSize();
50     switch (cageSize)
51     {
52       case 2:
53         killerCombinationCageSize2(c);
54         break;
55       default:
56         array = createRetainAllArray();
57         removeImpossibleValuesCage(c, array);
58         break;
59     }
60   }
61 }
62
63 private void killerCombinationCageSize2(Cage cage)
64 {
65   int gridSize = size;
66   ArrayList<Integer> array = new ArrayList<Integer>();
67   switch (gridSize)
68   {
69     case 3 :
70       array = killerCombinationCageSize2GridSize3(cage);
71       break;
72     case 4 :
73       array = killerCombinationCageSize2GridSize4(cage);
74       break;
75     case 5 :
76       array = killerCombinationCageSize2GridSize5(cage);
77       break;
78     case 6 :
79       array = killerCombinationCageSize2GridSize6(cage);
80       break;
81     case 7 :
82       array = killerCombinationCageSize2GridSize7(cage);
83       break;
84     case 8 :
85       array = killerCombinationCageSize2GridSize8(cage);
86       break;
87     case 9 :
88       array = killerCombinationCageSize2GridSize9(cage);
89       break;
90     default :
91       JOptionPane.showMessageDialog(null, "Invalid grid size.",
92                                 "Error", JOptionPane.ERROR_MESSAGE);
93       throw new IllegalStateException("Invalid grid size.");
94   }
95   removeImpossibleValuesCage(cage, array);
96 }
97
98 private ArrayList<Integer> killerCombinationCageSize2GridSize3(Cage cage)
99 {
100    char cageOperator = cage.getOperator();
101   int cageTargetNumber = cage.getTargetNumber();
102   ArrayList<Integer> array = new ArrayList<Integer>();
103   switch (cageOperator)

```

```

1      {
2          case '+':
3              switch (cageTargetNumber)
4              {
5                  case 3:
6                      array.add(1);
7                      array.add(2);
8                      break;
9                  case 4:
10                     array.add(1);
11                     array.add(3);
12                     break;
13                 case 5:
14                     array.add(2);
15                     array.add(3);
16                     break;
17                 default:
18                     array = createRetainAllArray();
19                     break;
20             }
21             break;
22         case '-':
23             switch (cageTargetNumber)
24             {
25                 case 2:
26                     array.add(1);
27                     array.add(3);
28                     break;
29                 default:
30                     array = createRetainAllArray();
31                     break;
32             }
33             break;
34         case '*':
35             switch (cageTargetNumber)
36             {
37                 case 2:
38                     array.add(1);
39                     array.add(2);
40                     break;
41                 case 3:
42                     array.add(1);
43                     array.add(3);
44                     break;
45                 case 6:
46                     array.add(2);
47                     array.add(3);
48                     break;
49                 default:
50                     array = createRetainAllArray();
51                     break;
52             }
53             break;
54         case '/':
55             switch (cageTargetNumber)
56             {
57                 case 2:
58                     array.add(1);
59                     array.add(2);
60                     break;
61                 case 3:
62                     array.add(1);
63                     array.add(3);
64                     break;
65                 default:
66                     array = createRetainAllArray();
67                     break;
68             }
69             break;
70         default:
71             JOptionPane.showMessageDialog(null,
72                         "Invalid operator.", "Error",
73                         JOptionPane.ERROR_MESSAGE);
74             throw new IllegalStateException("Invalid operator.");
75     }
76     return array;
77 }
78
79 private ArrayList<Integer> killerCombinationCageSize2GridSize4(Cage cage)
80 {
81     char cageOperator = cage.getOperator();
82     int cageTargetNumber = cage.getTargetNumber();
83     ArrayList<Integer> array = new ArrayList<Integer>();
84     switch (cageOperator)
85     {
86         case '+':
87             switch (cageTargetNumber)
88             {
89                 case 3:
90                     array.add(1);
91                     array.add(2);
92                     break;
93                 case 4:
94                     array.add(1);
95                     array.add(3);
96                     break;
97                 case 6:
98                     array.add(2);
99                     array.add(4);
100                    break;
101                case 7:
102                    array.add(3);
103                    array.add(4);

```

```

1         break;
2     default :
3         array = createRetainAllArray();
4         break;
5     }
6     break;
7     case '-' :
8         switch (cageTargetNumber)
9         {
10             case 3 :
11                 array.add(1);
12                 array.add(4);
13                 break;
14             default :
15                 array = createRetainAllArray();
16                 break;
17         }
18     break;
19     case '*' :
20         switch (cageTargetNumber)
21         {
22             case 2 :
23                 array.add(1);
24                 array.add(2);
25                 break;
26             case 3 :
27                 array.add(1);
28                 array.add(3);
29                 break;
30             case 4 :
31                 array.add(1);
32                 array.add(4);
33                 break;
34             case 6 :
35                 array.add(2);
36                 array.add(3);
37                 break;
38             case 8 :
39                 array.add(2);
40                 array.add(4);
41                 break;
42             case 12 :
43                 array.add(3);
44                 array.add(4);
45                 break;
46             default :
47                 array = createRetainAllArray();
48                 break;
49         }
50     break;
51     case '/' :
52         switch (cageTargetNumber)
53         {
54             case 3 :
55                 array.add(1);
56                 array.add(3);
57                 break;
58             case 4 :
59                 array.add(1);
60                 array.add(4);
61                 break;
62             default :
63                 array = createRetainAllArray();
64                 break;
65         }
66     break;
67     default :
68         JOptionPane.showMessageDialog(null,
69             "Invalid operator.", "Error",
70             JOptionPane.ERROR_MESSAGE);
71         throw new IllegalStateException("Invalid operator.");
72     }
73     return array;
74 }
75
76 private ArrayList<Integer> killerCombinationCageSize2GridSize5(Cage cage)
77 {
78     char cageOperator = cage.getOperator();
79     int cageTargetNumber = cage.getTargetNumber();
80     ArrayList<Integer> array = new ArrayList<Integer>();
81     switch (cageOperator)
82     {
83         case '+' :
84             switch (cageTargetNumber)
85             {
86                 case 3 :
87                     array.add(1);
88                     array.add(2);
89                     break;
90                 case 4 :
91                     array.add(1);
92                     array.add(3);
93                     break;
94                 case 8 :
95                     array.add(3);
96                     array.add(5);
97                     break;
98                 case 9 :
99                     array.add(4);
100                    array.add(5);
101                    break;
102                 default :
103                     array = createRetainAllArray();

```

```

1             break;
2         }
3     }
4     case '-' :
5     {
6         switch (cageTargetNumber)
7         {
8             case 4 :
9                 array.add(1);
10                array.add(5);
11                break;
12            default :
13                array = createRetainAllArray();
14                break;
15        }
16    }
17    case '*' :
18    {
19        switch (cageTargetNumber)
20        {
21            case 2 :
22                array.add(1);
23                array.add(2);
24                break;
25            case 3 :
26                array.add(1);
27                array.add(3);
28                break;
29            case 4 :
30                array.add(1);
31                array.add(4);
32                break;
33            case 5 :
34                array.add(1);
35                array.add(5);
36                break;
37            case 6 :
38                array.add(2);
39                array.add(3);
40                break;
41            case 8 :
42                array.add(2);
43                array.add(4);
44                break;
45            case 10 :
46                array.add(2);
47                array.add(5);
48                break;
49            case 12 :
50                array.add(3);
51                array.add(4);
52                break;
53            case 15 :
54                array.add(3);
55                array.add(5);
56                break;
57            case 20 :
58                array.add(4);
59                array.add(5);
60                break;
61            default :
62                array = createRetainAllArray();
63                break;
64        }
65    }
66    case '/' :
67    {
68        switch (cageTargetNumber)
69        {
70            case 3 :
71                array.add(1);
72                array.add(3);
73                break;
74            case 4 :
75                array.add(1);
76                array.add(4);
77                break;
78            case 5 :
79                array.add(1);
80                array.add(5);
81                break;
82            default :
83                array = createRetainAllArray();
84                break;
85        }
86    }
87    default :
88        JOptionPane.showMessageDialog(null,
89                     "Invalid operator .", "Error",
90                     JOptionPane.ERROR_MESSAGE);
91        throw new IllegalStateException("Invalid operator .");
92    }
93    return array;
94}
95 private ArrayList<Integer> killerCombinationCageSize2GridSize6(Cage cage)
96 {
97     char cageOperator = cage.getOperator();
98     int cageTargetNumber = cage.getTargetNumber();
99     ArrayList<Integer> array = new ArrayList<Integer>();
100    switch (cageOperator)
101    {
102        case '+' :
103            switch (cageTargetNumber)
104            {
105                case 3 :

```

```

1         array.add(1);
2         array.add(2);
3         break;
4     case 4 :
5         array.add(1);
6         array.add(3);
7         break;
8     case 10 :
9         array.add(4);
10        array.add(6);
11        break;
12    case 11 :
13        array.add(5);
14        array.add(6);
15        break;
16    default :
17        array = createRetainAllArray();
18        break;
19    }
20    break;
21 case '_':
22     switch (cageTargetNumber)
23     {
24         case 5 :
25             array.add(1);
26             array.add(6);
27             break;
28         default :
29             array = createRetainAllArray();
30             break;
31     }
32     break;
33 case '*':
34     switch (cageTargetNumber)
35     {
36         case 2 :
37             array.add(1);
38             array.add(2);
39             break;
40         case 3 :
41             array.add(1);
42             array.add(3);
43             break;
44         case 4 :
45             array.add(1);
46             array.add(4);
47             break;
48         case 5 :
49             array.add(1);
50             array.add(5);
51             break;
52         case 8 :
53             array.add(2);
54             array.add(4);
55             break;
56         case 10 :
57             array.add(2);
58             array.add(5);
59             break;
60         case 15 :
61             array.add(3);
62             array.add(5);
63             break;
64         case 18 :
65             array.add(3);
66             array.add(6);
67             break;
68         case 20 :
69             array.add(4);
70             array.add(5);
71             break;
72         case 24 :
73             array.add(4);
74             array.add(6);
75             break;
76         case 30 :
77             array.add(5);
78             array.add(6);
79             break;
80     default :
81         array = createRetainAllArray();
82         break;
83     }
84     break;
85 case '/':
86     switch (cageTargetNumber)
87     {
88         case 4 :
89             array.add(1);
90             array.add(4);
91             break;
92         case 5 :
93             array.add(1);
94             array.add(5);
95             break;
96         case 6 :
97             array.add(1);
98             array.add(6);
99             break;
100    default :
101        array = createRetainAllArray();
102        break;
103    }

```

