Perjanjian No: III/LPPM/2014-03/40-p

PENGEMBANGAN PROTOTIPE SUDOKU *SOLVER* BERBASIS SISTEM MULTI AGEN

Disusun Oleh:
Luciana Abednego, S.Kom., M.T.
Dr.rer.nat. Cecilia E. Nugraheni. S.T., M.T.

Lembaga Penelitian dan Pengabdian kepada Masyarakat
Universitas Katolik Parahyangan
2014

ABSTRAK

Sudoku adalah sejenis teka-teki logika yang tujuan akhirnya adalah mengisikan angka-angka 1 sampai dengan 9 ke dalam suatu kotak berukuran 9x9. Kotak ini memiliki 9 sub-kotak berukuran 3x3. Syarat teka-teki ini adalah tidak ada angka yang berulang pada setiap baris, kolom, atau sub-kotak. Teka-teki Sudoku termasuk ke dalam permasalahan kombinatorial (NP complete). Solusi untuk teka-teki ini dapat dicari dengan bermacam-macam cara seperti algoritma genetik, heuristik, dan sebagainya.

Penelitian ini merupakan penelitian lanjutan dari penelitian sebelumnya yang berjudul "Pemodelan Permainan Sudoku sebagai *Block-World Problem*" yang telah menghasilkan tiga model untuk Sudoku *solver*. Pada penelitian tersebut *Block-World Problem* dipandang sebagai sistem multi agen berparameter. Model ditulis secara formal dengan notasi TLA+.

Pada penelitian ini diusulkan satu model baru yang merupakan varian dari model yang sudah ada. Penelitian ini juga menghasilkan sebuah *framework* Sudoku *Solver* yaitu program yang dapat digunakan untuk membangun sebuah *solver* berdasarkan model tertentu. Dengan *framework* tersebut dikembangkan *prototype* Sudoku *solver* berdasarkan model yang telah dihasilkan. Keempat model yang telah diusulkan telah diimplementasikan pada *framework* tersebut. Eksperimen terhadap kinerja dari seluruh *solver* dilakukan dengan melihat keberhasilannya dalam mencari solusi untuk sekumpulan soal Sudoku dilihat dari sisi waktu dan banyaknya permainan yang dapat diselesaikan. .

DAFTAR ISI

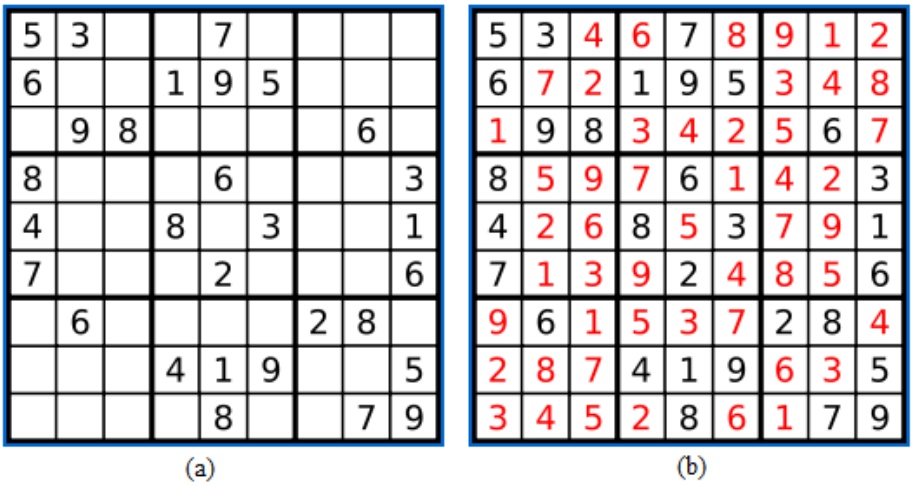
DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

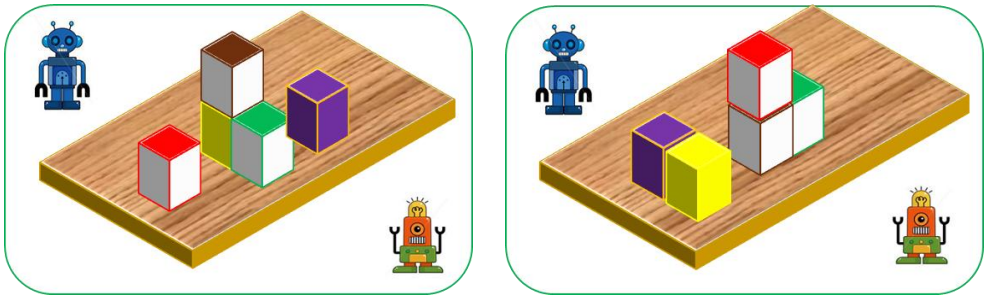
Sudoku adalah sejenis teka-teki logika yang tujuan akhirnya adalah mengisikan angka-angka 1 sampai dengan 9 ke dalam suatu kotak berukuran 9x9. Kotak ini memiliki 9 sub-kotak berukuran 3x3. Syarat teka-teki ini adalah tidak ada angka yang berulang pada setiap baris, kolom, atau sub-kotak. Pada saat awal, kotak sudoku belum terisi penuh dengan angka seperti terlihat pada gambar 1.1 (a). Solusi untuk teka-teki Sudoku pada gambar 1.1 (a) dapat dilihat pada gambar 1.1 (b), ditandai dengan angka berwarna merah. Teka-teki Sudoku pertama kali dipopulerkan sebuah perusahaan teka-teki Jepang bernama Nikoli pada tahun 1986, yang kemudian mendunia pada tahun 2005.



Gambar 1.1. Contoh teka-teki Sudoku (a) dan solusinya (b)

Teka-teki Sudoku termasuk ke dalam permasalahan kombinatorial (*NP complete*). Solusi untuk teka-teki ini dapat dicari dengan bermacam-macam cara seperti algoritma genetik [6], Particle Swarm Optimization [9, 10], dan sebagainya. Pada penelitian ini akan dilakukan suatu pendekatan yang berbeda untuk pencarian solusi dari permainan Sudoku, yaitu dengan memodelkannya sebagai *Block-World Problem.*

Pada *Block-World Problem*, terdapat sejumlah balok pada meja dengan susunan tertentu. Balok-balok tersebut kemudian diubah susunannya menjadi susunan balok akhir dengan bantuan dua jenis robot. Robot pertama bertugas memindahkan balok dari tumpukan balok ke meja, sementara robot yang lain bertugas memindahkan balok dari meja ke atas tumpukan balok yang lain. Kedua robot saling bekerja sama dengan berkomunikasi untuk menghasilkan susunan balok akhir. Pada setiap saat, hanya terdapat satu balok yang dapat dipindahkan ke meja atau ke atas tumpukan balok yang lain. Balok yang berada di bawah balok yang lain tidak dapat langsung dipindahkan sebelum balok-balok yang terdapat di atasnya dipindahkan. Gambar 1.2 memperlihatkan contoh tumpukan balok awal (a) dan tumpukan balok akhir (b). Fokus dari permasalahan *Block-world Problem* adalah pencari solusi yang berupa urutan langkah mulai dari susunan awal sampai dengan susunan akhir.



