Dynamic Scripting

Knowledge Engineering und Lernen in Spielen

Thomas Kellner

DS: Überblick

- Einleitung
- Scripte
- Adaptive Spiele KI
- Technische Anforderungen
- Funktionelle Anforderungen
- Beschreibung
- Ziele
- Illustration
- Algorithmus
- Minigate
- Weight-update function
- Measuring performance
- Eindeutige Überlegenheit?
- Difficulty Scaling
- Kommerzielle Anwendung?
- Fazit
- Quellen

DS: Einleitung

- Komplexe Spiele – Komplexe KI
  - Schach
  - Strategiespiele
  - CRPG‘s
- Scripte bevorzugt

DS: Scripte

- Vorteile:
  - Verständlich
  - Vorhersagbar
  - Anpassbar
  - Implementierbarkeit
  - Erweiterbarkeit
- Nachteil
  - Allgemein statisch
  - Lang und komplex
    - Schwachstellen
    - Keine Lösung bei unvorhersehbaren Taktik Verhalten
    - Skalierbarkeit

DS: Adaptive Spiele KI

- Anwendung von Online Machine Learning
- Entwicklung der neuen Machine Learning Technik „Dynamic Scripting“
  - Verbessert Gescriptete Spiele KI‘s um Fähigkeit des Online Learnings
    - Besonders in komplexen Spielen
      - (Wird anhand des Spiel‘s Mini Gate und Neverwinter Nights verdeutlicht)

DS: Adaptive Spiele KI

- Adaptive Spiele KI
  - „…Spiele KI die sich die sich erfolgreich an wechselnde Umstände anpaßt…“
- KI durch OML adaptiv
  - Menschen-gesteuert
  - Computer-gesteuert

DS: Adaptive Spiele KI

- Menschen-gesteuertes OL
  - Implementiert Veränderungen in der KI durch sofortiges Feedback des Spielers
  - Jede Entscheidung wird positiv oder negativ bewertet
    - Sporadisch in wenigen Spielen verwendet
      - Aber DS „will“ automatische Adaption der KI

DS: Adaptive Spiele KI

- Computer-gesteuert OL
  - Implementiert automatische Veränderungen auf die KI durch ständige Beobachtung der KI und deren direkte Effekte im Spiel
  - Von Entwicklern gemieden
    - Befürchten zu unterlegene KI
      - Erlauben nur wenige veränderbarer Parameter
    - Bisherige Versuche in Spielen
      - FEAR, Far Cry, etc.

DS: Adaptive Spiele KI

- Nachforschungen ergaben:
  - Vier computertechnische Anforderungen
    - Müssen erfüllt sein, sonst nutzlos in Anwendung
  - Vier funktionelle Anforderungen
    - Für Entwickler wichtig
    - Wenn nicht erfüllt oder Fehler in der KI
      - Nutzlos trotz technisch optimaler Funktionalität

DS: Technische Anforderungen

- Geschwindigkeit
  - Adaptive KI muss schnell sein, da lernen während des Spielens stattfindet
- Effektivität
  - Adaptive KI muss effektiv sein während des Lernprozesses um zu verhindern das sie einer normalen KI unterliegen würde
    - KI erzeugt verständliches, erfolgreiches Verhalten

DS: Technische Anforderungen

- Robustheit
  - Adaptive KI muss robust sein gegenüber Zufälligkeiten in den meisten Spielen
- Effizienz
  - Adaptive KI muss effizient lernen um bei möglichst wenigen Aufeinandertreffen schnell Erfolgreich zu sein
    - Menschen müssen auch schnell in wenigen Situationen lernen

DS: Funktionelle Anforderungen

- Klarheit
  - Adaptive KI muss einfache, interpretierbare Resultate liefern
- Mannigfaltigkeit
  - Adaptive KI muss viele, verschiedene Verhaltensweisen besitzen
    - Agenten mit festen Verhalten weniger unterhaltsam als solche mit unvorhersehbarem Verhalten

DS: Funktionelle Anforderungen

- Beständigkeit
  - Durchschnittsanzahl der Lernmöglichkeiten damit adaptive KI erfolgreiche Resultate liefert sollte sehr beständig, nur leicht variiert, sein
    - Unabhängig von menschlichem Verhalten und von zufälligen Fluktuationen im Lernprozess
- Skalierbarkeit
  - Anpassung an Fähigkeiten des Spielers

DS: Beschreibung

- OML-Technik für Spiele KI
  - Besser gesagt: Stochastische Optimisation
- DS besitzt mehrere Rulebases
  - Eine für jede Agenten Klasse im Spiel
- Wenn neue Instanz eines Agenten erstellt wird
  - Neues Skript wird erstellt
  - Welche Regeln werden genommen?
    - Abhängig von einem Gewichtwert

DS: Ziele

- Adaptieren jener Gewichte in den Rulebases
  - Erwartete Tauglichkeit des Verhaltens schnell steigern
    - Selbst in wechselnden Umgebungen
- Tauglichkeit = Wahrscheinlichkeit zu gewinnen
- Gewichtsveränderung durch Gewichts-Update Funktion

