PENYELESAIAN GAME TEKA-TEKI SILANG DENGAN MENERAPKAN ALGORITMA BACKTRACKING

Tiffany Adriana - 13505068

Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
if15068@students.if.itb.ac.id

ABSTRAK

Pada saat ini telah banyak permainan (*game*) yang diperkenalkan kepada masyarakat umum. Baik *game* tradisional maupun *game* digital yang membutuhkan media khusus untuk memainkannya. Berbagai jenis *game* dimainkan oleh kalangan umur tertentu. Salah satu jenis *game* yang banyak dimainkan oleh orang dari berbagai umur adalah *game* Teka-Teki Silang. Permainan ini merupakan salah satu contoh *Constraint Satisfaction Problem*, yang memiliki batasan tertentu untuk menghasilkan solusi. Batasan pada *game* Teka-Teki Silang adalah panjang kata yang akan dimasukkan dan kesesuaian huruf pada dua kata yang saling bersilangan. Ada orang yang menghabiskan banyak waktu untuk menyelesaikan Teka-Teki Silang. Hal ini dapat disebabkan karena kurang efektifnya modus yang digunakan untuk menyelesaikan Teka-Teki Silang. Pada makalah ini penulis menampilkan pembahasan mengenai pemilihan algoritma yang tepat untuk menyelesaikan *game* Teka-Teki Silang secara efektif dan dalam waktu yang diharapkan lebih singkat. Algoritma yang digunakan adalah *Backtracking* (Runut Balik). Algoritma ini dilakukan ketika User akan memasukkan kata ke kotak. Pemeriksaan dilakukan terhadap panjang kata yang dimasukkan, dan kesesuaian huruf pada kata yang saling bersilangan. Jika terdapat ketidak-sesuaian, maka dilakukan *backtrack*. Pemeriksaan dilakukan sampai semua kotak terpenuhi dengan kata yang tepat.

Kata kunci: Backtracking, Constraint Satisfaction Problem, Domain, Variabel, Game Teka-Teki Silang.

1. PENDAHULUAN

Manusia adalah makhluk dengan aktivitas tinggi. Setiap orang memiliki tugasnya masing-masing. Untuk melepaskan diri dari kepenatan pekerjaan, banyak orang yang menyegarkan pikiran dengan bermain *game.* Ada banyak jenis *game* yang telah dikenal luas. Mulai dari *game* tradisional sampai game *digital*.

Salah satu contoh *game* yang telah mendunia adalah teka-teki silang (*crossword puzzle*). Banyak orang memainkan *game* teka-teki silang, mulai dari anak kecil sampai orang dewasa. Tujuan *game* teka-teki silang adalah mengisi kotak-kotak dengan kata yang terdapat pada daftar kata yang tersedia.

Pada makalah ini penulis ingin menunjukkan cara menyelesaikan teka-teki silang secar aefektif. Algoritma yang digunakan di sisni adalah backtracking.

2. PEMECAHAN PUZZLE TEKA-TEKI SILANG

2.1 Constraint Satisfaction Problem

Constraint Satisfaction Problem (CSP) adalah problem matematis dimana harus ditemukan suatu state atau objek yang memenuhi sejumlah *constraint* (batasan) atau kriteria.

Constraint meng-enumerasi nilai yang mungkin digunakan oleh suatu set variabel. Biasanya, domain terbatas adalah set terbatas dari elemen *arbitrary*. CSP pada domain seperti ini mengandung sebuah set variabel yang nilainya hanya bias diambil dari domain, dan suatu set *constraint* yang tiap *constraint*-nya menspesifikasi nilai yang boleh untuk suatu grup variabel. Solusi untuk masalah ini adalah evaluasi dari variabel-variabel yang memenuhi semua *constraint*. Dengan kata lain, solusi untuk meng-*assign* suatu nilai pada tiap variabel adalah dengan suatu cara dimana semua *constraint* dapat dipenuhi oleh nilai-nilai ini.

