Pemanfaatan *ID3* dan *Influence Map* pada *Game Turn-Based Strategy*

Vincent Utomo[1], Liliana[2], Gregorius Satia Budhi[3]
Program Studi Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra
Jl. Siwalankerto 121-131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) - 8417658
E-mail: centralofmsv@gmail.com[1], lilian@petra.ac.id[2], greg@petra.ac.id[3]

ABSTRAK

Salah satu aspek menarik di dalam *game* adalah adanya *virtual enemy (AI)* sebagai lawan bagi *player*. Aspek ini cukup penting untuk memberikan perlawanan bagi *player*, terutama bagi *game* satu *player*. Salah satu *game* yang membutuhkan *AI* yang baik adalah *turn-based strategy game*. Untuk dapat membuat *AI* yang menarik dan menantang bagi *player*, dibutuhkan kemampuan yang sepadan dan dinamis pada *AI* yang dibuat.

Untuk dapat membuat *AI* yang dibutuhkan, penelitian ini menggunakan *ID3* dan *Influence Map* sebagai dasar dari *AI* yang digunakan untuk *game turn-based strategy*. Sistem yang dibuat meliputi program pelatihan *ID3* dan *class Influence Map* yang diletakkan di dalam *game* yang akan digunakan. Sistem *ID3* dibuat memiliki kemampuan untuk memproses data kontinu.

Hasil penelitian menunjukkan bahwa kemampuan *ID3* dalam membuat *decision tree* sangat dipengaruhi oleh data *training* yang digunakan. Hasil juga menunjukkan bahwa *propagation method* yang digunakan untuk sistem *Influence Map* dapat diatur agar sesuai dengan keinginan di dalam game.

Kata Kunci

ID3, Continuous Attribute, Influence Map, Artificial Intelligence, Turn-Based Strategy Game

ABSTRACT

One of interesting aspect in game is virtual enemy (AI) as enemy to the player. This aspect is important to give challenge to the player, especially for a single player game. One game that requires such AI is a turn-based strategy game. To be able to create an interesting and challenging AI for the player, it is require equally strong and dynamic AI.

To create such AI, ID3 and Influence Map will be used as the base of the AI used in turn-based strategy game. Applications made include ID3 training software and Influence Map class inside the game that will be used. ID3 System is made to have the ability to process continuous data.

The results showed that the ability to create a decision tree by ID3 is greatly influenced by the training data used. The results also showed that the propagation method used for the Influence Map system can be set to suit the requirements in the game.

Keywords

ID3, Continuous Attribute, Influence Map, Artificial Intelligence, Turn-Based Strategy Game

1. PENDAHULUAN

Pada saat ini *game* sudah berkembang dengan pesat baik dari segi grafis maupun permainan. Seiring perkembangan *game*, diperlukan juga adanya *AI* yang sepadan agar dapat memberikan perlawanan yang menantang bagi pemainnya. Salah satu *game* yang membutuhkan *AI* yang menantang adalah *game turn-based strategy*. Untuk dapat membuat *AI* yang dapat berpikir dengan baik dan dapat memberikan tantangan yang memadai untuk *game*, berbagai metode dapat digunakan.

Seiring dengan semakin tinggi kompleksitas suatu *game*, semakin rumit juga penyusunan *AI* yang harus dibuat, untuk mempermudah penyusunan suatu *AI* atau bahkan memberikan kemampuan belajar bagi *AI* yang dibuat dapat digunakan sistem pembuat *AI*, salah satunya adalah *ID3*. *AI* juga dapat diberi kemampuan untuk memiliki *map awareness* dengan menggunakan *Influence Map* agar dapat berpikir sesuai dengan keadaan terbaru.

2. TEORI PENUNJANG

2.1. *ID3*

ID3 adalah salah satu metode *decision tree* yang dapat berkembang secara otomatis (Belajar) dengan menggunakan data-data yang sebelumnya telah disediakan oleh pembuat *game*, atau dengan menggunakan penambahan data yang didapatkan seiring berjalannya *game*. Dengan adanya proses belajar ini, *AI* yang dikembangkan dapat berpikir menyerupai data yang dipelajari [1]. Untuk dapat membentuk *decision tree* yang optimal dari *ID3*, diperlukan jumlah pertanyaan yang tidak terlalu besar (sehingga mengurangi ke dalaman dari *decision tree*) [6].

