

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ  
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επίλυση του Προβλήματος του Σακιδίου

(Knapsack Problem)

με τη χρήση Νευρωνικών Δικτύων όλων των τύπων και Αλγορίθμων Βαθιάς Μάθησης-Deep Learning

|  |  |
| --- | --- |
| **Του φοιτητή**  **Βαλσαμή Αντώνιου**  **Αρ. Μητρώου: 174869** | **Επιβλέπων**  **Ονοματεπώνυμο Κωνσταντίνος Γουλιάνας**  **Βαθμίδα Καθηγητής** |

**Ημερομηνία**

Τίτλος Δ.Ε: Επίλυση του Προβλήματος του Σακιδίου (knapsack problem) με τη χρήση Νευρωνικών Δικτύων όλων των τύπων και Αλγορίθμων Βαθιάς Μάθησης-Deep Learning

Κωδικός Δ.Ε: 22176

Ονοματεπώνυμο φοιτητή: Βαλσαμής Αντώνιος

Ονοματεπώνυμο εισηγητή: Γουλιάνας Κωνσταντίνος

Ημερομηνία ανάληψης Δ.Ε: 31-03-2023

Ημερομηνία περάτωσης Δ.Ε:

*Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.*

*Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του φοιτητή Βαλσαμή Αντώνιου που την εκπόνησε. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης, ο συγγραφέας/δημιουργός εκχωρεί στο Διεθνές Πανεπιστήμιο της Ελλάδος άδεια χρήσης του δικαιώματος αναπαραγωγής, δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσης της εργασίας διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος. Η ανοικτή πρόσβαση στο πλήρες κείμενο της εργασίας, δεν σημαίνει καθ’ οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού, ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, πώληση, εμπορική χρήση, διανομή, έκδοση, μεταφόρτωση (downloading), ανάρτηση (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού.*

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

*«Στην οικογένειά και τους φίλους μου»*

Περίληψη

Η παρούσα εργασία αναλύει και παρουσιάζει το πρόβλημα του Σακιδίου (knapsack problem) και την επίλυσή με τη χρήση Νευρωνικών Δικτύων όλων των τύπων και Αλγορίθμων Βαθιάς Μάθησης (Deep Learning). Ο επιλύτης είναι εμπνευσμένος από μηχανισμούς και στρατηγικές που χρησιμοποιούνται τόσο από αλγοριθμικούς επιλύτες όσο και από ανθρώπους. Σε ορισμένες περιπτώσεις της επιστήμης των δεδομένων, απαιτείται η εκτέλεση ενός συγκεκριμένου αλγορίθμου στην έξοδο του μοντέλου για να προκύψει το αποτέλεσμα. Μερικές φορές είναι τόσο απλό όσο η εύρεση του δείκτη της μέγιστης εξόδου, ενώ άλλες φορές απαιτούνται πιο προηγμένοι αλγόριθμοι. Στόχος της παρούσας μελέτης είναι να εμβαθύνουμε στην ενσωμάτωση νευρωνικών δικτύων και τεχνικών βαθιάς μάθησης για την αντιμετώπιση του προβλήματος knapsack. Θα συζητήσουμε τη διατύπωση του προβλήματος στο πλαίσιο των νευρωνικών δικτύων, τις σκέψεις σχεδιασμού για την αρχιτεκτονική των μοντέλων, τη διαδικασία εκπαίδευσης και τις μετρικές αξιολόγησης για την αξιολόγηση της απόδοσης των μοντέλων. Αναγνωρίζοντας τις προκλήσεις που συνεπάγεται η εφαρμογή της βαθιάς μάθησης στη συνδυαστική βελτιστοποίηση, στοχεύουμε να αναδείξουμε τις δυνατότητες αυτής της προσέγγισης και τις επιπτώσεις της στην επίλυση σύνθετων προβλημάτων κατανομής πόρων.

**Λέξεις Κλειδιά:** *πρόβλημα του Σακιδίου (knapsack problem), Νευρωνικά Δίκτυα, Αλγόριθμοι Βαθιάς Μάθησης*

The knapsack problem and its solution using neural networks of all types and deep learning algorithms

Antonios Valsamis

Αbstract

This paper analyzes and presents the Knapsack problem and its solution using Neural Networks of all types and Deep Learning Algorithms. The solver is inspired by mechanisms and strategies used by both algorithmic and human solvers. In some cases of data science, it is necessary to run a specific algorithm on the output of the model to derive the result. Sometimes it is as simple as finding the index of maximum output, while other times more advanced algorithms are required. The aim of the present study is to delve into the integration of neural networks and deep learning techniques to deal with the knapsack problem. We will discuss the formulation of the problem in the context of neural networks, design considerations for model architecture, the training process, and evaluation metrics for evaluating model performance. Recognizing the challenges involved in applying deep learning to combinatorial optimization, we aim to highlight the potential of this approach and its implications for solving complex resource allocation problems.

**Keywords**: knapsack problem, Neural Networks, Deep Learning Algorithms

Περιεχόμενα

[Περίληψη vi](#_Toc175441918)

[Αbstract vii](#_Toc175441919)

[Περιεχόμενα viii](#_Toc175441920)

[Κεφάλαιο 1ο: Το πρόβλημα του σακιδίου 1](#_Toc175660596)

[1.1 Εισαγωγή 1](#_Toc175660597)

[1.2 Ανασκόπηση της βιβλιογραφίας για το πρόβλημα του σακιδίου 4](#_Toc175660598)

[1.3 Εισαγωγή 6](#_Toc175660599)

[1.4 Η πολυπλοκότητα του Προβλήματος του Σακιδίου 8](#_Toc175660600)

[1.5 Οι εφαρμογές του knapsack problem 9](#_Toc175660601)

[1.6 Παρουσίαση του προβλήματος του σακιδίου 11](#_Toc175660602)

[1.7 Παραλλαγές και επεκτάσεις του προβλήματος του σακιδίου 18](#_Toc175660603)

[1.8 Πρόβλημα μίας χωρητικότητας έναντι όλων των χωρητικοτήτων 22](#_Toc175660604)

[1.9 Υποθέσεις για τα δεδομένα εισόδου 23](#_Toc175660605)

[1.10 Απόδοση Αλγορίθμων 25](#_Toc175660606)

[1.11 Συμπεράσματα Κεφαλαίου 27](#_Toc175660607)

[Κεφάλαιο 2ο: Η χρήση των Νευρωνικών Δικτύων (Νeural Νetworks) στην επίλυση του Knapsack Problem………………………………………………………………………………………………….....29](#_Toc175660608)

[2.1 Εισαγωγή 29](#_Toc175660609)

[2.2 Ανασκόπηση της βιβλιογραφίας για τη χρήση των Νευρωνικών Δικτύων (Νeural Νetworks) στην επίλυση του ΚP 30](#_Toc175660610)

[2.3 Συμπεράσματα Κεφαλαίου 31](#_Toc175660611)

[Κεφάλαιο 3ο: Βαθιά Μάθηση (Deep Learning) για την επίλυση του Knapsack Problem 33](#_Toc175660612)

[3.1. Εισαγωγή 33](#_Toc175660613)

[3.2. Ανασκόπηση της βιβλιογραφίας στη Βαθιά Μάθηση (Deep Learning) για την επίλυση του KP……………………………………………………………………………………………………….33](#_Toc175660614)

[3.3. Οι αλγόριθμοι Deep Learning 34](#_Toc175660615)

[3.4. Συμπεράσματα Κεφαλαίου 39](#_Toc175660616)

[Κεφάλαιο 4ο: Μεθοδολογία 40](#_Toc175660617)

[4.1 Εισαγωγή 40](#_Toc175660618)

[4.2 Βαθύ Νευρωνικό Δίκτυο (Deep Neural Network, DNN) 40](#_Toc175660619)

[4.2.1 Επισκόπηση 40](#_Toc175660620)

[4.2.2 Αρχιτεκτονική 41](#_Toc175660621)

[4.2.3 Διαδικασία Εκπαίδευσης 41](#_Toc175660622)

[4.3 Επαναλαμβανόμενο Νευρωνικό Δίκτυο (Recurrent Neural Network, RNN) 42](#_Toc175660623)

[4.3.1 Επισκόπηση 42](#_Toc175660624)

[4.3.2 Αρχιτεκτονική 42](#_Toc175660625)

[4.3.3 Διαδικασία Εκπαίδευσης 42](#_Toc175660626)

[4.4 Μοντέλο Deep Q-Learning (DQN) 43](#_Toc175660627)

[4.4.1 Επισκόπηση 43](#_Toc175660628)

[4.4.2 Αρχιτεκτονική 43](#_Toc175660629)

[4.4.3 Διαδικασία Εκπαίδευσης 44](#_Toc175660630)

[4.5 Σύνολο Δεδομένων και Προεπεξεργασία 44](#_Toc175660631)

[4.5.1 Δημιουργία Συνόλου Δεδομένων 44](#_Toc175660632)

[4.5.2 Προεπεξεργασία Δεδομένων 45](#_Toc175660633)

[4.6 Επίλογος Κεφαλαίου 45](#_Toc175660634)

[Κεφάλαιο 5ο: Εφαρμογή 46](#_Toc175660635)

[5.1 Εισαγωγή 46](#_Toc175660636)

[5.2 Δημιουργία και Προετοιμασία Δεδομένων 46](#_Toc175660637)

[5.2.1 Δημιουργία δεδομένων 46](#_Toc175660638)

[5.2.2 Υλοποίηση του Δυναμικού Προγραμματισμού (knapsack\_dp) 47](#_Toc175660639)

[5.2.3 Συμπέρασμα για την Προετοιμασία Δεδομένων 48](#_Toc175660640)

[5.3 Βαθύ Νευρωνικό Δίκτυο (Deep Neural Network, DNN) 48](#_Toc175660641)

[5.3.1 Δημιουργία και Αρχιτεκτονική του Μοντέλου DNN 48](#_Toc175660642)

[5.3.2 Εκπαίδευση του Μοντέλου DNN 50](#_Toc175660643)

[5.3.3 Επίλυση του Προβλήματος με το Εκπαιδευμένο Μοντέλο DNN 53](#_Toc175660644)

[5.3.4 Βελτιστοποίηση Υπερπαραμέτρων Μοντέλου DNN 55](#_Toc175660645)

[5.4 Μοντέλο Επαναλαμβανόμενου Νευρωνικού Δικτύου (Recurrent Neural Network, RNN)…………………………………………………………………………………………………….56](#_Toc175660646)

[5.4.1 Δημιουργία και Αρχιτεκτονική του Μοντέλου RNN 56](#_Toc175660647)

[5.4.2 Προετοιμασία Δεδομένων για το RNN 59](#_Toc175660648)

[5.4.3 Εκπαίδευση του Μοντέλου RNN 60](#_Toc175660649)

[5.4.4 Επίλυση του Προβλήματος με το Εκπαιδευμένο Μοντέλο RNN 62](#_Toc175660650)

[5.4.5 Βελτιστοποίηση Υπερπαραμέτρων RNN 64](#_Toc175660651)

[5.5 Μοντέλο Deep Q-Learning (DQN) 65](#_Toc175660652)

[5.5.1 Δημιουργία και Αρχιτεκτονική του Μοντέλου DQN 65](#_Toc175660653)

[5.5.2 Διαχείριση Εμπειριών (Replay Memory) 66](#_Toc175660654)

[5.5.3 Περιβάλλον του Σακιδίου (Knapsack Environment) 68](#_Toc175660655)

[5.5.4 Εκπαίδευση και Επίλυση με το Μοντέλο DQN 71](#_Toc175660656)

[5.5.5 Βελτιστοποίηση Υπερπαραμέτρων Μοντέλου DQN 80](#_Toc175660657)

[5.6 Οπτικοποίηση – Γραφική Επίδειξη Εκπαίδευσης 81](#_Toc175660658)

[5.6.1 Κλάση Γραφικής Επίδειξης της Διαδικασίας Εκπαίδευσης (KnapsackVisualization)……………………………………………………………………………...81](#_Toc175660659)

[5.6.2 Διαχείρηση Θέσης Παραθύρου Οπτικοποίησης 89](#_Toc175660660)

[5.7 Κεντρική Εφαρμογή και Διεπαφή Χρήστη (GUI) 90](#_Toc175660661)

[5.7.1 Κεντρική Λειτουργία Εφαρμογής 90](#_Toc175660662)

[5.7.2 Πρόβλεψη Εκπαιδευμένου Μοντέλου με Νέο Δείγμα 92](#_Toc175660663)

[5.7.3 Δημιουργία και Λειτουργία Διεπαφής Χρήστη (GUI) 94](#_Toc175660664)

[5.8 Αυτοματοποιημένες δοκιμές και Παρακολούθηση Απόδοσης 96](#_Toc175660665)

[5.9 Επίλογος Κεφαλαίου 97](#_Toc175660666)

[Κεφάλαιο 6ο: Ανάλυση Αποτελεσμάτων και Συμπεράσματα 98](#_Toc175660667)

[6.1 Εισαγωγή 98](#_Toc175660668)

[6.2 Ανάλυση των Αποτελεσμάτων Εκπαίδευσης 98](#_Toc175660669)

[6.3 Σύγκριση των Μοντέλων 100](#_Toc175660670)

[6.4 Εξέταση Απόδοσης με Βάση τα Δεδομένα Εισόδου 100](#_Toc175660671)

[Συμπεράσματα 102](#_Toc175660672)

[Μελλοντική Έρευνα 104](#_Toc175660673)

[Βιβλιογραφία 106](#_Toc175660674)

[ΠΑΡΑΡΤΗΜΑ Α: Κώδικας Εφαρμογής 111](#_Toc175660675)

[ΠΑΡΑΡΤΗΜΑ Β: Κώδικες Βελτιστοποίησης Υπερπαραμέτρων Μοντέλων και Αυτοματοποιημένων Δοκιμών 112](#_Toc175660677)

Κατάλογος Σχημάτων

[5.1 Στιγμιότυπο κώδικα generate\_data 46](#_Toc175487474)

[5.2.Στιγμιότυπο κώδικα knapsack\_dp 47](#_Toc175487475)

[5.3.Σιτγμιότυπο κώδικα knapsack\_dp (2) 48](#_Toc175487476)

[5.4.Σιτγμιότυπο κλάσης KnapsackDNN 49](#_Toc175487477)

[5.5. Σιτγμιότυπο κλάσης KnapsackDNN (2) 49](#_Toc175487478)

[5.6. Σιτγμιότυπο κλάσης KnapsackDNN (3) 49](#_Toc175487479)

[5.7. Σιτγμιότυπο κλάσης KnapsackDNN (4) 50](#_Toc175487480)

[5.8. Σιτγμιότυπο κλάσης KnapsackDNN (5) 50](#_Toc175487481)

[5.9. Σιτγμιότυπο κώδικα train\_nn 51](#_Toc175487482)

[5.10. Σιτγμιότυπο κώδικα train\_nn (2) 51](#_Toc175487483)

[5.11. Σιτγμιότυπο κώδικα train\_nn (3) 52](#_Toc175487484)

[5.12. Σιτγμιότυπο κώδικα train\_nn (4) 52](#_Toc175487485)

[5.13. Σιτγμιότυπο κώδικα train\_nn (5) 53](#_Toc175487486)

[5.14. Σιτγμιότυπο κώδικα train\_nn (6) 53](#_Toc175487487)

[5.15. Σιτγμιότυπο κώδικα solve\_knapsack\_with\_nn 54](#_Toc175487488)

[5.16.Σιτγμιότυπο κώδικα solve\_knapsack\_with\_nn (2) 54](#_Toc175487489)

[5.17.Σιτγμιότυπο κώδικα solve\_knapsack\_with\_nn (3) 54](#_Toc175487490)

[5.18.Σιτγμιότυπο κώδικα solve\_knapsack\_with\_nn (4) 55](#_Toc175487491)

[5.19.Σιτγμιότυπο κλάσης KnapsackRNN 55](#_Toc175487492)

[5.20.Σιτγμιότυπο κλάσης KnapsackRNN (2) 55](#_Toc175487493)

[5.21.Σιτγμιότυπο κλάσης KnapsackRNN (3) 56](#_Toc175487494)

[5.22.Σιτγμιότυπο κλάσης KnapsackRNN (4) 56](#_Toc175487495)

[5.23.Σιτγμιότυπο κλάσης KnapsackRNN (5) 57](#_Toc175487496)

[5.24.Σιτγμιότυπο κλάσης KnapsackRNN (6) 57](#_Toc175487497)

[5.25.Σιτγμιότυπο κώδικα prepare\_rnn\_data 58](#_Toc175487498)

[5.26.Σιτγμιότυπο κώδικα prepare\_rnn\_data (2) 58](#_Toc175487499)

[5.27.Σιτγμιότυπο κώδικα prepare\_rnn\_data (3) 59](#_Toc175487500)

[5.28.Σιτγμιότυπο κώδικα train\_rnn 59](#_Toc175487501)

[5.29.Σιτγμιότυπο κώδικα train\_rnn (2) 60](#_Toc175487502)

[5.30.Σιτγμιότυπο κώδικα train\_rnn (3) 60](#_Toc175487503)

[5.31.Σιτγμιότυπο κώδικα train\_rnn (4) 61](#_Toc175487504)

[5.32.Σιτγμιότυπο κώδικα train\_rnn (5) 61](#_Toc175487505)

[5.33.Σιτγμιότυπο κώδικα train\_rnn (6) 62](#_Toc175487506)

[5.34.Σιτγμιότυπο κώδικα train\_rnn (7) 62](#_Toc175487507)

[5.35.Σιτγμιότυπο κώδικα train\_rnn (7) 62](#_Toc175487508)

[5.36.Σιτγμιότυπο κώδικα train\_rnn (8) 63](#_Toc175487509)

[5.37.Σιτγμιότυπο κλάσης KnapsackDQN 63](#_Toc175487510)

[5.38.Σιτγμιότυπο κλάσης KnapsackDQN (2) 64](#_Toc175487511)

[5.39.Σιτγμιότυπο κλάσης KnapsackDQN (3) 64](#_Toc175487512)

[5.40.Σιτγμιότυπο κλάσης KnapsackDQN (4) 64](#_Toc175487513)

[5.41.Σιτγμιότυπο κώδικα Transition namedtuple 65](#_Toc175487514)

[5.42.Σιτγμιότυπο κλάσης ReplayMemory 65](#_Toc175487515)

[5.43.Σιτγμιότυπο κλάσης ReplayMemory (2) 66](#_Toc175487516)

[5.44.Σιτγμιότυπο κλάσης ReplayMemory (3) 66](#_Toc175487517)

[5.45.Σιτγμιότυπο κλάσης ReplayMemory (4) 66](#_Toc175487518)

[5.46.Σιτγμιότυπο κλάσης DQNAgent 67](#_Toc175487519)

[5.47.Σιτγμιότυπο κλάσης DQNAgent (2) 68](#_Toc175487520)

[5.48.Σιτγμιότυπο κλάσης DQNAgent (3) 69](#_Toc175487521)

[5.49.Σιτγμιότυπο κλάσης DQNAgent (5) 69](#_Toc175487522)

[5.50.Σιτγμιότυπο κλάσης DQNAgent (6) 69](#_Toc175487523)

[5.51 Στιγμιότυπο κλάσης DQNAgent (7) 70](#_Toc175487524)

[5.52Στιγμιότυπο κλάσης DQNAgent (8) 70](#_Toc175487525)

[5.53Στιγμιότυπο κλάσης DQNAgent (9) 70](#_Toc175487526)

[5.54 Στιγμιότυπο κώδικα train\_dqn 71](#_Toc175487527)

[5.55 Στιγμιότυπο κώδικα train\_dqn (2) 71](#_Toc175487528)

[5.56 Στιγμιότυπο κώδικα train\_dqn (3) 72](#_Toc175487529)

[5.57 Στιγμιότυπο κώδικα train\_dqn (4) 72](#_Toc175487530)

[5.58 Στιγμιότυπο κώδικα train\_dqn (5) 73](#_Toc175487531)

[5.59 Στιγμιότυπο κώδικα train\_dqn (6) 73](#_Toc175487532)

[5.60 Στιγμιότυπο κώδικα train\_dqn (7) 74](#_Toc175487533)

[5.61 Στιγμιότυπο κώδικα train\_dqn (8) 74](#_Toc175487534)

[5.62 Στιγμιότυπο κώδικα train\_dqn (9) 74](#_Toc175487535)

[5.63 Στιγμιότυπο κώδικα solve\_with\_knapsack\_dqn 75](#_Toc175487536)

[5.64 Στιγμιότυπο κώδικα solve\_with\_knapsack\_dqn (2) 75](#_Toc175487537)

[5.65 Στιγμιότυπο κώδικα solve\_with\_knapsack\_dqn (3) 76](#_Toc175487538)

[5.66 Στιγμιότυπο κώδικα solve\_with\_knapsack\_dqn (4) 76](#_Toc175487539)

[5.67 Στιγμιότυπο κλάσης KnapsackVisualization 76](#_Toc175487540)

[5.68 Στιγμιότυπο κλάσης KnapsackVisualization (2) 77](#_Toc175487541)

[5.69 Στιγμιότυπο κλάσης KnapsackVisualization (3) 77](#_Toc175487542)

[5.70 Στιγμιότυπο κλάσης KnapsackVisualization (4) 77](#_Toc175487543)

[5.71 Στιγμιότυπο κλάσης KnapsackVisualization (5) 78](#_Toc175487544)

[5.72 Στιγμιότυπο κλάσης KnapsackVisualization (6) 78](#_Toc175487545)

[5.73 Στιγμιότυπο κλάσης KnapsackVisualization (7) 78](#_Toc175487546)

[5.74 Στιγμιότυπο κλάσης KnapsackVisualization (8) 78](#_Toc175487547)

[5.75 Εικόνα Άδειου Σαικιδίου, generated by DALL-E, OpenAI 79](#_Toc175487548)

[5.76 Στιγμιότυπο κλάσης KnapsackVisualization (9) 79](#_Toc175487549)

[5.77 Μαύρος κύκλος – μη επιλεγμένα αντικείμενα 79](#_Toc175487550)

[5.78 Μάυρος κύκλος με πράσινο περίγραμμα - επιλεγμένα αντικείμενα 79](#_Toc175487551)

[5.79 Στιγμιότυπο κλάσης KnapsackVisualization (10) 80](#_Toc175487552)

[5.80 Στιγμιότυπο κώδικα draw\_grid 80](#_Toc175487553)

[5.81 Στιγμιότυπο κώδικα draw\_grid (2) 80](#_Toc175487554)

[5.82 Στιγμιότυπο κώδικα draw\_grid (3) 80](#_Toc175487555)

[5.83 Στιγμιότυπο κώδικα update 81](#_Toc175487556)

[5.84 Στιγμιότυπο κώδικα update (2) 81](#_Toc175487557)

[5.85 Στιγμιότυπο κώδικα update (3) 81](#_Toc175487558)

[5.86 Στιγμιότυπο κώδικα update (4) 81](#_Toc175487559)

[5.87 Στιγμιότυπο κώδικα update (5) 82](#_Toc175487560)

[5.88 Στιγμιότυπο κώδικα change\_color\_and\_move 82](#_Toc175487561)

[5.89 Στιγμιότυπο κώδικα update (6) 82](#_Toc175487562)

[5.90 Στιγμιότυπο κώδικα position\_plot\_window 83](#_Toc175487563)

[5.91 Στιγμιότυπο κώδικα position\_plot\_window (2) 83](#_Toc175487564)

[5.92 Στιγμιότυπο κώδικα position\_plot\_window (3) 83](#_Toc175487565)

[5.93 Στιγμιότυπο κώδικα position\_plot\_window (4) 83](#_Toc175487566)

[5.94 Στιγμιότυπο κώδικα solve\_knapsack 84](#_Toc175487567)

[5.95 Στιγμιότυπο κώδικα solve\_knapsack (2) 84](#_Toc175487568)

[5.96 Στιγμιότυπο κώδικα solve\_knapsack (3) 84](#_Toc175487569)

[5.97 Στιγμιότυπο κώδικα solve\_knapsack (4) 84](#_Toc175487570)

[5.98 Στιγμιότυπο κώδικα solve\_knapsack (5) 84](#_Toc175487571)

[5.99 Στιγμιότυπο κώδικα solve\_knapsack (6) 85](#_Toc175487572)

[5.100 Στιγμιότυπο κώδικα solve\_knapsack (7) 85](#_Toc175487573)

[5.101 Στιγμιότυπο κώδικα solve\_knapsack (8) 85](#_Toc175487574)

[5.102 Στιγμιότυπο κώδικα test\_new\_sample 86](#_Toc175487575)

[5.103 Στιγμιότυπο κώδικα test\_new\_sample (2) 86](#_Toc175487576)

[5.104 Στιγμιότυπο κώδικα test\_new\_sample (3) 86](#_Toc175487577)

[5.105 Στιγμιότυπο κώδικα test\_new\_sample (4) 87](#_Toc175487578)

[5.106 Στιγμιότυπο κώδικα create\_gui 87](#_Toc175487579)

[5.107 Στιγμιότυπο κώδικα start\_solving 88](#_Toc175487580)

[5.108 Στιγμιότυπο κώδικα create\_gui (2) 88](#_Toc175487581)

[5.109 Στιγμιότυπο κώδικα create\_gui (3) 88](#_Toc175487582)

[5.110 Στιγμιότυπο κώδικα create\_gui (4) 88](#_Toc175487583)

[5.111 Στιγμιότυπο κώδικα mainloop 88](#_Toc175487584)

Συντομογραφίες

KP Knapsack Problem

NN Neural Networks

DNN Deep Neural Network

RNN Recurrent Neural Network

DQN Deep Q-Learning Network

CO Combinatorial Optimization

# Το πρόβλημα του σακιδίου

## Εισαγωγή

Αυτή η εργασία ερευνά και αναλύει το πρόβλημα του σακιδίου βασιζόμενη σε Νευρωνικά Δίκτυα όλων των τύπων και σε Αλγορίθμους Βαθιάς Μάθησης. Τα προβλήματα με το σακίδιο έχουν μελετηθεί εντατικά και επειδή προκύπτουν ως υποπροβλήματα σε διάφορα προβλήματα προγραμματισμού ακεραίων και μπορεί να αντιπροσωπεύουν πολλές πρακτικές καταστάσεις. Οι πιο τυπικές εφαρμογές είναι στον προϋπολογισμό κεφαλαίων και στην εργοστασιακή παραγωγή. Το πρόβλημα του σακιδίου είναι μια πρόκληση βελτιστοποίησης που βρίσκει εφαρμογές σε διάφορους τομείς, όπως η κατανομή πόρων, η χρηματοδότηση και η εφοδιαστική αλυσίδα [1]. Περιλαμβάνει την επιλογή ενός υποσυνόλου αντικειμένων με στόχο τη μεγιστοποίηση της συνολικής αξίας, δεδομένου ενός περιορισμού για το συνολικό βάρος. Αυτό το συνδυαστικό πρόβλημα βελτιστοποίησης είναι γνωστό για τη NP-δύσκολη φύση του, καθιστώντας υπολογιστικά δύσκολη την αποτελεσματική εύρεση ακριβούς λύσης. Οι παραδοσιακές προσεγγίσεις για την επίλυση του προβλήματος knapsack βασίζονται συχνά σε δυναμικό προγραμματισμό ή αλγορίθμους, οι οποίοι μπορεί να μην είναι κατάλληλοι για μεγάλες περιπτώσεις ή πολύπλοκα σενάρια. Τα τελευταία χρόνια, υπάρχει αυξανόμενο ενδιαφέρον για την αξιοποίηση της ισχύος των νευρωνικών δικτύων και των αλγορίθμων βαθιάς μάθησης για την αντιμετώπιση προβλημάτων συνδυαστικής βελτιστοποίησης, συμπεριλαμβανομένου του προβλήματος knapsack.

## Τι είναι το πρόβλημα του σακιδίου;

Ας φανταστούμε ότι ένας ταξιδιώτης πρέπει να γεμίσει το σακίδιό του επιλέγοντας ανάμεσα σε διάφορα αντικείμενα αυτά που θα του προσφέρουν τη μέγιστη άνεση. Αυτό το πρόβλημα του σακιδίου μπορεί να διατυπωθεί μαθηματικά, αριθμώντας τα αντικείμενα από το 1 έως το n και εισάγοντας ένα διάνυσμα δυαδικών μεταβλητών με την εξής σημασία:

Τότε, αν το είναι ένα μέτρο της άνεσης που προσφέρει το αντικείμενο , το το μέγεθός του, και το το μέγεθος του σακιδίου, το πρόβλημά μας θα είναι να επιλέξουμε, ανάμεσα σε όλα τα δυαδικά διανύσματα που ικανοποιούν τον περιορισμό:

αυτό που μεγιστοποιεί την συνάρτηση

## Οι εφαρμογές του knapsack problem

Οι εφαρμογές του δυναμικού και στοχαστικού αντίστοιχου προβλήματος του σακιδίου περιλαμβάνουν σε τομείς όπως τη βιομηχανία μεταφορών, τα πλοία, τα τρένα, τα αεροσκάφη ή φορτηγά που συχνά μεταφέρουν φορτία για διαφορετικούς πελάτες. Τα αιτήματα μεταφοράς φθάνουν στοχαστικά με την πάροδο του χρόνου και προσφέρονται ή διαπραγματεύονται τιμές για τη μεταφορά φορτίων. Εάν ένα φορτίο γίνει αποδεκτό, προκύπτουν έξοδα για την παραλαβή και το χειρισμό του φορτίου, καθώς και για τις σχετικές διοικητικές δραστηριότητες. Το κόστος αυτό είναι συγκεκριμένο για το φορτίο και μπορούν να αφαιρεθούν από την τιμή για να δώσουν την «ανταμοιβή» του του φορτίου. Τα φορτία μπορούν να έχουν διαφορετικά μεγέθη -όπως πακέτα- ή το ίδιο μέγεθος-όπως τα εμπορευματοκιβώτια. Συχνά υπάρχει ένα σταθερό χρονοδιάγραμμα για τη μετακίνηση των οχημάτων και μια προθεσμία μετά την οποία τα φορτία δεν μπορούν να γίνουν δεκτά για μια συγκεκριμένη αποστολή. Ακόμη και όταν δεν υπάρχει σταθερό χρονοδιάγραμμα, υπάρχει κίνητρο για την ενοποίηση και την αποστολή φορτίων με μεγάλη συχνότητα, για να διατηρηθούν σύντομοι χρόνοι παράδοσης και να μεγιστοποιηθεί το ποσοστό με το οποίο αποκτώνται έσοδα με τη δεδομένη επένδυση σε κεφάλαιο και το κόστος εργασίας [1]. Αυτό το κίνητρο μπορεί να μοντελοποιηθεί με ένα προεξοφλητικό επιτόκιο και ένα κόστος αναμονής ή κόστος διακράτησης ανά μονάδα χρόνου που προκύπτει μέχρι να αποσταλεί η αποστολή. Το κόστος αναμονής μπορεί να είναι σταθερό ή μπορεί να εξαρτάται από τον αριθμό των φορτίων που έχουν γίνει δεκτά, αλλά δεν έχουν ακόμη αποστέλλονται. Ο αποστολέας μπορεί να αποφασίσει να αποστείλει ένα όχημα ανά πάσα στιγμή πριν από την προθεσμία. Υπάρχει επίσης ένα κόστος αποστολής και μεταφοράς που προκύπτει για το σύνολο της αποστολής, το οποίο μπορεί να εξαρτάται από τον αριθμό των φορτίων της αποστολής.

Ένας χρονοπρογραμματιστής ενός επεξεργαστή παρτίδων πρέπει να προγραμματίσει τις εργασίες με τυχαίες απαιτήσεις χωρητικότητας και ανταμοιβές καθώς αυτές φθάνουν με την πάροδο του χρόνου. Τα σταθερά χρονοδιαγράμματα ή οι δεσμεύσεις των πελατών οδηγούν σε προθεσμίες. Η πίεση για την αύξηση της χρήσης του εξοπλισμού και της εργασίας και να διατηρηθεί ένα υψηλό επίπεδο εξυπηρέτησης των πελατών, οδηγούν σε κόστος αναμονής ανά μονάδα χρόνου. Το κόστος λειτουργίας του επεξεργαστή παρτίδων μπορεί να εξαρτηθεί από τον αριθμό των εργασιών στη δέσμη.

Ένας κτηματομεσίτης που πωλεί νέα διαμερίσματα λαμβάνει προσφορές στοχαστικά με την πάροδο του χρόνου και μπορεί να θέλει να πουλήσει τa περιουσιακά στοιχεία πριν από το χειμώνα ή πριν από το νέο φορολογικό έτος [1]. Ως εκ τούτου, ο εκπρόσωπος αντιμετωπίζει μια προθεσμία, ενδεχομένως με αξία διάσωσης για τα απούλητα περιουσιακά στοιχεία. Υπάρχει επίσης ένα κόστος ευκαιρίας που σχετίζεται με το κεφάλαιο που δεσμεύεται στα απούλητα περιουσιακά στοιχεία, και οι φόροι ακίνητης περιουσίας, οι οποίοι προκαλούν ένα κόστος αναμονής ανά μονάδα χρόνο.

Ένας επενδυτής που επιθυμεί να επενδύσει ένα συγκεκριμένο ποσό σε κεφάλαια και να αντιμετωπίζει ένα παρόμοιο πρόβλημα. Στον επενδυτή παρουσιάζεται με επενδυτικά σχέδια με τυχαίους χρόνους άφιξης, απαιτήσεις χρηματοδότησης και αποδόσεις. Η ευκαιρία του αχρησιμοποίητου κεφαλαίου αντιπροσωπεύεται από ένα κόστος αναμονής ανά μονάδα χρόνου, και ο στόχος είναι να μεγιστοποιηθεί η αναμενόμενη αξία που κερδίζεται από την επένδυση των κεφαλαίων

## Η πολυπλοκότητα του Προβλήματος του Σακιδίου

Η υπολογιστική πολυπλοκότητα του Προβλήματος του Σακιδίου είναι αυτό που κάνει το πρόβλημα τόσο ενδιαφέρον για μελέτη. Για την ταξινόμηση της δυσκολίας ή της πολυπλοκότητας των αλγοριθμικών προβλημάτων, μπορούμε να δούμε μια σειρά τάξεων πολυπλοκότητας από P έως NP-Hard [2]. Αυτή η ταξινόμηση μιας δυσκολίας προβλημάτων εισήχθη για πρώτη φορά από τον Stephen Cook [3] και εξαρτάται από τη χρονική κλίμακα εύρεσης λύσης σε ένα δεδομένο πρόβλημα. Τα πιο «εύκολα» προβλήματα στην κλίμακα ταξινομούνται ως P, πράγμα που δείχνει ότι μια λύση μπορεί να βρεθεί σε πολυωνυμικό χρόνο. Σε αυτήν την κατηγορία, ο χρόνος που εκτελείται ένας αλγόριθμος, T(s) είναι η συνάρτηση του s, το μέγεθος του προβλήματος και η σχέση τους μπορούν να εκφραστούν ως εξής [2]: T(s) = O(sk ) όπου k είναι κάποιος ακέραιος αριθμός. Προβλήματα, όπου η σχέση μεταξύ του μεγέθους του προβλήματος και του χρόνου είναι μη ντετερμινιστικό πολυώνυμο είναι γνωστά ως NP. Στην περίπτωση του kp ανήκει σε ειδική κατηγορία προβλημάτων εντός NP, γνωστή ως NP-Complete. Αυτό σημαίνει, ότι ενώ βρίσκουμε μια λύση επί του παρόντος δεν είναι δυνατή η επίλυση του προβλήματος σε πολυωνυμικό χρόνο, ωστόσο μπορεί κανείς να επαληθεύσει ότι μια λύση είναι σωστή σε πολυωνυμικό χρόνο (δεδομένης της λύσης). Λογικά, NP-Complete προβλήματα είναι τόσο δύσκολα όσο τα δυσκολότερα προβλήματα στο NP, αλλά όχι τόσο δύσκολα όσο τα προβλήματα NP-Hard (που είναι η πιο δύσκολη κατηγορία προβλημάτων και είναι τουλάχιστον τόσο δύσκολα όσο το NP-Complete ή πιο δύσκολα).

Τα προβλήματα αυτά χαρακτηρίζονται από την κατανομή των περιορισμένων πόρων σε ανταγωνιστικά αντικείμενα που φτάνουν τυχαία με την πάροδο του χρόνου. Τα αντικείμενα συνδέονται με απαιτήσεις πόρων καθώς και με ανταμοιβές, οι οποίες μπορεί να περιλαμβάνουν οποιοδήποτε συγκεκριμένο κόστος που προκύπτει. Συνήθως οι χρόνοι άφιξης, οι απαιτήσεις πόρων και οι ανταμοιβές είναι άγνωστες πριν από την άφιξη και γίνονται γνωστές κατά την άφιξη. Τα αντικείμενα που φθάνουν μπορεί να είναι είτε αποδεκτά ή να απορριφθούν. Κίνητρα, όπως μια προθεσμία μετά την οποία δεν μπορεί να γίνει αποδεκτή, εκπτώσεις και ένα κόστος αναμονής ανά μονάδα χρόνου χρησιμεύουν για την ενθάρρυνση της έγκαιρης αποδοχή των αντικειμένων. Το πρόβλημα μπορεί να σταματήσει σε οποιαδήποτε χρόνο πριν ή κατά τη λήξη της προθεσμίας. Μπορεί επίσης να υπάρχει ένα κόστος που σχετίζεται με την ομάδα των αποδεκτών αντικειμένων στο σύνολό της, ή μια αξία διάσωσης για τους αχρησιμοποίητους πόρους, η οποία μπορεί να εξαρτάται από την ποσότητα των πόρων που κατανέμονται στα αποδεκτά στοιχεία. Ένας τυπικός στόχος είναι η μεγιστοποίηση της αναμενόμενης συνολικής αξίας (ανταμοιβές μείον κόστος). Τα προβλήματα αυτής της γενικής φύσης ονομάζονται Δυναμικό και Στοχαστικό Πρόβλημα Σακιδίου (Dynamic and Stochastic Knapsack Problem).

## Παραλλαγές και επεκτάσεις του προβλήματος του σακιδίου

Τα αντικείμενα που συζητήθηκαν στην ενότητα 1.2 αναφέρονται γενικά ως «είδη», και ο συνολικός αριθμός τους σημειώνεται με το γράμμα . Η αξία και το βάρος του κάθε είδους αναφέρονται αντίστοιχα ως «κέρδος» και «βάρος» και σημειώνονται με και . Υπάρχουν προβλήματα όπου πρέπει να γεμίσει ένα μόνο σακίδιο με τον βέλτιστο συνδυασμό ειδών, και η χωρητικότητα του σακιδίου αναφέρεται ως . Υπάρχουν επίσης προβλήματα όπου υπάρχουν περισσότερα από ένα σακίδια. Συνήθως υποτίθεται, όπως αναφέρεται στη βιβλιογραφία, ότι τα κέρδη, τα βάρη και οι χωρητικότητες είναι θετικοί ακέραιοι αριθμοί. Παρ' όλα αυτά, τα αποτελέσματα μπορούν εύκολα να προσαρμοστούν για πραγματικές τιμές και, στις περισσότερες περιπτώσεις, για μη θετικές τιμές. Το αρχικό πρόβλημα της ενότητας 1.2,

είναι γνωστό ως το πρόβλημα σακιδίου 0-1 (**0-1 Knapsack Problem**)

Ας εξετάσουμε πρόβλημα του αποστολέα της εταιρείας φορτίου. Το κέρδος που προκύπτει από την αποδοχή ενός πακέτου μπορεί να είναι ευθέως ανάλογο με το βάρος του. Σε αυτή την περίπτωση, η βέλτιστη φόρτιση του επιπέδου επιτυγχάνεται με την πλήρωση όσο το δυνατόν περισσότερου βάρους στο επίπεδο ή ισοδύναμη ρύθμιση σε (KP). Το πρόβλημα βελτιστοποίησης που προκύπτει είναι γνωστό ως πρόβλημα αθροίσματος υποσυνόλου (**subset** **sum** **problem**, **SSP**) επειδή αναζητούμε ένα υποσύνολο των τιμών με το άθροισμα να είναι όσο το δυνατόν πιο κοντά, αλλά όχι να υπερβαίνει τη δεδομένη τιμή στόχο .

(SSP) maximize

subject to ,

, , …, .

Στο πρόβλημα του αρχικού φορτίου που περιγράφεται παραπάνω, συχνά συμβαίνει ότι δεν διαφέρουν όλα τα πακέτα μεταξύ τους. Ειδικότερα, στην πράξη μπορεί να δοθεί ένας αριθμός πανομοιότυπων αντιγράφων για κάθε αντικείμενο προς μεταφορά. Εάν πρέπει είτε να δεχθούμε ένα αίτημα για όλα τα πακέτα είτε να τα απορρίψουμε όλα, τότε μπορούμε να δημιουργήσουμε ένα τεχνητό αίτημα με βάρος που δημιουργεί ένα κέρδος . Εάν είναι δυνατό να επιλέξετε επίσης μόνο ένα υποσύνολο των στοιχείων από ένα αίτημα, μπορούμε είτε να αναπαραστήσουμε κάθε μεμονωμένο πακέτο με μια δυαδική μεταβλητή ή πιο αποτελεσματικά να αναπαραστήσουμε ολόκληρο το σύνολο των πανομοιότυπων πακέτων με μια ακέραια μεταβλητή που υποδεικνύει τον αριθμό συσκευασιών αυτού του τύπου που τοποθετούνται στο αεροπλάνο. Σε αυτήν την περίπτωση, ο αριθμός των μεταβλητών είναι ίσος με τον αριθμό των διαφορετικών πακέτων αντί του συνολικού αριθμού των πακέτων. Αυτό μπορεί να μειώσει σημαντικά το μέγεθος του μοντέλου εάν οι αριθμοί είναι σχετικά μεγάλοι. Τυπικά, ο περιορισμός (1.3) στο (KP) αντικαθίσταται από:

, ακέραιος, , …, . (1.4)

Το πρόβλημα που προκύπτει ονομάζεται πρόβλημα περιορισμένου σακιδίου (**bounded** **knapsack** **problem**, **BKP**). Μια ειδική παραλλαγή του είναι το πρόβλημα του απεριόριστου σακιδίου (**unbounded** **knapsack** **problem**, **UKP**). Είναι επίσης γνωστό ως πρόβλημα σακιδίου ακέραιου αριθμού όπου αντί για σταθερό αριθμό bj δίνεται ένας πολύ μεγάλος ή άπειρος αριθμός πανομοιότυπων αντιγράφων κάθε στοιχείου. Σε αυτήν την περίπτωση, ο περιορισμός (1.3) στο (KP) απλώς αντικαθίσταται από:

, ακέραιος, , …, . (1.5)

Προχωρώντας σε διαφορετική κατεύθυνση, εξετάζουμε ξανά το παραπάνω πρόβλημα φορτίου και πλέον λαμβάνουμε υπόψη όχι μόνο τον περιορισμό βάρους αλλά και τον περιορισμένο διαθέσιμο χώρο για τη μεταφορά των δεμάτων. Για πρακτικούς λόγους λαμβάνεται υπόψη μόνο ο όγκος των συσκευασιών και όχι τα διαφορετικά σχήματά τους. Δηλώνοντας το βάρος κάθε είδους με και τον όγκο του με και εισάγοντας την χωρητικότητα βάρους του επιπέδου ως και το άνω όριο στον όγκο ως , μπορούμε να διατυπώσουμε το πρόβλημα εκτεταμένου φορτίου αντικαθιστώντας τον περιορισμό (1.2) στο (KP) με οι δύο ανισότητες:

,

.

Η προφανής γενίκευση αυτής της προσέγγισης, όπου εισάγεται το αντί για δύο ανισότητες, αποδίδει το πρόβλημα του -διάστατου σακιδίου ή το πρόβλημα του πολυδιάστατου σακιδίου που ορίζεται επίσημα από:

(d-KP) maximize

subject to , , …, ,

, , …, .

Μια άλλη ενδιαφέρουσα παραλλαγή του προβλήματος του φορτίου προκύπτει από την αρχική έκδοση που περιγράφεται παραπάνω, αν σκεφτούμε μια πολύ πολυσύχναστη διαδρομή πτήσης, π.χ. Φρανκφούρτη - Νέα Υόρκη, η οποία πετά με πολλά αεροπλάνα καθημερινά. Σε αυτήν την περίπτωση, ο αποστολέας πρέπει να αποφασίσει για τη φόρτωση ενός αριθμού αεροπλάνων παράλληλα, δηλαδή πρέπει να αποφασιστεί εάν θα αποδεχτεί ένα συγκεκριμένο αίτημα μεταφοράς και στη θετική περίπτωση σε ποιο αεροπλάνο θα τοποθετήσει το αντίστοιχο πακέτο. Οι έννοιες του κέρδους, του βάρους και της ικανότητας παραμένουν αμετάβλητες. Αυτό μπορεί να διατυπωθεί με την εισαγωγή μιας δυαδικής μεταβλητής απόφασης για κάθε συνδυασμό ενός πακέτου με ένα επίπεδο. Εάν υπάρχουν στοιχεία στη λίστα των αιτημάτων μεταφοράς και αεροπλάνα διαθέσιμα σε αυτήν τη διαδρομή, χρησιμοποιούμε δυαδικές μεταβλητές για , …, και , …, με:

. (1.6)

Η μαθηματική διατύπωση προγραμματισμού αυτού του **προβλήματος** **πολλαπλών** **σακιδίων** (**multiple** **knapsack** **problem**, **MKP**) δίνεται από:

(MKP) maximize , (1.7)

subject to , , …, , (1.8)

, , …, , (1.9)

, , …, , , …, .

Ο περιορισμός (1.9) εγγυάται ότι κάθε στοιχείο τοποθετείται το πολύ σε ένα επίπεδο. Εάν οι χωρητικότητες των επιπέδων είναι πανομοιότυπες, μπορούμε εύκολα να απλοποιήσουμε το παραπάνω μοντέλο εισάγοντας μια χωρητικότητα για όλα τα επίπεδα και αντικαθιστώντας τους περιορισμούς (1.8) από:

, , …, , (1.10)

Το τελευταίο μοντέλο αναφέρεται συχνά ως το πρόβλημα του πολλαπλού σακιδίου με τις ίδιες χωρητικότητες (**multiple** **knapsack** **problem** **with** **identical** **capacities**, **MKP-I**). Σημειώστε ότι και για τις δύο εκδόσεις πολλαπλών σακιδίων διερευνήθηκαν επίσης οι παραλλαγές αθροίσματος υποσυνόλου με . Στην περίπτωση αυτή ο στόχος (1.7) αντικαθίσταται από:

maximize . (1.11)

Σε αναλογία με το πρόβλημα του σακιδίου, αυτό ορίζει το πρόβλημα αθροίσματος πολλαπλών υποσυνόλων (**multiple** **subset** **sum** **problem**, **MSSP**) με αυθαίρετες χωρητικότητες που δίνονται από τον περιορισμό (1.8) ή το πρόβλημα πολλαπλών αθροισμάτων υποσυνόλου (**multiple subset sum problem with identical capacities, MSSP-I**) με ίδιες χωρητικότητες και περιορισμό (1.10).

Μια εντελώς διαφορετική παραλλαγή του προβλήματος του φορτίου εμφανίζεται εάν χρησιμοποιείται ένα μόνο αεροπλάνο για τη μεταφορά του εξοπλισμού μιας αποστολής σε ένα έρημο μέρος. Η αποστολή χρειάζεται έναν αριθμό εργαλείων όπως μια λαστιχένια λέμβο, ένα όχημα, ειδικά όργανα κ.λπ. Ωστόσο, κάθε εργαλείο υπάρχει σε διάφορες παραλλαγές όπου η -η παραλλαγή έχει βάρος και τιμή χρησιμότητας . Καθώς το αεροπλάνο μπορεί να μεταφέρει μόνο περιορισμένη χωρητικότητα , ο στόχος είναι να επιλέξετε μία παραλλαγή κάθε εργαλείου έτσι ώστε η συνολική τιμή χρησιμότητας να μεγιστοποιείται χωρίς να υπερβαίνει τον περιορισμό χωρητικότητας.

Αυτό το πρόβλημα μπορεί να εκφραστεί ως το ακόλουθο πρόβλημα σακιδίου πολλαπλής επιλογής (**multiple-choice knapsack problem, MCKP**). Ας υποθέσουμε ότι το είναι το σύνολο των διαφορετικών παραλλαγών του εργαλείου . Χρησιμοποιώντας τις μεταβλητές απόφασης για να δηλώσουμε εάν η παραλλαγή επιλέχθηκε από το σύνολο , εμφανίζεται το ακόλουθο μοντέλο:

(MCKP) maximize ,

subject to ,

, , …, , (1.12)

, , …, , .

Ο περιορισμός (1.12) διασφαλίζει ότι επιλέγεται ακριβώς ένα εργαλείο από κάθε κλάση. Μια άλλη παραλλαγή εμφανίζεται εάν ένα στοιχείο έχει αντίστοιχο κέρδος και ένα πρόσθετο κέρδος εξαργυρώνεται μόνο εάν το στοιχείο μεταφέρεται μαζί με ένα άλλο στοιχείο , το οποίο μπορεί να αντικατοπτρίζει πόσο καλά ταιριάζουν τα δεδομένα μεταξύ τους. Αυτό το πρόβλημα εκφράζεται ως πρόβλημα τετραγωνικού σακιδίου (**quadratic knapsack problem, QKP**) το οποίο ορίζεται επίσημα ως εξής:

(QKP) maximize ,

subject to , (1.13)

, , …, .

Άλλη μια εκδοχή του προβλήματος του σακιδίου, είναι το Πρόβλημα του Δισδιάστατου Σακιδίου (**Bilevel Knapsack Problem, BKP**). Το Πρόβλημα Δισδιάστατου Σακιδίου (BKP) είναι ένα ιεραρχικό πρόβλημα βελτιστοποίησης στο οποίο το εφικτό σύνολο καθορίζεται από το σύνολο των βέλτιστων λύσεων του παραμετρικού Προβλήματος Σακιδίου (KP). Το BKP μπορεί να διατυπωθεί ως εξής:

,

,

όπου τα διανύσματα και (αντίστοιχα , , ) είναι διαστάσεων (αντίστοιχα ). Το διάνυσμα είναι διαστάσεων . Οι συντελεστές των πινάκων και είναι ακέραιοι. Υποθέτουμε ότι οι συντελεστές των διανυσμάτων και , καθώς και το δεύτερο μέλος , είναι θετικοί ακέραιοι. Το διάνυσμα (αντίστοιχα ) διαστάσεων (αντίστοιχα ) αντιπροσωπεύει τις μεταβλητές του ηγέτη (αντίστοιχα του ακόλουθου). Η συνάρτηση (αντίστοιχα ) αντιστοιχεί στον στόχο του ηγέτη (αντίστοιχα του ακόλουθου).

Εκτός από τις κλασικές παραλλαγές του (KP) που έχουν εισαχθεί σε αυτήν την ενότητα, υπάρχουν πολλές άλλες εκδοχές του προβλήματος.

## Πρόβλημα μίας χωρητικότητας έναντι όλων των χωρητικοτήτων

Μια φυσική επέκταση του προβλήματος του σακιδίου είναι η επίλυση του προβλήματος όχι μόνο για μια δεδομένη χωρητικότητα , αλλά για όλες τις χωρητικότητες έως ένα δεδομένο ανώτερο όριο . Θα ονομάσουμε αυτό το πρόβλημα ως πρόβλημα σακιδίου πλήρους χωρητικότητας.

Αυτή η παραλλαγή του προβλήματος έχει παραλειφθεί στη βιβλιογραφία, αλλά είναι εξίσου σχετική με την επίλυση του προβλήματος της συντομότερης διαδρομής όλων των ζευγών σε αντίθεση με το πρόβλημα της συντομότερης διαδρομής μιας πηγής. Σε πολλά προβλήματα σχεδιασμού, η ακριβής χωρητικότητα δεν είναι γνωστή εκ των προτέρων, αλλά μπορεί να αποτελέσει αντικείμενο διαπραγμάτευσης με βάση τις προτεινόμενες λύσεις.

Για να συνεχίσουμε το προηγούμενο παράδειγμά μας με την αεροπορική εταιρεία φορτίου, μπορεί να παρατηρήσουμε ότι η χωρητικότητα ενός αεροπλάνου εξαρτάται από την ποσότητα καυσίμου στο αεροπλάνο. Η μεταφορά περισσότερου φορτίου σημαίνει ότι χρειάζεται περισσότερο καύσιμο, αλλά το καύσιμο μετράει επίσης στο συνολικό βάρος φορτίου. Δεδομένου ότι η συνάρτηση καυσίμου έναντι φορτίου δεν είναι γραμμική, μπορεί να χρειαστεί να λύσουμε το πρόβλημα του σακιδίου όλων των χωρητικοτήτων για όλες τις χωρητικότητες μέχρι ένα δεδομένο ανώτατο όριο του βάρους φορτίου. Για κάθε χωρητικότητα υπολογίζουμε στη συνέχεια τις απαιτήσεις καυσίμου και αφαιρούμε το αντίστοιχο κόστος καυσίμου από το κέρδος. Η βέλτιστη λύση βρίσκεται ως το μέγιστο των τελικών κερδών.

Μια φυσική μέθοδος για την επίλυση του προβλήματος του σακιδίου όλων των χωρητικοτήτων είναι απλώς η επίλυση (KP) για κάθε χωρητικότητα , …, . Ωστόσο, ειδικοί αλγόριθμοι μπορούν να επωφεληθούν από το γεγονός ότι το πρόβλημα επιλύεται για πολλές χωρητικότητες, όπως ο συνολικός χρόνος εκτέλεσης της επίλυσης του προβλήματος του σακιδίου όλων των χωρητικοτήτων είναι μικρότερος από την επίλυση των μεμονωμένων προβλημάτων.

## Υποθέσεις για τα δεδομένα εισόδου

Για να αποφύγουμε ασήμαντες καταστάσεις και κουραστικές σκέψεις για άσκοπες υποπεριπτώσεις, θα επιβάλουμε μια σειρά από υποθέσεις στα δεδομένα εισόδου του προβλήματος του σακιδίου, οι οποίες θα ισχύουν σε όλα τα κεφάλαια. Ωστόσο, όπως περιγράφεται παρακάτω, αυτό δεν θα οδηγήσει σε απώλεια γενικότητας. Οι ανάλογες παραδοχές ισχύουν επίσης για τις περισσότερες από τις παραλλαγές του (KP).

Πρώτα από όλα, θα ληφθούν υπόψη μόνο προβλήματα με τουλάχιστον δύο στοιχεία, δηλαδή υποθέτουμε . Προφανώς, η περίπτωση αντιπροσωπεύει μια ενιαία δυαδική απόφαση που μπορεί να ληφθεί με απλή αξιολόγηση των δύο εναλλακτικών. Δύο άλλες απλές παραδοχές αφορούν τους συντελεστές στάθμισης. Φυσικά, δεν μπορούμε ποτέ να συσκευάσουμε ένα αντικείμενο στο σακίδιο αν το βάρος του υπερβαίνει τη χωρητικότητα. Ως εκ τούτου, μπορούμε να υποθέσουμε:

, , …, , (1.14)

γιατί διαφορετικά θα έπρεπε να ορίσουμε στο 0 οποιαδήποτε δυαδική μεταβλητή που αντιστοιχεί σε ένα στοιχείο που παραβιάζει το (1.14). Εάν από την άλλη, όλα τα αντικείμενα μαζί χωρούν στο σακίδιο, το πρόβλημα λύνεται επιπόλαια πακετάροντάς τα όλα. Επομένως, υποθέτουμε:

. (1.15)

Διαφορετικά, θα ορίζαμε για , …, . Χωρίς απώλεια γενικότητας μπορούμε να υποθέσουμε ότι όλα τα κέρδη και τα βάρη είναι θετικά:

, , , …, , (1.16)

Εάν δεν συμβαίνει αυτό, μπορούμε να μετατρέψουμε την περίπτωση για να ικανοποιήσει το (1.16) ως εξής. Ανάλογα με το πρόσημο των συντελεστών ενός δεδομένου στοιχείου λαμβάνει χώρα μία από τις ακόλουθες ενέργειες (εξαιρουμένης της περίπτωσης ενός ασήμαντου στοιχείου με , όπου το μπορεί να επιλεγεί αυθαίρετα).

