ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Δ.Π.Μ.Σ. ΕΠΙΣΤΗΜΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

Εξαμηνιαία Εργασία Μαθήματος

"Προγραμματιστικά Εργαλεία και Τεχνολογίες για Επιστήμη Δεδομένων"

Αλγοριθμική Στρατηγική Συναλλαγών σε Σενάριο «Ταξιδιού στον Χρόνο»

Αντώνιος Μπαροτσάκης
Αριθμός Μητρώου : 03400260
Μεταπτυχιακός Φοιτητής Ε.ΔΕ.ΜΜ

Ιανουάριος 2025

0 Περίληψη

Η παρούσα εργασία παρουσιάζει την ανάπτυξη και υλοποίηση μιας σειράς αλγοριθμικών στρατηγικών για την επίλυση του υποθετικού προβλήματος "Time-Travel Trading". Ο στόχος είναι η μεγιστοποίηση του οικονομικού κέρδους ξεκινώντας με κεφάλαιο ενός δολαρίου το 1960, αξιοποιώντας ιστορικά δεδομένα χρηματιστηριακών συναλλαγών και υπό συγκεκριμένους περιορισμούς, όπως το κόστος προμήθειας και ο ημερήσιος όγκος συναλλαγών. Η προσέγγιση που ακολουθήθηκε είναι εξελικτική: ξεκινώντας από έναν απλό άπληστο αναδρομικό αλγόριθμο, ο οποίος αποδείχθηκε μυωπικός και υποβέλτιστος, η στρατηγική εξελίχθηκε σε μια προηγμένη υλοποίηση (`extra greedy`) που ενσωματώνει έναν καινοτόμο μηχανισμό "look-back". Αυτός ο μηχανισμός επιτρέπει στον αλγόριθμο να εκμεταλλεύεται αναδρομικά κερδοφόρες ευκαιρίες στο παρελθόν πριν δεσμευτεί σε μια μελλοντική κίνηση, διορθώνοντας έτσι την απληστία του. Οι τελικές στρατηγικές, εκτελούμενες σε ετήσια και μηνιαία βάση, αποδείχθηκαν εξαιρετικά αποτελεσματικές, επιτυγχάνοντας τελικό κεφάλαιο ύψους 4.1 δισεκατομμυρίων δολαρίων για το σενάριο των 1.000 κινήσεων και 232.8 δισεκατομμυρίων δολαρίων για το σενάριο του 1.000.000 κινήσεων, επιβεβαιώνοντας την ανωτερότητα της εξελιγμένης μεθόδου.

Περιεχόμενα

Κατάλογος σχημάτων

1 Εισαγωγή

Το πρόβλημα που εξετάζουμε αφορά τη βελτιστοποίηση κέρδους από χρηματιστηριακές συναλλαγές, με στόχο τη μέγιστη αξιοποίηση ενός περιορισμένου κεφαλαίου και αριθμού κινήσεων. Η πρόκληση είναι να αναπτύξουμε μια στρατηγική που θα επιλέγει τις βέλτιστες συναλλαγές (αγορές και πωλήσεις) σε ένα περιβάλλον πολλαπλών περιορισμών, λαμβάνοντας υπόψη τα ιστορικά δεδομένα των τιμών μετοχών.

1.1 Περιγραφή του Προβλήματος

Το πρόβλημα βασίζεται σε ένα σύνολο δεδομένων που περιλαμβάνει τις ημερήσιες τιμές μετοχών (`Open`, `High`, `Low`, `Close`) και τον όγκο συναλλαγών (`Volume`) για διαφορετικές εταιρείες σε χρονική διάρκεια πολλών δεκαετιών.

Οι κύριοι στόχοι είναι:

- Να επιλέξουμε κινήσεις (αγορά/πώληση) που προσπαθούν να μεγιστοποιήσουν το τελικό κέρδος.
- Να τηρήσουμε αυστηρά τους περιορισμούς που ορίζονται στο πρόβλημα.