DS: Illustration

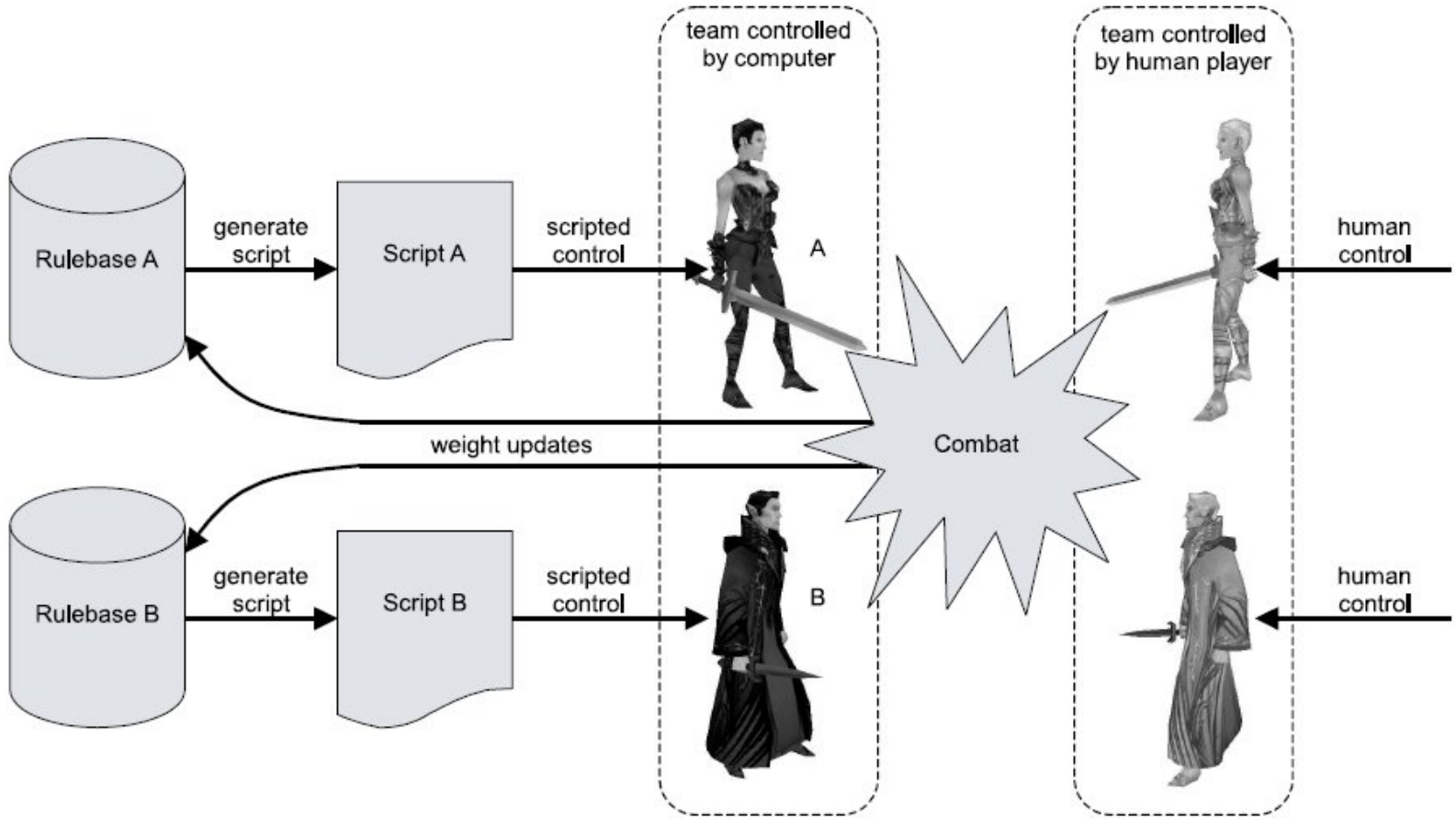


Figure 1. Dynamic scripting.

DS: Algorithmus

Algorithm 1 Script Generation

```
1:  ClearScript()
2:  sumweights = 0
3:  for i = 0 to rulecount − 1 do
4:     sumweights = sumweights + rule[i].weight
5:  end for
6:  {Repeated roulette wheel selection}
7:  for i = 0 to scriptsize − 1 do
8:     try = 0; lineadded = false
9:     while try < maxtries and not lineadded do
10:       j = 0; sum = 0; selected = −1
11:       fraction = random(sumweights)
12:       while selected < 0 do
13:          sum = sum + rule[j].weight
14:          if sum > fraction then
15:             selected = j
16:          else
17:             j = j + 1
18:          end if
19:       end while
20:       lineadded = InsertInScript(rule[selected].line)
21:       try = try + 1
22:    end while
23: end for
24: FinishScript()
```

DS: Algorithmus (2)

Algorithm 2 Weight Adjustment

```
1:  active = 0
2:  for i = 0 to rulecount − 1 do
3:     if rule[i].activated then
4:        active = active + 1
5:     end if
6:  end for
7:  if active <= 0 or active >= rulecount then
8:     return {no updates are needed.}
9:  end if
10: nonactive = rulecount − active
11: adjustment = CalculateAdjustment(Fitness)
12: compensation = −active ∗ adjustment/nonactive
13: remainder = 0
14: {Credit assignment}
15: for i = 0 to rulecount − 1 do
16:    if rule[i].activated then
17:       rule[i].weight = rule[i].weight + adjustment
18:    else
19:       rule[i].weight = rule[i].weight + compensation
20:    end if
21:    if rule[i].weight < minweight then
22:       remainder = remainder + (rule[i].weight − minweight)
23:       rule[i].weight = minweight
24:    else if rule[i].weight > maxweight then
25:       remainder = remainder + (rule[i].weight − maxweight)
26:       rule[i].weight = maxweight
27:    end if
28: end for
29: DistributeRemainder();
```