Karakterisasi suatu Constraint Satisfaction Problem biasanya dibentuk dan didefenisikan dengan

$$\langle X, D, C \rangle$$

dimana

X : suatu set variabel
D : domain nilai
C : suatu set *constraint*

Tiap *constraint* adalah kembalian sepasang

$$\langle t, R \rangle$$

dimana **t** adalah sebuah *tuple of variable* dan **R** adalah *set of tuple of nilai*, dengan hasil dari **R** adalah relasi. Semua tuple ini berjumlah sama dengan elemen. Evaluasi dari variabel adalah suatu fungsi dari variabel

$$v : X \rightarrow D \quad (3)$$

Evaluasi seperti ini memenuhi *constraint*

$$\langle (x_1, \ldots, x_n), R \rangle$$

jika

$$(v(x_1), \ldots, v(x_n)) \in R$$

Sedangkan solusi adalah evaluasi yang memenuhi semua *constraint*.

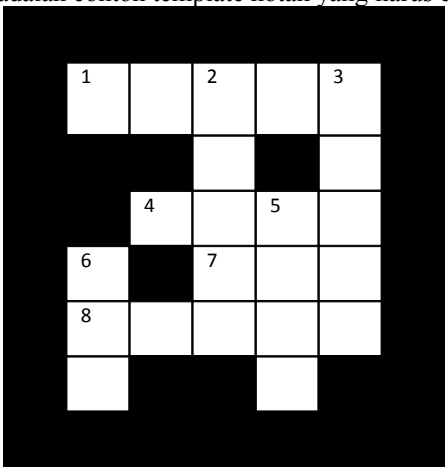
Pada kenyataannya, beberapa *constraint* yang sering digunakan diekspresikan dengan bentuk yang *compact*, bukan meng-enumerasi semua nilai variabel yang mungkin. Salah satu *constraint* yang paling sering digunakan adalah *constraint* yang mengekspresikan bahwa nilai dari beberapa variabel semuanya berbeda.

Salah satu contoh sederhana dari *Constraint* Satisfaction Problem adalah permainan Teka-Teki Silang.

2.2 Teka-Teki Silang

Teka-teki Silang merupakan permainan yang berkaitan dengan huruf. Pada awal permainan tersedia sebuah template game board berbentuk segi empat yang tersusun dari kotak-kotak kecil. Di antara kotak kecil ini ada yang dapat diisi oleh User, yaitu kotak kosong – biasanya direpresentasikan dengan kotak berwarna putih. Kotak putih yang harus diisi User terdiri dari dua jalur, yaitu mendatar dan menurun. Pada beberapa kotak tertentu terjadi persilangan antara dua kata, dimana tempat yang bersilangan harus diisi dengan huruf yang sama. Tujuan utama permainan adalah mengisi semua kotak berwarna putih dengan kata yang telah ditentukan. User harus menyesuaikan panjang kotak yang tersedia dengan panjang kata yang akan dipilih.

Berikut adalah contoh template kotak yang harus diisi:



Gb. 1 Template *Game* Teka-Teki Silang

Angka 1,2,3,4,5,6,7,8 pada board menunjukkan kata yang akan dimulai pada kotak tersebut.

2.3 Algoritma Backtracking

Backtracking adalah algoritma rekursif yang menjaga *assignment* parsial dari variabel. Pada awalnya semua variabel yang akan diperiksa belum di-*assign* dengan nilai apapun. Pada tiap tahap, sebuah variabel dipilih dan di-*assign* dengan semua nilai yang mungkin secara bergantian. Untuk setiap nilai, konsistensi dari *assignment* parsial dengan *constraint* diperiksa, jika konsisten maka dilakukan fungsi rekursif. Jika semua nilai sudah dicoba, algoritma akan melakukan backtrack. Pada algoritma Backtracking, konsistensi didefenisikan sebagai pemenuhan semua *constraint* yang semua variabelnya telah di-*assign*.

Algoritma Backtracking mencoba tiap kemungkinan sampai mendapatkan solusi yang benar. Selama pencarian, jika alternatif solusi tidak memberikan hasil yang benar, pencarian melakukan backtrack ke titik pilihan – titik yang memiliki alternatif yang berbeda, kemudian mencoba alternatif selanjutnya. Jika alternatif yang dijalankan tidak memberikan hasil, pencarian akan kembali ke titik pilihan sebelumnya dan mencoba alternatif berikutnya yang ada di sana. Jika tidak ada titik pilihan lagi, maka pencarian gagal.