1.2 Βασικοί Περιορισμοί και Κανόνες Συναλλαγών

Οι περιορισμοί που πρέπει να ληφθούν υπόψη κατά την ανάπτυξη της στρατηγικής είναι οι εξής:

- **Αρχικό Κεφάλαιο**: Η προσομοίωση ξεκινά με αρχικό κεφάλαιο 1 δολαρίου.
- **Προμήθεια**: Κάθε αγορά ή πώληση μετοχών επιβαρύνεται με προμήθεια 1% επί της αξίας της συναλλαγής.
- **Περιορισμός Όγκου**: Το συνολικό πλήθος των μετοχών μιας εταιρείας που αγοράζονται ή πωλούνται σε μία ημέρα δεν πρέπει να ξεπερνά το 10% του συνολικού όγκου συναλλαγών της μετοχής για την ίδια ημέρα.
- **Εγκυρότητα Συναλλαγών**: Πριν την εκτέλεση οποιασδήποτε συναλλαγής, πρέπει να υπάρχει επαρκές χρηματικό υπόλοιπο για τις αγορές και επαρκής αριθμός μετοχών στην κατοχή μας για τις πωλήσεις.
- **Όριο Κινήσεων**: Το πρόβλημα εξετάζεται σε δύο σενάρια:
  - **Σενάrio "Small"**: Περιορισμός σε το πολύ 1.000 συναλλαγές.
  - **Σενάριο "Large"**: Περιορισμός σε το πολύ 1.000.000 συναλλαγές.

1.3 Επισκόπηση της Προτεινόμενης Στρατηγικής

Η προσέγγιση που ακολουθήθηκε για την επίλυση του προβλήματος είναι **εξελικτική**. Αντί για την αναζήτηση μιας ενιαίας, βέλτιστης λύσης, η οποία είναι υπολογιστικά ανέφικτη, η στρατηγική αναπτύχθηκε σε στάδια, ξεκινώντας από έναν απλό, άπληστο αναδρομικό αλγόριθμο. Ο αρχικός αυτός αλγόριθμος, αν και λειτουργικός, επέδειξε μια «μυωπική» συμπεριφορά, εστιάζοντας σε λίγες, άμεσα κερδοφόρες κινήσεις και αγνοώντας το ευρύτερο δυναμικό του προβλήματος.

Για την αντιμετώπιση αυτών των περιορισμών, αναπτύχθηκε μια δεύτερη, πιο προηγμένη στρατηγική, η οποία ενσωματώνει έναν **μηχανισμό "look-back"**. Αυτή η τεχνική επιτρέπει στον αλγόριθμο, πριν δεσμευτεί σε μια κερδοφόρα μελλοντική κίνηση, να «κοιτάξει πίσω» στο χρόνο και να εκμεταλλευτεί μικρότερες, ενδιάμεσες ευκαιρίες που είχαν προηγουμένως αγνοηθεί.

Στις επόμενες ενότητες, θα αναλυθούν λεπτομερώς και οι δύο αυτές στρατηγικές. Θα παρουσιαστεί η λογική πίσω από κάθε αλγόριθμο, οι παρατηρήσεις από την εκτέλεσή τους, και τα τελικά αποτελέσματα που επιτεύχθηκαν για τα σενάρια "Small" και "Large" αντίστοιχα.

2 Προεπεξεργασία και Διαχείριση Δεδομένων

Η ποιότητα των δεδομένων είναι θεμελιώδους σημασίας για την επιτυχία οποιασδήποτε αλγοριθμικής στρατηγικής. Τα ακατέργαστα δεδομένα συχνά περιέχουν σφάλματα, ελλιπείς εγγραφές ή παράλογες τιμές που μπορούν να οδηγήσουν σε λανθασμένες αποφάσεις και αναξιόπιστα αποτελέσματα. Για τον λόγο αυτό, αναπτύχθηκε μια αυστηρή και πολυεπίπεδη διαδικασία προεπεξεργασίας, η οποία υλοποιείται στο module `src/data_preprocessor.py`, με στόχο τη δημιουργία ενός καθαρού και αξιόπιστου συνόλου δεδομένων.

2.1 Φιλτράρισμα και Επικύρωση Δεδομένων

Η διαδικασία καθαρισμού εφαρμόζεται σε κάθε αρχείο μετοχής ξεχωριστά και περιλαμβάνει τα παρακάτω βήματα:

- **Αξιολόγηση Αξιοπιστίας Μετοχής:** Πριν την επεξεργασία, κάθε μετοχή αξιολογείται ως προς τη συνολική της αξιοπιστία. Υπολογίζεται το ποσοστό των ημερών που περιέχουν μηδενικές τιμές στις βασικές στήλες. Εάν το ποσοστό αυτό υπερβαίνει ένα όριο (10%), η μετοχή θεωρείται αναξιόπιστη και τα δεδομένα της απορρίπτονται στο σύνολό τους.