DS: Algorithmus Resultat

- Erfüllt DS Anforderungen an adaptive KI?
  - Fünf der Acht
    - Geschwindigkeit
      - Nur Regelextraktion und Gewicht Update pro Kampf
    - Effektivität
      - Regeln in Rulebase müssen für jeden Agenten „sinnvoll“ sein
    - Robustheit
      - Zufällige Ereignisse nur kurzfristige bis keine Auswirkungen
        - Abhängig von Gewichtsparametern
    - Klarheit
      - Generierte Scripts verständlich
    - Mannigfaltigkeit
      - Jeder Agent kriegt jeweils eigenes neues Skript

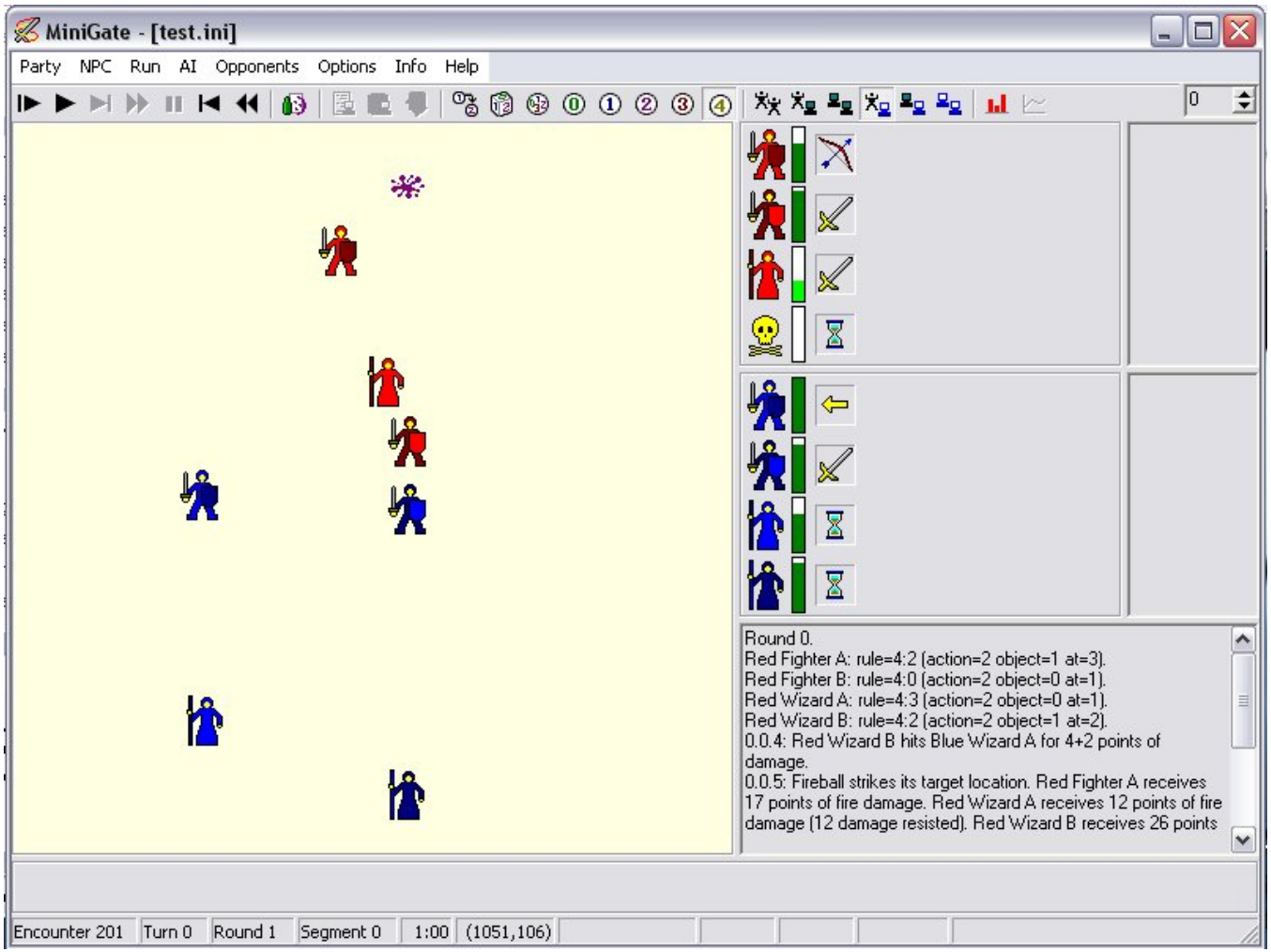
DS: Algorithmus Resultat

- Was ist mit den fehlenden drei?
  - Durch gute Wahl der Gewichtparameter
    - Effizienz und Beständigkeit erfüllt
  - Verbesserungen der DS Technik
    - Skalierbarkeit erfüllt

DS: Minigate

- Test dieser Technik
  - Analyse durch Partien gegen Menschen?
    - Zu langwierig und aufwendig
  - Entwicklung eines CRPG‘s
  - Mingate
    - KI verwendet starke Taktiken
      - DS sollte sich an alle Varianten anpassen und lernen können