1. και : Ορίστε . Εφόσον το στοιχείο δεν αυξάνει την αριστερή πλευρά του περιορισμού χωρητικότητας (1.2), αλλά πιθανώς αυξάνει την αντικειμενική συνάρτηση, είναι πάντα καλύτερο να συσκευάζετε το στοιχείο αντί να το αφήνετε μη-συσκευασμένο.
2. και : Ορίστε . Η συσκευασία του αντικειμένου στο σακίδιο δεν αυξάνει την αντικειμενική λειτουργία ούτε αυξάνει τη διαθέσιμη χωρητικότητα. Ως εκ τούτου, θα είναι πάντα καλύτερο να μην συσκευάζετε αυτό το αντικείμενο.
3. και : Αυτή είναι μια πιο ενδιαφέρουσα περίπτωση που μπορεί να θεωρηθεί ως μια πιθανότητα θυσίας κάποιου κέρδους προκειμένου να αυξηθεί η διαθέσιμη χωρητικότητα, «δημιουργώντας έτσι χώρο» για το βάρος ενός «πιο ελκυστικού» αντικειμένου. Μπορεί να αντιμετωπιστεί με την ακόλουθη κατασκευή. Ας υποθέσουμε ότι τα στοιχεία που ανήκουν στις δύο παραπάνω περιπτώσεις έχουν αντιμετωπιστεί και ως εκ τούτου έχουν εξαλειφθεί από το . Τώρα το σύνολο των στοιχείων μπορεί να χωριστεί σε και . Για όλα τα ορίστε και . Επιπλέον, ας εισαγάγουμε τη δυαδική μεταβλητή ως συμπλήρωμα του με την έννοια:

. (1.17)

Τώρα μπορούμε να διατυπώσουμε το ακόλουθο τυπικό πρόβλημα σακιδίου με θετικούς συντελεστές. Μπορεί να ερμηνευθεί ως η τοποθέτηση όλων των αντικειμένων από το σε ένα μεγεθυσμένο σακίδιο πριν από τη βελτιστοποίηση και, στη συνέχεια, ένα θετικό κέρδος καταναλώνοντας μια θετική χωρητικότητα εάν , δηλ. εάν το αντικείμενο αποσυσκευαστεί ξανά από το σακίδιο.

maximize ,

subject to ,

, , , .

Ένα μάλλον λεπτό σημείο είναι το ζήτημα των ορθολογικών συντελεστών. Πράγματι. τα περισσότερα σχολικά βιβλία απαλλάσσονται από αυτήν την περίπτωση, όπου ορισμένες ή όλες οι τιμές εισόδου είναι μη ακέραιες, με την ασήμαντη πρόταση ότι πολλαπλασιάζοντας με έναν κατάλληλο παράγοντα, π.χ. με το μικρότερο κοινό πολλαπλάσιο των παρονομαστών, αν οι τιμές δίνονται ως κλάσματα ή με κατάλληλη ισχύ του 10, μετατρέπει τα δεδομένα σε ακέραιους αριθμούς. Σαφώς, αυτό μπορεί να μετατρέψει ακόμη και ένα πρόβλημα μέτριου μεγέθους σε ένα μάλλον δυσάρεστο πρόβλημα με τεράστιους συντελεστές. Το μέγεθος των τιμών εισόδου έχει σημαντική αρνητική επίδραση στην απόδοση πολλών αλγορίθμων. Ωστόσο, ορισμένες μέθοδοι, ειδικά οι μέθοδοι διακλάδωσης και σύνδεσης, μπορούν να προσαρμοστούν σχετικά εύκολα για να προσαρμόσουν τις ορθολογικές τιμές εισόδου. Επίσης, οι αλγόριθμοι προσέγγισης συνήθως δεν επηρεάζονται από τις τιμές των συντελεστών αφού σε κάθε περίπτωση γίνεται κάποιου είδους κλιμάκωση των συντελεστών. Όλοι οι αλγόριθμοι δυναμικού προγραμματισμού λειτουργούν επίσης καλά, εφόσον τουλάχιστον είτε τα κέρδη είτε τα βάρη είναι ακέραιοι.

## Απόδοση Αλγορίθμων

Ο κύριος στόχος της μελέτης προβλημάτων σακιδίου είναι η ανάπτυξη μεθόδων λύσης, δηλαδή αλγορίθμων, που υπολογίζουν μια βέλτιστη ή μια κατά προσέγγιση λύση για κάθε δεδομένη περίπτωση προβλήματος. Σαφώς, δεν είναι όλοι οι αλγόριθμοι που υπολογίζουν μια βέλτιστη λύση ισοδύναμοι ως προς την απόδοσή τους. Επιπλέον, φαίνεται φυσικό ότι ένας αλγόριθμος που υπολογίζει μόνο κατά προσέγγιση αλλά όχι απαραίτητα βέλτιστες λύσεις θα πρέπει να αντισταθμίσει αυτό το μειονέκτημα κατά κάποιο τρόπο με μια καλύτερη υπολογιστική συμπεριφορά.

Σε αυτό το πλαίσιο, η απόδοση ορίζεται γενικά από τον χρόνο εκτέλεσης και την ποσότητα της μνήμης του υπολογιστή, δηλαδή του χώρου, που απαιτείται για την επίλυση του δεδομένου προβλήματος. Μια άλλη σημαντική πτυχή θα ήταν το επίπεδο δυσκολίας ενός αλγορίθμου, επειδή οι «εύκολες» μέθοδοι θα προτιμηθούν σαφώς από τους πολύ περίπλοκους αλγόριθμους που είναι πιο δαπανηροί στην εφαρμογή τους. Ωστόσο, αυτή η πτυχή είναι πολύ δύσκολο να ποσοτικοποιηθεί, καθώς το κόστος υλοποίησης εξαρτάται από πολλούς παράγοντες όπως η εμπειρία ή το υπολογιστικό περιβάλλον. Από από την περιγραφή των αλγορίθμων θα πρέπει να είναι ξεκάθαρο εάν είναι απλοί ή μάλλον εξαντλητικοί στην εφαρμογή τους.

Προφανώς, είναι ζωτικής σημασίας να έχουμε κάποιο μέτρο για να διακρίνουμε τους «γρηγορότερους» και «αργότερους» αλγόριθμους. Βασικά, υπάρχουν τρεις τρόποι εισαγωγής κριτηρίων για τον χαρακτηρισμό αλγορίθμων. Η πρώτη προσέγγιση θα ήταν η εφαρμογή του αλγόριθμου και η δοκιμή του για διαφορετικές περιπτώσεις προβλημάτων. Φυσικά, οποιαδήποτε σύγκριση προγραμμάτων ηλεκτρονικών υπολογιστών πρέπει να γίνεται με μεγάλη επιμέλεια για να αποφευχθούν παράγοντες σύγχυσης όπως διαφορετικό υλικό, διαφορετικό περιβάλλον λογισμικού και διαφορετική ποιότητα υλοποίησης. Μια σημαντική δυσκολία είναι επίσης η επιλογή των κατάλληλων περιπτώσεων δοκιμής.

Ένας δεύτερος τρόπος θα ήταν η διερεύνηση του μέσου χρόνου εκτέλεσης ενός αλγορίθμου. Ωστόσο, δεν είναι καθόλου ξεκάθαρο τι είδους «μέσο όρο» ψάχνουμε. Η εύρεση ενός κατάλληλου πιθανολογικού μοντέλου που αντανακλά την εμφάνιση πραγματικών περιπτώσεων είναι από μόνη της αρκετά απαιτητική εργασία. Επιπλέον, οι τεχνικές δυσκολίες αντιμετώπισης τέτοιων πιθανοτικών μοντέλων είναι συντριπτικές για τους περισσότερους από τους πιο περίπλοκους αλγόριθμους και μόνο πολύ απλά πιθανοτικά μοντέλα μπορούν να αναλυθούν. Ένα άλλο σημαντικό μειονέκτημα της μεθόδου είναι το γεγονός ότι τα περισσότερα αποτελέσματα είναι σχετικά μόνο εάν το μέγεθος του προβλήματος, δηλαδή ο αριθμός των αντικειμένων, είναι επαρκώς μεγάλο ή εάν εκτελείται αρκετά μεγάλος αριθμός δειγμάτων.

Η τρίτη και πιο κοινή μέθοδος μέτρησης της απόδοσης ενός αλγορίθμου είναι η ανάλυση στη χειρότερη περίπτωση του χρόνου εκτέλεσης του. Αυτό σημαίνει ότι δίνουμε ένα ανώτερο όριο στον αριθμό των βασικών αριθμητικών πράξεων που απαιτούνται για την επίλυση οποιασδήποτε παρουσίας ενός δεδομένου μεγέθους. Αυτό το άνω όριο θα πρέπει να είναι συνάρτηση του μεγέθους του στιγμιότυπου προβλήματος, το οποίο εξαρτάται από τον αριθμό των τιμών εισόδου και επίσης από το μέγεθός τους. Σκεφτείτε ότι μπορεί να χρειαστεί να εκτελέσουμε ένα συγκεκριμένο σύνολο λειτουργιών για κάθε αντικείμενο ή π.χ. για κάθε ακέραια τιμή από ένα έως τη μέγιστη χωρητικότητα . Για τα περισσότερα χρονικά όρια εκτέλεσης, θα αρκεί να λάβουμε υπόψη ως παραμέτρους για το μέγεθος ενός προβλήματος ο αριθμός των στοιχείων , η χωρητικότητα και η μεγαλύτερη τιμή κέρδους ή βάρους ή .

Η μέτρηση του ακριβούς αριθμού των απαραίτητων αριθμητικών πράξεων ενός αλγορίθμου είναι σχεδόν αδύνατη και εξαρτάται επίσης από λεπτομέρειες εφαρμογής. Για να συγκρίνουμε διαφορετικούς αλγόριθμους, ενδιαφερόμαστε κυρίως για ένα ασυμπτωτικό άνω όριο για το χρόνο εκτέλεσης για να απεικονίσουμε την τάξη μεγέθους της αύξησης του χρόνου εκτέλεσης, δηλαδή θέλουμε να γνωρίζουμε την αύξηση του χρόνου εκτέλεσης εάν για παράδειγμα ο αριθμός των στοιχείων είναι διπλασιάστηκε. Οι σταθεροί παράγοντες ή οι σταθεροί πρόσθετοι όροι αγνοούνται, γεγονός που επιτρέπει μεγαλύτερη γενικότητα, καθώς τα «κόλπα» υλοποίησης και τα ειδικά χαρακτηριστικά του μηχανήματος δεν παίζουν κανένα ρόλο σε αυτήν την αναπαράσταση. Μια ανάλογη διαδικασία ισχύει για την απαιτούμενη ποσότητα μνήμης υπολογιστή.

Για το πρόβλημα του σακιδίου και πολλές από τις γενικεύσεις του είναι γνωστοί ψευδοπολυωνυμικοί αλγόριθμοι. Σαφώς, θα προτιμούσαμε να έχουμε ακόμη και πολυωνυμικούς αλγόριθμους χρόνου για αυτά τα προβλήματα. Ωστόσο, υπάρχουν πολύ ισχυρές θεωρητικές ενδείξεις ότι για το πρόβλημα του σακιδίου και συνεπώς και για τις γενικεύσεις του δεν υπάρχει πολυωνυμικός αλγόριθμος χρόνου για τον υπολογισμό της βέλτιστης επίλυσής του. Στην πραγματικότητα, όλα αυτά τα προβλήματα ανήκουν σε μια κατηγορία λεγόμενων σκληρών προβλημάτων βελτιστοποίησης (hard optimization problems). Πιστεύεται ευρέως ότι δεν υπάρχει κανένας πολυωνυμικός αλγόριθμος χρόνου για την επίλυση ενός δύσκολου προβλήματος με βέλτιστο τρόπο επειδή όλα τα δύσκολα προβλήματα είναι ισοδύναμα με την έννοια ότι εάν οποιοδήποτε δύσκολο πρόβλημα θα να βρεθεί ότι είναι επιλύσιμο σε πολυωνυμικό χρόνο, τότε αυτή η ιδιότητα θα ισχύει για όλα τα δύσκολα προβλήματα .

## Παραδοσιακοί Αλγόριθμοι επίλυσης του Προβλήματος του Σακιδίου

Με την πάροδο του χρόνου, έχουν αναπτυχθεί πολυάριθμοι αλγόριθμοι για την επίλυση των διαφορετικών εκδόσεων του προβλήματος του σακιδίου.

**Προσεγγίσεις Δυναμικού Προγραμματισμού (Dynamic Programming)**

Ο Δυναμικός Προγραμματισμός (DP) είναι μία από τις πιο αποτελεσματικές μεθόδους για την επίλυση του προβλήματος του σακιδίου 0-1 (**Knapsack 0-1**), παρέχοντας ακριβείς λύσεις μέσω της διάσπασης του προβλήματος σε απλούστερα υποπροβλήματα[4].

**Άπληστοι Αλγόριθμοι (Greedy Algorithms)**

Οι άπληστοι αλγόριθμοι χρησιμοποιούνται συνήθως για το πρόβλημα του κλασματικού σακιδίου(Fractional KP), όπου τα αντικείμενα μπορούν να διαιρεθούν. Αυτοί οι αλγόριθμοι κάνουν την τοπικά βέλτιστη επιλογή σε κάθε βήμα, με την ελπίδα να βρουν το global βέλτιστο[5].

**Μέθοδοι Branch and Bound**

Η μέθοδος Branch and Bound είναι μια συστηματική μέθοδος για την επίλυση προβλημάτων προγραμματισμού ακεραίων, συμπεριλαμβανομένου του προβλήματος του σακιδίου 0-1. Εξερευνά τον χώρο των λύσεων μέσω της διακλάδωσης και χρησιμοποιεί όρια για να απορρίπτει υποβέλτιστες διακλαδώσεις[4].

**Προσεγγιστικοί Αλγόριθμοι**

Οι αλγόριθμοι προσέγγισης παρέχουν σχεδόν βέλτιστες λύσεις με εγγυημένα όρια ακρίβειας, κάτι που είναι ιδιαίτερα χρήσιμο για προβλήματα μεγάλου μεγέθους του σακιδίου, όπου οι ακριβείς λύσεις είναι υπολογιστικά δαπανηρές.[6]

**Πλήρως Πολυωνυμικά Σχήματα Προσέγγισης (FPTAS)**

Τα FPTAS είναι αλγόριθμοι που παρέχουν λύσεις που πλησιάζουν αυθαίρετα κοντά στη βέλτιστη με πολυωνυμικό χρόνο εκτέλεσης, καθιστώντας τους εξαιρετικά αποδοτικούς για πρακτικές εφαρμογές[6].

**Ευρετικοί και Μεταευρετικοί Αλγόριθμοι**

Οι ευρετικές μέθοδοι, όπως οι Γενετικοί Αλγόριθμοι (Genetic Algorithms), η Προσομοιωμένη Ανόπτηση(Simulated Annealing) και η Αναζήτηση Tabu (Tabu Search), χρησιμοποιούνται για να βρουν ικανοποιητικές λύσεις σε λογικά χρονικά πλαίσια, ιδιαίτερα για μεγάλες και πολύπλοκες περιπτώσεις του προβλήματος του σακιδίου[7], [8].

**Προσεγγίσεις Ακέραιου Γραμμικού Προγραμματισμού (Integer Linear Programming)**

Οι διατυπώσεις του προβλήματος του σακιδίου με ILP επιτρέπουν τη χρήση ισχυρών βελτιστοποιητικών λύσεων για την εύρεση ακριβών λύσεων, εξερευνώντας την εφικτή περιοχή που ορίζεται από τους περιορισμούς[9].

**Ακριβείς Αλγόριθμοι με Χρήση Branch-and-Cut**

Αυτοί οι αλγόριθμοι βελτιώνουν το πλαίσιο του branch-and-bound προσθέτοντας επίπεδα κοπής για να ενισχύσουν τη γραμμική χαλάρωση(linear relaxation), μειώνοντας έτσι τον χώρο αναζήτησης και βελτιώνοντας την αποδοτικότητα[10].

**Αλγόριμοι Pisinger**

Ο David Pisinger έχει αναπτύξει αρκετούς εξαιρετικά αποδοτικούς αλγόριθμους για το πρόβλημα του σακιδίου, συμπεριλαμβανομένων μεθόδων που βασίζονται στον δυναμικό προγραμματισμό και συνδυαστικών αλγόριθμων που ξεπερνούν τους γενικού σκοπού λύτες[11].

## Συμπεράσματα Κεφαλαίου

Ανακεφαλαιώνοντας, το πρόβλημα Knapsack αποτελεί ένα χαρακτηριστικό παράδειγμα μιας συνδυαστικής πρόκλησης βελτιστοποίησης με μεγάλη πολυπλοκότητα. Η δύσκολη φύση του, που χαρακτηρίζεται από την αύξηση του χώρου λύσεων με το μέγεθος του προβλήματος, καθιστά αναγκαία τη διερεύνηση αποτελεσματικών αλγορίθμων και ευρετικών προσεγγίσεων για την αντιμετώπιση πραγματικών περιπτώσεων.

Η βιβλιογραφία σχετικά με την πολυπλοκότητα του προβλήματος Knapsack εστιάζει στην ανάγκη για καινοτόμες τεχνικές, οι οποίες περιλαμβάνουν τον δυναμικό προγραμματισμό και τους άπληστους αλγορίθμους για ακριβείς λύσεις, τις μεταευρετικές μεθόδους και τους αλγορίθμους προσέγγισης για περιπτώσεις μεγάλης κλίμακας. Οι ερευνητές αναζητούν την επίτευξη ισορροπίας μεταξύ της βέλτιστης λύσης και της υπολογιστικής αποτελεσματικότητας, αναγνωρίζοντας τους συμβιβασμούς στην αντιμετώπιση προβλημάτων αυξανόμενης πολυπλοκότητας.

Οι πρακτικές εφαρμογές του προβλήματος Knapsack καλύπτουν διάφορους τομείς, συμπεριλαμβανομένων των οικονομικών, της εφοδιαστικής αλυσίδας, της διαχείρισης πόρων και του χρονοπρογραμματισμού. Η συμβολή του στις διαδικασίες λήψης αποφάσεων σε όλους τους κλάδους αναδεικνύει τη σημασία της εξεύρεσης αποτελεσματικών λύσεων για τη βελτιστοποίηση της κατανομής των πόρων, τη μεγιστοποίηση της χρησιμότητας και τον εξ ορθολογισμό των λειτουργιών. Καθώς η τεχνολογία εξελίσσεται, το πρόβλημα Knapsack συνεχίζει να βρίσκει νέες εφαρμογές σε αναδυόμενους τομείς, υπογραμμίζοντας περαιτέρω τη σημασία του για την αντιμετώπιση των σύγχρονων προκλήσεων. Οι ερευνητές συμβάλλουν όχι μόνο στην ανάπτυξη εξελιγμένων αλγορίθμων αλλά και στην προώθηση συστημάτων υποστήριξης αποφάσεων ικανών να χειρίζονται περίπλοκα σενάρια του πραγματικού κόσμου. Οι μελλοντικές προσπάθειες μπορούν να επικεντρωθούν στην βελτίωση των υφιστάμενων μεθοδολογιών, στη διερεύνηση υβριδικών προσεγγίσεων και στην προσαρμογή των λύσεων στα εξελισσόμενα τεχνολογικά τοπία.

# Η χρήση των Νευρωνικών Δικτύων (Νeural Νetworks) στην επίλυση προβλημάτων Συνδυαστικής Βελτιστοποίησης(CO)

## Εισαγωγή

Η αναπαράσταση πολλαπλών στρωμάτων ονομάζεται νευρωνικό δίκτυο, που αναφέρεται στη νευροβιολογία. Μπορούμε να αναπαραστήσουμε το νευρωνικό δίκτυο ως μια ακολουθία επιπέδων που μπορούν να υπολογίσουν αρκετούς μετασχηματισμούς για να επιστρέψετε μια έξοδο. Αυτό το πλαίσιο είναι δομημένο με είσοδο, κρυφά στρώματα και στρώματα εξόδου, και όλα αυτά αποτελούνται από κόμβους, που ονομάζονται επίσης νευρώνες ή μονάδες. Ένα νευρωνικό δίκτυο είναι ένα διασυνδεδεμένο σύνολο απλών στοιχείων επεξεργασίας, μονάδων ή κόμβων, των οποίων η λειτουργικότητα βασίζεται στον ζωικό νευρώνα [62]. Η ικανότητα επεξεργασίας του δικτύου είναι αποθηκευμένη στην ισχύ των συνδέσεων μεταξύ των μονάδων, ή βάρη, που προκύπτουν από μια διαδικασία προσαρμογής ή μάθησης από ένα σύνολο προτύπων εκπαίδευσης. Ο ανθρώπινος εγκέφαλος αποτελείται από περίπου 1011 (100 δισεκατομμύρια) νευρικά κύτταρα ή νευρώνες [63]. Οι νευρώνες επικοινωνούν μέσω ηλεκτρικών σημάτων που είναι βραχύβια ερεθίσματα ή «αιχμές» στην τάση του κυτταρικού τοιχώματος ή της μεμβράνης. Ο ενδονευρωνικές συνδέσεις διαμεσολαβούνται από ηλεκτροχημικές συνδέσεις που ονομάζονται συνάψεις, οι οποίες βρίσκονται σε διακλαδώσεις του κυττάρου που αναφέρονται σε που ονομάζονται δενδρίτες. Κάθε νευρώνας δέχεται συνήθως πολλές χιλιάδες συνδέσεις από άλλους νευρώνες και επομένως συνεχώς λαμβάνει πλήθος εισερχόμενων σημάτων, τα οποία τελικά φθάνουν στο κυτταρικό σώμα [64].

Εδώ, ενσωματώνονται ή αθροίζονται με κάποιο τρόπο και εάν το προκύπτον σήμα υπερβαίνει κάποιο κατώφλι, τότε ο νευρώνας θα «πυροδοτήσει» ή θα δημιουργήσει μια τάση σε απόκριση. Αυτό στη συνέχεια μεταδίδεται σε άλλους νευρώνες μέσω μιας διακλαδισμένης ίνας, γνωστής ως άξονας. Κατά τον προσδιορισμό του κατά πόσον πρέπει να παραχθεί ή όχι μια τάση, ορισμένα εισερχόμενα σήματα παράγουν ανασταλτικό αποτέλεσμα και τείνουν να αποτρέψουν την πυροδότηση, ενώ άλλα είναι διεγερτικά και προάγουν την παραγωγή ερεθίσματος. Η ξεχωριστή ικανότητα επεξεργασίας κάθε νευρώνα υποτίθεται ότι έγκειται στον τύπο -διεγερτική ή ανασταλτική- και στην ισχύ των συναπτικών συνδέσεών του με άλλους νευρώνες [63]. Ένα νευρωνικό δίκτυο μπορεί να περιγραφεί ως κατευθυνόμενος ακυκλικός γράφος όταν οι κόμβοι αντιστοιχούν στους νευρώνες και οι ακμές αντιστοιχούν στις συνδέσεις μεταξύ τους. Κάθε κόμβος λαμβάνει είσοδο από κάποιους άλλους κόμβους (ή από μια εξωτερική πηγή) και υπολογίζει μια έξοδο. Κάθε είσοδος έχει ένα σχετικό βάρος (w), το οποίο αποδίδεται με βάση τη σχετική της σημασία της σε σχέση με άλλες εισόδους. Στη συνέχεια, ο κόμβος εφαρμόζει μια συνάρτηση ενεργοποίησης f στο σταθμισμένο άθροισμα των εισόδων του. Επιπλέον, υπάρχει μια άλλη είσοδος με βάρος b που ονομάζεται προκατάληψη.

## Η αρχιτεκτονική ενός Νευρωνικού Δικτύου (NN) αποτελείται από τρία κύρια συστατικά:

**Επίπεδο Εισόδου(input layer)**: Αυτό το επίπεδο λαμβάνει τα ακατέργαστα δεδομένα και τα μεταφέρει στο δίκτυο. Ο αριθμός των νευρώνων αντιστοιχεί στον αριθμό των χαρακτηριστικών(features) στο σύνολο δεδομένων. Για παράδειγμα, στην αναγνώριση εικόνας, κάθε pixel μπορεί να είναι ένα χαρακτηριστικό, με αποτέλεσμα εκατοντάδες ή χιλιάδες νευρώνες.[63]

**Κρυφά Επίπεδα(hidden layers)**: Αυτά τα ενδιάμεσα επίπεδα εκτελούν τις περισσότερες υπολογιστικές και μετασχηματιστικές λειτουργίες στα δεδομένα. Κάθε νευρώνας σε ένα κρυφό επίπεδο εφαρμόζει ένα σταθμισμένο άθροισμα(weighted sum) ακολουθούμενο από μια συνάρτηση ενεργοποίησης(activation function), όπως ReLU ή Sigmoid, για να εισάγει μη γραμμικότητα[63]. Τα κρυφά επίπεδα συχνά ρυθμίζονται σε μεγέθη που είναι δυνάμεις του 2 (π.χ., 128, 256, 512 νευρώνες) για μερικούς πρακτικούς λόγους:

* Υπολογιστική Αποδοτικότητα: Οι σύγχρονες αρχιτεκτονικές υπολογιστών, ιδιαίτερα οι GPU, είναι βελτιστοποιημένες για την επεξεργασία δεδομένων σε μεγέθη που είναι δυνάμεις του 2. Αυτή η ευθυγράμμιση με τις δυνατότητες του υλικού μπορεί να οδηγήσει σε ταχύτερη επεξεργασία και πιο αποδοτική χρήση της μνήμης.
* Τυποποίηση και Σύμβαση: Η χρήση δυνάμεων του 2 έχει γίνει κοινή πρακτική στη μηχανική μάθηση, προσφέροντας καλή ισορροπία μεταξύ της χωρητικότητας του μοντέλου και των υπολογιστικών πόρων.

**Επίπεδο Εξόδου**: Αυτό το επίπεδο παράγει την τελική πρόβλεψη ή ταξινόμηση βάσει των επεξεργασμένων δεδομένων από τα κρυφά επίπεδα. Ο αριθμός των νευρώνων σε αυτό το επίπεδο εξαρτάται από τη συγκεκριμένη εργασία. Για δυαδική ταξινόμηση(binary classification), συνήθως υπάρχει ένας νευρώνας με ενεργοποίηση Sigmoid, παράγοντας αποτελέσματα μεταξύ 0 και 1. Για πολυ-κατηγορική ταξινόμηση(multi-class classification), το επίπεδο εξόδου μπορεί να έχει όσους νευρώνες υπάρχουν κατηγορίες, χρησιμοποιώντας μια συνάρτηση Softmax.[63]

Αυτά τα συστατικά συνεργάζονται για να επιτρέψουν στο νευρωνικό δίκτυο να μαθαίνει και να γενικεύει από δεδομένα, βελτιώνοντας την απόδοσή του με την πάροδο του χρόνου μέσω μιας διαδικασίας γνωστής ως "backpropagation," η οποία προσαρμόζει τα βάρη των συνδέσεων μεταξύ των νευρώνων βάσει του σφάλματος του αποτελέσματος.

## Γιατί να χρησιμοποιήσουμε NN εφόσον έχουμε ήδη τη λύση από τους παραδοσιακούς αλγορίθμους;

**Μάθηση για Μελλοντική Γενίκευση:**

* **Γενίκευση Πέρα από τα Εκπαιδευτικά Δεδομένα:** Ο κύριος στόχος της εκπαίδευσης ενός νευρωνικού δικτύου σε περιπτώσεις συνδυαστικής βελτιστοποίησης δεν είναι μόνο να λύσει τα συγκεκριμένα προβλήματα για τα οποία ήδη υπάρχουν λύσεις, αλλά να επιτρέψει στο δίκτυο να γενικεύσει σε νέες, άγνωστες περιπτώσεις. Αφού εκπαιδευτεί, το νευρωνικό δίκτυο μπορεί να παρέχει γρήγορες, προσεγγιστικές λύσεις για νέα προβλήματα χωρίς να χρειάζεται να χρησιμοποιηθεί ο παραδοσιακός αλγόριθμος κάθε φορά, κάτι που μπορεί να είναι υπολογιστικά δαπανηρό.

**Διαχείριση Δυναμικών ή Μεγάλου Κλίμακας Προβλημάτων:**

* **Αποδοτικότητα σε Πραγματικό Χρόνο:** Σε σενάρια όπου πρέπει να λυθούν νέες περιπτώσεις του προβλήματος γρήγορα ή σε πραγματικό χρόνο (π.χ. δυναμική τιμολόγηση), η χρήση ενός παραδοσιακού αλγόριθμου από την αρχή για κάθε νέα περίπτωση θα ήταν πολύ αργή. Το νευρωνικό δίκτυο, μόλις εκπαιδευτεί, μπορεί να παράγει λύσεις πολύ πιο γρήγορα, καθιστώντας το κατάλληλο για αυτές τις εφαρμογές.
* **Scalability:** Για προβλήματα πολύ μεγάλης κλίμακας ή όπου οι παράμετροι του προβλήματος αλλάζουν συχνά, η περιοδική επανεκπαίδευση ενός νευρωνικού δικτύου μπορεί να είναι πιο αποδοτική από την επανειλημμένη επίλυση με έναν παραδοσιακό αλγόριθμο.

**Προσεγγιστικές Λύσεις σε Πρακτικές Εφαρμογές:**

* **Σχεδόν Βέλτιστες Λύσεις:** Σε πολλές εφαρμογές του πραγματικού κόσμου, μια σχεδόν βέλτιστη λύση είναι συχνά αρκετή, και η μικρή απώλεια βελτιστοποίησης αντισταθμίζεται από τα κέρδη σε ταχύτητα και ευελιξία που παρέχει ένα νευρωνικό δίκτυο. Το νευρωνικό δίκτυο μπορεί να ρυθμιστεί ώστε να παρέχει καλές προσεγγίσεις γρήγορα, κάτι που είναι συχνά πιο πρακτικό από το να εξασφαλίζεται αυστηρή βέλτιστη λύση.

## Ανασκόπηση της βιβλιογραφίας για τη χρήση των Νευρωνικών Δικτύων (Νeural Νetworks) στην επίλυση προβλημάτων Συνδυαστικής Βελτιστοποίησης (CO)

Η βαθιά μάθηση αντιστοιχίζει τις εισόδους στους στόχους με την παρατήρηση πολλών παραδειγμάτων εισόδου και στόχων μέσω μιας βαθιάς ακολουθίας μετασχηματισμών δεδομένων που μαθαίνονται από τα παραδείγματα. Ο μετασχηματισμός που υλοποιείται από ένα επίπεδο παραμετροποιείται από τα βάρη του, μερικές φορές που ονομάζονται επίσης παράμετροι ενός στρώματος. Σε αυτό το πλαίσιο, μάθηση σημαίνει εύρεση ενός συνόλου τιμών για τα βάρη όλων των στρωμάτων του δικτύου. Ένα βαθύ νευρωνικό δίκτυο (ΝΝ) μπορεί να περιέχει δεκάδες εκατομμύρια παραμέτρους. Η εύρεση της σωστής τιμής για καθεμία από αυτές είναι πολύ δύσκολη, διότι η τροποποίηση της τιμής μιας παραμέτρου θα επηρεάσει τη συμπεριφορά όλων των άλλων. Χρειάζεται να μπορούμε να μετρήσουμε πόση διαφορά έχει η έξοδος του δικτύου από την αναμενόμενη. Η ενέργεια αυτή πραγματοποιείται από μια συνάρτηση απώλειας του δικτύου(loss function), που ονομάζεται επίσης objective function. Η συνάρτηση αυτή λαμβάνει την πρόβλεψη και υπολογίζει μια βαθμολογία απόστασης. Αυτή η βαθμολογία χρησιμοποιείται ως σήμα ανατροφοδότησης για την προσαρμογή της τιμής των βαρών προς μια κατεύθυνση που θα μειώσει τη βαθμολογία απώλειας για το τρέχον παράδειγμα. Αυτή η προσαρμογή γίνεται από τον βελτιστοποιητή(optimizer), ο οποίος υλοποιεί τον λεγόμενο αλγόριθμο οπισθοδιάδοσης(backpropagation). Ο αλγόριθμος αυτός ξεκινά με την τελική τιμή απώλειας και εργάζεται προς τα πίσω από τα κορυφαία στρώματα προς τα κατώτερα στρώματα για να υπολογίσει την παράγωγο κάθε παραμέτρου στην τελική τιμή απώλειας.

Η βαθιά μάθηση (deep learning) είναι μια ομάδα μεθόδων μάθησης που προσπαθούν να μοντελοποιήσουν δεδομένα με πολύπλοκες αρχιτεκτονικές συνδυάζοντας σχεδιασμούς διαφορετικούς και μη γραμμικούς [65]. Το θεμέλιο της βαθιάς μάθησης είναι τα νευρωνικά δίκτυα, τα οποία αφού συνδυαστούν, σχηματίζουν βαθιά νευρωνικά δίκτυα. Αυτές οι τεχνολογίες επέτρεψαν σημαντική ανάπτυξη και πρόοδο στην αυτόματη επεξεργασία ήχου και εικόνας, συμπεριλαμβανομένης της αναγνώρισης χαρακτηριστικών προσώπου, της αναγνώρισης ομιλίας, της όρασης υπολογιστή, της αυτόματης επεξεργασία φυσικής γλώσσας, της ταξινόμησης κειμένων. Υπάρχουν πολλές πρακτικές εφαρμογές. Χαρακτηριστικό παράδειγμα είναι το πρόγραμμα AlphaGo, όπου χρησιμοποιώντας τη μέθοδο βαθιάς μάθησης, έμαθε να παίζει το παιχνίδι και μάλιστα κατάφερε το 2016, να κερδίσει τον παγκόσμιο πρωταθλητή.

Η βαθιά μάθηση και τα νευρωνικά δίκτυα (ΝΝ) βρίσκονται στην καρδιά ορισμένων από τις μεγαλύτερες προόδους στη σύγχρονη επιστήμη των υπολογιστών. Επιτρέπουν τεράστιες ανακαλύψεις σε εφαρμογές όπως η όραση υπολογιστών, η μετάφραση, η αναγνώριση ομιλίας και η αυτόνομη οδήγηση, για να αναφέρουμε [15]. Ενώ πολυάριθμες υπολογιστικές μελέτες παρουσιάζουν εντυπωσιακές εμπειρικές αποδείξεις της υπολογιστικής ισχύος των νευρωνικών δικτύων, απέχουμε ακόμη πολύ από μια πιο αυστηρή θεωρητική εξήγηση αυτών των παρατηρήσεων. Εκτός από τις δημοφιλείς εφαρμογές που αναφέρονται παραπάνω, έχει αποδειχθεί ότι τα νευρωνικά δίκτυα έχουν μεγάλες δυνατότητες για πρακτικές την επίλυση προβλημάτων συνδυαστικής βελτιστοποίησης (CO) ή την εμπειρική βελτίωση των κλασικών μεθόδων επίλυσης [65]. Για παράδειγμα, οι Yang et al. (2018) και Xu et al. (2020) χρησιμοποιούν νευρωνικά δίκτυα προκειμένου να βελτιώσουν εμπειρικά τον δυναμικό προγραμματισμό, μια πολύ κλασική μέθοδο CO [66-67]. Ενώ οι μέθοδοι που χρησιμοποιούνται σε αυτές τις εργασίες παρέχουν πράγματι γρήγορες και εμπειρικά σχεδόν βέλτιστες λύσεις, η χρήση των ΝΝ τους καθιστά πρακτικά αδύνατη την παροχή βέλτιστης θεωρητικής ή εγγυήσεις προσέγγισης στη χειρότερη περίπτωση. Επιπλέον, υποστηρίζουμε ότι η προσέγγισή μπορεί να γενικευτεί σε πολλά άλλα προβλήματα CO, π.χ., διάφορα προβλήματα συντομότερης διαδρομής, το πρόβλημα της Μακρύτερης κοινής υποακολουθίας, και το Πρόβλημα του πλανόδιου ταξιδιώτη.

Ενώ έχει γίνει λιγότερη έρευνα στη διασταύρωση του CO και NN στη δεκαετία του 2000, σύγχρονες εξελίξεις στην περιοχή της βαθιάς μάθησης έχουν τονώσει το ενδιαφέρον προς αυτή την κατεύθυνση. Οι Bengio, Lodi και Prouvost (2018), εξετάζουν αυτές τις εξελίξεις από πρακτική άποψη [65]. Οι κοινές εφαρμογές περιλαμβάνουν την επιτάχυνση λύσεων για γραμμικά προγράμματα μικτού ακέραιου αριθμού, για παράδειγμα, με αυτόματη εκμάθηση. Η μηχανική μάθηση έχει επίσης εφαρμοστεί σε πτυχές μοντελοποίησης του CO, όπως αξιολογήθηκε από τους Lombardi και Milano (2018) [68], και σε αρκετά συγκεκριμένα προβλήματα CO, όπου το TSP είναι συχνά ένα από αυτά. Οι διαφορετικοί μέθοδοι που χρησιμοποιούνται από τους συγγραφείς περιλαμβάνουν την ανατροφοδότηση και τα αναδρομικά νευρωνικά δίκτυα, ενισχυτική μάθηση, μηχανισμοί προσοχής, δίκτυα δεικτών, ενσωματώσεις γραφημάτων και γραφήματα νευρωνικών δικτύων [22], [23], [69], [70], [71], [72].

## Συμπεράσματα Κεφαλαίου

Συνολικά, η βιβλιογραφική ανασκόπηση αποκαλύπτει ένα ευρύ φάσμα προσεγγίσεων και μεθοδολογιών που χρησιμοποιούνται από ερευνητές για την αντιμετώπιση προβλημάτων συνδυαστικής βελτιστοποίησης και αντανακλά ένα εξελισσόμενο πεδίο. Καθώς οι ερευνητές συνεχίζουν να βελτιώνουν τις τεχνικές, η ενσωμάτωση των νευρωνικών δικτύων σε διαδικασίες συνδυαστικής βελτιστοποίησης αναμένεται να συμβάλλει στην αντιμετώπιση σύνθετων προβλημάτων λήψης αποφάσεων σε διάφορους πρακτικούς τομείς. Έχουν διερευνηθεί διάφορες στρατηγικές κωδικοποίησης, αρχιτεκτονικές δικτύων, συναρτήσεις απωλειών και στρατηγικές εκπαίδευσης, γεγονός που αντικατοπτρίζει την προσαρμοστικότητα των νευρωνικών δικτύων στις ιδιαιτερότητες τέτοιων προβλημάτων. Μια σημαντική τάση είναι η συνεχής διερεύνηση υβριδικών μοντέλων, που συνδυάζουν τα πλεονεκτήματα των νευρωνικών δικτύων με παραδοσιακούς αλγορίθμους βελτιστοποίησης ή ευρετικές μεθόδους. Η ενσωμάτωση αυτή αποσκοπεί στην αξιοποίηση των συμπληρωματικών πλεονεκτημάτων και των δύο προτύπων, συμβάλλοντας στην ενίσχυση της ποιότητας των λύσεων, της ταχύτητας σύγκλισης και της επεκτασιμότητας.

Η βιβλιογραφία αναδεικνύει τη σημασία της συγκριτικής αξιολόγησης και της αξιολόγησης των επιδόσεων, συγκρίνοντας τα μοντέλα που βασίζονται σε νευρωνικά δίκτυα με τις παραδοσιακές μεθόδους σε μετρικές όπως η ποιότητα της λύσης και η ταχύτητα σύγκλισης. Εφαρμογές σε διάφορους τομείς, όπως η εφοδιαστική, η κατανομή πόρων και τα οικονομικά, αναδεικνύουν την ευελιξία των ΝΝ στην αντιμετώπιση πραγματικών περιπτώσεων συνδυαστικής βελτιστοποίησης. Αξίζει να σημειωθεί οτι συνεχίζουν να υπάρχουν προκλήσεις, όπως η ανάγκη για βελτιωμένη γενίκευση για την αντιμετώπιση μεγαλύτερων περιπτώσεων προβλημάτων και η διερεύνηση νέων αρχιτεκτονικών νευρωνικών δικτύων. Οι μελλοντικές ερευνητικές κατευθύνσεις μπορεί να περιλαμβάνουν την αντιμετώπιση αυτών των προκλήσεων, την βελτίωση των υφιστάμενων μεθοδολογιών και τη διερεύνηση εφαρμογών σε αναδυόμενους τομείς.

# Βαθιά Μάθηση (Deep Learning) για την επίλυση του Knapsack Problem

## Εισαγωγή

Το πρόβλημα του σακιδίου, ένα θεμελιώδες ζήτημα συνδυαστικής βελτιστοποίησης, έχει ωθήσει τους ερευνητές να διερευνήσουν διάφορες αρχιτεκτονικές νευρωνικών δικτύων για την εύρεση αποτελεσματικών λύσεων. Αυτές οι μέθοδοι περιλαμβάνουν προσεγγίσεις τα Αναδρομικά Νευρωνικά Δίκτυα (RNNs), με εποπτεία(supervised) και χωρίς εποπτεία(unsupervised), τα Graph Neural Networks(GNNs) , τα Spiking Neural Networks (SNNs) και τα Generative Adversarial Networks (GANs). Επιπλέον, η ενισχυτική μάθηση (reinforced learning ) διαδραματίζει σημαντικό ρόλο σε δυναμικά περιβάλλοντα. Αυτή η ενότητα θα εξετάσει αυτές τις τεχνικές και τις συνεισφορές τους στη λύση του προβλήματος του σακιδίου.

## Αλγόριθμοι Deep Learning που έχουν προταθεί για την επίλυση του KP

1. **Αναδρομικά Νευρωνικά Δίκτυα (Recurrent Neural Networks, RNNs)**

Το [26], παρουσιάζει μια νέα προσέγγιση για την επίλυση του προβλήματος του σακιδίου, χρησιμοποιώντας έναν συνδυασμό νευρωνικών δικτύων, με αρχιτεκτονική **sequence-to-sequence(seq2seq)** και έμφαση στη χρήση **Gated Recurrent Units (GRUs)**, ένα τύπο αρχιτεκτονικής Αναδρομικών Νευρωνικών Δικτύων (RNN). Αυτή η προσέγγιση ενσωματώνει στοιχεία της βαθιάς μάθησης, της αναπαραστατικής μάθησης και της συνδυαστικής βελτιστοποίησης, για να αντιμετωπίσει αποτελεσματικά τις πολυπλοκότητες του προβλήματος του σακιδίου, ιδιαίτερα της εκδοχής του προβλήματος 0-1.

**Αρχιτεκτονική**

**Είσοδοι(Inputs)**

Κάθε αντικείμενο αναπαρίσταται ως διάνυσμα που περιλαμβάνει την αξία, το βάρος και τη χωρητικότητα του σακιδίου. Η αναπαράσταση αυτή επιτρέπει στο μοντέλο να κατανοήσει τη σημαντικότητα του κάθε αντικειμένου και τη σχέση του με τον περιορισμό χωρητικότητας του σακιδίου. Το διάνυσμα αποτελεί τη βάση για την εισαγωγή των δεδομένων στο δίκτυο.

**Memory Constructor**

Το **Memory Constructor** δημιουργεί διανυσματικές αναπαραστάσεις για κάθε αντικείμενο, περιλαμβάνοντας την αξία, το βάρος και τη χωρητικότητα του σακιδίου. Το μοντέλο χρησιμοποιεί είτε **Dense layers**, είτε **1D-CNN layers**, είτε **GRU Layers**. Κάθε μοντέλο περιέχει ένα επίπεδο με 32 hidden units και activation function ReLU(Rectified Linear Unit), η οποία εισάγει μη-γραμμικότητα, για να μπορέσει το δίκτυο να μάθει πολύπλοκες σχέσεις μεταξύ των χαρακτηριστικών των αντικειμένων εισόδου.

**Encoder**

Ο κωδικοποιητής επεξεργάζεται τη σειρά των αντικειμένων και δημιουργεί μια σταθερή αναπαράσταση που περιλαμβάνει τις σχέσεις μεταξύ τους. Αυτή η διαδικασία γίνεται μέσω δύο επιπέδων GRU με 32 κρυφές μονάδες το καθένα. Το διάνυσμα εξόδου ​ συμπυκνώνει όλες τις πληροφορίες για τα αντικείμενα και τις διαβιβάζει στον αποκωδικοποιητή για τις τελικές αποφάσεις.

**Decoder**

Ο αποκωδικοποιητής λαμβάνει αποφάσεις για κάθε αντικείμενο με βάση τις αναπαραστάσεις που παράγει ο κωδικοποιητής και το Memory Constructor. Χρησιμοποιεί τρία επίπεδα GRU με 32 κρυφές μονάδες το καθένα και έναν μηχανισμό προσοχής(**attention mechanism**) που επιτρέπει στο μοντέλο να επικεντρώνεται σε σημαντικά μέρη της μνήμης κατά τη λήψη αποφάσεων. Το αποτέλεσμα είναι μια πιθανότητα {0, 1} που υποδεικνύει αν το αντικείμενο πρέπει να συμπεριληφθεί στο σακίδιο.

**Knapsack Handler**

Το **Knapsack Handler** παρακολουθεί την κατάσταση του σακιδίου, ενημερώνοντας το μοντέλο για τη χωρητικότητα και την αξία που απομένουν μετά από κάθε επιλογή αντικειμένου και μεταδίδει τις πληροφορίες στον decoder. Αυτό το στοιχείο αποτελείται από δύο GRU επίπεδα με 32 κρυφές μονάδες και διασφαλίζει ότι το τελικό αποτέλεσμα θα είναι εφικτό, τηρώντας τους περιορισμούς του προβλήματος.

**Memory Construction**

Η μνήμη που παράγεται από το CNN αποτελείται από διανύσματα που αναπαριστάνουν τις σχέσεις μεταξύ της αξίας και του βάρους του κάθε αντικειμένου καθώς και της χωρητικότητας. Το μέγεθος της μνήμης προσαρμόζεται δυναμικά ανάλογα με τον αριθμό των αντικειμένων, διασφαλίζοντας την επεκτασιμότητα για διαφορετικά μεγέθη του προβλήματος.

**Διαδικασία Εκπαίδευσης**

1. Εποπτευόμενη Μάθηση (Supervised Learnig)

Η εκπαίδευση πραγματοποιείται με εποπτευόμενο τρόπο(**supervised**), χρησιμοποιώντας τις βέλτιστες λύσεις που παράγονται από τον **CBC solver**. Το μοντέλο μαθαίνει να μιμείται αυτές τις λύσεις για να βελτιστοποιήσει την απόδοσή του.

2. Curriculum Learning

Το μοντέλο εκπαιδεύεται σταδιακά με προβλήματα αυξανόμενης δυσκολίας, μέσω μιας στρατηγικής **naive curriculum learning**. Καθώς η πολυπλοκότητα των προβλημάτων αυξάνεται, το μοντέλο βελτιώνει την ικανότητά του να λύνει πιο περίπλοκες περιπτώσεις.

3. Συνάρτηση Απώλειας (Loss Function)

Η συνάρτηση απώλειας που χρησιμοποιείται είναι η **binary cross-entropy**, η οποία αξιολογεί την ακρίβεια των δυαδικών αποφάσεων του μοντέλου σχετικά με το αν ένα αντικείμενο θα συμπεριληφθεί στο σακίδιο ή όχι.

4. Optimizer and Hyperparameters

Ο **Adam optimizer** χρησιμοποιείται με ρυθμό εκμάθησης (learning rate) 0.004. Η εκπαίδευση πραγματοποιείται με batch μεγέθους 100 και διαρκεί για 100 εποχές. Εφαρμόζεται επίσης gradient clipping για την αποφυγή gradient explosion.

**Αξιολόγηση**

Το μοντέλο αξιολογήθηκε χρησιμοποιώντας 3 κριτήρια: την αναλογία προσέγγισης (**approximation ratio**), το ποσοστό μη εφικτών λύσεων (**infeasibility rate**) και το ποσοστό βέλτιστων λύσεων (**optimal instances** **rate**)

1. Αναλογία Προσέγγισης (Approximation Ratio)

Η **αναλογία προσέγγισης** μετρά πόσο κοντά είναι η λύση του μοντέλου στη βέλτιστη λύση. Όσο χαμηλότερη είναι η αναλογία, τόσο καλύτερη είναι η απόδοση του μοντέλου. Υπολογίζεται ως το ποσοστό προσέγγισης της εξόδου του μοντέλου στη βέλτιστη λύση:

Όπου Y το κόστος της βέλτιστης λύσης, και Ys η λύση του μοντέλου.

2. Ποσοστό Μη Εφικτών Λύσεων (Infeasibility Rate)

Το **ποσοστό μη εφικτών λύσεων** αντιπροσωπεύει το ποσοστό των περιπτώσεων όπου η λύση του μοντέλου υπερβαίνει τη χωρητικότητα του σακιδίου, καθιστώντας τη λύση μη εφικτή. Αυτό το κριτήριο είναι σημαντικό για την κατανόηση της ικανότητας του μοντέλου να διαχειρίζεται τους περιορισμούς χωρητικότητας του προβλήματος του σακιδίου.

3. Ποσοστό Βέλτιστων Λύσεων (Optimal Instances Rate)

Το **ποσοστό βέλτιστων λύσεων** αναφέρεται στο ποσοστό των περιπτώσεων όπου το μοντέλο παράγει ακριβώς τη βέλτιστη λύση, όπως καθορίζεται από τον ακριβή λύτη. Αυτό το κριτήριο είναι χρήσιμο για την αξιολόγηση της ικανότητας του μοντέλου να λύσει περιπτώσεις του προβλήματος του σακιδίου με απόλυτη ακρίβεια.

**Δημιουργία Δεδομένων**

**Τυχαία Δημιουργία Περιπτώσεων**

Οι περιπτώσεις του προβλήματος του σακιδίου δημιουργήθηκαν τυχαία. Αυτή η διαδικασία περιείχε τον καθορισμό των τιμών και των βαρών των αντικειμένων, μαζί με τη χωρητικότητα του σακιδίου. Στόχος των ερευνητών ήταν η δημιουργία περιπτώσεων που αντικατοπτρίζουν με ακρίβεια τις πολυπλοκότητες των πραγματικών προβλημάτων του σακιδίου. Κάθε περίπτωση λύθηκε στη συνέχεια χρησιμοποιώντας έναν λύτηακριβείας**,** τον **CBC Solver,** μια βιβλιοθήκη branch-and-cut, και τα μεταβλητά αποτελέσματα (εάν θα συμπεριληφθεί ή θα αποκλειστεί ένα αντικείμενο) χρησιμοποιούνται ως ετικέτες-στόχοι κατά τη διαδικασία μάθησης.

**Διαφοροποίηση Χωρητικοτήτων**

Οι χωρητικότητες των σακιδίων τόσο στα σύνολα εκπαίδευσης όσο και στα σύνολα αξιολόγησης ποικίλλουν προσεκτικά για να διασφαλιστεί ότι το μοντέλο μπορεί να χειριστεί ένα ευρύ φάσμα τύπων προβλημάτων. Η χωρητικότητα κάθε περίπτωσης ορίζεται με βάση το συνολικό βάρος όλων των αντικειμένων στην περίπτωση. Αυτή η ποικιλομορφία είναι σημαντική για να αποτυπώσει το μοντέλο τη συμπεριφορά διαφορετικών περιπτώσεων του προβλήματος του σακιδίου και να διασφαλίσει ότι μπορεί να γενικεύσει καλά σε διαφορετικές πολυπλοκότητες προβλημάτων.

**Ορισμός Χωρητικότητας**

Σε κάθε περίπτωση, η χωρητικότητα καθορίστηκε με τον ακόλουθο τύπο:

Όπου χρησιμοποιήθηκε για όλα τα πειράματα. Αυτός ο τύπος διασφαλίζει ότι η χωρητικότητα είναι ένα κλάσμα του συνολικού βάρους των αντικειμένων στην περίπτωση και εφαρμόζεται ομοιόμορφα σε διαφορετικά σύνολα δεδομένων για να διατηρηθεί η συνέπεια.

**Τύποι Συνόλων Δεδομένων**

Το μοντέλο εκπαιδεύτηκε και αξιολογήθηκε σε τρεις τύπους περιπτώσεων του προβλήματος του σακιδίου, κάθε μία με διαφορετικές συσχετίσεις μεταξύ των τιμών και των βαρών των αντικειμένων:

* **Μη Συσχετιζόμενα (Uncorrelated - UC)**: Οι τιμές και τα βάρη των αντικειμένων κατανέμονται ομοιόμορφα σε ένα καθορισμένο εύρος.
* **Ισχυρά Συσχετιζόμενα (Strongly Correlated - SC)**: Τα βάρη κατανέμονται ομοιόμορφα και οι τιμές σχετίζονται άμεσα με τα βάρη, ορίζονται ως .
* **Subset-Sum (Subset Sum - SS)**: Σε αυτόν τον τύπο, οι τιμές και τα βάρη των αντικειμένων είναι ίσα.

Αυτή η ποικιλία τύπων δεδομένων εξασφαλίζει ότι το μοντέλο εκτίθεται σε ένα ευρύ φάσμα προβλημάτων κατά την εκπαίδευση, επιτρέποντάς του να γενικεύει πιο αποτελεσματικά σε άγνωστες περιπτώσεις δοκιμής

**Βασικές Παρατηρήσεις και Απόδοση**

Οι ερευνητές παρατήρησαν ότι ο αλγόριθμος νευρωνικών δικτύων που βασίζεται σε CNN αποδείχθηκε ότι υπερέχει των παραδοσιακών άπληστων αλγορίθμων (Greedy Algorithms) στην επίτευξη σχεδόν βέλτιστων λύσεων, ιδιαίτερα σε περιπτώσεις όπου υπάρχει συσχέτιση μεταξύ των αξιών και των βαρών των αντικειμένων. Η ικανότητα του μοντέλου να γενικεύει σε μεγαλύτερες προβληματικές περιπτώσεις (με έως και 200 αντικείμενα) επίσης δοκιμάστηκε, αποδεικνύοντας την ανθεκτικότητα του αλγορίθμου στην αντιμετώπιση σύνθετων σεναρίων.

Οι ερευνητές τόνισαν τη σημασία της χωρητικότητας για τον καθορισμό της πολυπλοκότητας μιας περίπτωσης, αντί για τον αριθμό των αντικειμένων, όπως πιστευόταν παραδοσιακά. Παρατήρησαν ότι ο λύτης τους απέδωσε καλύτερα όταν εκπαιδεύτηκε σε περιπτώσεις με διαβαθμισμένες χωρητικότητες, σε σύγκριση με τις περιπτώσεις με μεταβλητή ή σταθερή χωρητικότητα.

Αν και ο νευρωνικός λύτης δεν ξεπερνά τους πιο σύγχρονους λύτες ή μεταευρετικούς αλγόριθμους, προσφέρει πολύτιμες γνώσεις σχετικά με το πώς μπορούν να ενσωματωθούν προσεγγίσεις βασισμένες στη μάθηση στο σχεδιασμό ευρετικών λύσεων για το πρόβλημα του σακιδίου.

1. **Νευρωνικά Δίκτυα Γράφων (Graph Neural Networks, GNNs)**

Οι ερευνητές στο [27] εξέτασαν τη χρήση των Graph Neural Networks (GNN) για την επίλυση του Bilevel Knapsack Problem (BLKP). Το BLKP είναι ένα πολύπλοκο ιεραρχικό πρόβλημα βελτιστοποίησης με δύο λήπτες αποφάσεων: έναν ηγέτη(**leader**) και έναν ακόλουθο(**follower**), ο καθένας από τους οποίους βελτιστοποιεί το δικό του objective. Η πρόκληση έγκειται στο γεγονός ότι οι αποφάσεις του ηγέτη επηρεάζουν τις διαθέσιμες επιλογές του ακολούθου και την αντίστοιχη objective function, καθιστώντας το ένα κλασικό πρόβλημα NP-hard.

**Τύπος Προβλήματος Σακιδίου**

Η έρευνα αφορά το Bilevel Knapsack Problem (BLKP), όπου τόσο ο ηγέτης όσο και ο ακόλουθος μοιράζονται το ίδιο σακίδιο, αλλά ο καθένας έχει τα δικά του αντικείμενα που προσπαθεί να μεγιστοποιήσει. Ο ηγέτης στοχεύει να μεγιστοποιήσει το συνολικό κέρδος όλων των αντικειμένων στο σακίδιο, ενώ ο ακόλουθος προσπαθεί να μεγιστοποιήσει το κέρδος των δικών του αντικειμένων. Το BLKP χρησιμοποιεί την αισιόδοξη περίπτωση(**optimistic** **case**), όπου ο ακόλουθος επιλέγει τη λύση που είναι πιο επωφελής για τον ηγέτη.

**Αρχιτεκτονική GNN**

**1. Graph Representation**

Το BLKP αναπαρίσταται ως **tripartite** **graph**, όπου:

**Κόμβοι**: Αναπαριστούν τα αντικείμενα που ανήκουν είτε στον ηγέτη είτε στον ακόλουθο. Κάθε κόμβος έχει χαρακτηριστικά, όπως το κέρδος και το βάρος του αντικειμένου.

* Κόμβοι του ηγέτη: Αναπαριστούν τα αντικείμενα του ηγέτη.
* Κόμβοι του ακολούθου: Αναπαριστούν τα αντικείμενα του ακολούθου.
* Κόμβος περιορισμού: Αναπαριστά τη χωρητικότητα του σακιδίου και συνδέεται τόσο με τους κόμβους του ηγέτη όσο και του ακολούθου.

**Άκρες**: Συνδέουν τους κόμβους του ηγέτη και του ακολούθου με τον κόμβο περιορισμού, υποδεικνύοντας την αλληλεπίδραση με τη χωρητικότητα του σακιδίου.