- **Φιλτράρισμα σε Επίπεδο Εγγραφής:** Για τις μετοχές που κρίνονται αξιόπιστες, ακολουθεί φιλτράρισμα σε επίπεδο γραμμής (ημέρας). Μια εγγραφή διατηρείται μόνο εάν πληροί **ταυτόχρονα** τις παρακάτω συνθήκες:

  1. Όλες οι τιμές (`Open`, `High`, `Low`, `Close`, `Volume`) είναι αυστηρά θετικές.
  2. Οι τιμές έχουν λογική συνοχή (π.χ., `High` $\geq$ `Open` και `Low` $\leq$ `Close`).

  Αυτή η προσέγγιση διασφαλίζει τη διατήρηση του μέγιστου δυνατού όγκου έγκυρων δεδομένων, απορρίπτοντας μόνο τις μεμονωμένες, προβληματικές εγγραφές.

2.2 Ανίχνευση και Αφαίρεση Ακραίων Τιμών (Outliers)

Ως επιπλέον βήμα, εφαρμόζεται τεχνική για την εξάλειψη ακραίων ημερήσιων διακυμάνσεων (outliers) που μπορεί να οφείλονται σε σφάλματα καταχώρησης. Η διαδικασία βασίζεται στον κανόνα των τριών τυπικών αποκλίσεων (3-sigma rule), ο οποίος εφαρμόζεται στο ημερήσιο εύρος τιμών (`Range` = `High` - `Low`) κάθε μετοχής:

- Για κάθε μετοχή, υπολογίζεται ο μέσος όρος (μ) και η τυπική απόκλιση (σ) του `Range`.
- Διατηρούνται μόνο οι εγγραφές των οποίων το `Range` βρίσκεται εντός του διαστήματος $[\mu - 3\sigma, \mu + 3\sigma]$.

2.3 Τελική Διαμόρφωση και Υπολογισμός Χαρακτηριστικών

Μετά την ολοκλήρωση της διαδικασίας φιλτραρίσματος για όλες τις έγκυρες μετοχές, υπολογίζονται οι απαραίτητες για τη στρατηγική στήλες, όπως το `Max_Quantity` (10% του `Volume`) και το σύμβολο (`Stock`) της μετοχής. Τέλος, τα επιμέρους «καθαρά» DataFrames συνενώνονται σε ένα ενιαίο, συνολικό DataFrame. Το κρισιμότερο βήμα είναι η **ταξινόμηση** αυτού του DataFrame με βάση την ημερομηνία, ώστε οι συναλλαγές να εξετάζονται με τη σωστή χρονολογική σειρά κατά την εκτέλεση της στρατηγικής.

3 Σενάριο "Small": Στρατηγική greedy_trading_by_year

Για το σενάριο "Small", με τον περιορισμό των 1.000 κινήσεων, ο στόχος ήταν η ανάπτυξη μιας στρατηγικής που θα εκμεταλλευόταν αποτελεσματικά τις ευκαιρίες σε όλο το εύρος του χρόνου, χωρίς να εξαντλείται πρόωρα σε λίγες, μεγάλες συναλλαγές. Η λύση που υιοθετήθηκε βασίζεται σε έναν θεμελιώδη άπληστο αλγόριθμο, ο οποίος όμως εφαρμόζεται με έναν δομημένο τρόπο, σε ετήσια βάση.

3.1 Ο Θεμελιώδης Άπληστος Αλγόριθμος: greedy_trading_recursive

Η καρδιά της στρατηγικής είναι ο αναδρομικός αλγόριθμος greedy_trading_recursive, ο οποίος υλοποιείται στο module src/strategies.py. Η λογική του είναι απλή και άπληστη:

1. Σε κάθε βήμα, ο αλγόριθμος εξετάζει όλες τις πιθανές ενδοημερήσιες συναλλαγές (buy-open/sell-high και buy-low/sell-close) σε όλες τις μετοχές και για όλες τις μελλοντικές ημερομηνίες, με βάση το τρέχον διαθέσιμο κεφάλαιο.
2. Επιλέγει τη **μία και μοναδική συναλλαγή** που υπόσχεται το μέγιστο δυνατό κέρδος από όλες τις διαθέσιμες επιλογές.
3. Εκτελεί αυτή τη συναλλαγή, ενημερώνει το κεφάλαιο και καλεί αναδρομικά τον εαυτό του για το υπόλοιπο χρονικό διάστημα.
4. Η αναδρομή τερματίζει όταν δεν υπάρχουν άλλες κερδοφόρες κινήσεις.

