Σχεδίαση και Ανάλυση πολύπλοκων παιγνίων Slots και βελτιστοποίηση του RTP με χρήση της μεθευρετικής μεθόδου Variable Neighborhood Search (VNS)

Παντελής-Αρσένιος Καμανάς, mai20020

Πανεπιστήμιο Μακεδονίας, Σχολή Επιστημών Πληροφορίας
Τμήμα Εφαρμοσμένης Πληροφορικής, Εγνατίας 156, 54636 Θεσσαλονίκη

24 Φεβρουαρίου, 2021

Περιεχόμενα

Εμπορικά λογισμικά πακέτα για σχεδίαση Slots

- Slot Wizard
  `https://eliconsoft.com/`
- Slot Designer
  `http://slotdesigner.com/`

Παρόμοιες μεθοδολογίες

- DDE: Balabanov, Todor and Zankinski, Iliyan and Shumanov, Bozhidar (2015). Slot Machine RTP Optimization and Symbols Wins Equalization with Discrete Differential Evolution. 210-217. 10.1007/978-3-319-26520-9_22.
- GA: Balabanov, Todor & Zankinski, Iliyan & Shumanov, Bozhidar. (2015). Slot Machines RTP Optimization with Genetic Algorithms. 8962. 55-61. 10.1007/978-3-319-15585-2_6.
- EA: Keremedchiev, Delyan & Tomov, Petar & Barova, Maria. (2017). Slot Machine Base Game Evolutionary RTP Optimization. 406-413. 10.1007/978-3-319-57099-0_45.

Κίνητρο της εργασίας

- Εξειδίκευση στο αλγοριθμικό και μαθηματικό κομμάτι των τυχερών παιγνίων Slots
- Η δημιουργία μαθηματικών μοντέλων που βοηθούν τον μαθηματικό να αναπτύσσει παιχνίδια Slots γρήγορα και με αξιόπιστα αποτελέσματα
- Ανάπτυξη ενός αποτελεσματικού, αλλά και σχετικά μη-κοστοβόρου σε χρόνο, αλγορίθμου, με τη χρήση της μεθευρετικής μεθόδου VNS για την επίλυση του προβλήματος της βελτιστοποίησης-ελαχιστοποίησης **minimize** του RTP. Το πρόβλημα αυτό μας ανατέθηκε αλλά και χρηματοδοτήθηκε από την εταιρεία ZeusPlay που αναπτύσσει περιέχομενο (παιχνίδια, λογισμικά) με επίκεντρο τα τυχερά παιχνίδια (`https://zeusplay.com`)

Παράδειγμα παιχνιδιού

Amun's Book : A book game developed by *ZeusPlay*

Σημαντικοί όροι

- Monte Carlo Simulation : Θεωρείται ο τύπος της προσομοίωσης που μας δίνει αξιόπιστα στατιστικά αποτελέσματα εάν ο αριθμός των προσομοιώσεων είναι $n \geq 10{,}000{,}000$)
- Full Cycle Simulation : Θεωρείται ο τύπος της προσομοίωσης που υπολογίζει όλες τις μεταβλητές του παιχνιδιού λαμβάνοντας υπόψη όλους τους πιθανούς συνδυασμούς, όπου σε ένα κανονικό-συνηθισμένο παιχνίδι $n \geq 120{,}000{,}000$)
- RTP : Ή αλλιώς *Return-To-Player* θεωρείται το ποσοστό επιστροφής στον παίκτη, δηλαδή, εάν ,στατιστικά, γίνουν 10,000 γυρίσματα (spins) με 1 coin/spin, και το παιχνίδι χαρακτηρίζεται από RTP = 97%, στο τέλος θα επιστραφεί στον παίκτη περίπου το ποσό των 9,700 coins

Σημαντικοί όροι (2)

