# PENENTUAN SOLUSI SATISFIABILITY (SAT) PROBLEM DENGAN METODE KOHONEN SELF-ORGANIZING MAP (K-SOM)


Disusun Oleh:

N a m a : Alexander Ramadhan Suratinoyo

NIM : 16523073

**PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA**

**FAKULTAS TEKNOLOGI INDUSTRI**

**UNIVERSITAS ISLAM INDONESIA**

**2020**

**HALAMAN PENGESAHAN DOSEN PEMBIMBING**

# PENENTUAN SOLUSI SATISFIABILITY (SAT) PROBLEM DENGAN METODE KOHONEN SELF-ORGANIZING MAP (K-SOM)

## TUGAS AKHIR

Disusun Oleh:

N a m a : Alexander Ramadhan Suratinoyo

NIM : 16 523 073

Yogyakarta, 20 Juli 2020

Pembimbing,

( Taufiq Hidayat, S.T, M.Cs )

**HALAMAN PENGESAHAN DOSEN PENGUJI**

# TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Teknik Informatika di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 20 Juli 2020

Tim Penguji

Taufiq Hidayat, S.T, M.Cs. _____

**Anggota 1**

Dr. Syarif Hidayat, S.Kom., M.I.T. _____

**Anggota 2**

Ahmad Fathan Hidayatullah, S.T, M.Cs. _____

Mengetahui,
Ketua Program Studi Informatika – Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

( Dr. Raden Teduh Dirgahayu, S.T., M.Sc. )

# HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR



HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Alexander Ramadhan Suratinoyo

NIM : 16523073

Tugas akhir dengan judul:

**PENENTUAN SOLUSI SATISFIABILITY (SAT) PROBLEM DENGAN METODE KOHONEN SELF-ORGANIZING MAP (K-SOM)**

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila dikemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung resiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 20 Juli 2020


( Alexander Ramadhan Suratinoyo )

# HALAMAN PERSEMBAHAN

Karya ini saya persembahkan untuk

Ayah dan Bunda tercinta dan terkasih

Terimakasih atas kasih sayang yang berlimpah dari mulai lahir hingga saya sudah sebesar ini. Terimakasih atas limpahan doa yang tak berkesudahan, serta segala hal yang telah ayah dan bunda lakukan selama ini, semua yang terbaik. Karya ini saya persembahkan untuk kalian sebagai wujud terimakasih atas pengorbanan dan jerih payah kalian sehingga saya dapat menggapai cita-cita. Kelak cita-cita ini akan menjadi persembahan yang paling mulai untuk Ayah dan Bunda dan semoga dapat membahagiakan kalian.

Adik-adikku

Terimakasih kepada adik-adikku yang selalu mendoakan saya, dan selalu memberikan saya semangat dalam mengerjakan tugas akhir ini. Walaupun dulu saat dekat kita selalu bertengkar, selalu berbeda pendapat, namun pada saaat jauh kita saling merindukan. Semoga ini merupakan awal dari kesuksesan penulis yang dapat membanggakan dirimu dan semoga .

Dosen pembimbing

Terimakasih banyak kepada bapak Taufiq Hidayat selaku dosen pembimbing. Terimakasih banyak kepada bapak, karena telah menjadi orang tua kedua saya didalam kampus ini. Terimakasih atas bimbingan, ilmu yang telah bapak berikan dan dukungan yang selalu bapak berikan kepada saya agar saya selalu bersemangat dalam mengerjakan tugas akhir ini dengan baik.

Sahabat serta teman-teman yang ada dikampus ini

Terimakasih banyak telah menjadi teman selama saya berada di kampus ini. Tanpa kalian, saya bukan apa-apa. Saya meminta maaf apabila ada salah kata maupun perbuatan yang saya lakukan kepada teman-teman semua. Terimakasih banyak atas dukungan yang sangat luar biasa, terimakasih banyak karena saya bisa berada pada tahap untuk menyelesaikan skripsi ini dengan baik.

## HALAMAN MOTO

Waktu bagaikan pedang. Jika kamu tidak memanfaatkannya dengan baik, maka ia akan memanfaatkanmu

(HR.Muslim)

Dunia ini ibarat bayangan. Kalau kamu berusaha menangkapnya, ia akan lari. Tapi kalau kamu membelakanginya, ia tak punya pilihan selain mengikutimu.

(Ibnu Qayyim Al Jauziyyah)

Bukanlah ilmu yang semestinya mendatangimu, tetapi kamulah yang seharusnya mendatangi ilmu itu.

(Imam Malik)

Sesungguhnya jika kamu bersyukur, pasti Kami akan menambah (nikmat) kepadamu, namun jika kamu meningkari (nikmat-Ku), maka sesungguhnya azab-Ku sangatah pedih.

(QS. Ibrahim: 7)

# KATA PENGANTAR

Segala puji dan syukur atas kehadirat ALLAH SWT atas berkat, rahmat serta karunia dan hidayah-NYA yang selalu senantiasa dilimpahkan kepada penulis, sehingga penulis bisa menyelesaikan tugas akhir dengan judul PENENTUAN SOLUSI SATISFIABLITY (SAT) PROBLEM DENGAN METODE KOHONEN SELF-ORGANIZING MAP (K-SOM). Sholawat serta salam selalu senantiasa tercurahkan kepada Baginda Muhammad SAW yang mengantarkan umat manusia dari zamin kegelapan ke zaman yang terang benderang dengan penuh ilmu pengetahuan seperti saat sekarang ini

Dalam penyusunan tugas akhir ini, banyak hambatan serta ujian yang penulis hadapi. Namun, pada akhirnya penulis dapat melaluinya berkat adanya bimbingan serta bantuan dari berbagai pihak baik secara moral maupun spiritual. Untuk itu, pada kesempatan ini peulis menyampaikan ucapan terimakasih yang sebanyak-banyaknya kepada:

1. Orang tua serta adik atas segala, doa, usaha dan kasih sayang tak terbalaskan untuk penulis.
2. Dekan Fakultas Teknologi Industri, Universitas Islam Indonesia.
3. Ketua Jurusan Program Studi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia.
4. Bapak Taufiq Hidayat S.T, M.C.S selaku dosen pembimbing laporan tugas akhir yang sudah banyak membantu saya dalam mengerjakan tugas akhir baik secara motivasi, ilmu serta bimbingan yang bapak telah berikan kepada saya.
5. Keluarga besar Informatika Universitas Islam Indonesia yang telah menyumbangkan pengalaman yang sangat berharga bagi saya dalam menuntut ilmu dikampus ini.
6. Keluarga besar Himpunan Teknik Informatika yang telah memberikan saya banyak pengalaman.
7. Keluarga besar HexaDecima, yang saya sayangi dan selalu saya banggakan karena telah menjadi keluarga saya dikehidupan perantauan ini.
8. Grup KOBE yang telah memberikan pengalaman baru kepada saya.
9. Grup “U**** Hunter’ yang telah menemani saya selama pada masa pandemi
10. Grup “Keluarga Spotify” yang telah menjadi bagian saya untuk mencurahkan isi hati saya selama ini.
11. Kepada keluarga besar konsenstrasi Sistem Cerdas Program Studi Informatika.

12. Kepada teman saya Rizal Hamdan yang sangat membantu dalam melakukan penelitian tugas akhir ini.
13. Kepada kontrakan MUSLIMIN yang telah memberi saya tumpangan dalam masa-masa akhir kampus saya sebagai mahasiswa tingkat akhir.
14. Kepada Ainayya, Veda, Firza, sebagai tempat saya berkeluh kesah saat senang maupun susah.
15. Kepada teman-teman SINTANGNISASI yang telah membantu saya untuk dapat beradapatasi diperantauan ini.
16. Kepada semua pihak yang telah membantu saya dalam melaksanakan Tugas Akhir yang tidak dapat saya sebutkan satu persatu.

Semoga tugas akhir ini bisa bermanfaat di bidang akademik maupun non-akademik dan semoga dapat

Penulis menyadari bahwa penulisan Laporan Tugas Akhir ini masih banyak kekurangan. Karena hal ini semata-mata karena keterbatasan kemamapuan dan pengetahuan penulis. Oleh karena itu, penulis sangat terbuka untuk menerima kritik dan saran yang membangun.

Harapan penulis semoga laporan Tugas Akhir ini yang telah diselesaikan, bermanfaat dan bisa menjadi pembelajaran untuk semua pihak.

Yogyakarta, 20 Juli 2020

( Alexander Ramadhan Suratinoyo )

# SARI

SAT *Problem* (Boolean Satisfiablity Problem) merupakan salah satu permasalahan *semantic* yang menentukan ada atau tidak pemberian nilai kebenaran pada setiap proposisi yang memberikan hasil *satisfiable* pada formula logika *boolean*. SAT *Problem* masuk dalam permasalahan NP-*Complete* ( *Non Polynomial-Complete*), sehingga kasus terburuk memiliki kompleksitas waktu yang tidak dibatasi dengan fungsi polinom. Apabila formula proposisi yang ingin diketahui nilai *satisfiable* memiliki ukuran yang relatif besar, maka hal ini akan sulit untuk dikerjakan dengan cara konvensional karena akan banyak memakan waktu dan biaya (*resource*). Sudah banyak solusi yang ditawarkan berbentuk eksak, namun dalam penelitian kali ini, bertujuan untuk menambah sumber komputasi dengan dengan menggunakan metode salah satu dari *neural network*, yaitu metode *Kohonen Self-Organizing Map* (K-SOM).

Kohonen Self Organizing Map (K-SOM) merupakan salah satu *neural network* yang bersifat *unsupervised learning* atau tanpa pembelajaran yang membagi pola masukan ke dalam beberapa kelompok (klaster). Metode ini cocok digunakan pada saat memiliki data yang sangat besar, cocok untuk menyelesaikan SAT *Problem* dalam bentuk eksak karena kohonen ini merupakan salah satu bentuk dari *soft computing*, dimana *soft computing* ini dapat menyelesaikan problematika yang sulit serta menghitung kuantifikasi data dalam jumlah skala yang cukup besar dibanding dengan *hard computing*.

Dalam penelitian ini, SAT *Problem* akan direpresentasikan dalam bentuk CNF (*Conjunctive Normal Form*) dan akan diselesaikan dengan menggunakan metode *Kohonen Self Organizing Map* (K-SOM) dalam bentuk eksak. Kemudian penyelesaian tersebut akan diimplementasikan dalam bahasa pemograman *python* untuk membuat SAT Solver (*software*/perangkat lunak dalam menyelesaikan SAT *Problem*).

Setelah SAT *Solver* terbentuk, selanjutnya akan dilakukan pengujian, kesimpulan yang dapat diambil dari pengujian tersebut adalah SAT Solver yang dikembangkan dari penelitian ini dapat menyelesaikan SAT *Problem*.

Pada penelitian kali ini, terbentuk sebuah SAT Solver yang bernama Ander-SAT yang dapat menghitung SAT Problem menggunakan metode Kohonen Self-Organizing Map dengan jumlah data sebanyak 30 dengan format .cnf dan rata-rata jumlah klausa pada data tersebut lebih dari 1000 klausa dengan tingkat akurasi sebesar 83% dan dalam waktu relatif singkat yaitu kurang dari 15 detik.

Kata kunci: SAT *Problem*, SAT *Solver, Kohonen Self Organizing Map* (K-SOM).

# GLOSARIUM

| | |
|---|---|
| Boolean | Tipe data yang hanya mempunyai dua nilai, yaitu benar dan salah. Pada beberapa bahasa pemograman, nilai *true* dapat diganti dengan 1 dan nilai *false* diganti dengan 0. |
| Python | Merupakan bahasa pemograman yang dapat digunakan diberbagai komputer. |
| Integer | Tipe data untuk bilangan bulat. |
| Method | Sarana bagi pemograman yang memodularisasi atau memecah program kompleks menjadi bagian yang kecil sehingga dapat digunakan berulang-ulang, daripada membuat baris kode yang sama. |
| Flowchart | Suatu bagan dengan simbol-simbol tertentu yang menggambarkan urutan proses secara mendetail dan hubungan antara suatu proses (instruktsi) dengan proses lainnya dalam suatu program. |

# DAFTAR ISI

# DAFTAR TABEL

# DAFTAR GAMBAR

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

SAT *Problem* merupakan salah satu masalah *semantic* yang menentukan ada atau tidaknya pemberian nilai kebenaran pada setiap proposisi yang memberikan hasil *satisfiable* pada formula logika *boolean*. Formula dalam SAT Problem berbentuk CNF (*Conjunctive Normal Form*) yang membuat logika proposisi hanya memiliki unsur berupa variabel proposisi (untuk selanjutnya disebut dengan variabel), negasi, operator or, dan *and*. Variabel tersebut diberikan nilai *true* atau *false*, sehingga akan memiliki hasil akhir berupa nilai *true* atau *false* (Pratama, 2018). Oleh karena itu, untuk mempermudah proses penyelesaian SAT *problem*, di butuhkan SAT *Solver*. SAT *Solver* merupakan perangkat lunak yang bisa memecahkan SAT *Problem* yang berfungsi menyelesaikan sebuah formula logika itu apakah formula tersebut bernilai *satisfiable* atau *unsatisfiable*.