(a) (b)

Gambar 1.2 *Block-World Problem,* (a) tumpukan balok awal, (b) tumpukan balok akhir

Pada penelitian sebelumnya permainan Sudoku telah berhasil dimodelkan sebagai *Block-World Problem* [11]. Model yang dihasilkan sebanyak tiga buah yang ditulis secara formal dalam notasi Temporal Logic of Actions+ (TLA+). Model yang dikembangkan mengikuti kerangka spesifikasi umum untuk *parameterized systems* yaitu salah satu kelas sistem reaktif yang terdiri atas beberapa komponen sejenis dengan banyaknya komponen dinyatakan sebagai parameter. Lebih lanjut, model dikembangkan dengan konsep *multi-agent systems*.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat pada penelitian ini adalah:

1. Bagaimana mengimplementasikan model formal tersebut menjadi sudoku Solver dengan bahasa pemrograman Java?

Terdapat perbedaan sudut pandang antara spesifikasi formal dengan TLA dengan bahasa pemrograman Java. Salah satu hal penting yang harus diperhatikan pada saat implementasi adalah bagaimana mengimplementasikan pendekatan *interleaving* dan konsep *multi-agent systems* dengan Java. Selain itu masalah yang harus diperhatikan adalah bagaimana masalah penjadwalan robot dan bagaimana komunikasi antar robot.

2. Bagaimana performansi dari ketiga model yang telah dikembangkan?
   Untuk menjawab pertanyaan ini perlu dilakukan eksperimen terhadap prototipe dengan mencoba setiap model pada sekumpulan soal Sudoku.

1.3 Tujuan

Tujuan dari penelitian ini adalah

1. Mengembangkan *framework* Sudoku Solver.
2. Mengembangkan *prototype* Sudoku *solver* berdasarkan model yang telah dihasilkan sebelumnya, dimana Sudoku dimodelkan sebagai permasalahan *Block-World Problem*.
3. Melakukan eksperimen terhadap masing-masing *solver* untuk mengetahui kinerjanya.

1.4 Kontribusi Penelitian

Selain berkontribusi terhadap pengembangan ilmu pengetahuan, yaitu memperkaya alternatif pencarian solusi dari permasalahan Sudoku, penelitian ini juga akan menghasilkan sebuah *framework Sudoku Solver* yaitu sebuah program yang dapat digunakan untuk mengembangkan *Sudoku Solver*. *Framework* ini bisa digunakan untuk mempermudah pengembangan *Sudoku solver* yang lain.

1.5 Keluaran

Keluaran yang ditargetkan dari penelitian ini adalah publikasi dalam konferensi yang relevan atau dalam jurnal ilmiah nasional/internasional.

BAB II
TINJAUAN PUSTAKA

2.1 Permasalahan Sudoku

Studi pustaka yang telah dilakukan sejauh ini adalah dengan mempelajari beberapa hasil penelitian yang dapat ditemukan di internet. Khususnya yang terkait dengan pendekatan/teknik/ algoritma yang digunakan untuk pencarian solusi. Beberapa di antaranya adalah:

1. Pada referensi [7] dibahas pendekatan pencarian solusi permainan Sudoku dengan menggunakan pendekatan pencarian berbasis stokastik, yaitu *simulated annealing*. Selain itu, pada penelitian ini juga diperkenalkan metode baru untuk membangkitkan papan permainan Sudoku yang dapat diselesaikan dengan *simulated annealing*.
2. Pada referensi [13] dibahas tentang penyelesaian permainan Sudoku dengan memandang Sudoku sebagai *Constraint Satisfaction Problem*. Pendekatan yang digunakan adalah *constraint programming*. Selain masalah pencarian solusi permainan Sudoku, pada makalah ini juga dibahas tentang teknik untuk membangkitkan papan Sudoku dan menentukan tingkat kesulitan dari suatu permainan Sudoku.
3. Pada referensi [5] permainan Sudoku dimodelkan sebagai suatu permasalahan SAT problem. Permainan Sudoku disini direpresentasikan sebagai sekumpulan proposisi dalam bentuk Conjunctive Normal Form (CNF) dari logika proposisi. Pencarian solusinya dilakukan dengan prinsip inferensi dengan menggunakan teknik propagasi unit dan teknik lain yang lebih kompleks.
4. Pada referensi [4, 6, 8, 14] diusulkan pendekatan untuk penyelesaian dan pembangkitan permainan Sudoku dengan *evolutionary algorithms*, khususnya algoritma genetik. Tujuan dari penelitian yang dilakukan adalah menguji apakah algoritma genetic cukup efisien untuk memecahkan permainan Sudoku dan untuk membangkitkan papan permainan Sudoku.
5. Sedangkan untuk *Block-World problem*, digunakan referensi [11]. Pada makalah ini dikaji bagaimana *Block-world Problem* dapat dipandang sebagai suatu *parameterized multi agent system* dan bagaimana permasalahan tersebut dapat dinyatakan secara formal dengan menggunakan notasi Temporal Logic of Actions+ (TLA+).

2.2 Pemodelan Sudoku sebagai *Block-World Problem*

Pada bagian akan diberikan dengan singkat pemodelan Sudoku sebagai *Block-World Problem* yang dilakukan pada penelitian sebelumnya.

2.2.1 Pemetaan Komponen Permainan Sudoku kedalam *Block-World Problem*

Definisi Sudoku yang digunakan pada penelitian ini mengikuti definisi dari [4], yaitu sbb.:

Definisi 1..Diberikan sebuah n^2xn^2 sel yang terbagi atas n x n subblok berbeda, tujuan dari permainan Sudoku adalah mengisi setiap sel sedemikian sehingga kondisi berikut ini terpenuhi:

1) Pada setiap baris sel terdapat satu sel yang bernomor 1 sampai n^2
2) Pada setiap kolom sel terdapat satu sel yang bernomor 1 sampai n^2
3) Pada setiap subblok terdapat satu sel yang bernomor 1 sampai n^2

Pada penelitian ini, diambil nilai $n = 3$.

Pada bagian awal laporan ini, telah diperkenalkan komponen dari *Block-World Problem* dan tujuan dari permasalahan tersebut. Untuk memodelkan Sudoku kedalam permasalahan *Block- World Problem*, maka dilakukan modifikasi terhadap komponen dari *Block-World Problem* sbb.:

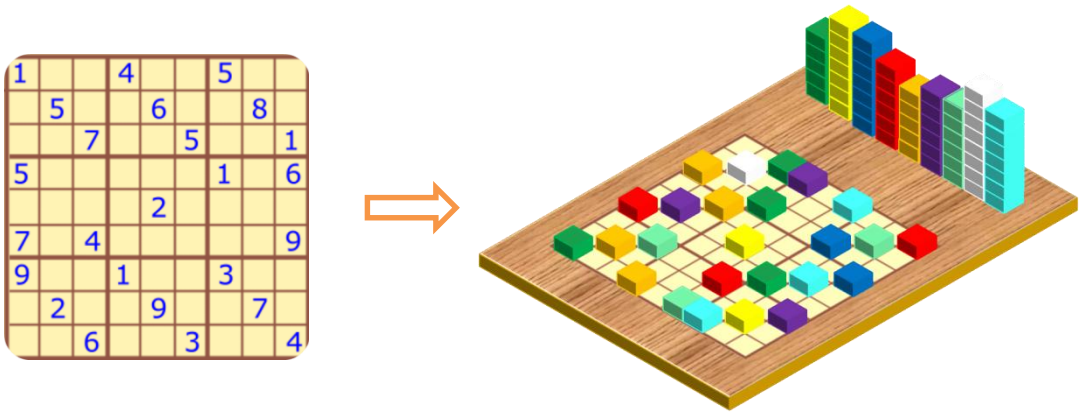
- Jumlah kotak di atas meja sebanyak 9 x 9.
- Setiap kotak diberi nomor antara 1 s/d 9.
- Setiap nomor $b = \{1, .., 9\}$ terdapat 9 kotak dengan nomor *b*.
- Terdapat bagian khusus pada meja yang disebut dengan papan yang terdiri atas 3 x 3 *grid*. Setiap *grid* disebut subblok dan setiap subblok terdiri atas 3 x 3 *grid* yang lebih kecil yang disebut sel.