Pencarian ini biasanya didapat dari fungsi rekursif dimana tiap *instance* mengambil satu atau lebih variabel dan secara alternatif meng-*assign* variabel tersebut dengan semua nilai yang mungkin. Algoritma Backtracking mirip dengan pencarian *Depth-First Search*, tapi algoritma Backtracking menggunakan ruang yang lebih kecil. Hal ini dikarenakan Backtracking hanya menyimpan satu solusi terakhir, dan meng-*update* solusi ini saja.

Untuk mempercepat pencarian, sebelum melakukan pemanggilan rekursif, jika suatu nilai sudah dipilih algoritma akan melakukan salah satu dari dua kemungkinan berikut. Algoritma akan menghapus nilai tersebut dari domain yang belum di-*assign* (melakukan forward checking), atau algoritma akan memeriksa semua batasan untuk melihat nilai lain yang tidak termasuk dalam nilai yang baru di-*assign* ini (*constraint propagation*).

2.4 Menyelesaikan *Game* Teka-Teki Silang dengan Algoritma Backtracking

Kita akan menyelesaikan template kotak *game* teka-teki silang yang telah ditunjukkan di atas dengan daftar kata yang harus diisi adalah sebagai berikut:

AFT	LASER	KNOT
ALE	LEE	TIE
EEL	LINE	KEEL
HEEL	SAILS	

HIKE SHEET
HOSES STEER

CSP biasanya direpresentasikan sebagai graf tidak berarah, disebut *Constraint Graph.* Pada graf ini, simpul merepresentasikan variabel dan busur merepresentasikan *constraint* biner.

Sekarang kita perkenalkan variabel untuk merepresentasikan tiap kata dalam puzzle. Jadi kita memiliki variabel:

Tabel 1, Daftar Variabel dan Domain

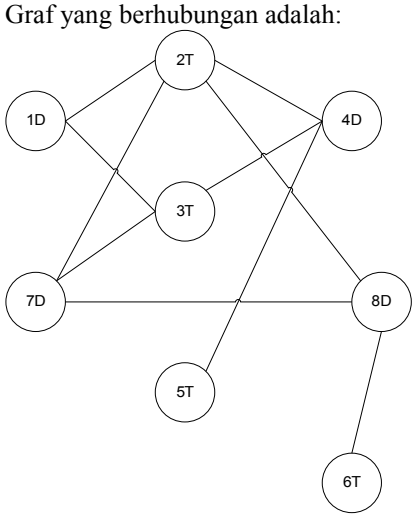
Variabel	Sel	Domain
1 Datar	1	{HOSES, LASER, SAILS, SHEET, STEER}
4 Datar	4	{HEEL, HIKE, KEEL, KNOT, LINE}
7 Datar	7	{AFT, ALE, EEL, LEE, TIE}
8 Datar	8	{HOSES, LASER, SAILS, SHEET, STEER}
2 Turun	2	{HOSES, LASER, SAILS, SHEET, STEER}
3 Turun	3	{HOSES, LASER, SAILS, SHEET, STEER}
5 Turun	5	{HEEL, HIKE, KEEL, KNOT, LINE}
6 Turun	6	{AFT, ALE, EEL, LEE, TIE}

Domain dari tiap variabel adalah daftar kata yang mungkin menjadi nilai variabel tersebut. Jadi, variabel 1Datar membutuhkan kata dengan panjang lima huruf, 2Turun membutuhkan kata dengan panjang lima huruf, 3Turun membutuhkan kata dengan panjang empat huruf, dst. Perlu diingat karena tiap domain memiliki 5 elemen dan ada 8 variabel, jadi jumlah state yang harus diperhitungkan mendekati 5^8=390.625.

Constraint pada puzzle ini seluruhnya adalah *constraint* biner:

1Datar[3] = 2Turun[1] huruf ketiga dari 1Datar harus sama dengan huruf pertama dari 2Turun
1Datar[5] = 3Turun[1]
4Datar[2] = 2Turun[3]
4Datar[3] = 5Turun[1]
4Datar[4] = 3Turun[3]
7Datar[1] = 2Turun[4]
7Datar[2] = 5Turun[2]
7Datar[3] = 3Turun[4]
8Datar[1] = 6Turun[2]
8Datar[3] = 2Turun[5]
8Datar[4] = 5Turun[3]
8Datar[5] = 3Turun[5]