Dalam permasalahan yang kompleks, terkadang SAT *Solver* mendapat kesulitan. Hal ini dapat terjadi karena SAT *Solver* merupakan perangkat lunak yang dibuat dengan alogirtma *basic* dan dapat menyelesaikan masalah yang tergolong mudah atau *simple*. Oleh karena itu, tingkat optimasi algoritma dalam membentuk SAT *Solver* merupakan hal yang sangat penting. Semakin tinggi peningkatan SAT *Solver*, maka semakin baik kualitas solusi yang didapatkan. Kemudian proses komputasi yang dibutuhkan memilki waktu yang lebih sedikit dan *resource* yang dibutuhkan lebih rendah dari SAT *Solver* terdahulu.

Dalam beberapa tahun terakhir tentang penelitian pembentukan SAT *Solver* yang efisien untuk menyelesaikan problem berkembang dengan cepat. Perkembangan ini sudah banyak berkontribusi khususnya pada kemajuan teknologi manusia yang secara otomatis dapat memecahkan masalah yang mengimplikasi puluhan bahkan ratusan ribu variabel dan jutaan klausa (Pratama, 2018). Salah satu contoh nyata adalah dalam permasalahan Electronic Design Automation atau yang biasa disebut EDA. Masalah yang ada pada EDA meliput *formal equivalence checking, model checking, formal of microprocessor* (Bryan, German, & Velev, 1999).

*Kohonen Self Organizing Map* (K-SOM) merupakan salah satu metode dalam *neural network* yang bersifat *unsupervised learnin*g atau tanpa pembelajaran yang pertama kali diperkenalkan oleh Teuvo Kohonen pada tahun 1996. Pada algoritma K-SOM, bobot vektor untuk setiap unit klaster berfungsi sebagai contoh dari input pola yang terkait dengan klaster tersebut. Selama proses *self-organizing*, klaster satuan yang bobotnya sesuai dengan pola vektor input yang paling dekat menjadi pemenang. Unit dari pemenang dan klaster tersebut terus mempebarui bobot mereka. Kohonen ini dapat menyelesaikan SAT Problem karena merupakan salah satu *soft computing* dimana *soft computing* ini dapat menyelesaikan problematika yang sulit serta menghitung kuantifikasi data dalam jumlah skala yang cukup besar dan waktu yang relatif singkat dibanding dengan *hard computing*

Sudah banyak SAT Solver yang dikembangkan dengan algoritma berbeda yang efektif dan efisien seperti, WalkSAT (Marques-Silva & Sakallah, 1997), GRASP yang mampu menyelesaikan SAT Problem dalam jumlah yang besar, namun membutuhkan komputasi dengan waktu 10.000 detik (3jam) (Marques-Silva & Sakallah, 1997), zChaff (Moskewicz, 2001), Satz (Li & Anbulagan, 1997) dan lain-lain. Namun, pada sebagian SAT *Solver* tersebut masih membutuhkan waktu komputasi yang cukup lama dalam proses eksekusinya.

Dalam penyelesaian SAT *Problem* dalam bentuk eksak adalah dengan pemberian nilai kebenaran kepada variabel proposisi sehingga tepat menyebebakan setiap klausa penyusun formula proposisi bernilai benar. Formula proposisi dinyatakan "*satisfiable*" jika dan hanya jika seluruh klausa penyusunnya bernilai benar, dan sebaliknya akan bersifat "*unsatisfiable*".

Oleh karena itu, penelitian kali ini akan mencoba menambah sumber komputasi SAT *Solver* (*software*/perangkat lunak) dengan menggunakan bahasa pemograman *Python* dan menggunakan metode *Kohonen Self-Organizing Map (K-SOM)*. Dengan menggunakan metode ini, diharapkan mampu dapat menambah sumber komputasi eksak dan diharapakan mampu meningkatkan kinerja SAT Solver dengan metode *Kohonen Self-Organizing Map* (K-SOM).

## 1.2 Rumusan Masalah

Berdasarkan latar belakang masalah di atas adalah mengimplementasikan metode Kohonen-Self Organizing Maps (K-SOM) pada SAT Problem yang memiliki permasalahan

komputasi pada waktu yang cukup lama dalam melakukan proses eksekusi, sehingga menghasilkan SAT Solver yang efektif.

## 1.3 Batasan Masalah

Batasan masalah pada penelitian ini adalah sebagai berikut:

1. Formula yang diselesaikan berbentuk CNF (*Conjunctive Normal Form*).
2. Data yang digunakan berasal dari SAT *Competition*.
3. Masukan serta keluaran program berbentuk teks.

## 1.4 Tujuan Penelitian

Adapun tujuan yang ingin dicapai dari penelitian ini adalah:

1. Membangun SAT *Solver* (*software*/perangkat lunak yang dapat menyelesaikan SAT *Problem*)
2. Mengimplementasikan Kohonen Self-Organizing Map (K-SOM) untuk menyelesaikan SAT *Problem* pada SAT *Solver*.

## 1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah sebagai berikut:

1. Mempermudah serta mempercepat penyelesaian SAT *Problem*.
2. Mengatasi kompleksitas penyelesaian SAT *Problem* dengan *Kohonen Self-Organizing Map* (K-SOM).
3. Sebagai bahan rujukan untuk pengembangan penelitian yang akan datang tentang SAT *Problem*.

## 1.6 Metodologi Penelitian

Dalam membuat suatu karya ilmiah, tentu perlu suatu metodologi penelitian agar penelitian yang dilakukan dapat berjalan dengan baik serta terarah. Metodologi penelitian digunakan dalam menyusun makalah ini meliputi identifikasi masalah, analisis kebutuhan, implementasi, dan pengujian. Hal ini dilakukan agar dalam penyelesaian tugas akhir dapat lebih mudah dan terarah. Diantaranya:

a. Identifikasi Masalah

Mengidentifikasi masalah, kebutuhan, cara kerja, dan ruang lingkup *software*/perangkat lunak yang akan dibangun dengan cara:

1. Membaca literatur ilmiah artikel/jurnal, makalah, buku, atau penelitian-penelitian yang terkait denga penelitian ini.
2. Melakukan analisis terhadap permasalahn SAT *Problem* dan cara penyelesaiannya.
3. Melakukan analisis terhadap Kohonen Self-Organizing Map (K-SOM) serta implementasinya untuk menyelesaikan permasalahan SAT *Problem*.
4. Melakukan analisis penggunaan bahasa pemograman *Python* untuk mengimplementasikan solusi SAT *Problem* menggunakan *Kohonen Self Organizing Map* (K-SOM).

b. Analisis Kebutuhan

Pada tahapan ini dilakukan sebuah analisis untuk mengetahui kebutuhan apa saja yang diperlukan untuk mengembangkan sebuah *software*/perangkat lunak (SAT Solver) berbasis bahasa pemograman *Python* yang dapat menyelesaikan permasalahn SAT *Problem* menggunakan Kohonen Self-Organizing Map (K-SOM). Pada tahapan ini juga mencakup analisis kebutuhan perangkat lunak serta perangkat keras apa saja yang dibutuhkan dalam melakukan pengembangan *software*/perangkat lunak tersebut.

c. Perancangan

Pada tahapan ini dilakukan perancangan SAT Solver dengan *Kohonen Self-Organizing Map* (K-SOM) dalam bentuk eksak agar dapat menyelesaikan SAT *Problem*.

d. Implementasi

Pada tahapan ini dilakukan implementasi dari perancangan SAT *Solver* yang telah dilakukan pada tahap sebelumnya. Impelementasi tersebut menggunakan bahasa pemograman *Python*.

e. Pengujian

Pada tahapan ini, SAT *Solver* yang telah dikembangkan harus diuji sehingga dapat diketahui hasilnya. Dari pengujian tersebut, akan diketahui apakah *SAT Solver* yang dikembangkan sudah sesuai dengan kebutuhan atau belum. Dalam penelitian ini, pengujian berbentuk penyelesaian SAT *Problem*.

## 1.7 Sistematika Penulisan

Sistematika penulisan digunakan sebagai pedoman/gambaran mengenai peneltiian yang akan dijalankan. Sistematika penulisan dibagi menjadi lima bab sebagai berikut:

BAB I PENDAHULUAN

Pendahuluan akan membahas permasalahan umum tentang latar belakang msalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian serta sistematika penulisan penelitian penyelesaian permasalahan SAT *Problem*.

BAB II LANDASAN TEORI

Pada bab ini, akan dijelaskan teori-teori yang akan digunakan dalam penelitian untuk penyelesaian permasalahan SAT *Problem*. Adapaun teori yang akan disampaikan meliputi SAT *Problem*, *Kohonen Self-Organizing Map* (K-SOM), serta Python.

BAB III METODOLOGI

Pada bab ini, akan menjabarkan tentang analisis kebutuhan dan perancangan software/perangkat lunak untuk menyelesaikan SAT *Problem* dalam penelitian ini.

BAB IV IMPLEMENTASI DAN PENGUJIAN

Pada bab ini, akan menjelaskan mengenai langkah-langkah dan hasil implementasi serta pengujian SAT *Solver* yang telah dikembangkan.

BAB V KESIMPULAN DAN SARAN

Pada bab ini, akan dijabarkan kesimpulan yang didapatkan pada pengujian serta penelitian ini secara keseluruhan. Selain itu akan diberikan saran untuk penelitian selanjutnya.

# BAB II
# LANDASAN TEORI

## 2.1 Tinjauan Pustaka

Tinjauan pustaka yang telah dilakukan adalah dengan mempelajarai beberapa hasil penelitian yang berupa jurnal, serta makalah ilmiah. Khususnya yang terkait dengan topik-topik SAT Problem, SAT Solver, serta Kohonen Self Organizing Map (K-SOM). Berikut merupakan rangkuman:

Penelitian mengenai SAT Solver terfokus pada algoritma pembentuknya. Sudah banyak penelitian pengembangan SAT Solver dengan berbagai jenis algoritma yang efektif dan efisien. penelitian yang dilakukan oleh Marques Silva & Sakallah yang berjudul "GRASP-A New Search Algorithm for satisfiability. Membuat kerangka algoritmik terintegrasi untuk SAT yang menjelaskan tentang teknik untuk memangkas pencarian beserta tambahannya. Metode Grasp ini didasarkan pada kemungkinan konflik pada saat melakukan pencarian dan salah satu fitur yang membedakan adalah adanya augmentasi backtracking dasar dengan prosedur analisis konflik yang kuat. Menganalisis konflik untuk menentukan klausa dapat memungkinkan GRASP untuk melakukan backtrack serta non-kronologis ke level sebelumnya pada bagian pencarian. Selain itu, dengan mencatatat penyebab konflik, GRASP dapat mengenali dan mencegah terjadinya konflik serupa dikemudian hari dalam melakukan pencarian. Kemudian, pembukuan langsung dari rantai kausalitas dapat mengarah kepada konflik yang memungkinkan GRASP untuk melakukan identifikasi tugas yang diperlukan untuk menemukan solusi. Hasil dari percobaan menunjukan bahwa teknik analisis konflik yang diberikan ke algoritma SAT bisa efektif untuk contoh kelas SAT dalam jumlah yang besar, namun kurang efisien dikarenakan komputasi yang diperlukan cukup lama dikarenakan pada saat komputasi memerlukan waktu sekitar 10.000 detik ( 3 jam ) dengan jumlah klausa yang besar.

Pada penelitian yang dilakukan oleh Matthew W. Moskewicz pada tahun 2001 yang berjudul "Chaff: Engineering an Efficient SAT Solver". Penelitian ini bertujuan melakukan efisiensi terhadap metode Boolean Constraint Propogation (BCP) dan strategi keputusan. Chaff ini berhasil memperoleh peningkatan performa sebanyak satu atau dua kali lipat efektif pada benchmark SAT dibandingkan SAT Solver yang lain seperti GRASP dan SATO. Pada data

SSS-SAT.1.0, GRASP melakukan komputasi sebesar 1000 detik, sedangkan dengan Chaff mampu melakukan komputasi dengan 100 detik.