Pada saat awal sebagian kotak sudah berada pada sel papan dan sebagian di luar papan. Kotak yang berada di luar papan disusun dalam sembilan tumpukan sesuai dengan nomor kotaknya. Hasil modifikasi tersebut dapat digambarkan seperti tampak pada Gambar 2.1.

Selanjutnya dilakukan juga modifikasi terhadap perilaku dari robot sbb. :

- Robot pertama berfungsi untuk mengambil sebuah kotak dari luar papan dan meletakkannya di salah satu sel di papan sedemikian sehingga syarat-syarat yang dinyatakan pada Definisi 1 dapat terpenuhi.
- Robot kedua berfungsi untuk mengambil sebuah kotak dari sel pada papan dan mengembalikannya ke tumpukan luar papan sesuai dengan nomor kotak yang diambil.

Dengan modifikasi tersebut maka *Block-World Problem* menjadi teka-teki Sudoku.



Gambar 2.1. Modifikasi *Block-World Problem*

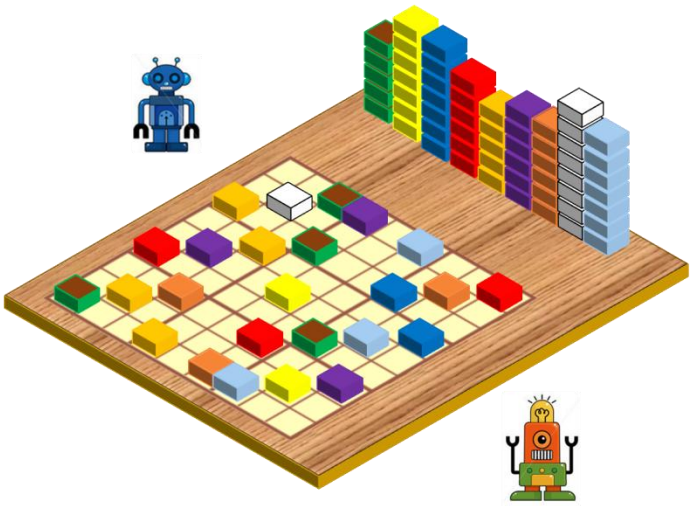
Selanjutnya akan dijelaskan tiga model *solver* yang diusulkan pada … untuk menyelesaikan permasalahan Sudoku. Ketiga model ini berbeda dari jumlah robot dan tugas dari robotnya.

2.2.2 Model 1

Mirip dengan *Block-world problem* yang asli pada model ini digunakan dua buah robot yang berbeda jenis. Robot ini akan secara bergiliran menjalankan tugasnya dengan aturan sbb.:

- Robot pertama mendapat giliran lebih dulu. Tugas robot ini adalah mengisi sel yang kosong dengan sebuah kotak dari tumpukan di luar papan tanpa melanggar syarat yang ditetapkan. Robot pertama ini akan bekerja terus-menerus sampai salah satu kondisi berikut:
  1. Semua sel papan telah terisi dengan kotak, yang berarti teka-teki berhasil dipecahkan. Jika kondisi ini tercapai, maka robot satu akan melaporkan bahwa solusi ditemukan.
  2. Tidak ada lagi sel yang dapat diisi dengan kotak di luar papan. Jika kondisi ini tercapai, maka robot satu akan melaporkan telah terjadi kegagalan dalam pencarian solusi.

Dalam proses pencarian sel yang kosong, robot pertama ini akan memilih sel dengan posisi terkecil. Didefinisikan posisi terkecil adalah (1,1) dan terbesar adalah (9,9). Robot ini akan mencatat sel yang kosong dan yang sudah terisi selama proses pencarian solusi berlangsung.



Gambar 2.2. Modifikasi *Block-World Problem* untuk Model 1.

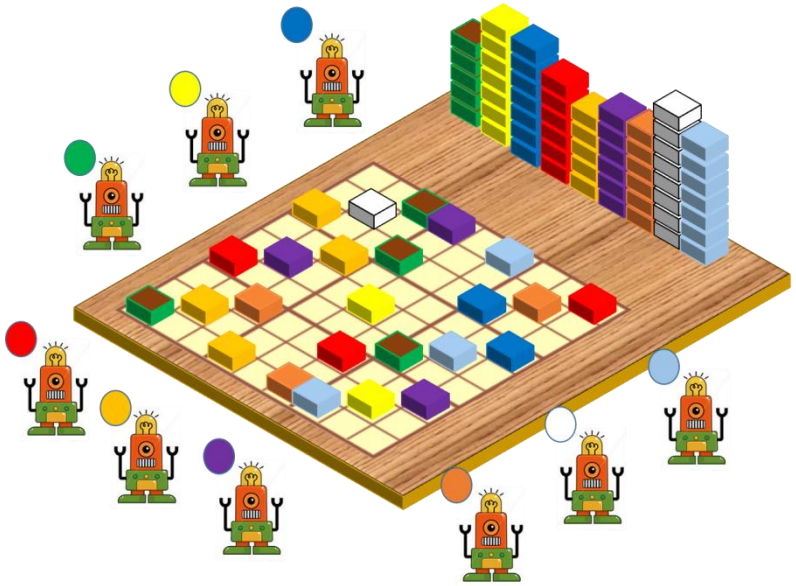
Robot kedua bertugas bilamana robot pertama gagal menemukan sel kosong yang dapat diisi dengan kotak. Robot ini bertugas untuk mengambil sebuah kotak di sel dan mengembalikan ke tumpukan yang sesuai untuk kotak tersebut. Pada model ini digunakan aturan kotak yang diambil adalah kotak yang paling akhir diletakkan oleh robot pertama. Untuk itu robot kedua memerlukan informasi tentang urutan penempatan kotak oleh robot pertama.

2.2.3 Model 2

Berbeda dengan model pertama, model 2 dan model3 menggunakan sembilan robot dengan tipe yang sama. Setiap robot bertanggung jawab terhadap kotak-kotak yang berada pada sebuah tumpukan kotak yang sesuai dengan nomernya. Semua robot mempunyai tugas yang sama yaitu berusaha meletakkan sebuah kotak dari tumpukannya ke sel di papan tanpa melanggar aturan yang berlaku.