Jika diinginkan, *ID3* dapat menggunakan seluruh *entropy* semaksimal mungkin (tanpa membuang *entropy* yang nilainya sangat kecil). Hal ini dilakukan untuk memastikan jumlah *output node* (*result*) sesuai dengan jumlah *result* pada data *training* yang berarti menghasilkan *decision tree* yang memiliki klasifikasi secara sempurna sesuai dengan data yang disediakan [2].

2.1.1. Perhitungan *Information Gain* dan *Entropy*

Information gain adalah nilai informasi yang diberikan oleh suatu atribut, atribut dengan nilai *information gain* terbesar akan diletakkan pada bagian atas *decision tree* yang dibuat. *Entropy* adalah nilai ketidakpastian yang ada dalam *dataset*, semakin tinggi nilainya, semakin tinggi juga nilai ketidakpastian yang ada. Rumus menghitung *entropy* dapat dilihat pada persamaan 1 [7].

$$\boldsymbol{E(S) = -\sum_{j=1}^{n} p(j) \log_2 p(j)} \quad (1)$$

E(S) adalah nilai *entropy* dari *set* S. n adalah jumlah nilai berbeda dari atribut yang dipilih pada *set* S. p(j) adalah proporsi nilai j pada *set* S. Rumus menghitung *information gain* dapat dilihat pada persamaan 2 [7].

$$\boldsymbol{IG(S,A) = E(S) - \sum_{i=1}^{m} p(A_i) E(S_i)} \quad (2)$$

IG(S,A) adalah *information gain* dari *set* S setelah dipecah dengan atribut A. m adalah jumlah nilai yang berbeda dari atribut A pada *set* S. A_i adalah salah satu nilai atribut A. S_i adalah *subset* dari S dimana semua isi atribut A adalah A_i. $p(S_i)$ adalah proporsi data dimana atribut A berisi nilai A_i pada *subset* S_i. $E(S_i)$ adalah entropi dari *subset* S_i.

2.1.2. *ID3* dengan *Atribut Continuous*

Data kontinu harus diproses secara khusus oleh *ID3* dengan cara pendiskretan. Cara pendiskretan yang dapat digunakan adalah mencari batasan *threshold* data kontinu menggunakan *greedy approaching*. Cara kerja *greedy approaching* adalah dengan menaruh sebuah pemisah sehingga terbentuk 2 buah grup, lalu melakukan pengecekan nilai *information gain* jika pemotongan dilakukan pada *threshold* yang bersangkutan, contoh metode ini dapat dilihat pada Gambar 1.

Category	Attribute Value	Action	Information Gain
A	17	Defend	
B	25	Defend	
	39	Attack	
	50	Defend	0.12

Category	Attribute Value	Action	Information Gain
A	17	Defend	
	25	Defend	
B	39	Attack	
	50	Defend	0.31

Category	Attribute Value	Action	Information Gain
A	17	Defend	
	25	Defend	
	39	Attack	
B	50	Defend	0.12

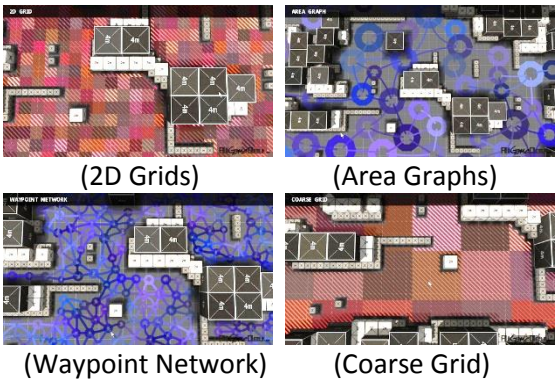
Gambar 1. Contoh *Greedy Approaching*

Sumber: Artificial Intelligence for *Game*s Second Edition

Metode ini menghasilkan single split, tidak peduli berapa banyak jumlah *continuous* data pada *set*. Jika pemecahan ingin dilakukan lebih dari 2 grup, dapat dilakukan pemecahan bertingkat. Sebenarnya hal ini tidak begitu diperlukan untuk *game*, karena *node* atribut yang sama mungkin muncul pada cabang *tree* yang berbeda, serta memiliki *subset* yang berbeda, sehingga hasil *threshold* juga akan berbeda [4].

