Solusi pendekatan SAT Problem dengan Algoritma Genetika

by Muhammad Arnesz

Submission date: 20-Nov-2019 11:00PM (UTC+0700)
Submission ID: 1217840252
File name: Paper.doc (145.5K)
Word count: 1743
Character count: 10709

Solusi pendekatan SAT Problem dengan Algoritma Genetika

***Abstrak*—SAT problem yang merupakan permasalahan NP-Complete dan telah memiliki banyak aplikasi (SAT Solver) untuk menyelesaikannya. Namun dari seluruh SAT Solver yang ada belum ditemukan yang menawarkan solusi yang bukan eksak. Penelitian ini bertujuan untuk mengembangkan Modelan SAT Solver menggunakkan pemrograman java dan konsep pendekatan solusi menggunakan Algoritma Genetika. Setelah dilakukan penujian, kesimpulan yang dapat diambil adalah Model SAT Solver dari penelitian ini dapat menyelesaikan SAT Problem yang sangat sederhana.**

Kata kunci*—*SAT Problem, SAT Solver, Algoritma Genetika.

I. PENDAHULUAN

Dalam kehidupan sehari-hari terdapat berbagai macam permasalahan yang terjadi. Beberapa permasalahan tersebut dapat ditangani secara matematis, salah satu metodenya adalah SAT (Boolean Satisfiability) problem. SAT problem dapat digunakan untuk menentukan permasalahan kombinasi nilai kebenaran semua variabel proposisi dalam sebuah klausa/formula logika proposisional, sehingga membuat klausa/formula logika proposisional tersebut bernilai benar [1].

Untuk mempermudah proses penyelesaian masalah menggunakan metode SAT problem perlu adanya SAT solver. SAT Solver adalah perangkat lunak atau software yang digunakan untuk menyelesaikan SAT Problem (satisfiability problem), yaitu untuk menentukan apakah sebuah formula logika proposisi itu satisfiable atau unsatisfiable. Formula logika proposisi yang diselesaikan dengan SAT Solver harus dalam bentuk CNF (conjunctive normal form)[2].

Dalam berbagai kasus dan permasalahan yang kompleks pada dunia nyata, SAT solver terkadang akan mengalami kesulitan. Hal ini disebabkan karena pada awalnya SAT solver dibuat dengan menggunakan algoritma basic dan hanya untuk mengatasi permasalahan yang simpel (mudah). Oleh karena itu optimasi algoritma pembentuk SAT solver akan menjadi essential (penting). Peningkatan SAT solver berarti kualitas solusi yang didapatkan akan lebih baik, proses komputasi yang memakan waktu lebih sedikit dan biaya (resource) yang diperlukan akan jadi jauh lebih rendah daripada SAT solver sebelumnya.

Selama beberapa dekade terakhir penelitian tentang algoritma pembentuk SAT solver yang efisien dan scalable untuk menyelesaikan SAT problem berkembang dengan pesat. Perkembangan ini telah banyak berkontribusi pada kemajuan teknologi terutama dalam hal perkembangan kemampuan kita (manusia) untuk secara otomatis memecahkan masalah yang melibatkan puluhan ribu variabel dan jutaan constraint (klausa) [3]. Dalam dunia nyata salah satu contoh penggunaan SAT solver adalah dalam permasalahan EDA (Electronic Design Automation). Permasalahan tersebut meliputi formal equivalence checking, model checking, formal verification of pipelined microprocessors [4], automatic test pattern generation, routing of FPGAs [5], perencanaan dan permasalahan penjadwalan, dan sebagainya. Oleh karena itu SAT solver sekarang dianggap sebagai komponen penting dalam desain EDA.

Belakangan ini trend perkembangan algoritma pembentuk SAT solver tersebut mengarah ke Soft Computing. Salah satu cabang dari Soft Computing adalah algoritma genetika (GA). Algoritma genetika adalah algoritma metaheuristik yang terinspirasi teori Darwin mengenai proses seleksi alam dan termasuk ke dalam kelas yang lebih besar yaitu evolutionary algorithm (EA). Algoritma genetika biasanya digunakan untuk menghasilkan solusi berkualitas tinggi untuk masalah optimasi dan pencarian dengan mengandalkan operator yang terinspirasi alam seperti mutasi, crossover dan seleksi [6].