Pada model 2 dan 3 digunakan prinsip *fixed-point*. Pencarian solusi dilakukan dengan melakukan iterasi sampai kondisi berhenti tercapai. Dalam setiap iterasi, setiap robot *i* mencari sel atau sebuah posisi valid yang dapat ditempati oleh kotak bernomor *i*. Jika pencarian sel berhasil, maka robot *i*

akan meletakkan kotak *i* pada sel tersebut. Setelah melakukan tugasnya, robot *i* memberikan giliran ke robot berikutnya. Setelah robot 9 melakukan tugasnya, akan ditentukan apakah iterasi dilanjutkan atau dihentikan. Jika dalam suatu iterasi tidak ada satu robot pun yang berhasil meletakkan sebuah kotak, maka iterasi dihentikan, iterasi dilanjutkan untuk kondisi sebaliknya.



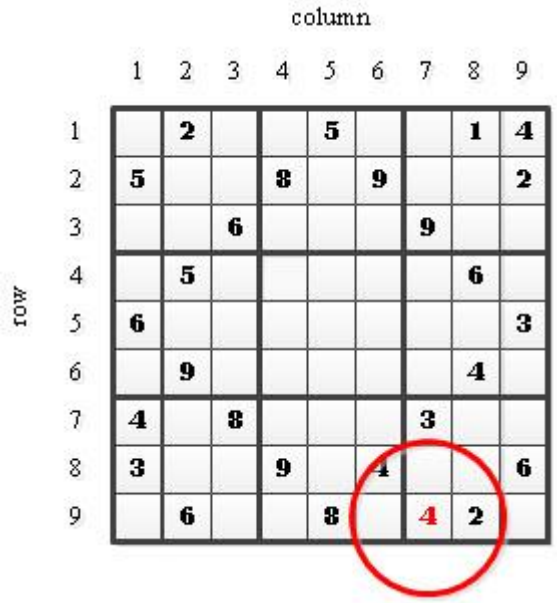
Gambar 2.3. Modifikasi *Block-World Problem* untuk Model 2 dan 3.

Perbedaan antara model 2 dan model 3 terletak pada definisi posisi valid. Untuk model 2 posisi valid didefinisikan sbb.:

Definisi 2. Sebuah robot *i* menyebut sebuah posisi (*x*,*y*) valid untuk kotak bernomor *i* jika kondisi-kondisi berikut ini terpenuhi:

1. Sel pada posisi tersebut kosong.
2. Baris *x* tidak mengandung kotak bernomor *i*.
3. Kolom *y* tidak mengandung kotak bernomor *i*.
4. Subblok dimana posisi (*x*,*y*) berada tidak mengandung kotak bernomor *i*
5. Untuk setiap baris *r* yang berada pada *subblok* dimana (*x*,*y*) berada terdapat satu sel yang mengandung kotak bernomor *i*.
6. Untuk setiap kolom *c* yang berada pada subblok dimana (*x*,*y*) berada terdapat satu sel yang mengandung kotak bernomor *i*.

Gambar 2.4 menampilkan contoh posisi valid untuk model 2. Posisi (9,7) adalah posisi valid bagi robot 4. Tampak pada gambar tersebut setiap baris pada subblok dimana (9,7) berada, yaitu baris 7 dan 8, masing-masing mengandung kotak bernomor 4. Demikian juga pada setiap kolom pada subblok dimana (9,7) berada, yaitu kolom 8 dan kolom 9 masing-masing mengandung kotak bernomor 4.



Gambar 2.4. Contoh posisi valid untuk model 2.

2.2.4 Model 3

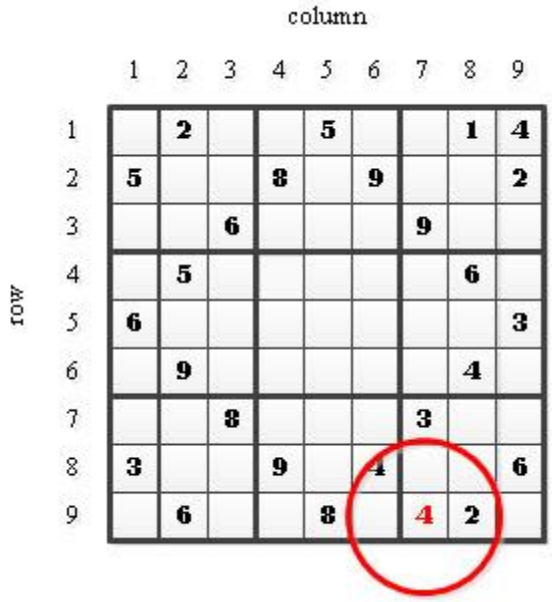
Model 3 ini sangat mirip dengan model 2. Perbedaannya hanya terletak pada definisi posisi valid. Untuk model 3 digunakan definisi berikut untuk menyatakan sebuah posisi valid atau tidak.

Definisi 3. Sebuah robot *i* menyebut sebuah posisi (*x*,*y*) valid jika kondisi-kondisi berikut ini terpenuhi:

1. Sel pada posisi tersebut kosong.
2. Baris *x* tidak mengandung kotak bernomor *i*.
3. Kolom *y* tidak mengandung kotak bernomor *i*.
4. Subblok dimana posisi (*x*,*y*) berada tidak mengandung kotak bernomor *i*
5. Untuk setiap baris *r* yang berada pada *subblok* dimana (*x*,*y*) berada terdapat satu sel yang mengandung kotak bernomor *i* atau sel pada posisi (*r*,*y*) tidak kosong.

6. Untuk setiap kolom *c* yang berada pada subblok dimana (*x*,*y*) berada terdapat satu sel yang mengandung kotak bernomor *i* atau sel pada posisi (*x*,*c*) tidak kosong.

Contoh posisi valid untuk model 3 diberikan pada Gambar 2.5. Mirip dengan contoh posisi valid untuk model 2 (gambar 2.4), robot 4 dapat menyatakan posisi (9,7) adalah posisi yang valid. Meskipun tidak terdapat sel bernomor 4 pada baris 7, tetapi posisi (7,7) tidak kosong sehingga dapat dipastikan kotak bernomor 4 tidak akan dapat diletakkan pada posisi (7,7) tersebut.



Gambar 2.5. Posisi valid untuk model 3.

BAB III
METODE PENELITIAN

Metode penelitian pada penelitian ini adalah:

- Studi literatur

  Studi literatur dilakukan dengan mempelajari makalah-makalah yang terkait dengan topik penelitian.

- Pengembangan *framework* dan *prototype solver*

  Untuk pengembangan *framework* akan digunakan tahapan yang umum dilaksanakan dalam pengembangan perangkat lunak, khususnya tahapan analisis, perancangan, implementasi, dan pengujian.

- Eksperimen

  Eksperimen dilakukan dengan mencoba sekumpulan soal Sudoku yang dapat ditemukan di internet atau dari buku-buku kumpulan soal Sudoku. Eksperimen ini bertujuan untuk menentukan sejauh mana model yang dibuat dapat menyelesaikan soal-soal Sudoku tersebut. Dari hasil eksperimen ini akan disimpulkan kinerja dari setiap model, dengan melihat prosentase soal yang berhasil diselesaikan, banyaknya sel kosong papan Sudoku yang berhasil diisi, dan dari sisi waktu yang diperlukan untuk menyelesaikan soal Sudoku.