- Cycle : Θεωρείται ο αριθμός όλων των πιθανών συνδυασμών ή των προσομοιώσεων (ανάλογα σε ποια προσομοίωση αναφερόμαστε)
- Hits : Θεωρείται ο αριθμός των νικητήριων συνδυασμών
- Volatility : Είναι η μεταβλητή που αναφέρεται στο πόσο ¨σκληρό' είναι ένα παιχνίδι
- Hitrate : Αναφέρεται στη συχνότητα εμφάνισης ενός νικητήριου συνδυασμού, δηλαδή εάν Hitrate $= 7$, αυτό σημαίνει πως , στατιστικά, κάθε 7 γυρίσματα θα υπάρχει και μία πληρωμή
- Wild Symbol : Το σύμβολο μπαλαντέρ
- Scatter Symbol : Το σύμβολο που πληρώνει σε οποιαδήποτε θέση και αν βρεθεί στην οθόνη

Μαθηματική προσέγγιση

Ο υπολογισμός των Hits γίνεται με τη χρήση των παρακάτων εξισώσεων. Οι τύποι αναφέρονται σε παιχνίδια Slots που έχουν μπαλαντέρ Wild, έχουν ως κανόνα πληρωμής Left to Right ή Right to Left και μόνο πρώτος τροχός δεν περιέχει μπαλαντέρ σύμβολα καθόλου. Αν $(n = c)$:

$$H_{ic} = \prod_{j=1}^{c} (t_{ij} + t_{kj}) \tag{1}$$

Αν $(c \leq n \leq 2+ c)$:

$$H_{ic} = \prod_{j=1}^{c} (t_{ij} + t_{kj}) \times (S_m - t_{im} - t_{km}) \tag{2}$$

Μαθηματική προσέγγιση (2)

Αν $(n \geq 2 + c)$:

$$H_{ic} = \prod_{j=1}^{c}(t_{ij} + t_{kj}) \times (S_m - t_{im} - t_{km}) \times \prod_{p}^{n} S_p \tag{3}$$

Η πιθανότητα να προκύψει ένας νικητήριος συνδυασμός του c-symbol δίνεται από την παρακάτων εξίσωση :

$$P_{H_{ic}} = \frac{H_{ic}}{CL} \tag{4}$$

Για τις παραπάνω εξισώσεις ισχύουν τα εξής:
c : Συνδυασμός διπλέτας, τριπλέτας, ..., τελευταία στήλη, δηλαδή εάν c=2 τότε ψάχνουμε για διπλέτα

Μαθηματική προσέγγιση (3)

i: Θέση της γραμμής i
n: Ο αριθμός των τροχών-στηλών σε έναν πίνακα
m : c+1
p : m+1
$\mathbf{S}_i$: Αναφέρεται στον συνολικό αριθμό των συμβόλων στη στήλη-τροχό i για i=1, 2, 3, ..., n
$\mathbf{t}_{ic}$: Ένα σύμβολο-στοιχείο στη γραμμή-i και συνδυασμός c στη σειρά (κανόνας γραμμής)
k : Θέση-γραμμή του συμβόλου μπαλαντέρ (wild)
CL: ή αλλιώς Cycle

Αλγοριθμική προσέγγιση

Οι παρακάτω αλγόριθμοι έχουν εφαρμοστεί σε παιχνίδια Slots στις γλώσσες προγραμματισμού Java και Python που χρησιμοποιήθηκαν για προσομοιώσεις και μπορούν να βρεθούν στο Gihub μαζί με τις μαθηματικές τους αναλύσεις, παραπάνω επεξηγήσεις αλλά και άλλους τύπους τυχερών παιγνίων. Οι προσομοιώσεις αυτές και οι αναλύσεις βοηθούν τον game designer να αναπτύξει ένα παιχνίδι με όσο το δυνατόν καλύτερο αποτέλεσμα.

Github Links :
`https://github.com/arseniumn/Slot-Machine-Wild-Sevens`
`https://github.com/arseniumn/Slot-Machine-Rolling-Bars`

Αλγοριθμική προσέγγιση (2)

Algorithm 1 Payment for winning combinations (including wild symbol)