DS: Minigate


MiniGate - [test.ini]
Party NPC Run AI Opponents Options Info Help
Round 0.
Red Fighter A: rule=4:2 (action=2 object=1 at=3).
Red Fighter B: rule=4:0 (action=2 object=0 at=1).
Red Wizard A: rule=4:3 (action=2 object=0 at=1).
Red Wizard B: rule=4:2 (action=2 object=1 at=2).
0.0.4: Red Wizard B hits Blue Wizard A for 4+2 points of damage.
0.0.5: Fireball strikes its target location. Red Fighter A receives 17 points of fire damage. Red Wizard A receives 12 points of fire damage (12 damage resisted). Red Wizard B receives 26 points
Encounter 201
Turn 0
Round 1
Segment 0
1:00
(1051,106)

DS: Minigate

- Auswahl der Sprüche und Tränke
  - Skript „entscheidet“ / wird gescannt
    - Regel die Trank / Spruch braucht wird gefunden
      - Füge dem Agenten hinzu (wenn er besitzen darf)
- Rundenbasierendes RPG
  - Jeder Agent bekommt Aktion zugewiesen
    - Je nach Schnelligkeit sequentielle Ausführung

DS: Minigate

- Skriptausführung sequentiell
  - Reihenfolge bei Skripterstellung durch manuell zugewiesene Prioritätswerte
  - Wenn gleiche Priorität?
    - Höheres Gewicht
  - Wenn beides gleich?
    - Random

DS: Minigate

- Skriptgröße
  - Krieger
    - Fünf Regeln aus Regelmenge von Zwanzig
  - Magier
    - Zehn Regeln aus Regelmenge von Fünfzig

DS: Weight-update function

- Team-fitness Function

$$F(g) = \sum_{c \in g} \begin{cases} 0 & \{g \text{ lost}\} \\ \frac{1}{2N_g}\left(1 + \frac{h_T(c)}{h_0(c)}\right) & \{g \text{ won}\} \end{cases}$$

DS: Weight-update function (2)

- Agent-fitness Function

$$F(a,g) = \frac{1}{10}\Big(3F(g) + 3A(a) + 2B(g) + 2C(g)\Big)$$

DS: Weight-update function (3)

- Komponenten

$$A(a) = \frac{1}{3} \begin{cases} \min\left(\frac{D(a)}{D_{max}}, 1\right) & \{h_T(a) \leq 0\} \\ 2 + \frac{h_T(a)}{h_0(a)} & \{h_T(a) > 0\} \end{cases}$$

$$B(g) = \frac{1}{2N_g} \sum_{c \in g} \begin{cases} 0 & \{h_T(c) \leq 0\} \\ 1 + \frac{h_T(c)}{h_0(c)} & \{h_T(c) > 0\} \end{cases}$$

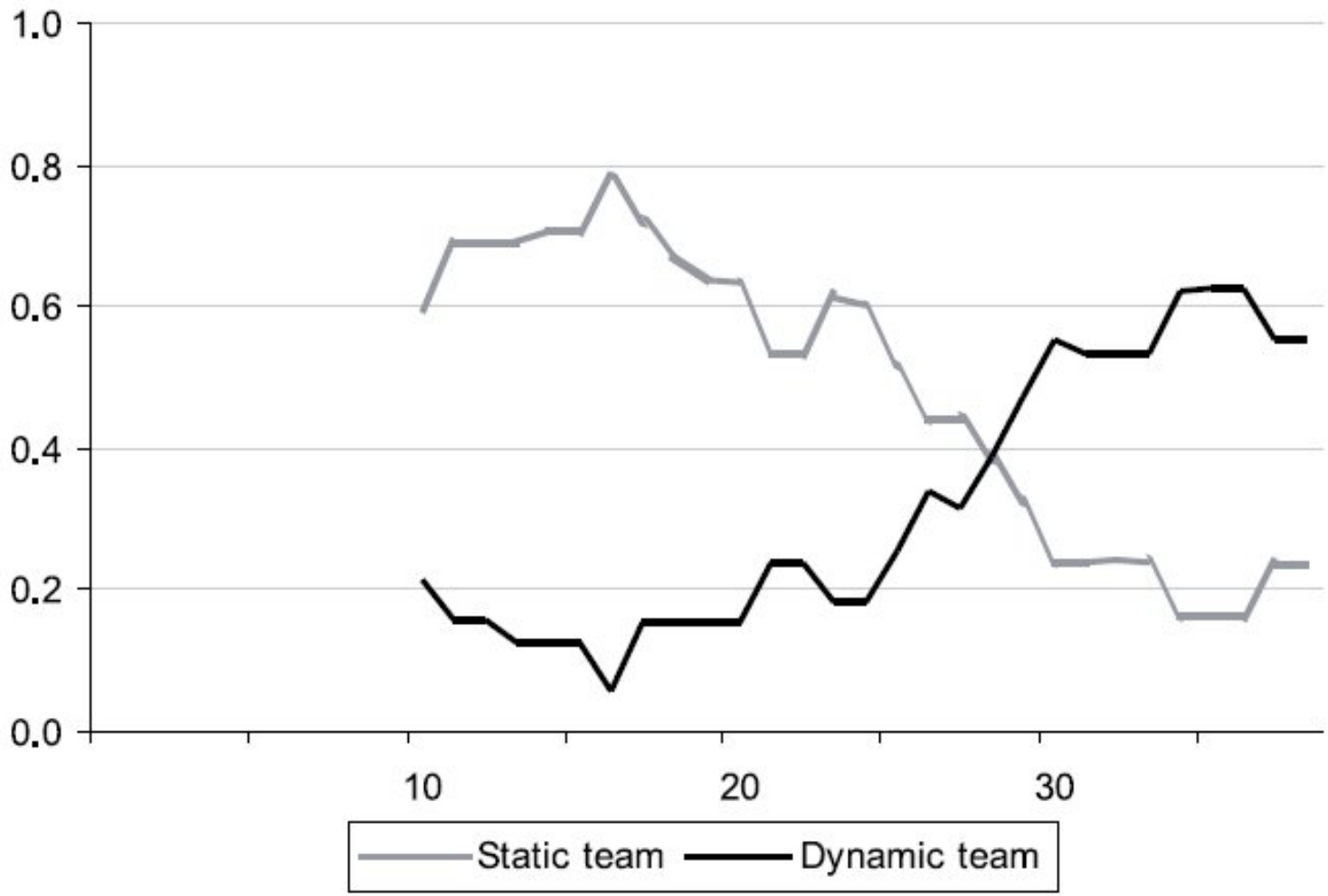
$$C(g) = \frac{1}{2N_{\neg g}} \sum_{c \notin g} \begin{cases} 1 & \{h_T(c) \leq 0\} \\ 1 - \frac{h_T(c)}{h_0(c)} & \{h_T(c) > 0\} \end{cases}$$