BAB IV
HASIL YANG DICAPAI

Pada bagian ini dijelaskan hasil yang dicapai pada penelitian yang telah dilakukan. Terdapat empat hasil yang dapat dilaporkan, yaitu:

1. Usulan model *solver* yang baru.
2. Pengembangan framework
3. Pengembangan *Sudoku solver*.
4. Hasil eksperimen terhadap performansi masing-masing *solver*.

4.1 Usulan Model Baru

Pada penelitian ini diusulkan sebuah model baru yang merupakan pengembangan dari model 3. Seperti model 3, pada model 4 ini digunakan sembilan robot dengan tugas yang sama yaitu mengambil sebuah kotak dari tumpukan di luar papan, dan meletakkannya pada salah satu sel di papan sedemikian sehingga tidak melanggar aturan Sudoku. Perbedaan model 4 ini dengan model 2 adalah pada definisi posisi valid.

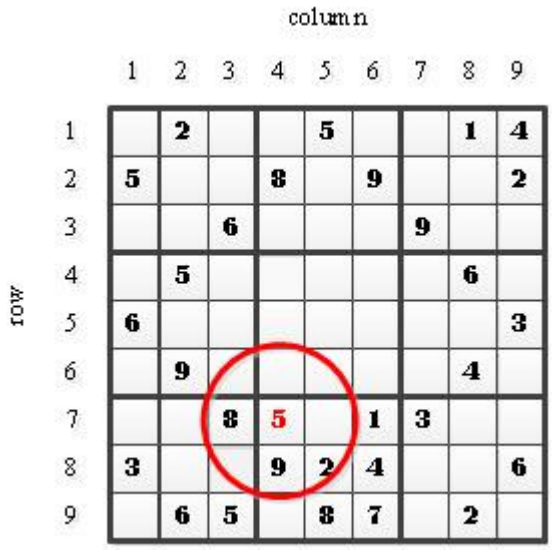
Definisi posisi valid untuk model 4 adalah sbb.:

Definisi 4. Sebuah robot *i* menyebut sebuah posisi (*x*,*y*) valid jika kondisi-kondisi berikut ini terpenuhi:

1. Sel pada posisi tersebut kosong.
2. Baris *x* tidak mengandung kotak bernomor *i*.
3. Kolom *y* tidak mengandung kotak bernomor *i*.
4. Subblok dimana posisi (*x*,*y*) berada tidak mengandung kotak bernomor *i*
5. Untuk setiap baris *r* yang berada pada *subblok* dimana (*x*,*y*) berada terdapat satu sel yang mengandung kotak bernomor *i* atau sel pada posisi (*r*,*y*) tidak kosong atau semua sel pada pada baris *r* tidak kosong .

6. Untuk setiap kolom *c* yang berada pada subblok dimana (*x*,*y*) berada terdapat satu sel yang mengandung kotak bernomor *i* atau sel pada posisi (*x*,*c*) tidak kosong atau semua sel pada kolom *c* tidak kosong.

Gambar 4.1 memperlihatkan contoh posisi valid bagi robot 5 untuk model 4. Tampak pada gambar, posisi (7,4) termasuk posisi valid. Meskipun tidak ada sel yang mengandung kotak bernomor 5 pada baris 8 dan kolom 6, robot 5 masih dapat menemukan posisi untuk meletakkan kotak bernomor 4 pada posisi ini karena semua sel pada baris 8 dan kolom 6 tidak kosong.



Gambar 4.1. Contoh posisi valid untuk model 4.

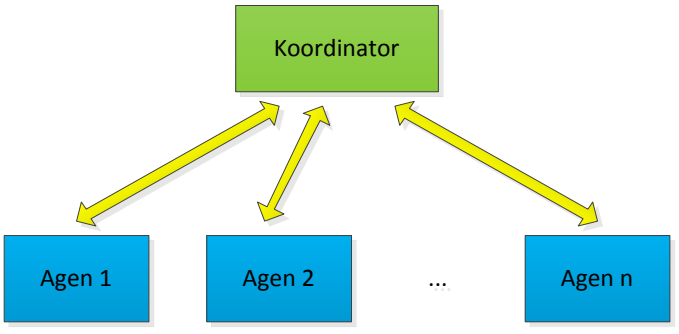
4.2 Implementasi *Framework*

Pada [11] Sudoku *solver* dimodelkan sebagai sebuah sistem multi agen berparameter. Sebuah sistem multi agen berparameter adalah sebuah sistem yang terdiri atas sejumlah komponen (subsistem) yang mirip yang bekerja bersama-sama, jumlah komponen dinyatakan sebagai parameter masukan dari sistem. Agennya adalah robot, lingkungannya adalah papan dan kotak. Model ditulis dengan menggunakan TLA+ sebagai bahasa spesifikasi.

Terdapat beberapa aspek penting yang harus dipertimbangkan dalam mengimplementasikan sistem multi agen berparameter tersebut, seperti arsitektur, komunikasi antar agen, dan penjadwalan agen. Pada penelitian ini karena penekanannya lebih kepada pengukuran performansi dari masing-masing model, maka diambil pendekatan yang sederhana dalam implementasi. Adapun pendekatan yang diambil adalah sbb.:

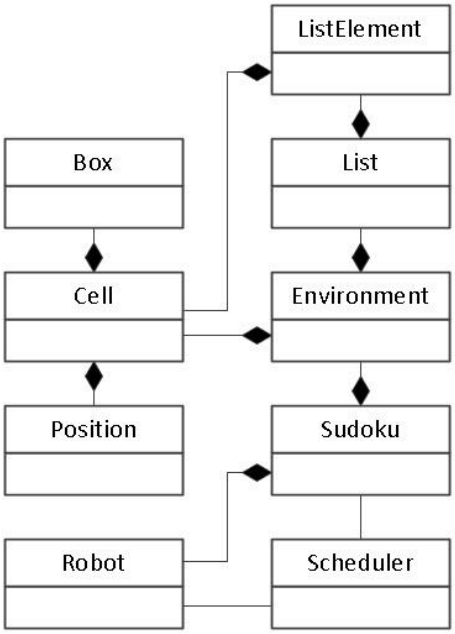
1. Berdasarkan prinsip kerja dari sistem multi agen berparameter yang diacu, digunakan prinsip *interleaving*, yang berarti agen akan bekerja secara bergiliran, setiap saat hanya ada maksimal satu agen yang aktif.
2. Penjadwalan menggunakan prinsip *ring counter* dan *token based* yaitu setiap agen akan diberi nomor identifikasi dan pengaktifan agen dilakukan secara berurutan mulai dari nomor terkecil sampai terbesar, setelah itu kembali ke nomor terkecil dst. Untuk memberitahu kapan sebuah agen aktif, digunakan token. Agen yang sedang memegang token berhak untuk melakukan aktivitas. Jika aktivitasnya telah selesai, agen akan memberikan token tersebut ke agen berikutnya.
3. Komunikasi antar agen, dalam hal ini adalah pemberian token kepada agen berikutnya, dilakukan secara tidak langsung, yaitu melalui koordinator (mediator). Dapat dikatakan bahwa koordinator inilah yang berfungsi untuk mengatur giliran agen.