```

1         break;
2     default :
3         JOptionPane.showMessageDialog(null ,
4             "Invalid operator .", "Error",
5             JOptionPane.ERROR_MESSAGE);
6         throw new IllegalStateException("Invalid operator .");
7     }
8     return array ;
9 }
10
11 private ArrayList<Integer> killerCombinationCageSize2GridSize7(Cage cage)
12 {
13     char cageOperator = cage.getOperator();
14     int cageTargetNumber = cage.getTargetNumber();
15     ArrayList<Integer> array = new ArrayList<Integer>();
16     switch (cageOperator)
17     {
18         case '+':
19             switch (cageTargetNumber)
20             {
21                 case 3:
22                     array.add(1);
23                     array.add(2);
24                     break;
25                 case 4:
26                     array.add(1);
27                     array.add(3);
28                     break;
29                 case 12:
30                     array.add(5);
31                     array.add(7);
32                     break;
33                 case 13:
34                     array.add(6);
35                     array.add(7);
36                     break;
37                 default:
38                     array = createRetainAllArray();
39                     break;
40             }
41             break;
42         case '-':
43             switch (cageTargetNumber)
44             {
45                 case 6:
46                     array.add(1);
47                     array.add(7);
48                     break;
49                 default:
50                     array = createRetainAllArray();
51                     break;
52             }
53             break;
54         case '*':
55             switch (cageTargetNumber)
56             {
57                 case 2:
58                     array.add(1);
59                     array.add(2);
60                     break;
61                 case 3:
62                     array.add(1);
63                     array.add(3);
64                     break;
65                 case 4:
66                     array.add(1);
67                     array.add(4);
68                     break;
69                 case 5:
70                     array.add(1);
71                     array.add(5);
72                     break;
73                 case 7:
74                     array.add(1);
75                     array.add(7);
76                     break;
77                 case 8:
78                     array.add(2);
79                     array.add(4);
80                     break;
81                 case 10:
82                     array.add(2);
83                     array.add(5);
84                     break;
85                 case 14:
86                     array.add(2);
87                     array.add(7);
88                     break;
89                 case 15:
90                     array.add(3);
91                     array.add(5);
92                     break;
93                 case 18:
94                     array.add(3);
95                     array.add(6);
96                     break;
97                 case 20:
98                     array.add(4);
99                     array.add(5);
100                    break;
101                case 21:
102                    array.add(3);
103                    array.add(7);

```

```

1         break;
2     case 24 :
3         array.add(4);
4         array.add(6);
5         break;
6     case 28 :
7         array.add(4);
8         array.add(7);
9         break;
10    case 30 :
11        array.add(5);
12        array.add(6);
13        break;
14    case 35 :
15        array.add(5);
16        array.add(7);
17        break;
18    case 42 :
19        array.add(6);
20        array.add(7);
21        break;
22    default :
23        array = createRetainAllArray();
24        break;
25    }
26    break;
27 case '/':
28 {
29     switch (cageTargetNumber)
30     {
31         case 4 :
32             array.add(1);
33             array.add(4);
34             break;
35         case 5 :
36             array.add(1);
37             array.add(5);
38             break;
39         case 6 :
40             array.add(1);
41             array.add(6);
42             break;
43         case 7 :
44             array.add(1);
45             array.add(7);
46             break;
47     default :
48         array = createRetainAllArray();
49         break;
50     }
51     break;
52 default :
53     JOptionPane.showMessageDialog(null,
54         "Invalid operator.", "Error",
55         JOptionPane.ERROR_MESSAGE);
56     throw new IllegalStateException("Invalid operator.");
57 }
58 return array;
59 }
60 private ArrayList<Integer> killerCombinationCageSize2GridSize8(Cage cage)
61 {
62     char cageOperator = cage.getOperator();
63     int cageTargetNumber = cage.getTargetNumber();
64     ArrayList<Integer> array = new ArrayList<Integer>();
65     switch (cageOperator)
66     {
67         case '+':
68             switch (cageTargetNumber)
69             {
70                 case 3 :
71                     array.add(1);
72                     array.add(2);
73                     break;
74                 case 4 :
75                     array.add(1);
76                     array.add(3);
77                     break;
78                 case 14 :
79                     array.add(6);
80                     array.add(8);
81                     break;
82                 case 15 :
83                     array.add(7);
84                     array.add(8);
85                     break;
86             default :
87                 array = createRetainAllArray();
88                 break;
89             }
90             break;
91         case '-':
92             switch (cageTargetNumber)
93             {
94                 case 7 :
95                     array.add(1);
96                     array.add(8);
97                     break;
98             default :
99                 array = createRetainAllArray();
100                break;
101            }
102            break;
103        case '*':

```

```

1   switch (cageTargetNumber)
2   {
3       case 2 :
4           array.add(1);
5           array.add(2);
6           break;
7       case 3 :
8           array.add(1);
9           array.add(3);
10          break;
11      case 4 :
12          array.add(1);
13          array.add(4);
14          break;
15      case 5 :
16          array.add(1);
17          array.add(5);
18          break;
19      case 7 :
20          array.add(1);
21          array.add(7);
22          break;
23      case 10 :
24          array.add(2);
25          array.add(5);
26          break;
27      case 14 :
28          array.add(2);
29          array.add(7);
30          break;
31      case 15 :
32          array.add(3);
33          array.add(5);
34          break;
35      case 16 :
36          array.add(2);
37          array.add(8);
38          break;
39      case 18 :
40          array.add(3);
41          array.add(6);
42          break;
43      case 20 :
44          array.add(4);
45          array.add(5);
46          break;
47      case 21 :
48          array.add(3);
49          array.add(7);
50          break;
51      case 28 :
52          array.add(4);
53          array.add(7);
54          break;
55      case 30 :
56          array.add(5);
57          array.add(6);
58          break;
59      case 32 :
60          array.add(4);
61          array.add(8);
62          break;
63      case 35 :
64          array.add(5);
65          array.add(7);
66          break;
67      case 40 :
68          array.add(5);
69          array.add(8);
70          break;
71      case 42 :
72          array.add(6);
73          array.add(7);
74          break;
75      case 48 :
76          array.add(6);
77          array.add(8);
78          break;
79      case 56 :
80          array.add(7);
81          array.add(8);
82          break;
83      default :
84          array = createRetainAllArray();
85          break;
86      }
87      break;
88  case '/':
89      switch (cageTargetNumber)
90      {
91          case 5 :
92              array.add(1);
93              array.add(5);
94              break;
95          case 6 :
96              array.add(1);
97              array.add(6);
98              break;
99          case 7 :
100             array.add(1);
101             array.add(7);
102             break;
103         case 8 :

```

```
1         array.add(1);
2         array.add(8);
3         break;
4     default :
5         array = createRetainAllArray();
6         break;
7     }
8     break;
9 default :
10    JOptionPane.showMessageDialog(null,
11        "Invalid operator ", "Error",
12        JOptionPane.ERROR_MESSAGE);
13    throw new IllegalStateException("Invalid operator.");
14}
15 return array;
16}
17
18 private ArrayList<Integer> killerCombinationCageSize2GridSize9(Cage cage)
19 {
20     char cageOperator = cage.getOperator();
21     int cageTargetNumber = cage.getTargetNumber();
22     ArrayList<Integer> array = new ArrayList<Integer>();
23     switch (cageOperator)
24     {
25         case '+':
26             switch (cageTargetNumber)
27             {
28                 case 3:
29                     array.add(1);
30                     array.add(2);
31                     break;
32                 case 4:
33                     array.add(1);
34                     array.add(3);
35                     break;
36                 case 16:
37                     array.add(7);
38                     array.add(9);
39                     break;
40                 case 17:
41                     array.add(8);
42                     array.add(9);
43                     break;
44                 default:
45                     array = createRetainAllArray();
46                     break;
47             }
48             break;
49         case '-':
50             switch (cageTargetNumber)
51             {
52                 case 8:
53                     array.add(1);
54                     array.add(9);
55                     break;
56                 default:
57                     array = createRetainAllArray();
58                     break;
59             }
60             break;
61         case '*':
62             switch (cageTargetNumber)
63             {
64                 case 2:
65                     array.add(1);
66                     array.add(2);
67                     break;
68                 case 3:
69                     array.add(1);
70                     array.add(3);
71                     break;
72                 case 4:
73                     array.add(1);
74                     array.add(4);
75                     break;
76                 case 5:
77                     array.add(1);
78                     array.add(5);
79                     break;
80                 case 7:
81                     array.add(1);
82                     array.add(7);
83                     break;
84                 case 9:
85                     array.add(1);
86                     array.add(9);
87                     break;
88                 case 10:
89                     array.add(2);
90                     array.add(5);
91                     break;
92                 case 14:
93                     array.add(2);
94                     array.add(7);
95                     break;
96                 case 15:
97                     array.add(3);
98                     array.add(5);
99                     break;
100                case 16:
101                    array.add(2);
102                    array.add(8);
103                    break;
```

```

1      case 20 :
2          array.add(4);
3          array.add(5);
4          break;
5      case 21 :
6          array.add(3);
7          array.add(7);
8          break;
9      case 27 :
10         array.add(3);
11         array.add(9);
12         break;
13     case 28 :
14         array.add(4);
15         array.add(7);
16         break;
17     case 30 :
18         array.add(5);
19         array.add(6);
20         break;
21     case 32 :
22         array.add(4);
23         array.add(8);
24         break;
25     case 35 :
26         array.add(5);
27         array.add(7);
28         break;
29     case 36 :
30         array.add(4);
31         array.add(9);
32         break;
33     case 40 :
34         array.add(5);
35         array.add(8);
36         break;
37     case 42 :
38         array.add(6);
39         array.add(7);
40         break;
41     case 45 :
42         array.add(5);
43         array.add(9);
44         break;
45     case 48 :
46         array.add(6);
47         array.add(8);
48         break;
49     case 54 :
50         array.add(6);
51         array.add(9);
52         break;
53     case 56 :
54         array.add(7);
55         array.add(8);
56         break;
57     case 63 :
58         array.add(7);
59         array.add(9);
60         break;
61     case 72 :
62         array.add(8);
63         array.add(9);
64         break;
65     default :
66         array = createRetainAllArray();
67         break;
68     }
69     break;
70   case '/':
71     switch (cageTargetNumber)
72     {
73       case 5 :
74           array.add(1);
75           array.add(5);
76           break;
77       case 6 :
78           array.add(1);
79           array.add(6);
80           break;
81       case 7 :
82           array.add(1);
83           array.add(7);
84           break;
85       case 8 :
86           array.add(1);
87           array.add(8);
88           break;
89       case 9 :
90           array.add(1);
91           array.add(9);
92           break;
93     default :
94         array = createRetainAllArray();
95         break;
96     }
97     break;
98   default :
99     JOptionPane.showMessageDialog(null,
100        "Invalid operator.", "Error",
101        JOptionPane.ERROR_MESSAGE);
102     throw new IllegalStateException("Invalid operator.");
103   }

```