Data: $W, P, L, Map, linebet$
Result: $coins$
$aPair \leftarrow 0$
$index \leftarrow -1$
$coins \leftarrow 0$
$m \leftarrow length(W)$
$n \leftarrow length(W[0])$
$saveWildCoordinates()$
for i=0 *to* $length(L)$ **do**
  if $W[L[i][j]][j] \neq$ '$Scatter$' **then**
    $aPair \leftarrow 0$
    for j=0 *to* n **do**
      if $W[L[i][j]][j] = W[L[i][j+1][j+1]$ *or* $W[L[i][j+1][j+1] =$ '$Wild$' **then**
        $W[L[i][j+1][j+1] \leftarrow W[L[i][j][j]$
        $aPair \leftarrow aPair + 1$
      else
        break
    end
  end
  $index \leftarrow Map.getKey(W[L[i][0]][0])$
  $coins \leftarrow coins + P[index][aPair] \times linebet$
  $retrieveWildSymbols()$
end

Εκτελείται για κάθε γραμμή επιλογής (Payline)

Αλγοριθμική προσέγγιση (3)

Algorithm 2 Payment for winning combinations (scatter symbols)

Data: $W, P, L, Map, totalbet$
Result: $coins$
$m \leftarrow length(W)$
$n \leftarrow length(W[0])$
$times \leftarrow 1$
$coins \leftarrow 0$
$C[0...n] \leftarrow 0$
$aPair \leftarrow 0$
foreach i *to* n **do**
  foreach j *to* m **do**
    if $W[i][j] =$ '$Scatter$' **then**
      $C[i] \leftarrow C[i] + 1$
    end
  end
  if $C[i] > 0$ **then**
    $times \leftarrow times \times C[i]$
    $aPair \leftarrow aPair + 1$
  end
end
$index \leftarrow Map.getKey('Scatter')$
$coins \leftarrow P[index][aPair] \times totalbet \times times$

Εκτελείται για κάθε σύμβολο με κανόνα οθόνης (Scatter)

Το RTP και η βελτιστοποίηση

Ένα από τα πιο σημαντικά και χρονοβόρα ζητήματα που έχει να αντιμετωπιστεί στο casino game industry, από την πλευρά των providers που αναπτύσσουν συνεχώς νέα παιχνίδια, είναι η βελτιστοποίηση του RTP. Με τον όρο βελτιστοποίηση, εννοούμε την αύξηση ή μείωση του RTP με τη χρήση ευρετικών ή και μεθευρετικών μεθόδων. Σε συνεργασία με την εταιρεία ZeusPlay και την χρηματοδότηση της, αναπτύχθηκε ένας αλγόριθμος ο οποίος έχει ως βάση τη μεθευρετική (metaheuristic) μέθοδο γνωστή ως **VNS** : Variable Neighborhood Search και συγκεκριμένα τη **VND** : Variable Neighborhood Descent και στοχεύει στην *ελαχιστοποίηση* του RTP (minimize problem). Για την εφαρμογή της εναλλαγής γειτονιάς, χρησιμοποιήθηκαν 2 **local search operators**, οι **RMS** και **RMC**.

Τοπικοί τελεστές αναζήτησης

- **RMS (Replace Maximum Symbol) :** Ο τελεστής αυτός αντικαθιστά στον τροχό το ακριβότερο σύμβολο στο παιχνίδι (π.χ sym10) με το αμέσως λιγότερο ακριβό (π.χ σψμ09)
- **RMC (Replace Minimum Combination) :** Το RTP αυξάνεται όταν υπάρχουν σύμβολα στους τροχούς που έχουν μεγάλη συχνότητα στους τροχούς και ταυτόχρονα πληρώνουν τον ελάχιστο συνδυασμό νίκης π.χ διπλέτα του sym01. Ο τελεστής αυτός αντικαθιστά το σύμβολο αυτό με ένα άλλο που δεν συμμετέχει στη διπλέτα π.χ sym02 αλλά στης τριπλέτας και άνω.

Ο αλγόριθμος VND

Επίσης, η προσομοίωση Monte-Carlo (*RunSimulation()*) χρησιμοποιείται για τον υπολογισμό του τρέχοντος RTP_{curr} με 100,000 ή 1,000,000 ξεχωριστές προσομοιώσεις. Οι δύο τελεστές δηλώνονται επίσης ως N_1 και N_2, αντίστοιχα. Επιπλέον, ο αριθμός των διαφορετικών τελεστών γειτονιάς δηλώνεται ως l_{max} και σε αυτή την εφαρμογή ισούται με δύο.