**2. Principal Neighborhood Aggregation (PNA)**

Η έρευνα χρησιμοποιεί το **Principal Neighborhood Aggregation** (**PNA**), έναν ισχυρό τύπο GNN που επεξεργάζεται τη δομή του γραφήματος. Το PNA μοντέλο χρησιμοποιείται για να συγκεντρώσει πληροφορίες από τους γειτονικούς κόμβους και τις άκρες, επιτρέποντας σε κάθε κόμβο να μάθει για το συνολικό γράφημα. Το PNA χρησιμοποιεί διάφορες συναρτήσεις όπως mean, max και standard deviation για την επεξεργασία των χαρακτηριστικών των κόμβων.

3. **Encoding**

Η διαδικασία encoding εξάγει σημαντικές πληροφορίες από κάθε κόμβο και άκρη. Το μοντέλο PNA εφαρμόζεται στους κόμβους του ηγέτη και του ακολούθου για να κωδικοποιήσει τα χαρακτηριστικά τους. Μετά το encoding, ο κόμβος περιορισμού αφαιρείται, μετατρέποντας το tripartite γράφημα σε bipartite. Το encoding χρησιμοποιεί **Multi-Layer Perceptrons (MLPs)** για την επεξεργασία των χαρακτηριστικών των αντικειμένων.

**4. Message Passing και Decoding**

Κατά τη διάρκεια του message passing, πληροφορίες για τους γειτονικούς κόμβους περνούν μέσα από το GNN με πολλαπλές επαναλήψεις. Κάθε κόμβος του ηγέτη και του ακολούθου ενημερώνει το διανυσματικό χαρακτηριστικό του με πληροφορίες από τους γείτονές του. Αυτή η διαδικασία επαναλαμβάνεται τρεις φορές. Στη decoding φάση, το μοντέλο προβλέπει την πιθανότητα ότι η μεταβλητή απόφασης του ηγέτη θα συμπεριληφθεί στη λύση, χρησιμοποιώντας ένα MLP με sigmoid activation function.

**5. Solution Search**

Αφού προσδιοριστούν οι αποφάσεις του ηγέτη, το πρόβλημα του ακολούθου λύνεται χρησιμοποιώντας έναν παραδοσιακό αλγόριθμο σακιδίου, τον **CPLEX**. Η λύση του ακολούθου χρησιμοποιείται για τον υπολογισμό της τελικής λύσης του BLKP, διασφαλίζοντας ότι τηρούνται οι περιορισμοί και βελτιστοποιούνται οι στόχοι και των δύο.

**Διαδικασία Εκπαίδευσης**

**1. Δημιουργία Δεδομένων**

Τα δεδομένα εκπαίδευσης δημιουργήθηκαν με χρήση του ακριβούς αλγορίθμου του Mansi [28], που παρείχε βέλτιστες λύσεις για κάθε πρόβλημα. Χρησιμοποιήθηκαν δύο τύποι δεδομένων:

* **Uncorrelated** (UC): Τα κέρδη και τα βάρη των αντικειμένων δημιουργήθηκαν τυχαία.
* **Correlated** (C): Τα κέρδη των αντικειμένων σχετίζονται με τα βάρη τους, προσομοιώνοντας πραγματικά σενάρια.

Για κάθε πρόβλημα δημιουργήθηκαν 1.000 περιπτώσεις με 100 μεταβλητές ηγέτη και 100 μεταβλητές ακολούθου. Συνολικά χρησιμοποιήθηκαν 22.000 περιπτώσεις για εκπαίδευση, εκ των οποίων το 80% χρησιμοποιήθηκε για εκπαίδευση και το 20% για επικύρωση.

**2. Εποπτευόμενη Μάθηση (Supervised Learning)**

Το μοντέλο GNN εκπαιδεύτηκε με εποπτευόμενο τρόπο, με στόχο να προβλέψει τις μεταβλητές του ηγέτη βάσει των βέλτιστων λύσεων που δημιουργήθηκαν από τον ακριβή αλγόριθμο. Η λειτουργία απώλειας ήταν η binary cross-entropy, ενώ ο βελτιστοποιητής που χρησιμοποιήθηκε ήταν ο **Adam** με ρυθμό εκμάθησης 0.002 για 5.000 εποχές.

**Αξιολόγηση**

Το μοντέλο GNN συγκρίθηκε με τον ακριβή αλγόριθμο του Mansi [28], που χρησιμοποιήθηκε ως σημείο αναφοράς για την επίλυση του BLKP. Η αξιολόγηση επικεντρώθηκε σε διάφορους δείκτες:

* **Optimality** **Gap**: Η διαφορά μεταξύ της λύσης που παρήγαγε το μοντέλο GNN και της βέλτιστης λύσης. Ο μέσος όρος του gap ήταν περίπου 1.7%.
* **Χρόνος Υπολογισμού**: Το μοντέλο GNN ήταν 500 φορές ταχύτερο από τον ακριβή αλγόριθμο στην εύρεση εφικτών λύσεων.
* **Γενίκευση**: Το μοντέλο δοκιμάστηκε σε μεγαλύτερα προβλήματα από αυτά που εκπαιδεύτηκε και παρουσίασε χαμηλά optimality gaps ακόμη και σε μεγαλύτερες περιπτώσεις.

**Συμπέρασμα**

Οι ερευνητές κατέληξαν στο συμπέρασμα ότι η προσέγγιση με βάση το GNN για την επίλυση του Bilevel Knapsack Problem είναι αποδοτική και κλιμακούμενη. Το μοντέλο παρουσίασε καλή απόδοση σε τύπους δεδομένων correlated και uncorrelated, με μικρά optimality gaps και σημαντική μείωση του χρόνου υπολογισμού σε σύγκριση με παραδοσιακούς ακριβείς λύτες. Η δυνατότητα του μοντέλου να γενικεύει σε μεγαλύτερα μεγέθη προβλημάτων αναδεικνύει την πρακτικότητά του και την εφαρμογή του σε πραγματικά σενάρια.

1. **Αναδρομικά νευρωνικά δίκτυα (RNN) και Pointer Networks**

Οι ερευνητές στο [1], εστιάζουν στο **Πρόβλημα του Σακιδίου 0-1 (0-1 KP)**, όπου κάθε αντικείμενο είτε συμπεριλαμβάνεται είτε αποκλείεται από το σακίδιο, κάνοντάς το πρόβλημα δυαδικής απόφασης. Ο στόχος είναι να μεγιστοποιηθεί η συνολική αξία των επιλεγμένων αντικειμένων χωρίς να υπερβαίνεται η χωρητικότητα βάρους του σακιδίου. Η έρευνα επικεντρώνεται στη χρήση μιας καθαρά **βασισμένης σε δεδομένα προσέγγισης (data-driven approach)**, χωρίς να βασίζεται σε παραδοσιακές τεχνικές βελτιστοποίησης, για την παραγωγή προσεγγιστικών λύσεων μέσω βαθιάς μάθησης. Χρησιμοποιήθηκε ένας συνδυασμός **LSTM (Long Short-Term Memory)** και **Pointer Network**.

**Αρχιτεκτονική**

**1. Pointer Networks**

Η αρχιτεκτονική που χρησιμοποιείται στην εργασία βασίζεται στα **Pointer Networks**, τα οποία αρχικά σχεδιάστηκαν για προβλήματα όπως το **Traveling Salesman Problem (TSP)**. Τα κύρια στοιχεία αυτής της αρχιτεκτονικής είναι:

* **Δομή Encoder-Decoder**: Το Pointer Network ακολουθεί τη δομή **Seq2seq** (sequence-to-sequence). Ο **encoder** λαμβάνει μια ακολουθία εισόδου και την ενσωματώνει σε ένα διάνυσμα σταθερών διαστάσεων, ενώ ο **decoder** επεξεργάζεται αυτό το διάνυσμα για να παράγει την ακολουθία εξόδου.
* **Μηχανισμός Προσοχής**(**Attention Mechanism)**: Ο μηχανισμός προσοχής συνδέει τον encoder και τον decoder, επιτρέποντας στο μοντέλο να εστιάζει σε διαφορετικά μέρη της ακολουθίας εισόδου σε κάθε βήμα της διαδικασίας αποκωδικοποίησης. Ο μηχανισμός προσοχής μετατοπίζει δυναμικά την εστίαση σε διαφορετικές τιμές και βάρη αντικειμένων, επιτρέποντας πιο ακριβείς αποφάσεις.
* **RNN με LSTM Cells**: Ο encoder και ο decoder υλοποιούνται με χρήση **Long Short-Term Memory (LSTM)**, ένας τύπος RNN που είναι αποτελεσματικός στην αποθήκευση μακροπρόθεσμων εξαρτήσεων σε ακολουθίες, κάνοντάς τον κατάλληλο για τη μοντελοποίηση των διαδοχικών αποφάσεων στο πρόβλημα του σακιδίου.

**2. Αναπαράσταση Εισόδου**

Η είσοδος στο δίκτυο είναι μια ακολουθία συντελεστών που αντιπροσωπεύουν τον λόγο αξίας προς βάρος (value-to-weight ratio) κάθε αντικειμένου. Για παράδειγμα, οι συντελεστές της αξίας ενός αντικειμένου και του κανονικοποιημένου βάρους εισάγονται στο δίκτυο, επιτρέποντας στο μοντέλο να μάθει ποια αντικείμενα είναι πιο πιθανό να συμπεριληφθούν στη βέλτιστη λύση.

**3. Αναπαράσταση Εξόδου**

Η έξοδος του δικτύου είναι μια ακολουθία που προβλέπει ποια αντικείμενα θα πρέπει να συμπεριληφθούν στο σακίδιο. Το δίκτυο παράγει μια κατανομή πιθανοτήτων για τα πιθανά αντικείμενα και παράγει την τελική απόφαση σχετικά με το αν κάθε αντικείμενο θα περιληφθεί ή όχι (τιμή 0 ή 1).

**Διαδικασία Εκπαίδευσης**

**1. Δημιουργία Δεδομένων**

Το σύνολο εκπαίδευσης αποτελείται από τυχαία δημιουργημένα παραδείγματα του **0-1 Knapsack Problem**, όπου το συνολικό όριο βάρους είναι μικρότερο από το συνολικό βάρος όλων των αντικειμένων. Κάθε δείγμα περιλαμβάνει τους συντελεστές της συνάρτησης στόχου (τιμές), τους συντελεστές περιορισμού (βάρη) και τις ακριβείς λύσεις που λαμβάνονται μέσω ενός αλγορίθμου δυναμικού προγραμματισμού. Το σύνολο δεδομένων περιλαμβάνει:

* **10.000 δείγματα εκπαίδευσης** και **1.000 δείγματα δοκιμών**.
* Οι συντελεστές συνάρτησης στόχου, οι συντελεστές περιορισμού και οι ακριβείς λύσεις αποθηκεύονται σε **αρχεία κειμένου για εύκολη πρόσβαση κατά τη διάρκεια της εκπαίδευσης.**

**2. Διαδικασία Εκπαίδευσης**

Το μοντέλο εκπαιδεύεται μέσω **εποπτευόμενης μάθησης**(**supervised learning**). Η διαδικασία εκπαίδευσης περιλαμβάνει την εκμάθηση της αντιστοίχισης των προβλέψεων του δικτύου με τις βέλτιστες λύσεις που δημιουργούνται από τον αλγόριθμο δυναμικού προγραμματισμού. Στόχος είναι η ελαχιστοποίηση της απώλειας cross-entropy, η οποία μετρά το σφάλμα μεταξύ των προβλεπόμενων πιθανοτήτων και των πραγματικών βέλτιστων αποφάσεων.

Η εκπαίδευση περιλαμβάνει:

* **LSTM** layers με **128** ή **256 hidden units**.
* **Adam optimizer** με ρυθμό εκμάθησης 0.001.
* Οι παράμετροι αρχικοποιούνται στο διάστημα [-0.08, 0.08].
* Η εκπαίδευση πραγματοποιείται για **10-20 εποχές**, με πρώιμη διακοπή βάσει της απόδοσης επικύρωσης.

**Αξιολόγηση**

**1. Πείραμα 1: Επίλυση Προβλημάτων Σακιδίου Διαφορετικών Διαστάσεων**

Το μοντέλο αξιολογήθηκε σε τέσσερα σύνολα δεδομένων, KNAP10, KNAP100, KNAP500 και KNAP1000, που αντιπροσωπεύουν προβλήματα με 10, 100, 500 και 1000 αντικείμενα, αντίστοιχα. Παρατηρήθηκαν τα εξής αποτελέσματα:

* **Χρόνος Εκπαίδευσης(Training Time)**: Αυξήθηκε με τον αριθμό αντικειμένων, από 7 λεπτά για το KNAP10 έως 15 ώρες για το KNAP1000.
* **Χρόνος Υπολογισμού(Inference Time)**: Ο χρόνος που απαιτείται για την επίλυση μιας νέας περίπτωσης αυξήθηκε σχεδόν γραμμικά με το μέγεθος του προβλήματος, από 0.014 δευτερόλεπτα (KNAP10) σε 1.27 δευτερόλεπτα (KNAP1000).
* **Ακρίβεια Λύσεων(Solution Accuracy)**: Η ακρίβεια του μοντέλου μειώθηκε καθώς αυξανόταν το μέγεθος του προβλήματος, επιτυγχάνοντας 80% ακρίβεια για το KNAP10 αλλά μόνο 40% για το KNAP1000.

**2. Πείραμα 2: Γενίκευση σε Διαφορετικά Μεγέθη Προβλημάτων**

Το δεύτερο πείραμα δοκίμασε την ικανότητα του μοντέλου να γενικεύει σε προβλήματα διαφορετικών μεγεθών. Ένα μοντέλο εκπαιδευμένο σε προβλήματα 10 και 50 αντικειμένων αξιολογήθηκε με τα εξής αποτελέσματα:

* **Ακρίβεια**: Το μοντέλο πέτυχε 70% ακρίβεια για τα προβλήματα με 10 αντικείμενα και 57% για τα προβλήματα με 50 αντικείμενα.
* **Βέλτιστες** **Λύσεις**: Το μοντέλο παρήγαγε τη βέλτιστη λύση για το 30% των προβλημάτων με 10 αντικείμενα και για το 15% των προβλημάτων με 50 αντικείμενα.

**Δεδομένα που Χρησιμοποιήθηκαν**

Η έρευνα χρησιμοποίησε τυχαία δημιουργημένα σύνολα δεδομένων, ειδικά σχεδιασμένα για να προσομοιώνουν περιπτώσεις του Προβλήματος του Σακιδίου 0-1 με διαφορετικά μεγέθη. Η διαδικασία δημιουργίας δεδομένων εξασφάλισε ότι το όριο βάρους του σακιδίου ήταν μικρότερο από το συνολικό βάρος όλων των αντικειμένων, καθιστώντας το πρόβλημα μη τετριμμένο. Για κάθε σύνολο δεδομένων, χρησιμοποιήθηκαν 10.000 δείγματα εκπαίδευσης και 1.000 δείγματα δοκιμών αποθηκευμένα σε αρχεία κειμένου.

**Συμπέρασμα**

Οι ερευνητές κατέληξαν στο συμπέρασμα ότι η προσέγγιση βασισμένη στα Pointer Networks για την επίλυση του Προβλήματος του Σακιδίου 0-1 προσφέρει μια ελπιδοφόρα εναλλακτική στις παραδοσιακές ακριβείς και ευρετικές μεθόδους. Το μοντέλο μπορεί να παράγει κατά προσέγγιση λύσεις γρήγορα, καθιστώντας το κατάλληλο για μεγαλύτερες περιπτώσεις όπου οι παραδοσιακές μέθοδοι θα ήταν υπολογιστικά δαπανηρές. Ωστόσο, η ποιότητα των λύσεων μειώνεται καθώς αυξάνεται το μέγεθος του προβλήματος, υποδεικνύοντας ότι χρειάζονται περαιτέρω βελτιώσεις για να κλιμακωθεί το μοντέλο με μεγαλύτερη ακρίβεια.

1. **Δίκτυα Βαθιάς Ενισχυτικής Μάθησης (Deep Reinforcement Learning - Deep Q-Learning, DQN)**

Στο [29], οι ερευνητές χρησιμοποίησαν μια προσέγγιση DQN με πλήρως συνδεδεμένα επίπεδα, μηχανισμούς προσοχής και μοντέλα transformer για την ανάπτυξη μιας αποδοτικής λύσης για το 0-1 Knapsack Problem. Ο στόχος τους ήταν να ξεπεράσουν τις υπολογιστικές προκλήσεις των ακριβών αλγορίθμων χρησιμοποιώντας την ενισχυτική μάθηση (RL) για την παραγωγή προσεγγιστικών λύσεων κοντά στη βέλτιστη.

**Αρχιτεκτονική του Μοντέλου DQN**

**1. Πλήρως Συνδεδεμένα Επίπεδα**

Το πρώτο μοντέλο που χρησιμοποιήθηκε στη μελέτη αποτελείται από **τρία πλήρως συνδεδεμένα επίπεδα** με **128**, **196** και **128** κόμβους αντίστοιχα. Αυτά τα επίπεδα χρησιμοποιήθηκαν για να αναπαραστήσουν τη συνάρτηση αξίας κατάστασης-δράσης στο DQN. Το νευρωνικό δίκτυο αρχικοποιήθηκε με τον **GlorotUniform** **initializer**, και η έξοδος του δικτύου παρείχε μια εκτίμηση των τιμών Q για κάθε δράση.

**2. Μηχανισμός Προσοχής**

Το δεύτερο μοντέλο εισάγει έναν **μηχανισμό προσοχής dot-product**. Σε αυτόν τον μηχανισμό, το μοντέλο υπολογίζει μια βαθμολογία παίρνοντας το εσωτερικό γινόμενο των τιμών query και key, που προέρχονται από τα χαρακτηριστικά εισόδου. Μετά την εφαρμογή της **συνάρτησης softmax**, το μοντέλο αποδίδει βάρη σε διαφορετικά αντικείμενα, δίνοντας έμφαση στα πιο σημαντικά. Μετά το επίπεδο προσοχής, εφαρμόστηκαν δύο πλήρως συνδεδεμένα επίπεδα (96 και 64 κόμβοι).

**3. Transformer Encoder Block**

Το τρίτο μοντέλο χρησιμοποιεί **δύο μπλοκ encoder transformer**. Σε αυτήν την αρχιτεκτονική, το μοντέλο εφαρμόζει έναν μηχανισμό multi-head self-attention, που επεξεργάζεται την είσοδο παράλληλα για να συλλάβει σύνθετες εξαρτήσεις μεταξύ των αντικειμένων. Το μοντέλο transformer ακολουθήθηκε από δύο πλήρως συνδεδεμένα επίπεδα (128 και 96 κόμβοι). Δεδομένου ότι η είσοδος ήταν αριθμητική, δεν χρησιμοποιήθηκαν embeddings. Αντί αυτού, η είσοδος μετασχηματίστηκε απευθείας σε έναν χώρο υψηλότερων διαστάσεων, και χρησιμοποιήθηκε γραμμική συνάρτηση ενεργοποίησης αντί για softmax.

**4. Απώλεια και Βελτιστοποιητής**

Όλα τα μοντέλα εκπαιδεύτηκαν με χρήση του **Adam optimizer** και της συνάρτησης απώλειας **Huber** με τιμή δέλτα 2.0, η οποία εξισορροπεί μεταξύ του **Mean Absolute Error (MAE)** και του **Mean Square Error (MSE)**, καθιστώντας την αποτελεσματική για δεδομένα με μεταβλητότητα. Οι ερευνητές δεν αναφέρουν την τιμή του ρυθμού εκμάθησης (learning rate) που χρησιμοποιήθηκε.

**Διαδικασία Εκπαίδευσης**

Η διαδικασία εκπαίδευσης περιλάμβανε την εκτέλεση του DQN σε ένα προσομοιωμένο περιβάλλον σακιδίου συμβατό με τη βιβλιοθήκη OpenAI Gym. Τα κύρια σημεία της διαδικασίας εκπαίδευσης περιλαμβάνουν:

**1. Αρχικοποίηση και Εξερεύνηση**

Το περιβάλλον προσομοίωσης ξεκινά με ένα σακίδιο που περιέχει 30 αντικείμενα, με τυχαίες τιμές για βάρος, όγκο και αξία. Το DQN ξεκινά σε μια φάση εξερεύνησης(exploration), όπου η τιμή epsilon είναι 1, πράγμα που σημαίνει ότι ο πράκτορας επιλέγει τυχαίες ενέργειες για να εξερευνήσει τον χώρο καταστάσεων. Καθώς προχωρά η εκπαίδευση, το epsilon μειώνεται σταδιακά σε 0.001, επιτρέποντας στον πράκτορα να μεταβεί στη φάση εκμετάλλευσης(exploitation), όπου βασίζεται στις μαθημένες εμπειρίες.

**2. Επαναναπαραγωγή Εμπειριών(Experience Replay) και Q-Networks**

Η αναπαραγωγή εμπειριών(experience replay) αποθηκεύει μεταβάσεις (state, action, reward, next state) κατά τη διάρκεια της προσομοίωσης, επιτρέποντας στο DQN να μαθαίνει από προηγούμενες εμπειρίες. Το μοντέλο χρησιμοποιεί δίδυμα Q-networks για να σταθεροποιήσει τη διαδικασία μάθησης. Ένα δίκτυο χρησιμοποιείται για να επιλέγει ενέργειες (main network), και ένα άλλο ενημερώνεται περιοδικά (target network για να υπολογίζει τις μελλοντικές τιμές Q.

**3. Παράγοντας Έκπτωσης(Discount Factor) και Εξέλιξη Epsilon (Epsilon Decay)**

Ένας παράγοντας έκπτωσης εφαρμόζεται, δίνοντας προτεραιότητα στις άμεσες ανταμοιβές έναντι των μελλοντικών ανταμοιβών. Το μοντέλο DQN εξισορροπεί την εξερεύνηση και την εκμετάλλευση μέσω της **epsilon-greedy exploration**, όπου η πιθανότητα επιλογής τυχαίων ενεργειών μειώνεται όσο προχωρά η μάθηση. Χρησιμοποιήθηκε μεγάλος χώρος αποθήκευσης εμπειριών(experience replay memory size) 50.000 για να διασφαλιστεί ότι το μοντέλο μαθαίνει αποτελεσματικά κατά τη διάρκεια πολλών επεισοδίων.

**Αξιολόγηση**

**1. Σχεδίαση Πειράματος**

Τα μοντέλα δοκιμάστηκαν λύνοντας διάφορα προβλήματα σακιδίου, με τυχαία ανατεθειμένα χαρακτηριστικά αντικειμένων (αξία, βάρος και όγκος). Η αξιολόγηση συνέκρινε τα μοντέλα DQN (πλήρως συνδεδεμένα επίπεδα, attention, transformer) με δυναμικό προγραμματισμό, μια ακριβή μέθοδο επίλυσης, για να εκτιμηθούν η ταχύτητα και η ακρίβεια. Η συνολική **χωρητικότητα του σακιδίου** ορίστηκε στα **1000** **για το βάρος και τον όγκο.**

**2. Δείκτες Απόδοσης**

* **Συνολική Ανταμοιβή(Total Reward)**: Η συνολική ανταμοιβή είναι το άθροισμα των αξιών των επιλεγμένων αντικειμένων σε κάθε σακίδιο. Καθώς προχωρούσε η εκπαίδευση, τα μοντέλα DQN βελτιστοποίησαν τις πολιτικές τους για τη μεγιστοποίηση της συνολικής ανταμοιβής, με το μοντέλο transformer να επιτυγχάνει τις υψηλότερες ανταμοιβές, ακολουθούμενο από τα μοντέλα προσοχής και πλήρως συνδεδεμένων επιπέδων.
* **Ταχύτητα**: Τα μοντέλα DQN ξεπέρασαν σημαντικά τον αλγόριθμο δυναμικού προγραμματισμού όσον αφορά τον χρόνο υπολογισμού, όντας 40 φορές ταχύτερα. Η ταχύτητα επεξεργασίας των μοντέλων DQN αυξήθηκε γραμμικά με τον αριθμό των αντικειμένων, καθιστώντας τα κατάλληλα για μεγαλύτερες περιπτώσεις προβλημάτων.

**3. Αποτελέσματα**

Το μοντέλο DQN με βάση το transformer παρουσίασε την καλύτερη συνολική απόδοση, πλησιάζοντας περισσότερο στη βέλτιστη λύση, διατηρώντας ταυτόχρονα υψηλή υπολογιστική αποδοτικότητα.

**Δεδομένα που Χρησιμοποιήθηκαν**

Το σύνολο δεδομένων περιλάμβανε 30 αντικείμενα ανά σακίδιο, με τυχαία ανατεθειμένες τιμές για βάρος, όγκο και αξία αντικειμένων. Ο χώρος κατάστασης περιλάμβανε χαρακτηριστικά όπως η υπολειπόμενη χωρητικότητα βάρους, η υπολειπόμενη χωρητικότητα όγκου, τα χαρακτηριστικά του τρέχοντος αντικειμένου και η συνολική αξία των αντικειμένων που τοποθετήθηκαν στο σακίδιο. Αυτά τα ποικίλα δεδομένα επέτρεψαν στο μοντέλο να μάθει πώς να εξισορροπεί την αξία των αντικειμένων και το βάρος για τη μεγιστοποίηση της συνολικής ανταμοιβής.

**Συμπέρασμα**

Οι ερευνητές κατέληξαν στο συμπέρασμα ότι η προσέγγιση DQN, ιδιαίτερα όταν ενισχύεται με μηχανισμούς προσοχής και μπλοκ encoder transformer, παρέχει μια αποδοτική και αποτελεσματική λύση στο πρόβλημα του σακιδίου. Το μοντέλο DQN με βάση το transformer παρήγαγε λύσεις κοντά στη βέλτιστη, σε ένα μικρό μέρος του χρόνου που απαιτούν οι ακριβείς αλγόριθμοι. Επιπλέον, το μοντέλο γενίκευσε καλά σε μεγαλύτερες περιπτώσεις προβλημάτων και διατήρησε γραμμική αύξηση στον χρόνο υπολογισμού, καθιστώντας το κατάλληλο για την επίλυση προβλημάτων μεγαλύτερης κλίμακας.

1. **Feed-Forward Neural Networks (Deep Neural Networks, DNNs)**

Στο [30], οι ερευνητές παρουσιάζουν μια προσέγγιση βαθιάς μη επιβλεπόμενης (unsupervised) μάθησης χρησιμοποιώντας ένα **Feed-Forward Deep Neural Network (DNN)** για την επίλυση του **Γενικευμένου Προβλήματος Ανάθεσης (Generalized Assignment Problem, GAP)**. Το GAP είναι ένα πρόβλημα συνδυαστικής βελτιστοποίησης που επεκτείνει το κλασικό πρόβλημα του σακιδίου με εισαγωγή τόσο ισότιμων όσο και ανισότιμων περιορισμών, κάνοντάς το πιο πολύπλοκο. Η μελέτη περίπτωσης επικεντρώνεται στην επίλυση του προβλήματος ανάθεσης χρηστών σε σταθμούς βάσης σε ασύρματα δίκτυα, το οποίο διαμορφώνεται ως GAP, με στόχο την ανάθεση χρηστών σε σταθμούς βάσης, διατηρώντας τους περιορισμούς του συστήματος.

Το πρόβλημα που εξετάζεται είναι μια παραλλαγή του προβλήματος του σακιδίου, συγκεκριμένα το **Γενικευμένο Πρόβλημα Ανάθεσης (GAP)**. Στο GAP, πολλά αντικείμενα (ή πόροι) πρέπει να ανατεθούν σε πολλούς «σακίδια» (ή πράκτορες), διασφαλίζοντας ότι δεν υπερβαίνονται οι περιορισμοί χωρητικότητας κάθε σακιδίου. Σε αυτό το πλαίσιο, το πρόβλημα είναι η ανάθεση χρηστών (αντικειμένων) σε σταθμούς βάσης (σακίδια), με στόχο τη μεγιστοποίηση της συνολικής απόδοσης του συστήματος, όπως ο ρυθμός δεδομένων, ενώ διασφαλίζεται ότι κανένας σταθμός βάσης δεν υπερβαίνει τη χωρητικότητά του.

**Αρχιτεκτονική του Μοντέλου DNN**

Η προτεινόμενη αρχιτεκτονική DNN είναι ένα πλήρως **συνδεδεμένο feed-forward νευρωνικό δίκτυο** σχεδιασμένο να μαθαίνει τις βέλτιστες αναθέσεις χρηστών-σταθμών βάσης απευθείας από τα δεδομένα, χωρίς την ανάγκη για επιβλεπόμενη εκπαίδευση με ετικέτες. Τα βασικά στοιχεία της αρχιτεκτονικής είναι τα εξής:

**1. Επίπεδο Εισόδου**

* Η είσοδος στο DNN είναι ένα διανυσματικό χαρακτηριστικό που αναπαριστά τους χρήστες και τους σταθμούς βάσης. Κάθε διάνυσμα περιέχει πληροφορίες όπως η αναλογία σήματος προς θόρυβο (signal-to-noise ratio, SNR), η απόσταση του χρήστη και η χωρητικότητα του σταθμού βάσης. Για τη μελέτη περίπτωσης, η διάσταση εισόδου ποικίλλει ανάλογα με τον αριθμό χρηστών και σταθμών βάσης, με το μέγεθος του διανύσματος εισόδου να είναι , όπου είναι ο αριθμός των χρηστών και ο αριθμός των σταθμών βάσης.

**2. Κρυφά Επίπεδα**

* Τα κρυφά επίπεδα του DNN χρησιμοποιούν συναρτήσεις ενεργοποίησης ReLU. Ο αριθμός των νευρώνων σε κάθε κρυφό επίπεδο αυξάνεται προοδευτικά, ξεκινώντας από 64 νευρώνες και φτάνοντας έως και 4096 νευρώνες στη μεγαλύτερη διάταξη. Αυτά τα επίπεδα συλλαμβάνουν σύνθετες σχέσεις μεταξύ των χρηστών και των σταθμών βάσης, επιτρέποντας στο δίκτυο να προσεγγίσει τη βέλτιστη στρατηγική ανάθεσης.

**3. Επίπεδο Εξόδου με Tensor Splitting**

* Το επίπεδο εξόδου διαχωρίζεται σε πολλούς tensors, καθένας από τους οποίους αντιστοιχεί σε έναν χρήστη. Κάθε tensor χρησιμοποιεί μια Softmax συνάρτηση ενεργοποίησης για να διασφαλίσει ότι οι ισότιμοι περιορισμοί του προβλήματος ικανοποιούνται (δηλαδή, κάθε χρήστης ανατίθεται ακριβώς σε έναν σταθμό βάσης). Αυτή η μοναδική προσέγγιση διαχωρισμού tensor απλοποιεί τη συνάρτηση απώλειας του μοντέλου και εγγυάται την ικανοποίηση των περιορισμών.

**Διαδικασία Εκπαίδευσης**

Η εκπαίδευση του DNN πραγματοποιείται με **μη επιβλεπόμενο τρόπο(unsupervised)**, που σημαίνει ότι δεν παρέχονται ετικέτες με λύσεις. Αντίθετα, το DNN μαθαίνει να βελτιστοποιεί τη συνάρτηση στόχου(objective function) απευθείας μέσω μιας προσαρμοσμένης συνάρτησης απώλειας(**custom loss function**) που ενσωματώνει τους περιορισμούς του GAP.

**1. Συνάρτηση Απώλειας (Loss Function)**

Η προσαρμοσμένη συνάρτηση απώλειας αποτελείται από δύο κύρια στοιχεία:

* **Objective Term**: Αυτός ο όρος μεγιστοποιεί τη συνολική απόδοση του συστήματος, όπως ο συνολικός **ρυθμός** δεδομένων.
* **Penalty Terms**: Οι όροι ποινής διασφαλίζουν ότι οι περιορισμοί ικανοποιούνται. Περιλαμβάνουν όρους που διασφαλίζουν ότι κάθε χρήστης ανατίθεται ακριβώς σε έναν σταθμό βάσης (ισότιμος περιορισμός, equality constraint) και ότι κανένας σταθμός βάσης δεν υπερβαίνει τη χωρητικότητά του (ανισότιμος περιορισμός, inequality constraint). Η συνάρτηση απώλειας ρυθμίζεται χρησιμοποιώντας δύο υπερπαραμέτρους, **λ1** και **λ2**, που ελέγχουν την ισορροπία μεταξύ βελτιστοποίησης του στόχου και ικανοποίησης των περιορισμών.

**2. Βελτιστοποίηση (Optimization)**

Το δίκτυο εκπαιδεύεται χρησιμοποιώντας τον Adam optimizer, έναν δημοφιλή αλγόριθμο βελτιστοποίησης βασισμένο σε gradients. Ο ρυθμός εκμάθησης ρυθμίζεται προσεκτικά για να επιτευχθεί ισορροπία μεταξύ γρήγορης σύγκλισης και σταθερότητας. Χρησιμοποιείται mini-batch με 128 δείγματα ανά παρτίδα για αποτελεσματική εκπαίδευση.

**3. Διαχείριση Παραβιάσεων Περιορισμών**

Κατά την εκπαίδευση, παρακολουθείται η πιθανότητα παραβίασης των περιορισμών και γίνονται προσαρμογές στη συνάρτηση απώλειας και στους όρους ποινής. Με τη δυναμική προσαρμογή αυτών των ποινών, το δίκτυο μαθαίνει να αποφεύγει τις παραβιάσεις των περιορισμών ενώ εξακολουθεί να βελτιστοποιεί τη συνάρτηση στόχου.

**Αξιολόγηση**

**1. Ρύθμιση Συστήματος**

Το μοντέλο δοκιμάστηκε σε ένα προσομοιωμένο ασύρματο δίκτυο αποτελούμενο από 4 σταθμούς βάσης (BSs) και 4 χρήστες στο πρώτο σενάριο, και 16 χρήστες σε ένα μεγαλύτερο σενάριο. Ο στόχος ήταν η βελτιστοποίηση του συνολικού ρυθμού δεδομένων, ενώ διασφαλίστηκε ότι κάθε σταθμός βάσης εξυπηρετούσε περιορισμένο αριθμό χρηστών.

**2. Δείκτες Απόδοσης**

* **Συνολικός Ρυθμός Δεδομένων (Sum Data Rate)**: Ο κύριος δείκτης που χρησιμοποιήθηκε για την αξιολόγηση του μοντέλου είναι ο συνολικός ρυθμός δεδομένων, ο οποίος μετρά τη συνολική απόδοση του συστήματος.
* **Πιθανότητα Παραβίασης Περιορισμών (Constraint Violation Probability)**: Αυτός ο δείκτης παρακολουθεί πόσο συχνά το δίκτυο παραβιάζει τους περιορισμούς χωρητικότητας των σταθμών βάσης. Μικρότερες πιθανότητες παραβίασης υποδεικνύουν καλύτερη τήρηση των περιορισμών.

**3. Αποτελέσματα**

* Στο σενάριο με 4 χρήστες, το DNN πέτυχε 98.68% του βέλτιστου ρυθμού δεδομένων, διατηρώντας παράλληλα χαμηλή πιθανότητα παραβίασης περιορισμών, 0.094.
* Στο σενάριο με 16 χρήστες, το μοντέλο πέτυχε 99.87% του βέλτιστου ρυθμού δεδομένων, αποδεικνύοντας την επεκτασιμότητά του σε μεγαλύτερα προβλήματα. Η μέση πιθανότητα παραβίασης περιορισμών παρέμεινε στο 0.094.

Το DNN ξεπέρασε σημαντικά τους παραδοσιακούς λύτες όσον αφορά την υπολογιστική αποδοτικότητα, μειώνοντας την πολυπλοκότητα του χρόνου κατά περίπου 1000 φορές σε σύγκριση με τη βέλτιστη λύση που επιτεύχθηκε χρησιμοποιώντας convex optimization (CVX).

**Δεδομένα που Χρησιμοποιήθηκαν**

Τα σύνολα δεδομένων για εκπαίδευση και δοκιμές δημιουργήθηκαν συνθετικά, προσομοιώνοντας τις αλληλεπιδράσεις χρηστών και σταθμών βάσης σε ένα ασύρματο δίκτυο. Τα σύνολα δεδομένων περιλάμβαναν:

* χρήστες και 4 σταθμούς βάσης για το αρχικό σενάριο, με 10.000 δείγματα εκπαίδευσης και 1.000 δείγματα δοκιμών.
* 16 χρήστες και 4 σταθμούς βάσης για το μεγαλύτερο σενάριο, με 16.000 δείγματα εκπαίδευσης και 1.000 δείγματα δοκιμών.

Κάθε δείγμα περιελάμβανε χαρακτηριστικά όπως η απόσταση των χρηστών από τους σταθμούς βάσης, η αναλογία σήματος προς θόρυβο και η χωρητικότητα του σταθμού βάσης.

Συμπέρασμα

Η μελέτη έδειξε ότι η προτεινόμενη προσέγγιση βαθιάς μη επιβλεπόμενης μάθησης (DUL) χρησιμοποιώντας ένα DNN μπορεί να επιλύσει αποτελεσματικά το γενικευμένο πρόβλημα ανάθεσης, ιδιαίτερα για το πρόβλημα ανάθεσης χρηστών σε ασύρματα δίκτυα. Το DNN κατάφερε να επιτύχει σχεδόν βέλτιστη απόδοση με σημαντικά χαμηλότερη υπολογιστική πολυπλοκότητα σε σύγκριση με τις παραδοσιακές μεθόδους βελτιστοποίησης. Η καινοτόμος χρήση του tensor splitting και της προσαρμοσμένης συνάρτησης απώλειας εξασφάλισε ότι οι περιορισμοί ικανοποιήθηκαν, καθιστώντας το μοντέλο τόσο πρακτικό όσο και επεκτάσιμο για μεγαλύτερα προβλήματα.

1. **Spiking Neural Networks (SNNs)**

Στο [31], οι ερευνητές εξετάζουν τη χρήση των **Spiking Neural Networks (SNNs)** στο πλαίσιο του membrane computing για την επίλυση προβλημάτων συνδυαστικής βελτιστοποίησης. Η έμφαση δίνεται στο **Πρόβλημα Σακιδίου 0/1 (KP)**, ένα κλασικό πρόβλημα NP-hard στη συνδυαστική βελτιστοποίηση. Η εργασία παρουσιάζει ένα βελτιωμένο μοντέλο, το **Distributed Adaptive Optimization Spiking Neural P System (DAOSNPS)**, το οποίο βελτιώνει προηγούμενα μοντέλα βασισμένα σε SNNs εισάγοντας μια δομή κατανεμημένου πληθυσμού και έναν προσαρμοστικό ρυθμό εκμάθησης που λαμβάνει υπόψη τη διαφοροποίηση του πληθυσμού.

**Αρχιτεκτονική του Μοντέλου SNN**

**1. Εκτεταμένο Σύστημα Νευρώνων Αιχμής P (ESNPS)**

Η βασική αρχιτεκτονική για την επίλυση του προβλήματος σακιδίου είναι το **Extended Spiking Neural P System (ESNPS)**, το οποίο αποτελείται από νευρώνες. Το ESNPS δομείται ως εξής:

* **Νευρώνες**: Υπάρχουν m νευρώνες εργασίας και δύο νευρώνες τροφοδοσίας, που παρέχουν spikes στους υπόλοιπους νευρώνες του συστήματος. Κάθε νευρώνας μεταδίδει δυαδικά spikes (0 ή 1), τα οποία αντιπροσωπεύουν αποφάσεις για το αν θα συμπεριληφθεί ή όχι ένα αντικείμενο στο σακίδιο.
* **Κανόνες** **Πυροδότησης**: Κάθε νευρώνας ακολουθεί συγκεκριμένους κανόνες πυροδότησης που καθορίζουν πότε εκπέμπεται ένα spike. Αυτοί οι κανόνες διέπονται από ένα σύνολο πιθανοτήτων, και κάθε νευρώνας μπορεί είτε να πυροδοτήσει (εκπέμψει spike) είτε να ξεχάσει (επαναφέρει) την κατάστασή του.
* **Αλγόριθμος** **Καθοδήγησης** (**Guider Algorithm**): Αυτός ο αλγόριθμος προσαρμόζει την πιθανότητα πυροδότησης για κάθε νευρώνα βάσει της απόδοσης της τρέχουσας λύσης, διασφαλίζοντας ότι το σύστημα εξελίσσεται προς καλύτερες λύσεις.

**2. Δομή Κατανεμημένου Πληθυσμού (Distributed Population Structure)**

Το Distributed Adaptive Optimization Spiking Neural P System (DAOSNPS) βελτιώνει το ESNPS εισάγοντας μια κατανεμημένη δομή πληθυσμού. Σε αυτή τη δομή, ο πληθυσμός διαιρείται σε αρκετούς υποπληθυσμούς, και κάθε υποπληθυσμός εξελίσσεται ανεξάρτητα. Αυτή η διάταξη προάγει τη διαφοροποίηση και επιτρέπει στο σύστημα να εξερευνήσει πολλαπλές λύσεις παράλληλα. Τα κύρια στοιχεία περιλαμβάνουν:

* **Υποπληθυσμοί**: Κάθε υποπληθυσμός διαθέτει το δικό του σύνολο ESNPS που εξελίσσεται ανεξάρτητα.
* **Ανταλλαγή** **Πληροφοριών**: Σε συγκεκριμένα χρονικά διαστήματα, ανταλλάσσονται πληροφορίες (δηλαδή, οι καλύτερες λύσεις) μεταξύ των υποπληθυσμών για τη βελτίωση της συνολικής διαδικασίας βελτιστοποίησης.
* **Στρατηγική** **Μετανάστευσης**: Τα άτομα από έναν υποπληθυσμό μπορούν να μεταναστεύσουν σε άλλους, βοηθώντας το σύστημα να διατηρήσει τη διαφοροποίηση και να αποφύγει τη γρήγορη σύγκλιση.

**3. Προσαρμοστικός Ρυθμός Εκμάθησης**

Ένας προσαρμοστικός ρυθμός εκμάθησης εφαρμόζεται για να προσαρμόσει την πιθανότητα πυροδότησης των νευρώνων. Ο ρυθμός εκμάθησης προσαρμόζεται δυναμικά με βάση τη διαφοροποίηση του πληθυσμού, διασφαλίζοντας ότι το βήμα για την ενημέρωση των πιθανοτήτων είναι μεγαλύτερο όταν ο πληθυσμός είναι λιγότερο διαφοροποιημένος και μικρότερο όταν ο πληθυσμός είναι πιο διαφοροποιημένος. Αυτός ο μηχανισμός βοηθά στην εξισορρόπηση μεταξύ της εξερεύνησης(exploration) (αναζήτηση νέων λύσεων) και της εκμετάλλευσης(exploitation) (βελτίωση των υφιστάμενων λύσεων).

**Διαδικασία Εκπαίδευσης**

Το μοντέλο DAOSNPS εκπαιδεύεται χρησιμοποιώντας έναν συνδυασμό εξελικτικών στρατηγικών και μη επιβλεπόμενης μάθησης(**unsupervised** **learning**). Τα βασικά σημεία της διαδικασίας εκπαίδευσης περιλαμβάνουν:

* **Αρχικοποίηση**: Το σύστημα ξεκινά με έναν αρχικό πληθυσμό τυχαία παραγόμενων λύσεων.
* **Αξιολόγηση** **Καταλληλότητας (Fitness Evaluation)**: Κάθε λύση αξιολογείται βάσει της καταλληλότητάς της, η οποία, στην περίπτωση του προβλήματος σακιδίου, καθορίζεται από τη συνολική αξία των επιλεγμένων αντικειμένων μείον τυχόν ποινές για την υπέρβαση της χωρητικότητας.
* **Εξέλιξη**: Ο πληθυσμός εξελίσσεται μέσα από πολλές γενιές, με τα άτομα να επιλέγονται βάσει της καταλληλότητάς τους για τη δημιουργία νέων λύσεων. Η εξέλιξη περιλαμβάνει διαδικασίες μετάλλαξης και crossover, καθώς και τη στρατηγική μετανάστευσης που αναφέρθηκε προηγουμένως.
* **Προσαρμογή** **Ρυθμού** **Εκμάθησης**: Ο προσαρμοστικός ρυθμός εκμάθησης διασφαλίζει ότι το σύστημα προσαρμόζει τη συμπεριφορά αναζήτησης με την πάροδο του χρόνου, εστιάζοντας περισσότερο καθώς προσεγγίζονται οι βέλτιστες λύσεις.

**Αξιολόγηση**

Το DAOSNPS δοκιμάστηκε σε πολλαπλές περιπτώσεις του Προβλήματος Σακιδίου 0/1, με τον αριθμό των αντικειμένων να κυμαίνεται από 1.000 έως 10.000. Τα κριτήρια αξιολόγησης περιλάμβαναν:

**Καταλληλότητα(Fitness)**: Ο κύριος δείκτης που χρησιμοποιήθηκε ήταν η τιμή καταλληλότητας των λύσεων, η οποία αντιπροσωπεύει τη συνολική αξία των αντικειμένων στο σακίδιο.

**Σύγκλιση(Convergence)**: Η τυπική απόκλιση των τιμών καταλληλότητας μετρήθηκε για να εκτιμηθεί η σύγκλιση του αλγορίθμου.

**Διαφοροποίηση(Diversity)**: Αξιολογήθηκε η ικανότητα του συστήματος να διατηρεί διαφοροποίηση στον πληθυσμό, διασφαλίζοντας ότι ο αλγόριθμος δεν συγκλίνει πρόωρα σε υποβέλτιστες λύσεις.

**Αποτελέσματα**

**Βελτιωμένη** **Απόδοση**: Το DAOSNPS ξεπέρασε προηγούμενα Spiking Neural Networks (OSNPS και AOSNPS), καθώς και άλλους αλγορίθμους όπως ο Genetic Quantum Algorithm (GQA) και ο Novel Quantum Evolutionary Algorithm (NQEA).

**Σύγκλιση**: Το DAOSNPS έδειξε ταχύτερη σύγκλιση προς λύσεις υψηλής ποιότητας σε σύγκριση με το OSNPS και το AOSNPS.

**Διαφοροποίηση** **Πληθυσμού**: Η κατανεμημένη δομή του πληθυσμού βοήθησε στη διατήρηση της διαφοροποίησης καθ' όλη τη διάρκεια της εξελικτικής διαδικασίας, βελτιώνοντας την ικανότητα του συστήματος να εξερευνά νέες λύσεις.

**Δεδομένα** **που** **Χρησιμοποιήθηκαν**

Τα πειράματα διεξήχθησαν χρησιμοποιώντας συνθετικά δεδομένα που δημιουργήθηκαν για το Πρόβλημα Σακιδίου 0/1, με τον αριθμό των αντικειμένων να κυμαίνεται από 1.000 έως 10.000. Σε κάθε αντικείμενο ανατέθηκε μια τυχαία αξία και βάρος, και η χωρητικότητα του σακιδίου ορίστηκε ως το ήμισυ του συνολικού βάρους όλων των αντικειμένων. Το ίδιο σύνολο δεδομένων χρησιμοποιήθηκε για να συγκριθεί η απόδοση του DAOSNPS με άλλους αλγορίθμους βελτιστοποίησης.

**Συμπέρασμα**

Η εισαγωγή του Distributed Adaptive Optimization Spiking Neural P System (DAOSNPS) αποτελεί σημαντική βελτίωση στην επίλυση του Προβλήματος Σακιδίου 0/1. Ο συνδυασμός της κατανεμημένης δομής πληθυσμού και του προσαρμοστικού ρυθμού εκμάθησης επέτρεψε στο σύστημα να επιτύχει καλύτερες τιμές καταλληλότητας και να διατηρήσει τη διαφοροποίηση του πληθυσμού κατά τη διάρκεια της διαδικασίας βελτιστοποίησης. Το DAOSNPS ξεπέρασε προηγούμενα Spiking Neural Networks και άλλους αλγορίθμους εμπνευσμένους από κβαντική υπολογιστική, αποδεικνύοντας τη δυνατότητά του ως ένα ισχυρό εργαλείο για την επίλυση προβλημάτων συνδυαστικής βελτιστοποίησης.

1. **Αναδρομικά Νευρωνικά Δίκτυα με Αρχιτεκτονική Δυναμικού Προγραμματισμού (RNN with Dynamic Programming-based Architecture)**

Στο [32], οι ερευνητές παρουσιάζουν μια νέα προσέγγιση για την επίλυση του **Προβλήματος Σακιδίου 0/1 (KP)** με τη χρήση **Recurrent Neural Networks (RNNs)**. Η συγκεκριμένη μελέτη επικεντρώνεται στη χρήση RNNs για τον υπολογισμό είτε ακριβών είτε σχεδόν βέλτιστων λύσεων μέσω μιας αρχιτεκτονικής που μιμείται τον δυναμικό προγραμματισμό (dynamic programming, DP).

**Αρχιτεκτονική του Μοντέλου RNN**

Η μελέτη παρουσιάζει δύο βασικές αρχιτεκτονικές RNN για την επίλυση του Προβλήματος του Σακιδίου:

**Dynamic Programming Neural Network (DP-NN):**

* **Βάθος**: 4 επίπεδα.
* **Πλάτος**: , όπου είναι το ανώτατο όριο αξίας της βέλτιστης λύσης. Το πλάτος αυξάνεται τετραγωνικά με την αξία, με αποτέλεσμα ένα δίκτυο ψευδο-πολυωνυμικού μεγέθους.
* **Κρυμμένες** **Μονάδες(hidden units)**: Τα κρυφά επίπεδα του δικτύου περιλαμβάνουν διάφορους αριθμούς νευρώνων, με το δεύτερο κρυφό επίπεδο να περιέχει νευρώνες, όπου είναι η παράμετρος αξίας. Αυτό επιτρέπει στο δίκτυο να παρακολουθεί την πρόοδο της αξίας σε όλα τα αντικείμενα.

**Functionality**: Αυτή η αρχιτεκτονική μιμείται ακριβώς τη σχέση αναδρομής του dynamic programming, επεξεργαζόμενη τα αντικείμενα και ενημερώνοντας το χώρο λύσης. Υπολογίζει την ακριβή λύση μέσω αναδρομικών μεταβάσεων καταστάσεων.

**Fully Polynomial-Time Approximation Scheme Neural Network (FPTAS-NN):**

* **Βάθος**: 5 επίπεδα.
* **Πλάτος**: Σταθερό στο , όπου είναι ο αριθμός αντικειμένων και ο επιθυμητός λόγος προσέγγισης (approximation ratio).
* **Functionality**: Το FPTAS-NN ανταλλάσσει την ακρίβεια με την υπολογιστική απόδοση, εφαρμόζοντας μια διαδικασία στρογγυλοποίησης στις τιμές αξίας. Εξασφαλίζει ότι το δίκτυο υπολογίζει σχεδόν βέλτιστες λύσεις με εγγυημένο λόγο προσέγγισης , όπου είναι το πλάτος του δικτύου.

Και οι δύο αρχιτεκτονικές χρησιμοποιούν Rectified Linear Units (ReLUs) ως συνάρτηση ενεργοποίησης, επιτρέποντας στο δίκτυο να προσομοιώνει αποτελεσματικά τις ενημερώσεις του dynamic programming με γραμμικό τρόπο.

**Εκπαίδευση**

Η εκπαίδευση αυτών των μοντέλων RNN ακολουθεί μια διαδικασία unsupervised learning, η οποία μιμείται τη δομή της αναδρομής του dynamic programming:

* **DP-NN Εκπαίδευση**: Το DP-NN εκπαιδεύεται να αναπαράγει ακριβώς τη διαδικασία του dynamic programming μέσω του υπολογισμού των αναδρομικών μεταβάσεων καταστάσεων. Οι παράμετροι του δικτύου προσαρμόζονται ώστε να διασφαλιστεί ότι μπορεί να υπολογίσει σωστά τις ελάχιστες απαιτήσεις για κάθε επίπεδο κέρδους.
* **FPTAS-NN Εκπαίδευση**: Το FPTAS-NN εκπαιδεύεται για να προσεγγίσει τη βέλτιστη λύση διατηρώντας παράλληλα την υπολογιστική απόδοση. Χρησιμοποιεί τεχνικές στρογγυλοποίησης για τον περιορισμό του χώρου καταστάσεων και μαθαίνει να ελαχιστοποιεί το σφάλμα προσέγγισης(approximation error).

Και τα δύο δίκτυα εκπαιδεύονται χρησιμοποιώντας gradient-based optimization, όπου οι παράμετροι προσαρμόζονται σταδιακά για να ελαχιστοποιηθεί το σφάλμα στους υπολογισμούς των καταστάσεων του dynamic programming.

**Αξιολόγηση**

Η αξιολόγηση των μοντέλων πραγματοποιήθηκε με διάφορες παραλλαγές του Προβλήματος του Σακιδίου, εστιάζοντας σε σημαντικές μετρικές:

* **Ακρίβεια**: Το DP-NN παρήγαγε με συνέπεια ακριβείς λύσεις, ακολουθώντας την αναδρομική διαδικασία του dynamic programming. Το δίκτυο μπορούσε να αναπαράγει την βέλτιστη λύση σε όλα τα δοκιμασμένα παραδείγματα.
* **Λόγος** **Προσέγγισης(Approximation Ratio)**: Το FPTAS-NN παρήγαγε προσεγγιστικές λύσεις που ήταν εντός εγγυημένων ορίων της βέλτιστης λύσης. Ο λόγος προσέγγισης ήταν όπου είναι το πλάτος του δικτύου, γεγονός που έκανε το FPTAS-NN κατάλληλο για μεγαλύτερα σύνολα δεδομένων.
* **Κλιμάκωση(Scalability)**: Το FPTAS-NN αποδείχθηκε κλιμακούμενο, με σταθερή απόδοση καθώς αυξανόταν ο αριθμός των αντικειμένων, αποδεικνύοντας την ικανότητά του να γενικεύεται σε διαφορετικά μεγέθη προβλημάτων.

**Δεδομένα που Χρησιμοποιήθηκαν**

Τα δεδομένα που χρησιμοποιήθηκαν για την εκπαίδευση και την αξιολόγηση περιλάμβαναν συνθετικές εκδοχές του Προβλήματος του Σακιδίου:

* **Αντικείμενα**: Ο αριθμός των αντικειμένων κυμάνθηκε από 1,000 έως 10,000 ανά περίπτωση, με τυχαία κατανεμημένες τιμές κέρδους και μεγέθους.
* **Χωρητικότητα** **Σακιδίου**: Η χωρητικότητα είχε οριστεί στο 1 για λόγους απλοποίησης.

Η ευρεία γκάμα μεγεθών προβλημάτων επέτρεψε στους ερευνητές να δοκιμάσουν την ικανότητα κλιμάκωσης και τη στιβαρότητα των RNN μοντέλων σε διάφορα δεδομένα.

**Συμπεράσματα Έρευνας**

Η έρευνα έδειξε ότι τα RNNs, όταν σχεδιάζονται σωστά, μπορούν να επιλύσουν αποδοτικά το Πρόβλημα του Σακιδίου μέσω της προσομοίωσης του dynamic programming. Το DP-NN παρέχει ακριβείς λύσεις, αν και απαιτεί μεγαλύτερο δίκτυο για προβλήματα με υψηλή αξία. Το FPTAS-NN προσφέρει μια πιο αποδοτική λύση, εξισορροπώντας την υπολογιστική απόδοση με την ακρίβεια, προσφέροντας σχεδόν βέλτιστες λύσεις με εγγυημένο λόγο προσέγγισης.

Η μελέτη αναδεικνύει τη δυνατότητα των neural networks στην επίλυση συνδυαστικών προβλημάτων βελτιστοποίησης, ιδιαίτερα για NP-hard προβλήματα όπως το Πρόβλημα του Σακιδίου.

1. **Generative Adversarial Networks (GANs)**

Στο [33], οι ερευνητές εισάγουν μια καινοτόμο προσέγγιση που συνδυάζει ένα **Generative Adversarial Network (GAN)** με έναν **Γενετικό Αλγόριθμο (GA)** για την επίλυση του **3D Bin-Packing Probelm(3D-BPP)**. Το 3D-BPP είναι ένα γνωστό NP-hard πρόβλημα συνδυαστικής βελτιστοποίησης, όπου ο στόχος είναι να τοποθετηθεί ένα σύνολο τρισδιάστατων αντικειμένων σε κιβώτια με συγκεκριμένους περιορισμούς όγκου και διαστάσεων, χρησιμοποιώντας τον ελάχιστο αριθμό κιβωτίων. Αυτή η έρευνα εξετάζει πώς ο συνδυασμός του GAN με τον GA ενισχύει τη διασπορά και την ποιότητα των λύσεων, βελτιώνοντας την απόδοση της τοποθέτησης, ενώ διατηρεί ανταγωνιστικούς χρόνους υπολογισμού.