Sebenarnya sudah ada banyak SAT solver yang sudah dikembangkan dengan algoritma yang berbeda seperti: GRASP[9], Satz[8], WalkSat[7], zChaff dan mChaff[10]. Namun SAT solver tersebut sebagian besar masih membutuhkan resource (komputasi) yang besar untuk dijalankan serta memakan banyak waktu dalam proses eksekusinya karena solusi yang ditawarkan berupa eksak. Algoritma Genetika sebagai algoritma optimasi yang dapat menghasilkan solusi berkualitas tinggi dengan resource relatif kecil menjadi kandidat yang kuat untuk menjadi algoritma pembentuk SAT solver. Penelitian ini akan mencoba mengembangkan algoritma (SAT solver) dengan algoritma genetika untuk menyelesaikan SAT Problem.

II. LANDASAN TEORI

A. *Formula CNF*

Sebuah formula proposisi yang memenuhi syarat sebagai formula normal-form disebut Formula CNF (conjunctive normal form). Syarat formula normal-form adalah sebagai berikut[11]:

1. Hanya memiliki operator ∧, ∨, dan ¬.
2. Operator ¬ hanya boleh pada proposisi

Contoh formula-formula CNF (conjunctive normal form):

1. a ∧ b
2. a ∧ (b ∨ ¬c)
3. a ∧ b ∧ (¬a ∨ ¬b)

4. $(a \vee \neg b) \vee (c \vee \neg d)$
5. $(a \vee \neg b) \wedge b \wedge (\neg a \vee \neg c)$
6. $(a \vee \neg b) \vee (c \vee \neg d) \wedge (b \vee d)$
7. $(a \vee \neg b) \wedge (b \vee \neg c) \wedge (c \vee \neg a)$

B. *SAT Problem dan SAT Solver*

SAT Problem adalah proses menentukan apakah sebuah formula adalah formula yang satisfiabel atau unsatisfiabel. Sebuah formula dapat dikatakan satisfiabel jika terdapat kombinasi nilai kebenaran komponen (variabel) pembentuknya yang membuat formula tersebut bernilai benar TRUE dan sebaliknya jika komponen penyusunya membuat formula menjadi FALSE maka disebut unsatisfiabel [2]. Berikut contohnya:

1. $a \wedge b \wedge (\neg c \vee \neg d)$: tidak satisfiabel, karena tiap kemungkinan variabel yang dimasukkan menjadikkan formula menjadi FALSE
2. $(a \vee \neg b) \wedge b \wedge (\neg a \vee \neg c)$: satisfiabel, karena jika a = F, b = T, dan c =F maka formula menjadi TRUE.
3. $a \wedge b$: satisfiabel, karena jika a=T dan b=T maka formula menjadi TRUE.
4. $a \wedge (b \vee \neg c)$: satisfiabel, karena jika a=T dan c=F maka formula menjadi TRUE.

C. *Algoritma Genetika*

Algoritma genetika adalah algoritma komputasi yang terinspirasi teori Darwin. Algoritma ini dikembangkan oleh Goldberg dan mengatakan bahwa algoritma genetika adalah komputasi untuk mencari solusi dari suatu permasalahan dengan cara yang lebih alami (by nature).

Susunan terkecil dari algoritma genetika adalah karakter, simbol, bilangan ataupun karakter dari sebuah permasalahan. Susunan terkecil ini merupakan sebuah nilai dari gen/allel kumpulan gen/allel merupakan penyusun suatu chromosome. Kumpulan chromosome dinamakan populasi. Choromosome sendiri merupakan representasi dari sebuah solusi.

Sebuah generasi terjadi ketika kumpulan Choromosome berevolusi secara berkelanjutan. Fitness adalah fungsi untuk mengukur nilai/fungsi objektif. fungsi objektif adalah tingkat keberhasilan solusi terhadap masalah yang ingin diselesaikan dalam tahap evaluasi tiap generasi chromosome. Jika sebuah chromosome punya nilai fitness yang tinggi dibandingkan dengan choromosome yang lain pada generasi tersebut. Maka choromosome tersebut akan memiliki peluang lebih besar untuk terpilih lagi pada generasi selanjutnya. Hal tersebut adalah tahapan seleksi.