Algorithm 10 VND

Input : $RTP_{des}, iteration_{max}, R, minNSR, Reels, l_{max}$
Output: $Reels$
$Reel_{curr} \leftarrow 1$
$RTP_{curr} \leftarrow RunSimulation()$
$Difference \leftarrow RTP_{curr} - RTP_{des}$
repeat
  $l \leftarrow 1$
  repeat
    Select $Reels' \in \mathcal{N}_l(x)$ such that $RTP(Reels') < RTP(Reels)$
    $RTP_{curr} \leftarrow$ **RunSimulation()**
    NeighborhoodChange$(Reels, Reels', l)$
    $Reel_{curr} \leftarrow (Reel_{curr} + 1) mod(n)$
  until $l = l_{max}$
until *stopping criterion*

Τοπικοί τελεστές αναζήτησης : RMS

Στην παρακάτω εικόνα παρατηρείται εφαρμογή του **RMS** σε ένα τυχαίο τροχό παιχνιδιού

$$\begin{bmatrix} sym01 \\ sym02 \\ sym02 \\ sym02 \\ sym01 \\ sym01 \\ sym03 \\ sym02 \\ sym01 \\ sym03 \\ sym01 \\ sym04 \\ sym04 \end{bmatrix} \Rightarrow \begin{bmatrix} sym01 \\ sym02 \\ sym02 \\ sym02 \\ sym01 \\ sym01 \\ sym03 \\ sym02 \\ sym01 \\ sym03 \\ sym01 \\ sym04 \\ sym03 \end{bmatrix}$$

Τοπικοί τελεστές αναζήτησης : RMC

Στην παρακάτω εικόνα παρατηρείται εφαρμογή του **RMC** σε ένα τυχαίο τροχό παιχνιδιού

$$\begin{bmatrix} sym01 \\ sym02 \\ sym02 \\ sym02 \\ sym01 \\ sym01 \\ sym03 \\ sym02 \\ sym01 \\ sym03 \\ sym01 \\ sym04 \\ sym04 \end{bmatrix} \Rightarrow \begin{bmatrix} sym02 \\ sym02 \\ sym02 \\ sym02 \\ sym01 \\ sym01 \\ sym03 \\ sym02 \\ sym01 \\ sym03 \\ sym01 \\ sym04 \\ sym04 \end{bmatrix}$$

Βελτιστοποίηση σε 3 βασικά παιχνίδια

Ο παραπάνω αλγόριθμος εφαρμόστηκε σε παιχνίδια της εταιρείας ZeusPlay έτσι ώστε να μειωθεί το RTP κατά το επιθυμητό. Όλοι οι υπολογισμοί έγιναν με τη χρήση ενός Dell Inspiron 5558 laptop με έναν Intel Core i5-5200U στα 2.20 GHz, 8 GB DDR3 RAM στα 1600 MHz και έχοντας ως λειτουργικό σύστημα Microsoft Windows 10 Professional 64-bit.

- **Le Mystere Du Prince**, a five-reel slot machine game with wild and scatter as special symbols,
- **Amun's Book**, a five-reel slot machine game that holds one special symbol called book and acts as a wild and scatter,
- **Wild Charger**, a five-reel slot machine game that holds one wild and acts as expanding and sticky at the same time (symbol that developed by Zeusplay).

Όλα τα παραπάνω παιχνίδια έχουν 10 βασικές γραμμές (Paylines) και όλα τα σύμβολα πληρώνουν από αριστερά προς τα δεξιά στο βασικό παιχνίδι

Βελτιστοποίηση σε 3 βασικά παιχνίδια - Αποτελέσματα

Runs	RTP Initial	RTP Target	RTP Obtained
100,000	718.374%	77-83%	82.025%
1,000,000	712.661%	77-83%	78,996%

Table 2: Le Mystere Du Prince

Runs	RTP Initial	RTP Target	RTP Obtained
100,000	56.105%	38-42%	41.277%
1,000,000	57.226%	38-42%	38.758%

Table 3: Wild Charger

Runs	RTP Initial	RTP Target	RTP Obtained
100,000	111.005%	52-54%	52.015%
1,000,000	104.852%	52-54%	52.803%