Αυτή η έρευνα δεν αντιμετωπίζει το κλασικό πρόβλημα σακιδίου, αλλά επικεντρώνεται στο **3D Bin-Packing Problem (3D-BPP)**, το οποίο έχει ομοιότητες με πολυδιάστατες παραλλαγές του προβλήματος σακιδίου (Multi-dimensional Knapsack Problem). Στο 3D-BPP, ο στόχος είναι να ελαχιστοποιηθεί ο αριθμός των κιβωτίων που χρησιμοποιούνται για την τοποθέτηση ενός δεδομένου συνόλου αντικειμένων, όπου κάθε αντικείμενο έχει συγκεκριμένο πλάτος, ύψος και βάθος, και κάθε κιβώτιο έχει συγκεκριμένη χωρητικότητα. Οι περιορισμοί περιλαμβάνουν ότι κανένα αντικείμενο δεν τοποθετείται περισσότερες από μία φορές και ότι ο συνολικός όγκος των αντικειμένων δεν ξεπερνά τη χωρητικότητα του κιβωτίου.

**Αρχιτεκτονική του Μοντέλου GAN**

Η αρχιτεκτονική του GAN που χρησιμοποιείται σε αυτή την εργασία αποτελείται από δύο βασικά στοιχεία:

**Generator:**

* Το **Generator (G)** δίκτυο λαμβάνει ένα τυχαίο **noise vector** ως είσοδο και παράγει συνθετικές λύσεις τοποθέτησης. Αυτές οι λύσεις αναπαριστούν πιθανές διατάξεις των αντικειμένων στα κιβώτια.
* Το generator αποτελείται από πλήρως συνδεδεμένα επίπεδα με ReLU ως συνάρτηση ενεργοποίησης στα κρυφά επίπεδα. Το τελικό επίπεδο εξόδου χρησιμοποιεί τη συνάρτηση ενεργοποίησης sigmoid για να εξασφαλίσει τιμές μεταξύ 0 και 1, που αναπαριστούν αν ένα αντικείμενο έχει τοποθετηθεί σε κιβώτιο ή όχι.

**Discriminator**:

* Το **Discriminator (D)** δίκτυο αξιολογεί τις λύσεις τοποθέτησης που παράγονται από το generator και προσπαθεί να διακρίνει αν είναι πραγματικές ή συνθετικές λύσεις.
* Το discriminator χρησιμοποιεί επίσης πλήρως συνδεδεμένα επίπεδα με Leaky ReLU ως συνάρτηση ενεργοποίησης στα κρυφά επίπεδα. Η έξοδος του είναι μια δυαδική ταξινόμηση (πραγματική ή ψεύτικη) και η εκπαίδευση πραγματοποιείται χρησιμοποιώντας τη συνάρτηση απώλειας binary cross-entropy.

Το GAN εκπαιδεύεται με μια ανταγωνιστική διαδικασία, όπου το generator προσπαθεί να δημιουργήσει ρεαλιστικές λύσεις τοποθέτησης, ενώ το discriminator βελτιώνεται ώστε να διακρίνει τις πραγματικές από τις συνθετικές λύσεις. Αυτή η αλληλεπίδραση βελτιώνει την ικανότητα του generator να παράγει υψηλής ποιότητας λύσεις τοποθέτησης με την πάροδο του χρόνου.

**Διαδικασία Εκπαίδευσης**

Η διαδικασία εκπαίδευσης για τον γενετικό αλγόριθμο που βασίζεται στο GAN περιλαμβάνει δύο κύριες φάσεις:

* **Εκπαίδευση του Generator**: Το generator εκπαιδεύεται να παράγει ρεαλιστικές λύσεις τοποθέτησης μαθαίνοντας από τα σχόλια που παρέχει το discriminator. Ο στόχος είναι να βελτιωθεί η ικανότητα του generator να μιμείται τις πραγματικές λύσεις τοποθέτησης.
* **Εκπαίδευση του Discriminator**: Το discriminator εκπαιδεύεται να διακρίνει τις πραγματικές από τις συνθετικές λύσεις τοποθέτησης, ελαχιστοποιώντας τη συνάρτηση απώλειας binary cross-entropy. Καθώς το discriminator γίνεται πιο ικανό, το generator αναγκάζεται να παράγει πιο ρεαλιστικές λύσεις.

Σε κάθε επανάληψη, και τα δύο δίκτυα ενημερώνονται χρησιμοποιώντας πραγματικά και συνθετικά δείγματα δεδομένων. Τα πραγματικά δείγματα προέρχονται από έναν πληθυσμό τοποθετήσεων αντικειμένων, ενώ τα συνθετικά δεδομένα δημιουργούνται από το generator χρησιμοποιώντας τυχαία noise vectors.

Το στοιχείο του GAN ενσωματώνεται στον γενετικό αλγόριθμο (GA) μέσω δύο μηχανισμών:

* **Εισαγωγή Διαφορετικότητας (Diversity Injection)**: Μετά από μερικές γενιές, το GAN παράγει συνθετικές λύσεις που εισάγονται στον πληθυσμό για τη βελτίωση της διαφορετικότητας.
* **Αύξηση Καταλληλότητας(Fitness Augmentation)**: Η έξοδος του discriminator χρησιμοποιείται για να υπολογιστεί μια βοηθητική συνάρτηση καταλληλότητας, η οποία καθοδηγεί τη διαδικασία επιλογής του γενετικού αλγόριθμου.

**Αξιολόγηση**

Η αξιολόγηση του προτεινόμενου γενετικού αλγόριθμου που βασίζεται στο GAN πραγματοποιήθηκε μέσω προσομοιώσεων σε πρότυπα παραδείγματα του προβλήματος 3D-BPP. Η απόδοση συγκρίθηκε με τον παραδοσιακό GA και τον Particle Swarm Optimization (PSO). Η αξιολόγηση επικεντρώθηκε σε δύο κύριες μετρικές:

**Αριθμός Κιβωτίων που Χρησιμοποιήθηκαν**: Αυτός είναι ο κύριος στόχος του 3D-BPP, που αντιπροσωπεύει την αποδοτικότητα του αλγορίθμου στην τοποθέτηση αντικειμένων.

**Χρόνος Υπολογισμού**: Αυτό μετρά την ταχύτητα και την αποδοτικότητα του αλγορίθμου στην εύρεση λύσεων.

Τα πειράματα περιλάμβαναν δοκιμές σε διάφορα παραδείγματα, με αριθμό αντικειμένων από 10 έως 100 και αριθμό κιβωτίων από 3 έως 20. Ο γενετικός αλγόριθμος με βάση το GAN ξεπέρασε σε απόδοση τόσο τον GA όσο και τον PSO όσον αφορά τον αριθμό των χρησιμοποιούμενων κιβωτίων, επιτυγχάνοντας κατά μέσο όρο βελτίωση 9.1% σε σχέση με τον GA και 5.8% σε σχέση με τον PSO. Οι χρόνοι υπολογισμού ήταν ανταγωνιστικοί, με τον GAN-based GA να επιτυγχάνει αντίστοιχους χρόνους με τους GA και PSO.

**Δεδομένα που Χρησιμοποιήθηκαν**

Το σύνολο δεδομένων που χρησιμοποιήθηκε για τις προσομοιώσεις περιλάμβανε πρότυπα παραδείγματα του προβλήματος 3D-BPP. Ο αριθμός των αντικειμένων κυμαινόταν από 10 έως 100, και ο αριθμός των κιβωτίων κυμαινόταν από 3 έως 20. Κάθε αντικείμενο είχε τυχαία κατανεμημένες διαστάσεις, και τα κιβώτια είχαν σταθερές διαστάσεις με περιορισμούς όγκου. Το σύνολο δεδομένων σχεδιάστηκε ώστε να είναι ρεαλιστικό και χρησιμοποιήθηκε σε όλους τους αλγόριθμους που συγκρίθηκαν, για να εξασφαλιστεί η δίκαιη σύγκριση των αποτελεσμάτων.

**Συμπέρασμα**

Ο γενετικός αλγόριθμος που βασίζεται στο GAN, όπως προτάθηκε σε αυτή την έρευνα, αποδεικνύει μια νέα προσέγγιση για την επίλυση του προβλήματος 3D-BPP. Με την ενσωμάτωση των GAN στους GA, ο αλγόριθμος βελτιώνει τόσο τη διασπορά όσο και την ποιότητα των παραγόμενων λύσεων. Τα αποτελέσματα των προσομοιώσεων έδειξαν ότι ο γενετικός αλγόριθμος που βασίζεται στο GAN ξεπερνά τον παραδοσιακό GA και τον PSO όσον αφορά τον αριθμό των κιβωτίων που χρησιμοποιούνται, ενώ διατηρεί ανταγωνιστικούς χρόνους υπολογισμού.

Το στοιχείο του GAN παίζει κρίσιμο ρόλο στην ενίσχυση των ικανοτήτων εξερεύνησης του GA, βοηθώντας στην αποφυγή τοπικών βέλτιστων λύσεων και στην προώθηση καλύτερων συνολικών λύσεων. Αυτή η έρευνα ανοίγει νέες δυνατότητες για τη χρήση των GAN σε άλλα προβλήματα βελτιστοποίησης και υπογραμμίζει τις δυνατότητες για περαιτέρω βελτιώσεις στις τεχνικές συνδυαστικής βελτιστοποίησης.

## Συμπεράσματα Κεφαλαίου

Ανακεφαλαιώνοντας, η συγχώνευση τεχνικών βαθιάς μάθησης με το πρόβλημα Knapsack συμβάλλει στην αντιμετώπιση των προκλήσεων που θέτει αυτό το κλασικό πρόβλημα συνδυαστικής βελτιστοποίησης. Η βιβλιογραφία που εξετάστηκε καταδεικνύει την ευελιξία των βαθιών νευρωνικών δικτύων στην αντιμετώπιση της πολυπλοκότητας του προβλήματος Knapsack, προσφέροντας λύσεις που υπερβαίνουν τις παραδοσιακές αλγοριθμικές προσεγγίσεις.

Οι ερευνητές έχουν διερευνήσει διάφορες στρατηγικές κωδικοποίησης, αρχιτεκτονικές δικτύων και μεθοδολογίες εκπαίδευσης για να αξιοποιήσουν τη δύναμη των deep learning μοντέλων για την επίλυση περιπτώσεων του προβλήματος Knapsack. Η προσαρμοστικότητα των νευρωνικών δικτύων επιτρέπει την ενσωμάτωση κωδικοποιήσεων πραγματικών τιμών και δυαδικών κωδικοποιήσεων, ενώ αρχιτεκτονικές που κυμαίνονται από δίκτυα τροφοδότησης έως πιο σύνθετα αναδρομικά νευρωνικά δίκτυα και εξελικτικούς αλγορίθμους αναδεικνύουν την ευελιξία αυτών των μοντέλων. Αυτά τα ολοκληρωμένα μοντέλα στοχεύουν στην αξιοποίηση των πλεονεκτημάτων και των δύο παραδειγμάτων, επιτυγχάνοντας βελτιωμένη ποιότητα λύσεων, ταχύτερη σύγκλιση και αυξημένη κλιμάκωση.

Η εφαρμογή της βαθιάς μάθησης στο πρόβλημα Knapsack επεκτείνεται πέρα από θεωρητικές εκτιμήσεις, βρίσκοντας πρακτική χρησιμότητα σε τομείς όπως η εφοδιαστική αλυσίδα, η χρηματοδότηση και η κατανομή πόρων. Η ικανότητα των deep learning να μαθαίνουν περίπλοκα μοτίβα και σχέσεις μέσα στα δεδομένα τους επιτρέπει να προσαρμόζονται στις πολυπλοκότητες του πραγματικού κόσμου, παρέχοντας στους υπεύθυνους λήψης αποφάσεων ισχυρά εργαλεία για τη βελτιστοποίηση της κατανομής των πόρων και την ενίσχυση της επιχειρησιακής αποδοτικότητας.

Παρά τα πολλά υποσχόμενα βήματα που έχουν γίνει σε αυτή τη διασταύρωση της βαθιάς μάθησης και του προβλήματος του σακιδίου, οι προκλήσεις εξακολουθούν να υφίστανται. Η γενίκευση για τον χειρισμό μεγαλύτερων περιπτώσεων και η διερεύνηση νέων αρχιτεκτονικών νευρωνικών δικτύων αποτελούν τομείς για συνεχή διερεύνηση. Καθώς η τεχνολογία εξελίσσεται, η μελλοντική έρευνα μπορεί να εμβαθύνει σε υβριδικά μοντέλα που ενσωματώνουν απρόσκοπτα τη βαθιά μάθηση με αναδυόμενες τεχνολογίες, εξασφαλίζοντας τη συνεχή συνάφεια αυτών των προσεγγίσεων στην αντιμετώπιση σύνθετων σεναρίων λήψης αποφάσεων.

# Μεθοδολογία

## Εισαγωγή

Για την ευκολότερη σύγκριση και κατανόηση των μοντέλων deep learning, έχει επιλεχθεί να προσομοιωθεί το πρόβλημα Knapsack στην απλούστερη μορφή του. Το πρόβλημα 0/1 Knapsack, ένα κλασικό πρόβλημα συνδυαστικής βελτιστοποίησης, περιλαμβάνει την επιλογή αντικειμένων με διακριτά βάρη και τιμές για τη μεγιστοποίηση της συνολικής αξίας, τηρώντας παράλληλα έναν προκαθορισμένο περιορισμό βάρους. Σε αυτή τη διατύπωση του προβλήματος, κάθε στοιχείο μπορεί να επιλεγεί μόνο μία φορά, που αντιπροσωπεύεται από τη δυαδική απόφαση για το αν θα συμπεριληφθεί ή θα εξαιρεθεί από το σακίδιο. Η πρόκληση έγκειται στην εύρεση του βέλτιστου υποσυνόλου αντικειμένων που αποδίδει την υψηλότερη αξία χωρίς να υπερβαίνει τη χωρητικότητα βάρους του σακιδίου. Αυτό το πρόβλημα παρουσιάζει σημαντικά υπολογιστικά εμπόδια, ειδικά για μεγάλες περιπτώσεις προβλήματος, γεγονός που καθιστά αναγκαία την ανάπτυξη αποτελεσματικών αλγορίθμων για την εύρεση εφικτών λύσεων σε λογικά χρονικά πλαίσια.

Ο πρωταρχικός στόχος του παρόντος έργου είναι η ανάπτυξη ενός λύτη του προβλήματος Knapsack 0/1 που χρησιμοποιεί τρία μοντέλα deep learning διαφορετικού τύπου: Deep Neural Network, Recurrent Neural Network και Deep Q-Learning Network. Αυτά τα μοντέλα προσφέρουν διαφορετικές προσεγγίσεις για την επίλυση του προβλήματος Knapsack, αξιοποιώντας τη δύναμη της μηχανικής μάθησης και των τεχνικών βελτιστοποίησης για τη δημιουργία σχεδόν βέλτιστων λύσεων. Συγκρίνοντας και αντιπαραβάλλοντας τις επιδόσεις αυτών των μοντέλων, το έργο αυτό αποσκοπεί στην παροχή πληροφοριών σχετικά με την αποτελεσματικότητα και την καταλληλότητά τους για την αντιμετώπιση του προβλήματος Knapsack σε διάφορα πλαίσια.

Σε αυτό το κεφάλαιο, παρουσιάζεται η μεθοδολογία που χρησιμοποιήθηκε για την επίλυση του προβλήματος Knapsack 0/1 με τη χρήση διαφόρων μοντέλων βαθιάς μάθησης και ενισχυτικής μάθησης. Αρχικά, περιγράφονται τα επιλεγμένα μοντέλα, ακολουθούμενα από την αρχιτεκτονική και τη διαδικασία εκπαίδευσης και αξιολόγησής τους. Τέλος, παρουσιάζεται η διαδικασία προετοιμασίας των δεδομένων.

## Βαθύ Νευρωνικό Δίκτυο (Deep Neural Network, DNN)

## Επισκόπηση

Το Βαθύ Νευρωνικό Δίκτυο (Deep Neural Network, DNN) είναι ένα πολυστρωματικό νευρωνικό δίκτυο που χρησιμοποιείται ευρέως στη βαθιά μάθηση για την επίλυση προβλημάτων αναγνώρισης προτύπων, ταξινόμησης και παρόμοιων προβλημάτων βελτιστοποίησης. Στο πλαίσιο του προβλήματος του Knapsack 0/1, το DNN αξιοποιείται για να μάθει τις βέλτιστες στρατηγικές επιλογής αντικειμένων που μεγιστοποιούν την αξία του σακιδίου χωρίς να υπερβαίνουν τη χωρητικότητά του.

## Αρχιτεκτονική

Η αρχιτεκτονική ενός DNN για το πρόβλημα του σακιδίου 0/1 βασίζεται σε πλήρως συνδεδεμένα στρώματα. Σε κάθε επίπεδο (στρώμα), οι είσοδοι περνούν μέσα από μια σειρά γραμμικών και μη γραμμικών μετασχηματισμών, επιτρέποντας στο δίκτυο να μάθει πολύπλοκες σχέσεις μεταξύ των δεδομένων.

* **Εισαγωγή Δεδομένων:** Η είσοδος του δικτύου αποτελείται από τα χαρακτηριστικά των αντικειμένων, όπως τα βάρη και οι αξίες τους, μαζί με τη συνολική χωρητικότητα του σακιδίου. Τα δεδομένα αυτά αποτελούν το αρχικό επίπεδο εισόδου που τροφοδοτείται στο δίκτυο.
* **Πλήρως Συνδεδεμένα Στρώματα (Hidden Layers):** Τα πλήρως συνδεδεμένα στρώματα αποτελούνται από νευρώνες που συνδέονται με όλους τους νευρώνες του προηγούμενου στρώματος. Κάθε νευρώνας εφαρμόζει έναν γραμμικό συνδυασμό των εισόδων του, ακολουθούμενο από μια μη γραμμική συνάρτηση ενεργοποίησης, όπως η ReLU (Rectified Linear Unit). Αυτή η προσέγγιση επιτρέπει στο δίκτυο να μάθει και να αναπαραστήσει περίπλοκες σχέσεις που υπάρχουν στα δεδομένα.
* **Στρώμα Εξόδου:** Το τελικό στρώμα του δικτύου παράγει μια δυαδική απόφαση για κάθε αντικείμενο, υποδεικνύοντας εάν το αντικείμενο θα πρέπει να συμπεριληφθεί στο σακίδιο ή όχι. Συνήθως χρησιμοποιείται μια συνάρτηση ενεργοποίησης όπως η Sigmoid, η οποία εξάγει τιμές μεταξύ 0 και 1, καθιστώντας εύκολη τη μετατροπή σε δυαδική απόφαση.

## Διαδικασία Εκπαίδευσης

Η εκπαίδευση ενός DNN για το πρόβλημα του σακιδίου 0/1 περιλαμβάνει τη βελτιστοποίηση των παραμέτρων του μέσω επαναληπτικής εκμάθησης από τα δεδομένα. Ο στόχος είναι η ελαχιστοποίηση της διαφοράς μεταξύ των προβλεπόμενων αποφάσεων του δικτύου και των βέλτιστων αποφάσεων, όπως αυτές καθορίζονται από την πραγματική λύση του προβλήματος.

* **Απώλεια (Loss Function):** Η εκπαίδευση βασίζεται σε μια συνάρτηση απώλειας, η οποία μετράει το σφάλμα μεταξύ της πρόβλεψης του μοντέλου και της πραγματικής ετικέτας. Αυτή η απώλεια καθοδηγεί την ενημέρωση των βαρών του δικτύου μέσω οπισθοδιάδοσης (backpropagation).
* **Βελτιστοποίηση:** Κατά τη διάρκεια της εκπαίδευσης, ένας αλγόριθμος βελτιστοποίησης, όπως ο Stochastic Gradient Descent (SGD), χρησιμοποιείται για την ενημέρωση των βαρών του δικτύου. Ο βελτιστοποιητής αυτός εφαρμόζει τις απαραίτητες αλλαγές στα βάρη ώστε να μειωθεί η συνάρτηση απώλειας, βελτιώνοντας έτσι την ικανότητα του δικτύου να κάνει σωστές προβλέψεις.
* **Επανάληψη και Αξιολόγηση:** Η διαδικασία εκπαίδευσης επαναλαμβάνεται για έναν καθορισμένο αριθμό εποχών (epochs), επιτρέποντας στο δίκτυο να βελτιώνει σταδιακά τις προβλέψεις του. Μετά την εκπαίδευση, το μοντέλο αξιολογείται σε ένα επαληθευτικό σύνολο δεδομένων για να διασφαλιστεί ότι μπορεί να γενικεύσει τις αποφάσεις του πέρα από τα δεδομένα εκπαίδευσης.

Η εκπαίδευση του DNN απαιτεί προσεκτική παρακολούθηση για να διασφαλιστεί ότι το μοντέλο δεν υπερπροσαρμόζεται στα δεδομένα εκπαίδευσης (overfitting), ενώ ταυτόχρονα μαθαίνει να λαμβάνει βέλτιστες αποφάσεις για το πρόβλημα του σακιδίου 0/1.

## Επαναλαμβανόμενο Νευρωνικό Δίκτυο (Recurrent Neural Network, RNN)

## Επισκόπηση

Το Επαναλαμβανόμενο Νευρωνικό Δίκτυο (Recurrent Neural Network, RNN) είναι μια κατηγορία νευρωνικών δικτύων που είναι ιδιαίτερα αποτελεσματική στην επεξεργασία ακολουθιακών δεδομένων. Σε αντίθεση με τα τυπικά νευρωνικά δίκτυα, το RNN έχει τη δυνατότητα να "θυμάται" προηγούμενες καταστάσεις και να τις χρησιμοποιεί για να επηρεάσει τις τρέχουσες αποφάσεις. Αυτή η ικανότητα καθιστά το RNN κατάλληλο για προβλήματα όπου οι αποφάσεις πρέπει να ληφθούν σε μια ακολουθιακή βάση, όπως το πρόβλημα του Knapsack 0/1, όπου η επιλογή ενός αντικειμένου μπορεί να επηρεάσει τις επόμενες επιλογές.

## Αρχιτεκτονική

Η αρχιτεκτονική ενός RNN για το πρόβλημα του σακιδίου 0/1 περιλαμβάνει αρκετά βασικά στοιχεία που το διαφοροποιούν από άλλα νευρωνικά δίκτυα:

* **Είσοδος:** Το RNN λαμβάνει ως είσοδο μια ακολουθία δεδομένων που αντιπροσωπεύει τα αντικείμενα που εξετάζονται για επιλογή. Κάθε στοιχείο της ακολουθίας περιλαμβάνει χαρακτηριστικά όπως το βάρος και την αξία του αντικειμένου, καθώς και την τρέχουσα χωρητικότητα του σακιδίου.
* **Επαναλαμβανόμενα Στρώματα (Recurrent Layers):** Το κεντρικό χαρακτηριστικό του RNN είναι τα επαναλαμβανόμενα στρώματα, τα οποία διατηρούν μια εσωτερική κατάσταση (ή μνήμη) που ενημερώνεται καθώς η ακολουθία δεδομένων προχωρά. Σε κάθε βήμα της ακολουθίας, το RNN λαμβάνει υπόψη τόσο την τρέχουσα είσοδο όσο και την κατάσταση της μνήμης του από το προηγούμενο βήμα, επιτρέποντας στο δίκτυο να συλλαμβάνει τις εξαρτήσεις μεταξύ των δεδομένων.
* **LSTM και Dropout:** Συχνά, ένα RNN ενισχύεται με την προσθήκη Long Short-Term Memory (LSTM) στρωμάτων, τα οποία είναι ειδικά σχεδιασμένα για να διατηρούν πληροφορίες για μεγαλύτερες ακολουθίες και να αποφεύγουν τα προβλήματα της "εξαφάνισης" ή "έκρηξης" των συναρτήσεων ενεργοποίησης (vanishing/exploding gradients). Επιπλέον, ένα στρώμα Dropout μπορεί να χρησιμοποιηθεί για να μειώσει την υπερπροσαρμογή, εισάγοντας τυχαία διακοπή συνδέσεων κατά τη διάρκεια της εκπαίδευσης.
* **Στρώμα Εξόδου:** Το τελικό στρώμα του δικτύου χρησιμοποιεί μια συνάρτηση ενεργοποίησης (όπως η sigmoid) για να παράγει δυαδικές αποφάσεις σχετικά με το αν κάθε αντικείμενο θα πρέπει να συμπεριληφθεί στο σακίδιο, λαμβάνοντας υπόψη την ακολουθία των προηγούμενων αποφάσεων.

## Διαδικασία Εκπαίδευσης

Η εκπαίδευση ενός RNN για το πρόβλημα του σακιδίου 0/1 ακολουθεί μια διαδικασία που στοχεύει στη βελτιστοποίηση των παραμέτρων του, ώστε να μπορεί να λάβει τις βέλτιστες αποφάσεις με βάση τις ακολουθίες δεδομένων.

* **Απώλεια (Loss Function):** Η συνάρτηση απώλειας που χρησιμοποιείται λαμβάνει υπόψη τις ακολουθίες των αποφάσεων και μετράει τη διαφορά μεταξύ των προβλεπόμενων και των πραγματικών αποφάσεων σε όλη την ακολουθία. Αυτή η συνάρτηση καθοδηγεί την ενημέρωση των παραμέτρων μέσω της οπισθοδιάδοσης σε βάθος χρόνου (Backpropagation Through Time, BPTT).
* **Βελτιστοποίηση:** Ένας αλγόριθμος βελτιστοποίησης εφαρμόζεται για την ενημέρωση των παραμέτρων του δικτύου. Δεδομένου ότι το RNN επεξεργάζεται ακολουθίες, ο βελτιστοποιητής πρέπει να λαμβάνει υπόψη την αλληλεξάρτηση των διαδοχικών βημάτων κατά την εκμάθηση.
* **Επανάληψη και Αξιολόγηση:** Η εκπαίδευση του RNN πραγματοποιείται μέσω επαναλαμβανόμενων εποχών, όπου το δίκτυο βελτιώνει τις προβλέψεις του για τις ακολουθίες δεδομένων. Η αξιολόγηση του μοντέλου γίνεται με βάση την ικανότητά του να γενικεύει τις αποφάσεις του σε νέα, άγνωστα δεδομένα, αποφεύγοντας ταυτόχρονα την υπερπροσαρμογή.

Το RNN είναι ιδιαίτερα κατάλληλο για το πρόβλημα του σακιδίου 0/1, καθώς μπορεί να διαχειριστεί την αλληλεξάρτηση των επιλογών σε μια ακολουθία, εξασφαλίζοντας ότι οι αποφάσεις που λαμβάνονται είναι βέλτιστες όχι μόνο σε τοπικό, αλλά και σε συνολικό επίπεδο.

## Μοντέλο Deep Q-Learning (DQN)

## Επισκόπηση

Το Μοντέλο Deep Q-Learning (DQN) είναι μια μέθοδος ενισχυτικής μάθησης που συνδυάζει τις αρχές της βαθιάς μάθησης με την ενισχυτική μάθηση (Reinforcement Learning). Στο DQN, ένας πράκτορας μαθαίνει να λαμβάνει ακολουθιακές αποφάσεις σε ένα δυναμικό περιβάλλον, με στόχο τη μεγιστοποίηση μιας συνολικής αμοιβής. Στο πλαίσιο του προβλήματος του Knapsack 0/1, το DQN χρησιμοποιείται για να μάθει τη βέλτιστη στρατηγική επιλογής αντικειμένων, έτσι ώστε να μεγιστοποιηθεί η αξία του σακιδίου χωρίς να υπερβαίνει τη διαθέσιμη χωρητικότητα.

## Αρχιτεκτονική

Η αρχιτεκτονική του DQN περιλαμβάνει τα παρακάτω βασικά στοιχεία:

* **Είσοδος:** Το δίκτυο λαμβάνει ως είσοδο την τρέχουσα κατάσταση του σακιδίου και τα χαρακτηριστικά του αντικειμένου που πρόκειται να αξιολογηθεί. Αυτά τα χαρακτηριστικά περιλαμβάνουν την τρέχουσα συνολική αξία του σακιδίου, την υπολειπόμενη χωρητικότητα, την αξία και το βάρος του εξεταζόμενου αντικειμένου.
* **Πλήρως Συνδεδεμένα Στρώματα (Hidden Layers):** Το δίκτυο αποτελείται από μια σειρά από πλήρως συνδεδεμένα στρώματα, τα οποία είναι υπεύθυνα για την επεξεργασία της εισόδου και την παραγωγή της εκτίμησης της Q-συνάρτησης για κάθε πιθανή δράση. Σε κάθε στρώμα εφαρμόζονται γραμμικοί μετασχηματισμοί, ακολουθούμενοι από μη γραμμικές συναρτήσεις ενεργοποίησης, όπως η ReLU, για να διευκολυνθεί η μάθηση σύνθετων σχέσεων μεταξύ των εισόδων.
* **Q-Συνάρτηση (Q-Function):** Η Q-συνάρτηση είναι το κεντρικό στοιχείο του DQN και αντιπροσωπεύει την αναμενόμενη αμοιβή που μπορεί να λάβει ο πράκτορας για κάθε πιθανή δράση (επιλογή ή απόρριψη ενός αντικειμένου) σε μια συγκεκριμένη κατάσταση. Η εκτίμηση της Q-συνάρτησης οδηγεί τη διαδικασία λήψης αποφάσεων του πράκτορα.
* **Στρώμα Εξόδου:** Το τελικό στρώμα του δικτύου αποδίδει τις εκτιμήσεις της Q-συνάρτησης για όλες τις πιθανές δράσεις. Η δράση με την υψηλότερη εκτίμηση επιλέγεται ως η επόμενη κίνηση του πράκτορα.
* **Μνήμη Replay (Replay Memory):** Για τη σταθεροποίηση της διαδικασίας μάθησης, το DQN χρησιμοποιεί μια μνήμη επαναλήψεων, όπου αποθηκεύονται οι εμπειρίες του πράκτορα (δηλαδή, καταστάσεις, δράσεις, αμοιβές και επόμενες καταστάσεις). Αυτές οι εμπειρίες ανακαλούνται τυχαία κατά την εκπαίδευση για να σπάσει η αλληλουχία των εμπειριών, διευκολύνοντας τη γενίκευση.

## Διαδικασία Εκπαίδευσης

Η εκπαίδευση του DQN περιλαμβάνει τη βελτιστοποίηση της Q-συνάρτησης μέσω επαναληπτικής μάθησης από τις εμπειρίες του πράκτορα.

* **Εμπειρία και Μάθηση (Experience Replay):** Κατά την εκπαίδευση, οι εμπειρίες του πράκτορα αποθηκεύονται στη μνήμη επαναλήψεων και επαναχρησιμοποιούνται για την εκπαίδευση του δικτύου. Αυτό βοηθά στη μείωση της συσχέτισης μεταξύ των διαδοχικών εμπειριών και βελτιώνει τη σταθερότητα της μάθησης.
* **Επιλογή Δράσης (Action Selection):** Η επιλογή δράσης βασίζεται σε μια ισορροπία μεταξύ εξερεύνησης (exploration) και εκμετάλλευσης (exploitation). Στην αρχή της εκπαίδευσης, ο πράκτορας τείνει να εξερευνά περισσότερο, επιλέγοντας τυχαίες δράσεις, ενώ όσο προχωρά η εκπαίδευση, βασίζεται περισσότερο στις εκτιμήσεις της Q-συνάρτησης.
* **Αλγόριθμος Q-Learning:** Ο πυρήνας της εκπαίδευσης του DQN είναι ο αλγόριθμος Q-learning, ο οποίος ενημερώνει την Q-συνάρτηση με βάση την αναμενόμενη αμοιβή που λαμβάνει ο πράκτορας από τις δράσεις του. Ο στόχος του Q-learning είναι να προσαρμόσει τις εκτιμήσεις της Q-συνάρτησης έτσι ώστε να αντικατοπτρίζουν με μεγαλύτερη ακρίβεια τις πραγματικές μακροπρόθεσμες αμοιβές.
* **Σύγκλιση και Αξιολόγηση:** Κατά τη διάρκεια της εκπαίδευσης, το DQN προσαρμόζει συνεχώς τις εκτιμήσεις της Q-συνάρτησης, με στόχο τη σύγκλιση σε μια βέλτιστη στρατηγική λήψης αποφάσεων. Η απόδοση του μοντέλου αξιολογείται μέσω της ικανότητάς του να μεγιστοποιεί τη συνολική αμοιβή σε νέα δεδομένα.

Το DQN είναι ένα ιδιαίτερα ισχυρό εργαλείο για την επίλυση σύνθετων προβλημάτων που απαιτούν αλληλεξαρτώμενες αποφάσεις, όπως το πρόβλημα του σακιδίου 0/1. Μέσω της ενσωμάτωσης της βαθιάς μάθησης με την ενισχυτική μάθηση, το DQN καταφέρνει να μάθει στρατηγικές που μεγιστοποιούν τη μακροπρόθεσμη αμοιβή του πράκτορα.

## Σύνολο Δεδομένων και Προεπεξεργασία

Η ποιότητα των δεδομένων είναι καθοριστικής σημασίας για την επιτυχία της εκπαίδευσης των νευρωνικών δικτύων. Σε αυτή την ενότητα, περιγράφεται η διαδικασία δημιουργίας και προετοιμασίας του συνόλου δεδομένων που χρησιμοποιήθηκε για την εκπαίδευση των μοντέλων DNN, RNN και DQN για το πρόβλημα του σακιδίου 0/1.

## Δημιουργία Συνόλου Δεδομένων

Δεδομένου ότι το πρόβλημα του σακιδίου 0/1 είναι ένα κλασικό πρόβλημα βελτιστοποίησης, τα δεδομένα που χρησιμοποιούνται για την εκπαίδευση των μοντέλων πρέπει να αντικατοπτρίζουν τα πραγματικά χαρακτηριστικά του προβλήματος. Συγκεκριμένα, κάθε δείγμα δεδομένων περιλαμβάνει έναν αριθμό αντικειμένων, το καθένα με ένα συγκεκριμένο βάρος και μια αξία.

Για την παραγωγή ενός τέτοιου συνόλου δεδομένων, χρησιμοποιήθηκε μια διαδικασία τυχαίας γεννήτριας δεδομένων. Αυτή η διαδικασία δημιουργεί ένα προκαθορισμένο πλήθος δειγμάτων, όπου κάθε δείγμα περιέχει έναν συγκεκριμένο αριθμό αντικειμένων, καθένα από τα οποία έχει αποδοθεί τυχαία ένα βάρος και μια αξία. Η γεννήτρια αυτή εξασφαλίζει ποικιλία στα δεδομένα, καθιστώντας τα μοντέλα ικανά να γενικεύσουν καλύτερα και να αποδίδουν σε διαφορετικά σενάρια.

## Προεπεξεργασία Δεδομένων

Πριν τα δεδομένα εισαχθούν στα μοντέλα, είναι απαραίτητο να υποστούν κατάλληλη προεπεξεργασία, ώστε να είναι συμβατά με τις απαιτήσεις των νευρωνικών δικτύων.

**Κανονικοποίηση**

Μια σημαντική διαδικασία στην προεπεξεργασία είναι η κανονικοποίηση των δεδομένων. Τα βάρη και οι αξίες των αντικειμένων μπορεί να έχουν διαφορετικές κλίμακες, γεγονός που μπορεί να επηρεάσει την απόδοση του μοντέλου. Η κανονικοποίηση εξασφαλίζει ότι όλα τα δεδομένα έχουν την ίδια κλίμακα, διευκολύνοντας τη διαδικασία εκμάθησης.

**Διαχωρισμός σε Εκπαιδευτικό και Επαληθευτικό Σύνολο**

Τα δεδομένα διαχωρίστηκαν σε εκπαιδευτικό και επαληθευτικό σύνολο. Το εκπαιδευτικό σύνολο χρησιμοποιείται για την εκπαίδευση των μοντέλων, ενώ το επαληθευτικό σύνολο χρησιμοποιείται για την αξιολόγηση της απόδοσης του μοντέλου κατά τη διάρκεια της εκπαίδευσης. Αυτός ο διαχωρισμός βοηθά στον εντοπισμό του φαινομένου της υπερπροσαρμογής (overfitting), διασφαλίζοντας ότι το μοντέλο μαθαίνει γενικά πρότυπα και όχι μόνο τα χαρακτηριστικά των δεδομένων εκπαίδευσης.

**Μετασχηματισμός για Χρήση σε RNN και DQN**

Εκτός από την κανονική προεπεξεργασία, τα δεδομένα υποβάλλονται σε επιπλέον μετασχηματισμούς ώστε να είναι κατάλληλα για χρήση στα μοντέλα RNN και DQN. Στην περίπτωση του RNN, τα δεδομένα πρέπει να παρουσιαστούν ως ακολουθίες, λαμβάνοντας υπόψη τη χρονική σειρά των αποφάσεων. Στην περίπτωση του DQN, τα δεδομένα διαρθρώνονται με τέτοιο τρόπο ώστε να είναι συμβατά με τις ανάγκες του πράκτορα, ο οποίος μαθαίνει μέσω αλληλεπίδρασης με ένα περιβάλλον.

## Επίλογος Κεφαλαίου

Σε αυτό το κεφάλαιο, παρουσιάσαμε τρεις προσεγγίσεις βαθιάς μάθησης για την επίλυση του προβλήματος Knapsack 0/1: ένα Βαθύ Νευρωνικό Δίκτυο (DNN), το Επαναλαμβανόνεμο Νευρωνικό Δίκτυο (RNN), και το Μοντέλο Deep Q-Learning (DQN). Κάθε μοντέλο προσφέρει διαφορετικά πλεονεκτήματα, ανάλογα με τη φύση των δεδομένων και τις απαιτήσεις του προβλήματος.

Το DNN είναι κατάλληλο για προβλήματα με στατικά δεδομένα, το RNN για προβλήματα που απαιτούν κατανόηση ακολουθιακών σχέσεων, και το DQN για περιπτώσεις όπου η μάθηση μέσω αλληλεπίδρασης με ένα περιβάλλον είναι κρίσιμη.

Στα επόμενα κεφάλαια, θα αναλυθεί ο κώδικας που υλοποιεί αυτά τα μοντέλα και θα παρουσιαστούν τα αποτελέσματα των δοκιμών. Μέσα από αυτή την ανάλυση, θα αξιολογηθεί η απόδοση κάθε μοντέλου και η καταλληλότητά τους στην επίλυση του προβλήματος Knapsack 0/1.

# Εφαρμογή

## Εισαγωγή

Σε αυτό το κεφάλαιο, θα εξεταστεί σε βάθος η υλοποίηση των μοντέλων βαθιάς μάθησης που χρησιμοποιήθηκαν για την επίλυση του προβλήματος Knapsack 0/1. Θα αναλυθεί ο κώδικας που δημιουργεί και εκπαιδεύει τα νευρωνικά δίκτυα, περιγράφοντας κάθε βήμα της διαδικασίας με λεπτομέρεια.

Αρχικά, θα εξετασθεί η διαδικασία δημιουργίας και προετοιμασίας των δεδομένων που χρησιμοποιούνται για την εκπαίδευση των μοντέλων. Στη συνέχεια, θα παρουσιαστεί η αρχιτεκτονική και η λειτουργία του Βαθιού Νευρωνικού Δικτύου (DNN), μαζί με τη λογική πίσω από τη ροή των δεδομένων μέσα στο δίκτυο. Ακολουθεί η λεπτομερής ανάλυση της εκπαίδευσης του μοντέλου, η οποία περιλαμβάνει την προσαρμογή των παραμέτρων του δικτύου για τη βελτίωση της απόδοσής του.

Ακολούθως, θα αναλυθεί η υλοποίηση του Επαναλαμβανόμενου Νευρωνικού Δικτύου (RNN) και του Μοντέλου Deep Q-Learning (DQN), προσφέροντας μια λεπτομερή ματιά στην κωδικοποίηση των μηχανισμών μάθησης και λήψης αποφάσεων που ενσωματώνονται σε αυτά τα μοντέλα.

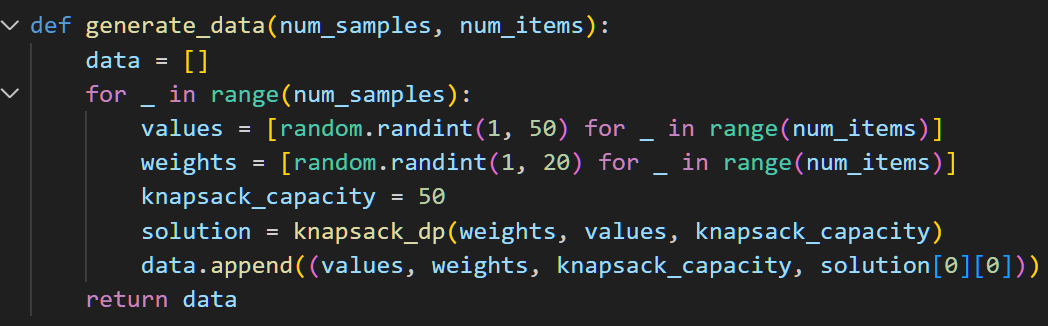
Η αναλυτική κατανόηση του κώδικα είναι κρίσιμη για την επιτυχή εφαρμογή και βελτιστοποίηση των μοντέλων. Αυτό το κεφάλαιο σκοπεύει να παρέχει σαφή και ακριβή εξήγηση του τρόπου με τον οποίο οι θεωρητικές έννοιες που συζητήθηκαν προηγουμένως υλοποιούνται στην πράξη.

## Δημιουργία και Προετοιμασία Δεδομένων

Η προετοιμασία των δεδομένων αποτελεί το πρώτο και σημαντικότερο βήμα στην εκπαίδευση των μοντέλων βαθιάς μάθησης για την επίλυση του προβλήματος Knapsack 0/1. Σε αυτή την ενότητα, θα αναλυθεί η διαδικασία δημιουργίας των δεδομένων καθώς και ο υπολογισμός των βέλτιστων λύσεων που χρησιμοποιούνται ως στόχοι εκμάθησης για τα μοντέλα.

### Δημιουργία δεδομένων

Η δημιουργία των δεδομένων πραγματοποιείται με τη χρήση της συνάρτησης `**generate\_data**`, η οποία δημιουργεί πολλαπλά δείγματα με διαφορετικές τιμές και βάρη αντικειμένων. Η συνάρτηση αυτή παράγει τα δεδομένα που θα χρησιμοποιηθούν κατά την εκπαίδευση των μοντέλων.



5.1 Στιγμιότυπο κώδικα generate\_data

**Αριθμός Δειγμάτων (`num\_samples`) και Αντικειμένων (num\_items)**:

* Αυτές οι παράμετροι καθορίζουν πόσα δείγματα θα δημιουργηθούν και πόσα αντικείμενα θα περιλαμβάνει το καθένα.

**Τυχαίες Τιμές και Βάρη**:

* Για κάθε δείγμα, δημιουργούνται λίστες με τυχαίες τιμές (**values**) και βάρη (**weights**) για τα αντικείμενα. Οι τιμές κυμαίνονται μεταξύ 1 και 50, ενώ τα βάρη μεταξύ 1 και 20.

**Χωρητικότητα Σακιδίου (knapsack\_capacity)**:

* Η χωρητικότητα του σακιδίου ορίζεται σταθερά σε 50 για όλα τα δείγματα.

**Βέλτιστη Λύση (solution)**:

* Η συνάρτηση **knapsack\_dp** χρησιμοποιείται για να υπολογίσει τη βέλτιστη επιλογή αντικειμένων για κάθε δείγμα, χρησιμοποιώντας τη μέθοδο του δυναμικού προγραμματισμού. Αυτή η βέλτιστη λύση χρησιμοποιείται ως ο στόχος εκμάθησης.

**Δομή Δεδομένων**:

* Κάθε δείγμα περιλαμβάνει τις λίστες **values**, **weights**, την **knapsack\_capacity**, και το διάνυσμα **solution**, το οποίο αντιπροσωπεύει τη βέλτιστη επιλογή αντικειμένων.

### Υλοποίηση του Δυναμικού Προγραμματισμού (knapsack\_dp)

Για να διασφαλιστεί ότι τα δεδομένα εκπαίδευσης είναι όσο το δυνατόν πιο ακριβή, χρησιμοποιείται η συνάρτηση **knapsack\_dp** για τον υπολογισμό των βέλτιστων λύσεων μέσω της μεθόδου του δυναμικού προγραμματισμού. Αυτή η συνάρτηση υπολογίζει τη βέλτιστη επιλογή αντικειμένων για κάθε σύνολο δεδομένων, η οποία στη συνέχεια χρησιμοποιείται ως στόχος κατά την εκπαίδευση των μοντέλων.



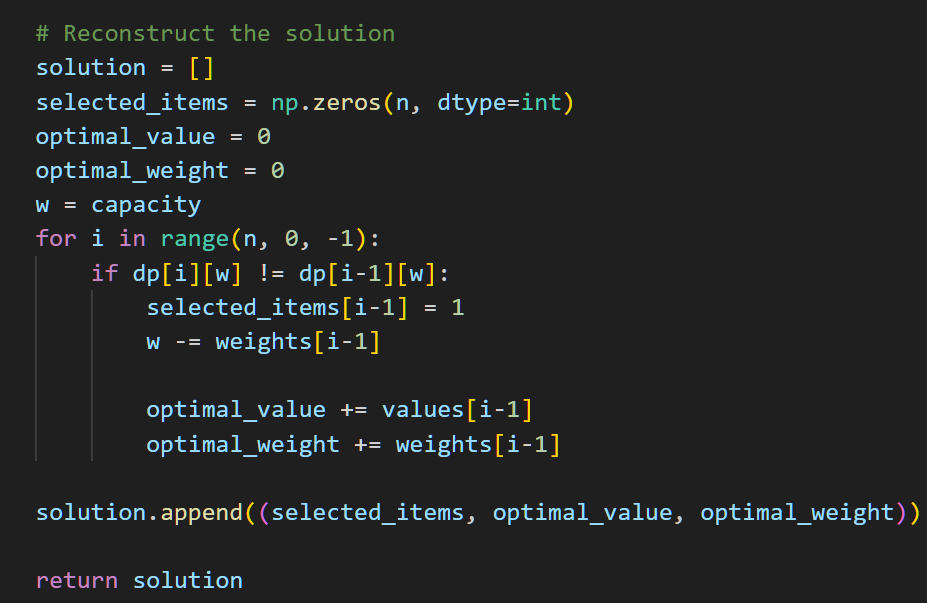
5.2.Στιγμιότυπο κώδικα knapsack\_dp

**Αρχικοποίηση Πίνακα Δυναμικού Προγραμματισμού (dp)**:

* Ο πίνακας **dp** είναι ένας δισδιάστατος πίνακας με διαστάσεις **(n + 1) x (capacity + 1)**, όπου n είναι ο αριθμός των αντικειμένων και **capacity** η χωρητικότητα του σακιδίου.

**Υπολογισμός Βέλτιστης Αξίας**:

* Η συνάρτηση υπολογίζει τη μέγιστη αξία που μπορεί να επιτευχθεί για κάθε αντικείμενο και κάθε δυνατό συνδυασμό βάρους, αποφασίζοντας αν το αντικείμενο θα πρέπει να συμπεριληφθεί στο σακίδιο ή όχι.



5.3.Σιτγμιότυπο κώδικα knapsack\_dp (2)

**Ανακατασκευή της Λύσης**:

* Μετά τον υπολογισμό της μέγιστης αξίας, η συνάρτηση ανακατασκευάζει τη βέλτιστη λύση. Ξεκινώντας από το τελευταίο αντικείμενο, καθορίζεται αν το αντικείμενο συμπεριλήφθηκε στο σακίδιο, και αν ναι, καταχωρείται ως επιλεγμένο.

**Επιστροφή Βέλτιστης Λύσης**:

* Η συνάρτηση επιστρέφει μια λίστα με τη βέλτιστη επιλογή αντικειμένων (**selected\_items**), τη συνολική αξία (**optimal\_value**), και το συνολικό βάρος (**optimal\_weight**) της λύσης.

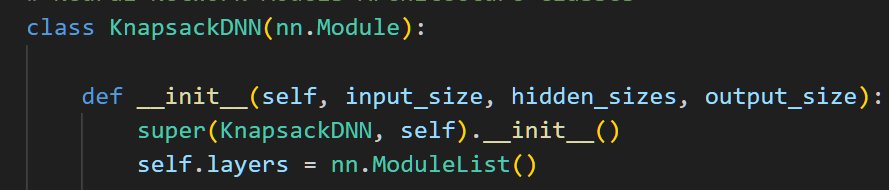
### Συμπέρασμα για την Προετοιμασία Δεδομένων

Η δημιουργία και προετοιμασία των δεδομένων αποτελεί ένα κρίσιμο βήμα για την επιτυχή εκπαίδευση των μοντέλων βαθιάς μάθησης. Τα δεδομένα πρέπει να είναι αξιόπιστα και κατάλληλα μορφοποιημένα, ώστε τα μοντέλα να μπορέσουν να μάθουν τις σωστές σχέσεις μεταξύ των εισόδων (χαρακτηριστικά των αντικειμένων) και των εξόδων (επιλογή αντικειμένων). Με αυτή τη διαδικασία, διασφαλίζεται ότι το DNN, το RNN και το DQN θα εκπαιδευτούν χρησιμοποιώντας ακριβή και κατάλληλα δεδομένα, οδηγώντας σε βέλτιστα αποτελέσματα.

## Βαθύ Νευρωνικό Δίκτυο (Deep Neural Network, DNN)

### Δημιουργία και Αρχιτεκτονική του Μοντέλου DNN

Το **KnapsackDNN** είναι μια κλάση που υλοποιεί ένα Βαθύ Νευρωνικό Δίκτυο (DNN) με πολλαπλά πλήρως συνδεδεμένα στρώματα, προκειμένου να λύσει το πρόβλημα του σακιδίου 0/1. Ας αναλύσουμε βήμα-βήμα τον τρόπο με τον οποίο η κλάση αυτή υλοποιείται:



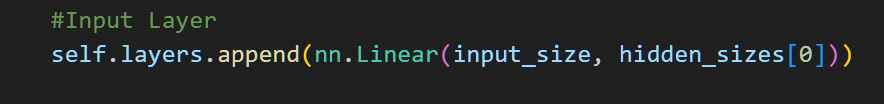
5.4.Σιτγμιότυπο κλάσης KnapsackDNN

**Κλάση KnapsackDNN:** Η κλάση αυτή κληρονομεί από την **nn.Module**, η οποία αποτελεί τη βάση για όλα τα νευρωνικά δίκτυα στο PyTorch. Στον αρχικοποιητή **(\_\_init\_\_),** ορίζονται οι διάφορες στρώσεις του δικτύου.

**Παράμετροι:**

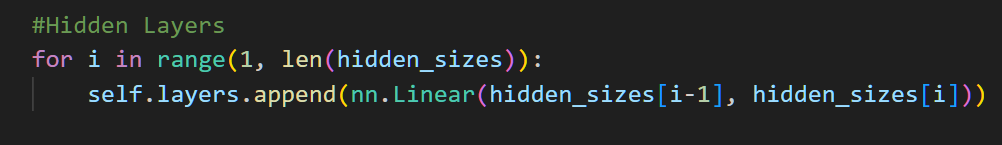
* **input\_size**: Το μέγεθος του διανύσματος εισόδου, δηλαδή ο αριθμός των χαρακτηριστικών (π.χ., βάρη, αξίες, χωρητικότητα).
* **hidden\_sizes**: Μια λίστα που περιέχει τα μεγέθη των νευρώνων για κάθε κρυφό στρώμα.
* **output\_size**: Το μέγεθος του διανύσματος εξόδου, το οποίο είναι συνήθως ίσο με τον αριθμό των αντικειμένων που πρέπει να επιλεγούν.

**nn.ModuleList:** Χρησιμοποιείται για τη διαχείριση μιας λίστας επιπέδων, που μπορούν εύκολα να προστεθούν και να χρησιμοποιηθούν στο δίκτυο.



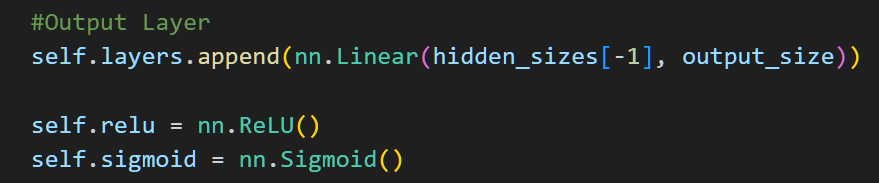
5.5. Σιτγμιότυπο κλάσης KnapsackDNN (2)

**Επίπεδο Εισόδου:** Το πρώτο επίπεδο προστίθεται στη λίστα των στρωμάτων. Είναι ένα πλήρως συνδεδεμένο επίπεδο (`**nn.Linear`**), που λαμβάνει το διάνυσμα εισόδου και το μετασχηματίζει με βάση τις παραμέτρους του πρώτου κρυφού στρώματος.



5.6. Σιτγμιότυπο κλάσης KnapsackDNN (3)

**Κρυφά Στρώματα:** Αυτή η λογική επανάληψης (loop) προσθέτει τα κρυφά στρώματα στο δίκτυο, συνδέοντας το κάθε στρώμα με το προηγούμενο μέσω πλήρως συνδεδεμένων επιπέδων. Το `**hidden\_sizes[i-1]`** καθορίζει το μέγεθος του προηγούμενου στρώματος, ενώ το `**hidden\_sizes[i]** `καθορίζει το μέγεθος του τρέχοντος στρώματος.



5.7. Σιτγμιότυπο κλάσης KnapsackDNN (4)

**Επίπεδο Εξόδου:** Το τελευταίο στρώμα είναι επίσης πλήρως συνδεδεμένο και συνδέει το τελευταίο κρυφό στρώμα με το επίπεδο εξόδου. Το `**output\_size**` είναι ο αριθμός των εξόδων, που αντιστοιχούν σε κάθε αντικείμενο (επιλογή ή μη επιλογή).

**Συναρτήσεις Ενεργοποίησης:** Δύο δημοφιλείς συναρτήσεις ενεργοποίησης χρησιμοποιούνται στο μοντέλο:

* **ReLU (Rectified Linear Unit):** Εφαρμόζεται στα κρυφά στρώματα και βοηθά στην εκμάθηση μη γραμμικών σχέσεων.
* **Sigmoid:** Χρησιμοποιείται στο επίπεδο εξόδου για να μετατρέψει την έξοδο σε πιθανότητες, που μπορούν να ερμηνευτούν ως δυαδικές αποφάσεις (0 ή 1).



5.8. Σιτγμιότυπο κλάσης KnapsackDNN (5)

**Μέθοδος forward:** Αυτή η μέθοδος καθορίζει τη ροή των δεδομένων μέσα από το δίκτυο:

* **Διαδοχική Επεξεργασία:** Τα δεδομένα περνούν μέσα από κάθε κρυφό στρώμα, εφαρμόζοντας τη συνάρτηση ReLU.
* **Τελική Απόφαση:** Στο τελευταίο επίπεδο, η έξοδος περνά από τη συνάρτηση sigmoid, που επιστρέφει τις τελικές αποφάσεις για κάθε αντικείμενο.

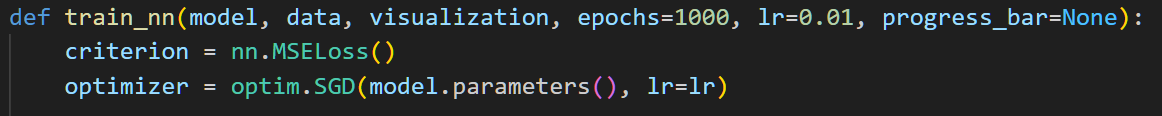
Η κλάση `**KnapsackDNN**` δημιουργεί ένα πλήρως συνδεδεμένο νευρωνικό δίκτυο, το οποίο είναι έτοιμο να εκπαιδευτεί για να προβλέπει ποια αντικείμενα πρέπει να συμπεριληφθούν στο σακίδιο ώστε να μεγιστοποιηθεί η αξία, χωρίς να ξεπεραστεί η χωρητικότητα.

### Εκπαίδευση του Μοντέλου DNN

Η εκπαίδευση ενός Deep Neural Network για την επίλυση του προβλήματος του σακιδίου 0/1 περιλαμβάνει τη διαδικασία προσαρμογής των παραμέτρων του δικτύου ώστε να επιτυγχάνει την καλύτερη δυνατή πρόβλεψη για το ποια αντικείμενα πρέπει να επιλεγούν. Η συνάρτηση **train\_nn** που παρατίθεται παρακάτω είναι υπεύθυνη για την εκπαίδευση του DNN.