Dengan pendekatan tersebut, maka arsitektur sistem multi agen yang digunakan pada *framework* yang dibangun dapat dijelaskan secara sederhana seperti pada gambar 4.2.



Gambar 4.2 Arsitektur Sistem Multi Agen.

Gambar 4.3 adalah diagram kelas dari sistem yang dikembangkan. Terdapat sembilan kelas yaitu Box, Cell, Environment, List, ListElement, Position, Robot, Scheduler, dan Sudoku. Penjelasan dari masing-masing kelas adalah sebagai berikut diberikan pada ttersebut adalah sbb.:



Gambar 4.3. Diagram Kelas.

Daftar atribut dan metode dari masing-masing kelas

Kelas Box
Box adalah kelas yang merepresentasikan kotak pada permasalahan *block-world problem*
Atribut • int Value menyimpan nilai kotak • int agentType menyimpan tipe agen
Metode • Box (int v) Mendefinisikan sebuah Box dengan Value = v. • Box (int v, int a) Mendefinisi sebuah Box dengan Value = v dan agentType = a. • int getValue() Mengembalikan Value dari sebuah Box. • void setValue(int v) Mengisi Value dengan v.

- int getAgentType()
  Mengembalikan agentType dari sebuah Box.
- void setAgentType(int a)
  Mengisi agentType dengan a.
- void printBox()
  Mencetak informasi dari sebuah Box.
- void printValue()
  Mencetak Value dari sebuah Box.

Kelas Cell

Cell adalah kelas yang merepresentasikan kotak pada papan permainan Sudoku.

Atribut

- Box box
  Menyimpan box yang diletakkan pada sebuah sel.
- Boolean fixed
  Apakah isi sel tetap atau bisa berubah.
- Position pos
  Menyimpan posisi sel pada papan Sudoku.

Metode

- Cell()
  Mendefinisikan sebuah sel kosong.
- Cell (int x, int y)
  Mendefinisikan sebuah sel pada posisi (x,y).
- Cell (int x, int y, int v)
  Mendefinisikan sebuah sel pada posisi (x,y) dengan nilai box v.
- Cell (Position p)
  Mendefinisikan sel kosong pada posisi p.
- Cell (Position p, Box b)
  Mendefinisikan sebuah sel pada posisi p dengan box b.
- Box getBox()
  Mengembalikan box dari sebuah sel.
- Position getPosition()
  Mengembalikan posisi sebuah sel.
- boolean isFixed()
  Mengembalikan nilai fixed.
- boolean isEmpty()
  Menguji apakah sel sudah terisi box atau belum.
- void setBox(Box b)
  Mengisi sel dengan box b.
- void setEmpty()
  Mengosongkan sel.
- void setFixed()
  Sel tidak boleh berubah.
- void setPosition(int x, int y)
  Menetapkan posisi sebuah sel.
- void printCell()
  Mencetak informasi sebuah sel.

Kelas Environment

Environment adalah kelas yang menyimpan informasi tentang lingkungan permainan Sudoku.

Atribut

- int SIZE
  Ukuran papan Sudoku.
- int numbers []
  Banyaknya kota pada setiap tumpukan.
- List FreeCellList
  Daftar sel yang masih kosong.
- Cell [][] Board
  Papan Sudoku.
- Box [][] bStack
  Tumpukan kotak di luar papan Sudoku.
- boolean backtrack
  Indikator terjadi backtrack.

Metode

- Environment()
  Mendefinisikan sebuah lingkungan permainan Sudoku.
- boolean isValidCell(int i, int j)
  Menguji apakah sel pada posisi (i,j) adalah sebuah sel yang valid.
- boolean isValidPuzzle()
  Menguji apakah puzzle valid atau tidak.
- void SetupBoxStacks()
  Mendefinisikan tumpukan kotak di luar papan Sudoku.
- boolean isValidNumber(int x)
  Menguji apakah x adalah nilai kotak yang valid.
- boolean emptyStacks()
  Menguji apakah seluruh tumpukan kotak kosong.
- boolean isInRow(int x, int id)
  Menguji apakah sebuah row x mengandung angka id.
- boolean isRowFull(int x, int y)
  Menguji apakah seluruh baris pada subblok (x,y) sudah terisi kotak.
- boolean isColumnFull(int x, int y)
  Menguji apakah seluruh kolom pada subblok yang mengandang posisi (x,y) sudah terisi kotak.
- boolean isInColumn(int x, int id)
  Menguji apakah sebuah kolom mengandung angka id.
- boolean isInSubblock(Position p, int id)
  Menguji apakah sebuah subblok yang mengandung posisi p terdapat kotak dengan angka id.
- boolean isSolved()
  Menguji apakah permainan Sudoku sudah terpecahkan.
- int getSize()
  Mengembalikan ukuran papan Sudoku.
- boolean readAPuzzle(String fileName)
  Membaca permainan Sudoku dari file.
- void setBacktrack (boolean b)
  Memberi nilai backtrack dengan nilai b.
- boolean isBacktrack()
  Menguji apakah backtrack bernilai true.
- void printBoard()
  Mencetak isi papan Sudoku.

Kelas List

Kelas list digunakan untuk menyimpan daftar sel.

Atribut

- int first
  Indeks elemen pertama list.
- int last
  Indeks elemen terakhir list.
- int length
  Panjang list.
- ListElement[] Content
  Isi list.

Metode

- List (int size)
  Mendefinisikan list dengan ukuran size.
- int getFirst()
  Mengembalikan indeks elemen pertama.
- void setFirst (int i)
  Menetapkan indeks elemen pertama.
- int getLast()
  Mengembalikan indeks elemen terakhir.
- void setLast(int i)
  Menetapkan indeks elemen terakhir.
- int getLength()
  Mengembalikan panjang list.
- void decLength()
  Mengurangi panjang list dengan satu.
- ListElement [] getContent()
  Mengembalikan isi list.
- ListElement getFirstElement()
  Mengembalikan elemen pertama list.
- ListElement getElement (int i)
  Mengembalikan elemen list ke-i.
- void InsertLastElement (Cell c)
  Menyisipkan sel c ke akhir list.
- void InsertFirstElement (Cell c)
  Menyisipkan sel c di awal list.
- void InsertElement (Cell c)
  Menyisipkan sel c kedalam list.
- void InsertLast (Cell c)
  Menyisipkan sel c di akhir list dengan menguji kondisi list terlebih dulu.
- boolean InsertElement (ListElement le)
  Penyisipan elemen pada proses backtrack.
- int findPosition (ListElement le)
  Mencari indeks sebuah elemen list.
- void delFirst()
  Menghapus elemen pertama list.
- void pushElement (Box b)
  Menyisipkan elemen dengan isi box b ke akhir list.

Kelas ListElement
Kelas ListElement merepresentasikan isi list.
Atribut • Cell cell Sel yang disimpan oleh elemen list. • int next indeks dari elemen di belakangnya.
Metode • ListElement (Cell c, int n) Mendefinisikan sebuah elemen list berisi cell c dan next n. • Cell getCell() Mengembalikan cell dari elemen list. • Int getNext() Mengembalikan next dari elemen list. • void setNext(int i) Menetapkan nilai next dengan i.