Pada proses crossover terjadi perkawinan antar chromosome-chromosome dalam satu generasi hasil dari perkawinan itu adalah offspring. Pada tahap selanjutnya Mutasi terjadi anomali terhadap susunan gen yang merepresentasikan perubahan susunan unsur genetik suatu mahkluk hidup akibat adanya faktor alam/seleksi.

Algoritma genetika dapat dinyatakan selesai jika sudah muncul chromosome yang terbaik/memenuhi kriteria/sudah mencapai batas iterasi. Chromosome itulah yang menjadi solusi permasalahan yang ingin diselesaikan.

.

III. HASIL DAN PEMBAHASAN

A. *Pemodelan*

Misal terdapat Formula CNF: $(p \vee q) \wedge (r \vee \neg s) \wedge (r) \wedge (s)$

1. Inisiasi
   Pertama kita tentukan jumlah populasi misalnya 4 kemudian kita tentukan gen/allelnya secara acak. Karena permasalahan ini adalah SAT Problem maka nilai variabel hanya 0 atau 1. Prosesnya adalah sebagai berikut:
   - khromosome[1]=[p;q;r;s]=[0;0;0;0]
   - khromosome[2]=[p;q;r;s]=[1;1;0;0]
   - khromosome[3]=[p;q;r;s]=[0;1;1;1]
   - khromosome[4]=[p;q;r;s]=[0;1;1;0]

2. Evaluasi Chromosome
   Evaluasi Chromosome atau penentuan nilai fungsi objektif dapat ditemukan menggunakan rumus fungsi_objektif_chromosome = jumlah klausa - $((p \wedge q) + (r \vee \neg s) + (r) + (s))$. Prosesnya adalah sebagai berikut:
   - fungsiObjektif(khromosome[1]) = jumlah klausa - $((0 \wedge 1) + (0 \vee \neg 1) + (0) + (1))$
     $= 4 - (0 + 1 + 0 + 0)$
     $= 3$
   - fungsiObjektif(khromosome[2]) = jumlah klausa - $((1 \wedge 1) + (0 \vee \neg 1) + (0) + (1))$
     $= 4 - (1 + 1 + 0 + 1)$
     $= 3$
   - fungsiObjektif(khromosome[3]) = jumlah klausa - $((0 \wedge 1) + (1 \vee \neg 1) + (1) + (1))$
     $= 4 - (1 + 1 + 1 + 1)$
     $= 0$
   - fungsiObjektif(khromosome[4]) = jumlah klausa - $((0 \wedge 1) + (1 \vee \neg 0) + (1) + (0))$
     $= 4 - (1 + 1 + 1 + 0)$
     $= 1$

3. Seleksi Chromosome
   Pada tahap ini akan ditentukan nilai fitness. Nilai tersebut akan menentukan seberapa besar kemungkinan terpilih chromosome tersebut. Untuk mendapatkan nilai fitness dalam permasalahan ini dapat menggunakan fungsi fitness = (1/(1+fungsiObjektif)), untuk menghindari error dari

pembagian nol maka fungsi-objektif ditambah dengan satu. Prosesnya adalah sebagai berikut:

- fitness[1] = 1 / (fungsiObjektif[1]+1)
  =1/(3+1)
  = 0.25
- fitness[2] = 1 / (fungsiObjektif[2]+1)
  =1/(3+1)
  = 0.25
- fitness[3] = 1 / (fungsiObjektif[3]+1)
  =1/ (0+1)
  = 1
- fitness[4] = 1 / (fungsiObjektif[4]+1)
  =1/(1+1)
  = 0.5
- totalFitness = 0.25 + 0.25 + 1 + 0.5 = 2

Setelah itu kita mencari probabilitas: P[i] = fitness[i]/total_fitness. Prosesnya adalah sebagai berikut:

- Peluang[1] = 0.25 /2
  = 0.125
- Peluang[2] = 0. 25/ 2
  = 0.125
- Peluang[3] = 1/ 2
  = 0.5
- Peluang[4] = 0.5 / 2
  = 0.25

Kemudian kita tentukan kumulatifnya

- Kumulatif[1] = 0.125
- Kumulatif [2] = 0.125+ 0.125
  = 0.25
- Kumulatif [3] = 0.125+ 0.125 + 0.5
  = 0.75
- Kumulatif [4] = 0.125+ 0.125 + 0.5 +0.25
  = 1

Kemudian kita generate angka acak:

- Acak[1] = 0.56
- Acak[2] = 0.67
- Acak[3] = 0.11
- Acak[4] = 0.82

Populasi berubah menjadi:

- khromosome[1]=khromosome[3]
- khromosome[2]=khromosome[3]
- khromosome[3]=khromosome[1]
- khromosome[4]=khromosome[4]

4. Crossover

Pertama kita bangkitkan bilangan acak R sebanyak jumlah populasi

- Acak[1] = 0.191
- Acak[2] = 0.259
- Acak[3] = 0.760
- Acak[4] = 0.006

Berdasarkan bilangan acak tersebut persilangan menjadi:

- khromosome[1] >< khromosome[4]
- khromosome[4] >< khromosome[1]

Kemudian kita bankitkan lagi bilangan acak untuk menentukan cut-point:

- Cut[1] = 1
- Cut[2] = 1

offSpring[1] = khromosome[1] >< khromosome[4]
= [0;1;1;1] >< [0;1;1;0]
= [0;1;1;0]
offSpring[4] = khromosome[4] >< khromosome[1]
= [0;1;1;0] >< [0;1;1;1]
= [0;1;1;1]
Sehingga populasi menjadi:

- khromosome[1]= [0;1;1;0]
- khromosome[2]= [0;1;1;1]
- khromosome[3]= [0;0;0;0]
- khromosome[4]= [0;1;1;1]

5. Mutasi

Misalkan yang mengalami mutasi adalah chromosome ke-2 gen/allel nomor 2 dan Chromosome ke-3 gen/allel nomor 3. Maka nilai gen/allel pada chromosome 2 sebut kita ganti dengan bilangan acak integer 0-1. Misalkan bilangan acak yang terpilih adalah 0 dan 1. Maka populasi chromosome setelah mengalami proses mutasi adalah:

- khromosome[1] = [0;1;1;0]
- khromosome[2] = [0;0;1;1]
- khromosome[3] = [0;0;1;0]
- khromosome[4] = [0;1;1;1]

Setelah itu proses akan terus berulang sampai jumlah iterasi yang telah ditentukan. Choromose dengan nilai fitness terbaik sampai denga akhir iterasi. Nantinya akan ditawarkan sebagai solusi/pendekatan solusi SAT Problem.

B. Implementasi

1. Inisiasi

```
//Initialize population
demo.population.initializePopulation(10);
```

2. Evaluasi Chromosome

```
//Calculate fitness of each individual
demo.population.calculateFitness();
```

3. Seleksi Chromosome

```
//Selection
void selection() {

    //Select the most fittest individual
    fittest = population.getFittest();

    //Select the second most fittest individual
    secondFittest = population.getSecondFittest();
}
```

4. Crossover

```
//Crossover
void crossover() {
    Random rn = new Random();

    //Select a random crossover point
    int crossOverPoint = rn.nextInt(population.individuals[0].geneLength);

    //Swap values among parents
    for (int i = 0; i < crossOverPoint; i++) {
        int temp = fittest.genes[i];
        fittest.genes[i] = secondFittest.genes[i];
        secondFittest.genes[i] = temp;

    }

}
```

5. Mutasi

```
//Mutation
void mutation() {
    Random rn = new Random();

    //Select a random mutation point
    int mutationPoint = rn.nextInt(population.individuals[0].geneLength);

    //Flip values at the mutation point
    if (fittest.genes[mutationPoint] == 0) {
        fittest.genes[mutationPoint] = 1;
    } else {
        fittest.genes[mutationPoint] = 0;
    }

    mutationPoint = rn.nextInt(population.individuals[0].geneLength);

    if (secondFittest.genes[mutationPoint] == 0) {
        secondFittest.genes[mutationPoint] = 1;
    } else {
        secondFittest.genes[mutationPoint] = 0;
    }
}
```