2.2. *Influence Map*

Influence map dapat digunakan untuk memberikan *location awareness* pada *AI* dan dapat juga digunakan untuk melakukan *predicting* untuk serangan [3]. Komponen dari *influence map* adalah partisi dan konektivitas. Partisi akan dibagi menjadi *node* dan konektivitas digunakan untuk penyebaran *influence*. *Mapping* pada *influence map* dilakukan dengan mendapatkan nilai dari bagian-bagian berpengaruh di dalam *game* lalu mem*propagate* nilai itu kepada *node* atau area disekitarnya [5]. Cara untuk membuat partisi dari *map* mencakup *2D Grids*, *Area Graphs*, *Waypoint Network*, dan *Coarse Grid* seperti yang dapat dilihat pada Gambar 2.



Gambar 2. Macam-macam partisi *map*

Sumber: http://aigamedev.com/open/tutorial/influence-map-mechanics/

Fungsi penting bagi *influence map* adalah *propagation* dan *decay*. *Propagation* digunakan untuk penyebaran *influence*, sedangkan *decay* digunakan untuk menghilangkan *influence* yang ada. Hal yang perlu diperhatikan dalam membuat algoritma *propagation* adalah *double buffering* (Menggunakan *local store* untuk *propagation* sebelum dikembalikan ke *influence map* sebenarnya) untuk menghindari adanya *influence* yang kurang sesuai.

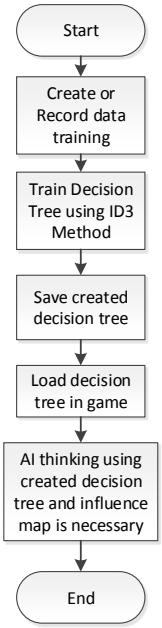
2.2.1. *Propagation parameter*

Ada beberapa parameter yang perlu dilakukan dalam melakukan *propagation*, yang pertama adalah *momentum*, sekuat apakah *bias* (pengaruh) dari suatu *event* terbaru terhadap *influence map* yang ada. Parameter kedua adalah *decay*, semakin tinggi tingkat *decay* semakin cepat juga suatu kejadian dilupakan dari *influence map*, sedangkan semakin rendah tingkat *decay*, *influence* yang ada sejak lama masih akan disimpan pada *incluence map*. Parameter ketiga adalah *update frequency*, yaitu secepat apakah sistem melakukan update *influence map Update* dapat dilakukan setiap beberapa detik/*frame* untuk *game real-time* atau setiap satu *turn* untuk *game turn-based.*

3. DESAIN SISTEM

Sistem yang dibuat terdiri dari 2 bagian, yaitu sistem *ID3* dan sistem *influence map*. Kedua sistem ini diuji coba ke dalam suatu *game turn-based strategy*. Rancangan arsitektur dan proses kerja sistem secara garis besar dapat

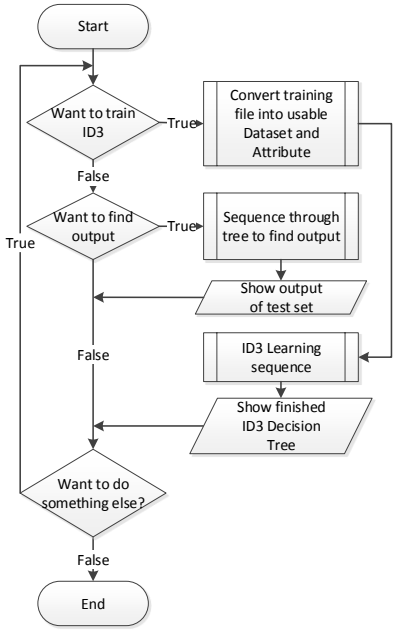
dilihat pada Gambar 3. Pertama *developer* butuh membuat *decision tree* menggunakan metode *ID3* pada program *ID3 Editor* (program yang dibuat penulis untuk melakukan pelatihan *ID3*), hasil dari *decision tree* itu kemudian dipakai di dalam *game*. *Influence map* akan diterapkan di dalam *game* yang digunakan dan dipakai sebagai *input* dari *decision tree* ketika diperlukan. Hasil dari *decision tree* menentukan gerakan yang akan dilakukan oleh *AI* di dalam *game*.