Penelitian yang dilakukan oleh Chu Min Li & Anbulagan yang berjudul "Heuristics Based on Unit Propagation for Satisfiability Problems pada tahun 1997. Penelitian ini menggabungkan unit propagation pada alogritma DPLL dengan kejadian maksimum pada klausa *heuristic* dengan ukuran minimum bernama Satz. Berdasarkan percobaan evaluasi dari berbagai alternatif yang berbeda, unit propagation pada penelitian ini lebih diutamakan dibanding dengan SAT Solver seperti C-SAT, Tableau, POSIT. Pada hasil penelitian ini, Satz mampu menyelesaikan SAT Problem dengan 300 variabel serta 300 *problem* dalam waktu 34 detik, lebih cepat dibanding dengan SAT Solver C-SAT, Tableau serta POSIT.

Perbedaan dari penelitian sebelumnya dengan penelitian ini adalah dalam waktu proses penyelesaian SAT Problem. Penelitian terdahulu membutuhkan waktu lebih lama dibandingkan penelitian ini dalam penyelesaian SAT Problem.

## 2.2 Dasar Teori

Dasar Teori untuk penyelesaian SAT *Problem* merupakan teori yang digunakan sebagai acuan/pedoman dalam melakukan penelitian yang sedang dilakukan. Berikut merupakan penjelasan dibawah ini:

### 2.2.1 SAT Problem (Boolean Satisfiability Problem) dan SAT Solver

SAT *Problem* adalah proses menentukan sebuah formula yang bernilai *satisfiable* atau *unsatisfiable*. Sebuah formula dapat dikatakan *satisfiable* apabila terdapat kombinasi nilai kebenaran komponen (variabel) pembentuknya yang membuat formula tersebut bernilai benar / TRUE dan sebaliknya jika komponen penyusunnya membuat formula menjadi salah / FALSE maka disebut *unsatisfiable* (Hidayat, 2013). Contohnya sebagai berikut:

a. $l \wedge m$: *satisfiable*, jika l=T dan m=T, maka formula menjadi *true*.
b. $l \wedge (m \vee \neg n)$: *satisfiable*, jika. l=T. dan n=F, maka formula menjadi *true*.
c. $l \wedge m \wedge (\neg l \vee \neg m)$ : *tidak satisfiable*.
d. $(l. \vee \neg m) \wedge m \wedge (\neg l \vee \neg n)$ : *satisfiable* karena akan bernilai *true* jika salah satu adalah m = T, l = F, dan n=F.

### 2.2.2 Formula Proposisi dan Formula CNF (Conjunctive Normal Form)

Salah satu bentuk formula dalam format logika yang dibuat dari kalimat proposisi, yaitu kalimat yang telah ditentukan dengan nilai kebenaran. Logika proposisi dinyatakan dalam sebuah variabel proposisi yang hanya memiliki 2 kemungkinan nilai, yaitu *true* atau *false* (Hidayat, Logika Proposisi, 2014).

Sedangkan CNF (*Conjunctive Normal Form*) adalah bentuk dari formula proposisi yang sudah memenuhi syarat formula dalam bentuk *normal form* (Hidayat, Logika Proposisi, 2014).

Syarat dari CNF adalah:

a. Operator hanya ∧., ∨., dan ¬.
b. Hanya operator ¬ yang boleh diterapkan di variabel proposisi.

Berikut definisi dari CNF sebagai berikut:

formula X dapat dikaitkan dengan CNF jika dari sebuah formula tersebut berbentuk:

$X = A1 \wedge A2 \wedge A3 \wedge \ldots \wedge An$

Dengan:

a. $Ai = Bi1 \vee Bi2 \vee \ldots \vee Bi,mi$
b. $Bij = p$ atau $\neg p$

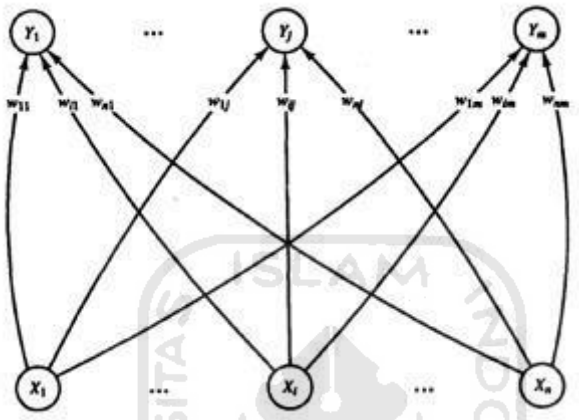
p merupakan proposisi, B disebut dengan literal, dan A adalah sebagai klausa.

Berikut ini merupakan contoh bentuk formula-formula CNF(Conjunctive Normal Form). Agar mempermudah membaca formula, jika literal pada klausa lebih dari satu, maka literal itu ditulis dalam tanda kurung. Contoh:

1. $l \wedge m$
2. $l \wedge m \wedge (\neg l \vee \neg m)$
3. $l. \wedge. (m. \vee \neg n)$
4. $(l \vee \neg m) \wedge m \wedge (\neg l \vee \neg n)$
5. $(l \vee \neg m) \wedge (m \vee \neg n) \wedge (n \vee \neg l)$
6. $(l \vee \neg m) \vee (n \vee \neg o)$
7. $(l \vee \neg m) \vee ( n \vee \neg o) \wedge (m \vee o)$

### 2.2.3 Kohonen Self-Organizing Map (K-SOM)

Kohonen Self-Organizing Map (K-SOM) merupakan salah satu *neural network* yang bersifat *unsupervised learning* atau tanpa pembelajaran yang ditemukan oleh peneliti asal Finlandia yang bernama Teuvo Kohonen pada tahun 1982. Tujuan dari K-SOM ini adalah untuk melakukan visualisasi data dengan cara mengurangi dimensi data melalui *high-dimensional* data yang dipetakan dalam bentuk *low-dimensional* data (Kohenan, 1989).



Gambar 2. 1 Arsitektur K-SOM
Sumber: (Fausett, 1993)

Menurut (Haykin, 1999) ada 3 elemen penting dari metode K-SOM, yaitu:

1. *Competition*

   Untuk setiap pola *input*, *neuron* masing-masing fungsi diskriminan yang memberi dasar untuk kompetisi. *Neuron* tertentu yang memiliki nilai terkecil dari fungsi diskriminan dinyatakan sebagai pemenang.

2. *Cooperation*

   *Neuron* pemenang menentukan lokasi spasial dari lingkungan topologi *excited neuron* untuk memberi dasar kerjasama dalam suatu lingkungan neuron.

3. *Synaptic Adaption*

   *Excited neuron* menurunkan nilai fungsi diskriminan yang berkaitan dengan pola *input* melalui penyesuaian bobot terkait sehingga respon dari *neuron* pemenang keaplikasi selanjutnya dengan pola *input* yang sama akan meningkat.

Setiap bobot vektor untuk setiap unit klaster berfungsi sebagai contoh dari input pola yang terkait dengan klaster tersebut. K-SOM mempunyai lapisan *input layer* dan lapisan *output layer*. Setiap node dari *input* terkoneksi satu sama lain terhadap *node* pada *output layer*. Setiap neuron output mempunyai bobot kepada masing-masing neuron input. Setiap *input* yang diberikan dihitung jarak *euclid* dengan setiap *neuron output*. Kemudian melakukan pencarian *neuron output* yang memiliki jarak paling minimum. Neuron yang mempunyai jarak terkecil disebut *neuron* pemenang.

A. Algoritma Kohonen Self-Organizing Map (K-SOM)

Berikut ini merupakan algoritma dari K-SOM, sebagai berikut:

0. Menginisiasi beberapa hal dibawah ini :
   a. Tetapkan bobot $w_{ij}$

      Nilai bobot $w_{ij}$ dipilih secara acak yang berada pada jarak nilai input.
   b. Tetapkan pembelajaran rate (α)

      Nilai pembelajaran rate yang digunakan merupakan nilai yang telah ditentukan
   c. Menentukan nilai radius R

1. Melakukan langkah dua sampai delapan pada saat kondisi untuk berhenti tidak dapat dipenuhi.
2. Melakukan langkah tiga sampai lima tiap masing vektor *input* $x_i$.
3. Melakukan perhitungan jarak antara bobot $w_{ij}$ dengan vektor *input* $x_i$ disetiap klaster :

$$D(j) = \sum_i (Wij - Xi)^2 \qquad (2.1)$$

   Dimana:

   $Dj$ = Jarak *Euclidean*

   $Wij$ = Bobot yang menghubungkan antara vektor *input xi* menuju unit *yi*
4. Nilai Dj yang paling minimum adalah sebagai pemenangnya.
5. Memperbarui bobot wij dari pemenang nilai vektor, menggunakan persamaan:

$$W_{ij}(\text{new}) = W_{ij}(\text{old}) + \alpha\,[\,x_i - w_{ij}(\text{old})\,] \qquad (2.2)$$

Keterangan:

$w_{ij}$ (*new*) = bobot $w_{ij}$ yang baru

*wij* (*old*) = bobot $w_{ij}$ awal

α = nilai *learning rate*

$x_i$ = vektor *input* ke-1

6. Memperbarui learning rate menggunakan persamaan:

$$\alpha(t+1) = 0.5 \; x \; \alpha(t) \qquad (2.3)$$

7. Mereduksi nilai radius R untuk mencapai nilai minimum yang *learning rate* yang diinginkan
8. Menguji kondisi pemberhentian iterasi.

### 2.2.4 Python

*Python* merupakan bahasa interpreatif yang mendukung multiparadigma pemograman namun tidak dibatasi pada pemograman berorientasi objek, pemograman *imperative* dan pemograman fungsional. Bahasa pemograman *python* memiliki bahasa yang lebih dinamis dan memiliki tata bahasa yang mudah untuk dipelajari (Python (Bahasa Pemograman), n.d.).

# BAB III
# METODOLOGI

## 3.1 Analisis Kebutuhan

Merupakan tahapan dalam melakukan proses pengumpulan data serta berbagai macam informasi yang akan digunakan untuk dalam membuat perangkat lunak. Aplikasi tersebut nantinya dirancang untuk mendapatkan sebuah hasil dari masalah yang sudah dibentuk sebelumnya. Metode ini nantinya digunakan adalah dengan menggunakan metode studi pustaka. Studi pustaka merupakan suatu kegiatan untuk mendapatkan informasi dalam pengumpulan data dengan membaca dan diperoleh dari karya ilmiah, buku-buku yang dalam penelitian ini membahasa tentang SAT *Problem* dan *Kohonen Self-Organizing Map*, serta mencari sebuah laporan tentang *hardware* maupun *software* yang sesuai agar sistem yang digunakan dapat berjalan lancar dan berfungsi dengan semestinya.

### 3.1.1 Analisis Kebutuhan Fungsi

Merupakan suatu tahapan dimana dilakukan penetapan fungsi yang dapat dijalankan oleh sistem, sehingga dapat menjawab rumusan masalah yang ada. Dalam sistem ini, akan memiliki fungsi sebagai berikut:

a. Membaca formula proposisi dalam bentuk CNF ( *Conjuncvite Normal Form* ).
b. Menyelesaikan permasalahn SAT *Problem*.
c. Memberikan solusi yang berbentuk eksak.

### 3.1.2 Analisis Kebutuhan Input

Merupakan tahapan untuk menentukan tahapan apa saja yang akan dibutuhkan. Masukan tersebut harus membuat sistem dapat menjalankan fungsi untuk dapat menyelesaikan SAT *Problem* tersebut. Kebutuhan masukan dalam penelitian ini sebagai berikut:

a. Masukan yang diberikan adalah sebuah *file* berisikan formula proposisi yang sesuai dengan standar CNF dan dalam bentuk format dimacs. Format dimacs merupakan format yang biasa digunakan untuk menunjukkan formula pada CNF. Contoh file dimacs pada gambar 3.1:

```
c  simple_v3_c2.cnf
c
p cnf 3 2
1 -3 0
2 3 -1 0
```

Gambar 3. 1 Contoh file berbentuk dimacs

Gambar 3.1 menunjukkan p merupakan sebuah problem, cnf merupakan conjunctive normal form, angka 3 merupakan jumlah variable, dan 2 merupakan jumlah klausa.

b. Setiap nilai klausa yang berisi variabel masuk kedalam vektor input.
c. Masukan untuk menentukan jumlah klaster.

