Laporan Tugas Kecil 1

IF3170 Inteligensi Buatan

Semester I Tahun Ajaran 2016/2017

**Sudoku sebagai *Constraint Satisfaction Problem* (CSP)**



Ade Surya Ramadhani 13514049

Geraldi Dzakwan 13514065

Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

**DAFTAR ISI**

[Deskripsi Persoalan dan *Tools* 3](#_Toc461467972)

[Sudoku sebagai CSP : Variabel, Domain, dan *Constraint* 5](#_Toc461467973)

[Parameter dan Pilihan Algoritma 8](#_Toc461467974)

[Hasil Solusi, Langkah Pencarian, dan Perbandingan 8](#_Toc461467975)

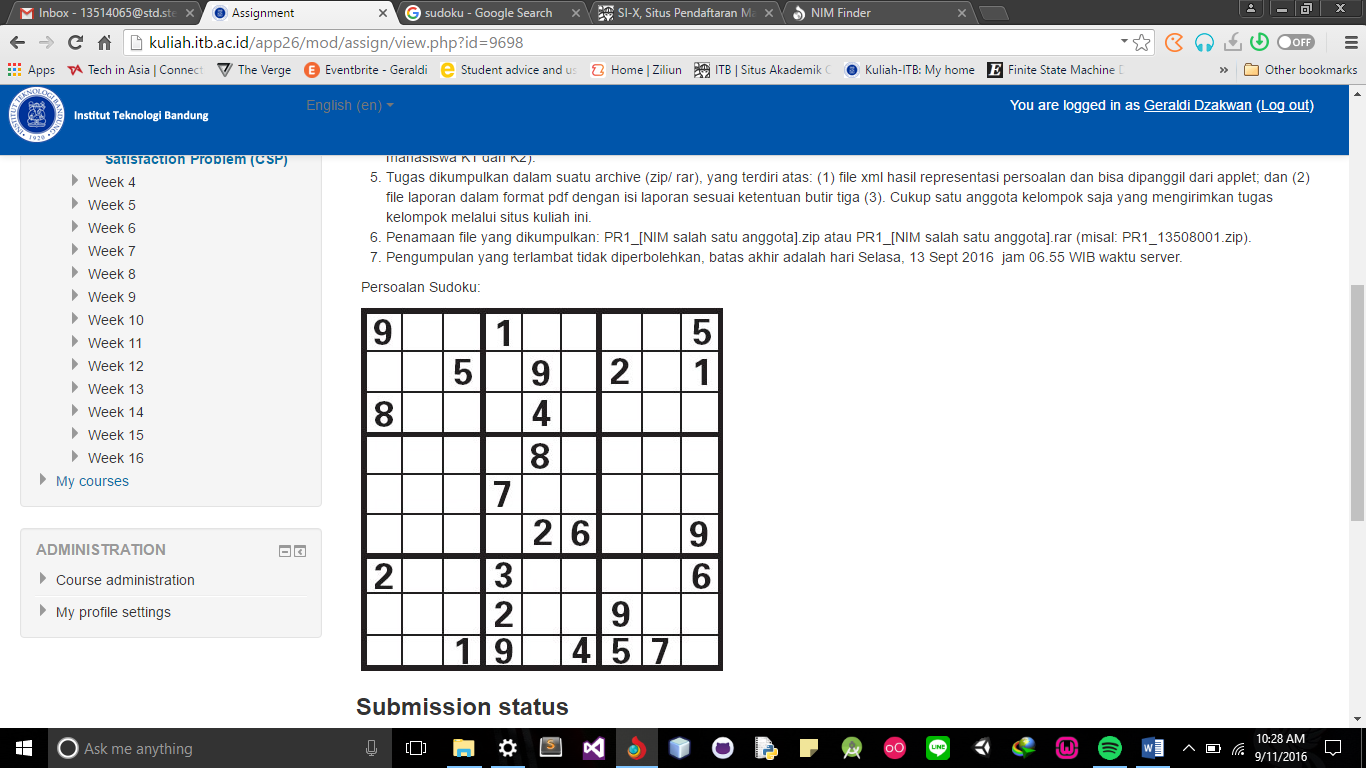
BAB I

# Deskripsi Persoalan dan *Tools*

Sudoku adalah sejenis [teka-teki](https://id.wikipedia.org/wiki/Teka-teki) logika. Nama "Sudoku" adalah singkatan [bahasa Jepang](https://id.wikipedia.org/wiki/Bahasa_Jepang) dari "Suuji wa dokushin ni kagiru" yang artinya "angka-angkanya harus tetap tunggal". Bentuk dari Sudoku adalah sebuah kotak besar berukuran 9x9 yang terdiri dari 9 kotak kecil berukuran 3x3. Sesuai dengan arti dari namanya, tujuan dari permainan ini adalah untuk mengisikan angka di antara 1 hingga 9 pada kotak-kotak kosong sedemikian sehingga:

1. Tidak ada angka yang sama/berulang pada setiap baris;
2. Tidak ada angka yang sama/berulang pada setiap kolom;
3. Tidak ada angka yang sama/berulang pada setiap kotak kecil.

Kasus Sudoku yang digunakan adalah seperti gambar berikut :



Pada tugas kecil ini, Sudoku harus direpresentasikan sebagai *Constraint Satisfaction Problem* (CSP). *Tools* yang digunakan adalah AISpace. AIspace akan membaca sebuah *file* XML dengan informasi variabel, domain, dan *constraint* untuk mengenali sebuah CSP. *File* XML bisa dibuat langsung/manual dengan *interface* yang disediakan AIspace, namun akan memakan waktu lama karena ada banyak sekali constraint. Maka, kami memutuskan untuk membuat sebuah *script* sederhana dalam C++ untuk men-*generate file* XML tersebut.