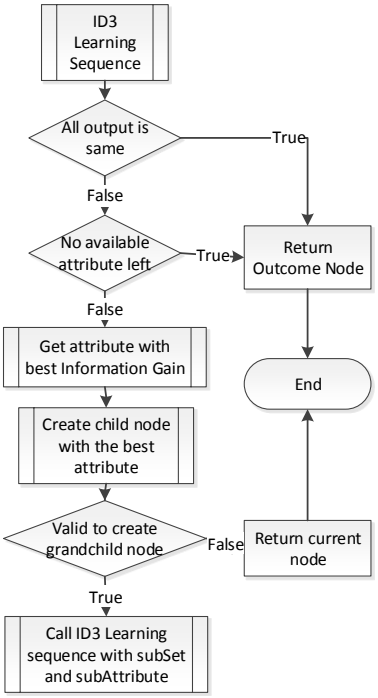
Gambar 3. Arsitektur secara garis besar

Program *ID3 Editor* dibuat menggunakan Visual Studio 2010 dengan bahasa *C#*. Cara kerja program *ID3 Editor* dapat dilihat pada Gambar 4. Untuk membuat *decision tree*, proses dapat dilakukan dengan memasukkan data *training* bertipe *Comma Separated Value (CSV)* ke dalam program. Data berikutnya diproses menjadi data yang dapat dipakai untuk pelatihan *ID3*.

Metode pelatihan *ID3* dilakukan menggunakan fungsi rekursi dengan parameter *dataset* dan atribut yang akan digunakan, fungsi dimulai dengan melakukan pengecekan terhadap *dataset* yang digunakan pada rekursi saat itu. Jika data memiliki nilai *output* yang sama, fungsi dapat melakukan *return node output* (*OutcomeNode*) dengan nilai *output* itu, jika atribut yang dapat digunakan sudah habis, fungsi akan melakukan *return node* output dengan nilai terbanyak *output* dari dataset yang digunakan. Jika bukan keduanya, maka fungsi akan melakukan pencarian atribut dengan nilai *information gain* terbaik, nilai atribut itu kemudian akan dipakai menjadi *node child*. Garis besar cara kerja pelatihan *ID3* dapat dilihat pada Gambar 5.

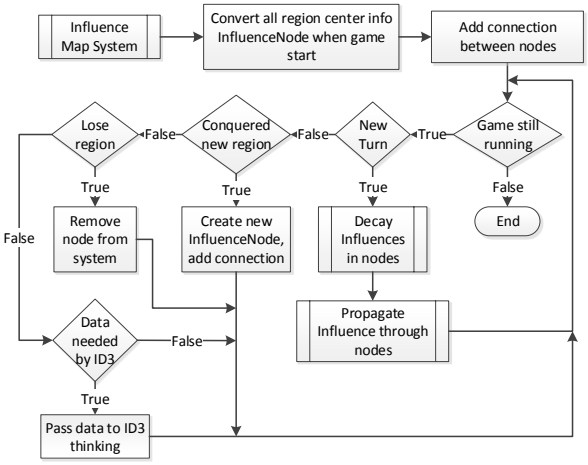


Gambar 4. *Flowchart* cara kerja program *ID3 Editor*



Gambar 5. *Flowchart* cara pelatihan *ID3*

Influence map merupakan sistem yang berbentuk *Class*, oleh karena itu *influence map* dapat langsung disusun di dalam *game* yang digunakan agar dapat langsung disesuaikan dengan data yang digunakan dalam *game*. Di dalam *game* yang digunakan, terdapat "*Region*" yang menandakan suatu daerah, dan setiap *region* akan diberi satu *node*, sehingga sistem yang dibentuk menyerupai *Area Graphs*. Cara kerja *influence map* dapat dilihat pada Gambar 6.



Gambar 6. *Flowchart* cara kerja *influence map*

4. PENGUJIAN

Pengujian terhadap aplikasi dibagi menjadi beberapa bagian di antaranya adalah sebagai berikut:

- Pengujian terhadap pembentukan *decision tree*
  Pengujian ini dilakukan untuk melihat *decision tree* yang dihasilkan oleh aplikasi dengan tipe data yang berbeda.