Ωστόσο, η απευθείας εφαρμογή αυτού του αλγορίθμου σε ολόκληρο το χρονικό εύρος των δεδομένων αποδείχθηκε προβληματική. Ο αλγόριθμος επέδειξε μια εξαιρετικά «μυωπική» συμπεριφορά: εντόπιζε μια πολύ κερδοφόρα συναλλαγή στα πρώτα έτη, την εκτελούσε, και στη συνέχεια δυσκολευόταν να βρει εξίσου ελκυστικές ευκαιρίες, τερματίζοντας μετά από ελάχιστες κινήσεις (ενδεικτικά, 14 κινήσεις με κέρδος $1.6 εκατ.).

3.2 Η Λύση της Ετήσιας Επανεκκίνησης: greedy_trading_by_year

Για να αντιμετωπιστεί η παραπάνω αδυναμία, αναπτύχθηκε η συνάρτηση-«περιτύλιγμα» run_small_scenario στο module src/trading_engine.py. Αυτή η συνάρτηση εφαρμόζει τον αλγόριθμο greedy_trading_recursive **ξεχωριστά για κάθε έτος**.

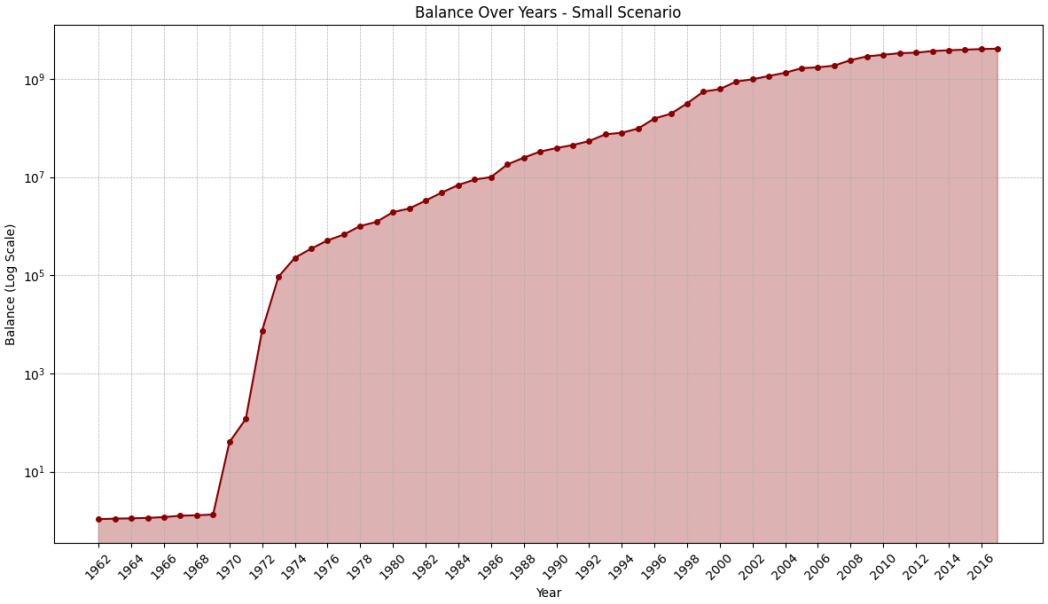
Η λογική είναι η εξής: στο τέλος κάθε έτους, το κεφάλαιο που έχει συγκεντρωθεί γίνεται το αρχικό κεφάλαιο για την έναρξη του επόμενου έτους. Αυτή η «ετήσια επανεκκίνηση» αναγκάζει τον αλγόριθμο να αναζητήσει τις βέλτιστες τοπικές ευκαιρίες εντός κάθε έτους, αντί να δεσμεύεται από μια παγκόσμια, αλλά υποβέλτιστη, μακροπρόθεσμη επιλογή. Το αποτέλεσμα είναι η αύξηση του αριθμού των συναλλαγών και η καλύτερη αξιοποίηση της ανατοκιζόμενης αξίας του κεφαλαίου.

3.3 Αποτελέσματα Σεναρίου "Small"

Η εφαρμογή της στρατηγικής greedy_trading_by_year οδήγησε σε εξαιρετικά αποτελέσματα, παραμένοντας εντός του ορίου των 1.000 κινήσεων.