Kelas Position
Kelas Position merepresentasikan sebuah posisi di papan Sudoku.
Atribut • int posX Baris pada papan Sudoku. • int posY Kolom pada papan Sudoku.
Metode • Position (int x, int y) Mendefinisikan sebuah posisi pada baris x dan kolom y. • int getPosX() Mengembalikan baris. • int getPosY() Mengembalikan kolom. • void setPosX (int x) Menetapkan baris. • void setPosY (int y) Menetapkan kolom.

Kelas Robot
Kelas Robot merepresentasikan agen/robot dari block-world problem.
Atribut • Thread t Thread dari robot. • String threadName Nama thread. • int type Tipe agent/robot. • Sudoku S Permainan Sudoku. • Cell resCell Sel hasil pencarian untuk ditempati sebuah kotak dengan nomor tertentu.

Metode

- Robot (Sudoku S, int i)
  Mendefinisikan sebuah robot dari permainan S dan bertipe i.
- void start()
- thread getThread()
  Mengembalikan thread.
- void run()
  Menjalankan/mengaktifkan robot.
- boolean putOn()
  Mencari sel yang dapat diisi dengan kotak dengan nomor tertentu.
- boolean takeBack()
  Mengambil sebuah kotak dari papan Sudoku.
- boolean Action()
  Aksi dari robot.
- int getValidNumber(ListElemen le)
- ListElement getValidFreeCell()
  Mengembalikan sebuah isi list yang kotaknya dapat diletakkan pada papan Sudoku.
- boolean isValidPosition (Position pos)
  Menguji apakah pos adalah sebuah posisi yang valid.

Kelas Scheduler

Kelas Scheduler merepresentasikan agen yang mengatur giliran kerja dari agen.

Atribut

- int turn
- boolean [] hasTurn
- boolean [] resAction
- int [] boxes
- Sudoku sudoku

Metode

- Scheduler (Sudoku S)
  Mendefinisikan sebuah Scheduler untuk permainan Sudoku S.
- int getTurn()
  Mengembalikan giliran saat itu.
- void setTurn()
  Menetapkan giliran.
- void nextTurn()
  Meneruskan giliran.
- void setHasTurn(int i)
  Menetapkan urutan berikutnya dari sebuah agen.
- void setResAction (int i, boolean res)
  Menetapkan aksi yang dilaksanakan oleh agen.
- boolean parAction ()
  Aksi-aksi yang dapat dilaksanakan oleh para agen.

Kelas Sudoku
Kelas Sudoku adalah kelas utama yang merepresentasikan permainan Sudoku.
Atribut • int model Model yang digunakan. • int agentNum Banyaknya agen yang digunakan. • String fileName File permainan Sudoku. • Environment Puzzle Environment. • Robot [] Agent Agen-agen yang digunakan. • Scheduler sch Scheduler dari permainan Sudoku.
Metode • Sudoku () Mendefinisikan Sudoku standar. • Sudoku (int model) Mendefinisikan Sudoku dengan model tertentu. • Sudoku (int model, String fN) Mendefinisikan Sudoku dengan model tertentu dan puzzle diambil dari file fN. • void setModel (int m) Menentukan model yang digunakan. • void defineAgents() Mendefinisikan agen-agen yang digunakan. • void killAgents() Menghapus agen-agen yang digunakan. • void loadPuzzle() Mengambil permainan Sudoku. • void main(String [] args) Memulai program. • Scheduler getScheduler() Mengembalikan Scheduler dari permainan Sudoku. • Environment getEnvironment() Mengembalikan Environment dari permainan Sudoku. • int getModel() Mengembalikan model yang digunakan. • int getAgentNum() Mengembalikan jumlah agen. • void run() Menjalan agen-agen.

4.3 Implementasi *Solver*

Dengan menggunakan *framework* yang telah berhasil dibangun, pengembangan sebuah *solver* dilakukan dengan melakukan penambahan metode pada beberapa kelas. Penambahan metode yang utama adalah pada kelas Robot, dengan menambahkan fungsi untuk menguji posisi valid dari setiap model *solver*:

- `boolean isValidPosition1 (Position p, int num)`
  `Menguji apakah p adalah posisi yang valid untuk kotak bernomor num, digunakan pada model 1.`
- `boolean isValidPosition2 (Position p)`
  `Menguji apakah p adalah posisi yang valid menurut model 2.`
- `boolean isValidPosition3 (Position p)`
  `Menguji apakah p adalah posisi yang valid menurut model 3.`
- `boolean isValidPosition4 (Position p)`
  `Menguji apakah p adalah posisi yang valid menurut model 4.`

4.4 Hasil Eksperimen

Tujuan dari eksperimen adalah untuk mengetahui performansi dari setiap *solver*. Eksperimen dilakukan dengan menjalankan setiap *solver* dan mencatat keberhasilan setiap *solver* dalam menyelesaikan seratus soal Sudoku dimana setiap soal Sudoku ini dijamin mempunyai paling sedikit satu solusi. Selain banyaknya soal yang berhasil dipecahkan, dicatat juga waktu yang diperlukan oleh masing-masing *solver* dalam memecahkan soal Sudoku. Hasil eksperimen ini ditunjukkan pada Tabel 4.1.

Tabel 4.1 Hasil Eksperimen 1

Model	Banyaknya soal sudoku yang berhasil dipecahkan (buah)
1	100
2	0
3	0
4	44

Seperti yang telah diperkirakan, *solver* model 1 berhasil menyelesaikan seluruh soal Sudoku yang diberikan. Sementara *solver* model 2 dan 3 tidak berhasil menyelesaikan satupun soal yang diberikan. Sedangkan *solver* model 4 berhasil menyelesaikan 44 soal dari seratus soal.

Performansi dari sisi waktu dilakukan dengan cara membandingkan rata-rata waktu yang diperlukan untuk menyelesaikan sebuah soal Sudoku. Karena *solver* model 2 dan 3 sama sekali tidak berhasil menyelesaikan satu soal pun, maka perbandingan hanya dilakukan terhadap *solver* model 1 dan 4. Tanpa memperhitungkan soal yang tidak berhasil dipecahkan, *solver* model 4 memerlukan waktu rata-rata 4.82 mili detik untuk menyelesaikan satu soal. Untuk 44 soal Sudoku yang berhasil dipecahkan oleh *solver* model 4, waktu yang diperlukan oleh *solver* model 1 untuk

menyelesaikan 44 soal yang sama adalah 231 mili detik atau rata-rata 4.84 mili detik. Dari hasil eksperimen ini dapat disimpulkan bahwa *solver* model 4 memerlukan waktu lebih sedikit dalam menyelesaikan sebuah soal Sudoku.

Pengukuran performansi selanjutnya adalah dengan menghitung banyaknya sel yang berhasil diisi oleh setiap *solver*. Hasil eksperimen ini ditunjukkan pada Tabel 2.