Tabel 1. Hasil pengujian pembentukan *decision tree*

Tipe *Dataset*	*Dataset training* dan hasil *decision tree*
Data diskret	No. SKY BARO... WIND RAIN 1 CLEAR RISING NORTH FALSE 2 CLOUDY RISING SOUTH TRUE 3 CLOUDY STEADY NORTH TRUE 4 CLEAR FALLING NORTH FALSE 5 CLOUDY FALLING NORTH TRUE 6 CLOUDY RISING NORTH TRUE 7 CLOUDY FALLING SOUTH FALSE 8 CLEAR RISING SOUTH FALSE *Dataset* (?) SKY (:) CLEAR (=) FALSE (:) CLOUDY (?) BAROMETER (:) RISING (=) TRUE (:) STEADY (=) TRUE (:) FALLING (?) WIND (:) NORTH (=) TRUE (:) SOUTH (=) FALSE Hasil *tree*
Data kontinu	No. HP RESULT 1 5.0 RETREAT 2 50.11 ATTACK 3 50.1 ATTACK 4 50.2 ATTACK 5 15.0 DEFEND 6 15.1 DEFEND 7 10.1 RETREAT 8 9.2 RETREAT 9 11.1 RETREAT 10 20.1 DEFEND 11 35.0 ATTACK *Dataset* (?) HP (:) <13.05 (=) RETREAT (:) >=13.05 (?) HP (:) <27.55 (=) DEFEND (:) >=27.55 (=) ATTACK Hasil *tree*
Data campuran (Diskret dan kontinu)	No. OUTLOOK TEMP... HUMIDITY WINDY PLAY 1 SUNNY 85 85 FALSE DON'TPLAY 2 SUNNY 80 90 TRUE DON'TPLAY 3 OVERCAST 83 78 FALSE PLAY 4 RAIN 70 96 FALSE PLAY 5 RAIN 68 80 FALSE PLAY 6 RAIN 65 70 TRUE DON'TPLAY 7 OVERCAST 64 65 TRUE PLAY 8 SUNNY 72 95 FALSE DON'TPLAY 9 SUNNY 69 70 FALSE PLAY 10 RAIN 75 80 FALSE PLAY 11 SUNNY 75 70 TRUE PLAY 12 OVERCAST 72 90 TRUE PLAY 13 OVERCAST 81 75 FALSE PLAY 14 RAIN 71 80 TRUE DON'TPLAY *Dataset* (?) OUTLOOK (:) SUNNY (?) HUMIDITY (:) <77.5 (=) PLAY (:) >=77.5 (=) DON'TPLAY (:) OVERCAST (=) PLAY (:) RAIN (?) WINDY (:) FALSE (=) PLAY (:) TRUE (=) DON'TPLAY Hasil *tree*

Seperti yang dapat dilihat dari Tabel 1, Atribut diskret dan atribut kontinu diperlakukan secara berbeda oleh aplikasi. Atribut diskret akan langsung digunakan oleh aplikasi sebagai *node-node tree*, sedangkan atribut kontinu akan diproses terlebih dahulu untuk menemukan batasan *threshold* sebagai pemisah. Semua *data* yang dimasukkan ke dalam aplikasi *ID3 Editor* akan di*uppercase* untuk menghindari adanya perbedaan huruf besar dan kecil di dalam data *training*. Seperti yang dapat diperhatikan dari hasil *decision tree*, terdapat 3 jenis *node* pada *tree* itu (*IntersectNode* (simbol ?),

AttributeAnswerNode (simbol :), dan *OutcomeNode* (simbol =)). Ketiga *node* ini menentukan tindakan yang diambil ketika mencari *outcome* dari *decision tree* yang sudah dihasilkan.

- Pengujian terhadap Cara Mengatasi Data Kontinu.
  Selain data diskret, *ID3* dapat juga disusun agar dapat menangani data kontinu. Data kontinu dapat menjadi data yang cukup penting dalam berpikir, selain itu dapat membuat *decision tree* yang dihasilkan lebih dinamis (Tidak membutuhkan *developer* menentukan batasan-batasan seperti *high*, *medium*, *low* secara manual). Untuk pengujian ini dicoba dengan perumpamaan 2 pasukan yang sedang bertarung