DS: Weight-update function (4)

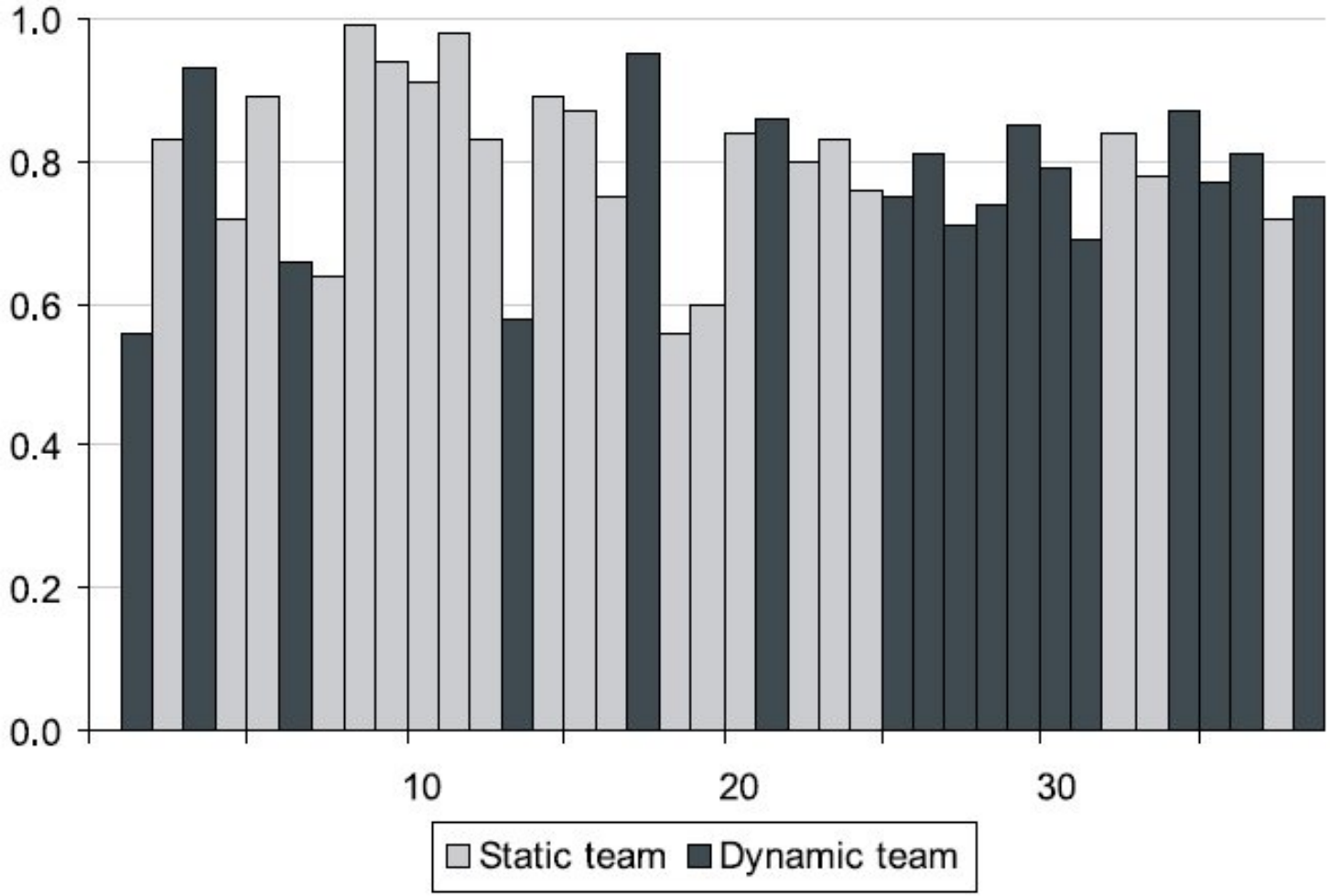
- Calculate Adjustment

$$\triangle W = \begin{cases} -\lfloor P_{max} \dfrac{b - F}{b} \rfloor & \{F < b\} \\ \lfloor R_{max} \dfrac{F - b}{1 - b} \rfloor & \{F \geq b\} \end{cases}$$

DS: Measuring Performance



DS: Measuring Performance(2)



DS: Eindeutige Überlegenheit?