```

1     return array;
2 }
3
4     private void nakedSingle()
5 {
6     nakedSingleRow();
7     nakedSingleColumn();
8 }
9
10    private void nakedSingleRow()
11 {
12     for (int i = 0; i < size; i++)
13     {
14         nakedSingleRow(i);
15     }
16 }
17
18    private void nakedSingleRow(int row)
19 {
20     ArrayList<Integer>[] rowPossibleValues = getRowPossibleValues(row);
21     for (int i = 0; i < rowPossibleValues.length; i++)
22     {
23         if (rowPossibleValues[i].size() == 1)
24         {
25             nakedSingle(row, i);
26         }
27     }
28 }
29
30    private void nakedSingleColumn()
31 {
32     for (int i = 0; i < size; i++)
33     {
34         nakedSingleColumn(i);
35     }
36 }
37
38    private void nakedSingleColumn(int column)
39 {
40     ArrayList<Integer>[] columnPossibleValues =
41         getColumnPossibleValues(column);
42     for (int i = 0; i < columnPossibleValues.length; i++)
43     {
44         if (columnPossibleValues[i].size() == 1)
45         {
46             nakedSingle(i, column);
47         }
48     }
49 }
50
51    private void nakedSingle(int row, int column)
52 {
53     int value = possibleValues[row][column].get(0);
54     setCellValue(row, column, value);
55 }
56
57    private void nakedDouble()
58 {
59     nakedDoubleRow();
60     nakedDoubleColumn();
61 }
62
63    private void nakedDoubleRow()
64 {
65     for (int i = 0; i < size; i++)
66     {
67         nakedDoubleRow(i);
68     }
69 }
70
71    private void nakedDoubleRow(int row)
72 {
73     ArrayList<Integer>[] rowPossibleValues = getRowPossibleValues(row);
74     ArrayList<Integer> column2PossibleIndexes = new ArrayList<Integer>();
75     ArrayList<ArrayList<Integer>> column2PossibleValues =
76         new ArrayList<>();
77     ArrayList<ArrayList<Integer>> uniquePossibleValues = new ArrayList<>();
78     ArrayList<Integer> uniquePossibleValuesFrequency =
79         new ArrayList<Integer>();
80     for (int i = 0; i < rowPossibleValues.length; i++)
81     {
82         if (rowPossibleValues[i].size() == 2)
83         {
84             column2PossibleIndexes.add(i);
85             column2PossibleValues.add(rowPossibleValues[i]);
86         }
87     }
88     if (column2PossibleIndexes.size() >= 2
89         && column2PossibleValues.size() >= 2)
90     {
91         for (int i = 0; i < column2PossibleValues.size(); i++)
92         {
93             if (!uniquePossibleValues.contains(
94                 column2PossibleValues.get(i)))
95             {
96                 uniquePossibleValues.add(column2PossibleValues.get(i));
97             }
98         }
99         for (int i = 0; i < uniquePossibleValues.size(); i++)
100        {
101            uniquePossibleValuesFrequency.add(Collections.frequency(
102                column2PossibleValues, uniquePossibleValues.get(i)));
103        }
104    }
105 }
```

```

1     for (int i = 0; i < uniquePossibleValuesFrequency.size(); i++)
2     {
3         if (uniquePossibleValuesFrequency.get(i) == 2)
4         {
5             ArrayList<Integer> doublePossibleValues =
6                 uniquePossibleValues.get(i);
7             ArrayList<Integer> doublePossibleIndexes =
8                 new ArrayList<Integer>();
9             for (int j = 0; j < column2PossibleValues.size(); j++)
10            {
11                if (column2PossibleValues.get(j).equals(
12                    uniquePossibleValues.get(i)))
13                {
14                    doublePossibleIndexes.add(
15                        column2PossibleIndexes.get(j));
16                }
17            }
18            nakedDoubleRow(row, doublePossibleValues,
19                            doublePossibleIndexes);
20        }
21    }
22 }
23
24
25 private void nakedDoubleRow(int row,
26     ArrayList<Integer> doublePossibleValues,
27     ArrayList<Integer> doublePossibleIndexes)
28 {
29     for (int i = 0; i < size; i++)
30     {
31         if (!doublePossibleIndexes.contains(i))
32         {
33             possibleValues[row][i].removeAll(doublePossibleValues);
34         }
35     }
36 }
37
38 private void nakedDoubleColumn()
39 {
40     for (int i = 0; i < size; i++)
41     {
42         nakedDoubleColumn(i);
43     }
44 }
45
46 private void nakedDoubleColumn(int column)
47 {
48     ArrayList<Integer>[] columnPossibleValues =
49         getColumnPossibleValues(column);
50     ArrayList<Integer> row2PossibleIndexes = new ArrayList<Integer>();
51     ArrayList<ArrayList<Integer>> row2PossibleValues = new ArrayList<ArrayList<Integer>>();
52     ArrayList<ArrayList<Integer>> uniquePossibleValues = new ArrayList<ArrayList<Integer>>();
53     ArrayList<Integer> uniquePossibleValuesFrequency =
54         new ArrayList<Integer>();
55     for (int i = 0; i < columnPossibleValues.length; i++)
56     {
57         if (columnPossibleValues[i].size() == 2)
58         {
59             row2PossibleIndexes.add(i);
60             row2PossibleValues.add(columnPossibleValues[i]);
61         }
62     }
63     if (row2PossibleIndexes.size() >= 2
64         && row2PossibleValues.size() >= 2)
65     {
66         for (int i = 0; i < row2PossibleValues.size(); i++)
67         {
68             if (!uniquePossibleValues.contains(
69                 row2PossibleValues.get(i)))
70             {
71                 uniquePossibleValues.add(row2PossibleValues.get(i));
72             }
73         }
74         for (int i = 0; i < uniquePossibleValues.size(); i++)
75         {
76             uniquePossibleValuesFrequency.add(Collections.frequency(
77                 row2PossibleValues, uniquePossibleValues.get(i)));
78         }
79         for (int i = 0; i < uniquePossibleValuesFrequency.size(); i++)
80         {
81             if (uniquePossibleValuesFrequency.get(i) == 2)
82             {
83                 ArrayList<Integer> doublePossibleValues =
84                     uniquePossibleValues.get(i);
85                 ArrayList<Integer> doublePossibleIndexes =
86                     new ArrayList<Integer>();
87                 for (int j = 0; j < row2PossibleValues.size(); j++)
88                 {
89                     if (row2PossibleValues.get(j).equals(
90                         uniquePossibleValues.get(i)))
91                     {
92                         doublePossibleIndexes.add(
93                             row2PossibleIndexes.get(j));
94                     }
95                 }
96                 nakedDoubleColumn(column, doublePossibleValues,
97                                   doublePossibleIndexes);
98             }
99         }
100    }
101 }
102
103 private void nakedDoubleColumn(int column,

```

```

1      ArrayList<Integer> doublePossibleValues ,
2      ArrayList<Integer> doublePossibleIndexes)
3  {
4      for (int i = 0; i < size; i++)
5      {
6          if (!doublePossibleIndexes.contains(i))
7          {
8              possibleValues[i][column].removeAll(doublePossibleValues);
9          }
10     }
11 }
12
13 private void hiddenSingle()
14 {
15     hiddenSingleRow();
16     hiddenSingleColumn();
17 }
18
19 private void hiddenSingleRow()
20 {
21     for (int i = 0; i < size; i++)
22     {
23         hiddenSingleRow(i);
24     }
25 }
26
27 private void hiddenSingleRow(int row)
28 {
29     ArrayList<Integer>[] rowPossibleValues = getRowPossibleValues(row);
30     int[] possibleValuesFrequency = new int[size];
31     ArrayList<Integer> columnValues = new ArrayList<Integer>();
32     ArrayList<Integer> columnIndexes = new ArrayList<Integer>();
33     for (ArrayList<Integer> rowPossibleValue : rowPossibleValues)
34     {
35         for (int i = 1; i <= possibleValuesFrequency.length; i++)
36         {
37             if (rowPossibleValue.contains(i))
38             {
39                 possibleValuesFrequency[i - 1]++;
40             }
41         }
42     }
43     for (int i = 0; i < possibleValuesFrequency.length; i++)
44     {
45         if (possibleValuesFrequency[i] == 1)
46         {
47             columnValues.add(i + 1);
48         }
49     }
50     for (int i = 0; i < columnValues.size(); i++)
51     {
52         for (int j = 0; j < rowPossibleValues.length; j++)
53         {
54             if (rowPossibleValues[j].contains(columnValues.get(i)))
55             {
56                 columnIndexes.add(j);
57             }
58         }
59     }
60     for (int i = 0; i < columnValues.size(); i++)
61     {
62         if (rowPossibleValues[columnIndexes.get(i)].size() >= 2)
63         {
64             hiddenSingle(row, columnIndexes.get(i), columnValues.get(i));
65         }
66     }
67 }
68
69 private void hiddenSingleColumn()
70 {
71     for (int i = 0; i < size; i++)
72     {
73         hiddenSingleColumn(i);
74     }
75 }
76
77 private void hiddenSingleColumn(int column)
78 {
79     ArrayList<Integer>[] columnPossibleValues
80         = getColumnPossibleValues(column);
81     int[] possibleValuesFrequency = new int[size];
82     ArrayList<Integer> rowValues = new ArrayList<Integer>();
83     ArrayList<Integer> rowIndexes = new ArrayList<Integer>();
84     for (ArrayList<Integer> columnPossibleValue : columnPossibleValues)
85     {
86         for (int i = 1; i <= possibleValuesFrequency.length; i++)
87         {
88             if (columnPossibleValue.contains(i))
89             {
90                 possibleValuesFrequency[i - 1]++;
91             }
92         }
93     }
94     for (int i = 0; i < possibleValuesFrequency.length; i++)
95     {
96         if (possibleValuesFrequency[i] == 1)
97         {
98             rowValues.add(i + 1);
99         }
100    }
101   for (int i = 0; i < rowValues.size(); i++)
102   {
103       for (int j = 0; j < columnPossibleValues.length; j++)
104       {
105           if (rowValues.get(i) == columnIndexes.get(j))
106           {
107               columnPossibleValues[j].removeAll(columnPossibleValues[i]);
108           }
109       }
110   }
111 }
```