**Περιγραφή της Συνάρτησης Εκπαίδευσης train\_nn**

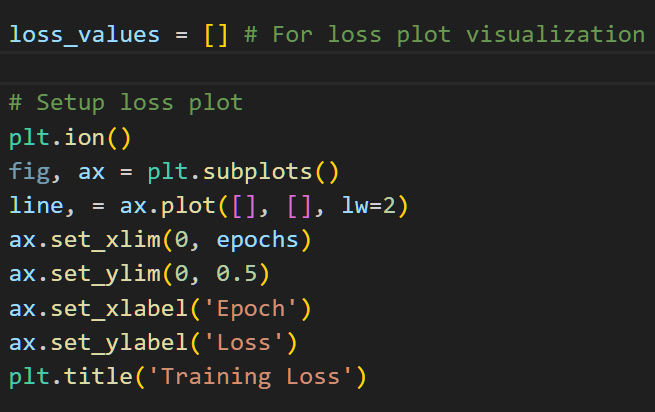
Η συνάρτηση train\_nn αναλαμβάνει την εκπαίδευση του DNN χρησιμοποιώντας τα δεδομένα εκπαίδευσης που έχουν παραχθεί και προετοιμαστεί. Ακολουθεί μια ανάλυση της λειτουργίας της.



5.9. Σιτγμιότυπο κώδικα train\_nn

**Αρχικοποίηση Συνάρτησης Εκπαίδευσης**:

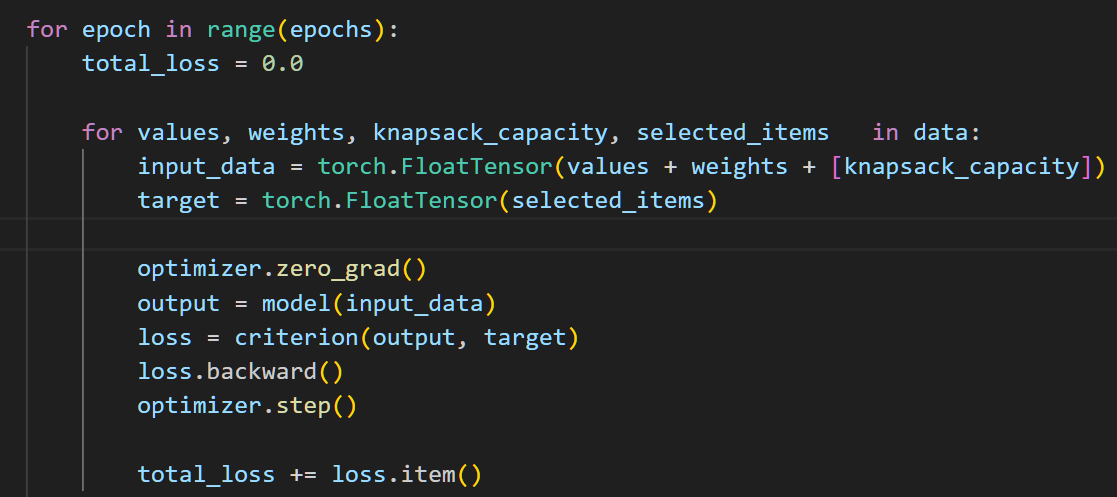
* Η εκπαίδευση του μοντέλου πραγματοποιείται για έναν ορισμένο αριθμό εποχών (**epochs**). Σε κάθε εποχή, τα δεδομένα εισάγονται στο δίκτυο, και τα βάρη του δικτύου προσαρμόζονται για να μειωθεί το σφάλμα.
* Η συνάρτηση χρησιμοποιεί την απώλεια μέσου τετραγωνικού σφάλματος (**MSELoss**) ως το κριτήριο απώλειας, που υπολογίζει τη διαφορά μεταξύ της πρόβλεψης του δικτύου και της πραγματικής λύσης.
* Ο βελτιστοποιητής που χρησιμοποιείται είναι η Στοχαστική Καθοδική Βελτιστοποίηση (**SGD**), η οποία προσαρμόζει τα βάρη του μοντέλου κατά τη διάρκεια της εκπαίδευσης με ρυθμό εκμάθησης (**lr**) 0.01.



5.10. Σιτγμιότυπο κώδικα train\_nn (2)

**Οπτικοποίηση της Απώλειας Εκπαίδευσης**:

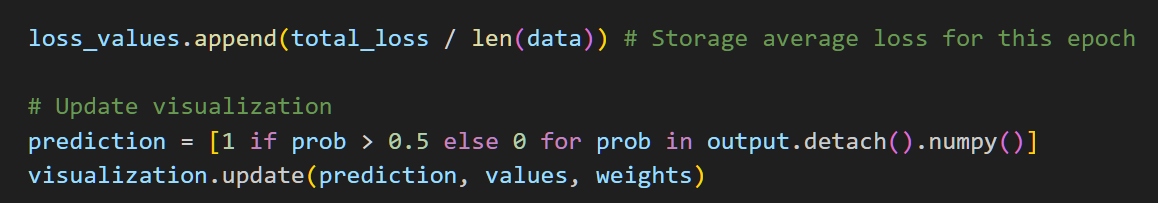
* Η συνάρτηση δημιουργεί ένα διάγραμμα για την οπτικοποίηση της απώλειας εκπαίδευσης σε πραγματικό χρόνο, χρησιμοποιώντας τη βιβλιοθήκη **matplotlib**. Η απώλεια υπολογίζεται σε κάθε εποχή και αποθηκεύεται στη λίστα **loss\_values**, η οποία στη συνέχεια χρησιμοποιείται για την ενημέρωση του γραφήματος.



5.11. Σιτγμιότυπο κώδικα train\_nn (3)

**Εκπαίδευση του Μοντέλου σε Κάθε Εποχή**:

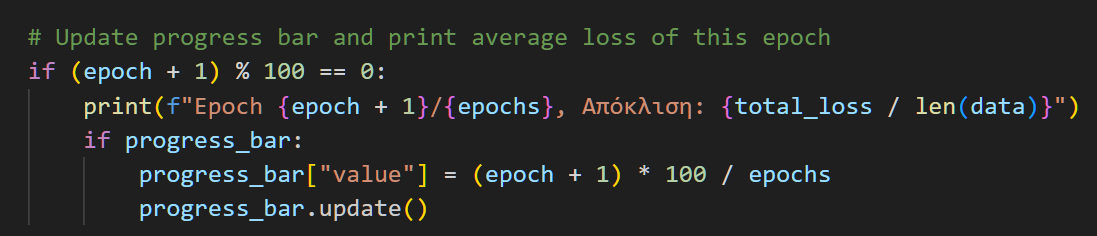
* Η εκπαίδευση πραγματοποιείται σε κύκλους (**epochs**), όπου σε κάθε κύκλο το δίκτυο εκπαιδεύεται σε όλο το σύνολο δεδομένων.
* Για κάθε δείγμα εκπαίδευσης, οι τιμές και τα βάρη των αντικειμένων, καθώς και η χωρητικότητα του σακιδίου, συνδυάζονται σε ένα ενιαίο διάνυσμα εισόδου (**input\_data**).
* Το μοντέλο παράγει μια πρόβλεψη (**output**) για την επιλογή των αντικειμένων. Το σφάλμα μεταξύ της πρόβλεψης και της πραγματικής λύσης υπολογίζεται με τη χρήση της συνάρτησης απώλειας (**loss**).
* Το σφάλμα αυτό χρησιμοποιείται για την προσαρμογή των παραμέτρων του μοντέλου μέσω της οπισθοδιάδοσης και της ενημέρωσης των βαρών.



5.12. Σιτγμιότυπο κώδικα train\_nn (4)

**Αποθήκευση και Οπτικοποίηση Απώλειας**:

* Η μέση απώλεια για κάθε εποχή αποθηκεύεται και χρησιμοποιείται για την ενημέρωση του γραφήματος απώλειας. Επιπλέον, η οπτικοποίηση της πρόβλεψης του μοντέλου ενημερώνεται σε κάθε εποχή, επιτρέποντας την παρακολούθηση της απόδοσης του μοντέλου σε πραγματικό χρόνο.



5.13. Σιτγμιότυπο κώδικα train\_nn (5)

**Ενημέρωση Γραμμής Προόδου και Εμφάνιση Αποτελεσμάτων**:

* Κατά τη διάρκεια της εκπαίδευσης, η συνάρτηση ενημερώνει μια γραμμή προόδου που δείχνει την πρόοδο της εκπαίδευσης σε κάθε εποχή. Επίσης, εμφανίζει την απώλεια κάθε 100 εποχές, επιτρέποντας στον χρήστη να παρακολουθεί τη βελτίωση του μοντέλου.



5.14. Σιτγμιότυπο κώδικα train\_nn (6)

**Τελική Ενημέρωση Γραφήματος και Θέσης Παραθύρου**:

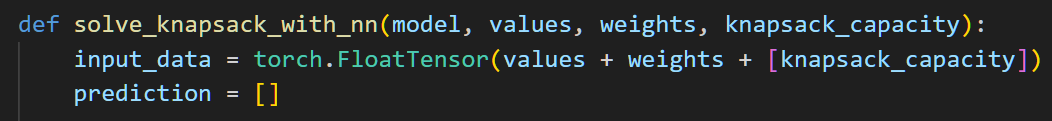
* Η τελική ενημέρωση του γραφήματος και της θέσης του παραθύρου οπτικοποίησης πραγματοποιείται μετά την ολοκλήρωση της εκπαίδευσης. Το παράθυρο του γραφήματος τοποθετείται δίπλα στο παράθυρο του tkinter, επιτρέποντας τη σύγχρονη προβολή των αποτελεσμάτων.

### Επίλυση του Προβλήματος με το Εκπαιδευμένο Μοντέλο DNN

Μετά την εκπαίδευση του Deep Neural Network, το μοντέλο μπορεί να χρησιμοποιηθεί για την επίλυση νέων περιπτώσεων του προβλήματος Knapsack 0/1. Η συνάρτηση **solve\_knapsack\_with\_nn** υλοποιεί αυτή τη διαδικασία, επιτρέποντας στο εκπαιδευμένο μοντέλο να κάνει προβλέψεις για το ποια αντικείμενα θα πρέπει να επιλεγούν, προκειμένου να μεγιστοποιηθεί η συνολική αξία χωρίς να υπερβεί τη χωρητικότητα του σακιδίου.

**Περιγραφή της Συνάρτησης solve\_knapsack\_with\_nn**

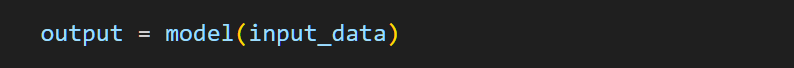
Η συνάρτηση **solve\_knapsack\_with\_nn** χρησιμοποιεί το εκπαιδευμένο μοντέλο DNN για να λύσει το πρόβλημα του σακιδίου. Ας δούμε τα βήματα της διαδικασίας.



5.15. Σιτγμιότυπο κώδικα solve\_knapsack\_with\_nn

**Δημιουργία του Δεδομένου Εισόδου**:

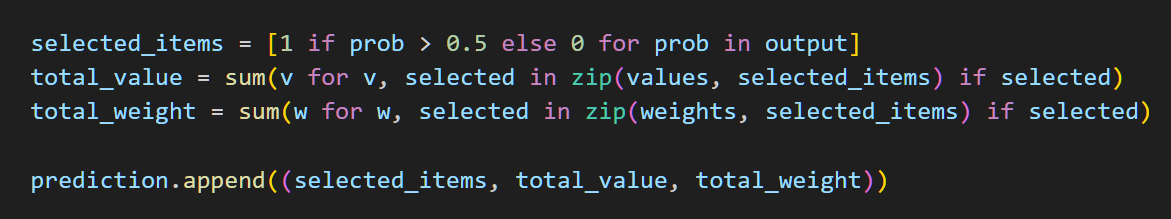
* Η συνάρτηση δέχεται ως είσοδο τις τιμές (**values**) και τα βάρη (**weights**) των αντικειμένων, καθώς και τη χωρητικότητα του σακιδίου (**knapsack\_capacity**).
* Αυτά τα δεδομένα συνδυάζονται σε ένα ενιαίο διάνυσμα εισόδου (**input\_data**), το οποίο περιλαμβάνει όλες τις τιμές, τα βάρη, και τη χωρητικότητα, παρόμοια με τη διαδικασία κατά την εκπαίδευση.



5.16.Σιτγμιότυπο κώδικα solve\_knapsack\_with\_nn (2)

**Πρόβλεψη μέσω του Εκπαιδευμένου Μοντέλου**:

* Το εκπαιδευμένο μοντέλο (**model**) λαμβάνει το διάνυσμα εισόδου και παράγει ένα αποτέλεσμα (**output**), το οποίο αντιπροσωπεύει την πιθανότητα επιλογής για κάθε αντικείμενο.



5.17.Σιτγμιότυπο κώδικα solve\_knapsack\_with\_nn (3)

**Καθορισμός Επιλεγμένων Αντικειμένων**:

* Για κάθε αντικείμενο, η συνάρτηση συγκρίνει την πιθανότητα που προβλέπεται από το μοντέλο με ένα κατώφλι (0.5). Αν η πιθανότητα υπερβαίνει το κατώφλι, το αντικείμενο θεωρείται επιλεγμένο (**selected\_items**).
* Η συνολική αξία (**total\_value**) και το συνολικό βάρος (**total\_weight**) των επιλεγμένων αντικειμένων υπολογίζονται στη συνέχεια.



5.18.Σιτγμιότυπο κώδικα solve\_knapsack\_with\_nn (4)

**Επιστροφή του Αποτελέσματος**:

* Η συνάρτηση επιστρέφει μια λίστα (**prediction**), η οποία περιέχει τις επιλεγμένες αντικείμενα, τη συνολική αξία, και το συνολικό βάρος. Αυτά τα αποτελέσματα μπορούν να συγκριθούν με τη βέλτιστη λύση για να αξιολογηθεί η απόδοση του μοντέλου.

**Εφαρμογή και Χρήση του Εκπαιδευμένου Μοντέλου**

Με τη χρήση της συνάρτησης **solve\_knapsack\_with\_nn**, το εκπαιδευμένο DNN μπορεί να εφαρμοστεί σε νέες περιπτώσεις του προβλήματος του σακιδίου, προβλέποντας τη βέλτιστη επιλογή αντικειμένων βάσει των τιμών και των βαρών τους. Το αποτέλεσμα μπορεί στη συνέχεια να συγκριθεί με τη βέλτιστη λύση που υπολογίζεται με άλλες μεθόδους, όπως ο δυναμικός προγραμματισμός, για να αξιολογηθεί η αποτελεσματικότητα του μοντέλου.

### Βελτιστοποίηση Υπερπαραμέτρων Μοντέλου DNN

Η βελτιστοποίηση των υπερπαραμέτρων είναι ένα κρίσιμο βήμα για τη βελτίωση της απόδοσης ενός μοντέλου νευρωνικού δικτύου. Για το μοντέλο DNN, χρησιμοποιήθηκε η μέθοδος grid search για την εξερεύνηση διαφόρων συνδυασμών βασικών υπερπαραμέτρων: ρυθμός μάθησης, μέγεθος των κρυφών επιπέδων και αριθμός εποχών εκπαίδευσης. Στόχος ήταν ο εντοπισμός της ρύθμισης που ελαχιστοποιεί το βέλτιστο κενό, δηλαδή τη διαφορά μεταξύ της λύσης που παρέχει το μοντέλο και της βέλτιστης λύσης.

**Α. Πλέγμα Υπερπαραμέτρων**

Οι ακόλουθες υπερπαράμετροι εξετάστηκαν για βελτιστοποίηση:

* Ρυθμός Μάθησης (**lr**): [0.01, 0.001, 0.0001, 0.00001]
* Μέγεθος Κρυφών Επιπέδων (**hidden**\_**sizes**): [[64, 64], [128, 128], [256, 128]]
* Αριθμός Εποχών (**epochs**): [500, 1000]

Αυτές οι παράμετροι επιλέχθηκαν βάσει προηγούμενων πειραμάτων και βέλτιστων πρακτικών στην εκπαίδευση βαθιών νευρωνικών δικτύων.

**Β. Υλοποίηση Grid Search**

Η διαδικασία **grid** **search** υλοποιήθηκε χρησιμοποιώντας την κλάση **ParameterGrid**, η οποία δημιουργεί συστηματικά όλους τους δυνατούς συνδυασμούς των καθορισμένων υπερπαραμέτρων. Για κάθε συνδυασμό, αρχικοποιήθηκε και εκπαιδεύτηκε ένα νέο μοντέλο DNN χρησιμοποιώντας τη λειτουργία **train**\_**nn**. Μετά την εκπαίδευση, η απόδοση του μοντέλου αξιολογήθηκε σε ένα σύνολο επικύρωσης χρησιμοποιώντας τη λειτουργία **evaluate**\_**model**\_**nn**.

Η μετρική αξιολόγησης ήταν η **απώλεια βελτιστοποίησης (optimality\_gap)**, το οποίο υπολογίζεται ως:

Ο στόχος ήταν η ελαχιστοποίηση αυτού του κενού, υποδεικνύοντας ότι οι προβλέψεις του μοντέλου ευθυγραμμίζονται στενά με την πραγματική βέλτιστη λύση.

**Γ. Βέλτιστες Υπερπαράμετροι**

Μετά την εξέταση όλων των δυνατών συνδυασμών, αναγνωρίστηκε το σύνολο των υπερπαραμέτρων με την καλύτερη απόδοση βάσει του μικρότερου βέλτιστου κενού. Οι βέλτιστες υπερπαράμετροι ήταν:

* **Ρυθμός Μάθησης (lr)**: [0.01]
* **Μέγεθος Κρυφών Επιπέδων (hidden\_sizes)**: [256, 128]
* **Αριθμός Εποχών (epochs)**: [1000]

**Δ. Συμπέρασμα**

Η διαδικασία **grid** **search** για τη βελτιστοποίηση των υπερπαραμέτρων παρείχε μια ικανοποιητική διαμόρφωση για το μοντέλο DNN, διασφαλίζοντας ότι αυτό ήταν καλά ρυθμισμένο για το πρόβλημα. Αυτή η διαμόρφωση χρησιμοποιήθηκε στη συνέχεια για όλες τις επόμενες εκπαιδευτικές και αξιολογικές εργασίες, διασφαλίζοντας συνέπεια και βέλτιστη απόδοση στα πειράματα.

Ο πλήρης κώδικας για τη διαδικασία βελτιστοποίησης υπερπαραμέτρων παρέχεται στο Παράρτημα Β. Αυτό περιλαμβάνει την υλοποίηση του **grid** **search**, της εκπαίδευσης του μοντέλου και των λειτουργιών αξιολόγησης.

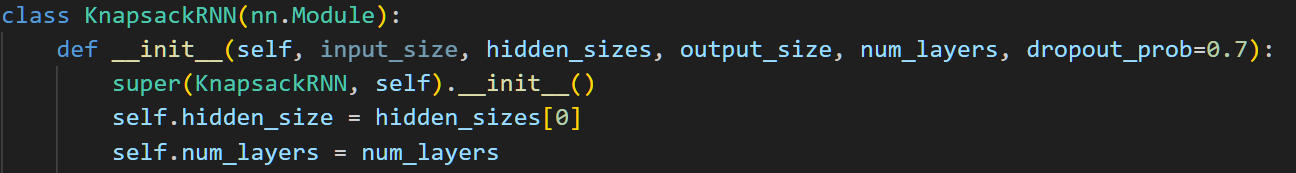
## Μοντέλο Επαναλαμβανόμενου Νευρωνικού Δικτύου (Recurrent Neural Network, RNN)

### Δημιουργία και Αρχιτεκτονική του Μοντέλου RNN

Στην ενότητα αυτή, θα αναλυθεί η δημιουργία και η αρχιτεκτονική του Επαναλαμβανόμενου Νευρωνικού Δικτύου (RNN) που χρησιμοποιείται για την επίλυση του προβλήματος Knapsack 0/1. Το συγκεκριμένο μοντέλο βασίζεται σε LSTM (Long Short-Term Memory) δίκτυα, τα οποία είναι τύπος επαναλαμβανόμενου νευρωνικού δικτύου, σχεδιασμένα να επεξεργάζονται και να προβλέπουν ακολουθίες δεδομένων.

**Αρχιτεκτονική του Μοντέλου KnapsackRNN**

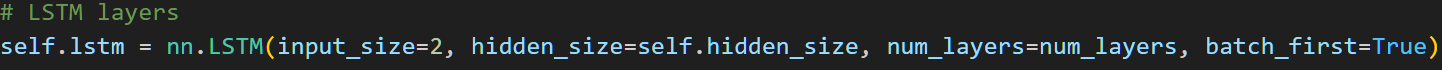
Το μοντέλο **KnapsackRNN** αποτελείται από διάφορα επίπεδα, όπως LSTM, πλήρως συνδεδεμένα επίπεδα, και dropout, τα οποία συνεργάζονται για την επίλυση του προβλήματος του σακιδίου.



5.19.Σιτγμιότυπο κλάσης KnapsackRNN

**Αρχικοποίηση του Μοντέλου**:

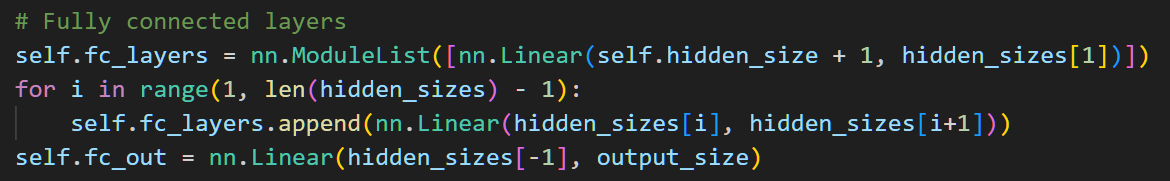
* Η κλάση **KnapsackRNN** επεκτείνει τη βασική κλάση **nn.Module** του PyTorch.
* Η αρχικοποίηση περιλαμβάνει τον καθορισμό του μεγέθους του κρυφού επιπέδου (**hidden\_size**), του αριθμού των επιπέδων LSTM (**num\_layers**), και της πιθανότητας dropout (**dropout\_prob**).



5.20.Σιτγμιότυπο κλάσης KnapsackRNN (2)

**Επίπεδα LSTM**:

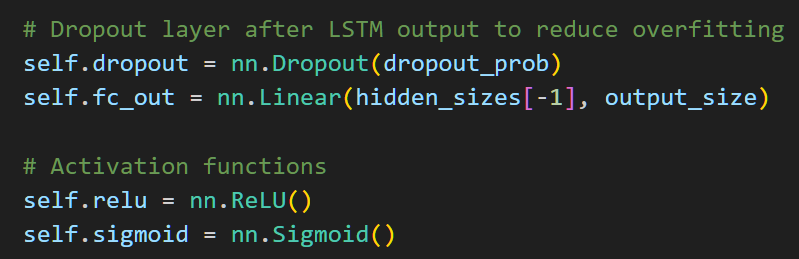
* Το LSTM αποτελεί τον πυρήνα του επαναλαμβανόμενου μέρους του δικτύου. Είναι υπεύθυνο για την επεξεργασία ακολουθιών δεδομένων (όπως οι τιμές και τα βάρη των αντικειμένων) και τη διατήρηση της πληροφορίας από προηγούμενες χρονικές στιγμές μέσω των μηχανισμών μνήμης.
* Το **input\_size** είναι 2, καθώς το δίκτυο λαμβάνει ως είσοδο ένα ζεύγος τιμών και βαρών σε κάθε χρονικό βήμα.
* Το LSTM είναι διαμορφωμένο να λειτουργεί με την παράμετρο **batch\_first=True**, ώστε το πρώτο διάστατο της εισόδου να αντιπροσωπεύει τα δείγματα της παρτίδας (batch).



5.21.Σιτγμιότυπο κλάσης KnapsackRNN (3)

**Πλήρως Συνδεδεμένα Επίπεδα**:

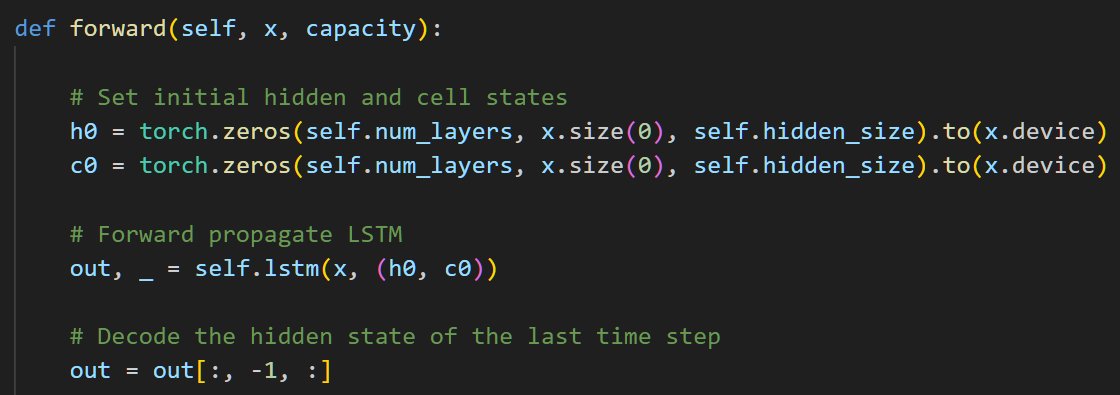
* Μετά την επεξεργασία των δεδομένων από το LSTM, το δίκτυο περνά τις εξόδους του μέσα από μια σειρά από πλήρως συνδεδεμένα επίπεδα (fully connected layers).
* Το πρώτο πλήρως συνδεδεμένο επίπεδο λαμβάνει ως είσοδο το κρυφό επίπεδο από το LSTM και την χωρητικότητα του σακιδίου. Το τελευταίο επίπεδο παράγει την έξοδο του δικτύου, που είναι ένα δυαδικό διάνυσμα που υποδεικνύει ποια αντικείμενα θα πρέπει να επιλεγούν.



5.22.Σιτγμιότυπο κλάσης KnapsackRNN (4)

**Dropout και Συναρτήσεις Ενεργοποίησης**:

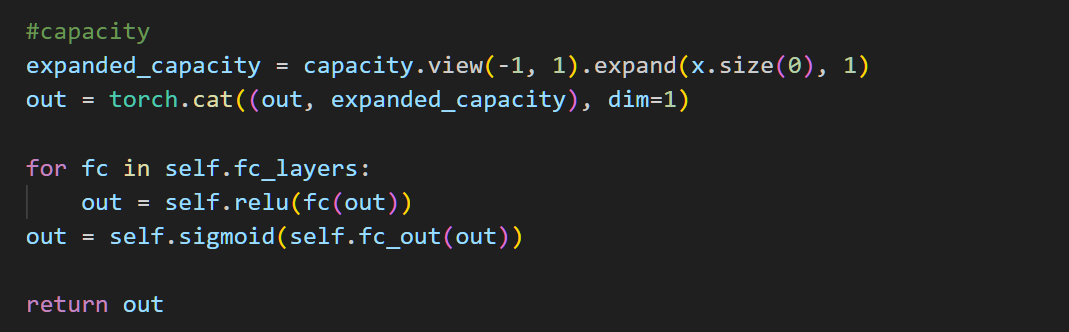
* Το Dropout εφαρμόζεται μετά την έξοδο του LSTM για να αποτρέψει το overfitting, δηλαδή την υπερβολική προσαρμογή του μοντέλου στα δεδομένα εκπαίδευσης.
* Το μοντέλο χρησιμοποιεί τη συνάρτηση ενεργοποίησης ReLU (Rectified Linear Unit) στα πλήρως συνδεδεμένα επίπεδα και τη συνάρτηση sigmoid στην τελική έξοδο για να περιορίσει τις προβλέψεις σε τιμές μεταξύ 0 και 1.



5.23.Σιτγμιότυπο κλάσης KnapsackRNN (5)

**Forward Μέθοδος (Προώθηση)**:

* Στην μέθοδο **forward**, το δίκτυο διαδίδει τα δεδομένα μέσω του LSTM, αρχικά ρυθμίζοντας τις καταστάσεις των κρυφών επιπέδων και των κυψελών.
* Το LSTM παράγει μια έξοδο για κάθε χρονικό βήμα. Ωστόσο, για το πρόβλημα του σακιδίου, ενδιαφέρει κυρίως η έξοδος από το τελευταίο χρονικό βήμα, που περιλαμβάνει την τελική κρυφή κατάσταση.



5.24.Σιτγμιότυπο κλάσης KnapsackRNN (6)

**Συνδυασμός με την Χωρητικότητα του Σακιδίου και Τελική Πρόβλεψη**:

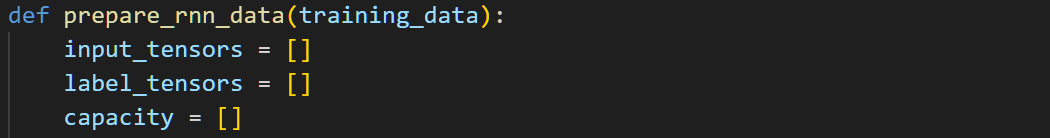
* Η τελική κρυφή κατάσταση από το LSTM συνδυάζεται με τη χωρητικότητα του σακιδίου, που αναπαράγεται και επεκτείνεται σε ολόκληρο το batch.
* Το συνδυασμένο διάνυσμα περνάει μέσα από τα πλήρως συνδεδεμένα επίπεδα με ReLU και τέλος από την έξοδο του δικτύου με sigmoid, που δίνει την πρόβλεψη για την επιλογή των αντικειμένων.

### Προετοιμασία Δεδομένων για το RNN

Η προετοιμασία των δεδομένων για το Επαναλαμβανόμενο Νευρωνικό Δίκτυο (RNN) είναι μια κρίσιμη διαδικασία, καθώς τα δεδομένα πρέπει να μορφοποιηθούν κατάλληλα για να εισαχθούν στο δίκτυο. Στην περίπτωση του RNN που χρησιμοποιούμε για την επίλυση του προβλήματος Knapsack 0/1, τα δεδομένα μορφοποιούνται έτσι ώστε κάθε ζεύγος τιμής και βάρους ενός αντικειμένου να αντιμετωπίζεται ως ένα χρονικό βήμα σε μια ακολουθία. Η συνάρτηση **prepare\_rnn\_data** αναλαμβάνει να μετατρέψει τα δεδομένα εκπαίδευσης σε κατάλληλη μορφή για το RNN.

**Περιγραφή της Συνάρτησης prepare\_rnn\_data**

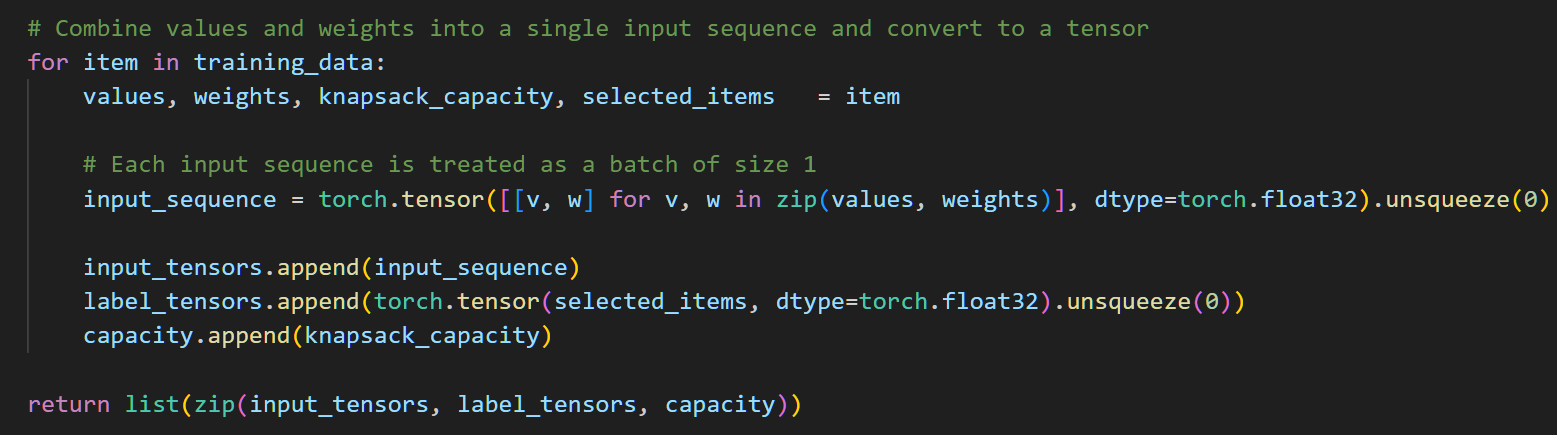
Η συνάρτηση **prepare\_rnn\_data** διαχειρίζεται τη μετατροπή των δεδομένων εκπαίδευσης σε μορφή που μπορεί να επεξεργαστεί το RNN, δημιουργώντας ακολουθίες εισόδων και αντίστοιχες ετικέτες (labels).



5.25.Σιτγμιότυπο κώδικα prepare\_rnn\_data

**Αρχικοποίηση των Δομών Δεδομένων**:

* Η συνάρτηση αρχικοποιεί τρεις λίστες: **input\_tensors** για τις εισόδους, **label\_tensors** για τις ετικέτες των δεδομένων, και **capacity** για την αποθήκευση της χωρητικότητας του σακιδίου που συνδέεται με κάθε δείγμα.



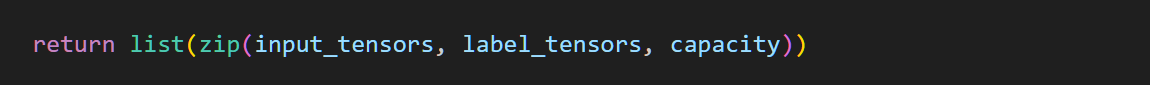
5.26.Σιτγμιότυπο κώδικα prepare\_rnn\_data (2)

**Συνδυασμός Τιμών και Βαρών σε Ακολουθία**:

* Για κάθε δείγμα στα δεδομένα εκπαίδευσης, η συνάρτηση συνδυάζει τις τιμές και τα βάρη των αντικειμένων σε ακολουθίες ζευγών [**τιμή, βάρος**]. Αυτή η ακολουθία μετατρέπεται σε tensor τύπου **torch.FloatTensor**.
* Η προσθήκη της διάστασης **unsqueeze(0)** καθιστά το κάθε δείγμα μια παρτίδα (batch) με μέγεθος 1, κάτι που είναι απαραίτητο για τη σωστή επεξεργασία από το RNN.

**Ετικέτες και Χωρητικότητα Σακιδίου**:

* Οι ετικέτες για κάθε δείγμα, που αντιπροσωπεύουν την επιλογή των αντικειμένων, αποθηκεύονται ως tensors στο **label\_tensors**.
* Η χωρητικότητα του σακιδίου για κάθε δείγμα αποθηκεύεται στη λίστα **capacity**.



5.27.Σιτγμιότυπο κώδικα prepare\_rnn\_data (3)

**Δημιουργία Συνδυασμένων Δεδομένων**:

* Η συνάρτηση επιστρέφει μια λίστα με ζεύγη (triples) που περιλαμβάνουν τις εισόδους, τις ετικέτες, και την χωρητικότητα του σακιδίου για κάθε δείγμα. Αυτή η μορφή δεδομένων είναι κατάλληλη για να εισαχθεί στο RNN κατά την εκπαίδευση.

**Χρήση της Προετοιμασμένης Δομής Δεδομένων**

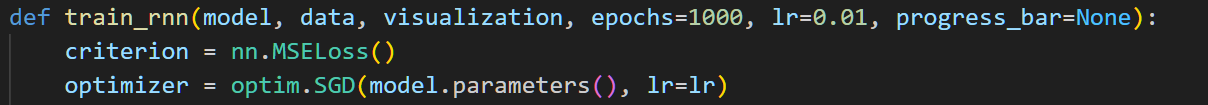
Με τη συνάρτηση **prepare\_rnn\_data**, τα δεδομένα μορφοποιούνται κατάλληλα ώστε να εισαχθούν στο RNN, το οποίο επεξεργάζεται ακολουθίες δεδομένων σε κάθε χρονικό βήμα. Αυτή η διαδικασία διασφαλίζει ότι το RNN μπορεί να μάθει τις σχέσεις μεταξύ των τιμών και των βαρών των αντικειμένων και να προβλέψει τη βέλτιστη επιλογή για το σακίδιο.

### Εκπαίδευση του Μοντέλου RNN

Η εκπαίδευση του Επαναλαμβανόμενου Νευρωνικού Δικτύου (RNN) για την επίλυση του προβλήματος Knapsack 0/1 περιλαμβάνει τη διαδικασία προσαρμογής των παραμέτρων του δικτύου ώστε να μάθει να επιλέγει τα αντικείμενα που μεγιστοποιούν τη συνολική αξία χωρίς να υπερβαίνει τη χωρητικότητα του σακιδίου. Η συνάρτηση **train\_rnn** που παρατίθεται παρακάτω είναι υπεύθυνη για την εκπαίδευση του RNN.

**Περιγραφή της Συνάρτησης Εκπαίδευσης train\_rnn**

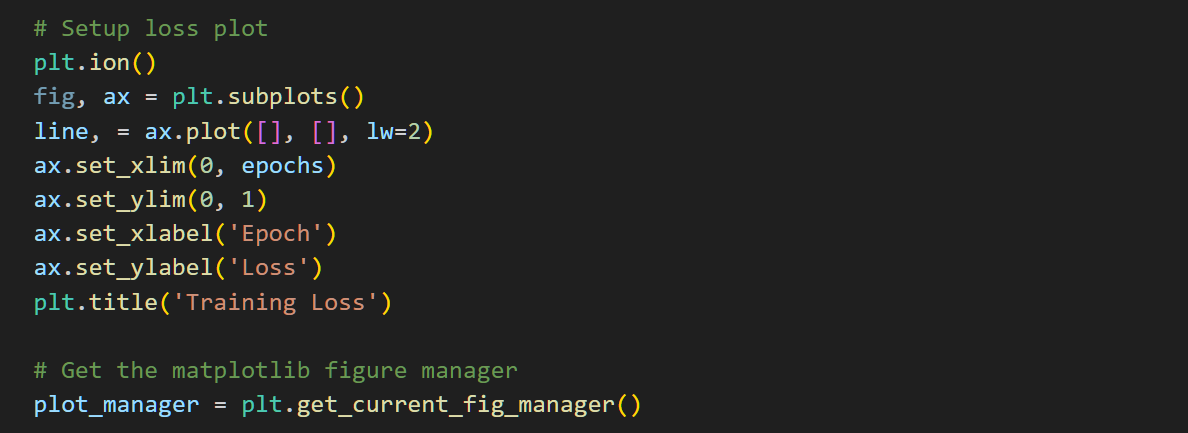
Η συνάρτηση **train\_rnn** εκπαιδεύει το RNN χρησιμοποιώντας δεδομένα εκπαίδευσης που έχουν προετοιμαστεί. Η διαδικασία περιλαμβάνει την προώθηση των δεδομένων μέσα από το δίκτυο, τον υπολογισμό της απώλειας, και την προσαρμογή των βαρών του δικτύου μέσω της οπισθοδιάδοσης.



5.28.Σιτγμιότυπο κώδικα train\_rnn

**Αρχικοποίηση Εκπαίδευσης**:

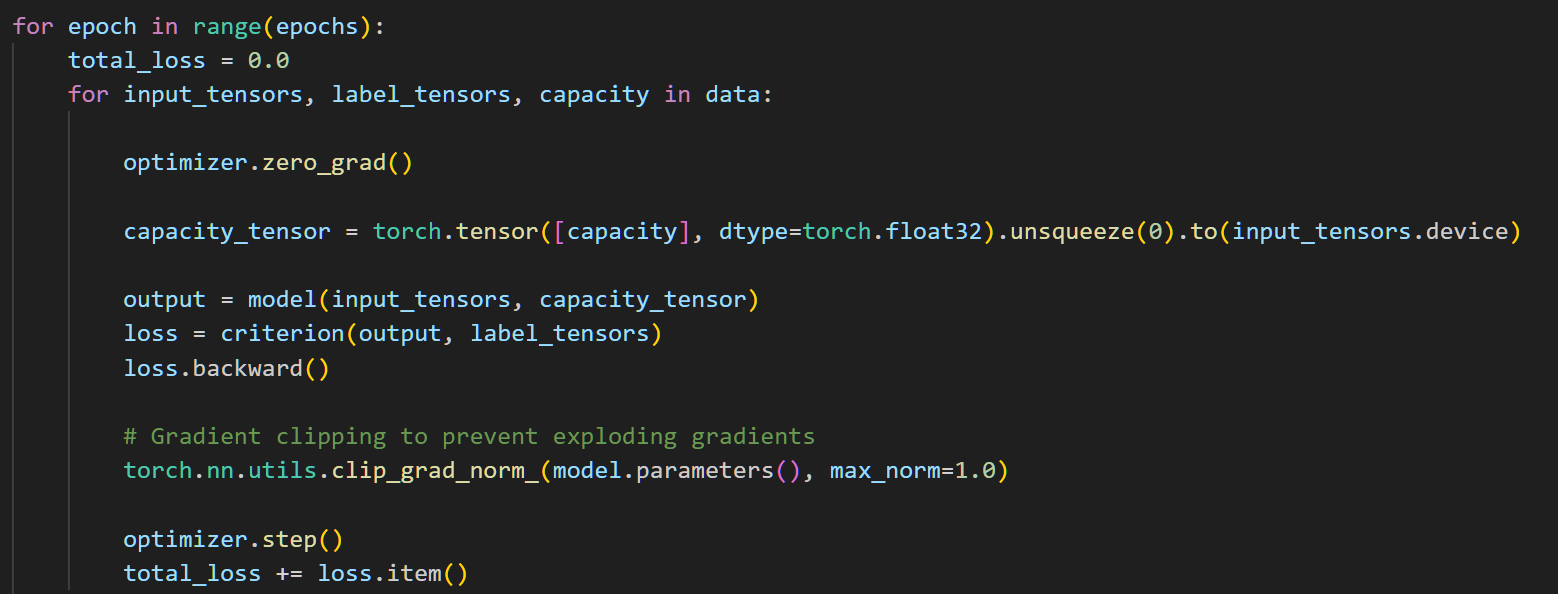
* Το μοντέλο εκπαιδεύεται για έναν συγκεκριμένο αριθμό εποχών (**epochs**). Κατά τη διάρκεια της εκπαίδευσης, χρησιμοποιείται η συνάρτηση απώλειας MSE (**MSELoss**), η οποία μετρά τη διαφορά μεταξύ των προβλέψεων του μοντέλου και των πραγματικών ετικετών (labels).
* Η Στοχαστική Καθοδική Βελτιστοποίηση (**SGD**) χρησιμοποιείται ως βελτιστοποιητής για την προσαρμογή των παραμέτρων του δικτύου.



5.29.Σιτγμιότυπο κώδικα train\_rnn (2)

**Οπτικοποίηση της Απώλειας**:

* Η απώλεια εκπαίδευσης παρακολουθείται και οπτικοποιείται σε πραγματικό χρόνο χρησιμοποιώντας τη βιβλιοθήκη **matplotlib**. Η συνάρτηση δημιουργεί ένα διάγραμμα που ενημερώνεται σε κάθε εποχή, επιτρέποντας τη διαρκή παρακολούθηση της απόδοσης του μοντέλου.



5.30.Σιτγμιότυπο κώδικα train\_rnn (3)

**Διαδικασία Εκπαίδευσης σε Κάθε Εποχή**:

* Σε κάθε εποχή, το δίκτυο εκπαιδεύεται σε όλα τα δείγματα δεδομένων. Για κάθε δείγμα, οι τιμές και τα βάρη των αντικειμένων, καθώς και η χωρητικότητα του σακιδίου, προωθούνται μέσα από το δίκτυο.
* Το σφάλμα υπολογίζεται ως η διαφορά μεταξύ της πρόβλεψης του δικτύου και των πραγματικών ετικετών, και αυτή η απώλεια χρησιμοποιείται για την ενημέρωση των βαρών του μοντέλου μέσω της οπίσθιας διάδοσης.
* Η χρήση του gradient clipping βοηθάει στην αποτροπή εκρηκτικών gradient κατά τη διάρκεια της εκπαίδευσης, διασφαλίζοντας σταθερότερη εκπαίδευση.



5.31.Σιτγμιότυπο κώδικα train\_rnn (4)

**Αποθήκευση και Οπτικοποίηση Απώλειας**:

* Η μέση απώλεια για κάθε εποχή αποθηκεύεται και χρησιμοποιείται για την ενημέρωση του γραφήματος απώλειας.
* Παράλληλα, η οπτικοποίηση του τρόπου επιλογής των αντικειμένων από το μοντέλο ενημερώνεται σε κάθε εποχή, παρέχοντας μια γραφική αναπαράσταση της απόδοσης του δικτύου.



5.32.Σιτγμιότυπο κώδικα train\_rnn (5)

**Ενημέρωση Γραμμής Προόδου και Γραφήματος Απώλειας**:

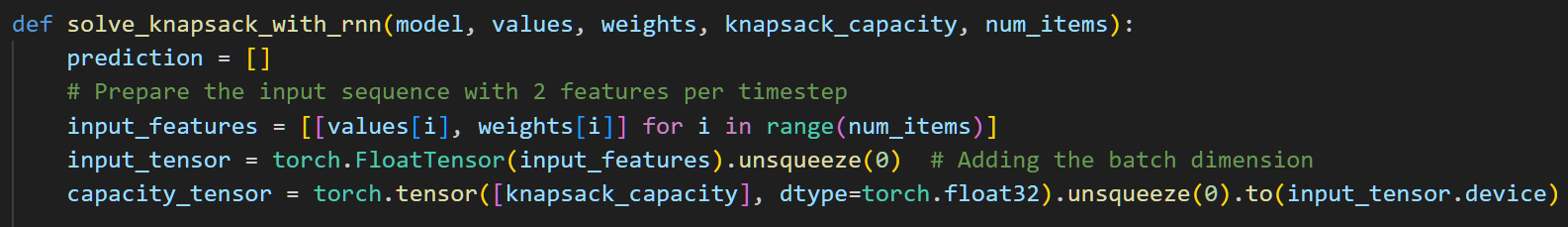
* Κατά τη διάρκεια της εκπαίδευσης, η γραμμή προόδου και το γράφημα απώλειας ενημερώνονται τακτικά, επιτρέποντας την παρακολούθηση της απόδοσης του μοντέλου σε πραγματικό χρόνο.

### Επίλυση του Προβλήματος με το Εκπαιδευμένο Μοντέλο RNN

Μετά την εκπαίδευση του Επαναλαμβανόμενου Νευρωνικού Δικτύου (RNN), το μοντέλο μπορεί να χρησιμοποιηθεί για την επίλυση νέων περιπτώσεων του προβλήματος Knapsack 0/1. Η συνάρτηση **solve\_knapsack\_with\_rnn** είναι υπεύθυνη για την αξιοποίηση του εκπαιδευμένου μοντέλου προκειμένου να προβλέψει την επιλογή αντικειμένων που μεγιστοποιούν τη συνολική αξία χωρίς να υπερβεί τη χωρητικότητα του σακιδίου.

**Περιγραφή της Συνάρτησης solve\_knapsack\_with\_rnn**

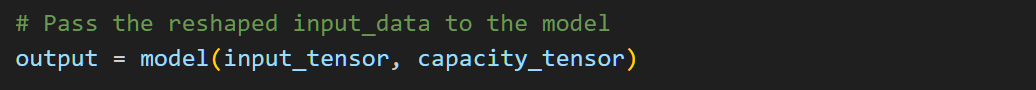
Η συνάρτηση **solve\_knapsack\_with\_rnn** χρησιμοποιεί το εκπαιδευμένο RNN για να επιλύσει το πρόβλημα του σακιδίου. Η διαδικασία περιλαμβάνει την προετοιμασία των εισόδων, τη διέλευση των δεδομένων μέσω του μοντέλου και την ερμηνεία των εξόδων για την επιλογή των αντικειμένων.



5.33.Σιτγμιότυπο κώδικα train\_rnn (6)

**Προετοιμασία των Εισόδων**:

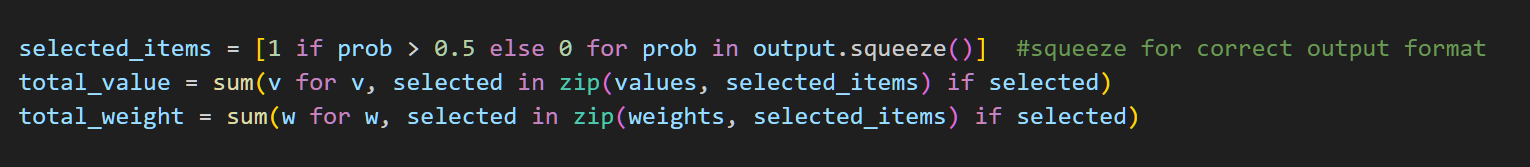
* Η συνάρτηση λαμβάνει τις τιμές (**values**) και τα βάρη (**weights**) των αντικειμένων, καθώς και τη χωρητικότητα του σακιδίου (**knapsack**\_**capacity**).
* Τα δεδομένα αυτά συνδυάζονται σε ακολουθίες χαρακτηριστικών (**input**\_**features**), όπου κάθε χρονικό βήμα περιέχει ένα ζεύγος τιμής και βάρους.
* Η ακολουθία αυτή μετατρέπεται σε ένα tensor (**input**\_**tensor**), στο οποίο προστίθεται η διάσταση batch μέσω της χρήσης της **unsqueeze(0)** για να διασφαλιστεί η συμβατότητα με το RNN.
* Επίσης, η χωρητικότητα του σακιδίου μετατρέπεται σε tensor (**capacity**\_**tensor**) για να χρησιμοποιηθεί στη συνέχεια από το μοντέλο.



5.34.Σιτγμιότυπο κώδικα train\_rnn (7)

**Πρόβλεψη μέσω του Εκπαιδευμένου Μοντέλου**:

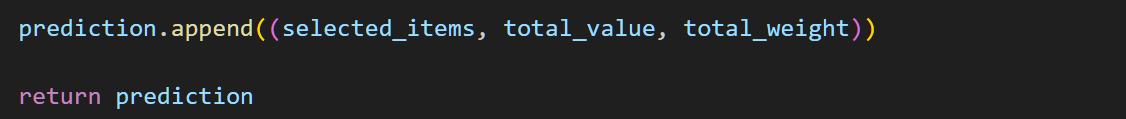
* Το προετοιμασμένο tensor εισόδου προωθείται μέσω του εκπαιδευμένου RNN για να παραχθεί μια πρόβλεψη (**output**). Η πρόβλεψη αυτή αντιπροσωπεύει την πιθανότητα επιλογής για κάθε αντικείμενο.



5.35.Σιτγμιότυπο κώδικα train\_rnn (7)

**Επιλογή Αντικειμένων και Υπολογισμός Αποτελεσμάτων**:

* Οι πιθανότητες επιλογής των αντικειμένων μετατρέπονται σε δυαδικές τιμές (**selected**\_**items**), όπου το κατώφλι επιλογής είναι 0.5. Αν η πιθανότητα υπερβαίνει το 0.5, το αντικείμενο θεωρείται επιλεγμένο.
* Στη συνέχεια, υπολογίζεται η συνολική αξία (**total**\_**value**) και το συνολικό βάρος (**total**\_**weight**) των επιλεγμένων αντικειμένων, συνδυάζοντας τις επιλεγμένες τιμές και τα βάρη αντίστοιχα.



5.36.Σιτγμιότυπο κώδικα train\_rnn (8)

**Επιστροφή του Αποτελέσματος**:

* Η συνάρτηση επιστρέφει μια λίστα (**prediction**) που περιέχει τις επιλεγμένες αντικείμενα, τη συνολική αξία, και το συνολικό βάρος. Αυτά τα αποτελέσματα μπορούν να συγκριθούν με τη βέλτιστη λύση για να αξιολογηθεί η απόδοση του μοντέλου.

**Εφαρμογή του Εκπαιδευμένου RNN**

Η συνάρτηση **solve\_knapsack\_with\_rnn** επιτρέπει την εφαρμογή του εκπαιδευμένου RNN για την επίλυση νέων περιπτώσεων του προβλήματος του σακιδίου. Με αυτή τη συνάρτηση, το μοντέλο μπορεί να προβλέψει ποια αντικείμενα θα πρέπει να επιλεγούν με βάση τις τιμές και τα βάρη τους, διασφαλίζοντας ότι το αποτέλεσμα παραμένει εντός των περιορισμών της χωρητικότητας του σακιδίου.

### Βελτιστοποίηση Υπερπαραμέτρων RNN

Για τη βελτιστοποίηση των υπερπαραμέτρων του μοντέλου RNN, εφαρμόστηκε μια διαδικασία παρόμοια με αυτή που χρησιμοποιήθηκε για το DNN, με την κύρια διαφορά να βρίσκεται στην προσαρμογή των υπερπαραμέτρων που είναι ειδικές για τα επαναλαμβανόμενα νευρωνικά δίκτυα.

Οι βασικές υπερπαράμετροι που εξετάστηκαν για το RNN περιλαμβάνουν:

* Ρυθμός Μάθησης (**lr**)
* Μέγεθος Κρυφών Επιπέδων (**hidden**\_**sizes**)
* Αριθμός Επιπέδων (**num**\_**layers**)
* Αριθμός Εποχών (**epochs**)

**Βέλτιστες Υπερπαράμετροι**

Μετά την αξιολόγηση διαφόρων συνδυασμών των παραπάνω υπερπαραμέτρων, οι βέλτιστες ρυθμίσεις για το RNN εντοπίστηκαν ως εξής:

* Ρυθμός Μάθησης (**lr**): [0.01]
* Μέγεθος Κρυφών Επιπέδων (**hidden**\_**sizes**): [64, 64]
* Αριθμός Επιπέδων (**num**\_**layers**): [1]
* Αριθμός Εποχών (**epochs**): [1000]

Η διαδικασία βελτιστοποίησης υπερπαραμέτρων για το RNN επικεντρώθηκε στις ιδιαιτερότητες των επαναλαμβανόμενων νευρωνικών δικτύων, όπως η χρήση πολλαπλών επιπέδων. Το αποτέλεσμα ήταν η επιλογή μιας ρυθμισμένης διαμόρφωσης που παρέχει τη βέλτιστη απόδοση για το πρόβλημα του σακιδίου. Ο πλήρης κώδικας για τη διαδικασία βελτιστοποίησης υπερπαραμέτρων παρέχεται στο Παράρτημα Β. Αυτό περιλαμβάνει την υλοποίηση του **grid** **search**, της εκπαίδευσης του μοντέλου και των λειτουργιών αξιολόγησης.

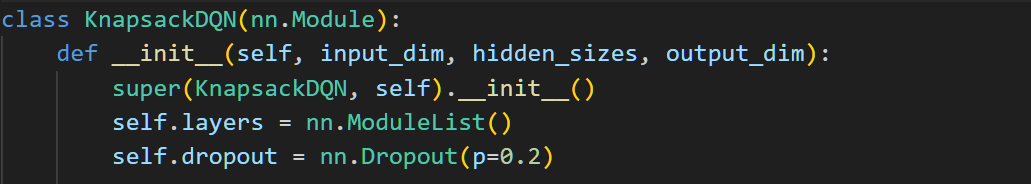
## Μοντέλο Deep Q-Learning (DQN)

### Δημιουργία και Αρχιτεκτονική του Μοντέλου DQN

Στην ενότητα αυτή, θα αναλυθεί το περιβάλλον του σακιδίου όπως αυτό χρησιμοποιείται στο πλαίσιο του Μοντέλου Deep Q-Learning (DQN). Το περιβάλλον του σακιδίου διαδραματίζει βασικό ρόλο στην εκπαίδευση του DQN, καθώς παρέχει το πλαίσιο μέσα στο οποίο ο πράκτορας (**agent**) αλληλεπιδρά για να μάθει πώς να επιλέγει τα αντικείμενα που θα τοποθετήσει στο σακίδιο.

**Περιγραφή της Κλάσης KnapsackDQN**

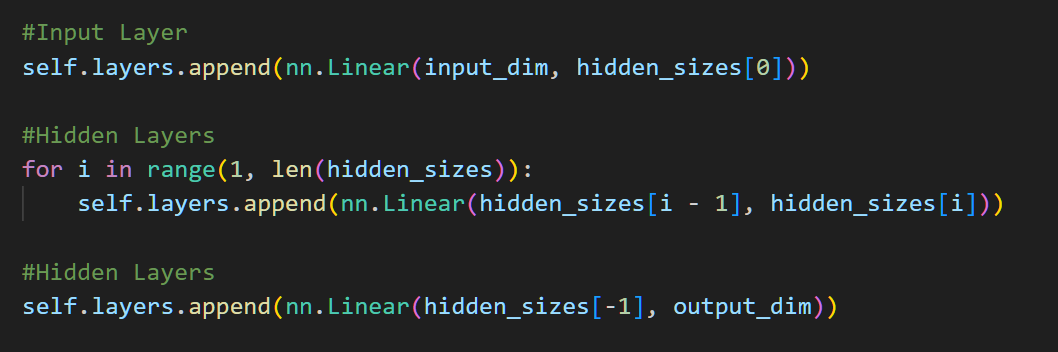
Η κλάση **KnapsackDQN** υλοποιεί το νευρωνικό δίκτυο που χρησιμοποιείται από τον πράκτορα DQN. Το δίκτυο αυτό λαμβάνει ως είσοδο την τρέχουσα κατάσταση του περιβάλλοντος και επιστρέφει μια εκτίμηση της αξίας (**Q-value**) για κάθε πιθανή δράση (**action**). Η δομή του δικτύου περιλαμβάνει πλήρως συνδεδεμένα επίπεδα με ενεργοποιήσεις **ReLU** και dropout για αποτροπή του overfitting.