- Bestes Team gewinnt nicht immer (und umgekehrt)
  - Z. B. ständige kritische Treffer oder Misserfolge bei Angriff
    - DS vielleicht zu früh überlegen?
    - Besonders bei vorgewichteten Rulebases
      - Test mit nicht lernendem Dynamic Scripting
        - Alle Gewichte gleich
        - Kein verändern der Gewichte
        - Dadurch zufällige Regelsuche

DS: Eindeutige Überlegenheit?

- Non-learning Dynamic Scripting

Table I. Baseline performance against seven different tactics.

Tactic	TP > 241	Avg.TP	St.dev.	Med.TP	Wins	St.dev.
Offensive	100%	> 241	> 0.0	> 241	21	4.1
Disabling	0%	15	7.2	10	67	5.3
Cursing	77%	> 217	> 62.9	> 241	32	4.3
Defensive	33%	> 134	> 89.3	99	36	5.0
Random team	23%	> 125	> 81.0	98	39	4.7
Random agent	7%	> 87	> 64.5	60	42	5.6
Consecutive	53%	> 191	> 65.9	> 241	33	4.7

DS: Eindeutige Überlegenheit?

- Standard Performance Dynamic Scripting

Table II. Turning points against seven different tactics, averaged over 100 tests.

Tactic	Average	St.dev.	Median	Highest	Top 5
Offensive	53	24.8	52	120	107
Disabling	13	8.4	10	79	39
Cursing	44	50.4	26	304	222
Defensive	24	15.3	17	79	67
Random team	51	64.5	29	480	271
Random agent	41	40.7	25	251	178
Consecutive	52	56.2	37	393	238

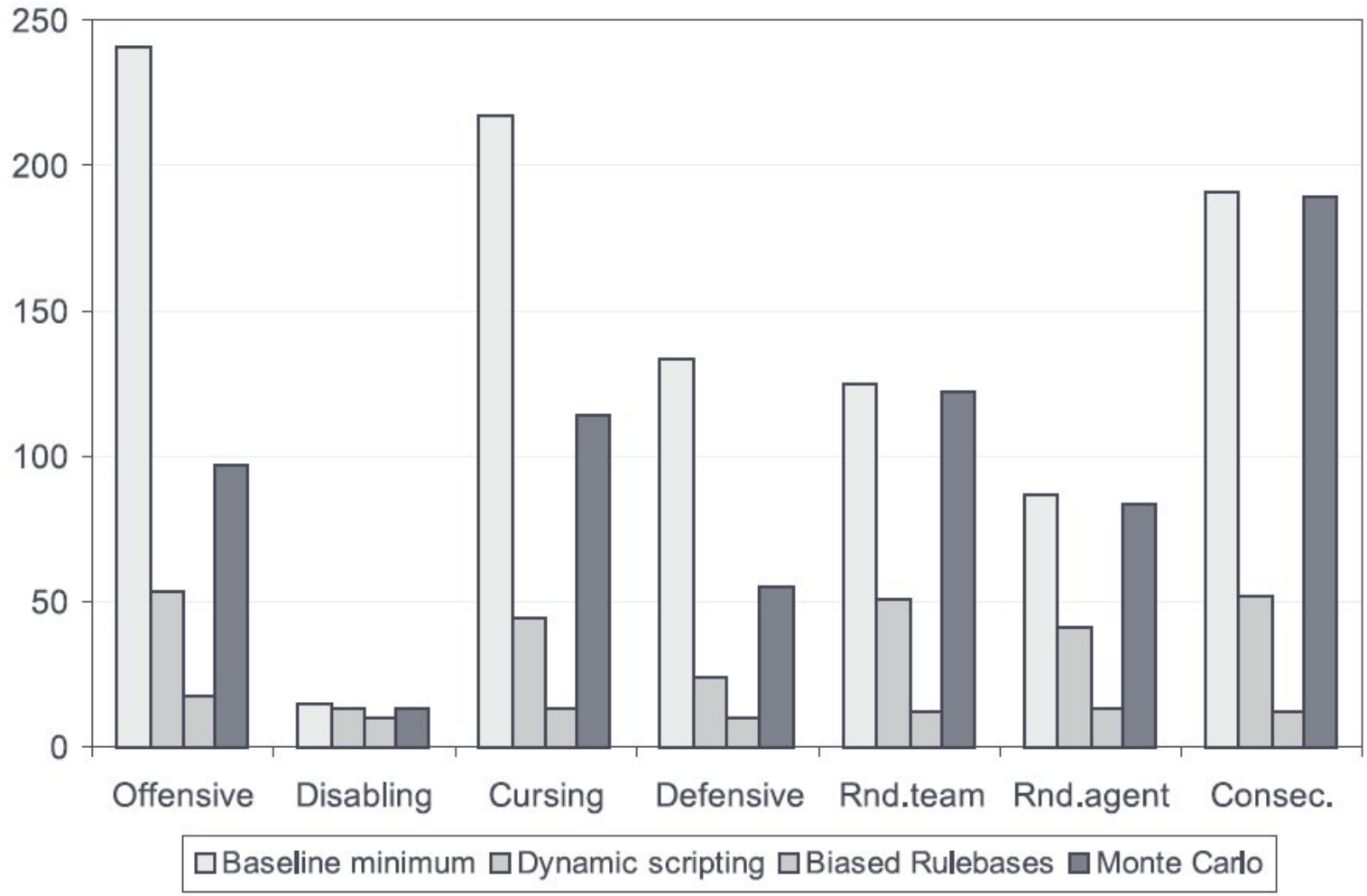
DS: Eindeutige Überlegenheit?