```

1      {
2          if (columnPossibleValues[j].contains(rowValues.get(i)))
3          {
4              rowIndexes.add(j);
5          }
6      }
7  }
8  for (int i = 0; i < rowValues.size(); i++)
9  {
10     if (columnPossibleValues[rowIndexes.get(i)].size() >= 2)
11     {
12         hiddenSingle(rowIndexes.get(i), column, rowValues.get(i));
13     }
14 }
15 }
16
17 private void hiddenSingle(int row, int column, int value)
18 {
19     setCellValue(row, column, value);
20 }
21
22 private void setCellValue(int row, int column, int value)
23 {
24     grid.solverSetCellValue(row, column, value);
25     removePossibleValues(row, column, value);
26 }
27
28 private void removePossibleValues(int row, int column, int value)
29 {
30     possibleValues[row][column].clear();
31     for (int i = 0; i < size; i++)
32     {
33         if (possibleValues[i][column].contains(value))
34         {
35             possibleValues[i][column].remove(
36                 possibleValues[i][column].indexOf(value));
37         }
38     }
39     for (int i = 0; i < size; i++)
40     {
41         if (possibleValues[row][i].contains(value))
42         {
43             possibleValues[row][i].remove(
44                 possibleValues[row][i].indexOf(value));
45         }
46     }
47 }
48
49 private void removeImpossibleValuesCage(Cage cage,
50     ArrayList<Integer> values)
51 {
52     for (int i = 0; i < cage.getSize(); i++)
53     {
54         removeImpossibleValuesCell(cage.getCells().get(i).getRow(),
55             cage.getCells().get(i).getColumn(), values);
56     }
57 }
58
59 private void removeImpossibleValuesCell(int row, int column,
60     ArrayList<Integer> values)
61 {
62     possibleValues[row][column].retainAll(values);
63 }
64
65 private ArrayList<Integer> createRetainAllArray()
66 {
67     ArrayList<Integer> array = new ArrayList<Integer>();
68     for (int i = 1; i <= size; i++)
69     {
70         array.add(i);
71     }
72     return array;
73 }
74
75 public ArrayList<ArrayList<Integer>> getGridArrayList()
76 {
77     ArrayList<ArrayList<Integer>> gridArrayList = new ArrayList<>();
78     for (int i = 0; i < size; i++)
79     {
80         ArrayList<Integer> gridArrayListRow = new ArrayList<Integer>();
81         for (int j = 0; j < size; j++)
82         {
83             gridArrayListRow.add(grid.getCellValue(i, j));
84         }
85         gridArrayList.add(gridArrayListRow);
86     }
87     return gridArrayList;
88 }
89
90 public Grid getGrid()
91 {
92     return grid;
93 }
94
95 public Grid getSolution()
96 {
97     return solution;
98 }
99
100 private void printGrid()
101 {
102     Cell[][] cells = grid.getGridContents();
103     for (int i = 0; i < size; i++)
104     {
105         for (int j = 0; j < size; j++)
106         {
107             System.out.print(cells[i][j] + " ");
108         }
109         System.out.println();
110     }
111 }
112
113 }
```

```

1   {
2     for (int j = 0; j < size; j++)
3     {
4       System.out.print(cells[i][j].getValue() + " ");
5     }
6     System.out.println("");
7   }
8   System.out.println("");
9 }
10
11 private void printPossibleValues()
12 {
13   for (int i = 0; i < possibleValues.length; i++)
14   {
15     for (int j = 0; j < possibleValues[i].length; j++)
16     {
17       System.out.println("Row " + i + ", Column " + j);
18       for (int k = 0; k < possibleValues[i][j].size(); k++)
19       {
20         System.out.print(possibleValues[i][j].get(k) + " ");
21       }
22       System.out.println("");
23     }
24   }
25   System.out.println("");
26 }
27 System.out.println("");
28 }
29 }
```

30 D.7 SolverGenetic.java

Listing D.7: SolverGenetic.java

```

31 package model;
32
33 import java.util.ArrayList;
34 import java.util.Collections;
35 import java.util.Random;
36
37 public class SolverGenetic
38 {
39
40   private final Grid grid;
41   private final Integer size;
42   private final boolean[][] isCellFixed;
43   private final Random randomGenerator;
44   private final Integer generations;
45   private final Integer populationSize;
46   private final Double elitismRate;
47   private final Double mutationRate;
48   private final Double crossoverRate;
49   private Grid solution;
50   private ArrayList<Chromosome> currentGeneration =
51     new ArrayList<Chromosome>();
52   private ArrayList<Chromosome> nextGeneration = new ArrayList<Chromosome>();
53
54   public SolverGenetic(Grid grid, Integer generations,
55     Integer populationSize, double elitismRate, double crossoverRate,
56     double mutationRate)
57   {
58     this.grid = grid;
59     this.size = grid.getSize();
60     this.generations = generations;
61     this.populationSize = populationSize;
62     this.elitismRate = elitismRate;
63     this.crossoverRate = crossoverRate;
64     this.mutationRate = mutationRate;
65     this.isCellFixed = generateIsCellFixedArray();
66     this.randomGenerator = new Random();
67     generatePopulation();
68     for (int i = 0; i < currentGeneration.size(); i++)
69     {
70       printGrid(currentGeneration.get(i).getGrid().getGridContents());
71       System.out.println(currentGeneration.get(i).getFitness());
72     }
73   }
74
75   public boolean solve()
76   {
77     for (int i = 0; i < generations; i++)
78     {
79       solveLoop();
80       sortChromosomes();
81       for (int j = 0; j < populationSize; j++)
82       {
83         printGrid(
84           currentGeneration.get(j).getGrid().getGridContents());
85         System.out.println(currentGeneration.get(j).getFitness());
86         if (currentGeneration.get(j).getFitness() == 1.0)
87         {
88           this.solution = currentGeneration.get(j).getGrid();
89           return true;
90         }
91       }
92     }
93   }
94   return false;
95 }
```

```

1  private void solveLoop()
2  {
3      int elitismNumber = (int) Math.round(populationSize * elitismRate);
4      int mutationNumber = (int) Math.round(populationSize * mutationRate);
5      int crossoverNumber
6          = (int) Math.round((populationSize * crossoverRate) / 2);
7      sortChromosomes();
8      for (int i = 0; i < populationSize; i++)
9      {
10         printGrid(currentGeneration.get(i).getGrid().getGridContents());
11         System.out.println(currentGeneration.get(i).getFitness());
12     }
13     for (int i = 0; i < elitismNumber; i++)
14     {
15         if (!nextGeneration.contains(currentGeneration.get(i)))
16         {
17             nextGeneration.add(cloneChromosome(currentGeneration.get(i)));
18         }
19     }
20     for (int i = 0; i < mutationNumber; i++)
21     {
22         Chromosome parent = randomSelection(currentGeneration);
23         nextGeneration.add(mutation(parent));
24     }
25     for (int i = 0; i < crossoverNumber; i++)
26     {
27         nextGeneration.addAll(crossover(randomSelection(currentGeneration),
28                                     randomSelection(currentGeneration)));
29     }
30     if (nextGeneration.size() < populationSize)
31     {
32         while (nextGeneration.size() < populationSize)
33         {
34             nextGeneration.add(randomSelection(currentGeneration));
35         }
36     }
37     if (nextGeneration.size() > populationSize)
38     {
39         int extraChromosomes = nextGeneration.size() - populationSize;
40         for (int i = 0; i < extraChromosomes; i++)
41         {
42             int index = randomGenerator.nextInt(nextGeneration.size());
43             nextGeneration.remove(index);
44         }
45     }
46     currentGeneration = nextGeneration;
47     nextGeneration = new ArrayList<Chromosome>();
48 }
49
50 private boolean[][] generateIsCellFixedArray()
51 {
52     boolean[][] array = new boolean[size][size];
53     for (int i = 0; i < size; i++)
54     {
55         for (int j = 0; j < size; j++)
56         {
57             array[i][j] = grid.getCellValue(i, j) != null;
58         }
59     }
60     return array;
61 }
62
63 private void generatePopulation()
64 {
65     for (int i = 0; i < populationSize; i++)
66     {
67         currentGeneration.add(generateChromosome());
68     }
69 }
70
71 private Chromosome generateChromosome()
72 {
73     Grid chromosomeGrid = new Grid(size, grid.getNumberofCages(),
74                                     grid.getCageCells(), grid.getCageObjectives());
75     for (int i = 0; i < size; i++)
76     {
77         for (int j = 0; j < size; j++)
78         {
79             if (grid.getCellValue(i, j) != null)
80             {
81                 chromosomeGrid.solverSetCellValue(i, j,
82                                                 grid.getCellValue(i, j));
83             }
84         }
85     }
86     for (int i = 0; i < size; i++)
87     {
88         for (int j = 0; j < size; j++)
89         {
90             if (!grid.getRow(i).contains(j + 1))
91             {
92                 int index = randomGenerator.nextInt(size);
93                 while (chromosomeGrid.getCellValue(i, index) != null)
94                 {
95                     index = randomGenerator.nextInt(size);
96                 }
97                 chromosomeGrid.solverSetCellValue(i, index, j + 1);
98             }
99         }
100    }
101    Chromosome c = new Chromosome(chromosomeGrid);
102    return c;
103}

```