5.37.Σιτγμιότυπο κλάσης KnapsackDQN

**Αρχικοποίηση του Μοντέλου**:

* Η κλάση **KnapsackDQN** κληρονομεί από την κλάση **nn.Module** του PyTorch.
* Κατά την αρχικοποίηση, το μοντέλο δημιουργεί μια λίστα από επίπεδα (**layers**), που θα χρησιμοποιηθούν για την κατασκευή του δικτύου. Ένα dropout επίπεδο προστίθεται με πιθανότητα 0.2 για την αποφυγή υπερβολικής προσαρμογής του μοντέλου στα δεδομένα εκπαίδευσης.



5.38.Σιτγμιότυπο κλάσης KnapsackDQN (2)

**Δημιουργία των Επιπέδων του Δικτύου**:

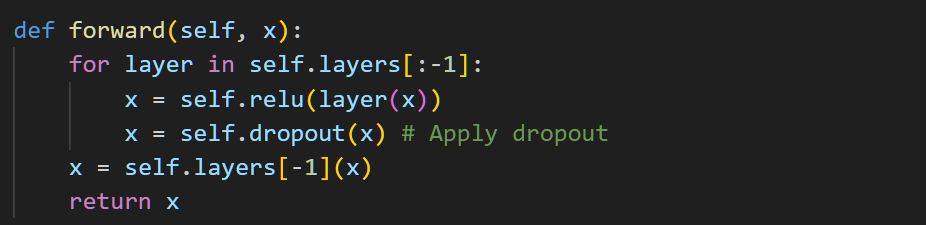
* Το δίκτυο αποτελείται από ένα επίπεδο εισόδου (**input** **layer**), πολλαπλά κρυφά επίπεδα (**hidden** **layers**), και ένα επίπεδο εξόδου (**output** **layer**).
* Το επίπεδο εισόδου λαμβάνει το διάνυσμα εισόδου, το οποίο περιέχει την τρέχουσα κατάσταση του περιβάλλοντος, και το μετασχηματίζει μέσω των κρυφών επιπέδων σε ένα διάνυσμα εξόδου. Η έξοδος αυτή αντιπροσωπεύει τις Q-values, που αντιστοιχούν στις διαφορετικές πιθανές δράσεις που μπορεί να λάβει ο πράκτορας.



5.39.Σιτγμιότυπο κλάσης KnapsackDQN (3)

**Συνάρτηση Ενεργοποίησης ReLU**:

* Η συνάρτηση ενεργοποίησης ReLU χρησιμοποιείται σε κάθε κρυφό επίπεδο για την προσθήκη μη γραμμικότητας στο δίκτυο. Αυτό επιτρέπει στο δίκτυο να μάθει πιο σύνθετες αναπαραστάσεις των δεδομένων.



5.40.Σιτγμιότυπο κλάσης KnapsackDQN (4)

**Forward Μέθοδος (Προώθηση)**:

* Η μέθοδος **forward** καθορίζει τη διαδρομή που ακολουθούν τα δεδομένα μέσα από το δίκτυο. Κάθε επίπεδο εφαρμόζεται στη σειρά, αρχικά με ενεργοποίηση ReLU και στη συνέχεια με dropout για τα κρυφά επίπεδα.
* Το τελικό επίπεδο επιστρέφει τις Q-values για κάθε πιθανή δράση, οι οποίες χρησιμοποιούνται από τον πράκτορα για να επιλέξει την καλύτερη δράση στην τρέχουσα κατάσταση.

**Συνολική Αρχιτεκτονική**

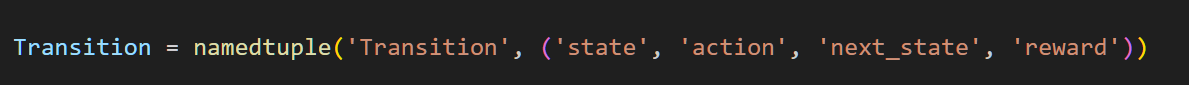
Η κλάση **KnapsackDQN** αποτελεί τον πυρήνα του νευρωνικού δικτύου που χρησιμοποιείται στο DQN. Το δίκτυο είναι υπεύθυνο για την εκτίμηση των Q-values, που επιτρέπουν στον πράκτορα να επιλέξει τις κατάλληλες δράσεις με βάση την τρέχουσα κατάσταση του περιβάλλοντος. Αυτή η διαδικασία είναι θεμελιώδης για την εκπαίδευση και την απόδοση του DQN στο πρόβλημα του σακιδίου.

### Διαχείριση Εμπειριών (Replay Memory)

Η διαχείριση εμπειριών είναι ένα κρίσιμο συστατικό στην εκπαίδευση των μοντέλων Deep Q-Learning (DQN). Η διαδικασία αυτή περιλαμβάνει την αποθήκευση εμπειριών που αποκτά ο πράκτορας κατά τη διάρκεια της αλληλεπίδρασής του με το περιβάλλον, καθώς και την επανειλημμένη χρήση αυτών των εμπειριών για τη βελτίωση της απόδοσης του μοντέλου. Στην ενότητα αυτή, θα εξετασθεί η υλοποίηση της **ReplayMemory** και η χρήση της στο DQN.

**Περιγραφή της Κλάσης ReplayMemory**

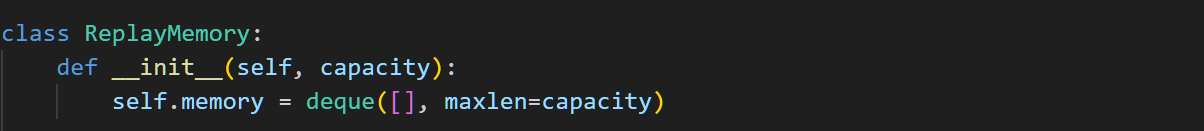
Η κλάση ReplayMemory λειτουργεί ως ένας δακτυλιοειδής buffer που αποθηκεύει τις εμπειρίες του πράκτορα κατά τη διάρκεια της εκπαίδευσης. Κάθε εμπειρία αποτελείται από την τρέχουσα κατάσταση (**state**), την ενέργεια που επιλέχθηκε (**action**), την επόμενη κατάσταση (**next**\_**state**), και την ανταμοιβή (**reward**).



5.41.Σιτγμιότυπο κώδικα Transition namedtuple

**Ορισμός της Δομής Transition**:

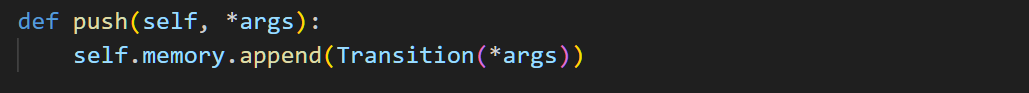
* Η δομή **Transition** είναι μια ονομαστική συλλογή (**namedtuple**) που χρησιμοποιείται για να αποθηκεύσει κάθε εμπειρία με τη μορφή ενός **tuple** που περιέχει την τρέχουσα κατάσταση, την επιλεγμένη ενέργεια, την επόμενη κατάσταση και την ανταμοιβή.



5.42.Σιτγμιότυπο κλάσης ReplayMemory

**Αρχικοποίηση του Replay Memory**:

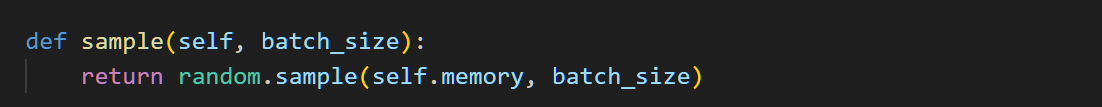
* Η **ReplayMemory** αρχικοποιείται με μια συγκεκριμένη χωρητικότητα (**capacity**), που ορίζει τον μέγιστο αριθμό εμπειριών που μπορεί να αποθηκεύσει. Η δομή αυτή υλοποιείται με τη χρήση του **deque**, το οποίο είναι ένα double-ended queue, που επιτρέπει την ταχεία εισαγωγή και διαγραφή στοιχείων.
* Η μέγιστη χωρητικότητα εξασφαλίζει ότι το buffer παραμένει περιορισμένο σε μέγεθος, αποτρέποντας την υπερβολική κατανάλωση μνήμης.



5.43.Σιτγμιότυπο κλάσης ReplayMemory (2)

**Αποθήκευση Εμπειριών**:

* Η μέθοδος **push** προσθέτει μια νέα εμπειρία στη μνήμη. Η εμπειρία αποθηκεύεται με τη χρήση της δομής **Transition**, που περιλαμβάνει την κατάσταση, την ενέργεια, την επόμενη κατάσταση και την ανταμοιβή. Αν το buffer είναι γεμάτο, η παλαιότερη εμπειρία διαγράφεται αυτόματα για να χωρέσει η νέα.



5.44.Σιτγμιότυπο κλάσης ReplayMemory (3)

**Δειγματοληψία Εμπειριών**:

* Η μέθοδος **sample** επιλέγει τυχαία έναν αριθμό εμπειριών από το buffer, ίσο με το μέγεθος της παρτίδας (**batch**\_**size**). Αυτές οι εμπειρίες χρησιμοποιούνται για την ενημέρωση του DQN κατά τη διάρκεια της εκπαίδευσης, επιτρέποντας στο μοντέλο να μάθει από ένα ευρύ φάσμα εμπειριών.



5.45.Σιτγμιότυπο κλάσης ReplayMemory (4)

**Έλεγχος Μεγέθους**:

* Η μέθοδος **\_\_len\_\_** επιστρέφει το τρέχον μέγεθος του buffer, δηλαδή τον αριθμό των αποθηκευμένων εμπειριών. Αυτό μπορεί να χρησιμοποιηθεί για να διασφαλιστεί ότι υπάρχει επαρκής αριθμός εμπειριών για τη δειγματοληψία κατά τη διάρκεια της εκπαίδευσης.

**Σημασία του Replay Memory στο DQN**

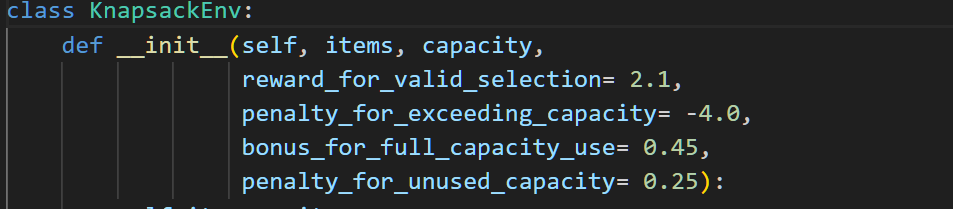
Το **ReplayMemory** είναι ζωτικής σημασίας για την εκπαίδευση του DQN. Η δυνατότητα δειγματοληψίας από μια μεγάλη ποικιλία εμπειριών βοηθά στη σταθερότητα της εκπαίδευσης και αποτρέπει την υπερεξάρτηση από πρόσφατες εμπειρίες. Αυτό καθιστά το DQN πιο ανθεκτικό και ικανό να μάθει από παλαιότερες εμπειρίες που μπορεί να είναι κρίσιμες για την επίλυση του προβλήματος.

### Περιβάλλον του Σακιδίου (Knapsack Environment)

Η κλάση KnapsackEnv είναι η προσομοίωση του περιβάλλοντος στο οποίο λειτουργεί ο πράκτορας DQN. Αποτελεί το χώρο στο οποίο ο πράκτορας αλληλεπιδρά, λαμβάνει αποφάσεις, και επιστρέφει επιβραβεύσεις (**rewards**) με βάση τις ενέργειές του. Ας δούμε αναλυτικά τη δομή και τις λειτουργίες της κλάσης.

**Κατασκευή της Κλάσης KnapsackEnv**

Η κλάση KnapsackEnv έχει έναν αριθμό παραμέτρων που ρυθμίζουν τη συμπεριφορά της:



5.46 Στιγμιότυπο κλάσης KnapsackEnv

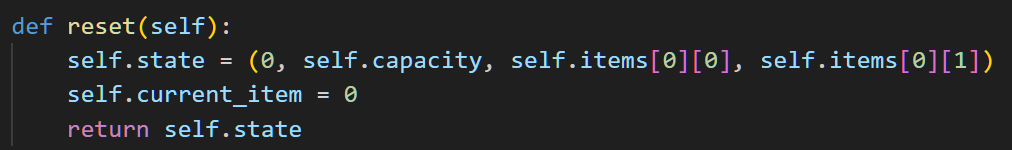
Αυτές οι παράμετροι περιλαμβάνουν τα αντικείμενα που πρέπει να επιλεγούν, τη χωρητικότητα του σακιδίου, καθώς και διάφορους συντελεστές επιβράβευσης και ποινής που χρησιμοποιούνται στη διαδικασία επιλογής. Ορίζονται τέσσερις βασικές παράμετροι επιβράβευσης και ποινής:

* **reward\_for\_valid\_selection**: Επιβράβευση για την επιλογή ενός έγκυρου αντικειμένου (αυτό που χωράει στο σακίδιο).
* **penalty\_for\_exceeding\_capacity**: Ποινή για την επιλογή ενός αντικειμένου που υπερβαίνει τη χωρητικότητα του σακιδίου.
* **bonus\_for\_full\_capacity\_use**: Επιπλέον επιβράβευση αν το σακίδιο γεμίσει ακριβώς.
* **penalty\_for\_unused\_capacity**: Ποινή για την αχρησιμοποίητη χωρητικότητα στο τέλος της επιλογής.

Η μέθοδος **\_\_init\_\_** αρχικοποιεί την κατάσταση του περιβάλλοντος και καλεί τη μέθοδο **reset** για να θέσει την αρχική κατάσταση.

**Μέθοδος reset**

Η μέθοδος **reset** επαναφέρει το περιβάλλον στην αρχική του κατάσταση, με μηδενική αξία, πλήρη χωρητικότητα και το πρώτο αντικείμενο προς αξιολόγηση:



5.47 Στιγμιότυπο κλάσης KnapsackEnv (2)

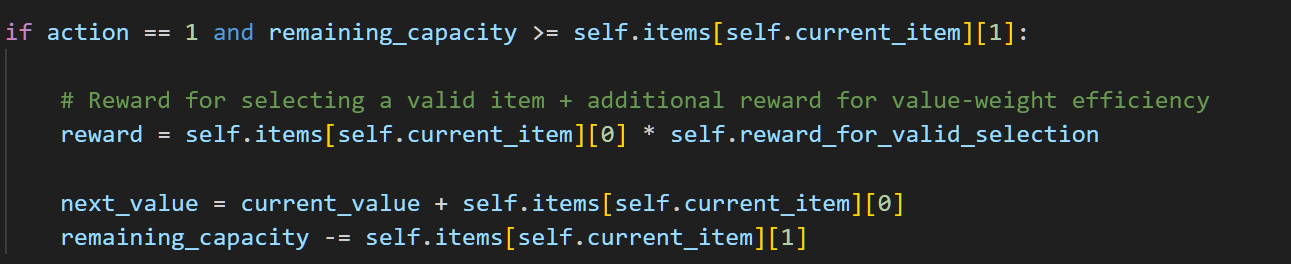
Η μέθοδος **reset** αρχικοποιεί την κατάσταση του περιβάλλοντος, επαναφέροντας το σακίδιο στην αρχική του κατάσταση, δηλαδή με πλήρη χωρητικότητα και χωρίς επιλεγμένα αντικείμενα. Η τρέχουσα κατάσταση επιστρέφεται και αποτελείται από το αρχικό σύνολο τιμών, την πλήρη χωρητικότητα και τα χαρακτηριστικά του πρώτου αντικειμένου. Με αυτόν τον τρόπο, ο πράκτορας ξεκινά πάντα από την ίδια αφετηρία.

**Μέθοδος step**

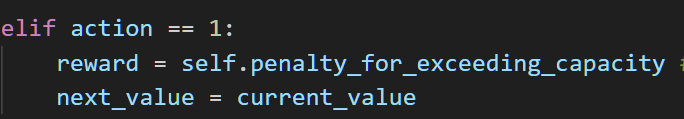
Η μέθοδος **step** είναι υπεύθυνη για την εκτέλεση μιας ενέργειας από τον πράκτορα και την ενημέρωση της κατάστασης του περιβάλλοντος. Αυτή η μέθοδος έχει πιο περίπλοκη δομή και είναι καίριας σημασίας για την αλληλεπίδραση του πράκτορα με το περιβάλλον.

* **Έλεγχος Ενέργειας και Υπολογισμός Επιβράβευσης**:

Αν ο πράκτορας επιλέξει να προσθέσει το αντικείμενο στο σακίδιο και η χωρητικότητα επαρκεί, προστίθεται η αξία του αντικειμένου και αφαιρείται το βάρος του από την υπόλοιπη χωρητικότητα. Ο πράκτορας λαμβάνει επιβράβευση με βάση την αξία του αντικειμένου:



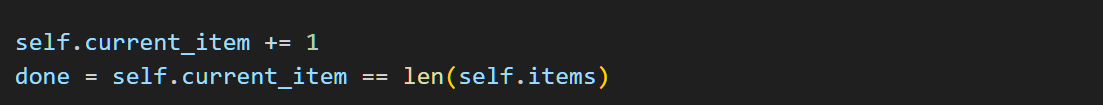
5.48 Στιγμιότυπο κλάσης KnapsackEnv (3)

Αν η χωρητικότητα δεν επαρκεί, επιβάλλεται ποινή και δεν γίνεται καμία αλλαγή στην αξία: 

5.49 Στιγμιότυπο κλάσης KnapsackEnv (4)

* **Ενημέρωση Κατάστασης**:

Η κατάσταση του περιβάλλοντος ενημερώνεται με την επόμενη κατάσταση:

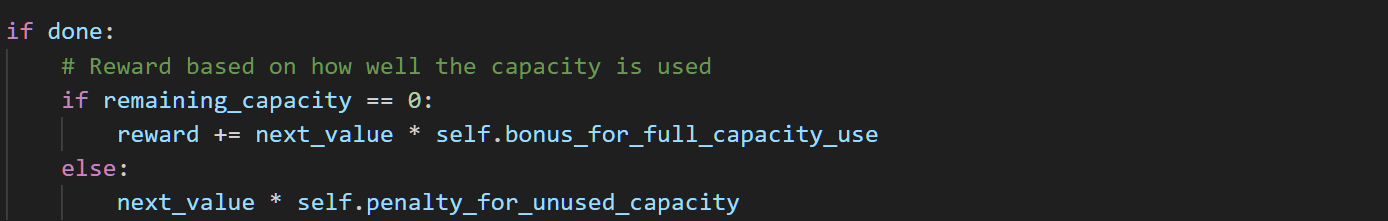


5.50 Στιγμιότυπο κλάσης KnapsackEnv (5)

Αν έχουν αξιολογηθεί όλα τα αντικείμενα, η επιλογή ολοκληρώνεται.

* **Πρόσθετες Επιβραβεύσεις ή Ποινές**:

Στο τέλος της επιλογής, ο πράκτορας λαμβάνει επιπλέον επιβράβευση αν έχει χρησιμοποιήσει πλήρως τη χωρητικότητα του σακιδίου ή ποινή αν έχει απομείνει αχρησιμοποίητη χωρητικότητα:



5.51 Στιγμιότυπο κλάσης KnapsackEnv (6)

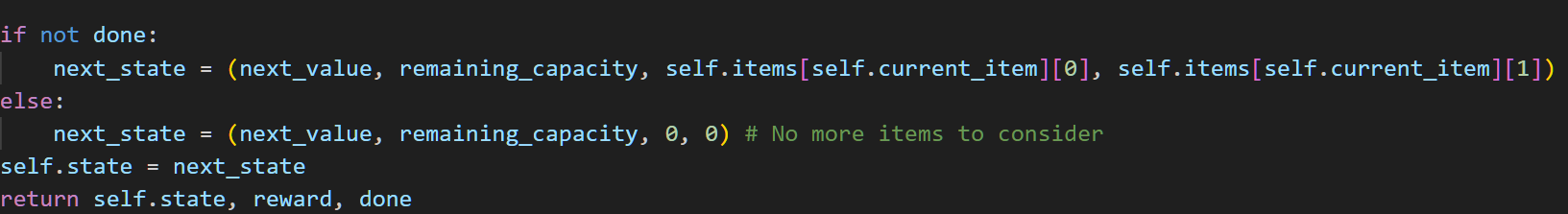
Τέλος, η επιβράβευση κανονικοποιείται:



5.52 Στιγμιότυπο κλάσης KnapsackEnv (7)

* **Επιστροφή Αποτελεσμάτων**:

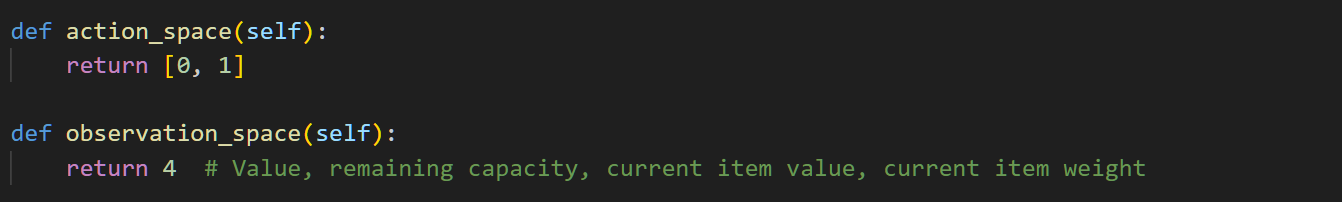
Η μέθοδος επιστρέφει την επόμενη κατάσταση του περιβάλλοντος, την επιβράβευση που κέρδισε ο πράκτορας και μια ένδειξη για το αν η επιλογή έχει ολοκληρωθεί:



5.53 Στιγμιότυπο κλάσης KnapsackEnv (8)

**Μέθοδοι action\_space και observation\_space**

Η κλάση περιλαμβάνει επίσης δύο βοηθητικές μεθόδους που επιστρέφουν τον χώρο των ενεργειών (επιλογή ή μη επιλογή αντικειμένου) και τον χώρο των παρατηρήσεων (κατάσταση του σακιδίου):



5.54 Στιγμιότυπο κλάσης KnapsackEnv (9)

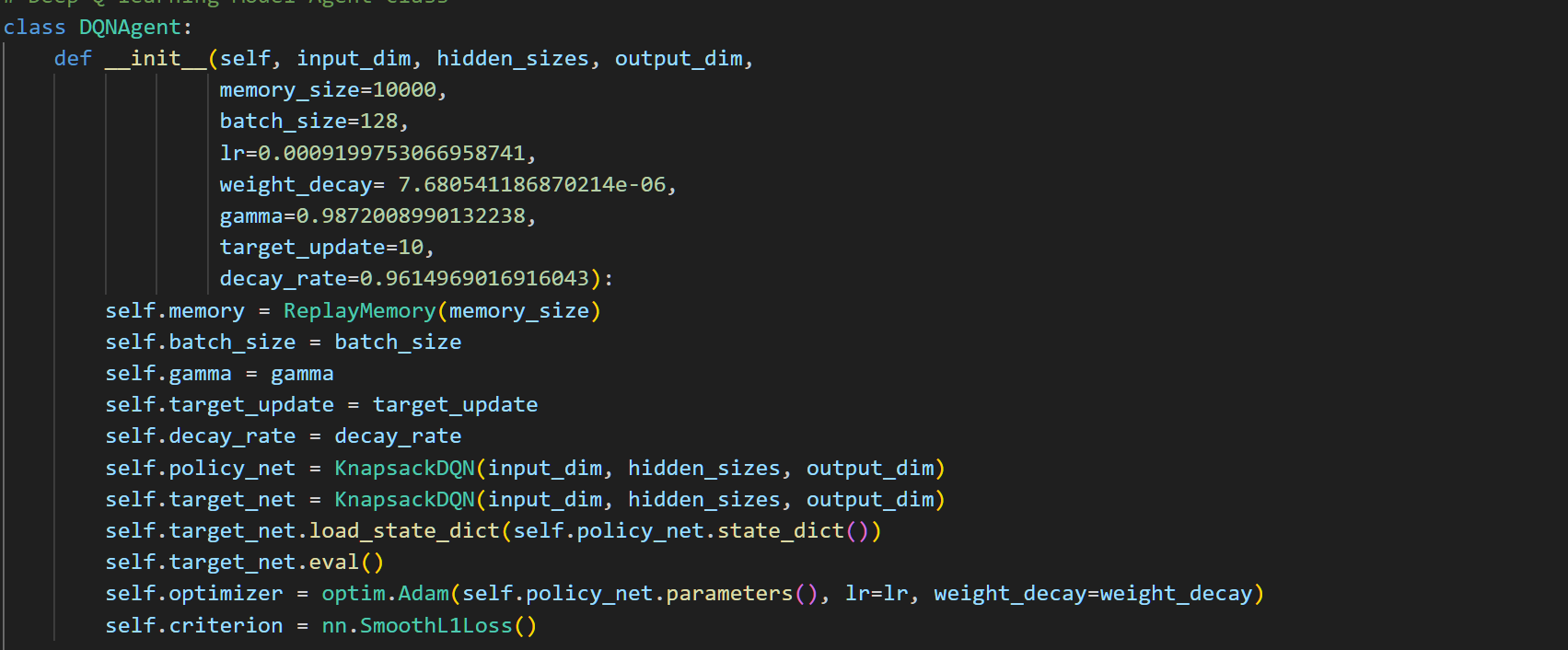
Η μέθοδος **action**\_**space** απλώς επιστρέφει τις δύο πιθανές ενέργειες (να προστεθεί ή όχι το αντικείμενο στο σακίδιο), ενώ η **observation**\_**space** επιστρέφει τις τέσσερις παραμέτρους που περιγράφουν την τρέχουσα κατάσταση του σακιδίου.

### Εκπαίδευση και Επίλυση με το Μοντέλο DQN

**Περιγραφή της Κλάσης DQNAgent**

Η κλάση **DQNAgent** είναι υπεύθυνη για τη διαχείριση της διαδικασίας εκμάθησης μέσω Deep Q-Learning. Αυτό περιλαμβάνει την επιλογή δράσεων, τη διαχείριση της μνήμης επανάληψης (replay memory), τη βελτιστοποίηση του νευρωνικού δικτύου, και τελικά την εκπαίδευση του πράκτορα DQN.

**Α. Αρχικοποίηση του Πράκτορα DQN (\_\_init\_\_)**



5.55.Σιτγμιότυπο κλάσης DQNAgent

1. **Μνήμη Εμπειριών (ReplayMemory):**

* Το **ReplayMemory** είναι μια δομή που αποθηκεύει τις εμπειρίες του πράκτορα κατά τη διάρκεια της εκπαίδευσης. Αυτό είναι σημαντικό για το DQN, καθώς οι εμπειρίες μπορούν να αναπαραχθούν πολλές φορές κατά τη διάρκεια της εκπαίδευσης, βελτιώνοντας τη γενίκευση και τη σταθερότητα του μοντέλου. Η παράμετρος **memory\_size** καθορίζει τη μέγιστη χωρητικότητα της μνήμης αυτής.

1. **Παράμετροι Εκπαίδευσης:**

* **batch\_size**: Το μέγεθος της παρτίδας που θα χρησιμοποιείται κατά τη βελτιστοποίηση του μοντέλου.
* **gamma**: Ο συντελεστής έκπτωσης (discount factor), που καθορίζει τη σημασία των μελλοντικών ανταμοιβών. Η τιμή του κοντά στο 1 σημαίνει ότι οι μελλοντικές ανταμοιβές έχουν μεγάλη σημασία.
* **target\_update**: Το πλήθος των επεισοδίων μετά τα οποία ενημερώνεται το **target\_net** με τα βάρη του **policy\_net**.
* **decay\_rate**: Ο ρυθμός μείωσης του **epsilon**, που καθορίζει πόσο γρήγορα μειώνεται η πιθανότητα εξερεύνησης (exploration).

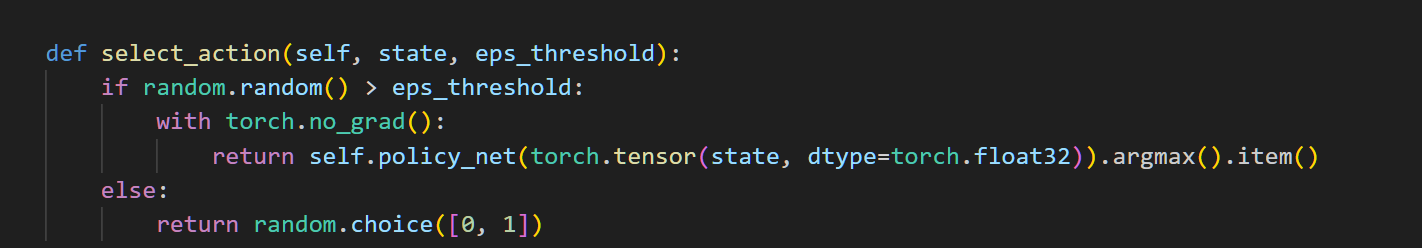
1. **Νευρωνικά Δίκτυα:**

* Το **policy\_net** είναι το κύριο νευρωνικό δίκτυο που χρησιμοποιεί ο πράκτορας για να αποφασίσει τις ενέργειες που θα ακολουθήσει. Αποτελείται από τις δομές που ορίζονται στην κλάση **KnapsackDQN**.
* Το **target\_net** είναι ένα αντίγραφο του **policy\_net** που χρησιμοποιείται για την παροχή σταθερών εκτιμήσεων των Q-values. Αυτό το δίκτυο ενημερώνεται λιγότερο συχνά για να αποφεύγονται οι ταλαντώσεις κατά την εκπαίδευση.

1. **Βελτιστοποιητής και Συνάρτηση Απώλειας:**

* Ο βελτιστοποιητής Adam χρησιμοποιείται για να ενημερώνει τα βάρη του **policy\_net** κατά την εκπαίδευση. Ο ρυθμός μάθησης (**lr**) και η παράμετρος **weight\_decay** (η οποία προσθέτει L2 κανονικοποίηση) καθορίζουν την ταχύτητα και τη σταθερότητα της εκπαίδευσης.
* Η συνάρτηση απώλειας είναι η Smooth L1 Loss, η οποία είναι πιο ανθεκτική σε ακραίες τιμές από τη Mean Squared Error και συχνά χρησιμοποιείται για την εκπαίδευση δικτύων που προβλέπουν συνεχείς τιμές.

**Β. Επιλογή Δράσης (select\_action)**

****

**5.56**.Σιτγμιότυπο κλάσης DQNAgent (2)

1. **Στρατηγική Εξερεύνησης-Εκμετάλλευσης:**

* Η συνάρτηση **select\_action** είναι υπεύθυνη για την επιλογή της επόμενης δράσης του πράκτορα. Χρησιμοποιεί μια στρατηγική ε-greedy, που σημαίνει ότι με μια πιθανότητα **eps\_threshold**, ο πράκτορας θα επιλέξει μια τυχαία δράση (εξερεύνηση), ενώ με την υπόλοιπη πιθανότητα θα επιλέξει τη δράση που μεγιστοποιεί την Q-value που προβλέπει το δίκτυο (εκμετάλλευση).

1. **Εκμετάλλευση (exploitation):**

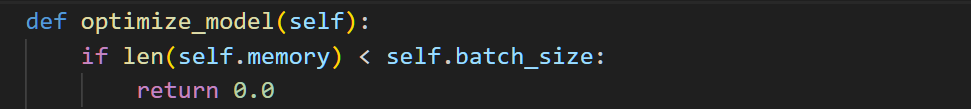
* Όταν η τυχαία τιμή είναι μεγαλύτερη από το **eps\_threshold**, ο πράκτορας εκμεταλλεύεται τις γνώσεις που έχει αποκτήσει μέχρι τώρα. Αυτό σημαίνει ότι υπολογίζει τις Q-values για την τρέχουσα κατάσταση χρησιμοποιώντας το **policy\_net** και επιλέγει τη δράση με την υψηλότερη Q-value.

1. **Εξερεύνηση (exploration):**

* Όταν η τυχαία τιμή είναι μικρότερη από το **eps\_threshold**, ο πράκτορας εξερευνά το περιβάλλον επιλέγοντας τυχαία μια δράση. Αυτό βοηθά τον πράκτορα να μην παγιδευτεί σε τοπικά μέγιστα και να ανακαλύψει καλύτερες στρατηγικές με την πάροδο του χρόνου.

**Γ. Βελτιστοποίηση Μοντέλου (optimize\_model)**

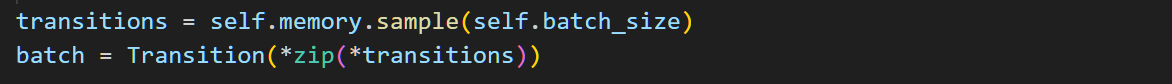
* 1. **Έλεγχος Διαθέσιμων Δεδομένων στη Μνήμη**



5.57.Σιτγμιότυπο κλάσης DQNAgent (3)

Η συνάρτηση ξεκινά με έναν έλεγχο για να διαπιστώσει αν υπάρχουν αρκετά δείγματα στη μνήμη επαναληψιμότητας (**Replay Memory**). Αν ο αριθμός των δειγμάτων είναι μικρότερος από το μέγεθος του batch (**batch\_size**), τότε η συνάρτηση επιστρέφει **0.0**, υποδηλώνοντας ότι δεν μπορεί να γίνει εκπαίδευση σε αυτό το στάδιο.

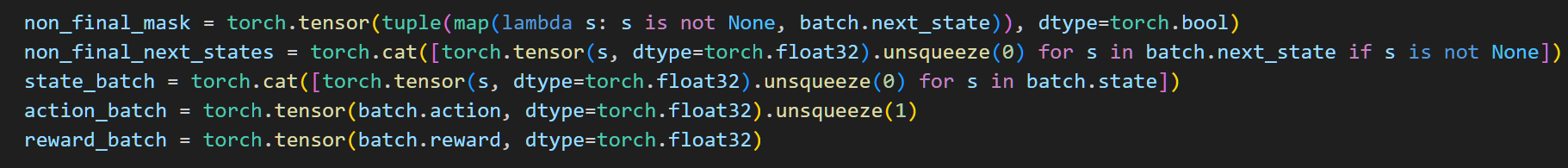
* 1. **Δειγματοληψία Εμπειριών από τη Μνήμη**

****

5.58.Σιτγμιότυπο κλάσης DQNAgent (5)

Αν υπάρχουν αρκετά δείγματα, η συνάρτηση επιλέγει τυχαία ένα **batch** δεδομένων από τη μνήμη επαναληψιμότητας χρησιμοποιώντας τη μέθοδο **sample**. Τα δεδομένα αυτά περιλαμβάνουν καταστάσεις (**state**), ενέργειες (**action**), επόμενες καταστάσεις (**next\_state**), και ανταμοιβές (**reward**). Στη συνέχεια, τα δεδομένα ομαδοποιούνται σε ένα batch με τη βοήθεια του **namedtuple** **Transition**.

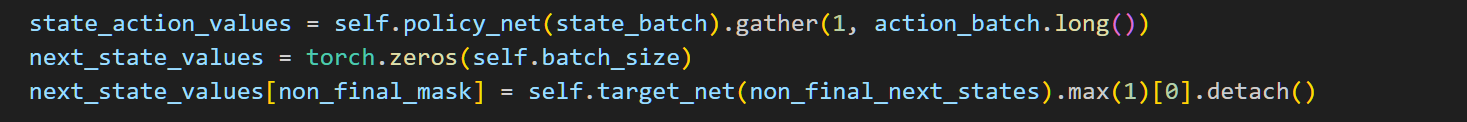
* 1. **Προετοιμασία των Δεδομένων**



5.59.Σιτγμιότυπο κλάσης DQNAgent (6)

Ακολουθεί η επεξεργασία των δεδομένων του **batch**:

* **non\_final\_mask**: Δημιουργεί μια μάσκα που υποδεικνύει ποιες επόμενες καταστάσεις (**next\_state**) είναι έγκυρες (δηλαδή δεν είναι None).
* **non\_final\_next\_states**: Δημιουργεί ένα **tensor** που περιέχει τις έγκυρες επόμενες καταστάσεις.
* **state\_batch**: Δημιουργεί ένα tensor που περιέχει τις αρχικές καταστάσεις του batch.
* **action\_batch**: Δημιουργεί ένα tensor που περιέχει τις ενέργειες που έγιναν σε κάθε κατάσταση του batch.
* **reward\_batch**: Δημιουργεί ένα tensor που περιέχει τις ανταμοιβές για κάθε κατάσταση του batch.
  1. **Υπολογισμός Q-values**



**5.60** Στιγμιότυπο κλάσης DQNAgent (7)

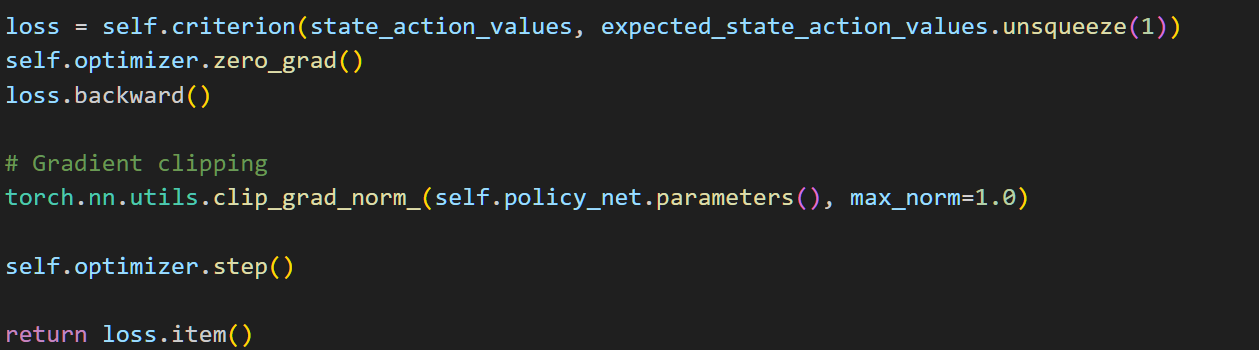
**Σε αυτό το στάδιο, υπολογίζονται οι τιμές των Q-functions:**

* **state\_action\_values**: Υπολογίζει τις Q-values για το τρέχον **state\_batch** από το **policy\_net**, χρησιμοποιώντας τις ενέργειες του **action\_batch**.
* **next\_state\_values**: Δημιουργεί ένα **tensor** με μηδενικά για την αποθήκευση των Q-values των επόμενων καταστάσεων. Για τις έγκυρες επόμενες καταστάσεις, οι Q-values υπολογίζονται από το **target\_net** με την επιλογή της μέγιστης τιμής για κάθε κατάσταση.
  1. **Υπολογισμός της Αναμενόμενης Q-value**

****

**5.61**Στιγμιότυπο κλάσης DQNAgent (8)

Οι αναμενόμενες τιμές των Q-functions υπολογίζονται χρησιμοποιώντας την εξίσωση Bellman, η οποία περιλαμβάνει τον συντελεστή έκπτωσης (**gamma**) και τις ανταμοιβές (**reward**\_**batch**).

****

**5.62**Στιγμιότυπο κλάσης DQNAgent (9)

Στη συνέχεια, υπολογίζεται η απώλεια (**loss**) χρησιμοποιώντας τη διαφορά μεταξύ των Q-values που υπολογίστηκαν και των αναμενόμενων Q-values. Ο βρόχος εκπαίδευσης ακολουθεί τα τυπικά βήματα της οπισθοδιάδοσης (backpropagation): μηδενισμός των gradients, υπολογισμός των νέων gradients μέσω της μεθόδου **backward**, και τελικά η βελτιστοποίηση των παραμέτρων του μοντέλου χρησιμοποιώντας την **optimizer**.**step**().

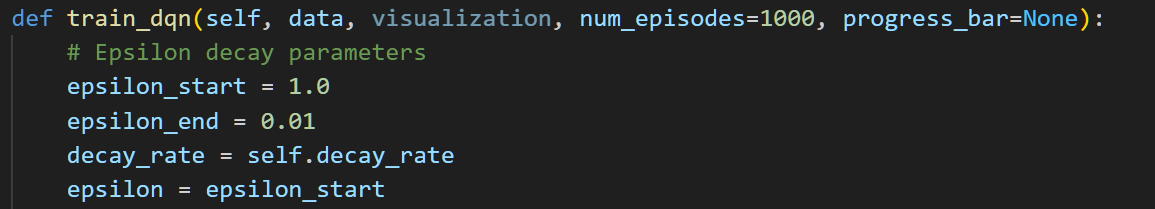
Για να αποφευχθεί το φαινόμενο των **exploding gradients**, εφαρμόζεται **gradient clipping**, το οποίο περιορίζει τη μέγιστη τιμή του gradient στο **max\_norm=1.0**.

Τέλος, η συνάρτηση **optimize\_model** επιστρέφει την τιμή της απώλειας για το συγκεκριμένο βήμα εκπαίδευσης, η οποία μπορεί να χρησιμοποιηθεί για την παρακολούθηση της απόδοσης του μοντέλου κατά τη διάρκεια της εκπαίδευσης.

**Δ. Εκπαίδευση του Πράκτορα DQN (train\_dqn)**

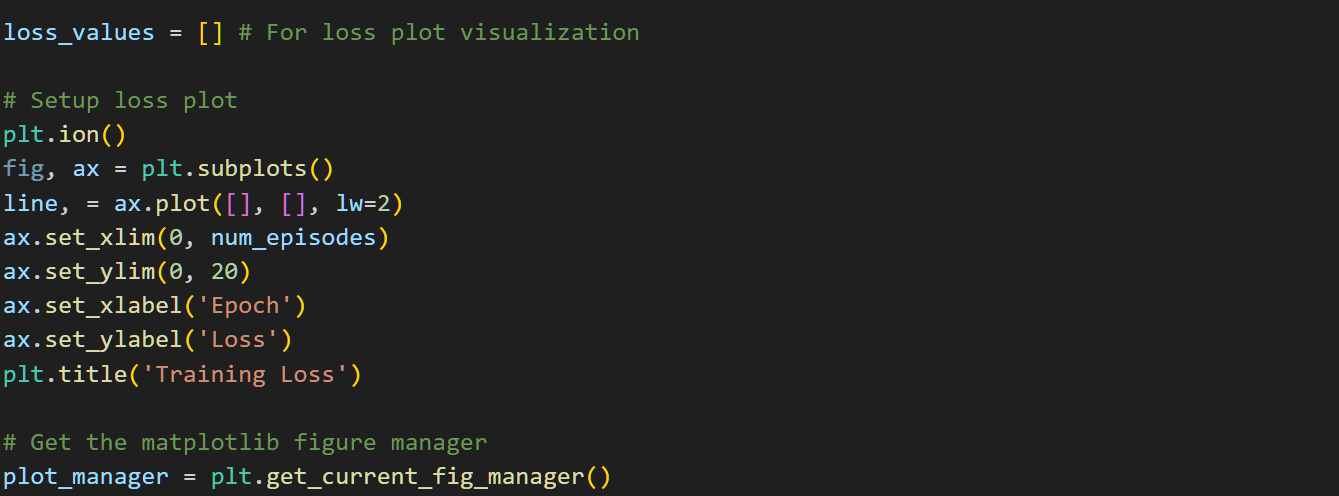
Η διαδικασία εκπαίδευσης του πράκτορα DQN είναι το πιο σημαντικό κομμάτι στην ανάπτυξη ενός πράκτορα που μπορεί να μάθει να επιλύει το πρόβλημα του σακιδίου. Η συνάρτηση **train\_dqn** είναι υπεύθυνη για την εκπαίδευση του πράκτορα μέσω πολλών επεισοδίων, όπου ο πράκτορας αλληλεπιδρά με το περιβάλλον, λαμβάνει ανταμοιβές, και ενημερώνει τις πολιτικές του για να βελτιώσει τη λήψη αποφάσεων.

* 1. **Αρχικοποίηση και Ετοιμασία**

****

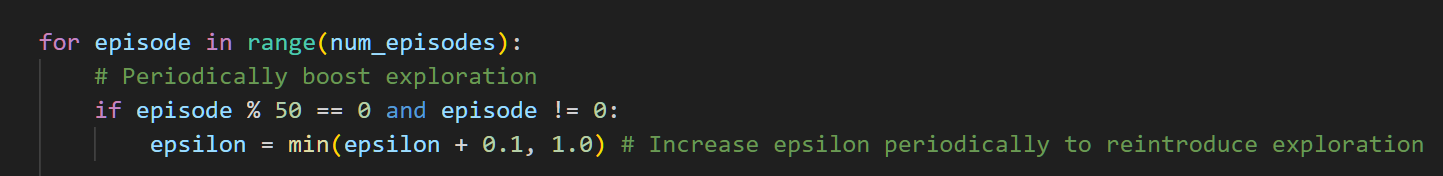
**5.63** Στιγμιότυπο κώδικα train\_dqn

Η συνάρτηση ξεκινά με την αρχικοποίηση των παραμέτρων που σχετίζονται με την παράμετρο **epsilon** (**ε**), η οποία ελέγχει την ισορροπία μεταξύ εξερεύνησης και εκμετάλλευσης κατά τη διάρκεια της εκπαίδευσης. Το **epsilon** αρχίζει από την τιμή **1.0** και σταδιακά μειώνεται προς το **epsilon\_end (0.01)**, ανάλογα με τον **decay**\_**rate**.

* 1. **Προετοιμασία Γραφήματος Απώλειας**

**5.64**Στιγμιότυπο κώδικα train\_dqn (2)

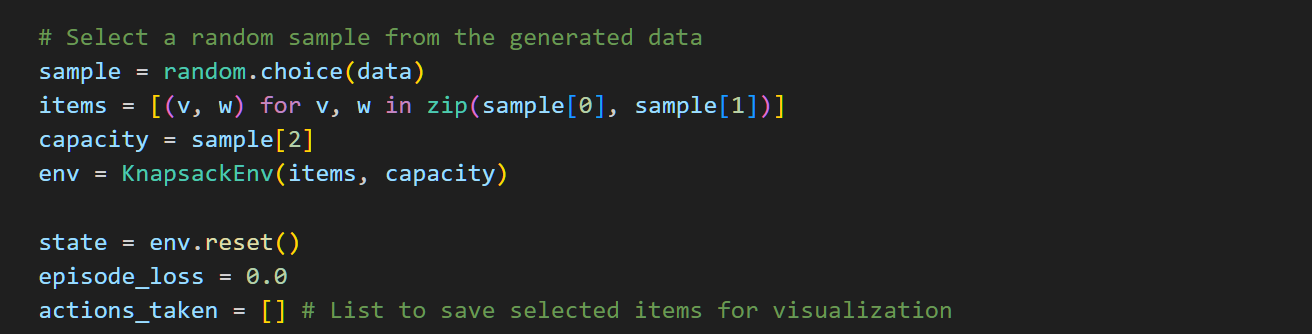
Εδώ, δημιουργείται ένα γράφημα για την παρακολούθηση της απώλειας κατά τη διάρκεια της εκπαίδευσης. Αυτό το γράφημα θα ενημερώνεται δυναμικά καθώς εξελίσσεται η εκπαίδευση, δίνοντας μια οπτική αναπαράσταση της απόδοσης του πράκτορα.

* 1. **Βρόγχος Επεισοδίων**

5.65 Στιγμιότυπο κώδικα train\_dqn (3)

Ο κύριος βρόγχος εκπαίδευσης διατρέχει τα επεισόδια (**num\_episodes**). Κάθε 50 επεισόδια, το **epsilon** αυξάνεται για να επανεισάγει την εξερεύνηση, ώστε να αποφευχθεί το φαινόμενο της σύγκλισης σε τοπικά ελάχιστα.

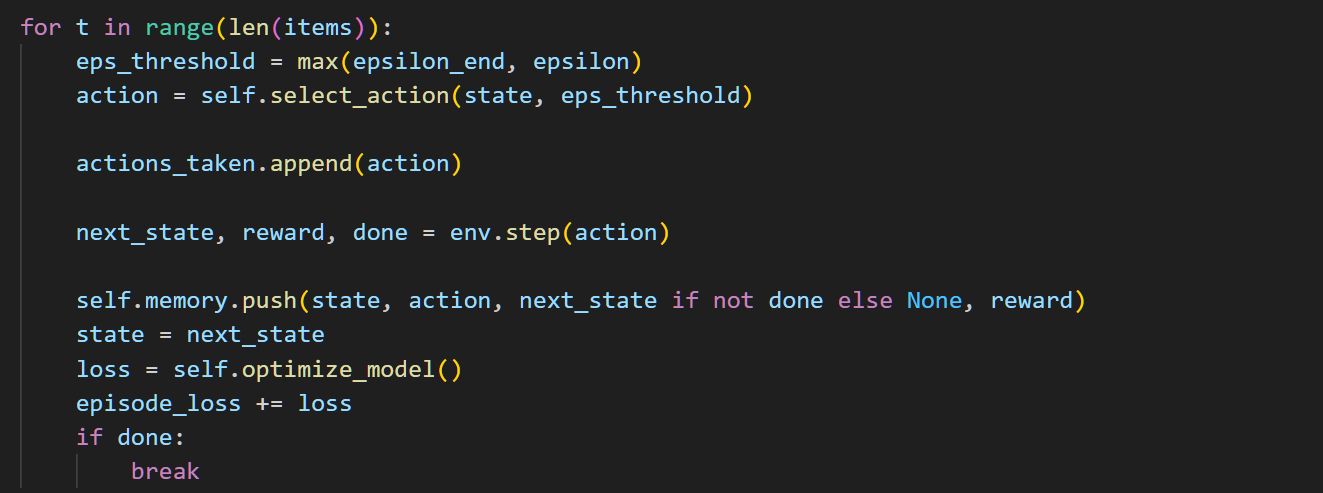
* 1. **Επιλογή Δείγματος και Αρχικοποίηση Περιβάλλοντος**



5.66 Στιγμιότυπο κώδικα train\_dqn (4)

Για κάθε επεισόδιο, επιλέγεται τυχαία ένα δείγμα δεδομένων από το σύνολο των δεδομένων. Το δείγμα περιέχει τα αντικείμενα και την ικανότητα του σακιδίου. Ένα νέο περιβάλλον **KnapsackEnv** δημιουργείται για το επεισόδιο, και ο πράκτορας ξεκινά από την αρχική κατάσταση (**state**).

* 1. **Εκπαίδευση μέσα σε κάθε επεισόδιο**



5.67 Στιγμιότυπο κώδικα train\_dqn (5)

Στον εσωτερικό βρόγχο κάθε επεισοδίου, ο πράκτορας λαμβάνει αποφάσεις με βάση την τρέχουσα κατάσταση και το **epsilon**, που καθορίζει το αν θα γίνει εξερεύνηση ή εκμετάλλευση. Οι αποφάσεις (**actions\_taken**) αποθηκεύονται για την οπτικοποίηση, και η κατάσταση ενημερώνεται μετά από κάθε βήμα.

Οι εμπειρίες του πράκτορα (κατάσταση, ενέργεια, επόμενη κατάσταση, ανταμοιβή) αποθηκεύονται στη μνήμη επαναληψιμότητας (**Replay Memory**). Κατόπιν, καλείται η **optimize\_model** για να εκπαιδεύσει το δίκτυο πολιτικής και να ενημερώσει τα βάρη του. Ο βρόγχος σταματά όταν ο πράκτορας έχει ολοκληρώσει το επεισόδιο (αντικείμενα τελειώνουν ή ικανότητα σακιδίου εξαντληθεί).

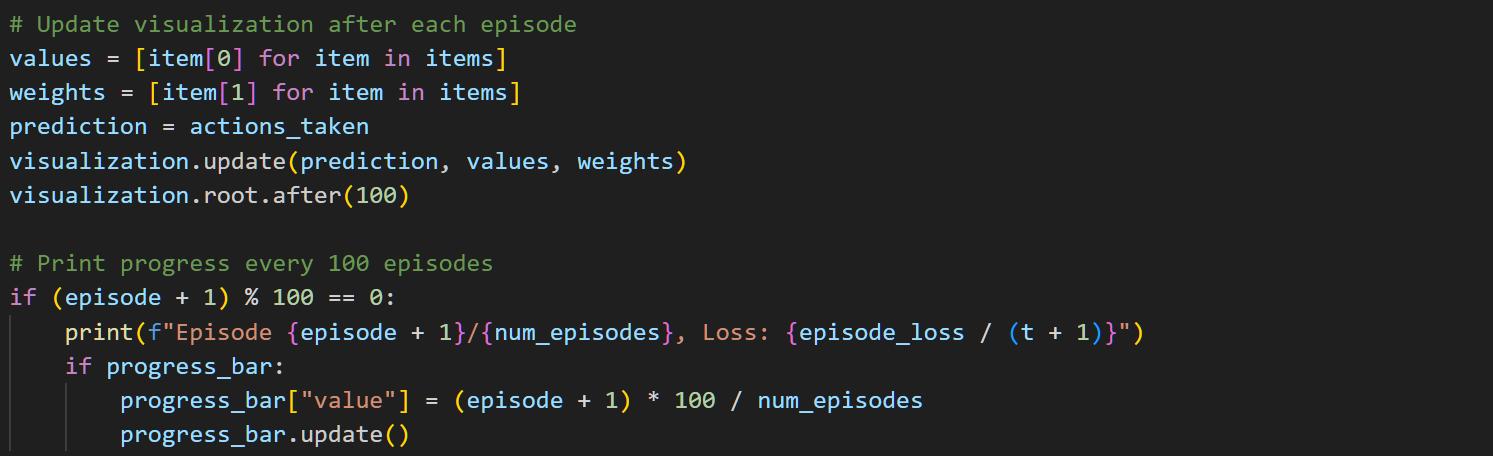
* 1. **Ενημέρωση και Προσαρμογή των Παραμέτρων**



5.68 Στιγμιότυπο κώδικα train\_dqn (6)

Καθώς προχωρούν τα επεισόδια, το δίκτυο στόχου (**target\_net**) ενημερώνεται περιοδικά με τα βάρη του δικτύου πολιτικής (**policy\_net**). Επιπλέον, η παράμετρος **epsilon** μειώνεται για να ενισχύσει την εκμετάλλευση, ενώ η εξερεύνηση περιορίζεται.

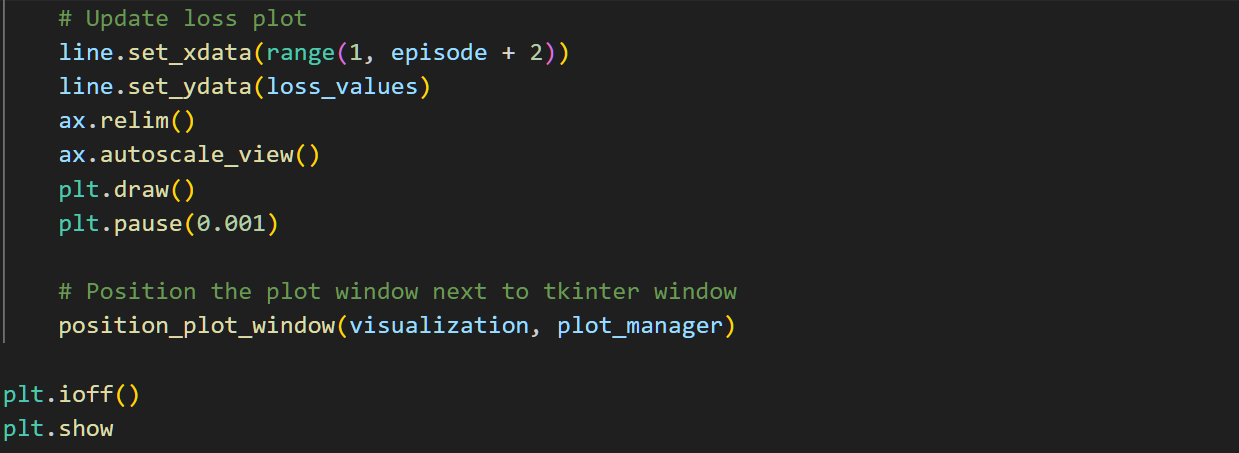
* 1. **Παρακολούθηση της Προόδου Εκπαίδευσης**



5.69 Στιγμιότυπο κώδικα train\_dqn (7)

Τα αποτελέσματα του κάθε επεισοδίου ενημερώνονται μέσω της οπτικοποίησης και της γραμμής προόδου, επιτρέποντας στον χρήστη να παρακολουθεί την εξέλιξη της εκπαίδευσης. Κάθε 100 επεισόδια, η τρέχουσα απώλεια εμφανίζεται στην κονσόλα, δίνοντας μια ένδειξη για την απόδοση του μοντέλου.