### 3.1.3 Analisis Kebutuhan Output

Tahap ini merupakan tahap dilakukan pembuatan suatu sistem yang diharapkan mampu menyelesaikan masukan yang diberikan dan dapat memberikan keluaran. Keluaran tersebut dapat membuktikan sistem telah berhasil menyelesaikan masukan sesuai dengan fungsi yang telah ditentukan. Kebutuhan keluaran dalam penelitian ini adalah sebagai berikut:

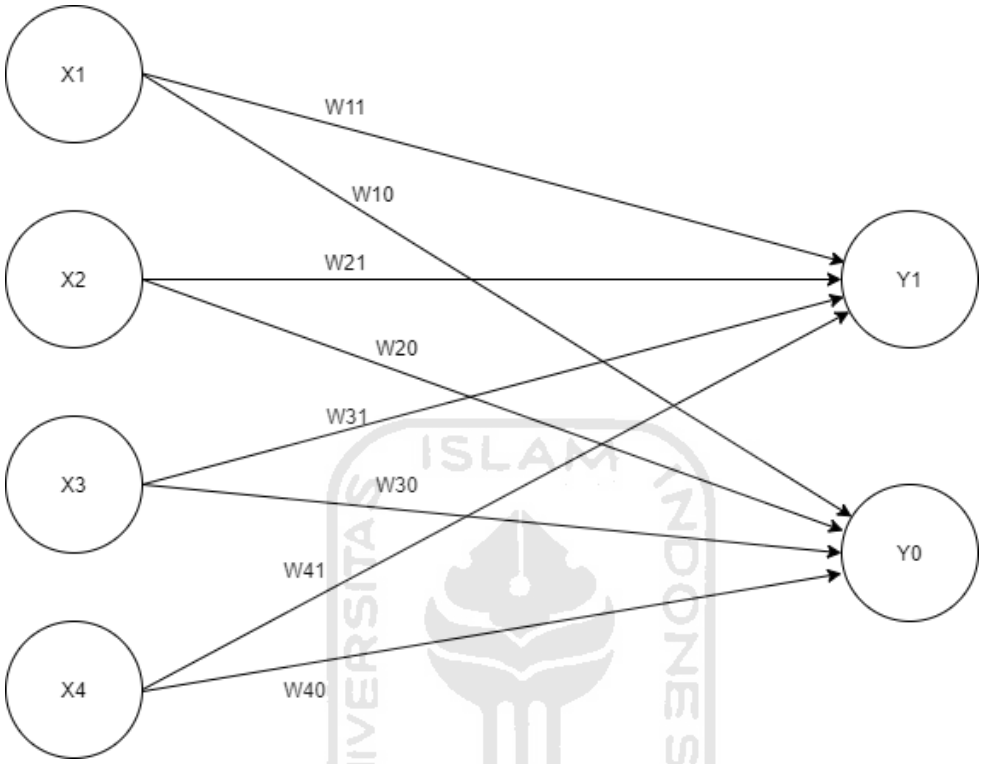
a. Sebuah keluaran string berupa "*satisfiable*" atau "*unsatisfiable*". Keluaran ini hasil dari berapa banyak jumlah satisfiable yang masuk ke klaster 1, atau berapa banyak jumlah unsatisfiable masuk ke klaster 0.

## 3.2 Metode Perancangan

Metode ini adalah metode untuk menerangkan perancangan terhadap sebuah penelitian. Pada tahap ini, merupakan tahapan penting sebelum melanjutkan ke tahap implementasi penelitian. Dalam tahap ini akan memperoleh metode yang dipakai untuk penelitian sehingga dapat memahami, menerapkan serta melakukan evaluasi apa saja yang akan menjadi kendala dalam penelitian ini. Dalam penelitian ini, metode yang digunakan adalah metode *Kohonen Self-Organizing Maps* (K-SOM) dengan menggunakan bahasa pemograman python 3.8.3

### 3.2.1 Perancangan model Metode K-SOM untuk penyelesaian SAT Problem

Pada tahap ini, akan dilakukan perancangan pemodelan K-SOM untuk menyelesaikan SAT *Problem*. Oleh karena itu, akan dijelaskan secara singkat tahapan pemodelan metode K-SOM untuk penyelesaian SAT *Problem* pada gambar 3.2:




Gambar 3. 2 Struktur Neuron Input dan Output

Gambar 3.2 menunjukkan struktur *neuron input* dan *neuron output* pada metode K-SOM. Jumlah *neuron input* merupakan jumlah klausa dalam formula CNF. Sedangkan vektor input, merupakan jumlah variabel pada setiap klausa. Untuk setiap vektor input pada neuron, vektor input yang bernilai 1 merupakan bentuk dari literal posiitf. Sedangkan vektor input yang bernilai -1 merupakan bentuk dari literal negatif. Kemudian pada neuron output adalah untuk menentukan jumlah klaster. Jumlah klaster pada penelitian ini adalah 2, yaitu 1 bernilai *satisfiable* dan 0 yang bernilai *unsatisfiable*. Berikut ini adalah contoh pemodelan metode K-SOM pada SAT Problem:

Diketahui sebuah SAT *Problem* dengan formula CNF memiliki variabel serta klausa, dimana:

$$( p \vee q ) \wedge ( r \vee \neg s) \wedge ( r ) \wedge ( s )$$

1. Inisialisasi Input

   Langkah ini merupakan langkah menentukan jumlah vektor input pada setiap input $W_{ij}$. Semisal nilai yang ditentukan yaitu 4 input:

   Vektor *Input* X1 ( p, q, r, s ) = 1, 1, -1, -1

   Vektor *Input* X2 ( p, q, r, s ) = -1, -1, -1, 1

   Vektor *Input* X3 ( p, q, r, s ) = 1, -1, -1, -1

   Vektor *Input* X4 ( p, q, r, s ) = -1, -1, 1, 1

2. Inisialisasi Neuron Ouput

   SAT *Problem* memiliki 2 kemungkinan, yaitu *satisfiable* atau *unsatisfiable*. oleh karena itu, *neuron output* pada kasus ini terbagi menjadi 2 klaster, yaitu klaster 1 dan klaster 0. Klaster 1 merupakan neuron output yang bernilai *satisfiable*. Sedangkan klaster 0 merupakan *neuron output* yang bernilai *unsatisfiable*.

*3.* Inisialisasi bobot random serta *learning rate*.

   Pada tahap ini, yaitu melakukan inisiasi bobot secara random dengan nilai antara x-min dan x-max secara acak. Kemudian menentukan learning rate α sebesar 0.6

   Bobot Y0 = [0.2, 0.6, 0.5, 0.9]

   Bobot Y1 = [0.8, 0.4, 0.7, 0.3]

4. Setiap vektor input melakukan langkah 5 sampai 7.

5. Untuk setiap j, melakukan perhitungan dibawah ini:

$$D(j) = \sum_i (Wij - Xi)^2 \qquad (3.1)$$

   Kemudian menemukan indeks j yang memiliki nilai D(j) terkecil:

   Vektor *Input* X1 ( p, q, r, s ) = 1, 1, -1, -1

   $D(1) = (0.2 - 1)^2 + (0.6 - 1)^2 + (0.5 - 0)^2 + (0.9 - 0)^2 = 1.86$

   $D(0) = (0.8 - 1)^2 + (0.4 - 1)^2 + (0.7 - 0)^2 + (0.3 - 0)^2 = \mathbf{0.98}$

   Yang menjadi pemenang pada Vektor Input X1 adalah D(0).

   Vektor *Input* X2 ( p, q, r, s ) = -1, -1, -1, 1

   $D(1) = (0.2 - 0)^2 + (0.6 - 0)^2 + (0.5 - 0)^2 + (0.9 - 1)^2 = \mathbf{0.66}$

$D(0) = (0.92 - 1)^2 + (0.76 - 1)^2 + (0.28 - 0)^2 + (0.12 - 0)^2 = 2.28$

Yang menjadi pemenang pada Vektor Input X2 adalah D(1).

Vektor *Input* X3 ( p, q, r, s ) = 1, -1, -1, -1

$D(1) = (0.08 - 1)^2 + (0.24 - 0)^2 + (0.2 - 0)^2 + (0.96 - 0)^2 = 1.87$

$D(0) = (0.92 - 1)^2 + (0.76 - 0)^2 + (0.28 - 0)^2 + (0.12 - 0)^2 = \mathbf{0.68}$

Yang menjadi pemenang pada Vektor Input X3 adalah D(0).

Vektor *Input* X4 ( p, q, r, s ) = -1, -1, 1, 1

$D(1) = (0.08 - 0)^2 + (0.24 - 0)^2 + (0.2 - 1)^2 + (0.96 - 1)^2 = \mathbf{0.71}$

$D(0) = (0.97 - 0)^2 + (0.31 - 0)^2 + (0.12 - 0)^2 + (0.05 - 0)^2 = 2.73$

Yang menjadi pemenang pada Vektor Input X4 adalah D(1).

6. Menemukan indeks *j* yang memiliki nilai $D_{(j)}$ paling minimum/terkecil.

   Vektor *Input* X1 (p, q, r, s ) = 1, 1, -1, -1

   *D(j)* minimum untuk *j* = 0

   Vektor *Input* X2 (p, q, r, s ) = 1, 1, -1, -1

   *D(j)* minimum untuk *j* = 1

   Vektor *Input* X3 (p, q, r, s ) = 1, 1, -1, -1

   *D(j)* minimum untuk *j* = 0

   Vektor *Input* X4 (p, q, r, s ) = 1, 1, -1, -1

   *D(j)* minimum untuk *j* = 1

7. Memperbarui semua bobot tiap unit *j* dengan rumus:

$$W_{ij}(\text{new}) = W_{ij}(\text{old}) + \alpha [\, x_i - w_{ij}(\text{old}) \,] \qquad (3.2)$$

   Vektor *Input* X1 (p, q, r, s ) = 1, 1, -1, -1

   $W = 0.8 + 0.6(1 - 0.8) = 0.92$

   $W = 0.4 + 0.6(1 - 0.4) = 0.76$

$W = 0.7 + 0.6( 0 - 0.7 ) = 0.28$

$W = 0.3 + 0.6( 0 - 0.3 ) = 0.12$

Vektor *Input* X2 (p, q, r, s ) = -1, -1, -1, 1

$W = 0.2 + 0.6( 0 - 0.2 ) = 0.08$

$W = 0.6 + 0.6( 0 - 0.4 ) = 0.24$

$W = 0.5 + 0.6( 0 - 0.5 ) = 0.2$

$W = 0.9 + 0.6( 1 - 0.9 ) = 0.96$

Vektor *Input* X3 (p, q, r, s ) = 1, -1, -1, -1

$W = 0.92 + 0.6( 1 - 0.92 ) = 0.97$

$W = 0.76 + 0.6( 0 - 0.76 ) = 0.31$

$W = 0.28 + 0.6( 0 - 0.28 ) = 0.12$

$W = 0.12 + 0.6( 0 - 0.12 ) = 0.05$

Vektor *Input* X4 (p, q, r, s ) = -1, -1, 1, 1

$W = 0.08 + 0.6( 0 - 0.08 ) = 0.04$

$W = 0.24 + 0.6( 0 - 0.24 ) = 0.09$

$W = 0.20 + 0.6( 1 - 0.20 ) = 0.68$

$W = 0.96 + 0.6( 1 - 0.96 ) = 0.98$

8. Memperbarui *learning rate* dengan dengan mengalikan *learning rate* yang lama dengan 0.5.

$$\alpha( 0 ) = 0.6 \qquad (3.3)$$
$$\alpha( t + 1 ) = 0.5\ \alpha( t )$$
$$\alpha_{(baru)} = 0.5\ ( 0.6 ) = 0.3$$

9. Memperbarui bobot sampai bobot tersebut bersifat konvergen

Berikut tabel 3.1 yang merupakan hasil perhitungan pemodelan metode K-SOM untuk penyelesaian SAT *Problem*:

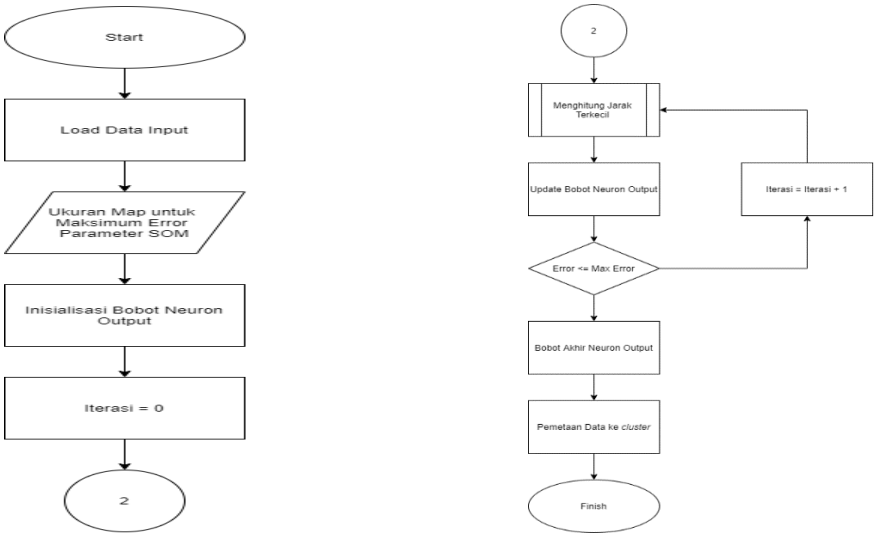
Tabel 3. 1 Hasil perhitungan pemodelan K-SOM untuk SAT Problem

| X1 | X2 | X3 | X4 | KLASTER |
|---|---|---|---|---|
| 1 | 1 | -1 | -1 | 1 |
| -1 | -1 | -1 | 1 | 0 |
| 1 | -1 | -1 | -1 | 1 |
| -1 | -1 | 1 | 1 | 0 |

Dari hasil perhitungan pada tabel 3.1 menunjukkan bahwa input X1 dan X3 masuk kedalam klaster 1, yang berarti bahwa input tersebut masuk dalam kategori *satisfiable*. Sedangkan input X2 dan X4 masuk kedalam klaster 0, yang berati input tersebut masuk kedalam *unsatisfiable*. Perlu diketahui, pemodelan ini hanya menunjukkan tiap input / klausa masuk kedalam klaster yang bernilai *satisfiable* dan *unsatisfible*, dan belum mendapatkan hasil secara kesuluruhan.

### 3.2.2 Flowchart Kohonen Self Organizing Map (K-SOM)

Flowchart merupakan suatu bagan dengan simbol tertentu untuk menggambarkan urutan proses atau biasa disebut diagram alir. Flowchart ini digunakan untuk menerangkan proses-proses yang terjadi dengan metode K-SOM. Berikut merupakan *flowchart* Pada metode perancangan dari *Kohonen Self Organizing Map* (K-SOM) pada gambar 3.3:



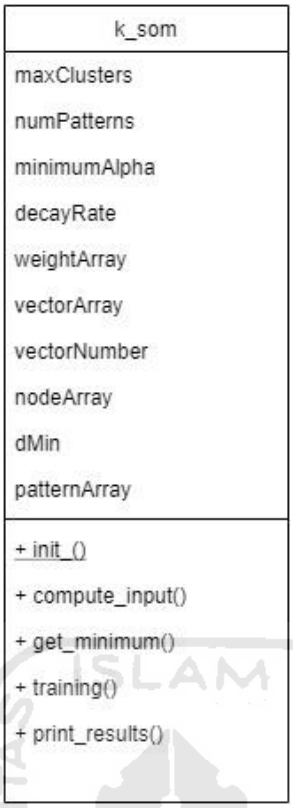

Gambar 3. 3 Flowchart dari Kohonen Self Organizing Map

Berdasarkan gambar 3.3, berikut merupakann proses penguraiannya :

1. Memuat data inputan
2. Menentukan jumlah cluster, max error yang sudah disudah tetapkan serta indikator dari SOM tersebut.
3. Menginisalisasi bobot neuron output secara acak dengan rentang minimum dan maksimum sesuai dengan bobot di data tersebut.
4. Membuat iterasi awal = 0, digunakan sebagai iterasi awal untuk memulai proses *training* data.
5. Hitung jarak paling kecil. Kemudian melakukan perhitungan dari tersebut dengan bobot random sehingga mendapat nilai terkecil. Spesifikasi dari pemenang neuron ini adalah dengan menunjukkan jarang yang paling terkecil. Dan untuk rumus yang digunakan adalah *euclidian distance.*
6. Kemudian setelah mendapat pemenang dari neuron tersebut, kemudian melakukan pengubahan nilai bobot yang tersambung dengan pemenang dari neuron tersebut.
7. Untuk kondisi apakah *error* tersebut saat ini lebih kecil dari error maksimum. Jika error, maka lanjut ke langkah 8. Jika sebaliknya maka kembali ke langkah 5.
8. Untuk bobot akhir neuron yang telah didapatkan nantinya pada masing-masing cluster setelah proses *training*.
9. Kemudian setelah itu, melakukan pemetaan kedalam klaster. Data masukan tersebut dihitung jaraknya. Kemudian output neuron terkecil menjadi klasternya.

### 3.2.3 Diagram kelas

Diagram kelas merupakan diagram yang digunakan untuk menampilkan beberapa kelas serta paket-paket yang ada pada suatu sistem tersebut yang akan dikembangkan. Diagram kelas memberi sebuah gambaran statis tentang sistem yang akan dibuat. Didalam diagram kelas juga terdapat deskripsi dari masing-masing objek berupa metode dan lain-lain. Diagram kelas akan ditunjukan pada gambar 3.4 :




Gambar 3. 4 Diagram kelas

1. def __init__

   Merupakan metode yang sudah direservasi python sebagai metode yang dipanggil saat akan membuat sebuah fungsi. Disetiap metode class selalu harus ada self, sebagai parameter pertamanya. Variabel dari self tersebut merujuk objek dari class SOM_Class2 ini. Dari metode ini terdiri dari beberapa atribut pada tabel 3.2 sebagai berikut:

Tabel 3. 2 Atribut yang ada pada def __init__

| **Atribut** | **Tipe** | **Keterangan** |
|---|---|---|
| vectorLength | Int | Sebuah Panjang dari vektor input. |
| maxClusters | Int | Menentukan berapa banyak klaster tersebut. |
| NumPaterns | Int | Menentukan berapa nilai input dari data tersebut |
| minimumAlpha | float | Minimum nilai dari alpha. |
| decayRate | float | Nilai yang untuk menentukan berapa learning rate |
| weightArray | Double | Menentukan nilai bobot random dari berapa banyak input yang masuk. |

2. def compute_input

   Merupakan fungsi dari kelas SOM_Class2 yang berfungsi untuk melakukan perhitungan jarak antara bobot $w_{ij}$ dengan vektor *input* $x_i$ untuk setiap klaster. Fungsi ini terdari dari beberapa atribut pada tabel 3.3 sebagai berikut :

Tabel 3. 3 Atribut dari def compute_input

| **Atribut** | **Tipe** | **Keterangan** |
|---|---|---|
| vectorArray | Array | Merupakan kumpulan beberapa klausa |
| vectorNumber | Int | Merupakan indeks klausa ke berapa |
| vectorInput | | Merupakan atribut untuk menyimpan nilai dari vector array dan vectorNumber |

3. def get_minimum

   Merupakan fungsi dari kelas SOM_Class2 yang berfungsi untuk mencari nilai *minimum*/terkecil dari hasil perhitungan jarak euclidean dan akan dijadikan sebagai pemenang neuron hasil dari perhitungan pada tabel 3.2. def get_minimum ini atribut pada tabel 3.4 :

Tabel 3. 4 Atribut dari def get_minimum

| **Atribut** | **Tipe** | **Keterangan** |
|---|---|---|
| nodeArray | arraydouble | Merupakan kumpulan dari hasil perhitungan jarak euclidean |

4. def update_weights

   Merupakan fungsi dari kelas SOM_Class2 yang berfungsi untuk *mengupdate*/memperbaharui dari pemenang neuron yang memiliki nilai *minimum*/terkecil. Berikut merupakan atribut dari def update_weights pada tabel 3.5 :

Tabel 3. 5 Atribut pada fungsi def update_weights

| Atribut | Tipe | Keterangan |
|---|---|---|
| vectorNumber | Int | Merupakan nilai variabel vektor dari tiap input |
| dMin | - | Merupakan nilai yang minimum dari hasil perhitungan jarak euclidean |
| patternArray | Array 2 Dimensi | Merupakan kumpulan dari beberapa klausa dari input klausa |

5. def training

   Merupakan fungsi dari kelas SOM_Class2 yang berfungsi untuk melakukan iterasi yang ada pada algoritma Self-Organizing Maps (K-SOM). Berikut merupakan atribut dari def training pada tabel 3.6 :

Tabel 3. 6 Atribut dari def training

| Atribut | Tipe | Keterangan |
|---|---|---|
| PatternArray | Array 2 Dimensi | Merupakankumpulan beberapa klausa dari input klausa |

6. def print_results

   Merupakan fungsi dari kelas SOM_Class2 yang berfungsi untuk memunculkan hasil akhir dari komputasi yang sudah dilewati. Atribut dari def print_results sendiri adalah pada tabel 3.7 :

Tabel 3. 7 Atribut dari def print_results

| Atribut | Tipe | Keterangan |
|---|---|---|
| PatternArray | Array 2 Dimensi | Merupakan Kumpulan beberapa klausa dari input klausa |

### 3.2.4 Perancangan Masukan / Input

Dalam melakukan tahapan perancangan masukan, dibentuk beberapa masukan sesuai dengan analisis masukan yaitu formula , *maxCluster,* vector, serta input_patterns. Adapun masukan dari software perangkat lunak akan dijelaskan sebagai berikut:

1. Formula

   Perancangan masukan ini memuat logika proposisi dengan standar CNF dan dalam bentuk format dimacs. Standar dari CNF tersebut terdiri atas komentar, variabel yang berbentuk integer serta parameter. Pada baris awal sebuah masukin diawali dengan sebuah karakter ‘c’ yang berarti adalah sebuah komentar yang tidak perlu dibaca oleh sebuah program. Komentar tersebut berisi identitas dari pembuat *file* tersebut ataupun keterangan tambahan yang diperlukan. Kemudian untuk baris selanjutnya adalah diawali dengan huruf ‘p’ yang berarti adalah parameter seberapa banyak variabel serta klausa yang ada pada *file* masukan tersebut. Variabel yang sudah diberikan pada masukan itu berformat CNF dan di ekspresikan dalam bentuk integer. Kemudian konjungsi antar klausa di ekspresikan dengan angka 0. Sedangkan untuk disjungsi antar literal di ekspresika dengan spasi. Variabel yang memiliki nilai negasi ditandai dengan ditambah simbol negatif(-). Contoh file CNF dapat dilihat pada Lampiran A dan contoh untuk konversi masukan logika proposisi menjadi format CNF bisa dilihat pada Tabel 3.8 :

Tabel 3. 8 Mengkonversikan logika proposisi ke CNF

| Keterangan | Konversi ke CNF |
|---|---|
| Logika Proposisi | (e v ¬f) ^ (¬h v e) ^ (f v g) |
| Setiap Variabel diubah menjadi Integer | (1 v 2) ^ (¬4 V 1) ^ (2 v 3) |
| Simbol negasi diubah menjadi negatif | (1 v 2) ^ (-4 v 1) ^ (2 v 3) |
| Simbol disjungsi diubah menjadi spasi | (1 2) ^ (-4 1) ^ (2 3) |
| Simbol konjungsi diubah menjadi 0 | (1 2) 0 (-4 1) 0 (2 3) |
| Tanda kurung dihapus | 1 2 0 -4 1 0 2 3 |
| Setiap klausa dipisah dengan baris baru | 1 2 0<br>-4 1 0<br>2 3 |

2. Input _patterns

   Menghitung banyaknya jumlah klausa yang masuk.

3. MaxClusters

   Merupakan masukan untuk menentukan jumlah klaster.

4. Decay_Rate

   Merupakan masukan untuk *reduce learning rate*.

5. Min_alpha

   Menentukan minimum alpha yang akan digunakan untuk melakukan iterasi.

Adapun contoh rancangan masukan pada gambar 3.5 :

```
Input:
MAX_CLUSTER: 2
DECAY_RATE = 0.96
MIN_ALPHA = 0.01

f1 = CNF (from_file='16-18.cnf')
INPUT_PATTERNS = len(f1.clauses)

p cnf 16 18
  1    2  0
 -2   -4  0
  3    4  0
 -4   -5  0
  5   -6  0
  6   -7  0
  6    7  0
  7  -16  0
  8   -9  0
 -8  -14  0
```

```
  9   10   0
  9  -10   0
-10  -11   0
 10   12   0
 11   12   0
 13   14   0
 14  -15   0
 15   16   0
```

Gambar 3. 5 Perancangan Masukan.