Kemudian, persoalan CSP akan diselesaikan dengan algoritma yang sudah disediakan pada *applet* AIspace.

Terdapat dua *applet* AIspace yang digunakan, yakni :

1. *Consistency Based CSP Solver*. Algoritma yang digunakan memiliki pendekatan konsistensi, yakni mengisi kotak satu per satu sedemikian sehingga *state* saat itu konsisten (*tidak melanggar constraint*). Solusi ditemukan saat *state* sudah *complete* (seluruh variabel pada CSP sudah memiliki nilai). Hanya ada satu algoritma yang dapat digunakan, yakni *arc consistency* + *domain splitting* dengan bantuan *backtracking*. Langkah global pencarian solusi serta parameter-parameter yang digunakan akan dijelaskan kemudian.
2. *Stochastic Local Search Based CSP Solver*. Algoritma yang digunakan memiliki pendekatan *completeness*, yakni terlebih dahulu menginisialisasinilai seluruh variabel dengan cara *random* atau mulai dari nilai domain terendah. Barulah kemudian nilai-nilai variabel yang melanggar *constraint* akan diubah. Solusi ditemukan saat *state* sudah *consistent* (tidak ada variabel yang melanggar *constraint*). Terdapat delapan pilihan algoritma *local search* yang dapat digunakan. Algoritma-algoritma tersebut merupakan kombinasi dari *hill climbing*/*greedy descent*  dengan *stochastic/random moves.* Algoritma mana yang dipilih, langkah global pencarian solusi serta parameter-parameter yang digunakan akan dijelaskan kemudian.