- Biased Version

Table III. Turning points with biased rulebases, averaged over 100 tests.

Tactic	Average	St.dev.	Median	Highest	Top 5
Offensive	17	8.6	15	71	42
Disabling	10	0.0	10	10	10
Cursing	13	5.3	10	40	30
Defensive	10	1.0	10	18	13
Random team	12	5.6	10	38	32
Random agent	13	5.8	10	44	31
Consecutive	12	4.6	10	28	26

DS: Eindeutige Überlegenheit?



DS: Difficulty Scaling

- KI sollte herausfordernd, aber besiegbar sein um Spaß zu machen
- Computer sollte keine absichtlichen, schwachen Züge machen sondern generelle „nicht-so-starke“ Züge.
  - Difficulty Scaling
    - Automatische Adaption an Spieler
      - Gleiche Spielstärke

DS: Difficulty Scaling

- Schwierigkeitsgrade
  - Grob
    - Nur grobe Einstellungen möglich (Easy, Normal, Hard, etc.)
  - Spielerentscheidung
    - „Passt die Einstellung zu mir?“
  - Begrenzte Bereiche
    - Verändert nicht KI sondern Stärke der Computer Einheiten

DS: Difficulty Scaling

- Erweiterung von Dynamic Scripting
  - Skalierbarkeit
    - High Fitness Penalising
    - Weight Clipping
    - Top Culling

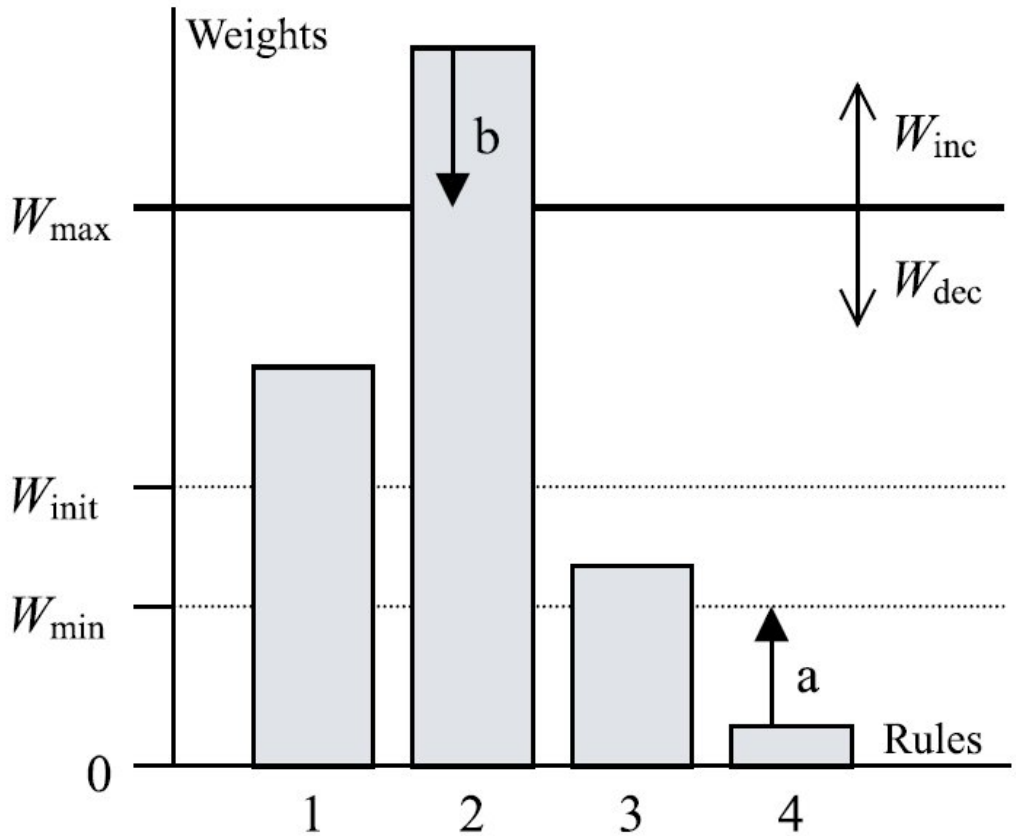
DS: Difficulty Scaling

- High Fitness Penalising
  - Fitness Function Erweiterung

$$F' = \begin{cases} \dfrac{F}{p} & \{F \leq p\} \\ \dfrac{1-F}{p} & \{F > p\} \end{cases}$$

DS: Difficulty Scaling

- Weight Clipping / Top Culling



DS: Difficulty Scaling Ergebnisse

Table V. Difficulty-scaling results, averaged over 100 tests.