```

1  private void sortChromosomes()
2  {
3      Collections.sort(currentGeneration,
4          new ChromosomeComparator().reversed());
5  }
6
7  private Chromosome randomSelection(ArrayList<Chromosome> chromosomes)
8  {
9      int randomIndex = randomGenerator.nextInt(chromosomes.size());
10     Chromosome c = cloneChromosome(chromosomes.get(randomIndex));
11     return c;
12 }
13
14 private Chromosome cloneChromosome(Chromosome c)
15 {
16     Grid gridClone = new Grid(size, grid.getNumberOfCages(),
17         grid.getCageCells(), grid.getCageObjectives());
18     for (int i = 0; i < size; i++)
19     {
20         for (int j = 0; j < size; j++)
21         {
22             gridClone.solverSetCellValue(i, j,
23                 c.getGrid().getCellValue(i, j));
24         }
25     }
26     Chromosome chromosomeClone = new Chromosome(gridClone);
27     return chromosomeClone;
28 }
29
30
31 private ArrayList<Chromosome> crossover(Chromosome parent1,
32     Chromosome parent2)
33 {
34     while (parent1 == parent2 || parent1.equals(parent2))
35     {
36         parent1 = randomSelection(currentGeneration);
37         parent2 = randomSelection(currentGeneration);
38     }
39     Grid childGrid1 = new Grid(size, grid.getNumberOfCages(),
40         grid.getCageCells(), grid.getCageObjectives());
41     Grid childGrid2 = new Grid(size, grid.getNumberOfCages(),
42         grid.getCageCells(), grid.getCageObjectives());
43     for (int i = 0; i < size; i++)
44     {
45         int randomIndex = randomGenerator.nextInt(2);
46         for (int j = 0; j < size; j++)
47         {
48             if (randomIndex == 0)
49             {
50                 childGrid1.solverSetCellValue(i, j,
51                     parent1.getGrid().getCellValue(i, j));
52                 childGrid2.solverSetCellValue(i, j,
53                     parent2.getGrid().getCellValue(i, j));
54             }
55             else
56             {
57                 childGrid1.solverSetCellValue(i, j,
58                     parent2.getGrid().getCellValue(i, j));
59                 childGrid2.solverSetCellValue(i, j,
60                     parent1.getGrid().getCellValue(i, j));
61             }
62         }
63     }
64     ArrayList<Chromosome> childChromosomes = new ArrayList<Chromosome>();
65     Chromosome child1 = new Chromosome(childGrid1);
66     Chromosome child2 = new Chromosome(childGrid2);
67     childChromosomes.add(child1);
68     childChromosomes.add(child2);
69     return childChromosomes;
70 }
71
72 private Chromosome mutation(Chromosome parent)
73 {
74     Grid childGrid = new Grid(size, grid.getNumberOfCages(),
75         grid.getCageCells(), grid.getCageObjectives());
76     int randomRow = randomGenerator.nextInt(size);
77     int randomColumn1 = randomGenerator.nextInt(size);
78     int randomColumn2 = randomGenerator.nextInt(size);
79     while (isCellFixed[randomRow][randomColumn1] == true
80         || isCellFixed[randomRow][randomColumn2] == true
81         || randomColumn1 == randomColumn2)
82     {
83         randomRow = randomGenerator.nextInt(size);
84         randomColumn1 = randomGenerator.nextInt(size);
85         randomColumn2 = randomGenerator.nextInt(size);
86     }
87     for (int i = 0; i < size; i++)
88     {
89         if (i == randomRow)
90         {
91             childGrid.solverSetCellValue(i, randomColumn1,
92                 parent.getGrid().getCellValue(i, randomColumn2));
93             childGrid.solverSetCellValue(i, randomColumn2,
94                 parent.getGrid().getCellValue(i, randomColumn1));
95         }
96         for (int j = 0; j < size; j++)
97         {
98             if (childGrid.getCellValue(i, j) == null)
99             {
100                 childGrid.solverSetCellValue(i, j,
101                     parent.getGrid().getCellValue(i, j));
102             }
103         }
104     }
105 }
```

```

1     }
2     Chromosome child = new Chromosome(childGrid);
3     return child;
4 }
5
6 public Grid getGrid()
7 {
8     return grid;
9 }
10
11 public Grid getSolution()
12 {
13     return solution;
14 }
15
16 private void printGrid(Cell[][][] cells)
17 {
18     for (int i = 0; i < size; i++)
19     {
20         for (int j = 0; j < size; j++)
21         {
22             System.out.print(cells[i][j].getValue() + " ");
23         }
24         System.out.println("");
25     }
26     System.out.println("");
27 }
28 }
29 }
```

30 D.8 Chromosome.java

Listing D.8: Chromosome.java

```

31 package model;
32
33 import java.util.Comparator;
34
35 public class Chromosome
36 {
37
38     private final Grid grid;
39     private final int size;
40     private final double fitness;
41
42     public Chromosome(Grid grid)
43     {
44         this.grid = grid;
45         this.size = grid.getSize();
46         this.fitness = setFitness();
47     }
48
49     private double setFitness()
50     {
51         double numberValidCells = 0;
52         double numberCells = size * size;
53         for (int i = 0; i < size; i++)
54         {
55             for (int j = 0; j < size; j++)
56             {
57                 if (grid.solverIsCellValueValid(i, j) == true)
58                 {
59                     numberValidCells++;
60                 }
61             }
62         }
63         double value = numberValidCells / numberCells;
64         return value;
65     }
66
67     public double getFitness()
68     {
69         return fitness;
70     }
71
72     public Grid getGrid()
73     {
74         return grid;
75     }
76 }
77
78 class ChromosomeComparator implements Comparator<Chromosome>
79 {
80
81     @Override
82     public int compare(Chromosome c1, Chromosome c2)
83     {
84         if (c1.getFitness() - c2.getFitness() > 0)
85         {
86             return 1;
87         }
88         else if (c1.getFitness() - c2.getFitness() < 0)
89         {
90             return -1;
91         }
92         else
93         {
94             return 0;
95         }
96     }
97 }
```

```

1     }
2     }
3   }
4 }
```

5 D.9 Controller.java

Listing D.9: Controller.java

```

6 package controller;
7
8 import model.Grid;
9 import model.Cage;
10 import model.Cell;
11
12 public class Controller
13 {
14
15     private Grid g;
16
17     public Controller(Integer size, Integer numberOfCages,
18                       Integer [][] cageCells, String [] cageObjectives)
19     {
20         g = new Grid(size, numberOfCages, cageCells, cageObjectives);
21     }
22
23     public boolean setCellValue(int row, int column, int value)
24     {
25         return g.setCellValue(row, column, value);
26     }
27
28     public void unsetCellValue(int row, int column)
29     {
30         g.unsetCellValue(row, column);
31     }
32
33     public Boolean checkGrid()
34     {
35         return g.checkGrid();
36     }
37
38     public Integer getCellValue(int row, int column)
39     {
40         return g.getCellValue(row, column);
41     }
42
43     public Integer getSize()
44     {
45         return g.getSize();
46     }
47
48     public Integer getNumberOfCages()
49     {
50         return g.getNumberOfCages();
51     }
52
53     public Integer [][] getCageCells()
54     {
55         return g.getCageCells();
56     }
57
58     public String [] getCageObjectives()
59     {
60         return g.getCageObjectives();
61     }
62
63     public int getCageTargetNumber(int cageID)
64     {
65         return g.getCages () [cageID].getTargetNumber ();
66     }
67
68     public char getCageOperator(int cageID)
69     {
70         return g.getCages () [cageID].getOperator ();
71     }
72
73     public Cell [][] getGrid()
74     {
75         return g.getGridContents ();
76     }
77
78     public Cage [] getCages()
79     {
80         return g.getCages ();
81     }
82
83     public Grid getGame()
84     {
85         return g.getGame ();
86     }
87
88 }
```

89 D.10 Calcudoku.java

Listing D.10: Calcudoku.java

```

1 package view;
2
3 import controller.Controller;
4 import java.awt.Dimension;
5 import java.awt.EventQueue;
6 import java.awt.event.ActionEvent;
7 import java.awt.event.WindowAdapter;
8 import java.awt.event.WindowEvent;
9 import java.io.File;
10 import java.io.FileNotFoundException;
11 import java.util.NoSuchElementException;
12 import java.util.Scanner;
13 import javax.swing.JFileChooser;
14 import javax.swing.JFrame;
15 import javax.swing.JMenu;
16 import javax.swing.JMenuBar;
17 import javax.swing.JMenuItem;
18 import javax.swing.JOptionPane;
19 import javax.swing.filechooser.FileFilter;
20
21 public class Calcudoku extends JFrame
22 {
23
24     private File puzzleFile;
25     private Integer size;
26     private Integer numberOfCages;
27     private Integer[][][] cageCells;
28     private String[] cageObjectives;
29     private Controller c;
30     private final JMenuBar menuBar;
31     private final JMenu menuFile;
32     private final JMenu menuSolve;
33     private final JMenuItem menuItemLoad;
34     private final JMenuItem menuItemReset;
35     private final JMenuItem menuItemClose;
36     private final JMenuItem menuItemCheck;
37     private final JMenuItem menuItemExit;
38     private final JMenuItem menuItemBacktracking;
39     private final JMenuItem menuItemHybridGenetic;
40     private final JMenuItem menuItemGeneticParameters;
41     private final JFileChooser fileChooser;
42     private GUI gui;
43
44     public Calcudoku()
45     {
46         this.menuBar = new JMenuBar();
47         this.menuFile = new JMenu();
48         this.menuSolve = new JMenu();
49         this.menuItemLoad = new JMenuItem();
50         this.menuItemReset = new JMenuItem();
51         this.menuItemClose = new JMenuItem();
52         this.menuItemCheck = new JMenuItem();
53         this.menuItemExit = new JMenuItem();
54         this.menuItemBacktracking = new JMenuItem();
55         this.menuItemHybridGenetic = new JMenuItem();
56         this.menuItemGeneticParameters = new JMenuItem();
57         this.fileChooser = new JFileChooser();
58         initComponents();
59     }
60
61     public static void main(String args[])
62     {
63         EventQueue.invokeLater(() ->
64         {
65             new Calcudoku().setVisible(true);
66         });
67     }
68
69     private void initComponents()
70     {
71         this.setTitle("Calcudoku");
72         this.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
73         this.setMinimumSize(new Dimension(216, 216));
74         this.setLocationRelativeTo(null);
75         this.setResizable(false);
76         initWindowListener();
77         initActionListeners();
78         initMenu();
79         initMenuBar();
80         this.validate();
81         this.revalidate();
82         this.pack();
83     }
84
85     private void initWindowListener()
86     {
87         this.addWindowListener(new WindowListener(this));
88     }
89
90     private void initActionListeners()
91     {
92         this.fileChooser.setFileFilter(new PuzzleFileFilter());
93         this.menuItemLoad.addActionListener(this::menuItemLoadActionPerformed);
94         this.menuItemReset.addActionListener(
95             this::menuItemResetActionPerformed);
96         this.menuItemCheck.addActionListener(
97             this::menuItemCheckActionPerformed);
98         this.menuItemClose.addActionListener(
99             this::menuItemCloseActionPerformed);
100        this.menuItemExit.addActionListener(this::menuItemExitActionPerformed);
101        this.menuItemBacktracking.addActionListener(
102            this::menuItemBacktrackingActionPerformed);

```