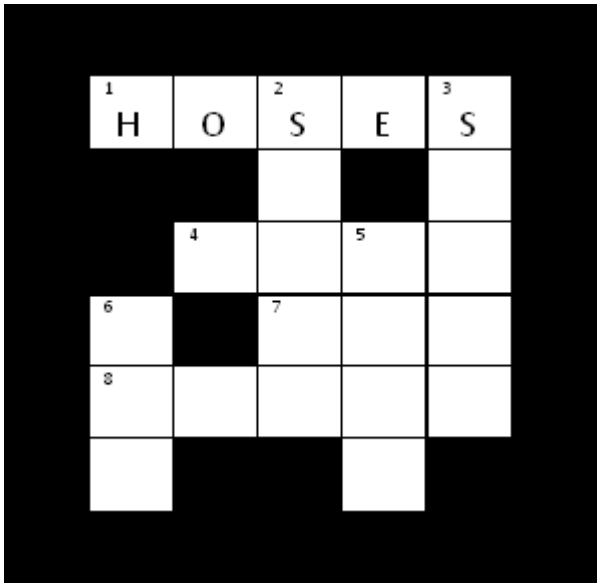
Graf yang berhubungan adalah:



Gb. 2 Graf keterhubungan antar variabel

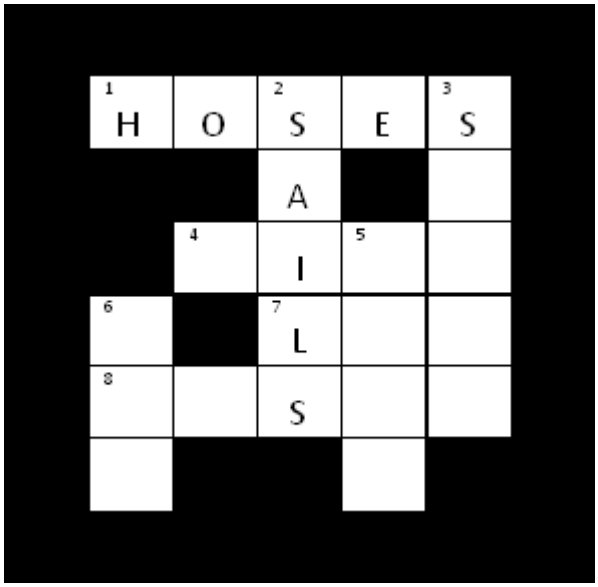
Kita akan menyelesaikan puzzle ini dengan Backtracking. Pertama kita dapat mengurutkan variabel dengan urutuan sebagai berikut: 1Datar, 2Turun, 3Turun, 4Datar, 7Datar, 5Turun, 8Datar, 6Turun. Kemudian kita mulai meng-*assign*

1Datar ← HOSES



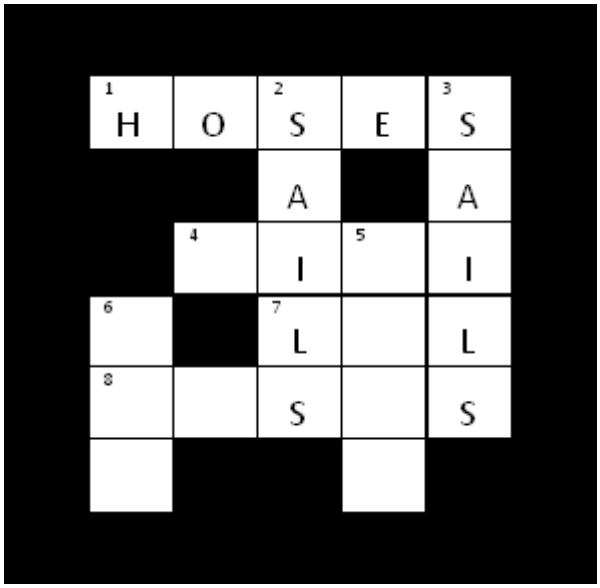
Gb. 3 1Datar di-*assign* dengan kata HOSES

2Turun ← HOSES //gagal, 1Datar[3] tidak sama dengan 2Turun[1]
← LASER //gagal
← SAILS