### 3.2.5 Perancangan Keluaran / Output

Untuk perancangan keluaran pada *software*/perangkat lunak ini menghasilkan 1 keluaran yaitu SAT problem yang solusinya ditentukan dengan berapa banyak klausa yang masuk kedalam klaster *satisfiable* atau *unsatisfiable*. Tiap perhitungan klausa masuk antara klaster 1 yang merupakan *satisfied* atau klaster 0 yang merupakan *unsatisfied*. Kemudian menjumlah kan berapa banyak input perbandingan dari 2 klaster tersebut. Kemudian yang paling banyak akan menjadi kesimpulan dari klaster tersebut, apakah *satisfiabel* atau *unsatisfiabel* tersebut. Berikut merupakan gambaran perancangan keluaran yang ditunjukan pada gambar dibawah ini terbagi menjadi 2, yaitu perancangan keluaran satisfiable pada gambar 3.6 dan perancangan keluaran unsatisfiael pada gambar 3.7:

```
Input:
MAX_CLUSTER: 2

f1 = CNF (from_file='16-18.cnf')

INPUT_PATTERNS = len(f1.clauses)

p cnf 16 18
  1    2  0
 -2   -4  0
  3    4  0
 -4   -5  0
  5   -6  0
  6   -7  0
  6    7  0
  7  -16  0
  8   -9  0
 -8  -14  0
  9   10  0
  9  -10  0
-10  -11  0
 10   12  0
 11   12  0
 13   14  0
 14  -15  0
 15   16  0
Satisfied = 11
Unsatisfied = 7
```

```
Conclusion = SATISFIABEL
```

Gambar 3. 6 Contoh perancangan keluaran yang bersifat *Satisfiabel*.

Gambar pada 3.6 menunjukkan apabila dari perhitungan diatas bahwa klaster pada *satisfied* berjumlah 11, sedangkan klaster pada *unsatisfied* berjumlah 7. Oleh karena itu *conclusion* / kesimpulan pada data diatas adalah *satisfiabel*.

```
Input:
MAX_CLUSTER: 2

f1 = CNF (from_file='100-160.cnf')

INPUT_PATTERNS = len(f1.clauses)

p cnf 100 160
16 30 95 0
-16 30 95 0
-30 35 78 0
-30 -78 85 0
-78 -85 95 0
8 55 100 0
8 55 -95 0
9 52 100 0
9 73 -100 0
-8 -9 52 0
38 66 83 0
-38 83 87 0
-52 83 -87 0
66 74 -83 0
-52 -66 89 0
-52 73 -89 0
-52 73 -74 0
-8 -73 -95 0
40 -55 90 0
-40 -55 90 0
25 35 82 0
-25 82 -90 0
-55 -82 -90 0
11 75 84 0
11 -75 96 0
23 -75 -96 0
-11 23 -35 0
-23 29 65 0
29 -35 -65 0
-23 -29 84 0
-35 54 70 0
-54 70 77 0
19 -77 -84 0
-19 -54 70 0
22 68 81 0
-22 48 81 0
-22 -48 93 0
3 -48 -93 0
7 18 -81 0
```

```
-7 56 -81 0
3 18 -56 0
-18 47 68 0
-18 -47 -81 0
-3 68 77 0
-3 -77 -84 0
19 -68 -70 0
-19 -68 74 0
-68 -70 -74 0
54 61 -62 0
50 53 -62 0
-50 61 -62 0
-27 56 93 0
4 14 76 0
4 -76 96 0
-4 14 80 0
-14 -68 80 0
-10 -39 -89 0
1 49 -81 0
1 26 -49 0
17 -26 -49 0
-1 17 -40 0
16 51 -89 0
-9 57 60 0
12 45 -51 0
2 12 69 0
2 -12 40 0
-12 -51 69 0
-33 60 -98 0
5 -32 -66 0
2 -47 -100 0
-42 64 83 0
20 -42 -64 0
20 -48 98 0
-20 50 98 0
-32 -50 98 0
-24 37 -73 0
-24 -37 -100 0
-57 71 81 0
-37 40 -91 0
31 42 81 0
-31 42 72 0
-31 42 -72 0
7 -19 25 0
-1 -25 -94 0
-15 -44 79 0
-6 31 46 0
-39 41 88 0
28 -39 43 0
28 -43 -88 0
-4 -28 -88 0
-30 -39 -41 0
-29 33 88 0
-16 21 94 0
-10 26 62 0
-11 -64 86 0
-6 -41 76 0
38 -46 93 0
26 -37 94 0
-26 53 -79 0
```

```
78 87 -94 0
65 76 -87 0
23 51 -62 0
-11 -36 57 0
41 59 -65 0
-56 72 -91 0
13 -20 -46 0
-13 15 79 0
-17 47 -60 0
-13 -44 99 0
-7 -38 67 0
37 -49 62 0
-14 -17 -79 0
-13 -15 -22 0
32 -33 -34 0
24 45 48 0
21 24 -48 0
-36 64 -85 0
10 -61 67 0
-5 44 59 0
-80 -85 -99 0
6 37 -97 0
-21 -34 64 0
-5 44 46 0
58 -76 97 0
-21 -36 75 0
-15 58 -59 0
-58 -76 -99 0
-2 15 33 0
-26 34 -57 0
-18 -82 -92 0
27 -80 -97 0
6 32 63 0
-34 -86 92 0
13 -61 97 0
-28 43 -98 0
5 39 -86 0
39 -45 92 0
27 -43 97 0
13 -58 -86 0
-28 -67 -93 0
-69 85 99 0
42 71 -72 0
10 -27 -63 0
-59 63 -83 0
36 86 -96 0
-2 36 75 0
-59 -71 89 0
36 -67 91 0
36 -60 63 0
-63 91 -93 0
25 87 92 0
-21 49 -71 0
-2 10 22 0
6 -18 41 0
6 71 -92 0
-53 -69 -71 0
-2 -53 -58 0
43 -45 -96 0
34 -45 -69 0
```

```
63 -86 -98 0

Satisfied = 79
Unsatisfied = 81

Conclusion = UNSATISFIABEL
```

Gambar 3. 7 Contoh perancangan keluaran yang bersifat Unsatisfiable.

Gambar pada 3.7 menunjukkan apabila dari perhitungan diatas bahwa klaster pada *satisfied* berjumlah 79, sedangkan klaster pada *unsatisfied* berjumlah 81. Oleh karena itu *conclusion* / kesimpulan pada data diatas adalah *unsatisfiable.*

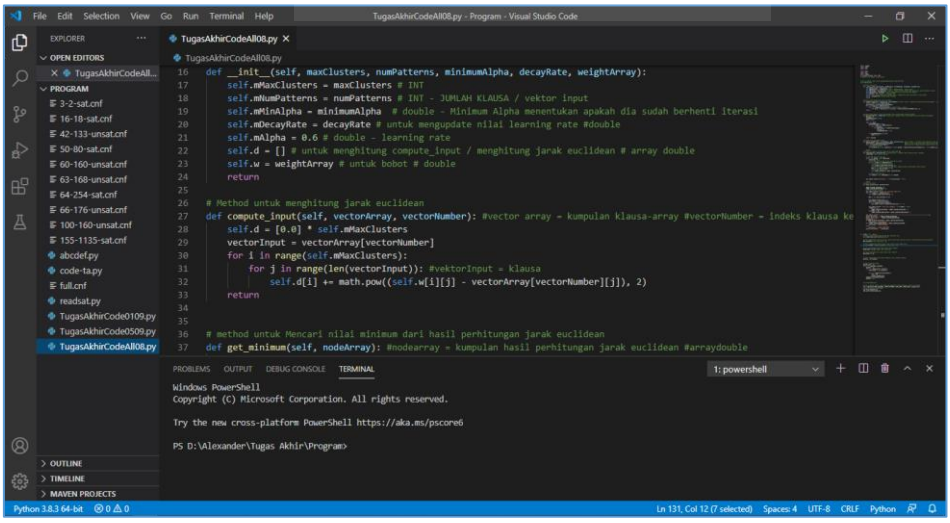
# BAB IV
# IMPLEMENTASI DAN PENGUJIAN

## 4.1 Implementasi

Implementasi merupakan tahapan selanjutnya untuk mengerjakan apa yang sudah dirancang pada tahapan di bab sebelumnya. Implementasi akan menunjukkan apakah sebuah rancangan yang telah dilakukan sebelumnya dapat berjalan serta digunakan dengan baik. Pada tahap ini akna dibuat SAT *Solver* yang mempresentasikan implementasi SAT *Problem* dengan nilai eksak menggunakan metode *Kohonen Self-Organizing Map* (K-SOM).

### 4.1.1 Microsoft Visual Studio Code IDE

Microsoft Visual Studio Code merupakan sebuah proyek yang dibuat oleh microsoft secara gratis untuk sistem operasi seperti windows, linux dll pada tahun 2015. Pada tahun 2016, visual studio code lulus tahap pratinjau publik dan akhirnya dirilis ke *web*. Visual studio code merupakan produk yang digunakan untuk melakukan pemograman, baik menulis kode, kompilasi, debug serta mendistribusika program. Visual studio code ini banyak mendukung sejumlah bahasa pemograman serta serangkaian fitur yang berbeda per bahasa. Salah satu nya adalah python. Teks editor ini juga bersifat *open source* yang mana kode sumbernya dapat dilihat serta bisa berkontribusi untuk pengembangannya. Tampilan untuk Microsoft Visual Studio Code dapat dilihat pada Gambar 4.1




Gambar 4. 1 Tampilan Gambar Pada Microsoft Visual Studio Code

### 4.1.2 Class K-SOM

Pada tahapan ini akan dilakukan pembuatan *class* untuk metode Kohonen Self-Organizing Maps (K-SOM). Class ini diberi dengan nama k_som. Class ini terdiri dari beberapa method diantaranya sebagai di lihat pada beberapa gambar berikut :

```
class k_som:


    def __init__(self, maxClusters, numPatterns, minimumAlpha, decayRate,
weightArray):
        self.mMaxClusters = maxClusters
        self.mNumPatterns = numPatterns
        self.mMinAlpha = minimumAlpha
        self.mDecayRate = decayRate
        self.mAlpha = 0.6
        self.d = []
        self.w = weightArray
        return
```

Gambar 4. 2 Method def__init__

Pada Gambar 4.2 ini merupakan method constructor dari class k-som. Method constructor merupakan method khusus yang dijalankan secara otomatis untuk menginisiaslisasi pembuatan objek dari kelas k-som. Metode ini terdiri dari beberapa atribut diantaranya maxCluster, numPatterns, minimumAlpha, decayRate, weightArray.

```
def compute_input(self, vectorArray, vectorNumber):
        self.d = [0.0] * self.mMaxClusters
        vectorInput = vectorArray[vectorNumber]
        for i in range(self.mMaxClusters):
            for j in range(len(vectorInput)):
                self.d[i]        +=        math.pow((self.w[i][j]        -
vectorArray[vectorNumber][j]), 2)
        return
```

Gambar 4. 3 Method def compute_input

Pada Gambar 4.3 merupakan method dari perhitungan euclidean.

```
def get_minimum(self, nodeArray):
        minimum = 0
        foundNewMinimum = False
        done = False
```

Gambar 4. 4 Method def get_minimum

Gambar 4.4 merupakan method yang befungsi untuk mencari nilai minimum hasil dari perhitungan euclidean.

```
def update_weights(self, vectorNumber, dMin, patternArray):
        vectorInput = patternArray[vectorNumber]
        for i in range(len(vectorInput)):

            self.w[dMin][i]    =    self.w[dMin][i]    +    (self.mAlpha    *
(patternArray[vectorNumber][i] - self.w[dMin][i]))
```

Gambar 4. 5 Method def update_weights

Gambar 4. 5 adalah method untuk mengupdate bobot hasil mencari nilai minium dari perhitungan euclidean.