```

1     this.menuItemHybridGenetic.addActionListener(
2         this::menuItemHybridGeneticActionPerformed);
3     this.menuItemGeneticParameters.addActionListener(
4         this::menuItemGeneticParametersActionPerformed);
5 }
6
7 private void initMenu()
8 {
9     File directory = new File(System.getProperty("user.dir"));
10    this.menuFile.setText("File");
11    this.menuSolve.setText("Solve");
12    this.menuItemLoad.setText("Load\u00a9Puzzle\u00a9File");
13    this.menuItemReset.setText("Reset\u00a9Puzzle");
14    this.menuItemClose.setText("Close\u00a9Puzzle\u00a9File");
15    this.menuItemCheck.setText("Check\u00a9Puzzle");
16    this.menuItemExit.setText("Exit");
17    this.menuItemBacktracking.setText("Backtracking");
18    this.menuItemHybridGenetic.setText("Hybrid\u00a9Genetic");
19    this.menuItemGeneticParameters.setText(
20        "Set\u00a9Genetic\u00a9Algorithm\u00a9Parameters");
21    this.fileChooser.setDialogTitle("Load\u00a9Puzzle\u00a9File");
22    fileChooser.setCurrentDirectory(directory);
23 }
24
25 private void initMenuBar()
26 {
27     this.menuFile.add(menuItemLoad);
28     this.menuFile.add(menuItemReset);
29     this.menuFile.add(menuItemClose);
30     this.menuFile.addSeparator();
31     this.menuFile.add(menuItemCheck);
32     this.menuFile.addSeparator();
33     this.menuFile.add(menuItemExit);
34     this.menuBar.add(menuFile);
35     this.menuSolve.add(menuItemBacktracking);
36     this.menuSolve.add(menuItemHybridGenetic);
37     this.menuSolve.addSeparator();
38     this.menuSolve.add(menuItemGeneticParameters);
39     this.menuBar.add(menuSolve);
40     this.setJMenuBar(menuBar);
41 }
42
43 private void menuItemLoadActionPerformed(ActionEvent evt)
44 {
45     if (fileChooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION)
46     {
47         if (c != null && gui != null)
48         {
49             if ( JOptionPane.showConfirmDialog(null,
50                 "Are\u00a9you\u00a9sure\u00a9you\u00a9want\u00a9to\u00a9load\u00a9another\u00a9puzzle\u00a9file?",
51                 "Load\u00a9Puzzle\u00a9File", JOptionPane.YES_NO_OPTION)
52                 == JOptionPane.YES_OPTION)
53             {
54                 selectPuzzleFile();
55             }
56         }
57         else
58         {
59             selectPuzzleFile();
60         }
61     }
62 }
63
64 private void selectPuzzleFile()
65 {
66     this.puzzleFile = fileChooser.getSelectedFile();
67     try
68     {
69         if (puzzleFile.getAbsolutePath().endsWith(".txt"))
70         {
71             try
72             {
73                 loadPuzzleFile(puzzleFile);
74             }
75             catch (IllegalStateException ise)
76             {
77                 resetFrame();
78                 clearVariables();
79                 JOptionPane.showMessageDialog(null,
80                     "Error\u00a9in\u00a9loading\u00a9puzzle\u00a9file.", "Error",
81                     JOptionPane.ERROR_MESSAGE);
82                 throw new IllegalStateException("Error\u00a9in\u00a9loading\u00a9"
83                     + "puzzle\u00a9file.");
84             }
85         }
86         else
87         {
88             resetFrame();
89             clearVariables();
90             JOptionPane.showMessageDialog(null, "Invalid\u00a9puzzle\u00a9file.",
91                 "Error", JOptionPane.ERROR_MESSAGE);
92             throw new IllegalStateException("Invalid\u00a9puzzle\u00a9file.");
93         }
94     }
95     catch (FileNotFoundException fnfe)
96     {
97         resetFrame();
98         clearVariables();
99         JOptionPane.showMessageDialog(null, "Puzzle\u00a9file\u00a9not\u00a9found.",
100            "Error", JOptionPane.ERROR_MESSAGE);
101        throw new IllegalStateException("Puzzle\u00a9file\u00a9not\u00a9found.");
102    }
103 }

```

```

1  private void menuItemResetActionPerformed(ActionEvent evt)
2  {
3      if (c == null || gui == null)
4      {
5          resetFrame();
6          JOptionPane.showMessageDialog(null, "Puzzle file not loaded.", "Error", JOptionPane.ERROR_MESSAGE);
7          throw new IllegalStateException("Puzzle file not loaded.");
8      }
9      else
10     {
11         if (JOptionPane.showConfirmDialog(null,
12             "Are you sure you want to reset this puzzle?", "Reset Puzzle",
13             JOptionPane.YES_NO_OPTION) == JOptionPane.YES_OPTION)
14         {
15             resetFrame();
16             this.c = new Controller(size, numberOfCages, cageCells,
17                 cageObjectives);
18             this.gui = new GUI(c);
19             this.getContentPane().add(gui);
20             this.validate();
21             this.revalidate();
22             this.pack();
23             this.setLocationRelativeTo(null);
24             this.setTitle("Calcudoku (" + puzzleFile.getName() + ")");
25         }
26     }
27 }
28
29
30
31 private void menuItemCheckActionPerformed(ActionEvent evt)
32 {
33     if (c == null || gui == null)
34     {
35         JOptionPane.showMessageDialog(null, "Puzzle file not loaded.", "Error", JOptionPane.ERROR_MESSAGE);
36         throw new IllegalStateException("Puzzle file not loaded.");
37     }
38     else
39     {
40         c.checkGrid();
41     }
42 }
43
44
45 private void menuItemCloseActionPerformed(ActionEvent evt)
46 {
47     if (c == null || gui == null)
48     {
49         JOptionPane.showMessageDialog(null, "Puzzle file not loaded.", "Error", JOptionPane.ERROR_MESSAGE);
50         throw new IllegalStateException("Puzzle file not loaded.");
51     }
52     else
53     {
54         if (JOptionPane.showConfirmDialog(null,
55             "Are you sure you want to close this puzzle file?", "Close Puzzle File",
56             JOptionPane.YES_NO_OPTION)
57             == JOptionPane.YES_OPTION)
58         {
59             resetFrame();
60             clearVariables();
61         }
62     }
63 }
64
65
66 private void menuItemExitActionPerformed(ActionEvent evt)
67 {
68     if (JOptionPane.showConfirmDialog(null,
69         "Are you sure you want to exit this application?", "Exit",
70         JOptionPane.YES_NO_OPTION) == JOptionPane.YES_OPTION)
71     {
72         destroyFrame();
73     }
74 }
75
76 private void menuItemBacktrackingActionPerformed(ActionEvent evt)
77 {
78     if (c == null || gui == null)
79     {
80         JOptionPane.showMessageDialog(null, "Puzzle file not loaded.", "Error", JOptionPane.ERROR_MESSAGE);
81         throw new IllegalStateException("Puzzle file not loaded.");
82     }
83     else
84     {
85         gui.solveBacktracking();
86     }
87 }
88
89
90 private void menuItemHybridGeneticActionPerformed(ActionEvent evt)
91 {
92     if (c == null || gui == null)
93     {
94         JOptionPane.showMessageDialog(null, "Puzzle file not loaded.", "Error", JOptionPane.ERROR_MESSAGE);
95         throw new IllegalStateException("Puzzle file not loaded.");
96     }
97     else
98     {
99         gui.solveHybridGenetic();
100    }
101 }
102
103 }
```

```

1  private void menuItemGeneticParametersActionPerformed(ActionEvent evt)
2  {
3      if (c == null || gui == null)
4      {
5          JOptionPane.showMessageDialog(null, "Puzzle file not loaded.",
6              "Error", JOptionPane.ERROR_MESSAGE);
7          throw new IllegalStateException("Puzzle file not loaded.");
8      }
9      else
10     {
11         GeneticParameters gp = new GeneticParameters(gui);
12         gp.setVisible(true);
13     }
14 }
15
16 private void loadPuzzleFile(File puzzleFile) throws FileNotFoundException
17 {
18     resetFrame();
19     clearVariables();
20     try
21     {
22         try (Scanner sc = new Scanner(puzzleFile))
23         {
24             this.size = sc.nextInt();
25             this.cageCells = new Integer[size][size];
26             this.numberOfCages = sc.nextInt();
27             this.cageObjectives = new String[numberOfCages];
28             for (int i = 0; i < size; i++)
29             {
30                 for (int j = 0; j < size; j++)
31                 {
32                     this.cageCells[i][j] = sc.nextInt();
33                 }
34             }
35             for (int i = 0; i < numberOfCages; i++)
36             {
37                 this.cageObjectives[i] = sc.next();
38             }
39             if (sc.hasNext())
40             {
41                 JOptionPane.showMessageDialog(null, "Invalid puzzle file.",
42                     "Error", JOptionPane.ERROR_MESSAGE);
43                 throw new IllegalStateException("Invalid puzzle file.");
44             }
45             sc.close();
46         }
47         this.c = new Controller(size, numberOfCages, cageCells,
48             cageObjectives);
49         this.gui = new GUI(c);
50         this.getContentPane().add(gui);
51         this.validate();
52         this.revalidate();
53         this.pack();
54         this.setLocationRelativeTo(null);
55         this.puzzleFile = puzzleFile;
56         this.setTitle("Calcudoku (" + puzzleFile.getName() + ")");
57     }
58     catch (NoSuchElementException nsee)
59     {
60         resetFrame();
61         clearVariables();
62         JOptionPane.showMessageDialog(null, "Invalid puzzle file.",
63             "Error", JOptionPane.ERROR_MESSAGE);
64         throw new IllegalStateException("Invalid puzzle file.");
65     }
66 }
67
68 private void resetFrame()
69 {
70     this.getContentPane().removeAll();
71     this.c = null;
72     this.setTitle("Calcudoku");
73     this.validate();
74     this.revalidate();
75     this.pack();
76     this.setLocationRelativeTo(null);
77 }
78
79 private void clearVariables()
80 {
81     this.puzzleFile = null;
82     this.size = null;
83     this.cageCells = null;
84     this.numberOfCages = null;
85     this.cageObjectives = null;
86 }
87
88 public void destroyFrame()
89 {
90     resetFrame();
91     clearVariables();
92     this.dispose();
93 }
94
95 }
96
97 class WindowListener extends WindowAdapter
98 {
99
100    private final Calcudoku frame;
101
102    public WindowListener(Calcudoku frame)
103    {

```

```

1     this.frame = frame;
2 }
3
4 @Override
5 public void windowClosing(WindowEvent e)
6 {
7     if ( JOptionPane.showConfirmDialog(null,
8             "Are you sure you want to exit the application?", "Exit",
9             JOptionPane.YES_NO_OPTION) == JOptionPane.YES_OPTION)
10    {
11        frame.destroyFrame();
12    }
13 }
14
15 @Override
16 public void windowClosed(WindowEvent e)
17 {
18     System.exit(0);
19 }
20 }
21
22 class PuzzleFileFilter extends FileFilter
23 {
24
25     @Override
26     public boolean accept(File puzzleFile)
27     {
28         return puzzleFile.isDirectory()
29                 || puzzleFile.getAbsolutePath().endsWith(".txt");
30     }
31 }
32
33 @Override
34 public String getDescription()
35 {
36     return "Text documents (*.txt)";
37 }
38 }
```

40 D.11 GUI.java

Listing D.11: GUI.java

