

Generatie van spelinhoud d.m.v. probabilistisch programmeren

KU LEUVEN

Robin Haveneers

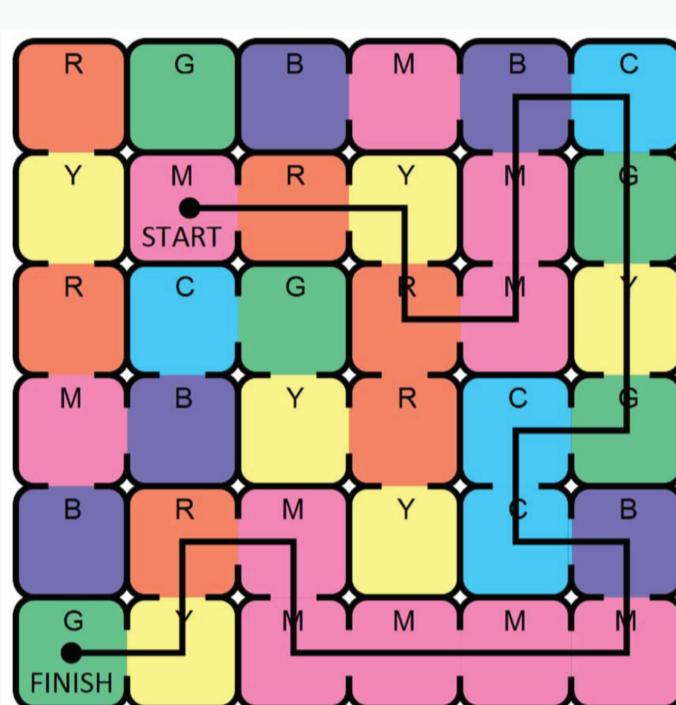
Dr. A. Kimmig & Prof. L. De Raedt

I. Introductie

Situering

- Generatie van spelinhoud

- Plattegronden (kerkers, doolhoven ...)
- Rekwisieten (diamantjes, altaars, wapens ...)



```
(() == () == () == () == ()  
|| || || || || ||  
(() == () == () == () == ()  
|| || || || || ||  
(() == () == () == () == ()  
|| || || || || ||  
(() == () == () == () == ()  
|| || || || || ||  
(() == () == () == () == ()  
|| || || || || ||  
(() == () == () == () == ()  
|| || || || || ||  
(() == () == () == () == ()  
|| || || || || ||  
(() == () == () == () == ()  
|| || || || || ||  
FINISH
```

- ProbLog

- Probabilistische programmeertaal
- Uitbreiding op ProLog
- Waarheden kansen geven

- Spelinhouder en probabilistisch programmeren

- AnsProlog (en anderen)
- Beperkt in probabilitet

Hypothese

- ProbLog biedt minstens evenveel

- Niet-uniforme verdelingen
- Eenvoudigere representaties van bepaalde constraints

II. Probleem & Motivatie

Probleem

- ?
- Systeem

- Is ProbLog inzetbaar voor genereren van spelinhouder?
- Zijn de uitvoeringstijden voorspelbaar en beperkt?

- ?
- Syntax

- Predicaten eenvoudig?

Motivatie

- Onderzoek naar efficiëntie van ProbLog systeem.
- Invloed van andere kansverdeling op uitkomst

V. Toekomst & Verder Onderzoek

Implementatie

- Puzzels nog meer benchmarken

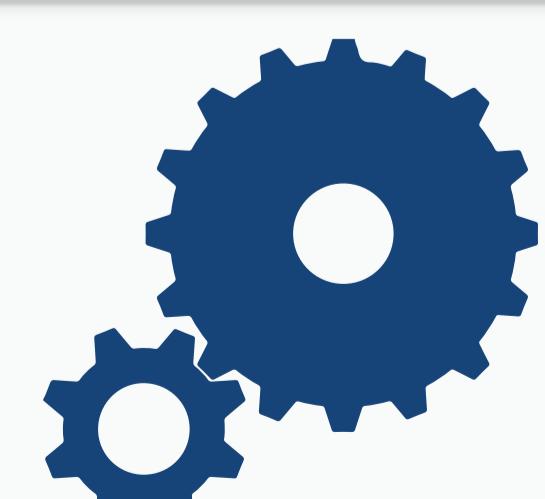
- Wat met nog hogere dimensies?
- Code optimaliseren?
- Zoeken waar het 'mis' gaat



Meer voorbeelden

- Omzetting van nog meer puzzels

- Levels met nog meer constraints
- Complexere spellen
- Procedurale generatie (mogelijk?)



Referenties

III. Aanpak

Syntactische omzetting

- Bestaande code vertalen
- Geen algoritmische wijzigingen
 - Hier en daar een optimalisatie
- Kansverdelingen en integriteitsvoorwaarden interessant om nader te bekijken

Kansverdelingen

AnsProlog: `min { predicaat } max`

ProbLog: `select_uniform` (voor min=1, max=1)



Integriteitsvoorwaarden

AnsProlog: `:- not oplosbaar`

ProbLog: `evidence(oplosbaar)`

Voorbeeld

```
1 {start(X,Y):dim(X):dim(Y)} 1.  
:- not victory
```

```
start_and_finish((A,B),(C,D)) :-  
pairs(P),  
select_uniform(I,P,(A,B),R).  
  
victory :- victory_at(T), time(T)
```

* Hierbij is P de lijst van alle mogelijke startcoördinaten

Systeem analyseren

- Puzzels genereren (chromatic maze, perfect maze)
- Uitvoeringstijd analyseren
 - Verschillende dimensies
- Grafische weergave om correctheid te controleren

```
(() == () == () == () == ()  
|| || || || ||  
(() == () == () == () == ()  
|| || || || ||  
(() == () == () == () == ()  
|| || || || ||  
(() == () == () == () == ()  
|| || || || ||  
()
```

f	4	1	8	1
0	3	2	1	6
9	4	3	4	5
8	5	4	7	6
7	6	5	6	7

IV. Conclusies

- Sampling methode is onvoorspelbaar en traag.

- Gevolg: genereren van puzzels vaak langzaam.

- Voor grotere dimensies stijgt de tijd nagenoeg exponentieel

- Syntactisch: predicaten eenvoudig voorstellen is geen probleem.

- Grafiek hiernaast is voor perfect mazes

- Dimensie 5: per puzzel min. 400 seconden

- Bij chromatic mazes: lange uitvoeringstijd bij

- Hoge dimensies
- Voorwaarden op minimale oploslengte