* 1. **Ενημέρωση Γραφήματος Απώλειας**

****

**5.70** Στιγμιότυπο κώδικα train\_dqn (8)

Το γράφημα απώλειας ενημερώνεται σε πραγματικό χρόνο, δίνοντας μια οπτική αναπαράσταση της προόδου εκπαίδευσης. Η θέση του παραθύρου του γραφήματος ρυθμίζεται ώστε να εμφανίζεται δίπλα στο παράθυρο της εφαρμογής.



5.71 Στιγμιότυπο κώδικα train\_dqn (9)

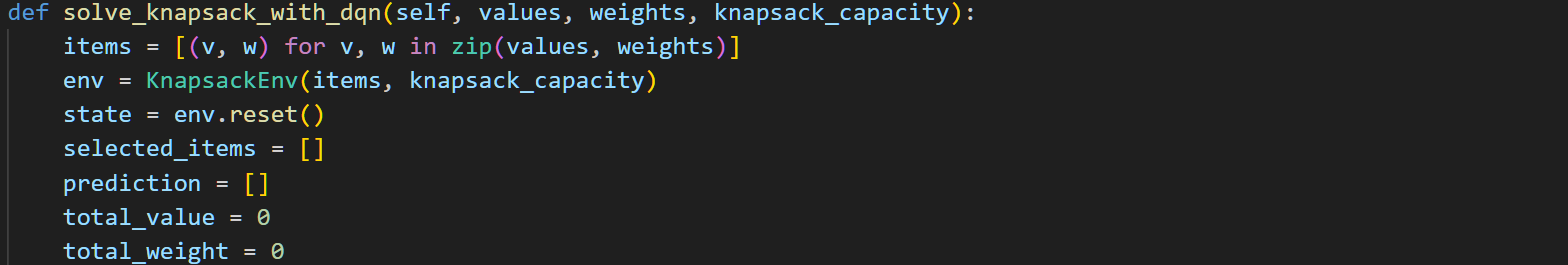
Με την ολοκλήρωση της εκπαίδευσης, η γραμμή προόδου ενημερώνεται στο 100%, σηματοδοτώντας το τέλος της διαδικασίας εκπαίδευσης.

**Συνοψίζοντας**

Η συνάρτηση **train\_dqn** αποτελεί το βασικό εργαλείο για την εκπαίδευση του πράκτορα DQN, συνδυάζοντας εξερεύνηση και εκμετάλλευση μέσω της παραμέτρου **epsilon**, αποθηκεύοντας εμπειρίες και ενημερώνοντας τα βάρη του δικτύου πολιτικής. Η συνεχής ενημέρωση της οπτικοποίησης και της γραμμής προόδου επιτρέπει την άμεση παρακολούθηση της απόδοσης του πράκτορα καθ’ όλη τη διάρκεια της εκπαίδευσης.

**5.5.5 Επίλυση του Προβλήματος με το DQN (solve\_knapsack\_with\_dqn)**

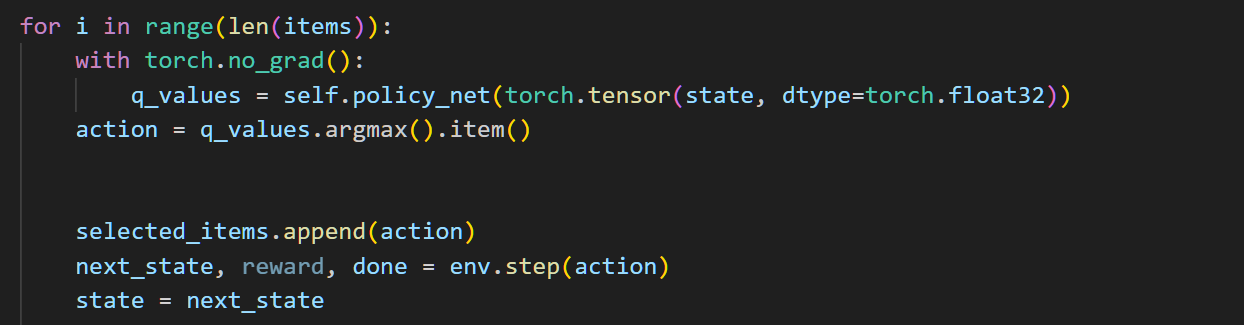
Η συνάρτηση **solve\_knapsack\_with\_dqn** είναι υπεύθυνη για την εφαρμογή του εκπαιδευμένου πράκτορα DQN σε ένα νέο σύνολο δεδομένων, με σκοπό την επίλυση του προβλήματος του σακιδίου. Αυτή η συνάρτηση αναλύει την απόδοση του πράκτορα σε ένα πραγματικό σενάριο και παρέχει τα τελικά αποτελέσματα της πρόβλεψης.



5.72 Στιγμιότυπο κώδικα solve\_with\_knapsack\_dqn

Η συνάρτηση αρχίζει με τη δημιουργία ενός συνόλου αντικειμένων (**items**) που περιλαμβάνει τις τιμές και τα βάρη των αντικειμένων. Ένα νέο περιβάλλον **KnapsackEnv** δημιουργείται χρησιμοποιώντας τα αντικείμενα και την ικανότητα του σακιδίου. Η αρχική κατάσταση (**state**) του περιβάλλοντος λαμβάνεται με τη μέθοδο **reset**.

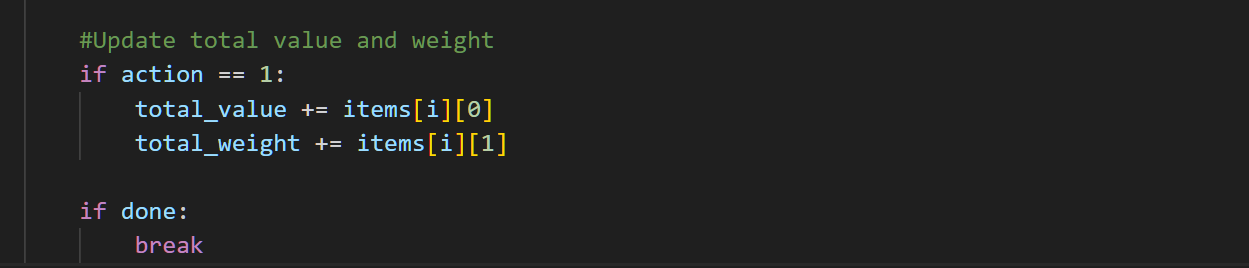
Δημιουργούνται επίσης οι μεταβλητές **selected\_items**, prediction, **total\_value**, και **total\_weight** για την αποθήκευση των επιλεγμένων αντικειμένων, των προβλέψεων, της συνολικής αξίας, και του συνολικού βάρους αντίστοιχα.



5.73 Στιγμιότυπο κώδικα solve\_with\_knapsack\_dqn (2)

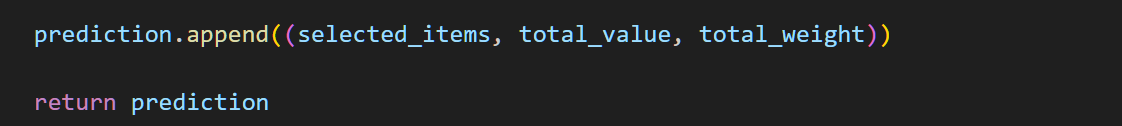
Ο βρόγχος επιλογής αντικειμένων διατρέχει όλα τα αντικείμενα του σακιδίου. Για κάθε αντικείμενο, ο πράκτορας χρησιμοποιεί το εκπαιδευμένο δίκτυο πολιτικής (**policy\_net**) για να υπολογίσει τις Q-values της τρέχουσας κατάστασης. Στη συνέχεια, επιλέγεται η ενέργεια με την υψηλότερη Q-value (η καλύτερη απόφαση σύμφωνα με το μοντέλο).

Η επιλεγμένη ενέργεια προστίθεται στη λίστα **selected\_items**, και το περιβάλλον ενημερώνεται με τη νέα κατάσταση. Η διαδικασία συνεχίζεται μέχρι να ολοκληρωθεί η επιλογή ή να μην υπάρχουν άλλα αντικείμενα.



5.74 Στιγμιότυπο κώδικα solve\_with\_knapsack\_dqn (3)

Μετά από κάθε ενέργεια, αν το αντικείμενο επιλέγεται (δηλαδή αν **action == 1**), οι τιμές και τα βάρη του προστίθενται στη συνολική αξία (**total\_value**) και στο συνολικό βάρος (**total\_weight**). Ο βρόχος διακόπτεται αν δεν υπάρχουν άλλα αντικείμενα για επιλογή ή αν έχει ολοκληρωθεί η επιλογή.



5.75 Στιγμιότυπο κώδικα solve\_with\_knapsack\_dqn (4)

Τέλος, η συνάρτηση επιστρέφει τις προβλέψεις του πράκτορα. Τα αποτελέσματα περιλαμβάνουν τα επιλεγμένα αντικείμενα, τη συνολική αξία και το συνολικό βάρος του σακιδίου. Αυτά τα δεδομένα αποθηκεύονται στη λίστα prediction και επιστρέφονται ως αποτέλεσμα της συνάρτησης.

### Βελτιστοποίηση Υπερπαραμέτρων Μοντέλου DQN

Για το μοντέλο DQN, χρησιμοποιήθηκε μια παρόμοια προσέγγιση με αυτήν που χρησιμοποιήθηκε για το DNN, για τη ρύθμιση των υπερπαραμέτρων του DQN μοντέλου. Χρησιμοποιήθηκε η τεχνική Grid Search για να βρούμε το βέλτιστο σύνολο υπερπαραμέτρων, δοκιμάζοντας διάφορους συνδυασμούς τιμών.

Συγκεκριμένα για το DQN μοντέλο, οι υπερπαράμετροι που εξετάστηκαν ήταν:

* **Ρυθμός Μάθησης (Learning Rate):** Η ταχύτητα εκμάθησης για το DQN πράκτορα.
* **Κρυφά Στρώματα (Hidden Sizes):** Οι διαστάσεις των κρυφών στρωμάτων στο νευρωνικό δίκτυο.
* **Αριθμός Επεισοδίων (Episodes):** Ο αριθμός των επεισοδίων εκπαίδευσης.
* **Μέγεθος Παρτίδας (Batch Size):** Το μέγεθος της παρτίδας για την εμπειρία αναπαραγωγής.
* **Γάμμα (Gamma):** Ο παράγοντας έκπτωσης για την ενίσχυση μελλοντικών ανταμοιβών.
* **Ενημέρωση Στόχου (Target Update):** Η συχνότητα ενημέρωσης του στόχου δικτύου.
* **Μείωση Εξερεύνησης (Epsilon Decay):** Ο ρυθμός με τον οποίο μειώνεται η πιθανότητα τυχαίας επιλογής (exploration).

Η αξιολόγηση των διαφόρων συνδυασμών πραγματοποιήθηκε με τη χρήση του "Optimality Gap" όπως και στο DNN, με στόχο την ελαχιστοποίηση της διαφοράς μεταξύ της βέλτιστης και της προβλεπόμενης λύσης.

**Συμπεράσματα**

Από τη διαδικασία αυτή προέκυψαν παράμετροι για το μοντέλο DQN, οι οποίες βελτίωσαν τη συνολική απόδοση και ακρίβεια του μοντέλου στην επίλυση του προβλήματος του σακιδίου.

* Learning Rate (lr): 0.0001
* Hidden Sizes (hidden\_sizes): [256, 128]
* Episodes (num\_episodes): 1000
* Batch size (batch\_size): 128
* Gamma (gamma): 0.99
* Target update (target\_update): 10
* Decay rate (decay\_rate): 0.00

Όπως και για τα προηγούμενα μοντέλα, ο πλήρης κώδικας για τη διαδικασία βελτιστοποίησης υπερπαραμέτρων παρέχεται στο Παράρτημα Β.

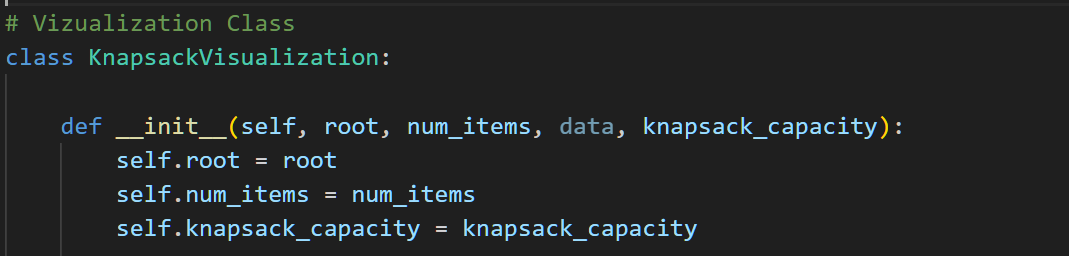
## Οπτικοποίηση – Γραφική Επίδειξη Εκπαίδευσης

Σε αυτήν την ενότητα, θα αναλυθούν οι λειτουργίες που αφορούν τη γραφική επίδειξη της διαδικασίας εκπαίδευσης.

### Κλάση Γραφικής Επίδειξης της Διαδικασίας Εκπαίδευσης (KnapsackVisualization)

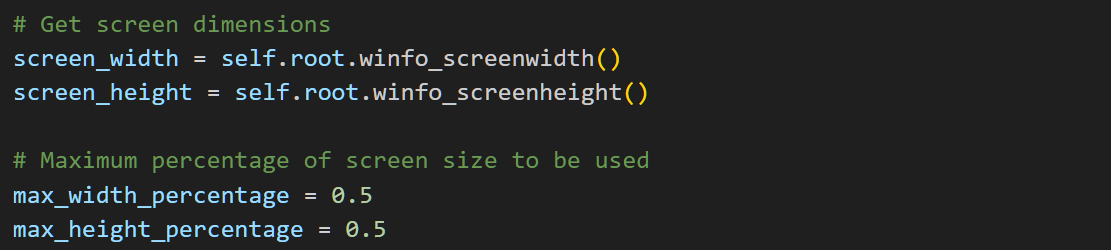
Η κλάση KnapsackVisualization είναι υπεύθυνη για την οπτικοποίηση του προβλήματος του σακιδίου, δημιουργώντας ένα παράθυρο όπου απεικονίζονται τα αντικείμενα και το σακίδιο. Περιλαμβάνει λειτουργίες για τη διαχείριση του μεγέθους του παραθύρου, την τοποθέτηση των αντικειμένων σε ένα grid και την ενημέρωση της οπτικοποίησης με βάση τις επιλογές που γίνονται.

**Α. Αρχικοποίηση της Κλάσης (\_\_init\_\_)**



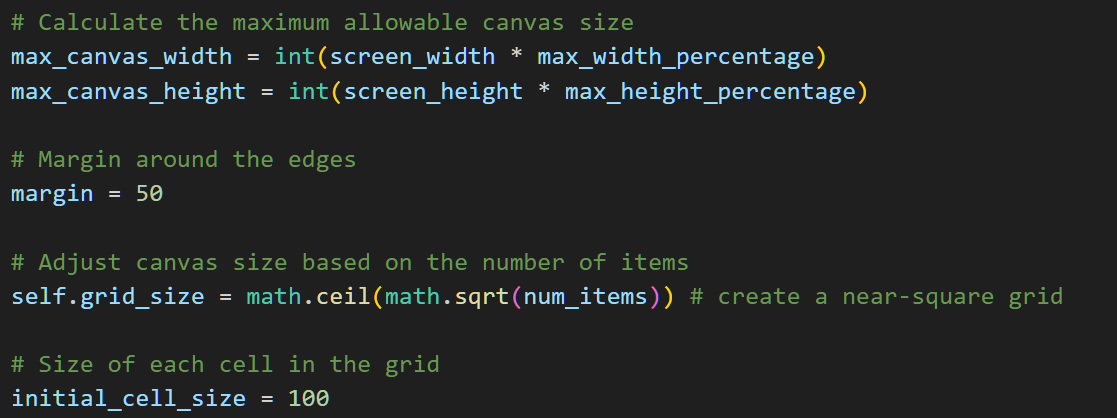
5.76 Στιγμιότυπο κλάσης KnapsackVisualization

Αρχικά, η κλάση λαμβάνει το κύριο παράθυρο της εφαρμογής (**root**), τον αριθμό των αντικειμένων και τη χωρητικότητα του σακιδίου. Αυτές οι παράμετροι αποθηκεύονται για να χρησιμοποιηθούν στις επόμενες λειτουργίες της κλάσης.



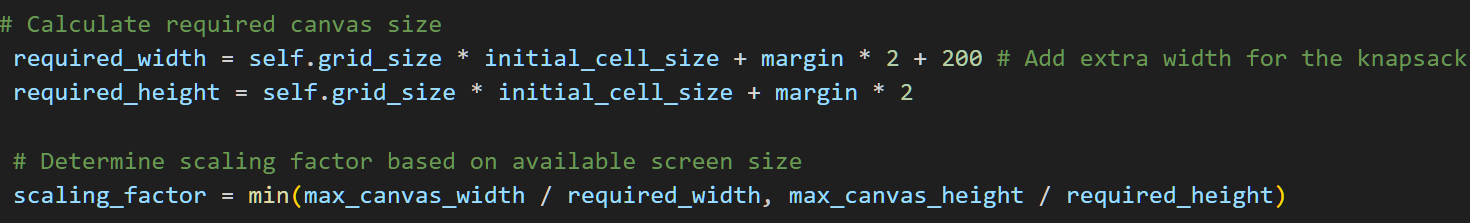
5.77 Στιγμιότυπο κλάσης KnapsackVisualization (2)

Στη συνέχεια, η κλάση παίρνει τις διαστάσεις της οθόνης και καθορίζει το μέγιστο ποσοστό αυτών που θα χρησιμοποιηθεί για την απεικόνιση του grid. Το 50% του πλάτους και του ύψους της οθόνης θεωρούνται ως τα όρια για τη δημιουργία του καμβά.



5.78 Στιγμιότυπο κλάσης KnapsackVisualization (3)

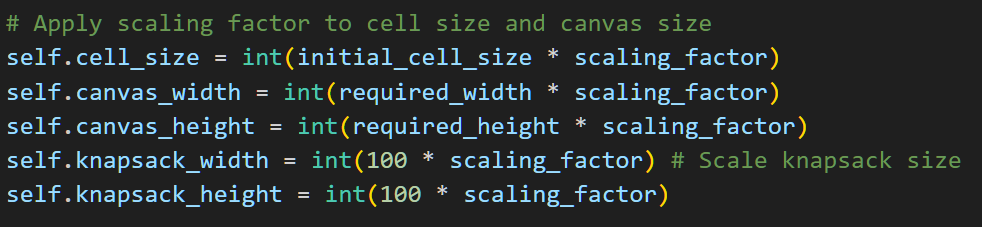
Με βάση το διαθέσιμο μέγεθος της οθόνης, υπολογίζονται οι μέγιστες διαστάσεις του καμβά που θα φιλοξενήσει το grid των αντικειμένων. Ο αριθμός των αντικειμένων επηρεάζει το μέγεθος του grid, το οποίο είναι σχεδιασμένο έτσι ώστε να προσεγγίζει μια τετράγωνη διάταξη.



5.79 Στιγμιότυπο κλάσης KnapsackVisualization (4)

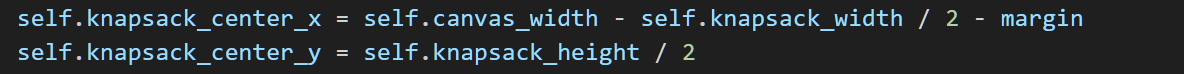
Εδώ, υπολογίζεται το απαραίτητο μέγεθος του καμβά και ένας παράγοντας κλίμακας για να διασφαλιστεί ότι ο καμβάς θα χωρέσει στις διαθέσιμες διαστάσεις της οθόνης. Το **scaling factor** προσαρμόζει τις διαστάσεις του grid έτσι ώστε να παραμένουν εντός του ορίου που έχει τεθεί από το μέγεθος της οθόνης.

**Β. Δημιουργία και Διαχείριση του Καμβά**

****

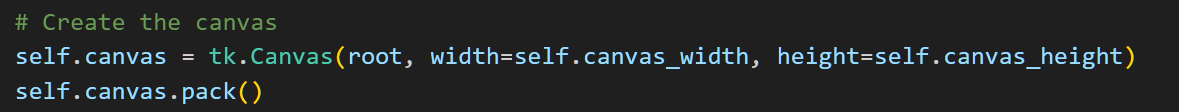
**5.80** Στιγμιότυπο κλάσης KnapsackVisualization (5)

Οι διαστάσεις του καμβά και των κυψελών του grid προσαρμόζονται σύμφωνα με τον παράγοντα κλίμακας. Το μέγεθος του σακιδίου καθορίζεται επίσης με βάση αυτόν τον παράγοντα, ώστε να ταιριάζει στο συνολικό μέγεθος του καμβά.



5.81 Στιγμιότυπο κλάσης KnapsackVisualization (6)

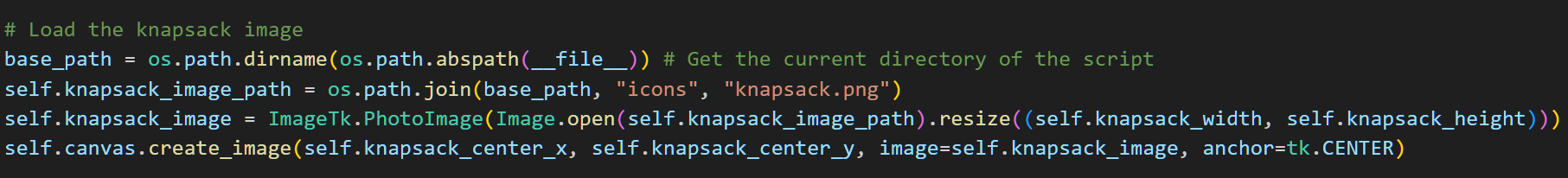
Οι συντεταγμένες του κέντρου του σακιδίου στον καμβά καθορίζονται με τέτοιο τρόπο ώστε να εμφανίζεται σωστά στο κάτω μέρος του grid.



5.82 Στιγμιότυπο κλάσης KnapsackVisualization (7)

Δημιουργείται ο καμβάς χρησιμοποιώντας το Tkinter, με τις διαστάσεις που υπολογίστηκαν προηγουμένως. Ο καμβάς αυτός θα φιλοξενήσει όλα τα στοιχεία της οπτικοποίησης.

**Γ. Φόρτωση και Εμφάνιση των Εικόνων**



5.83 Στιγμιότυπο κλάσης KnapsackVisualization (8)

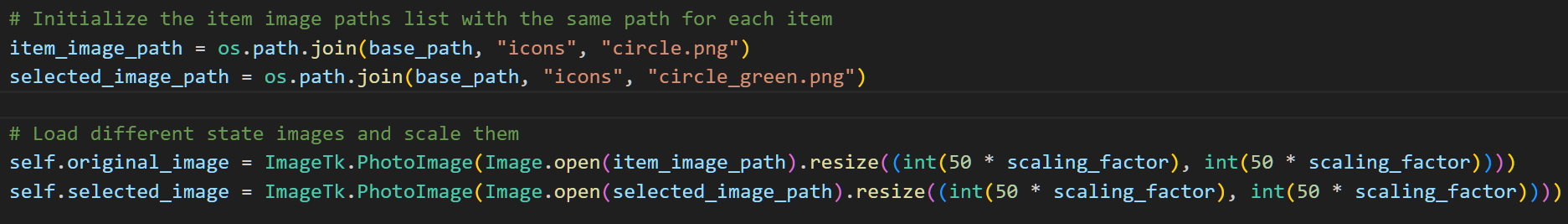
Αρχικά, η κλάση υπολογίζει το απόλυτο μονοπάτι του τρέχοντος script χρησιμοποιώντας τη μεταβλητή **base\_path**. Αυτό εξασφαλίζει ότι η σωστή διαδρομή προς τα αρχεία εικόνας προσδιορίζεται ανεξάρτητα από το που βρίσκεται το script στο σύστημα αρχείων.

Με τη βοήθεια του **base\_path**, η διαδρομή προς την εικόνα του σακιδίου κατασκευάζεται χρησιμοποιώντας τη συνάρτηση **os.path.join**, η οποία συνδυάζει το **base\_path** με το σχετικό μονοπάτι προς την εικόνα μέσα στο φάκελο **icons.** Η εικόνα του σακιδίου φορτώνεται, προσαρμόζεται στο κατάλληλο μέγεθος με βάση τις υπολογισμένες διαστάσεις και τοποθετείται στο κέντρο του καμβά.



5.84 Εικόνα Άδειου Σαικιδίου, generated by DALL-E, OpenAI

.

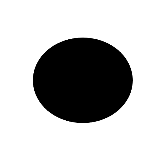


5.85 Στιγμιότυπο κλάσης KnapsackVisualization (9)

Στη συνέχεια, οι διαδρομές προς τις εικόνες των αντικειμένων δημιουργούνται με τον ίδιο τρόπο. Η **item\_image\_path** οδηγεί στην εικόνα που χρησιμοποιείται για τα μη επιλεγμένα αντικείμενα, ενώ η **selected\_image\_path** οδηγεί στην εικόνα για τα επιλεγμένα αντικείμενα.

Αφού οριστούν οι διαδρομές προς τις εικόνες των αντικειμένων, αυτές οι εικόνες φορτώνονται και προσαρμόζονται σε μέγεθος σύμφωνα με τον παράγοντα κλίμακας που έχει υπολογιστεί νωρίτερα. Η **original\_image** αντιπροσωπεύει την αρχική εικόνα που θα χρησιμοποιηθεί για τα μη επιλεγμένα αντικείμενα, ενώ η **selected\_image** είναι η εικόνα που θα εμφανίζεται όταν ένα αντικείμενο επιλέγεται.

Αυτός ο κώδικας διασφαλίζει ότι οι εικόνες των αντικειμένων προσαρμόζονται σωστά και είναι έτοιμες να χρησιμοποιηθούν για την οπτικοποίηση του προβλήματος του σακιδίου. Η χρήση διαφορετικών εικόνων για τα επιλεγμένα και μη επιλεγμένα αντικείμενα προσφέρει μια πιο δυναμική και κατανοητή οπτική αναπαράσταση της διαδικασίας επιλογής των αντικειμένων.

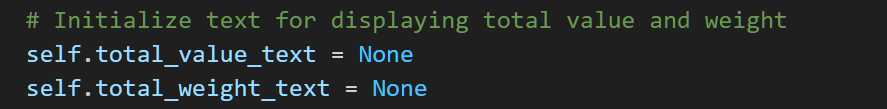


5.86 Μαύρος κύκλος – μη επιλεγμένα αντικείμενα



5.87 Μάυρος κύκλος με πράσινο περίγραμμα - επιλεγμένα αντικείμενα

**Δ. Προετοιμασία για την Εμφάνιση Δεδομένων**

****

**5.88** Στιγμιότυπο κλάσης KnapsackVisualization (10)

Αυτές οι δύο μεταβλητές προετοιμάζονται για να εμφανίσουν τις συνολικές τιμές και τα βάρη των επιλεγμένων αντικειμένων κάτω από το σακίδιο.

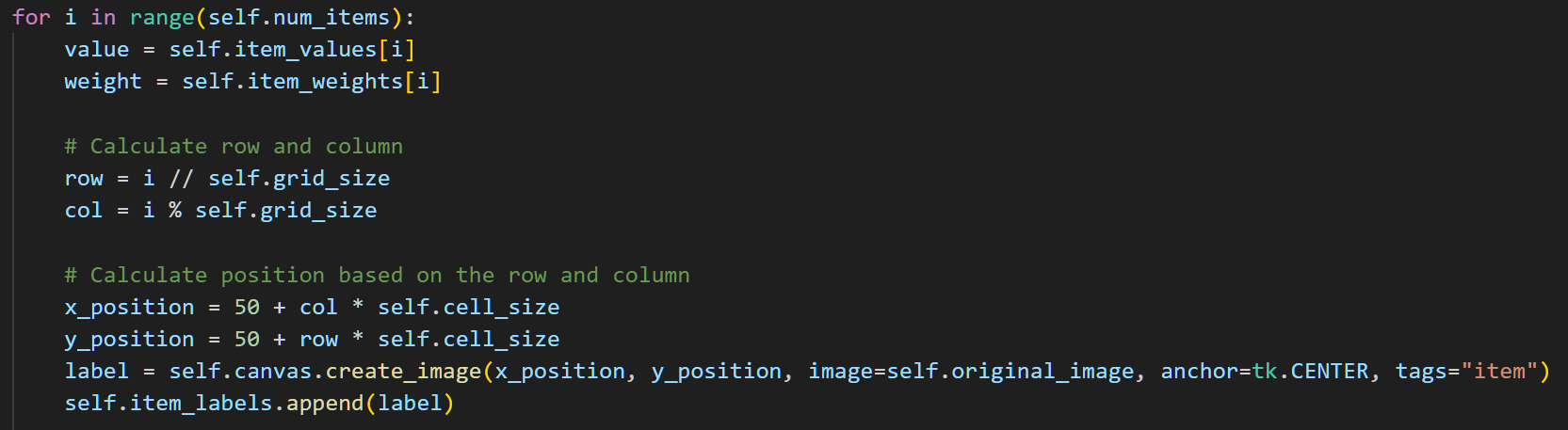
**Σχεδίαση του Grid και Ενημέρωση Οπτικοποίησης**

**Α. Σχεδίαση του Grid**

****

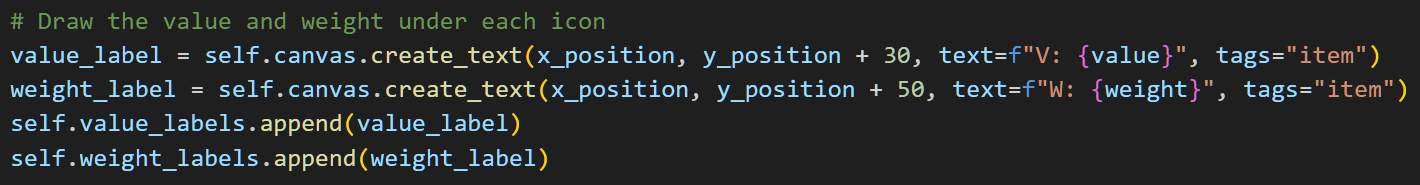
**5.89** Στιγμιότυπο κώδικα draw\_grid

Η μέθοδος **draw\_grid** καθαρίζει τον καμβά από οποιαδήποτε προηγούμενα δεδομένα και προετοιμάζει λίστες για την αποθήκευση των ετικετών και των εικόνων των αντικειμένων.



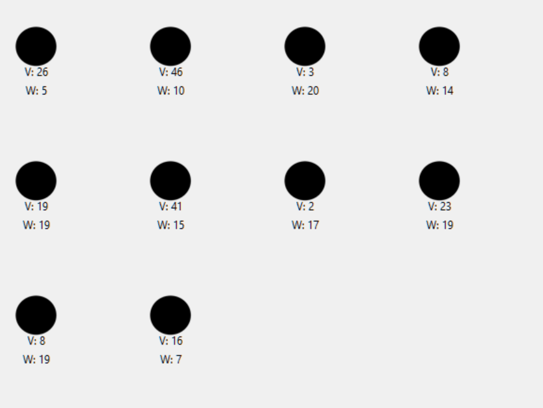
5.90 Στιγμιότυπο κώδικα draw\_grid (2)

Για κάθε αντικείμενο, η θέση του στο **grid** καθορίζεται από τον αριθμό της σειράς και της στήλης στην οποία ανήκει. Η εικόνα του αντικειμένου τοποθετείται σε αυτήν τη θέση.



5.91 Στιγμιότυπο κώδικα draw\_grid (3)

Κάτω από κάθε εικόνα αντικειμένου, προστίθενται ετικέτες που δείχνουν την αξία και το βάρος του αντικειμένου, ώστε να είναι ορατά στους χρήστες.

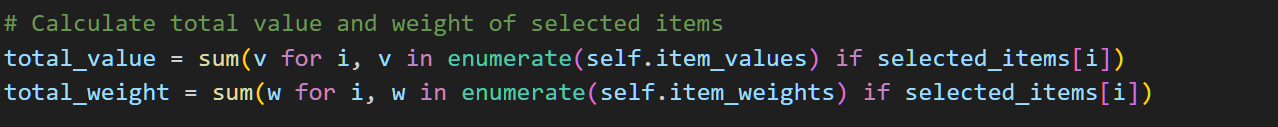


**Β. Ενημέρωση της Οπτικοποίησης**



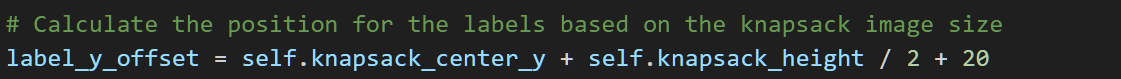
5.92 Στιγμιότυπο κώδικα update

Η μέθοδος **update** ενημερώνει τις τιμές και τα βάρη των αντικειμένων με τα νέα δεδομένα και στη συνέχεια επανασχεδιάζει το **grid**. Καλείται κατά τη διάρκεια της εκπαίδευσης των μοντέλων μέσα από τις μεθόδους εκπαίδευσης για την ενημέρωση της γραφική επίδειξης της διαδικασίας εκπαίδευσης σε πραγματικό χρόνο.

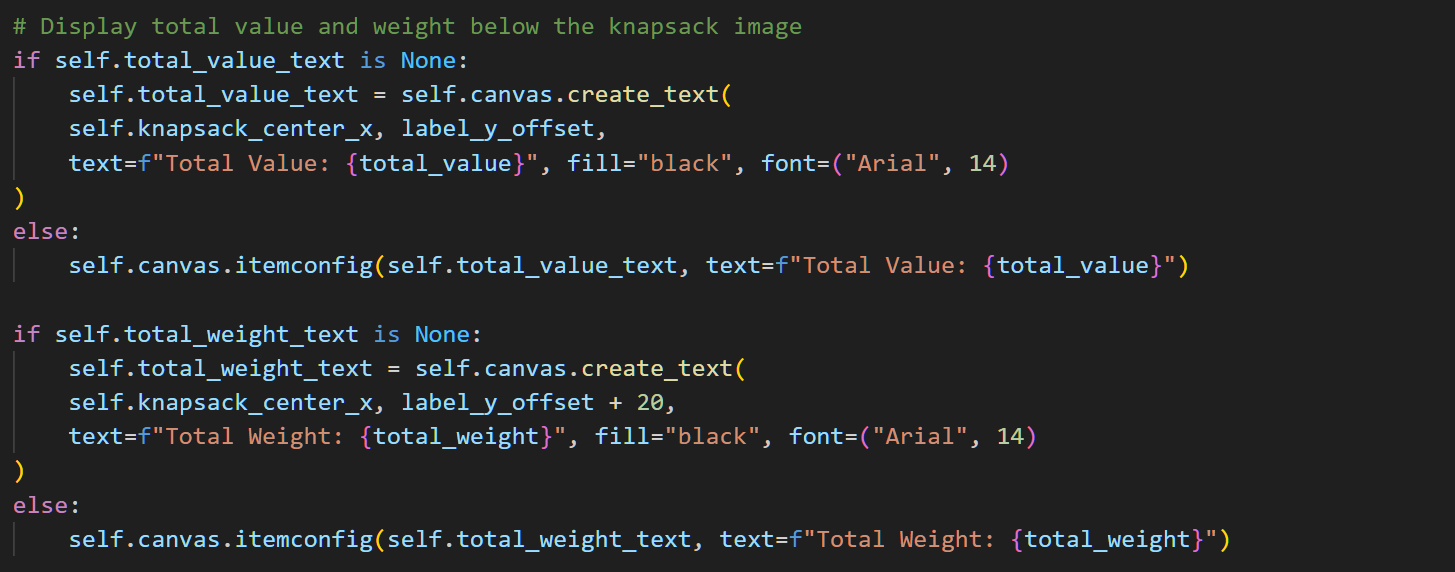


5.93 Στιγμιότυπο κώδικα update (2)

Υπολογίζεται η συνολική αξία και το βάρος όλων των επιλεγμένων αντικειμένων, τα οποία θα εμφανιστούν κάτω από την εικόνα του σακιδίου.



5.94 Στιγμιότυπο κώδικα update (3)

Καθορίζεται η θέση για την τοποθέτηση των ετικετών συνολικής αξίας και βάρους κάτω από το σακίδιο. 

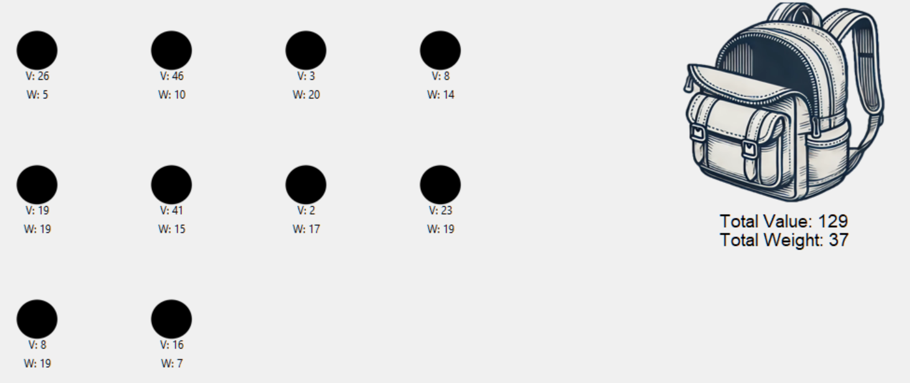
5.95 Στιγμιότυπο κώδικα update (4)

Οι ετικέτες που δείχνουν τη συνολική αξία και το βάρος ενημερώνονται ή δημιουργούνται αν δεν υπάρχουν ήδη, και εμφανίζονται κάτω από το σακίδιο.

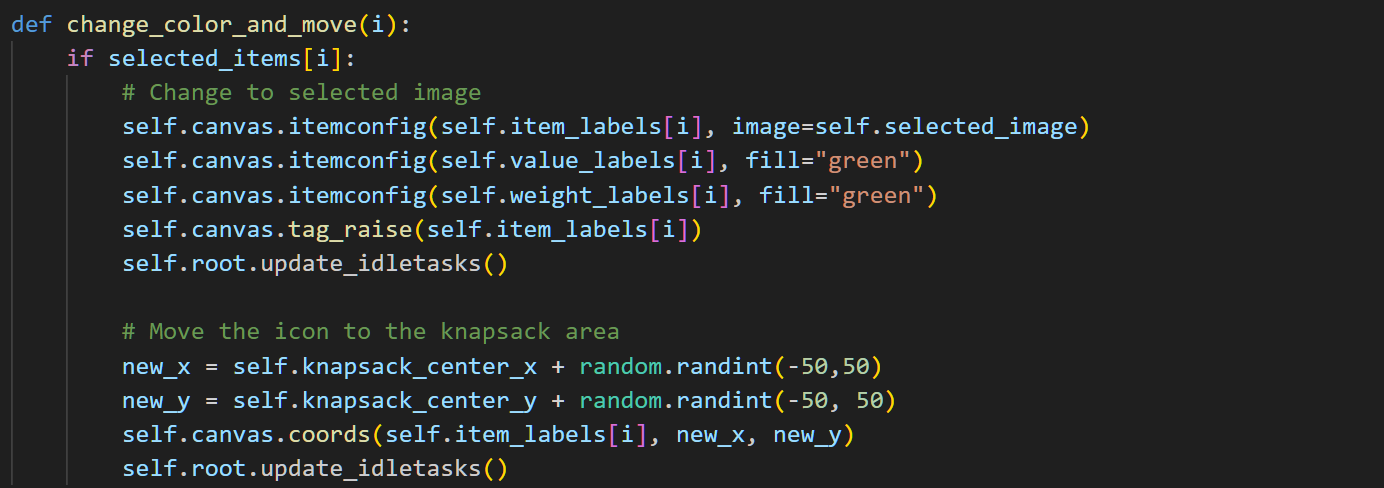


5.96 Στιγμιότυπο κώδικα update (5)

Τέλος, η μέθοδος καλεί το **Tkinter** για να ενημερώσει το παράθυρο, ώστε να αντικατοπτρίζονται οι τελευταίες αλλαγές.

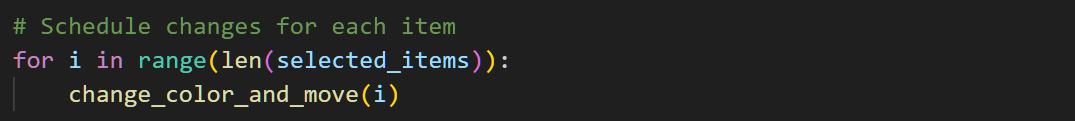


**Γ. Κίνηση και Αλλαγή Χρώματος των Επιλεγμένων Αντικειμένων**

****

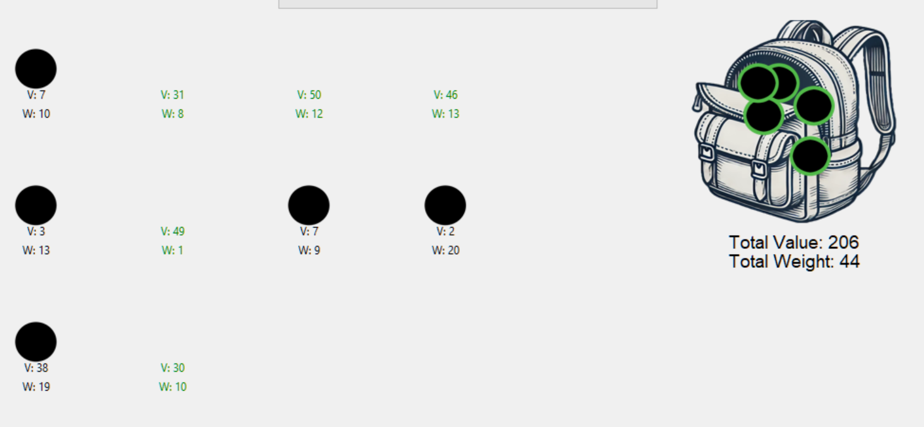
**5.97** Στιγμιότυπο κώδικα change\_color\_and\_move

Η μέθοδος **change\_color\_and\_move** αλλάζει την εικόνα των επιλεγμένων αντικειμένων σε πράσινη και τα μετακινεί κοντά στο σακίδιο, προσομοιώνοντας την επιλογή τους για τοποθέτηση μέσα σε αυτό. Αυτή η κίνηση γίνεται με τυχαία μετατόπιση γύρω από το κέντρο του σακιδίου.



5.98 Στιγμιότυπο κώδικα update (6)

Η **change\_color\_and\_move** εφαρμόζεται σε όλα τα επιλεγμένα αντικείμενα, δίνοντας μια δυναμική αίσθηση της επιλογής τους κατά την επίλυση του προβλήματος.

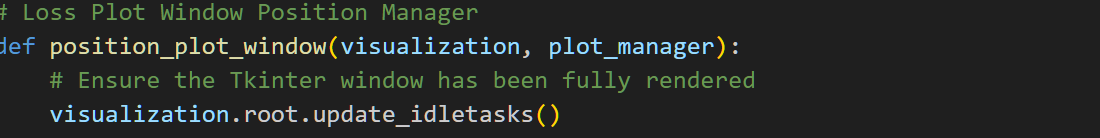


**Συνοψίζοντας**

Η κλάση **KnapsackVisualization** προσφέρει μια πλήρη και διαδραστική οπτικοποίηση του προβλήματος του σακιδίου, δίνοντας τη δυνατότητα στους χρήστες να βλέπουν τις επιλογές αντικειμένων και τις επιπτώσεις τους σε πραγματικό χρόνο.

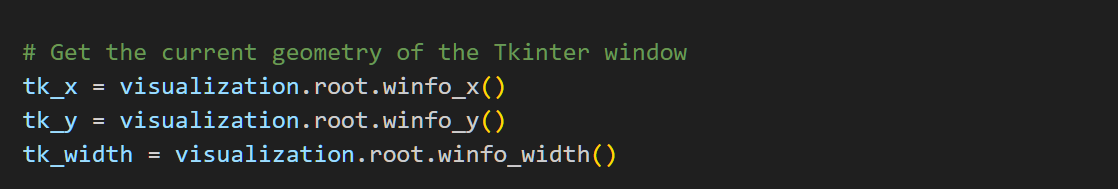
### Διαχείρηση Θέσης Παραθύρου Οπτικοποίησης

Η ενότητα αυτή αφορά τη λειτουργία που είναι υπεύθυνη για τη διαχείριση της θέσης του παραθύρου οπτικοποίησης σε σχέση με το κύριο παράθυρο της εφαρμογής. Στόχος είναι το παράθυρο της οπτικοποίησης να τοποθετείται δίπλα στο κύριο παράθυρο της εφαρμογής, εξασφαλίζοντας έτσι εύκολη πρόσβαση και καλύτερη οπτική εμπειρία.



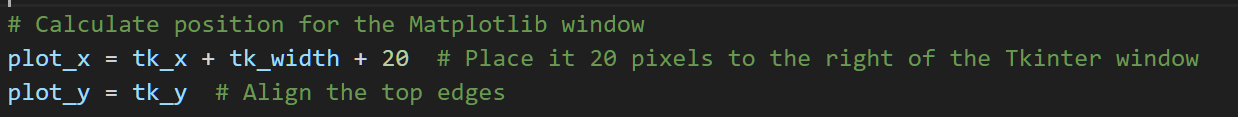
5.99 Στιγμιότυπο κώδικα position\_plot\_window

Αρχικά, η συνάρτηση **position\_plot\_window** ανανεώνει το κύριο παράθυρο της εφαρμογής (**Tkinter**) για να διασφαλίσει ότι όλες οι αλλαγές έχουν εφαρμοστεί και ότι οι διαστάσεις του παραθύρου έχουν υπολογιστεί σωστά.



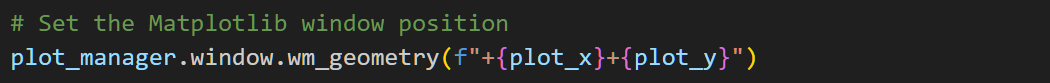
5.100 Στιγμιότυπο κώδικα position\_plot\_window (2)

Στη συνέχεια, η συνάρτηση λαμβάνει τις τρέχουσες συντεταγμένες **(x, y**) και τις διαστάσεις (πλάτος και ύψος) του παραθύρου **Tkinter**. Αυτές οι πληροφορίες είναι απαραίτητες για να τοποθετηθεί το παράθυρο οπτικοποίησης σε σωστή θέση σε σχέση με το κύριο παράθυρο.



5.101 Στιγμιότυπο κώδικα position\_plot\_window (3)

Η θέση του παραθύρου οπτικοποίησης (**Matplotlib**) υπολογίζεται έτσι ώστε να βρίσκεται 20 pixels δεξιά από το κύριο παράθυρο **Tkinter** και να ευθυγραμμίζεται με το πάνω μέρος του κύριου παραθύρου. Αυτό επιτρέπει στο παράθυρο οπτικοποίησης να είναι εύκολα προσβάσιμο και ορατό κατά τη διάρκεια της αλληλεπίδρασης με την εφαρμογή.



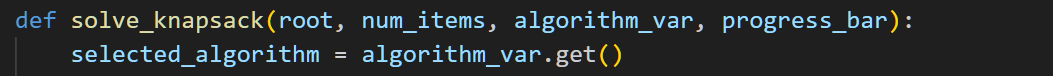
5.102 Στιγμιότυπο κώδικα position\_plot\_window (4)

Τέλος, η συνάρτηση ορίζει τη θέση του παραθύρου **Matplotlib** χρησιμοποιώντας τις συντεταγμένες που υπολογίστηκαν νωρίτερα. Με αυτόν τον τρόπο, το παράθυρο οπτικοποίησης εμφανίζεται ακριβώς δίπλα στο κύριο παράθυρο της εφαρμογής, δημιουργώντας μια πιο ολοκληρωμένη και λειτουργική διεπαφή χρήστη.

## Κεντρική Εφαρμογή και Διεπαφή Χρήστη (GUI)

Η ενότητα αυτή επικεντρώνεται στη βασική λειτουργία της εφαρμογής, η οποία περιλαμβάνει την κύρια λογική για την εκτέλεση και επίλυση του προβλήματος του σακιδίου χρησιμοποιώντας τα εκπαιδευμένα μοντέλα. Η λειτουργία αυτή είναι υπεύθυνη για τη διαχείριση των δεδομένων, την επιλογή του κατάλληλου μοντέλου και την εκκίνηση της διαδικασίας εκπαίδευσης ή πρόβλεψης.

### Κεντρική Λειτουργία Εφαρμογής

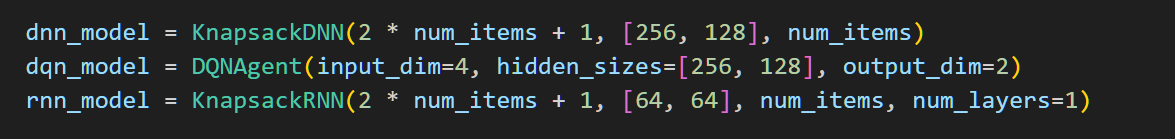


5.103 Στιγμιότυπο κώδικα solve\_knapsack

Αρχικά, η συνάρτηση **solve\_knapsack** λαμβάνει ως είσοδο το κύριο παράθυρο της εφαρμογής (**root**), τον αριθμό των αντικειμένων (**num**\_**items**), την επιλογή του αλγορίθμου από τον χρήστη (**algorithm\_var**) και μια γραμμή προόδου (**progress\_bar**). Η πρώτη ενέργεια της συνάρτησης είναι να αποθηκεύσει την επιλογή του αλγορίθμου σε μια μεταβλητή.

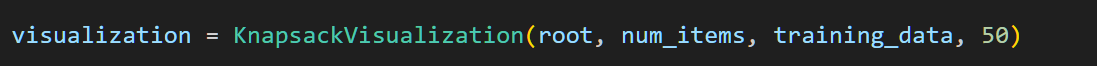


5.104 Στιγμιότυπο κώδικα solve\_knapsack (2)

Στη συνέχεια, δημιουργούνται τα δεδομένα εκπαίδευσης για το πρόβλημα του σακιδίου, χρησιμοποιώντας τη συνάρτηση **generate\_data**, η οποία παράγει δείγματα δεδομένων που θα χρησιμοποιηθούν για την εκπαίδευση των μοντέλων. 

5.105 Στιγμιότυπο κώδικα solve\_knapsack (3)

Εδώ, δημιουργούνται τα αντικείμενα των τριών διαφορετικών μοντέλων που μπορούν να χρησιμοποιηθούν για την επίλυση του προβλήματος: ένα βαθύ νευρωνικό δίκτυο (**KnapsackDNN**), ένας πράκτορας Deep Q-Learning (**DQNAgent**), και ένα επαναλαμβανόμενο νευρωνικό δίκτυο (**KnapsackRNN**).



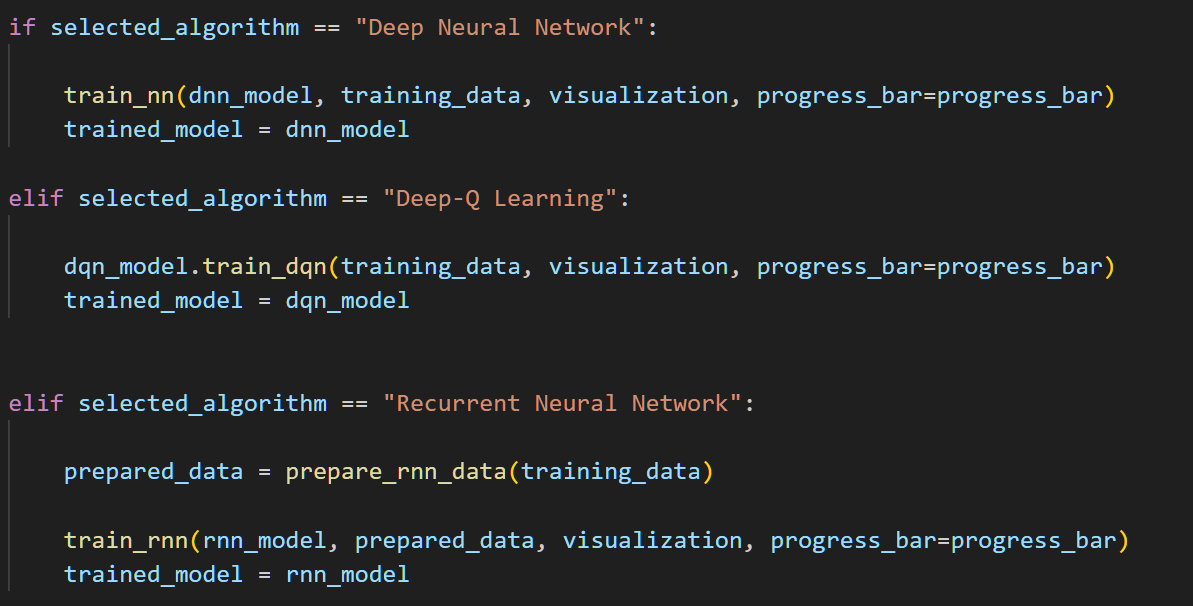
5.106 Στιγμιότυπο κώδικα solve\_knapsack (4)

Η οπτικοποίηση του προβλήματος του σακιδίου ρυθμίζεται με τη δημιουργία ενός αντικειμένου **KnapsackVisualization**, το οποίο αναλαμβάνει την απεικόνιση των αντικειμένων και του σακιδίου.



5.107 Στιγμιότυπο κώδικα solve\_knapsack (5)

Δύο global μεταβλητές, **trained\_model** και **current\_algorithm**, αρχικοποιούνται. Η **trained\_model** θα χρησιμοποιηθεί για να αποθηκεύσει το εκπαιδευμένο μοντέλο, ενώ η **current\_algorithm** καταγράφει τον αλγόριθμο που επιλέχθηκε από τον χρήστη.

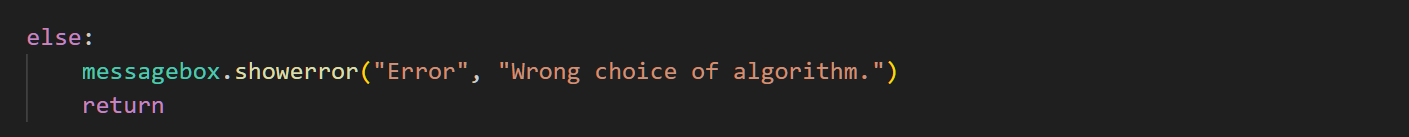


5.108 Στιγμιότυπο κώδικα solve\_knapsack (6)

Σε αυτό το τμήμα κώδικα, ελέγχεται ποιος αλγόριθμος επιλέχθηκε από τον χρήστη.

Αν επιλεγεί το **Deep Neural Network (DNN)**, τα δεδομένα εκπαίδευσης δημιουργούνται και το μοντέλο εκπαιδεύεται χρησιμοποιώντας τη συνάρτηση **train\_nn**. Το εκπαιδευμένο μοντέλο αποθηκεύεται στην μεταβλητή **trained\_model**.

Αν ο χρήστης επιλέξει τον αλγόριθμο **Deep-Q Learning**, καλείται η συνάρτηση **train\_dqn** του αντικειμένου **DQNAgent** για την εκπαίδευση του πράκτορα και αποθηκεύεται το εκπαιδευμένο μοντέλο στην ίδια μεταβλητή.

Αν επιλεγεί το **Επαναλαμβανόμενο Νευρωνικό Δίκτυο (RNN)**, τα δεδομένα εκπαίδευσης προετοιμάζονται με τη βοήθεια της **prepare\_rnn\_data**, και το μοντέλο εκπαιδεύεται με τη χρήση της **train**\_**rnn**. Το εκπαιδευμένο **RNN** αποθηκεύεται στην ίδια μεταβλητή. 

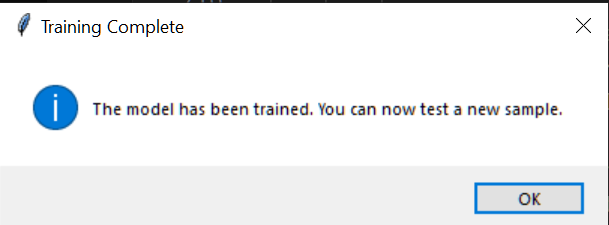
5.109 Στιγμιότυπο κώδικα solve\_knapsack (7)

Αν η επιλογή του αλγορίθμου δεν αναγνωριστεί, εμφανίζεται ένα μήνυμα λάθους στον χρήστη και η συνάρτηση τερματίζεται.



5.110 Στιγμιότυπο κώδικα solve\_knapsack (8)

Τέλος, όταν ολοκληρωθεί η εκπαίδευση του μοντέλου, ο χρήστης ενημερώνεται με ένα μήνυμα ότι η διαδικασία εκπαίδευσης έχει ολοκληρωθεί και ότι μπορεί πλέον να δοκιμάσει το μοντέλο με νέα δεδομένα.