Isi laporan mencakup :

1. Deskripsi persoalan dan *tools* yang digunakan (bab I)
2. Deskripsi variabel, domain tiap variabel, dan *constraint* (bab II)
3. Deskripsi parameter dan pilihan algoritma yang digunakan serta penjelasan singkat pemilihan algoritma (bab III)
4. Hasil solusi untuk masing-masing *applet*, langkah global pencarian solusi dengan algoritma yang dipilih, dan perbandingan proses dan hasil dari kedua *applet* tersebut

BAB II

# Sudoku sebagai CSP : Variabel, Domain, dan *Constraint*

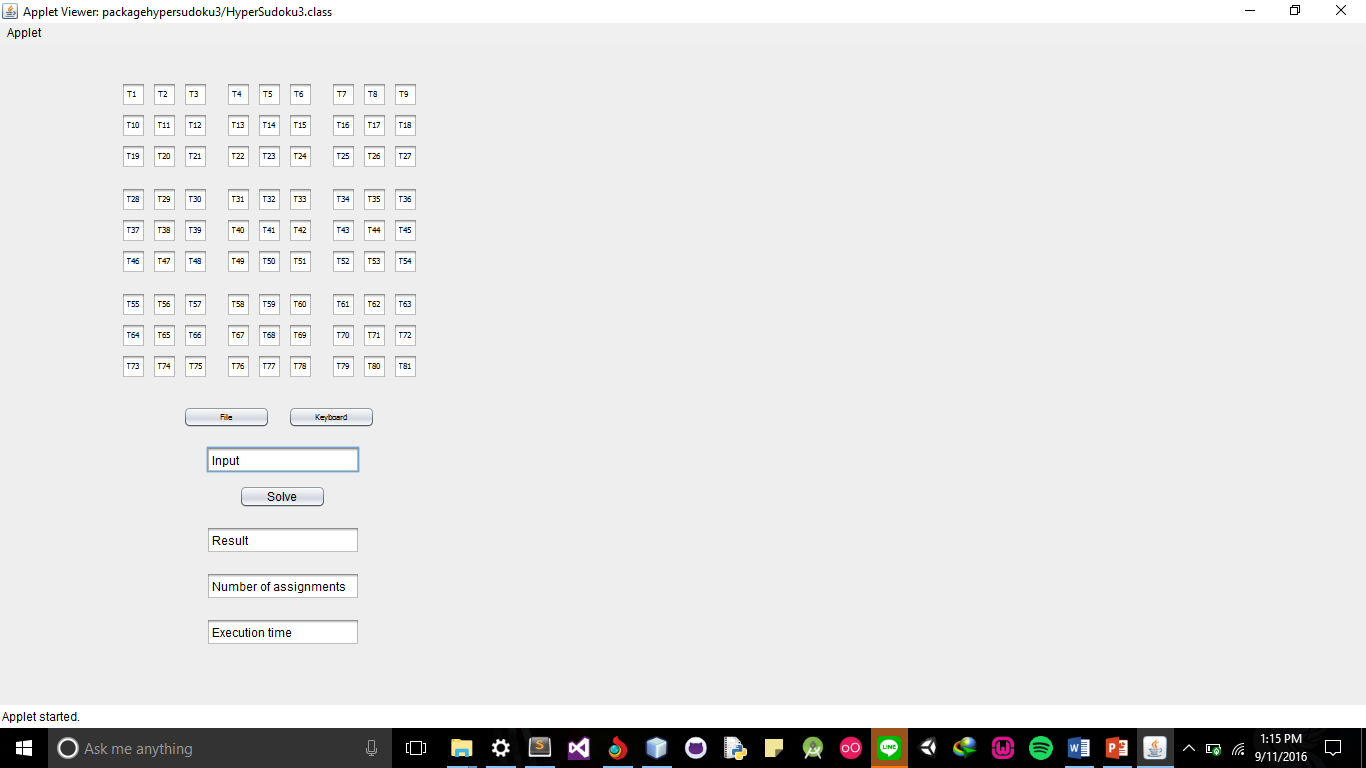
Sudoku sebagai CSP memiliki definisi sebagai berikut :

1. Variabel

Untuk persoalan Sudoku ini, terdapat 81 variabel yang digunakan, masing-masing untuk sebuah kotak/*tile* pada Sudoku. Misalkan set variabel disimbolkan dengan T (*tiles*) dan masing-masing variabel disimbolkan dengan Ti. Maka, definisi variabel untuk persoalan Sudoku ini :