Tabel 2. Pengujian terhadap cara memroses data kontinu

Dataset	Hasil
Data *power* dimasukkan secara langsung No. | MY POWER | ENEMY POWER | RESULT 1 | 110 | 120 | RETREAT 2 | 105 | 100 | ATTACK 3 | 50 | 150 | RETREAT 4 | 100 | 80 | ATTACK 5 | 10 | 50 | RETREAT 6 | 200 | 40 | ATTACK 7 | 260 | 250 | ATTACK 8 | 150 | 230 | RETREAT 9 | 55 | 50 | ATTACK 10 | 15 | 160 | RETREAT 11 | 215 | 300 | RETREAT 12 | 405 | 10 | ATTACK 13 | 350 | 700 | RETREAT	(?) ENEMY POWER (:) <110 (?) MY POWER (:) <32.5 (=) RETREAT (:) >=32.5 (=) ATTACK (:) >=110 (?) MY POWER (:) <237.5 (=) RETREAT (:) >=237.5 (?) MY POWER (:) <305 (=) ATTACK (:) >=305 (=) RETREAT *My Power* = 250 *Enemy Power* = 150 *Result: Attack* (Benar) --- *My Power* = 150 *Enemy Power* = 120 *Result: Retreat* (Salah)
Data *power* diolah menjadi perbandingan *MyPower:EnemyPower* (Dibagi) No. | MYP... | RESULT 1 | 0.916... | RETREAT 2 | 1.05 | ATTACK 3 | 0.333... | RETREAT 4 | 1.25 | ATTACK 5 | 0.2 | RETREAT 6 | 5 | ATTACK 7 | 1.04 | ATTACK 8 | 0.652... | RETREAT 9 | 1.1 | ATTACK 10 | 0.09375 | RETREAT 11 | 0.716... | RETREAT 12 | 40.5 | ATTACK 13 | 0.5 | RETREAT	(?) MYPOWER:ENEMYPOWER (:) <0.9783333335 (=) RETREAT (:) >=0.9783333335 (=) ATTACK *My Power* = 250 *Enemy Power* = 150 *Result: Attack* (Benar) --- *My Power* = 150 *Enemy Power* = 120 *Result: Attack* (Benar)

Dari hasil pengujian pada Tabel 2, dapat dilihat bahwa ketika data *MyPower* dan *EnemyPower* di*training* secara langsung sebagai atribut terpisah, hasil pemikiran *decision tree* dapat memiliki kesalahan, namun jika data *MyPower* diproses terlebih dahulu dengan cara membandingkannya dengan *EnemyPower*, hasil pemikiran menjadi lebih baik, dan hasil *decision tree* lebih efisien. Jika suatu data kontinu memiliki kaitan antara satu dengan lainnya sebaiknya atribut yang digunakan adalah atribut hasil proses dari atribut yang saling berhubungan. Jika suatu data yang sebenarnya butuh diproses secara berhubungan diproses secara sendiri-sendiri, hasil dari *decision tree* mungkin tidak memuaskan atau bahkan memiliki kesalahan.

- Pengujian terhadap *Dataset* dengan Pola Tertentu
  Pengujian yang akan dilakukan kali ini adalah menguji *decision tree* yang dihasilkan oleh aplikasi jika data *training* yang dimasukkan memiliki pola tertentu. Pola yang diuji adalah *dataset* dengan nilai *result/output/outcome* yang sama semua dan *dataset* yang nilai *result*nya hanya memanfaatkan satu atribut saja.