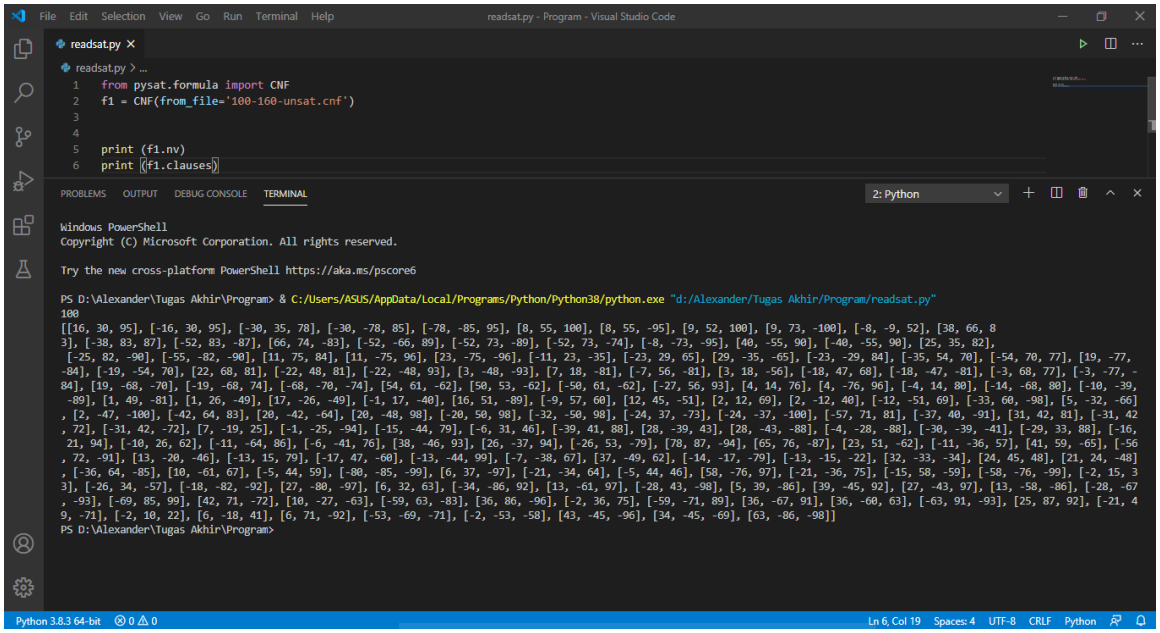
```
def training(self, patternArray):
        iterations = 0 # counter iterasi


        while self.mAlpha > self.mMinAlpha:
            iterations += 1

            for i in range(self.mNumPatterns):

                self.compute_input(patternArray, i)

                dMin = self.get_minimum(self.d)

                self.update_weights(i, dMin, patternArray)


            self.mAlpha = self.mDecayRate * self.mAlpha
```

Gambar 4. 6 Method def training

Gambar 4.6 merupakan method untuk memulai iterasi yang ada pada algoritma SOM

### 4.1.3 Read SAT data

Pada tahap ini, akan membaca data yang akan di input. File ini berbentuk dimacs sesuai dengan data yang ada pada sat competition. Dalam bab perancangan pemodelan K-SOM untuk SAT problem, nilai input variable harus di ubah menjadi 1 untuk literal posiif dan -1 untuk literal negatif. Namun dalam implementasi menggunakan bahasa pemograman python, ada package untuk membaca data SAT berbentuk dimacs, yaitu dengan menginstall pacakge *python-sat*. Berikut merupakan gambar untuk membaca data SAT:




Read SAT data

### 4.1.4 Inisiaslisasi Klaster, Decay_Rate, Minimum Alpha

Pada pembahasan ini akan melakukan penentuan jumlah klaster. Pada penelitian saat ini jumlah klaster yang ditentukan ada 2, yaitu 1 dan 0. 1 bernilai satisfiable dan 0 bernilai unsatisfiable. Kemudian menentukan decay_rate. Decay_rate merupakan nilai untuk melakukan *learning rate*. Untuk minimum alpha adalah nilai awal pada alpha. Berikut adalah inisiasi pada Gambar 4.7 :

```
MAX_CLUSTERS = 2

INPUT_PATTERNS = len(f1.clauses)

DECAY_RATE = 0.96

MIN_ALPHA = 0.01
```

Gambar 4.7 Inisiasi Klaster, decay_rate, dan minimum_alpha

Pada gambar 4.7 ini, atribut-atribut pada gambar tersebut sudah di inisiasi sebelumnya yang masuk pada method def __init__ yang merupakan method constructor.

### 4.1.5 Input data SAT

Pada tahapan ini, data SAT yang sudah terbaca, dimasukkan kedalam array 2 dimensi. Data yang diambil merupakan klausa dari input data SAT. pada gambar ini, atribut ini sudah di inisiasi kedalam method def __init__ yang merupakan method constructor. Berikut merupakan gambar pada 4.8:

```
if __name__ == '__main__':

 pattern = f1.clauses
```

Gambar 4. 8 Input data SAT

Pada gambar 4.8, F1 merupakan variabel yang membaca data SAT. Sedangkan *clauses* merupakan fungsi dari package python-sat yang digunakan untuk mengambil klausa dari data input tersebut.

### 4.1.6 Inisiasilisasi Bobot Random

Pada tahap ini dilakukan inisiasi bobot neuron secara random untuk melakukan perhitungan euclidean. Berikut gambar 4.9:

```
    weightsList = []
    for i in range(MAX_CLUSTERS):
        weights = []
        for v in range(len(f1.clauses)):
            vectorInput = f1.clauses[v]

            for j in range(len(vectorInput)):
                num = round(random.randint(10,90))/100
                weights.append(num)
        weightsList.append(weights)
        weights = []
```

Gambar 4. 9 Inisiasi bobot random

Pada gambar 4.9, weightlist merupakan variabel untuk menyimpan bobot random. Bobot random ini akan digunakan untuk perhitungan *euclidean*.

### 4.1.7 Object Class som

Pada tahapan ini, akan dilakukann proses perhitungan algoritma *Kohonen Self-Organizing Map* dari class k_som. Nantinya akan dibuat sebuah object bernama som untuk memanggil

class tersebut. Dari variabel ini, ada beberapa atribut yang akan dipanggil untuk melakukan perhitungan. Berikut merupakan gambar 4.10:

```
if __name__ == '__main__':

som =k_som(MAX_CLUSTERS, INPUT_PATTERNS, MIN_ALPHA, DECAY_RATE, weightsList)
```

Gambar 4. 10 Object Class som

### 4.1.8 Perhitungan SOM

Pada tahapan ini, akan memulai perhitungan dari algoritma K-SOM. Pada perhitungan ini akan dilakukan perhitungan dari algoritma ini pada tahap ini akan dilakukan perhitungan jarak setiap input yang ada dengan melakukan perhitungan *euclidean*. Nilai input disini merupakan nilai variabel dari *neuron* input. Neuron input merupakan nilai dari tiap klausa. Setelah melakukan perhitungan, didapatkan nilai pemenang neuron berdasarkan nilai terkecil. Setelah mendapatkan nilai terkecil dari perhitungan tersebut, kemudian langkah selanjutnya mengupdate bobot nilai dari pemenang neuron tersebut. kemudian menghitung learning rate. Learning rate. Apabila alpha lebih kecil dari minAlpha, maka akan terus dilakukan iterasi. Berikut merupakan gambar 4.11:

```
if __name__ == '__main__':

som.training(pattern)
som.print_results(pattern)

. . . . . .
        while self.mAlpha >
            iterations += 1

            for i in range(self.mNumPatterns):

                self.compute_input(patternArray, i)

                dMin = self.get_minimum(self.d)

                self.update_weights(i, dMin, patternArray)


            self.mAlpha = self.mDecayRate * self.mAlpha
```

Gambar 4. 11 Perhitungan

## 4.2 Pengujian

Didalam penelitian ini, dilakukan sebuah pengujian terhadap SAT solver yang telah dibuat. Pengujian ini menyelesaikan beberapa masalah SAT. Bentuk file dari SAT Problem sendiri adalah file CNF dengan format dimacs. Pengujian dilakukan dengan menggunakan Laptop dengan berkapasitas i5-6200 @2,4Ghz serta RAM sebesar 8 GB. Aturan penulisan sesuai dengan standar sat competition (http://satcompetition.org).

Masalah yang akan diujikan merupakan sample file yang merupakan CNF file yang digunakan untuk menilai SAT Solver dengan resource yang relatif lumayan besar. File ini diambil dari situs web (http://people.sc.fsu.edu/~jbukardt/data/cnf/cnf.html) dan dari situs web (http://satcompetition.org). Pengujian ini dilakukan secara kuantitatif yang terdiri dari jumlah *Satisfiable*, jumlah *Unsatisfiable*, kesimpulan (conclusion), serta kecepatan komputasi pada setiap data yang diujikan. Parameter keberhasilan dari pengujian ini adalah dapat menghitung seberapa banyak *Satisfiable* dan *Unsatisfiable*. Sedangkan parameter kegagalan dari pengujian ini adalah apabila pada setiap data SAT Problem ini memiliki nilai *Satisfiable* dan *Unsatisfiable* yang sama kuat, oleh karena itu hasil dari pengujian dianggap *Unknown* (tidak diketahui).Sample file ini dapat dilihat pada tabel 4.1 :

Tabel 4. 1 CNF files dalam bentuk dimacs

| No | Nama | Variable | Klausa | Nilai Formula |
|---|---|---|---|---|
| 1 | c simple_v3_c2.cnf | 3 | 2 | Satisfiable |
| 2 | c quinn.cnf | 16 | 18 | Satisfiable |
| 3 | hole6.cnf | 42 | 133 | Unsatisfiable |
| 4 | aim-50-1_6-yes1-4.cnf | 50 | 80 | Satisfiable |
| 5 | dubois20.cnf | 60 | 160 | Unsatisfiable |
| 6 | dubois21.cnf | 63 | 168 | Unsatisfiable |
| 7 | par8-1-c.cnf | 64 | 254 | Satisfiable |
| 8 | dubois22.cnf | 66 | 176 | Unsatisfiable |
| 9 | aim-100-1_6-no-1.cnf | 100 | 160 | Unsatisfiable |
| 10 | The zebra problem. | 155 | 1135 | Satisifiable |

| | | | | |
|---|---|---|---|---|
| 11 | Unif-k3-A SAT07 Contest shuffled benchmark 450-1912 | 450 | 1912 | Unsatisfiable |
| 12 | Unit-k3-A SAT07 Contest shuffled benchmark 360-1533 | 360 | 1533 | Unsatisfiable |
| 13 | Unit-k3- A SAT07 Contest shuffled benchmark 400-1704 | 400 | 1704 | Satisfiable |
| 14 | Unit-k3- A SAT07 Contest shuffled benchmark 500-2130 | 500 | 2130 | Satisfiable |
| 15 | Unit-k3- A SAT07 Contest shuffled benchmark 550-2343 | 550 | 2343 | Satisfiable |
| 16 | Unit-k3- A SAT07 Contest shuffled benchmark 600-2556 | 600 | 2556 | Satisfiable |
| 17 | Unit-k3- A SAT07 Contest shuffled benchmark 650-2769 | 650 | 2769 | Unsatisfiable |
| 18 | Unit-k5- A SAT07 Contest shuffled benchmark 70-1491 | 70 | 1491 | Satisfiable |
| 19 | Unit-k5- A SAT07 Contest shuffled benchmark 80-1704 | 80 | 1704 | Unsatisfiable |
| 20 | Unit-k5- A SAT07 Contest shuffled benchmark 90-1917 | 90 | 1917 | Satisfiable |
| 21 | Unit-k5- A SAT07 Contest shuffled benchmark 100-2130 | 100 | 2130 | Unsatisfiable |
| 22 | Unit-k5- A SAT07 Contest shuffled benchmark 110-2343 | 110 | 2343 | Satisfiable |
| 23 | Unit-k5- A SAT07 Contest shuffled benchmark 120-2556 | 120 | 2556 | Unsatisfiable |
| 24 | Unit-k5- A SAT07 Contest shuffled benchmark 130-2769 | 130 | 2769 | Satisfiable |
| 25 | Unit-k7- A SAT07 Contest shuffled benchmark 45-4005 | 45 | 4005 | Unsatisfiable |

| | | | | |
|---|---|---|---|---|
| 26 | Unit-k7- A SAT07 Contest shuffled benchmark 50-4450 | 50 | 4450 | Satisfiable |
| 27 | Unit-k7- A SAT07 Contest shuffled benchmark 55-4895 | 55 | 4895 | Satisfiable |
| 28 | Unit-k7- A SAT07 Contest shuffled benchmark 60-5340 | 60 | 5340 | Satisfiable |
| 29 | Unit-k7- A SAT07 Contest shuffled benchmark 65-5785 | 65 | 5785 | Unsatisfiable |
| 30 | Unit-k7- A SAT07 Contest shuffled benchmark 70-6230 | 70 | 6230 | Satisfiable |

Di lakukan percobaan dengan 30 sample cnf file berbentuk dimacs. Dalam perhitungan penelitian ini, terdapat 5 file yang tidak dapat disimpulkan karena memiliki nilai *satisfiable* dan *unsatisfiable* yang sama kuat, sehingga tidak dapat disimpulkan data tersebut bernilai *satisfiable* atau *unsatisfiable*. Oleh karena itu, diberi keterangan *Unknown* ( tidak diketahui ). Namun, sisa dari data tersebut yang berjumlah 25 berhasil diuji dengan metode *Kohonen-Self Organizing Maps* (K-SOM) dengan bernilai eksak. Kemudian dari hasil pengujian ini, rata-rata file yang sedang di uji memiliki nilai klausa dan variabel yang lumayan besar dan dapat di komputasikan dalam waktu yang cukup singkat, diantaranya adalah cnf file yang memiliki variabel 70 dan memiliki klausa sebesar 6230 bisa dikerjakan dalam waktu hanya dengan kurang dari 15 detik.