**T = {T1, T2, T3, …  , T79, T80, T81}**

Berikut ilustrasi variabel untuk masing-masing kotak :



1. Domain

Pada dasarnya, domain untuk setiap variabel pada permainan Sudoku adalah bilangan bulat positif dari 1 hingga 9. Misalkan set domain disimbolkan dengan D dan Di adalah domain untuk variabel Ti. Maka, definisi domain adalah sebagai berikut :

**D = { (1,2,3,4,5,6,7,8,9), (1,2,3,4,5,6,7,8,9), … , (1,2,3,4,5,6,7,8,9) } -> 81 kali**

**atau**

**Di = {1,2,3,4,5,6,7,8,9} untuk i bilangan bulat, i = 1,2,3,…,81**

Namun, itu hanya berlaku untuk Sudoku yang belum terisi sama sekali. Dengan definisi domain seperti itu, kotak yang sudah terisi akan memiliki *unary constraint*. Misalkan saja, variabel T1 pada persoalan Sudoku yang diberikan untuk tugas ini sudah terisi dengan 9. Maka, kita harus mendefinisikan salah satu dari *unary constraint* berikut :

**T1= 9**

**atau**

**T1  <> 1 dan T1  <> 2 dan T1  <> 3 … dan T1  <> 8 (<> berarti tidak sama dengan)**

Seperti dikutip dari <https://www.ics.uci.edu/~rickl/courses/cs-171/cs171-lecture-slides/cs-171-08a-ConstraintSatisfactionIntro.pdf>, CSP sebenarnya hanya memerlukan *binary constraint*. Oleh karena itu, untuk menghilangkan *unary constraint*, kita perlu mempersempit domain (menghapus nilai domain yang melanggar *constraint*) untuk variabel-variabel yang sudah terisi. Misalkan, untuk variabel T1, maka domainnya adalah angka yang sudah terisi saja, yakni :

**D1 = { 9 }**

**Nilai domain lain yakni 1,2,3,4,5,6,7,8 dihapus karena melanggar *constraint***

Maka, domain untuk persoalan Sudoku yang diberikan adalah sebagai berikut :

**D4 = D18 = D75 = { 1 }**

**D16 = D50 = D55 = D67 = { 2 }**

**D58 = { 3 }**

**D23 = D78 = { 4 }**

**D9 = D12 = D79 = { 5 }**

**D51 = D63 = { 6 }**

**D40 = D80 = { 7 }**

**D19 = D32 = { 8 }**

**D1 = D14 = D54 = D76 = { 9 }**

**Di = { 1,2,3,4,5,6,7,8,9} untuk i selain di atas**

Berikut contoh *file* XML untuk variabel kotak kosong dan variabel kotak terisi :

1. Kotak kosong (variabel T2)

**<VARIABLE TYPE="Integer">**

**<NAME>T2</NAME>**

**<VALUE>1</VALUE>**

**<VALUE>2</VALUE>**

**<VALUE>3</VALUE>**

**<VALUE>4</VALUE>**

**<VALUE>5</VALUE>**

**<VALUE>6</VALUE>**

**<VALUE>7</VALUE>**

**<VALUE>8</VALUE>**

**<VALUE>9</VALUE>**

**<PROPERTY>position = (800.0, 1075.0)</PROPERTY>**

**</VARIABLE>**

1. Kotak terisi (variabel T1)

**<VARIABLE TYPE="Integer">**

**<NAME>T1</NAME>**

**<VALUE>9</VALUE>**

**<PROPERTY>position = (300.0, 1075.0)</PROPERTY>**

**</VARIABLE>**

1. *Constraint*

Seperti yang sudah disinggung sedikit sebelumnya di bagian domain, kami hanya menggunakan *binary constraint*. Terdapat tiga jenis *binary constraint*, yakni :

1. **Dua kotak/variabel dalam satu baris tidak boleh memiliki nilai yang sama**

Untuk **satu baris**, akan terdapat -> 9C2 = **36 *constraint***. Yakni, kombinasi dua kotak dari total 9 kotak. Maka, untuk **9 baris, total** terdapat 36 \* 9 = **324 *constraint***. Sebagai contoh, berikut *constraint* dari variabel T2 dan T3 yang sama-sama ada pada baris pertama (sama-sama kotak kosong) :