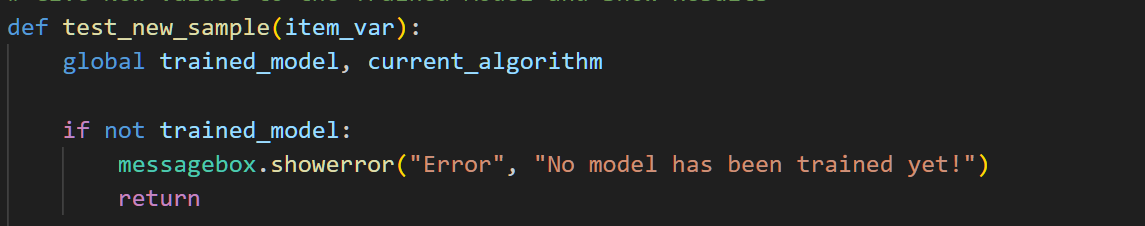


**Συνοψίζοντας**

Η βασική λειτουργία της εφαρμογής είναι υπεύθυνη για την εκκίνηση της διαδικασίας εκπαίδευσης με βάση την επιλογή του χρήστη και τη διασφάλιση ότι το κατάλληλο μοντέλο εκπαιδεύεται και αποθηκεύεται για μετέπειτα χρήση. Επιπλέον, παρέχει οπτικοποίηση σε πραγματικό χρόνο, καθιστώντας τη διαδικασία πιο διαδραστική και κατανοητή.

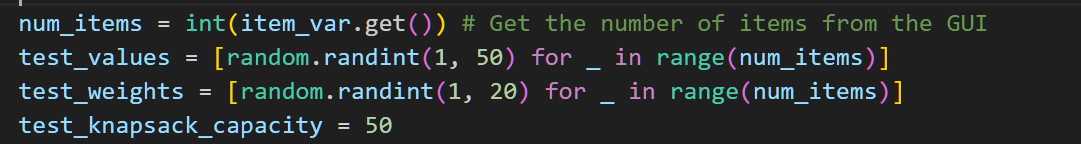
### Πρόβλεψη Εκπαιδευμένου Μοντέλου με Νέο Δείγμα

Σε αυτήν την ενότητα, εξετάζουμε τη λειτουργία που επιτρέπει τη δοκιμή του εκπαιδευμένου μοντέλου με νέα δεδομένα, τα οποία δεν έχουν χρησιμοποιηθεί κατά την εκπαίδευση. Η λειτουργία αυτή επιτρέπει στον χρήστη να εισάγει ένα νέο δείγμα και να δει πώς το εκπαιδευμένο μοντέλο αντιμετωπίζει το πρόβλημα.



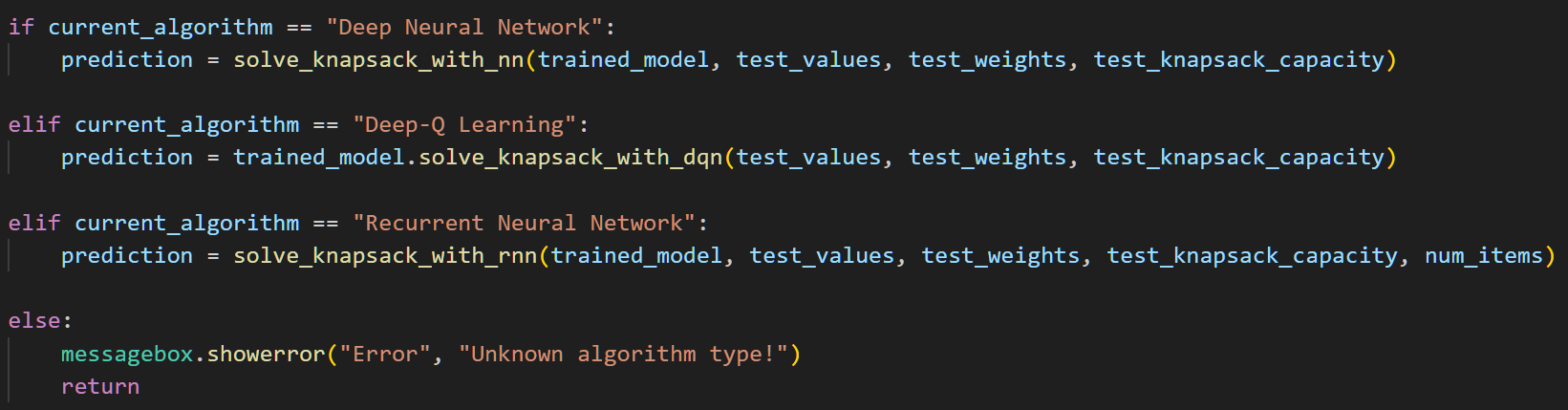
5.111 Στιγμιότυπο κώδικα test\_new\_sample

Η συνάρτηση **test\_new\_sample** ξεκινά με την πρόσβαση στις παγκόσμιες μεταβλητές **trained\_model** και **current\_algorithm**, οι οποίες αποθηκεύουν το εκπαιδευμένο μοντέλο και τον αλγόριθμο που επιλέχθηκε. Αν δεν υπάρχει εκπαιδευμένο μοντέλο (δηλαδή αν δεν έχει προηγηθεί εκπαίδευση), εμφανίζεται ένα μήνυμα λάθους και η διαδικασία διακόπτεται.



5.112 Στιγμιότυπο κώδικα test\_new\_sample (2)

Στη συνέχεια, η συνάρτηση λαμβάνει τον αριθμό των αντικειμένων από τη διεπαφή χρήστη (**GUI**) και δημιουργεί τυχαίες τιμές και βάρη για τα αντικείμενα αυτά. Το μέγιστο επιτρεπόμενο βάρος του σακιδίου ορίζεται σε **50** μονάδες.

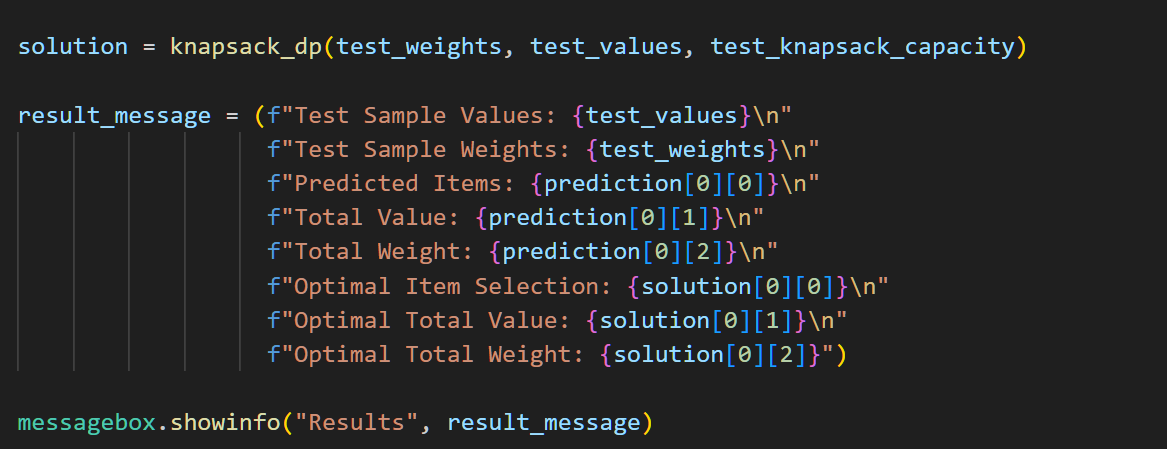


5.113 Στιγμιότυπο κώδικα test\_new\_sample (3)

Ανάλογα με τον αλγόριθμο που χρησιμοποιήθηκε για την εκπαίδευση, καλείται η αντίστοιχη συνάρτηση για την επίλυση του προβλήματος με το νέο δείγμα. Για το **Deep Neural Network (DNN),** καλείται η **solve\_knapsack\_with\_nn**.

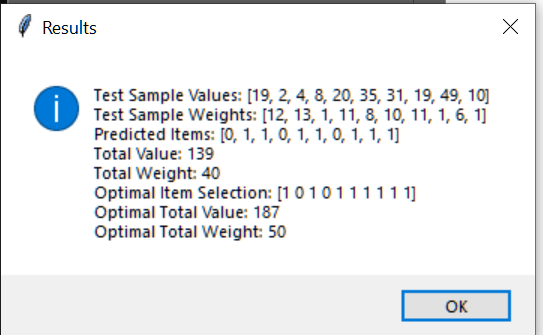
Για το **Deep-Q Learning**, χρησιμοποιείται η μέθοδος **solve\_knapsack\_with\_dqn** του εκπαιδευμένου πράκτορα. Τέλος, για το **Recurrent Neural Network (RNN)**, καλείται η **solve\_knapsack\_with\_rnn**.

Αν για κάποιο λόγο το **current\_algorithm** δεν ταιριάζει με κανέναν από τους αναγνωρισμένους αλγορίθμους, εμφανίζεται ένα μήνυμα λάθους και η συνάρτηση τερματίζεται.



5.114 Στιγμιότυπο κώδικα test\_new\_sample (4)

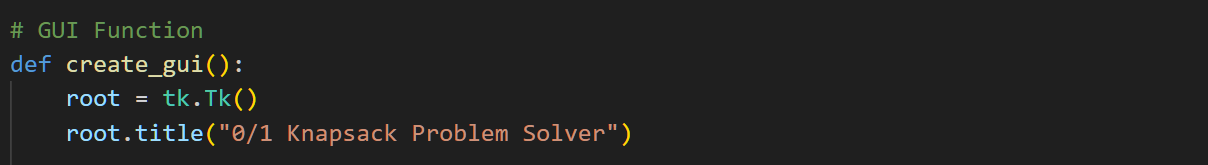
Στο τέλος, η συνάρτηση υπολογίζει τη βέλτιστη λύση του προβλήματος χρησιμοποιώντας τον αλγόριθμο δυναμικού προγραμματισμού (**knapsack**\_**dp**) για σύγκριση. Στη συνέχεια, δημιουργείται ένα μήνυμα που περιλαμβάνει τις τιμές και τα βάρη του νέου δείγματος, τα αντικείμενα που επιλέχθηκαν από το μοντέλο, καθώς και τη συνολική αξία και το βάρος αυτών των αντικειμένων. Αυτά τα αποτελέσματα συγκρίνονται με την πραγματική βέλτιστη λύση και εμφανίζονται στο χρήστη μέσω ενός μηνύματος.



**Συνοψίζοντας**

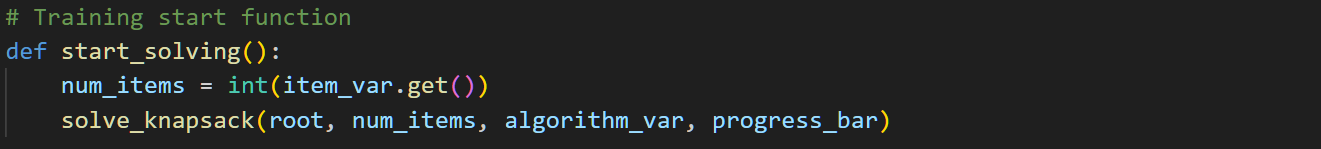
Η λειτουργία **test\_new\_sample** επιτρέπει στο χρήστη να δοκιμάσει το εκπαιδευμένο μοντέλο με νέα, τυχαία δεδομένα και να συγκρίνει την απόδοση του μοντέλου με την ιδανική λύση του προβλήματος. Με αυτόν τον τρόπο, ο χρήστης μπορεί να αξιολογήσει την αποτελεσματικότητα του μοντέλου και να διαπιστώσει την ακρίβεια των προβλέψεών του σε διαφορετικά σενάρια.

### Δημιουργία και Λειτουργία Διεπαφής Χρήστη (GUI)

Η ενότητα αυτή περιγράφει τη διαδικασία δημιουργίας της διεπαφής χρήστη (**Graphical User Interface - GUI)**, η οποία επιτρέπει στο χρήστη να αλληλεπιδρά με την εφαρμογή, να επιλέγει τον αλγόριθμο που θα χρησιμοποιηθεί, να εκπαιδεύει τα μοντέλα και να δοκιμάζει νέα δείγματα. 

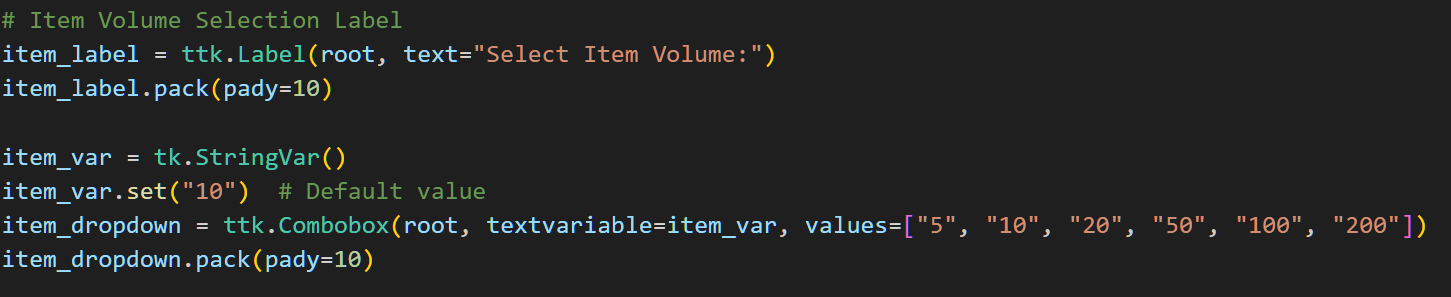
5.115 Στιγμιότυπο κώδικα create\_gui

Η συνάρτηση **create**\_**gui** ξεκινά με τη δημιουργία του κύριου παραθύρου της εφαρμογής, το οποίο αντιπροσωπεύεται από το αντικείμενο root. Ο τίτλος του παραθύρου ορίζεται ως "**Knapsack Problem Solver**"**.**



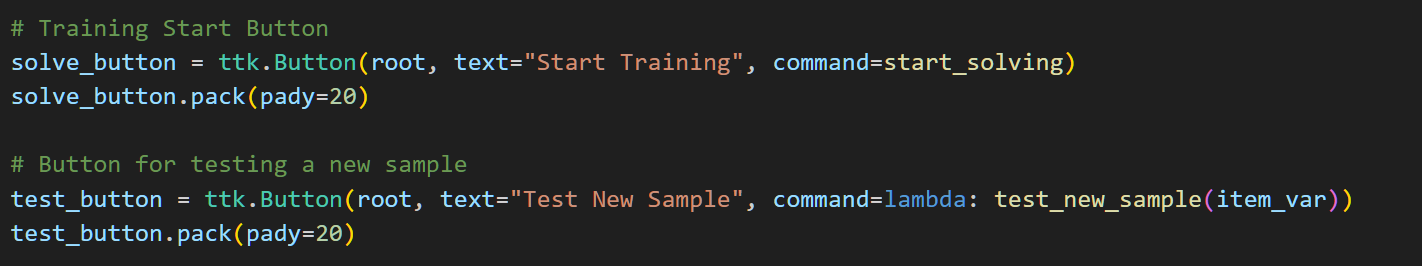
5.116 Στιγμιότυπο κώδικα start\_solving

Εδώ, ορίζεται μια εσωτερική συνάρτηση **start\_solving**, η οποία καλείται όταν ο χρήστης πατήσει το κουμπί για να ξεκινήσει την εκπαίδευση. Η συνάρτηση αυτή παίρνει τον αριθμό των αντικειμένων από το **GUI** και καλεί τη **solve\_knapsack** για να ξεκινήσει την επίλυση του προβλήματος.



5.117 Στιγμιότυπο κώδικα create\_gui (2)

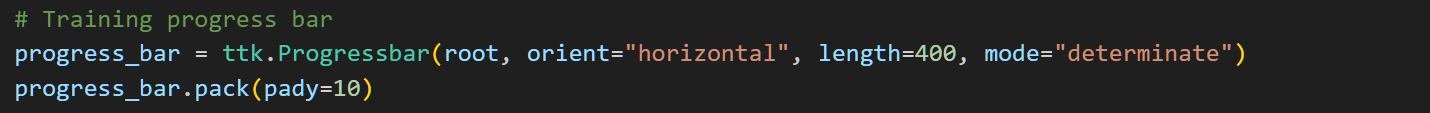
Εδώ, προστίθεται μια ετικέτα και ένα **dropdown menu (Combobox**) για την επιλογή του αριθμού αντικειμένων που θα περιέχει το κάθε δείγμα. Οι διαθέσιμες επιλογές είναι 5, 10, 20, 50, 100 και 200, με προεπιλογή τα 10 αντικείμενα.



5.118 Στιγμιότυπο κώδικα create\_gui (3)

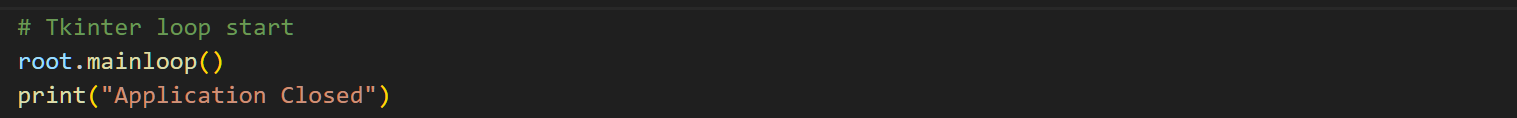
Εδώ, προστίθεται μια δεύτερη ετικέτα και ένα δεύτερο **dropdown menu** (**Combobox**) για την επιλογή του αλγορίθμου εκπαίδευσης. Οι διαθέσιμοι αλγόριθμοι περιλαμβάνουν το **Deep Neural Network**, το **Deep-Q Learning** και το **Recurrent Neural Network**, με προεπιλεγμένη επιλογή το **Deep Neural Network**.

Τα δύο κουμπιά προστίθενται για να επιτρέψουν στο χρήστη να ξεκινήσει την εκπαίδευση και να δοκιμάσει το εκπαιδευμένο μοντέλο με νέα δείγματα. Το κουμπί "**Start Training**" καλεί τη συνάρτηση **start\_solving**, ενώ το κουμπί "**Test New Sample**" καλεί τη συνάρτηση **test\_new\_sample**.



5.119 Στιγμιότυπο κώδικα create\_gui (4)

Η γραμμή προόδου χρησιμοποιείται για την οπτική απεικόνιση της προόδου της εκπαίδευσης. Η γραμμή ενημερώνεται κατά τη διάρκεια της εκπαίδευσης, επιτρέποντας στον χρήστη να παρακολουθεί πόσο κοντά είναι η ολοκλήρωση της διαδικασίας.

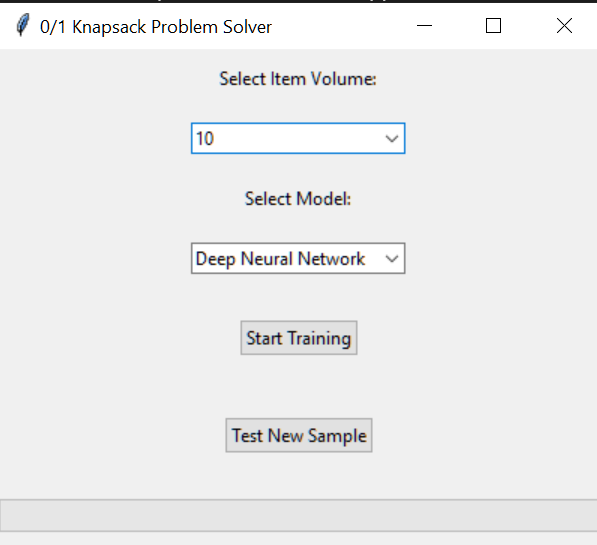


5.120 Στιγμιότυπο κώδικα mainloop

Η συνάρτηση **mainloop** του **Tkinter** ξεκινά τον κυρίως βρόχο της εφαρμογής, επιτρέποντας στο **GUI** να παραμένει ενεργό και να αλληλεπιδρά με το χρήστη. Όταν η εφαρμογή κλείσει, τυπώνεται το μήνυμα "**Application** **Closed**" στην κονσόλα.



Η κλήση της διεπαφής χρήστη γίνεται απλά με την εκτέλεση της συνάρτησης **create\_gui()** που έχουμε ήδη αναλύσει. Αυτό το κομμάτι κώδικα τοποθετείται συνήθως στο τέλος του κύριου προγράμματος ή του αρχείου Python, εξασφαλίζοντας ότι η διεπαφή χρήστη ενεργοποιείται όταν εκτελείται το πρόγραμμα.



5.121 GUI Menu

**Συνοψίζοντας**

Η διεπαφή χρήστη, παρέχει στον χρήστη εύκολες επιλογές για τον όγκο των αντικειμένων και τον αλγόριθμο εκπαίδευσης. Τα κουμπιά εκπαίδευσης και δοκιμής προσφέρουν άμεση πρόσβαση στις βασικές λειτουργίες της εφαρμογής, ενώ η γραμμή προόδου ενημερώνει τον χρήστη για την εξέλιξη της διαδικασίας.

## Αυτοματοποιημένες δοκιμές και Παρακολούθηση Απόδοσης

Για τις δοκιμές και την παρακολούθηση της απόδοσης των μοντέλων, χρησιμοποιήθηκε μια διαδικασία αυτοματοποίησης. Η διαδικασία αυτή επιτρέπει την σύγκριση των αποτελεσμάτων των μοντέλων σε όλους τους επιλεγμένους όγκους δεδομένων και την εξαγωγή κρίσιμων τιμών όπως η ακρίβεια, ο χρόνος εκπαίδευσης, η απώλεια και η χρήση μνήμης.

**Βήματα Αυτόματης Δοκιμής**

1. **Δημιουργία Συνόλων Δεδομένων:** Για κάθε μέγεθος δεδομένων (5, 10, 20, 50, 100, 200 αντικείμενα), δημιουργούνται νέα δεδομένα με την υπάρχουσα συνάρτηση **generate\_data**. Αυτά τα δεδομένα χρησιμεύουν τόσο για την εκπαίδευση όσο και για τη δοκιμή των μοντέλων.
2. **Εκπαίδευση των Μοντέλων:** Η εκπαίδευση των μοντέλων γίνεται για κάθε μέγεθος δεδομένων. Οι συναρτήσεις εκπαίδευσης έχουν τροποποιηθεί ώστε να καταγράφουν την απώλεια και τη χρήση μνήμης κατά τη διάρκεια της εκπαίδευσης.
3. **Αξιολόγηση των Μοντέλων:** Μετά την εκπαίδευση καταγράφεται η ακρίβεια των προβλέψεων του κάθε μοντέλου, ο χρόνος που χρειάστηκε για την εκπαίδευση και η μέγιστη χρήση μνήμης.
4. **Αποτελέσματα:** Τα αποτελέσματα συγκεντρώνονται και παρουσιάζονται σε συγκριτική μορφή, επιτρέποντας την άμεση σύγκριση των μοντέλων.

Ο πλήρης κώδικας για τις αυτοματοποιημένες δοκιμές παρέχεται στο Παράρτημα Β. Περιλαμβάνει τις μεθόδους **train\_and\_evaluate\_model**, **automate\_process** και **analyze\_results**.

## Επίλογος Κεφαλαίου

Στο κεφάλαιο αυτό, παρουσιάστηκε η υλοποίηση και η εκπαίδευση τριών μοντέλων νευρωνικών δικτύων για την επίλυση του προβλήματος του σακιδίου: το Βαθύ Νευρωνικό Δίκτυο (DNN), το Επαναλαμβανόμενο Νευρωνικό Δίκτυο (RNN), και το μοντέλο Deep Q-Learning (DQN). Καταγράφηκαν οι αρχιτεκτονικές, οι διαδικασίες εκπαίδευσης, καθώς και η προσαρμογή των υπερπαραμέτρων για κάθε μοντέλο.

Κατά την υλοποίηση, χρησιμοποιήθηκαν σύγχρονες τεχνολογίες και βιβλιοθήκες όπως το **PyTorch** για την κατασκευή και εκπαίδευση των νευρωνικών δικτύων, καθώς και το **Tkinter** για τη δημιουργία ενός φιλικού γραφικού περιβάλλοντος χρήστη (**GUI**). Επιπλέον, η διαδικασία αυτοματοποίησης των δοκιμών και της αξιολόγησης υλοποιήθηκε ενσωματώνοντας μεθόδους για τη διαχείριση των υπερπαραμέτρων και την ανάλυση των επιδόσεων.

Το κεφάλαιο αυτό αποτελεί ένα καθοριστικό βήμα για την κατανόηση της πρακτικής εφαρμογής των μεθόδων μηχανικής μάθησης στην επίλυση του προβλήματος του σακιδίου, ενώ ταυτόχρονα προετοιμάζει το έδαφος για την ανάλυση και σύγκριση των αποτελεσμάτων. Οι τεχνολογίες που χρησιμοποιήθηκαν έπαιξαν σημαντικό ρόλο στην αποδοτικότητα και την ακρίβεια των μοντέλων, συμβάλλοντας ουσιαστικά στη συνολική επιτυχία της μελέτης.

# Ανάλυση Αποτελεσμάτων και Συμπεράσματα

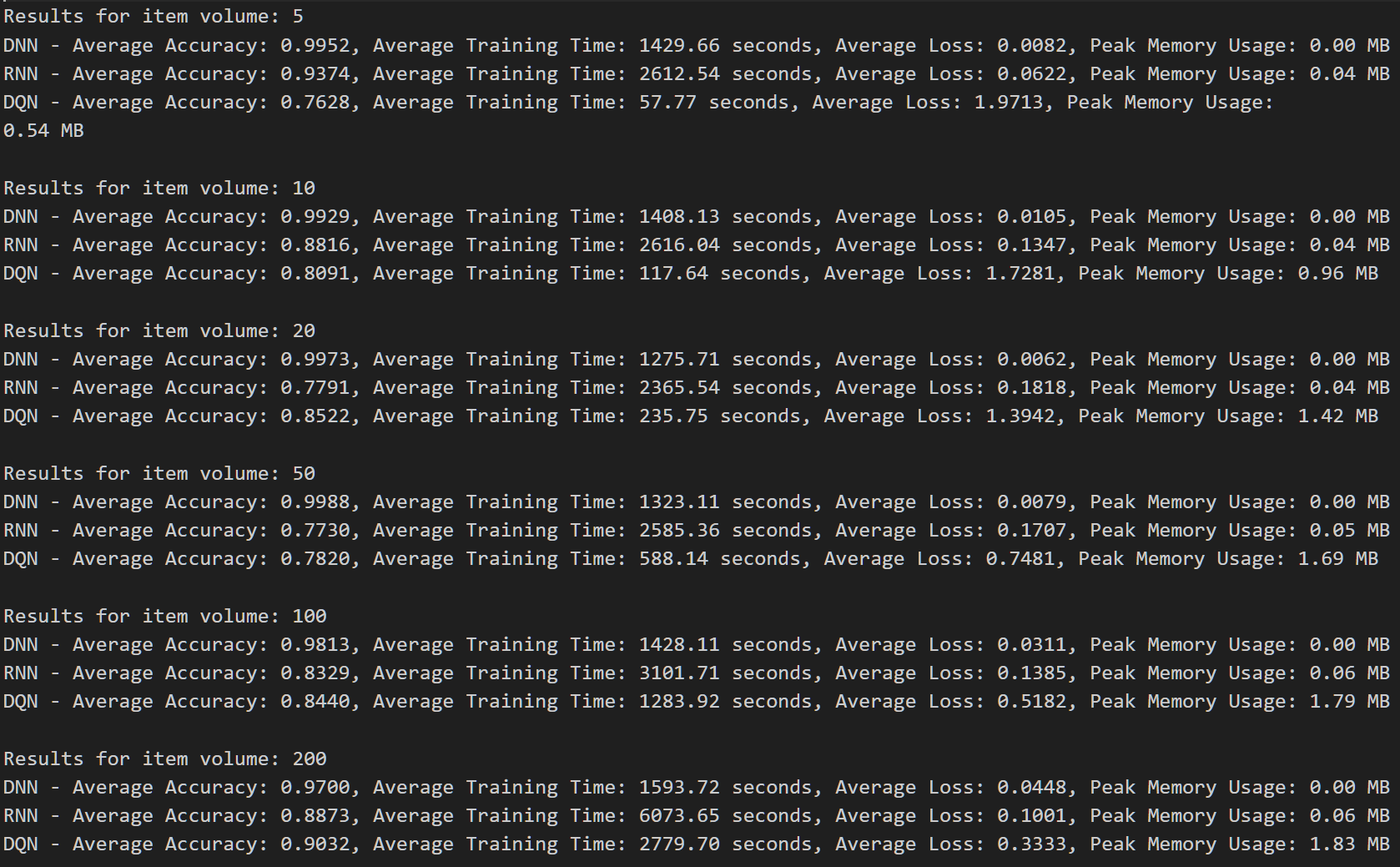
## Εισαγωγή

Στο κεφάλαιο αυτό, παρουσιάζονται και αναλύονται τα αποτελέσματα που προέκυψαν από την εκπαίδευση και αξιολόγηση τριών διαφορετικών μοντέλων νευρωνικών δικτύων για την επίλυση του προβλήματος του σακιδίου: το Βαθύ Νευρωνικό Δίκτυο (DNN), το Επαναλαμβανόμενο Νευρωνικό Δίκτυο (RNN), και το μοντέλο Deep Q-Learning (DQN). Μέσα από συστηματικές αυτοματοποιημένες δοκιμές, αξιολογείται η απόδοση κάθε μοντέλου σε όρους ακρίβειας, χρόνου εκπαίδευσης, απόκλισης και χρήσης μνήμης, για διαφορετικούς όγκους δεδομένων (5, 10, 20, 50, 100, 200).

Στόχος του κεφαλαίου είναι να δώσει μια σαφή εικόνα για το πώς κάθε μοντέλο ανταποκρίνεται στις απαιτήσεις του προβλήματος, επισημαίνοντας τα πλεονεκτήματα και τα μειονεκτήματα κάθε προσέγγισης. Με βάση τα αποτελέσματα, εξάγονται συμπεράσματα σχετικά με την καταλληλότητα κάθε μοντέλου, παρέχοντας έτσι πολύτιμη καθοδήγηση για τη βέλτιστη επιλογή μοντέλου ανάλογα με τις ανάγκες της εφαρμογής.

## Ανάλυση των Αποτελεσμάτων Εκπαίδευσης

**Έξοδος του Python Script Αυτοματοποιημένων Δοκιμών:**

****

**Ανάλυση Αποτελεσμάτων:**

**Βαθύ Νευρωνικό Δίκτυο (DNN):**

* Το DNN παρουσιάζει εξαιρετικά υψηλή ακρίβεια σε όλες τις περιπτώσεις, με μέσες τιμές που κυμαίνονται από 0.9700 έως 0.9988, υποδηλώνοντας ότι είναι ικανό να προσαρμόζεται καλά σε διάφορους όγκους αντικειμένων.
* Ο χρόνος εκπαίδευσης κυμαίνεται μεταξύ 1275.71 και 1593.72 δευτερολέπτων, παραμένοντας αρκετά σταθερός, ενώ η απόκλιση (loss) παραμένει πολύ χαμηλή, δείχνοντας ότι το μοντέλο μαθαίνει αποτελεσματικά.
* Η μνήμη που χρησιμοποιήθηκε παραμένει σταθερά χαμηλή, με κορυφαία τιμή 0.00 MB, γεγονός που καταδεικνύει την αποδοτικότητα του μοντέλου όσον αφορά τη διαχείριση πόρων.

**Επαναλαμβανόμενο Νευρωνικό Δίκτυο (RNN):**

* Το RNN δείχνει χαμηλότερη ακρίβεια σε σχέση με το DNN, με τιμές που κυμαίνονται από 0.7730 έως 0.9374, ειδικά για μεγαλύτερους όγκους αντικειμένων όπου η ακρίβεια πέφτει αισθητά.
* Ο χρόνος εκπαίδευσης είναι ο υψηλότερος μεταξύ των τριών μοντέλων, φτάνοντας μέχρι και τα 6073.65 δευτερόλεπτα για όγκο αντικειμένων 200, γεγονός που υποδηλώνει υψηλή υπολογιστική απαίτηση.
* Η απόκλιση (loss) είναι σημαντικά μεγαλύτερη από αυτή του DNN, δείχνοντας ότι το RNN δυσκολεύεται να εκπαιδευτεί αποτελεσματικά στα δεδομένα του προβλήματος.

**Μοντέλο Deep Q-Learning (DQN):**

* Το DQN παρουσιάζει τη χαμηλότερη ακρίβεια από τα τρία μοντέλα, με τιμές που κυμαίνονται από 0.7628 έως 0.9032. Η απόδοση του DQN βελτιώνεται με την αύξηση του όγκου των αντικειμένων, αλλά παραμένει χαμηλότερη από τα άλλα μοντέλα.
* Ο χρόνος εκπαίδευσης του DQN είναι ο μικρότερος από όλα τα μοντέλα, κάτι που το καθιστά μια γρήγορη αλλά λιγότερο ακριβή επιλογή.
* Η μνήμη που χρησιμοποιείται είναι μεγαλύτερη από τα άλλα μοντέλα, με κορυφαία τιμή 1.83 MB, γεγονός που δείχνει ότι το DQN έχει μεγαλύτερες απαιτήσεις μνήμης κατά τη διάρκεια της εκπαίδευσης.

**Συμπεράσματα:**

* Το DNN αναδεικνύεται ως το πιο ακριβές και αποδοτικό μοντέλο όσον αφορά την ακρίβεια, την απόκλιση και τη χρήση μνήμης, καθιστώντας το την προτιμώμενη επιλογή για την επίλυση του προβλήματος του σακιδίου.
* Το RNN, αν και αποδίδει αρκετά καλά σε μικρότερους όγκους αντικειμένων, παρουσιάζει υψηλές υπολογιστικές απαιτήσεις και μειωμένη ακρίβεια σε μεγαλύτερους όγκους.
* Το DQN προσφέρει έναν καλό συμβιβασμό μεταξύ χρόνου εκπαίδευσης και απόδοσης, ωστόσο, υπολείπεται στην ακρίβεια σε σχέση με τα άλλα δύο μοντέλα.

## Σύγκριση των Μοντέλων

**Ακρίβεια των Μοντέλων**

Στα αποτελέσματα των δοκιμών, το DNN εμφάνισε τη μεγαλύτερη ακρίβεια, φτάνοντας το 99% σε πολλές περιπτώσεις, ακόμα και με μεγαλύτερα μεγέθη δεδομένων. Το RNN είχε χαμηλότερη απόδοση, ειδικά σε σύνολα δεδομένων με περισσότερα αντικείμενα, όπου η ακρίβειά του κυμαινόταν γύρω στο 77% με 88%. Το DQN, αν και βελτιώθηκε σε σύνολα δεδομένων με περισσότερα αντικείμενα, παρέμεινε πίσω σε σχέση με το DNN, επιτυγχάνοντας ακρίβειες από 76% έως 90%.

Από τα παραπάνω συμπεραίνεται ότι το DNN είναι το πιο αξιόπιστο μοντέλο για την επίλυση του προβλήματος του σακιδίου, καθώς διατηρεί υψηλή ακρίβεια ανεξάρτητα από το μέγεθος του συνόλου δεδομένων.

**Χρόνος Εκπαίδευσης**

Όσον αφορά τον χρόνο εκπαίδευσης, το DNN απαιτούσε σημαντικά λιγότερο χρόνο από το RNN, ενώ το DQN είχε το μικρότερο χρόνο εκπαίδευσης από όλα τα μοντέλα. Αυτό δείχνει ότι το DQN είναι πιο γρήγορο στην εκπαίδευση, αλλά δεν επιτυγχάνει την ίδια ακρίβεια με το DNN. Το RNN είχε τον μεγαλύτερο χρόνο εκπαίδευσης, γεγονός που πιθανώς οφείλεται στην πολυπλοκότητα του μοντέλου και την ανάγκη για διαδοχική επεξεργασία των δεδομένων.

**Απώλεια (Loss)**

Η απώλεια (loss) κατά την εκπαίδευση είναι ένας κρίσιμος δείκτης για την απόδοση των μοντέλων. Το DNN είχε την χαμηλότερη απώλεια σε όλα τα μεγέθη δεδομένων, επιβεβαιώνοντας την ικανότητά του να μαθαίνει αποτελεσματικά τις σχέσεις μεταξύ των δεδομένων. Το RNN είχε μεγαλύτερη απώλεια, ενώ το DQN, παρά το γεγονός ότι βελτίωνε την απόδοσή του όσο αυξανόταν το μέγεθος των δεδομένων, εξακολουθούσε να έχει υψηλότερες απώλειες σε σύγκριση με το DNN.

**Κατανάλωση Μνήμης**

Όσον αφορά την κατανάλωση μνήμης, το DNN είχε τη χαμηλότερη κατανάλωση, σχεδόν μηδενική, ενώ το RNN κατανάλωνε λίγο περισσότερη μνήμη. Το DQN παρουσίασε τη μεγαλύτερη κατανάλωση μνήμης, η οποία αυξάνονταν με το μέγεθος των δεδομένων. Αυτό υποδηλώνει ότι το DQN, παρότι ταχύτερο στην εκπαίδευση, απαιτεί περισσότερους πόρους μνήμης, καθιστώντας το λιγότερο αποδοτικό από πλευράς χρήσης πόρων.

**Συμπεράσματα**

Συνολικά, το DNN ξεχώρισε ως το καλύτερο μοντέλο για την επίλυση του προβλήματος του σακιδίου, παρέχοντας υψηλή ακρίβεια με χαμηλό κόστος σε χρόνο εκπαίδευσης και μνήμη. Το RNN, ενώ είναι ισχυρό σε προβλήματα με διαδοχική φύση, φαίνεται να μην υπερτερεί στο συγκεκριμένο πρόβλημα, και το DQN, αν και γρήγορο στην εκπαίδευση, υστερεί σε ακρίβεια και απαιτεί περισσότερους πόρους μνήμης. Τα αποτελέσματα αυτά καταδεικνύουν την ανάγκη για επιλογή του κατάλληλου μοντέλου βάσει των απαιτήσεων της εφαρμογής και των διαθέσιμων πόρων.

## Εξέταση Απόδοσης με Βάση τα Δεδομένα Εισόδου

**Επίδραση του Αριθμού Αντικειμένων στην Απόδοση**

Η απόδοση των μοντέλων επηρεάζεται σημαντικά από τον αριθμό των αντικειμένων που χρησιμοποιούνται σε κάθε δοκιμή. Όπως φαίνεται από τα αποτελέσματα, το DNN διατηρεί σταθερά υψηλή ακρίβεια ανεξαρτήτως του πλήθους των αντικειμένων, με ελαφρά μείωση όταν ο αριθμός των αντικειμένων αυξάνεται δραματικά. Αντίθετα, το RNN παρουσιάζει μεγαλύτερη μείωση στην ακρίβεια καθώς ο αριθμός των αντικειμένων αυξάνεται, γεγονός που υποδηλώνει ότι το μοντέλο δυσκολεύεται να διαχειριστεί μεγαλύτερα σύνολα δεδομένων.

Το DQN, παρόλο που αρχικά εμφανίζει χαμηλότερη ακρίβεια από το DNN, παρουσιάζει βελτίωση όταν αυξάνεται ο αριθμός των αντικειμένων, φτάνοντας σε υψηλότερα επίπεδα απόδοσης για μεγαλύτερα σύνολα δεδομένων. Αυτό δείχνει ότι το DQN μπορεί να προσαρμοστεί καλύτερα σε προβλήματα μεγαλύτερης πολυπλοκότητας.

**Επίδραση της Πολυπλοκότητας του Προβλήματος**

Η πολυπλοκότητα του προβλήματος, που μπορεί να εκφραστεί μέσω της σχέσης μεταξύ του αριθμού των αντικειμένων και της χωρητικότητας του σακιδίου, επηρεάζει επίσης την απόδοση των μοντέλων. Το DNN δείχνει να αντιμετωπίζει την πολυπλοκότητα με επιτυχία, επιτυγχάνοντας υψηλή ακρίβεια και χαμηλή απώλεια ακόμα και σε πιο σύνθετα προβλήματα.

Το RNN, από την άλλη πλευρά, δείχνει να δυσκολεύεται περισσότερο καθώς αυξάνεται η πολυπλοκότητα, γεγονός που επιβεβαιώνεται από την αυξημένη απώλεια και τη μειωμένη ακρίβεια. Το DQN, αν και αρχικά φαίνεται να επηρεάζεται περισσότερο από την πολυπλοκότητα, παρουσιάζει βελτιωμένη απόδοση σε υψηλότερα επίπεδα πολυπλοκότητας, πιθανώς λόγω της ικανότητάς του να μάθει στρατηγικές λήψης αποφάσεων σε περιβάλλοντα με αυξημένη πολυπλοκότητα.

**Συμπεράσματα**

Η ανάλυση της απόδοσης με βάση τα δεδομένα εισόδου αποκαλύπτει ότι το DNN είναι το πιο σταθερό και αποδοτικό μοντέλο ανεξαρτήτως των χαρακτηριστικών των δεδομένων. Το RNN, ενώ είναι αποτελεσματικό για προβλήματα με λιγότερα δεδομένα, φαίνεται να υστερεί καθώς η πολυπλοκότητα αυξάνεται. Το DQN, παρότι αρχικά φαίνεται λιγότερο αποδοτικό, δείχνει βελτιωμένη απόδοση σε πιο σύνθετα προβλήματα, γεγονός που το καθιστά μια καλή επιλογή για συγκεκριμένα σενάρια.

# Συμπεράσματα

Η επίλυση του προβλήματος του σακιδίου με τη χρήση νευρωνικών δικτύων και αλγορίθμων βαθιάς μάθησης αποτελεί μια σημαντική πρόκληση που συνδυάζει τις απαιτήσεις της συνδυαστικής βελτιστοποίησης με τις δυνατότητες της μηχανικής μάθησης. Ενώ οι παραδοσιακές μέθοδοι βελτιστοποίησης, όπως ο δυναμικός προγραμματισμός και οι άπληστες προσεγγίσεις, μπορεί να αποδίδουν καλά για μικρής κλίμακας περιπτώσεις του προβλήματος, τα νευρωνικά δίκτυα προσφέρουν μια ευέλικτη και καθοδηγούμενη από δεδομένα προσέγγιση που μπορεί να προσαρμοστεί σε πιο σύνθετα σενάρια.

Στην παρούσα εργασία, αναπτύχθηκαν και αξιολογήθηκαν τρία διαφορετικά μοντέλα: το Βαθύ Νευρωνικό Δίκτυο (DNN), το Επαναλαμβανόμενο Νευρωνικό Δίκτυο (RNN) και το μοντέλο Deep Q-Learning (DQN). Τα αποτελέσματα έδειξαν ότι το DNN υπερείχε σε ακρίβεια, ιδίως σε προβλήματα μικρότερης κλίμακας, παρέχοντας υψηλή ακρίβεια που έφτασε έως και το 99.88% για σύνολα δεδομένων με 50 αντικείμενα. Αντίθετα, το RNN παρουσίασε αξιοσημείωτες επιδόσεις, αν και χαμηλότερες από το DNN, ενώ το DQN απέδειξε την ευελιξία και την ταχύτητά του σε μεγαλύτερα προβλήματα, με ακρίβεια που κυμαινόταν από 76.28% έως 90.32%, ανάλογα με τον όγκο των δεδομένων.

Η ανάπτυξη των μοντέλων βασίστηκε σε μια προσεκτική διαδικασία ρύθμισης των υπερπαραμέτρων, όπου χρησιμοποιήθηκε grid search για τον προσδιορισμό των βέλτιστων ρυθμίσεων. Τα αποτελέσματα της ρύθμισης επιβεβαίωσαν τη σημασία της σωστής επιλογής των υπερπαραμέτρων για την επίτευξη βέλτιστων επιδόσεων, με το DNN να εμφανίζει την καλύτερη συμπεριφορά όταν οι υπερπαράμετροι του ήταν προσαρμοσμένοι στο εκάστοτε πρόβλημα.

Τα νευρωνικά δίκτυα προσφέρουν πολλά πλεονεκτήματα στην επίλυση του προβλήματος του σακιδίου. Εκτός από την ικανότητά τους να μαθαίνουν πολύπλοκα μοτίβα και σχέσεις εντός των δεδομένων, μπορούν να προσαρμοστούν σε μεγαλύτερες και πιο πολύπλοκες περιπτώσεις του προβλήματος, όπου οι παραδοσιακοί αλγόριθμοι μπορεί να αποτυγχάνουν ή να είναι αναποτελεσματικοί. Επιπλέον, τα νευρωνικά δίκτυα παρέχουν ευέλικτες αναπαραστάσεις εισόδου και μπορούν να γενικεύσουν καλά σε παραλλαγές του προβλήματος του σακιδίου, επιτρέποντας την αυτοματοποιημένη λήψη αποφάσεων για νέες περιπτώσεις του προβλήματος.

Η παρούσα έρευνα δεν είχε ως στόχο την ανάπτυξη μιας πρωτοποριακής λύσης, αλλά την αντιμετώπιση ορισμένων κενών στην τρέχουσα βιβλιογραφία που αφορούν τη μάθηση σε προβλήματα συνδυαστικής βελτιστοποίησης. Τα αποτελέσματα των διαφόρων στρατηγικών βελτιστοποίησης που εφαρμόστηκαν στο πρόβλημα του σακιδίου ήταν αξιοσημείωτα, με το DNN να καταλαμβάνει την κορυφαία θέση ως η πιο αποτελεσματική στρατηγική. Ωστόσο, τα ευρήματα υποδεικνύουν ότι και τα άλλα μοντέλα έχουν σημαντικές εφαρμογές, ανάλογα με τις συγκεκριμένες απαιτήσεις του προβλήματος και τον διαθέσιμο υπολογιστικό πόρο.

Συνολικά, η χρήση νευρωνικών δικτύων για το πρόβλημα του σακιδίου αποδεικνύεται μια ισχυρή και αποτελεσματική προσέγγιση, η οποία μπορεί να εφαρμοστεί ευρέως σε διάφορες εκδοχές του προβλήματος, επιδεικνύοντας τη δυναμική της βαθιάς μάθησης στην επίλυση σύνθετων υπολογιστικών προκλήσεων.

# Μελλοντική Έρευνα

Η παρούσα έρευνα, αν και παρείχε σημαντικά ευρήματα σχετικά με τη χρήση νευρωνικών δικτύων για την επίλυση του προβλήματος του σακιδίου, αφήνει ανοιχτά πολλά πεδία για περαιτέρω διερεύνηση και βελτίωση. Οι προτάσεις για μελλοντική έρευνα περιλαμβάνουν:

1. **Εξερεύνηση πιο προηγμένων αρχιτεκτονικών νευρωνικών δικτύων:** Ενώ στην παρούσα εργασία χρησιμοποιήθηκαν βασικές αρχιτεκτονικές όπως τα DNN, RNN και DQN, η μελλοντική έρευνα θα μπορούσε να επικεντρωθεί σε πιο προηγμένες αρχιτεκτονικές όπως τα Transformer Networks ή τα Graph Neural Networks (GNN). Αυτά τα δίκτυα έχουν αποδείξει την αξία τους σε προβλήματα με δομή παρόμοια με αυτή του σακιδίου και θα μπορούσαν να βελτιώσουν περαιτέρω την ακρίβεια και την αποδοτικότητα.
2. **Ενσωμάτωση τεχνικών μεταφοράς μάθησης (Transfer Learning):** Ένας άλλος τομέας έρευνας θα μπορούσε να είναι η χρήση μεταφοράς μάθησης για την προσαρμογή προϋπάρχοντων μοντέλων σε νέες περιπτώσεις του προβλήματος του σακιδίου. Η μεταφορά μάθησης θα μπορούσε να επιτρέψει τη χρήση πληροφοριών που έχουν μάθει τα μοντέλα από διαφορετικά αλλά συναφή προβλήματα, μειώνοντας το χρόνο εκπαίδευσης και βελτιώνοντας την απόδοση.
3. **Αντιμετώπιση πιο πολύπλοκων εκδοχών του προβλήματος του σακιδίου:** Μελλοντικές έρευνες θα μπορούσαν να εστιάσουν σε πιο πολύπλοκες εκδοχές του προβλήματος, όπως το Multidimensional Knapsack Problem (MKP) ή το Quadratic Knapsack Problem (QKP). Οι παραλλαγές αυτές παρουσιάζουν μεγαλύτερες προκλήσεις και θα μπορούσαν να δοκιμάσουν τα όρια των νευρωνικών δικτύων.
4. **Διερεύνηση συνδυαστικών προσεγγίσεων:** Η ενσωμάτωση νευρωνικών δικτύων με παραδοσιακές μεθόδους βελτιστοποίησης ή ευρετικές στρατηγικές θα μπορούσε να προσφέρει υβριδικά μοντέλα που εκμεταλλεύονται τα πλεονεκτήματα και των δύο προσεγγίσεων. Τέτοιες υβριδικές λύσεις μπορεί να είναι πιο ισχυρές και αποτελεσματικές σε διαφορετικά σενάρια.
5. **Βελτιστοποίηση της απόδοσης και της κλιμάκωσης:** Η έρευνα θα μπορούσε να επικεντρωθεί στην περαιτέρω βελτίωση της αποδοτικότητας των μοντέλων, ιδίως σε όρους χρόνου εκπαίδευσης και χρήσης μνήμης, καθώς και στην κλιμάκωση για την επίλυση προβλημάτων μεγαλύτερης κλίμακας. Η χρήση εξειδικευμένου υλικού, όπως οι GPUs ή TPUs, και η παράλληλη εκπαίδευση θα μπορούσαν να αποτελέσουν σημαντικές βελτιώσεις. Ιδιαίτερα για το DQN, αν και έδειξε πολλά υποσχόμενα αποτελέσματα, πιστεύεται ότι η απόδοσή του θα μπορούσε να βελτιωθεί σημαντικά με πιο προηγμένη και λεπτομερή ρύθμιση υπερπαραμέτρων, όπως η βελτιστοποίηση του ρυθμού εκμάθησης, η προσαρμογή του γ-παράγοντα (gamma) και η εξερεύνηση πιο πολύπλοκων στρατηγικών διασύνδεσης μεταξύ των δικτύων πολιτικής και στόχου.
6. **Εξερεύνηση της δυνατότητας γενίκευσης σε διαφορετικούς τομείς:** Η μελλοντική έρευνα θα μπορούσε επίσης να εξετάσει την εφαρμογή των μεθόδων αυτών σε άλλους τομείς όπου το πρόβλημα του σακιδίου ή παρόμοια προβλήματα βελτιστοποίησης είναι κρίσιμα, όπως στη διαχείριση αποθεμάτων, στην κατανομή πόρων ή στη βελτιστοποίηση δικτύων.

Με την εξέλιξη των τεχνικών βαθιάς μάθησης και την αυξανόμενη υπολογιστική ισχύ, υπάρχει η δυνατότητα να αναπτυχθούν πιο εξελιγμένα μοντέλα που μπορούν να αντιμετωπίσουν ολοένα και πιο σύνθετες και μεγάλες περιπτώσεις του προβλήματος του σακιδίου. Η συνδυαστική βελτιστοποίηση μέσω νευρωνικών δικτύων συνεχίζει να είναι ένα πεδίο με σημαντικές προοπτικές και προκλήσεις για τη μελλοντική έρευνα.

# Βιβλιογραφία

[1] S. Gu and T. Hao, "A pointer network based deep learning algorithm for 0–1 knapsack problem," in 2018 Tenth International Conference on Advanced Computational Intelligence (ICACI), Mar. 2018, pp. 473-477.

[2] V. D. Blondel and J. N. Tsitsiklis, "A survey of computational complexity results in systems and control," Automatica, vol. 36, no. 9, pp. 1249-1274, 2000.

[3] S. A. Cook, "The complexity of theorem-proving procedures," in Proceedings of the third annual ACM symposium on Theory of computing (STOC '71), ACM, New York, NY, USA, 1971, pp. 151–158.

[4] S. Martello and P. Toth, Knapsack Problems: Algorithms and Computer Implementations, John Wiley & Sons, 1990.

[5] H. Kellerer, U. Pferschy, and D. Pisinger, Knapsack Problems, Springer, 2004.

[6] Ibarra, R., & Kim, D. (1975). "Fast Approximation Algorithms for Knapsack Problems." Journal of the ACM, 22(1), 80-97.

[7] Glover, F., & Laguna, M. (1997). Tabu Search. Kluwer Academic Publishers.

[8] Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. University of Michigan Press.

[9] Wolsey, L. A. (1998). Integer Programming. Wiley-Interscience.

[10] Balas, E. (1975). "Branch-and-Cut Algorithms for the Knapsack Problem." Mathematical Programming, 3(3), 295-308.

[11] Pisinger, D. (2005). "An Efficient Algorithm for the 0–1 Knapsack Problem." Computers & Operations Research, 32(3), 597-607.

[12] F. Chollet, Deep Learning with Python, Manning Publications Co., 2018.

[13] C. C. Aggarwal, Neural Networks and Deep Learning, vol. 10, no. 978, Springer, 2018.

[14] Nielsen, Michael A. Neural networks and deep learning. Vol. 25. San Francisco, CA, USA: Determination Press, 2015.

[15] H. I. Suk, "An introduction to neural networks and deep learning," in Deep Learning for Medical Image Analysis, Academic Press, 2017, pp. 3-24.

[16] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, p. 436, 2015.

[17] F. Yang, T. Jin, T.-Y. Liu, X. Sun, and J. Zhang, "Boosting Dynamic Programming with Neural Networks for Solving NP-hard Problems," in \*Asian Conference on Machine Learning\*, 2018, pp. 726–739.

[18] S. Xu, S. S. Panwar, M. S. Kodialam, and T. V. Lakshman, "Deep Neural Network Approximated Dynamic Programming for Combinatorial Optimization," in \*AAAI Conference on Artificial Intelligence\*, 2020, pp. 1684–1691.

[19] M. Lombardi and M. Milano, "Boosting combinatorial problem modeling with machine learning," in \*Proceedings of the 27th International Joint Conference on Artificial Intelligence\*, 2018, pp. 5472–5478.

[20] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer Networks," in Advances in Neural Information Processing Systems 28, C. Cortes, N. D. Lawrence, and D. D. Lee, Eds., 2015.

[21] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," arXiv:1611.09940, 2016. [Online]. Available: http://arxiv.org/abs/1611.09940.

[22] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in \*Advances in Neural Information Processing Systems 30\*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., 2017, pp. 6348–6358.

[23] A. Nowak, S. Villar, A. S. Bandeira, and J. Bruna, "Revised note on learning algorithms for quadratic assignment with graph neural networks," arXiv:1706.07450, 2017.

[24] P. Emami and S. Ranka, "Learning permutations with Sinkhorn policy gradient," arXiv:1805.07010, 2018.

[25] W. Kool, H. Van Hoof, M. Welling, “Attention, Learn to Solve Routing Problems!”, arXiv:1803.08475, 2019

[26] H. A. Nomer, K. A. Alnowibet, A. Elsayed, and A. W. Mohamed, "Neural knapsack: A neural network based solver for the knapsack problem," \*IEEE Access\*, vol. 8, pp. 224200-224210, 2020.

[27] Kwon S, Choi H, Park S. "Solving bilevel knapsack problem using graph neural networks." arXiv preprint arXiv:2211.13436. 2022 Nov 24.

[28] R. Mansi, C. Alves, J. Val´erio de Carvalho, and S. Hanafi. "An exact algorithm for bilevel 0-1 knapsack problems." Mathematical Problems in Engineering, 2012, 2012.

[29] Yildiz B. "Reinforcement learning using fully connected, attention, and transformer models in knapsack problem solving." Concurrency Computat Pract Exper. 2022; 34(9). https://doi.org/10.1002/cpe.6509

[30] A. Kaushik, M. Alizadeh, O. Waqar, and H. Tabassum, "Deep Unsupervised Learning for Generalized Assignment Problems: A Case-Study of User-Association in Wireless Networks," 2021 IEEE International Conference on Communications Workshops (ICC Workshops), Montreal, QC, Canada, 2021, pp. 1-6, doi: 10.1109/ICCWorkshops50388.2021.9473497.

[31] J. Dong, G. Zhang, B. Luo, Q. Yang, D. Guo, H. Rong, M. Zhu, K. Zhou, "A distributed adaptive optimization spiking neural P system for approximately solving combinatorial optimization problems," Information Sciences, Volume 596, 2022, Pages 1-14, ISSN 0020-0255, https://doi.org/10.1016/j.ins.2022.03.007.

[32] C. Hertrich, M. Skutella, "Provably Good Solutions to the Knapsack Problem via Neural Networks of Bounded Size," arXiv preprint arXiv:2005.14105. 2020.

[33] Zhang, B., Yao, Y., Kan, H.K., et al. "A GAN-based genetic algorithm for solving the 3D bin packing problem." Sci Rep 14, 7775 (2024). https://doi.org/10.1038/s41598-024-56699-7

# ΠΑΡΑΡΤΗΜΑ Α: Κώδικας Εφαρμογής

# 

# ΠΑΡΑΡΤΗΜΑ Β: Κώδικες Βελτιστοποίησης Υπερπαραμέτρων Μοντέλων και Αυτοματοποιημένων Δοκιμών