Tactic	Plain		High-fitness Penalising		Weight Clipping		Top Culling	
	Avg.	Dev.	Avg.	Dev.	Avg.	Dev.	Avg.	Dev.
Offensive	61.2	16.4	**46.0**	15.1	**50.6**	9.4	**46.3**	7.5
Disabling	86.3	10.4	56.6	8.8	67.8	4.5	**52.2**	3.9
Cursing	56.2	11.7	42.8	9.9	**48.4**	6.9	**46.4**	5.6
Defensive	66.1	11.9	39.7	8.2	**52.7**	4.2	**49.2**	3.6
Novice	75.1	13.3	**54.2**	13.3	**53.0**	5.4	**49.8**	3.4
Random team	55.8	11.3	37.7	6.5	**50.0**	6.9	**47.4**	5.1
Random agent	58.8	9.7	44.0	8.6	**51.8**	5.9	**48.8**	4.1
Consecutive	**51.1**	11.8	34.4	8.8	**48.7**	7.7	**45.0**	7.3

DS: Difficulty Scaling Ergebnisse

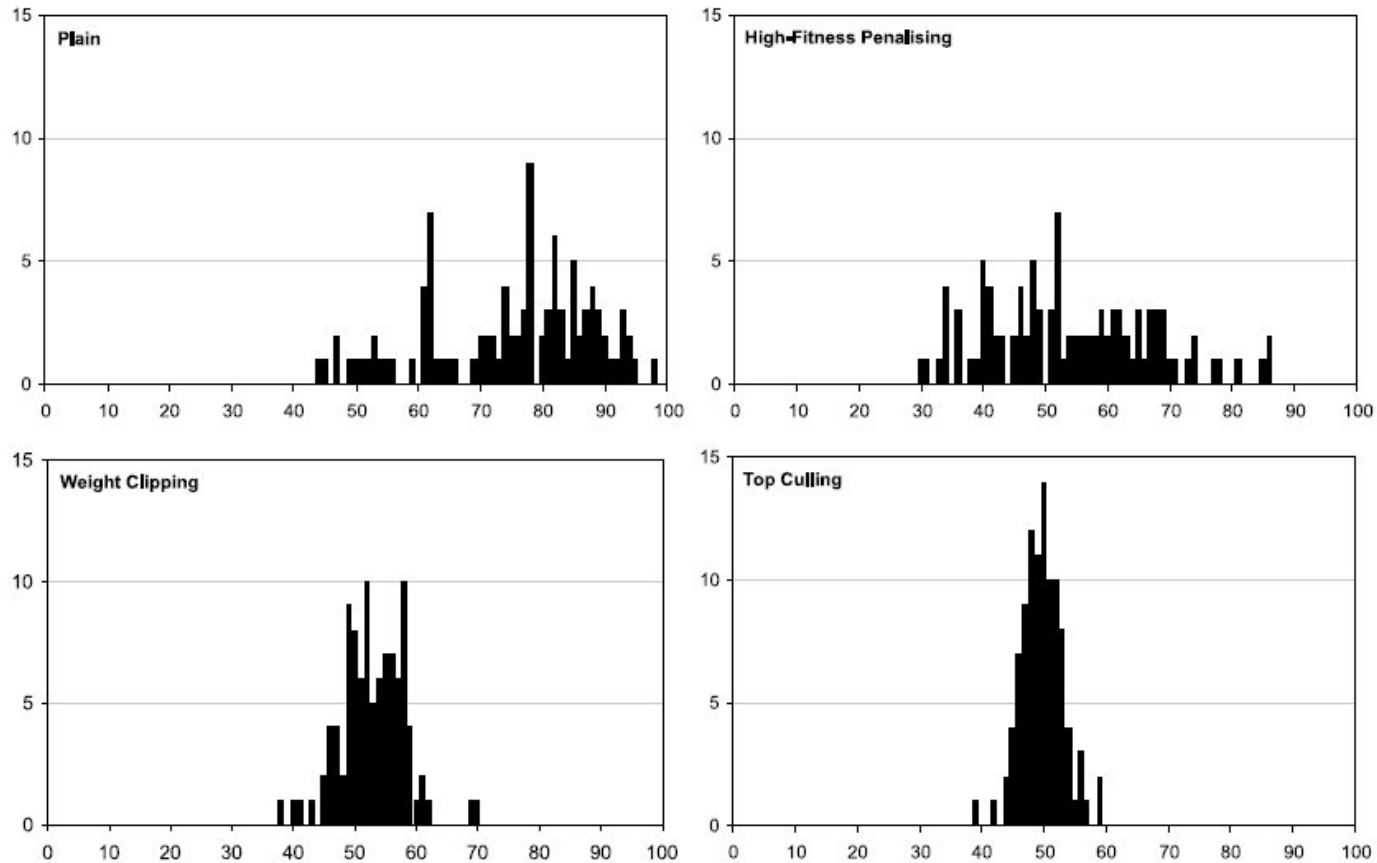


Figure 9. Histograms of 100 tests of the achieved number of wins in 100 fights, against the 'novice' tactic.

DS: Kommerzielle Anwendung?

- Modul für Neverwinter Nights

DS: Kommerzielle Anwendung?

- Besser als im entwickelten Minigate!

Table VI. Turning point values for dynamic scripting in NEVERWINTER NIGHTS.

Tactic	Tests	Avg.	St.dev.	Median	Highest	Top 5
AI 1.29	50	21	8.8	16	101	58
AI 1.61	31	35	18.8	32	75	65
Cursed AI	21	33	21.8	24	92	64

DS: Fazit

- Adaptive KI mit Zukunft
  - Braucht „mutige“ Entwickler
- Lernen noch besser verstecken
  - Flucht bei Unterlegenheit
  - Spieler „verhält“ sich anders?
    - Ihn zuerst angreifen lassen
- Wenn 50 / 50 zu schwer
  - Variieren auf 40 / 60 oder 60 / 40 (für Masochisten)

Quellen

- www.wikipedia.com
- http://www.ercim.org/publication/Ercim_News/enw57/spronck.html
- http://www.cs.unimaas.nl/p.spronck/
- http://nwn.bioware.com/