**Nilai** **T2 <> T3**

**atau**

**(T2, T3) ada pada {(1,2), (1,3), … , (1,9), (2,1), (2,3), … , (2,9), … , (9,1), (9,2), … (9,8)}**.

*File* XML yang dibaca AIspace akan menggunakan representasi kedua (me-*list* kombinasi nilai yang mungkin dari kedua variabel). Berikut *list* seluruh *constraint* untuk baris pertama :

**T1<>T2, T1<>T3, … , T1<>T9, T2<>T3, T2<>T4, … , T2<>T9, … , T7<>T8, T7<>T9, T8<>T9**.

Yang perlu diperhatikan adalah jangan sampai ada dua *constraint* berulang. Misal, T1<>T2 dan T2<>T1. Baris lain (selain baris satu) menyesuaikan.

1. **Dua kotak/variabel dalam satu kolom tidak boleh memiliki nilai yang sama**

Total jumlah *constraint* untuk 9 kolom sama seperti total jumlah *constraint* untuk 9 baris, yakni **324 *constraint***. Sebagai contoh, berikut *constraint* dari variabel T28 dan T37 yang sama-sama ada pada kolom pertama (sama-sama kotak kosong) :

**Nilai** **T28 <> T37**

**atau**

**(T28, T37) ada pada {(1,2), (1,3), … , (1,9), (2,1), (2,3), … , (2,9), … , (9,1), (9,2), … (9,8)}**.

Berikut *list* seluruh *constraint* untuk kolom pertama :

**T1<>T10, T1<>T19, … , T1<>T73, T10<>T19, T10<>T28, … , T10<>T73, … , T55<>T64, T55<>T73, T64<>T73.**

Kolom lain menyesuaikan.

1. **Dua kotak/variabel yang berbeda baris dan kolom namun berada dalam satu kotak 3x3 yang sama tidak boleh memiliki nilai yang sama**

Ada kombinasi dua variabel yang berada pada baris yang sama dan kotak 3x3 yang sama, misal T2 dan T3. Ada juga yang berada pada kolom yang sama dan kotak 3x3 yang sama, misal T28 dan T37. *Constraint* seperti ini jangan sampai terulang. Semua *constraint* yang sudah dinyatakan dalam *constraint* baris atau kolom jangan dinyatakan lagi dalam *constraint* kotak 3x3.

Untuk satu kotak 3x3, terdapat 18 *constraint*. Yakni memilih 2 dari 9 kotak -> 9C2 = **36 *constraint***. Karena dalam kotak 3x3 terdapat 3 baris yang terdiri dari 3 kotak, maka jumlah *constraint* yang harus dikurangi karena sudah dihitung pada *constraint* baris adalah -> 3 x 3C2 = 3 x 3 = **9 *constraint*.** Begitu pula jumlah *constraint* yang harus dikurangi karena sudah dihitung pada *constraint* kolom yakni **9 *constraint*.** Sehingga, jumlah *constraint* untuk satu kotak 3x3 -> 36 - 9 - 9 = **18 *constraint***. Total untuk 9 kotak menjadi 18 x 9 = **162 *constraint***.

Sebagai contoh, *constraint* untuk kotak 3x3 pertama (ujung kiri atas) :

**T1<>T5, T1<>T6, T1<>T8, T1<>T9, T2<>T4, T2<>T6, T2<>T7, T2<>T9, T3<>T4, T3<>T5, T3<>T7, T3<>T8, T4<>T8, T4<>T9, T5<>T7, T5<>T9, T6<>T7, T6<>T8.**

Kotak 3x3 lain menyesuaikan.