Tabel 3. Pengujian pada *dataset* berpola tertentu

Dataset	Hasil *decision tree*
Nilai *result* kembar semuanya No. | CLOUD | SUN | WIND | RESULT 1 | LITTLE | SEEN | WEAK | RAIN 2 | MID | SEEN | STRONG | RAIN 3 | MUCH | HIDING | MID | RAIN 4 | LITTLE | HIDING | WEAK | RAIN 5 | LITTLE | HIDING | STRONG | RAIN 6 | MUCH | SEEN | MID | RAIN	(=) RAIN
Result bergantung satu atribut No. | CLOUD | SUN | WIND | RESULT 1 | LITTLE | SEEN | WEAK | NORAIN 2 | MID | SEEN | STRONG | MAYBE 3 | MUCH | HIDING | MID | RAIN 4 | LITTLE | HIDING | WEAK | NORAIN 5 | LITTLE | HIDING | STRONG | NORAIN 6 | MUCH | SEEN | MID | RAIN	(?) CLOUD (:) LITTLE (=) NORAIN (:) MID (=) MAYBE (:) MUCH (=) RAIN
Dataset dengan value berkonflik (Tanpa Pola) No. | RAIN | SUN | RESULT 1 | MANY | SEEN | FLOOD 2 | MID | HIDING | NOFLOOD 3 | LOW | HALFSEEN | MAYBE 4 | MANY | HALFSEEN | NOFLOOD 5 | MID | SEEN | FLOOD 6 | LOW | SEEN | NOFLOOD 7 | MANY | HIDING | MAYBE 8 | MID | HALFSEEN | FLOOD 9 | LOW | HIDING | FLOOD	(?) RAIN (:) MANY (?) SUN (:) SEEN (=) FLOOD (:) HIDING (=) MAYBE (:) HALFSEEN (=) NOFLOOD (:) MID (?) SUN (:) SEEN (=) FLOOD (:) HIDING (=) NOFLOOD (:) HALFSEEN (=) FLOOD (:) LOW (?) SUN (:) SEEN (=) NOFLOOD (:) HIDING (=) FLOOD (:) HALFSEEN (=) MAYBE

Pengujian dari Tabel 3 menunjukkan bahwa *ID3* akan berusaha dalam membentuk *decision tree* yang optimal dan membuang atribut yang tidak penting. Hasil *decision tree* akan sangat bergantung dengan *dataset* yang digunakan, jika suatu data berkonflik digunakan, hasil *decision tree* juga dapat memiliki masalah.

- Pengujian terhadap jarak hubungan antar node *Influence Map*
  Untuk pengujian *Influence Map*, digunakan suatu *game turn-based strategy* dengan peta indonesia berfokus di daerah jawa. Game ini bersistem *tile* dengan *region-*

region yang terdiri dari *tile-tile*. *Influence map* yang dipakai akan diletakkan untuk setiap *region* yang ada di dalam *game*. Saat ini dilakukan pengujian terhadap hasil koneksi antar *node* dengan nilai jarak maksimum yang berbeda-beda. Jarak yang disebutkan melambangkan jarak *tile*.

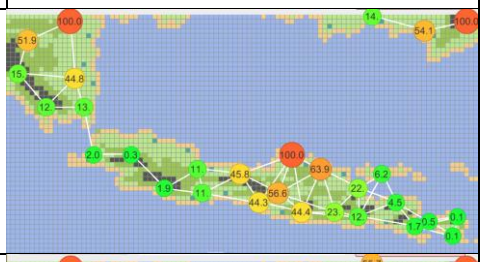
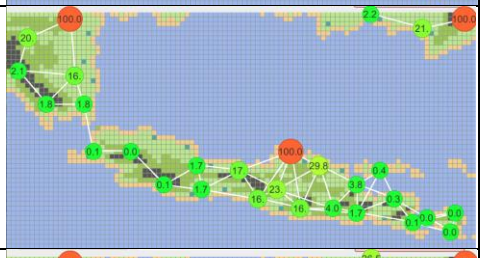
Tabel 4. Pengujian terhadap jarak hubungan antar *node*

Jarak maksimum hubungan antar *node*	Hasil konesi antar *node* pada *Influence Map*
Jarak maksimum 20	
Jarak maksimum 15	
Jarak maksimum 10	

Hasil pengujian pada Tabel 4 menunjukkan bahwa jika jarak terlalu besar, hubungan antar *node* akan menjadi terlalu banyak dan mengakibatkan sistem *propagation* menjadi kurang sesuai. Jika jarak terlalu kecil, terdapat *node* yang tidak memiliki satupun koneksi ke *node* lain, mengakibatkan *node* itu terisolasi, sehingga tidak akan terpengaruhi *influence* ketika dilakukan *propagation*.

- Pengujian terhadap Metode Perhitungan *Propagation*
  Metode yang diuji untuk *propagation* adalah rumus perhitungan nilai *influence* yang disebarkan dari satu *node* ke *node* lain. Metode perhitungan yang digunakan adalah metode eksponensial dan metode linear dengan nilai berbeda-beda. Pada pengujian kali ini sumber *influence* diletakkan pada 3 *node* (dapat dilihat dari 3 *node* dengan *nilai* 100).