Table 4: Amun's Book

Βελτιστοποίηση σε 3 βασικά παιχνίδια - Αποτελέσματα (2)

Game	Simulation Runs	Time (secs)	Iterations
Le Mystere Du Prince	100,000	353	2456
Le Mystere Du Prince	1,000,000	786	501
Wild Charger	100,000	2	2062
Wild Charger	1,000,000	15	350
Amun's Book	100,000	35	110
Amun's Book	1,000,000	223	149

Table 5: Results for the three games

Βελτιστοποίηση σε 3 βασικά παιχνίδια - Αποτελέσματα (3)

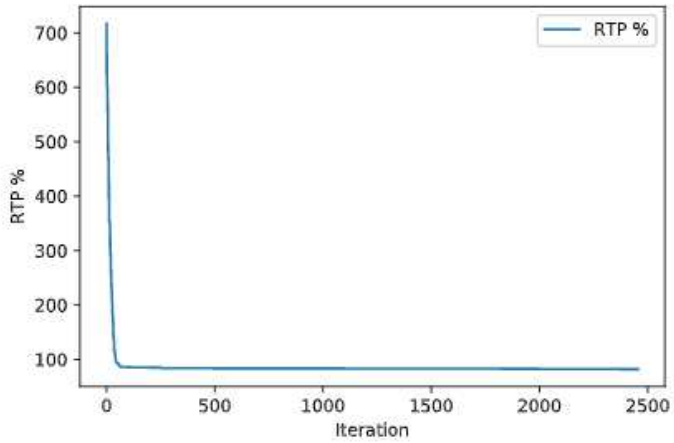

Figure 2: Le Mystere Du Prince: 100,000 runs

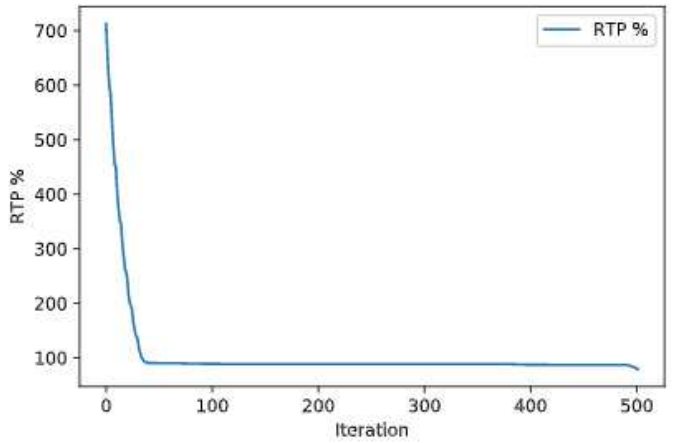

Figure 3: Le Mystere Du Prince: 1,000,000 runs

Βελτιστοποίηση σε 3 βασικά παιχνίδια - Αποτελέσματα (4)