IV. EKSPERIMEN

Dilakukan sebuah pengujian untuk menyelesaikan sebuah SAT Problem. SAT Problem tersebut disimpan dalam java class. Tingkat kesulitan permasalahan yang diujikan adalah SAT Problem yang sederhana. Pengujian dilakukan dengan menggunakan Notebook dengan prossesor i5-7200U @2,5Ghz dan RAM 8 GB.

Pemodelan SAT Solver dengan Algoritma Genetika ini berhasil menyelesaikan permasalahan yang diujikan dalam waktu kurang lebih 3 detik. Pemodelan SAT Solver ini masih belum mampu untuk menyelesaikan masalah komplekks seperti problem-problem yang ada di Kompetisi SAT.

V. KESIMPULAN

Telah berhasil dibentuk sebuah pemodelan SAT Solver menggunakan bahasa pemrograman java. Algoritma yang dipakai pada pemodelan ini adalah Algoritma Genetika. Pemodelan SAT Solver ini mampu menyelesaikan SAT Problem yang diujikan. Namun perlu diketahui bahwa problem yang diujikan masih sangatlah sederhana.

Kinerja Model SAT Solver ini masih belum bisa dibandingkan dengan SAT Solver yang mengikuti kompetisi SAT. Perlu pengujian untuk menyelesaikan SAT Problem dengan kasus besar dan komputer dengan performa besar juga.

Pada penelitian selanjutnya, perlu pengembangan lebih lanjut terhadap Pemodelan SAT Solver ini dengan menyesuaikan konsep pemodelan dengan struktur java yang sebenarnya sehingga dapat meningkatkan kinerja SAT Solver nantinya.

VI. REFERENSI

[1] D. Du, J. Gu and P. M. Pardalos, Satisfiability Problem: Theory and Applications : DIMACS Workshop, March 11-13, 1996, American Mathematical Soc, 1997.

[2] M. Huth and M. Ryan, Logic in Computer Science: Modelling and Reasoning about Systems, New York: Cambridge University Press, 2004.

[3] O. Ohrimenko, P. J. Stuckey and M. Codish, "Propagation = lazy clause generation," *Principles and Practice of Constraint Programming – CP 2007*, p. 544–558, 2007.

[4] R. E. Bryant, S. M. German and M. N. Velev, "Microprocessor Verification Using Efficient Decision Procedures for a Logic of Equality with Uninterpreted Functions," *TABLEAUX '99 Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods* , pp. 1-13, 1999.

[5] G.-J. Nam, K. A. Sakallah and R. A. Rutenbar, "A new FPGA detailed routing approach via search-based Boolean satisfiability," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2002.

[6] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, 1996.

[7] J. Marques-Silva and K. Sakallah, "GRASP: A New Search Algorithm for Satisfiability," *Proceedings of International Conference on Computer-Aided Design*, pp. 220-227, 1996.

[8] C. Li and Anbulagan, "Heuristics Based on Unit Propagation for Satisfiability Problems," *Proceedings of IJCAI*, pp. 366-371, 1997.

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

[10] Y. Vizel, Y. Weissenbacher and S. Malik, "Boolean Satisfiability Solvers and Their Applications in Model Checking," *Proceedings of the IEEE*, p. 103, 2015.

[11] T. Hidayat, Logika Proposisi, Yogyakarta: Dar Firqin, 2014.

Solusi pendekatan SAT Problem dengan Algoritma Genetika

ORIGINALITY REPORT

19%	9%	0%	18%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Universitas Islam Indonesia Student Paper	11%
2	softskillrifki.blogspot.com Internet Source	2%
3	raisnugrohop.blogspot.com Internet Source	2%
4	Submitted to Universitas Brawijaya Student Paper	2%
5	en.wikipedia.org Internet Source	1%
6	id.scribd.com Internet Source	1%
7	eprints.uny.ac.id Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		