Gb. 4 2Turun di-*assign* dengan kata SAILS

3Turun ← HOSES //gagal
← LASER //gagal
← SAILS

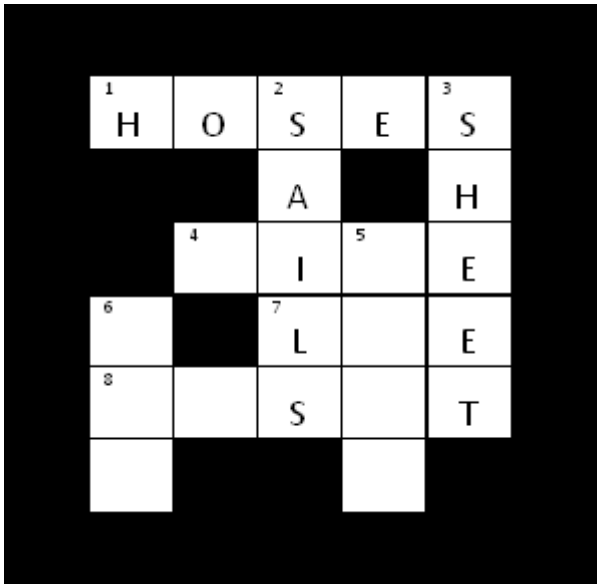


Gb. 5 3Turun di-*assign* dengan kata SAILS

4Datar ← HEEL //gagal
← HIKE //gagal
← KEEL //gagal
← KNOT //gagal
← LINE

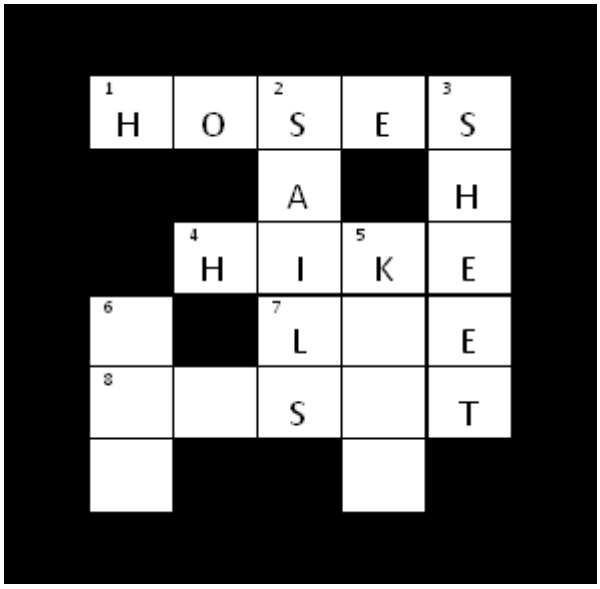
Karena semua kemungkinan gagal, maka dilakukan backtrack pada variabel yang terkahir diisi, ysitu 3Turun

3Turun ← SHEET



Gb. 6 3Turun di-*assign* dengan kata SHEET

4Datar ← HEEL //gagal
4Datar ← HIKE



Gb.7 4Datar di-*assign* dengan kata HIKE

Begitu seterusnya sampai ditemukan hasil sebagai berikut:

Gb.8 *Game* Teka-Teki Silang berhasil diselesaikan

3. KESIMPULAN

Algoritma backtrack adalah algoritma yang cukup mangkus, karena kita tidak perlu memeriksa semua variabel. Tapi hanya perlu mengembangkan satu variabel, dan melakukan backtrack jika terjadi kesalahan. Hal ini dapat memberikan keuntungan, yaitu efektivitas waktu dalam pengerjaan. Karena itu algoritma backtracking banyak digunakan dalam berbagai bidang yang berkaitan dengan Intelijensi Buatan (Artificial Inteligence).

Untuk menyelesaikan *game* teka-teki silang, algoritma backtracking dapat dikatakan cukup mangkus. Pada satu waktu tertentu backtracking hanya melakukan pemeriksaan terhadap satu variabel dan melakukan pengembangan terhadap variabel tersebut. Hal ini memungkinkan kita untuk menyelesaikan *game* dalam waktu yang relatif lebih singkat dan pencarian yang dilakukan lebih efektif.

REFERENSI

[1] Munir, Rinaldi. “Strategi Algoritmik”, 2007.
[2] Backtracking Algorithm.
http:\\en.wikipedia.org
Diakses tanggal 17 Mei 2007
[3] Constraint Satisfaction Problem
http:\\en.wikipedia.org
Diakses tanggal 17 Mei 2007