Tabel 4. 2 Hasil Pengujian

| No | Name | Satisfiable | Unsatisfiable | Conclusion | Time |
|---|---|---|---|---|---|
| 1 | C simple_v3_c2.cnf | 1 | 1 | Unknown | 00.89 detik |
| 2 | c quinn.cnf | 11 | 7 | Satisfiable | 01.17 detik |
| 3 | hole6.cnf | 6 | 127 | Unsatisfiable | 01.90 detik |
| 4 | aim-50-1_6-yes1-4.cnf | 46 | 34 | Satisfiable | 01.20 detik |
| 5 | dubois20.cnf | 80 | 80 | Unknown | 01.30 detik |
| 6 | dubois21.cnf | 84 | 84 | Unknown | 01.93 detik |
| 7 | par8-1-c.cnf | 141 | 115 | Satisfiable | 01.14 detik |
| 8 | dubois22.cnf | 88 | 88 | Unknown | 02.03 detik |

| | | | | | |
|---|---|---|---|---|---|
| 9 | aim-100-1_6-no-1.cnf | 80 | 80 | Unknown | 02.05 detik |
| 10 | The zebra problem. | 640 | 520 | Satisfiable | 03.25 detik |
| 11 | Unif-k3-A SAT07 Contest shuffled benchmark 450-1912 | 945 | 967 | Unsatisfiable | 04.55 detik |
| 12 | Unit-k3-A SAT07 Contest shuffled benchmark 360-1533 | 766 | 777 | Unsatisfiable | 03.93 detik |
| 13 | Unit-k3- A SAT07 Contest shuffled benchmark 400-1704 | 872 | 832 | Satisfiable | 04.32 detik |
| 14 | Unit-k3- A SAT07 Contest shuffled benchmark 500-2130 | 1093 | 1037 | Satisfiable | 04.62 detik |
| 15 | Unit-k3- A SAT07 Contest shuffled benchmark 550-2343 | 1164 | 1179 | Satisfiable | 04.94 detik |
| 16 | Unit-k3- A SAT07 Contest shuffled benchmark 600-2556 | 1326 | 1230 | Satisfiable | 05.30 detik |
| 17 | Unit-k3- A SAT07 Contest shuffled benchmark 650-2769 | 1361 | 1408 | Unsatisfiable | 05.68 detik |
| 18 | Unit-k5- A SAT07 Contest shuffled benchmark 70-1491 | 756 | 735 | Satisfiable | 04.79 detik |
| 19 | Unit-k5- A SAT07 Contest shuffled benchmark 80-1704 | 844 | 860 | Unsatisfiable | 04.62 detik |
| 20 | Unit-k5- A SAT07 Contest shuffled benchmark 90-1917 | 963 | 954 | Satisfiable | 04.99 detik |
| 21 | Unit-k5- A SAT07 Contest shuffled benchmark 100-2130 | 1004 | 1126 | Unsatisfiable | 05.49 detik |
| 22 | Unit-k5- A SAT07 Contest shuffled benchmark 110-2343 | 1174 | 1169 | Satisfiable | 05.90 detik |
| 23 | Unit-k5- A SAT07 Contest shuffled benchmark 120-2556 | 1202 | 1354 | Unsatisfiable | 06.22 detik |
| 24 | Unit-k5- A SAT07 Contest shuffled benchmark 130-2769 | 1415 | 1354 | Satisfiable | 06.44 detik |
| 25 | Unit-k7- A SAT07 Contest shuffled benchmark 45-4005 | 1961 | 2044 | Unsatisfiable | 09.90 detik |
| 26 | Unit-k7- A SAT07 Contest shuffled benchmark 50-4450 | 2264 | 2186 | Satisfiable | 10.41 detik |

| | | | | | |
|---|---|---|---|---|---|
| 27 | Unit-k7- A SAT07 Contest shuffled benchmark 55-4895 | 2511 | 2384 | Satisfiable | 11.47 detik |
| 28 | Unit-k7- A SAT07 Contest shuffled benchmark 60-5340 | 2732 | 2608 | Satisfiable | 12.67 detik |
| 29 | Unit-k7- A SAT07 Contest shuffled benchmark 65-5785 | 2821 | 2964 | Unsatisfiable | 13.14 detik |
| 30 | Unit-k7- A SAT07 Contest shuffled benchmark 70-6230 | 3122 | 3108 | Satisfiable | 13.75 detik |

Secara umum, kinerja SAT Solver menggunakan metode K-SOM ini sudah cukup baik, karena dapat menyelesaikan SAT Problem dengan akurasi sebesar 83% dari jumlah data yang sudah diujikan. Dan rata-rata file yang diujikan memiliki lebih dari 1000 klausa dan mampu menjalankan komputasi dalam waktu yang relatif singkat, yaitu kurang dari 15 detik.

# BAB V
# KESIMPULAN

## 5.1 Kesimpulan

Berdasarkan rancangan serta pengujian yang sudah dilakukan, maka kesimpulan yang dapat diambil dalam penelitian ini adalah sebagai berikut:

1. SAT *Solver* yang dibuat untuk mampu mengimpelementasikan metode K-SOM dengan solusi eksak untuk menyelesaikan SAT *Problem*. Hal ini dapat dibuktikan dengan SAT *Solver* yang mampu menyelesaikan SAT Problem yang diujikan pada tahap pengujian
2. SAT solver menggunakan metode Kohonen Self-Organizing Maps ini dapat menghasilkan akurasi sebesar 83 % dari total data uji sebanyak 30 file berbentuk dimacs dan mampu melakukan komputasi dalam waktu yang singkat, waktu kurang dari 15 detik.

## 5.2 Saran

Saran untuk perbaikan serta pengembangan dari penelitian ini sebagai berikut:

1. Melakukan pengembangan lebih lanjut terhadap SAT Solver dengan cara memodifikasi metode K-SOM agar lebih efektif dan efisien.
2. Menggunakan konsep K-SOM untuk menyelesaikan permasalahan yang lain.
3. Untuk mendapatkan nilai yang maksimal, diperlukan beberapa kali komputasi untuk mendapatkan hasil yang terbaik.

# DAFTAR PUSTAKA


Ansotegui, C., Bonet, M. L., & Levy, J. (2013). SAT-based MaxSAT algorithms. *Artificial Intelligence , 196*, 77-105.

Bryan, R., German, S., & Velev, M. (1999). *Microprocessor Verification Efficient Decision Procedures for a Logic of Equality with Uninterpreted Function.*

D'Urso, P., De Giovanni, L., & Massari, R. (n.d.). Smoothed Self-Organizing Map for Robust Clustering.

Fausett, L. (1993). *Fundamentals of Neural Networks: Architectures, Algorithms And Applications.* Pearson.

Hameed, A. A., Karlik, B., Salman, M. S., & Eleyan, G. (2019). Robust adaptive learning approach to self-organizing map.

Haykin, S. O. (1999). *Neural Networks: A Comprehensive Foundation.* England: Pearson Education.

Hendrik, Anjomshooa, A., & Tjoa, A. M. (2014). Towards Semantic Mashup Tools For Big Data Analysis. *Proceding of the Information & Communication Technology-EurAsia Conference 2014*, (pp. 100-145). Bali.

Hidayat, T. (2013). Logic in Computer Science. *Proceedings of International Conference on Information*.

Hidayat, T. (2014). *Logika Proposisi.* Yogyakarta: Dar Firqin.

Kohenan, T. (1989). *Self-organizing feature maps." Self-organization and associative memory.* Springer Berlin Heidelberg.

Li, C. M., & Anbulagan, A. (1997). Heuristics based on unit propagation for satisfiability problems. *Proceedings of IJCAI*, 366-371.

Marques-Silva, J. P., & Sakallah, K. A. (1997). GRASP-a new search algorithm for satisfiability. *Proceddings of the 1996 IEEE/ACM international conference on Computer-aided design*, 220-227.

Moskewicz, M. (2001). Chaff: Engineering an Efficient SAT Solver. *39th Design Automation Conference*.

Ouimet, M., & Lundqvist, K. (2007). Automated Verification of Completeness and Consistency of Abstract State Machine Specifications.

Palikareva, H., Ouaknine, J., & Roscoe, A. (2011). SAT-solving in CSP trace refinement.

Pratama, D. P. (2018). Penyelesaian Boolean Satisfiability Problem Dengan Algoritma Davis Putnam Logemann Loveland (DPLL) Menggunakan JAVA.

*Python (Bahasa Pemograman)*. (n.d.). Retrieved from https://id.wikipedia.org/wiki/Python_(bahasa_pemrogram an)

Saraswati, A., Nguyen, V., Hagenbuchner, M., & Tsoi, A. C. (2018). High-Resolution Self-Organizing Map for Advanced Visualization and Dimension Reduction.

Setiawan, A. M. (2013). *Integrated Framework For Business Process Complexity Analysis*. Retrieved from ECIS 2013 Completed Research: http://aisel.aisnet.org/ecis2013_cr/49

Shacham, O., & Yorav, K. (2006). Adaptive Application of SAT Solving Techniques.

Taufiq, H. (2015). *Argumentasi dan Validitas.* Yogyakarta: Darqin.

Wahid, F. (2014). The Antecedents And Impacts of a Green Eprocurement Infrastructure: Evidence From The Indonesian Public Sector. *International Journal of internet Protocol Technology*, *7*(4), 210-218.

## LAMPIRAN

LAMPIRAN A. contoh dari CNF *files* dengan nama ‘aim-50-1_6-yes1-4.cnf’

```
c FILE: aim-50-1_6-yes1-4.cnf
c
c SOURCE: Kazuo Iwama, Eiji Miyano (miyano@cscu.kyushu-u.ac.jp),
c          and Yuichi Asahiro
c
c DESCRIPTION: Artifical instances from generator by source.  Generators
c              and more information in sat/contributed/iwama.
c
c NOTE: Satisfiable
c
p cnf 50 80
16 17 30 0
-17 22 30 0
-17 -22 30 0
16 -30 47 0
16 -30 -47 0
-16 -21 31 0
-16 -21 -31 0
-16 21 -28 0
-13 21 28 0
13 -16 18 0
13 -18 -38 0
13 -18 -31 0
31 38 44 0
-8 31 -44 0
8 -12 -44 0
8 12 -27 0
12 27 40 0
-4 27 -40 0
12 23 -40 0
-3 4 -23 0
3 -23 -49 0
3 -13 -49 0
-23 -26 49 0
12 -34 49 0
-12 26 -34 0
19 34 36 0
-19 26 36 0
-30 34 -36 0
24 34 -36 0
-24 -36 43 0
6 42 -43 0
-24 42 -43 0
-5 -24 -42 0
5 20 -42 0
5 -7 -20 0
4 7 10 0
-4 10 -20 0
7 -10 -41 0
-10 41 46 0
-33 41 -46 0
33 -37 -46 0
32 33 37 0
6 -32 37 0
-6 25 -32 0
-6 -25 -48 0
```

```
-9 28 48 0
-9 -25 -28 0
19 -25 48 0
2 9 -19 0
-2 -19 35 0
-2 22 -35 0
-22 -35 50 0
-17 -35 -50 0
-29 -35 -50 0
-1 29 -50 0
1 11 29 0
-11 17 -45 0
-11 39 45 0
-26 39 45 0
-3 -26 45 0
-11 15 -39 0
14 -15 -39 0
14 -15 -45 0
14 -15 -27 0
-14 -15 47 0
17 17 40 0
1 -29 -31 0
-7 32 38 0
-14 -33 -47 0
-1 2 -8 0
35 43 44 0
21 21 24 0
20 29 -48 0
23 35 -37 0
2 18 -33 0
15 25 -45 0
9 14 -38 0
-5 11 50 0
-3 -13 46 0
-13 -41 43 0
```