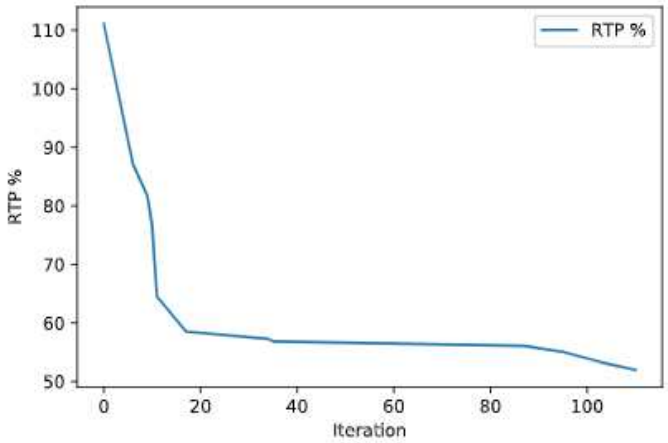

Figure 4: Amun's Book: 100,000 runs

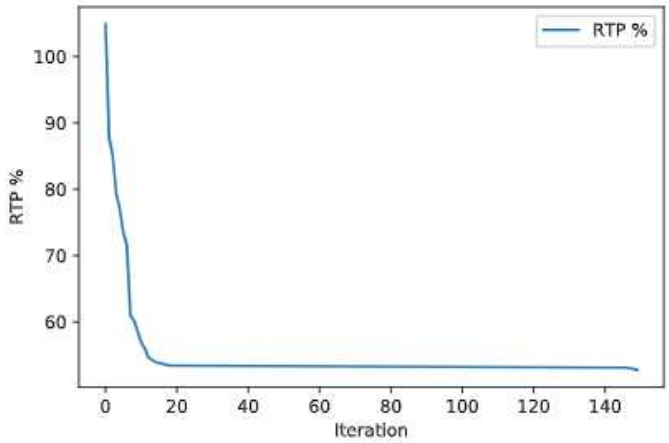

Figure 5: Amun's Book: 1,000,000 runs

Βελτιστοποίηση σε 3 βασικά παιχνίδια - Αποτελέσματα (5)

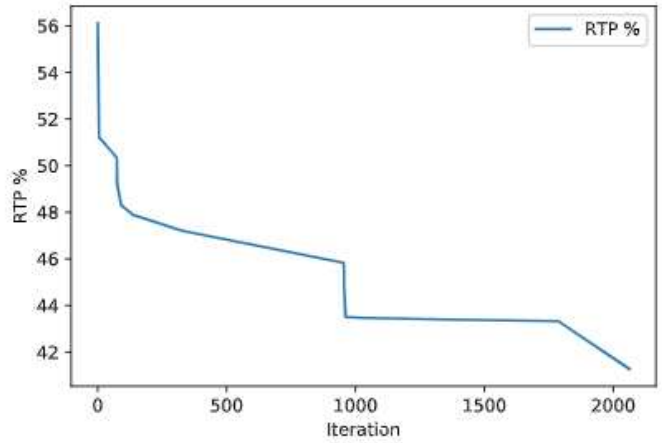

Figure 6: Wild Charger: 100,000 runs

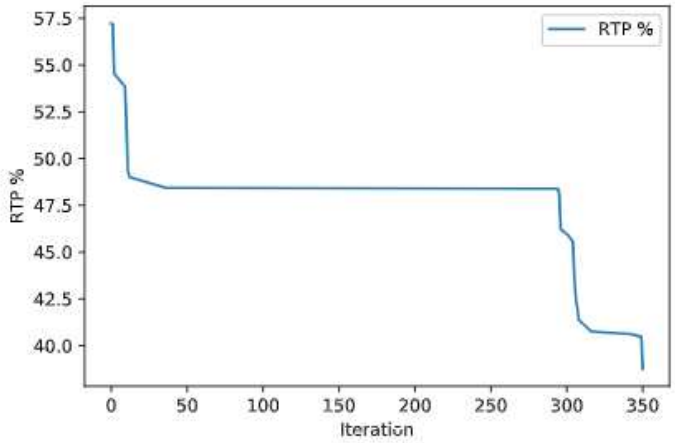

Figure 7: Wild Charger: 1,000,000 runs

Μελλοντικές υλοποιήσεις

Καθώς αυξάνεται η πολυπλοκότητα των παιχνιδιών, ο απαιτούμενος χρόνος ανάλυσης και βελτιστοποίησης αυξάνεται. Το γεγονός αυτό δημιουργεί την ανάγκη επιτάχυνσης ολόκληρης της διαδικασίας. Ως μελλοντική έρευνα, μπορεί να προταθεί

- Μια εφαρμογή με πιο περίπλοκα παιχνίδια, χρησιμοποιώντας τεχνικές παραλληλοποίησης για CPU αλλά και GPU για τη μείωση του υπολογιστικού χρόνου για την εξεύρεση μιας αποδεκτής λύσης (RTP)
- ο προτεινόμενος αλγόριθμος είναι μια προσέγγιση **ενός** κριτηρίου για τη βελτιστοποίηση το RTP. Μια άλλη ερευνητική ιδέα είναι να αναπτυχθεί μια μέθοδος πολλαπλών κριτηρίων για τη βελτιστοποίηση του RTP, όπως είναι το Volatility και το Hitrate.

Τέλος της παρουσίασης

Ευχαριστώ για την προσοχή σας!