Tabel 2 Hasil Eksperimen 2

Model	Banyaknya sel yang berhasil diisi (%)
1	100
2	1.4
3	22
4	48

Tampak pada tabel 2, performansi terbaik masih dicapai oleh *solver* model 1. Karena semua soal berhasil dipecahkan maka seluruh sel berhasil diisi oleh *solver* tersebut. Urutan kedua dan ketiga dicapai oleh *solver* model 4 dan model 3 dengan 48 % dan 22%. Sementara *solver* model 2 mempunyai performansi terburuk dengan keberhasilan hanya 1.4 %.

Dari hasil-hasil eksperimen yang telah dilakukan dapat disimpulkan bahwa dari sisi keberhasilan memecahkan soal dan mengisi sel kosong, performansi terbaik dicapai oleh *solver* model 1 dan terbaik kedua dicapai oleh *solver* model 4. Namun dari sisi waktu yang diperlukan untuk menyelesaikan sebuah soal, *solver* model 4 mempunyai performansi yang lebih baik dibanding *solver* model 1.

Dari kesimpulan-kesimpulan tersebut, maka dilakukan eksperimen berikutnya yaitu mengkombinasikan *solver* model 4 dan 1. Tujuan dari eksperimen ini adalah untuk mengetahui apakah ada peningkatan performansi jika kedua *solver* ini digabungkan. Untuk eksperimen diambil sepuluh soal Sudoku tersulit dilihat dari lamanya waktu yang diperlukan oleh solver model 1 untuk menyelesaikan soal tersebut. Untuk setiap soal, *solver* model 4 dijalankan sampai tidak ada lagi sel kosong yang berhasil diisi, setelah itu dijalan solver model 1 untuk melanjutkan pencarian solusi sampai selesai. Waktu yang diperlukan oleh kedua *solver* tersebut dalam mencari solusi

dicatat dan dibandingkan dengan waktu yang diperlukan oleh *solver* model 1 dalam menyelesaikan soal tersebut. Hasilnya eksperimen ini ditampilkan pada Tabel 3.

Tabel 3. Hasil Eksperimen 3

Soal	Waktu penyelesaian oleh solver model 1 (ms)	Waktu penyelesaikan oleh solver model 4 dan model 1 (ms)
1	390	390
2	312	312
3	250	250
4	234	202
5	203	203
6	156	156
7	140	16
8	125	31
9	94	46
10	57	47
Total	**1951**	**1653**

Pada Tabel 3 tampak bahwa total waktu yang diperlukan oleh kombinasi *solver* model 4 dan 1 lebih kecil dibandingkan yang diperlukan oleh *solver* model 1 sendiri. Sehingga dapat disimpulkan bahwa kombinasi tersebut dapat meningkatkan performansi.

BAB V
SIMPULAN

5.1 Kesimpulan

Kesimpulan yang dapat diambil dari penelitian ini adalah:

1. Telah berhasil dikembangkan sebuah model baru sebagai variasi model-model yang telah diusulkan sebelumnya.
2. Telah berhasil dibuat sebuah *framework* pengembangan Sudoku *solver* dimana Sudoku *puzzle* dimodelkan sebagai *Block-World Problem*.
3. Dengan menggunakan *framework* tersebut, telah berhasil pula dikembangkan empat buah *solver* yang mengimplementasikan masing-masing model.
4. Telah dilakukan eksperimen untuk mengukur performansi dari keempat *solver*. Dari sisi keberhasilan menyelesaikan soal Sudoku dan mengisi sel kosong, *solver* model 1 mempunyai performansi terbaik disusul oleh *solver* model 4. Kombinasi dari *solver* model 4 dan 1 dapat meningkatkan performansi dari sisi waktu.

5.2 Rencana penelitian lebih lanjut

Model untuk Sudoku solver yang dikaji pada penelitian ini diambil dari [] dimana *solver* dimodelkan sebagai *block-world problem* yang dinyatakan sebagai sebuah sistem multi agen berparameter. Pada penelitian ini, aspek penting dari sistem multi agen, seperti komunikasi antar agen dan penjadwalan agen belum dikaji dengan cukup dalam. Pada saat implementasi diambil pendekatan yang paling sederhana untuk aspek-aspek tersebut. Ke depan, direncanakan untuk mengembangkan *solver* dengan kajian yang lebih mendalam terhadap aspek-aspek tersebut.

DAFTAR PUSTAKA

[1] A. Bartlett, et.al. *An Integer Programming Model for the Sudoku Problem*, The Journal of Online Mathematics and Its Applications: Volume 8, Issue May, 2008.

[2] X.Q. Deng & Y.D. Li. *A novel hybrid genetic algorithm for solving Sudoku puzzle*., Optimization Letters, February 2013, Volume 7, Issue 2, pp 241-257, Springer.

[3] Wayne Gould. *Su Doku Junior*. Penerbit Erlangga. 2005.

[4] R. Lewis, *Metaheuristics can solve Sudoku puzzles*, Journal of Heuristics, Vol. 13, 2007, pp. 387-401.

[5] I. Lynce & J. Ouaknine. *Sudoku as a SAT problem*, Proc. of the 9th Symposium on Artificial Intelligence and Mathematics, 2006.

[6] T. Mantere. *Solving, rating and generating Sudoku puzzles with GA*., Proc. of IEEE Congress on of Evolutionary Computation, 2007. CEC 2007, Singapore.

[7] P. Malakonakis, et.al. *An FPGA-based Sudoku Solver based on Simulated Annealing methods*., Proc. of International Conference on Field-Programmable Technology, 2009. FPT 2009.

[8] V. Mladenov, et.al., *Solving Sudoku Puzzles by using Hopfield Neural Networks*, Proc. of ICACM'11 Proceedings of the 2011 international conference on Applied & computational mathematics, pp.174-179 World Scientific and Engineering Academy and Society (WSEAS) Stevens Point, Wisconsin, USA, 2011.

[9] J. Monk, et.al. *Solving Sudoku using Particle Swarm Optimization on CUDA*, Proc. of The 2012 International Conference on Parallel and Distributed Processing Techniques and Applications, 2012,.

[10] A. Moraglio & J. Togelius, *Geometric Particle Swarm Optimization for the Sudoku Puzzle*, Proc. of Conference in Genetic and Evolutionary Computation, GECCO 2007, Proceedings, London, England, UK, July 7-11, 2007. ACM 2007.

[11] C.E. Nugraheni & L. Abednego. *Modelling Sudokus as Block World Problems*. International Journal of Computer, Information, Systems and Control Engineering, volume 7, no. 8, pp. 48-54, World Academy of Science, Engineering and Technology, 2013.

[12] Thomas Ag. S. *Sudoku - Edisi Lengkap dengan 270 Aneka Teka-teki Sudoku*. Penerbit Andi. 2012.

[13]] Helmut Simonis. *Sudoku as a constraints problem*. Proc. 4th Int. Works. Modelling and Reformulating Constraint Satisfaction Problems, Brahim Hnich, Patrick Prosser, and Barbara Smith, ed., 2005, pp. 13-27. http://4c.ucc.ie/ brahim/mod-proc.pdf.

[14] T.-W. Yue & Z.-C. Lee, *Sudoku Solver by Q´tron Neural Networks*, Proc. of International Conference in Intelligent Computing 2006, ICIC2006, LNCS 4113, pp.943-952, 2006, Springer-Verlag, Berlin Heidelberg 2006.