```

41 package view;
42
43 import controller.Controller;
44 import java.awt.Color;
45 import java.awt.Dimension;
46 import java.awt.Font;
47 import java.awt.GridLayout;
48 import java.awt.Point;
49 import java.awt.event.ActionEvent;
50 import java.awt.event.ActionListener;
51 import java.awt.event.KeyEvent;
52 import java.awt.event.KeyListener;
53 import java.util.HashMap;
54 import java.util.Map;
55 import java.util.Objects;
56 import javax.swing.BorderFactory;
57 import javax.swing.JMenuItem;
58 import javax.swing.JOptionPane;
59 import javax.swing.JPanel;
60 import javax.swing.JPopupMenu;
61 import javax.swing.JTextField;
62 import javax.swing.event.DocumentEvent;
63 import javax.swing.event.DocumentListener;
64 import model.Cage;
65 import model.Cell;
66 import model.Grid;
67 import model.SolverBacktracking;
68 import model.SolverHybridGenetic;
69
70 public class GUI extends JPanel
71 {
72
73     private final Controller c;
74     private final Grid game;
75     private final Integer size;
76     private final Integer numberCages;
77     private final Integer[][] cageCells;
78     private final String[][] cageObjectives;
79     private final Cell[][] grid;
80     private final Cage[] cages;
81     private final JTextField[][] textFields;
82     private final Map<JTextField, Point> textFieldCoordinates;
83     private final CellTextFieldListener[][] cellTextFieldListeners;
84     private final Font font;
85     private final int cellSize;
86     private final int cellBorderWidth;
87     private final int cageBorderWidth;
88     private Integer generations;
89     private Integer populationSize;
90     private Double elitismRate;
91     private Double crossoverRate;
92     private Double mutationRate;
93
94     public GUI(Controller c)
95     {
```

```

1   this.c = c;
2   this.game = c.getGame();
3   this.size = c.getSize();
4   this.cageCells = c.getCageCells();
5   this.numberOfCages = c.getNumberOfCages();
6   this.cageObjectives = c.getCageObjectives();
7   this.grid = c.getGrid();
8   this.cages = c.getCages();
9   this.textFields = new JTextField[size][size];
10  this.textFieldCoordinates = new HashMap<>();
11  this.cellTextFieldListeners = new CellTextFieldListener[size][size];
12  this.font = new Font("Courier-New", Font.CENTER_BASELINE, 36);
13  this.cellSize = 72;
14  this.cellBorderWidth = 1;
15  this.cageBorderWidth = 3;
16  initComponents();
17 }
18
19 private void initComponents()
20 {
21     this.setPreferredSize(new Dimension(cellSize * size, cellSize * size));
22     this.setLayout(new GridLayout(size, size));
23     initTextFields();
24 }
25
26 public Controller getController()
27 {
28     return c;
29 }
30
31 public Integer getGridSize()
32 {
33     return size;
34 }
35
36 public Integer getNumberOfCages()
37 {
38     return numberOfCages;
39 }
40
41 public Integer[][] getCageCells()
42 {
43     return cageCells;
44 }
45
46 public String[] getCageObjectives()
47 {
48     return cageObjectives;
49 }
50
51 public Cell[][] getGrid()
52 {
53     return grid;
54 }
55
56 public Cage[] getCages()
57 {
58     return cages;
59 }
60
61 public void setGeneticAlgorithmParameters(Integer generations,
62                                         Integer populationSize, Double elitismRate,
63                                         Double crossoverRate,
64                                         Double mutationRate)
65 {
66     this.generations = generations;
67     this.populationSize = populationSize;
68     this.elitismRate = elitismRate;
69     this.crossoverRate = crossoverRate;
70     this.mutationRate = mutationRate;
71 }
72
73 private void initTextFields()
74 {
75     for (int y = 0; y < size; y++)
76     {
77         for (int x = 0; x < size; x++)
78         {
79             JTextField textField = new JTextField();
80             textField.addKeyListener(new CellKeyListener(this));
81             cellTextFieldListeners[y][x] = new CellTextFieldListener(c,
82                           textField, y, x);
83             textField.getDocument().addDocumentListener(
84                 cellTextFieldListeners[y][x]);
85             textFieldCoordinates.put(textField, new Point(x, y));
86             textFields[y][x] = textField;
87         }
88     }
89     for (int y = 0; y < size; y++)
90     {
91         for (int x = 0; x < size; x++)
92         {
93             JTextField textField = textFields[y][x];
94             int cageID;
95             int topBorderWidth;
96             int leftBorderWidth;
97             int bottomBorderWidth;
98             int rightBorderWidth;
99             if (y == 0)
100             {
101                 topBorderWidth = cageBorderWidth;
102                 if (Objects.equals(cageCells[y][x], cageCells[y + 1][x]))
103                 {
104                     bottomBorderWidth = cellBorderWidth;
105                 }
106             }
107         }
108     }
109 }
110
111 
```

```

1         }
2     else
3     {
4         bottomBorderWidth = cageBorderWidth;
5     }
6 }
7 else if (y == size - 1)
8 {
9     bottomBorderWidth = cageBorderWidth;
10    if (Objects.equals(cageCells[y][x], cageCells[y - 1][x]))
11    {
12        topBorderWidth = cellBorderWidth;
13    }
14    else
15    {
16        topBorderWidth = cageBorderWidth;
17    }
18 }
19 else
20 {
21     if (Objects.equals(cageCells[y][x], cageCells[y + 1][x]))
22     {
23         bottomBorderWidth = cellBorderWidth;
24     }
25     else
26     {
27         bottomBorderWidth = cageBorderWidth;
28     }
29     if (Objects.equals(cageCells[y][x], cageCells[y - 1][x]))
30     {
31         topBorderWidth = cellBorderWidth;
32     }
33     else
34     {
35         topBorderWidth = cageBorderWidth;
36     }
37 }
38 if (x == 0)
39 {
40     leftBorderWidth = cageBorderWidth;
41     if (Objects.equals(cageCells[y][x], cageCells[y][x + 1]))
42     {
43         rightBorderWidth = cellBorderWidth;
44     }
45     else
46     {
47         rightBorderWidth = cageBorderWidth;
48     }
49 }
50 else if (x == size - 1)
51 {
52     rightBorderWidth = cageBorderWidth;
53     if (Objects.equals(cageCells[y][x], cageCells[y][x - 1]))
54     {
55         leftBorderWidth = cellBorderWidth;
56     }
57     else
58     {
59         leftBorderWidth = cageBorderWidth;
60     }
61 }
62 else
63 {
64     if (Objects.equals(cageCells[y][x], cageCells[y][x + 1]))
65     {
66         rightBorderWidth = cellBorderWidth;
67     }
68     else
69     {
70         rightBorderWidth = cageBorderWidth;
71     }
72     if (Objects.equals(cageCells[y][x], cageCells[y][x - 1]))
73     {
74         leftBorderWidth = cellBorderWidth;
75     }
76     else
77     {
78         leftBorderWidth = cageBorderWidth;
79     }
80 }
81 cageID = grid[y][x].getCageID();
82 textField.setToolTipText(cages[cageID].getObjective());
83 textField.setBorder(BorderFactory.createMatteBorder(
84     topBorderWidth, leftBorderWidth, bottomBorderWidth,
85     rightBorderWidth, Color.BLACK));
86 textField.setFont(font);
87 textField.setHorizontalAlignment(JTextField.CENTER);
88 textField.setPreferredSize(new Dimension(cellSize, cellSize));
89 JPopupMenu popupMenu = new JPopupMenu();
90 for (int i = 1; i <= size; i++)
91 {
92     JMenuItem menuItem = new JMenuItem("" + i);
93     menuItem.addActionListener(new PopupMenuListener(textField,
94         i));
95 }
96 textField.add(popupMenu);
97 textField.setComponentPopupMenu(popupMenu);
98 }
99 }
100 for (int y = 0; y < size; y++)
101 {
102     for (int x = 0; x < size; x++)
103 {

```

```

1      {
2          this.add(textFields[y][x]);
3      }
4  }
5 }
6
7 public void solveBacktracking()
8 {
9     removeCellTextFieldListeners();
10    clearGrid();
11    Cell[][] solution;
12    float startTime = System.nanoTime();
13    float endTime;
14    float duration;
15    SolverBacktracking sb = new SolverBacktracking(game);
16    if (sb.solve() == true)
17    {
18        solution = sb.getSolution().getGridContents();
19        printGridToScreen(solution);
20        endTime = System.nanoTime();
21        duration = (endTime - startTime) / 1000000000;
22        System.out.println("The backtracking algorithm has successfully solved the puzzle." +
23                            + "\nTime elapsed: " + duration +
24                            + " seconds");
25        JOptionPane.showMessageDialog(null,
26                                    "The backtracking algorithm has successfully solved the puzzle." +
27                                    + "\nTime elapsed: " + duration +
28                                    + " seconds", "Information",
29                                    JOptionPane.INFORMATION_MESSAGE);
30    }
31    else
32    {
33        endTime = System.nanoTime();
34        duration = (endTime - startTime) / 1000000000;
35        System.out.println("The backtracking algorithm has failed to solve the puzzle." +
36                            + "\nTime elapsed: " + duration +
37                            + " seconds");
38        JOptionPane.showMessageDialog(null,
39                                    "The backtracking algorithm has failed to solve the puzzle." +
40                                    + "\nTime elapsed: " + duration +
41                                    + " seconds", "Information",
42                                    JOptionPane.INFORMATION_MESSAGE);
43    }
44    addCellTextFieldListeners();
45 }
46
47 public void solveHybridGenetic()
48 {
49     if (generations == null || populationSize == null ||
50         || elitismRate == null || crossoverRate == null ||
51         || mutationRate == null)
52     {
53         JOptionPane.showMessageDialog(null,
54                                     "Genetic algorithm parameters have not been set.", "Error",
55                                     JOptionPane.ERROR_MESSAGE);
56         throw new IllegalStateException(
57             "Genetic algorithm parameters have not been set.");
58     }
59     else
60     {
61         removeCellTextFieldListeners();
62         clearGrid();
63         Cell[][] solution;
64         float startTime = System.nanoTime();
65         float endTime;
66         float duration;
67         SolverHybridGenetic shg = new SolverHybridGenetic(game, generations,
68                                         populationSize, elitismRate, crossoverRate, mutationRate);
69         if (shg.solve() == true)
70         {
71             solution = shg.getSolution().getGridContents();
72             printGridToScreen(solution);
73             endTime = System.nanoTime();
74             duration = (endTime - startTime) / 1000000000;
75             System.out.println("The hybrid genetic algorithm has successfully solved the puzzle." +
76                                + "\nTime elapsed: " + duration +
77                                + " seconds");
78             JOptionPane.showMessageDialog(null,
79                                     "The hybrid genetic algorithm has successfully solved the puzzle." +
80                                     + "\nTime elapsed: " + duration +
81                                     + " seconds", "Information",
82                                     JOptionPane.INFORMATION_MESSAGE);
83         }
84         else
85         {
86             endTime = System.nanoTime();
87             duration = (endTime - startTime) / 1000000000;
88             System.out.println("The hybrid genetic algorithm has failed to solve the puzzle." +
89                                + "\nTime elapsed: " + duration +
90                                + " seconds");
91             JOptionPane.showMessageDialog(null,
92                                     "The hybrid genetic algorithm has failed to solve the puzzle." +
93                                     + "\nTime elapsed: " + duration +
94                                     + " seconds", "Information",
95                                     JOptionPane.INFORMATION_MESSAGE);
96         }
97         addCellTextFieldListeners();
98     }
99 }
100
101 private void printGridToScreen(Cell[][] grid)
102 {
103     for (int x = 0; x < size; x++)

```