Tabel 5. Pengujian terhadap perhitungan yang digunakan untuk *propagation*

Rumus hitung *influence*	Hasil konesi antar *node* pada *Influence Map*
Eksponen Value/ Exp(jarak /15)	
Eksponen Value/ Exp(jarak /50)	
Linier Value/ Max ((jarak/2), 1)	
Linier Value/ Max ((jarak/7), 1)	

Dari hasil pengujian pada Tabel 5 dapat dilihat bahwa pada rumus exponensial, *node* yang berdekatan akan memiliki perbedaan nilai yang cukup tinggi, dan perbedaan itu menjadi semakin kecil dengan semakin jauhnya jarak *node*. Rumus ini dapat membantu jika *developer* ingin tetap mempertahankan adanya penyebaran *influence* sampai pada *node* yang berjauhan. Sedangkan untuk rumus linier, perbedaan *influence* berbanding setara pada *node* yang dekat maupun *node* yang jauh, sehingga meskipun nilai *influence* bernilai besar pada *node* yang besar, pada *node* yang berjauhan sedikit saja nilai *influence* itu mudah menghilang.

5. KESIMPULAN

Berdasarkan hasil pengujian didapatkan beberapa kesimpulan yaitu:

- Meskipun suatu data memiliki banyak atribut, *ID3* dapat membentuk *decision tree* menggunakan atribut yang benar-benar berpengaruh dan menghilangkan

atribut yang tidak memiliki pengaruh pada data yang digunakan.

- *Decision tree* yang dibentuk oleh *ID3* sangat bergantung dengan *data training* yang digunakan. Jika suatu data *ID3* memiliki pola yang baik, pelatihan *tree* akan menghasilkan *decision tree* yang terbentuk dengan baik. Namun jika data *training* yang digunakan memiliki konflik, hasil *decision tree* dapat memiliki konflik juga
- Jika data kontinu terpecah menjadi lebih dari 2 *output*, pemotongan bertingkat dapat digunakan oleh *ID3*. Jika 2 atau lebih data kontinu memiliki hubungan, sebaiknya diproses terlebih dahulu sebelum dimasukkan untuk proses pelatihan, jika tidak *decision tree* yang dihasilkan mungkin kurang sesuai dengan keinginan.
- *Influence map* dapat digunakan sebagai input dari *decision tree* yang sudah dihasilkan oleh *ID3*. *Influence map* dapat memberikan kesadaran akan kondisi sekitar pada *AI*. Jika cara pembentukan *Influence Map* pada *game* dilakukan dengan cara *area graph* yang dihubungkan dengan jarak tertentu, sebaiknya batas jarak terjauh koneksi antar *node* tidak terlalu besar, karena dapat menyebabkan koneksi *node* satu dengan *node* lainnya terlalu rumit (tidak efektif)
- Jika ingin suatu *influence* dapat menyebar hingga ke daerah yang berjauhan letaknya, metode *propagation* menggunakan rumus eksponensial dapat digunakan, sedangkan jika ingin penyebaran terjadi pada proporsi yang sama baik dekat ataupun jauh, rumus linier dapat digunakan.

6. REFERENSI

[1] Bahety, A. *Extension and Evaluation of ID3 – Decision Tree Algorithm*. Retreived May 26, 2014, from https://www.cs.umd.edu/sites/default/files/scholarly_papers/Bahety_1.pdf

[2] Baumgarten, R., Colton, S., Morris, M. 2009. *Combining AI Methods for Learning Bots in a Real-Time Strategy Game*. London: Imperial College London

[3] Champandard, A. 2011. *The Mechanics of Influence Mapping: Representation, Algorithm & Parameters*. Retreived May 26, 2014, from http://aigamedev.com/open/tutorial/influence-map-mechanics

[4] Millington, I., Funge, J. 2009. *Artificial Intelligence for Games Second Edition*. USA: Morgan Kaufmann Publishers

[5] Moller, M. W., Thykier, N. 2010. *Turn-based Game Engine and AI*. Denmark: Technical University of Denmark

[6] Peng, W., Chen, J., Zhou, H. *An Implementation of ID3 - Decision Tree Learning Algorithm*. Retreived May 26, 2014, from http://cis.k.hosei.ac.jp/~rhuang/Miccl/AI-2/L10-src/DecisionTree2.pdf

[7] Slocum, M. 2012. *Decision Making Using ID3 Algorithm*. USA: Rivier University