Maka, sejauh ini kita dapatkan sebanyak 324 + 324 + 162 = **810 *constraint***. Namun, kita ketahui terdapat variabel dengan domain {1,2,3,…,9} -> kotak kosong dan variabel dengan domain satu angka saja misal {9} -> kotak yang sudah terisi dengan angka 9 (soal).

Maka, penulisan *constraint* pada *file* XML pun berbeda-beda untuk kasus-kasus berikut :

1. *Constraint* untuk dua variabel kotak kosong

Kita ambil contoh untuk variabel T2 dan T3. Berikut *constraint* pada XML-nya :

**<CONSTRAINT TYPE="Custom">**

**<CUSTOMNAME>C\_2\_3</CUSTOMNAME>**

**<GIVEN>T2</GIVEN>**

**<GIVEN>T3</GIVEN>**

**<TABLE>**

**F T T T T T T T T T F T T T T T T T T T F T T T T T T T T T F T T T T T T T T T F T T T T T T T T T F T T T T T T T T T F T T T T T T T T T F T T T T T T T T T F**

**</TABLE>**

**<PROPERTY>position = (4622.507, 4338.3887)</PROPERTY>**

**</CONSTRAINT>**

Terdapat 81 pernyataan True (T) atau False (F), yakni permutasi dari domain variabel T2 {1,2,3,..,9} dan domain variabel T3 {1,2,3,…,9}. Pernyataan pertama False menyatakan bahwa kombinasi nilai (T2,T3) -> (1,1) melanggar *constraint* karena nilai variabel dalam satu baris. Sebaliknya, pernyataan kedua True menyatakan bahwa kombinasi nilai (T2,T3) -> (1,2) benar karena nilainya berbeda. Begitu seterusnya hingga kombinasi nilai (9,9).

1. *Constraint* variabel kotak kosong dan variabel terisi

Kita ambil contoh untuk variabel T1 dan T2. Berikut *constraint* pada XML-nya :

**<CONSTRAINT TYPE="Custom">**

**<CUSTOMNAME>C\_1\_2</CUSTOMNAME>**

**<GIVEN>T1</GIVEN>**

**<GIVEN>T2</GIVEN>**

**<TABLE>**

**T T T T T T T T F**

**</TABLE>**

**<PROPERTY>position = (4622.507, 4338.3887)</PROPERTY>**

**</CONSTRAINT>**

Kita lihat hanya ada 9 pernyataan. Hal ini karena variabel T1 sudah terisi dan hanya memiliki domain satu nilai saja, yakni 9. Semua pernyataan True untuk kombinasi (9,1), (9,2), hingga (9,8). Sedangkan, pernyataan salah untuk kombinasi (9,9).

1. *Constraint* dua variabel terisi

Dua variabel yang sudah terisi tidak memerlukan *constraint* lagi karena untuk soal Sudoku yang valid nilai keduanya sudah pasti berbeda (sudah dicek bahwa soal Sudoku yang diberikan untuk tugas ini valid). Terdapat 24 kotak yang sudah terisi pada soal.

Misalkan, untuk **baris pertama**, *constraint* **antara T1 dan T4** tidak perlu dinyatakan. Untuk **kolom pertama**, *constraint* **antara T1 dan T19** tidak perlu dinyatakan. Untuk **kotak 3x3 pertama (ujung kiri atas)**, *constraint* **antara T1 dan T12** tidak perlu dinyatakan.

Sesudah dihitung, ada sebanyak **27** kombinasi antar dua variabel terisi sebaris (untuk seluruh 9 baris), **30** kombinasi antar dua variabel terisi sekolom (untuk seluruh 9 kolom), dan **16** kombinasi antar variabel satu kotak 3x3 (untuk seluruh 9 kotak 3x3). Jadi, **total terdapat 73 *constraint* yang tidak harus dinyatakan pada *file* XML**..

Sehingga, **total *binary* *constraint*** pada *file* XML ada **sebanyak** 810 - 73 = **737 *constraints***.

BAB III

# Parameter dan Pilihan Algoritma

BAB IV

# Hasil Solusi, Langkah Pencarian, dan Perbandingan