```

1      {
2          for (int y = 0; y < size; y++)
3          {
4              String value = Integer.toString(grid[x][y].getValue());
5              textFields[x][y].setText(value);
6          }
7      }
8  }
9
10 private void clearGrid()
11 {
12     for (int i = 0; i < size; i++)
13     {
14         for (int j = 0; j < size; j++)
15         {
16             c.unsetCellValue(i, j);
17         }
18     }
19 }
20
21 private void addCellTextFieldListeners()
22 {
23     for (int x = 0; x < size; x++)
24     {
25         for (int y = 0; y < size; y++)
26         {
27             textFields[x][y].getDocument().addDocumentListener(
28                 cellTextFieldListeners[x][y]);
29         }
30     }
31 }
32
33 private void removeCellTextFieldListeners()
34 {
35     for (int x = 0; x < size; x++)
36     {
37         for (int y = 0; y < size; y++)
38         {
39             textFields[x][y].getDocument().removeDocumentListener(
40                 cellTextFieldListeners[x][y]);
41         }
42     }
43 }
44
45 public void moveCursor(JTextField textField, int keyCode)
46 {
47     Point coordinates = textFieldCoordinates.get(textField);
48     switch (keyCode)
49     {
50         case KeyEvent.VK_LEFT:
51             if (coordinates.x > 0)
52             {
53                 textFields[coordinates.y][coordinates.x - 1].requestFocus();
54             }
55             break;
56         case KeyEvent.VK_KP_LEFT:
57             if (coordinates.x > 0)
58             {
59                 textFields[coordinates.y][coordinates.x - 1].requestFocus();
60             }
61             break;
62         case KeyEvent.VK_UP:
63             if (coordinates.y > 0)
64             {
65                 textFields[coordinates.y - 1][coordinates.x].requestFocus();
66             }
67             break;
68         case KeyEvent.VK_KP_UP:
69             if (coordinates.y > 0)
70             {
71                 textFields[coordinates.y - 1][coordinates.x].requestFocus();
72             }
73             break;
74         case KeyEvent.VK_RIGHT:
75             if (coordinates.x < size - 1)
76             {
77                 textFields[coordinates.y][coordinates.x + 1].requestFocus();
78             }
79             break;
80         case KeyEvent.VK_KP_RIGHT:
81             if (coordinates.x < size - 1)
82             {
83                 textFields[coordinates.y][coordinates.x + 1].requestFocus();
84             }
85             break;
86         case KeyEvent.VK_DOWN:
87             if (coordinates.y < size - 1)
88             {
89                 textFields[coordinates.y + 1][coordinates.x].requestFocus();
90             }
91             break;
92         case KeyEvent.VK_KP_DOWN:
93             if (coordinates.y < size - 1)
94             {
95                 textFields[coordinates.y + 1][coordinates.x].requestFocus();
96             }
97             break;
98     }
99 }
100 }
101
102 class CellKeyListener implements KeyListener

```

```

1  {
2
3     private final GUI gui;
4
5     CellKeyListener(GUI gui)
6     {
7         this.gui = gui;
8     }
9
10    @Override
11    public void keyPressed(KeyEvent e)
12    {
13        int keyCode = e.getKeyCode();
14        JTextField textField = (JTextField) e.getSource();
15        switch (keyCode)
16        {
17            case KeyEvent.VK_LEFT :
18                e.consume();
19                gui.moveCursor(textField, KeyEvent.VK_LEFT);
20                break;
21            case KeyEvent.VK_UP :
22                e.consume();
23                gui.moveCursor(textField, KeyEvent.VK_UP);
24                break;
25            case KeyEvent.VK_RIGHT :
26                e.consume();
27                gui.moveCursor(textField, KeyEvent.VK_RIGHT);
28                break;
29            case KeyEvent.VK_DOWN :
30                e.consume();
31                gui.moveCursor(textField, KeyEvent.VK_DOWN);
32                break;
33            case KeyEvent.VK_KP_LEFT :
34                e.consume();
35                gui.moveCursor(textField, KeyEvent.VK_KP_LEFT);
36                break;
37            case KeyEvent.VK_KP_UP :
38                e.consume();
39                gui.moveCursor(textField, KeyEvent.VK_KP_UP);
40                break;
41            case KeyEvent.VK_KP_RIGHT :
42                e.consume();
43                gui.moveCursor(textField, KeyEvent.VK_KP_RIGHT);
44                break;
45            case KeyEvent.VK_KP_DOWN :
46                e.consume();
47                gui.moveCursor(textField, KeyEvent.VK_KP_DOWN);
48                break;
49        }
50    }
51
52    @Override
53    public void keyTyped(KeyEvent e)
54    {
55        JTextField textField = (JTextField) e.getSource();
56        char c = e.getKeyChar();
57        if (!Character.isDigit(c))
58        {
59            e.consume();
60        }
61        String gridSize = "" + gui.getGridSize();
62        int gridSizeDigits = gridSize.length();
63        if (textField.getText().length() >= gridSizeDigits)
64        {
65            e.consume();
66        }
67    }
68
69    @Override
70    public void keyReleased(KeyEvent e)
71    {
72    }
73 }
74 }
75 }
76
77 class PopupMenuListener implements ActionListener
78 {
79
80     private final JTextField textField;
81     private final int number;
82
83     PopupMenuListener(JTextField textField, int number)
84     {
85         this.textField = textField;
86         this.number = number;
87     }
88
89     @Override
90     public void actionPerformed(ActionEvent e)
91     {
92         textField.setText(number + "");
93     }
94 }
95
96
97 class CellTextFieldListener implements DocumentListener
98 {
99
100    private final Controller c;
101    private final JTextField textField;
102    private final int x;
103    private final int y;

```

```

1  public CellTextFieldListener(Controller c, JTextField textField, int x,
2      int y)
3  {
4      this.c = c;
5      this.textField = textField;
6      this.x = x;
7      this.y = y;
8  }
9
10
11 @Override
12 public void insertUpdate(DocumentEvent e)
13 {
14     Integer value = Integer.parseInt(textField.getText());
15     c.setCellValue(x, y, value);
16 }
17
18 @Override
19 public void removeUpdate(DocumentEvent e)
20 {
21     c.unsetCellValue(x, y);
22 }
23
24 @Override
25 public void changedUpdate(DocumentEvent e)
26 {
27     Integer value = Integer.parseInt(textField.getText());
28     c.unsetCellValue(x, y);
29     c.setCellValue(x, y, value);
30 }
31
32 }

```

33 D.12 GeneticParameters.java

Listing D.12: GeneticParameters.java

```

34 package view;
35
36 import java.awt.event.ActionEvent;
37 import javax.swing.GroupLayout;
38 import javax.swing.JButton;
39 import javax.swing.JFrame;
40 import javax.swing.JLabel;
41 import javax.swing.JOptionPane;
42 import javax.swing.JTextField;
43 import javax.swing.SwingConstants;
44 import javax.swing.WindowConstants;
45
46 public class GeneticParameters extends JFrame
47 {
48
49     private final GUI gui;
50     private final JLabel labelGenerations;
51     private final JLabel labelPopulation;
52     private final JLabel labelElitism;
53     private final JLabel labelCrossover;
54     private final JLabel labelMutation;
55     private final JTextField textFieldGenerations;
56     private final JTextField textFieldPopulation;
57     private final JTextField textFieldElitism;
58     private final JTextField textFieldCrossover;
59     private final JTextField textFieldMutation;
60     private final JButton buttonOK;
61     private final JButton buttonCancel;
62
63     /**
64      * Creates new form Test
65      */
66     public GeneticParameters(GUI gui)
67     {
68         this.gui = gui;
69         labelGenerations = new JLabel();
70         labelPopulation = new JLabel();
71         labelElitism = new JLabel();
72         labelCrossover = new JLabel();
73         labelMutation = new JLabel();
74         textFieldGenerations = new JTextField();
75         textFieldPopulation = new JTextField();
76         textFieldElitism = new JTextField();
77         textFieldCrossover = new JTextField();
78         textFieldMutation = new JTextField();
79         buttonOK = new JButton();
80         buttonCancel = new JButton();
81         initComponents();
82         this.setVisible(true);
83     }
84
85     private void initComponents()
86     {
87         this.setTitle("Set Genetic Algorithm Parameters");
88         this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
89         this.setLocationRelativeTo(null);
90         this.setResizable(false);
91         labelGenerations.setText("Generations");
92         labelPopulation.setText("Population Size");
93         labelElitism.setText("Elitism Rate");
94         labelCrossover.setText("Crossover Rate");
95         labelMutation.setText("Mutation Rate");

```



```
1         . addContainerGap( GroupLayout.DEFAULT_SIZE,
2                         Short.MAX_VALUE ) );
3     this.validate();
4     this.revalidate();
5     this.pack();
6     this.setLocationRelativeTo( null );
7 }
8
9 private void buttonOKActionPerformed( ActionEvent evt )
10 {
11     try
12     {
13         Integer generations = Integer.parseInt( textFieldGenerations.getText() );
14         Integer population = Integer.parseInt( textFieldPopulation.getText() );
15         Double elitism = Double.parseDouble( textFieldElitism.getText() );
16         Double crossover = Double.parseDouble( textFieldCrossover.getText() );
17         Double mutation = Double.parseDouble( textFieldMutation.getText() );
18         gui.setGeneticAlgorithmParameters( generations, population, elitism,
19                                           crossover, mutation );
20         destroyFrame();
21     }
22     catch ( NumberFormatException nfe )
23     {
24         JOptionPane.showMessageDialog( null, " Invalid number format. ",
25                                       " Error ", JOptionPane.ERROR_MESSAGE );
26         throw new IllegalStateException( " Invalid number format. " );
27     }
28 }
29
30 private void buttonCancelActionPerformed( ActionEvent evt )
31 {
32     destroyFrame();
33 }
34
35 private void destroyFrame()
36 {
37     this.getContentPane().removeAll();
38     this.validate();
39     this.revalidate();
40     this.pack();
41     this.setLocationRelativeTo( null );
42     this.dispose();
43 }
44 }
45 }
```