- **Συνολικές Κινήσεις**: 806
- **Τελικό Κεφάλαιο**: $4,110,278,133.32

Το Σχήμα 1 απεικονίζει την εκθετική αύξηση του κεφαλαίου σε λογαριθμική κλίμακα, αναδεικνύοντας την αποτελεσματικότητα της στρατηγικής σε βάθος χρόνου.



Σχήμα 1: Εξέλιξη του κεφαλαίου σε λογαριθμική κλίμακα για το σενάριο "Small".

4 Σενάριο "Large": Στρατηγική extra_greedy_trading_by_month

Για το σενάριο "Large", με το όριο του 1.000.000 κινήσεων, ο απλός άπληστος αλγόριθμος ήταν ανεπαρκής. Απαιτήθηκε η ανάπτυξη μιας πιο εξελιγμένης στρατηγικής που θα μπορούσε να εκμεταλλευτεί τον τεράστιο αριθμό επιτρεπόμενων συναλλαγών για να μεγιστοποιήσει το κέρδος. Η λύση που αναπτύχθηκε είναι ο αλγόριθμος extra_greedy_trading_recursive, ο οποίος εισάγει έναν καινοτόμο μηχανισμό αναδρομής στο παρελθόν, ή "look-back".

4.1 Ο Μηχανισμός "Look-back" του extra_greedy_trading_recursive

Σε αντίθεση με τον απλό άπληστο αλγόριθμο, ο extra_greedy_trading_recursive (υλοποιημένος στο src/strategies.py) δεν εκτελεί αμέσως την πιο κερδοφόρα κίνηση που εντοπίζει. Αντιθέτως, ακολουθεί μια πιο σύνθετη λογική, την οποία ονομάζουμε μηχανισμό "look-back":

1. **Εντοπισμός Βέλτιστης "Παρούσας" Κίνησης**: Ο αλγόριθμος αρχικά εντοπίζει την πιο κερδοφόρα ενδοημερήσια συναλλαγή σε ολόκληρο το διαθέσιμο χρονικό ορίζοντα. Ας την ονομάσουμε «παρούσα» κίνηση, καθώς αποτελεί το σημείο αναφοράς για τα επόμενα βήματα.

2. **Προσωρινή Δέσμευση Κεφαλαίου**: Υπολογίζει το κόστος αυτής της κίνησης και το αφαιρεί προσωρινά από το διαθέσιμο κεφάλαιο.

3. **Αναδρομή στο Παρελθόν**: Πριν εκτελέσει την «παρούσα» κίνηση, ο αλγόριθμος καλεί αναδρομικά τον εαυτό του, αλλά αυτή τη φορά για το **χρονικό διάστημα που προηγείται** της «παρούσας» κίνησης. Αυτή η αναδρομική κλήση είναι η καρδιά του μηχανισμού.

Το κρίσιμο σημείο που διαφοροποιεί αυτή τη στρατηγική είναι η **βαθιά φύση αυτής της αναδρομής**. Η κλήση για το παρελθόν θα εκτελέσει την ίδια ακριβώς λογική: θα βρει την καλύτερη κίνηση στο "παρελθόν", θα δεσμεύσει το κόστος της και θα καλέσει τον εαυτό της για το "παρελθόν του παρελθόντος". Αυτή η διαδικασία δημιουργεί μια αλυσιδωτή, "ζικ-ζακ" εξερεύνηση του χρόνου: ο αλγόριθμος βυθίζεται όλο και πιο βαθιά στο παρελθόν, και καθώς κάθε αναδρομική κλήση ολοκληρώνεται, "αναδύεται" προς το παρόν, εκτελώντας τις κινήσεις που έχει προγραμματίσει με αντίστροφη χρονολογική σειρά.

Αφού ολοκληρωθεί ολόκληρη η αλυσίδα των παρελθοντικών κλήσεων, ο αλγόριθμος επιστρέφει στο αρχικό σημείο αναφοράς και εκτελεί την αρχική, πολύ κερδοφόρα κίνηση που είχε εντοπίσει, πριν συνεχίσει την αναζήτησή του στο μέλλον. Με αυτόν τον τρόπο, ο αλγόριθμος δεν αφήνει καμία σημαντική ευκαιρία κέρδους ανεκμετάλλευτη, εξαντλώντας όλες τις πιθανές διαδρομές στο παρελθόν πριν προχωρήσει μπροστά.

4.2 Διαχείριση Αποδοτικότητας με Δυναμικές Παραμέτρους

Ο μηχανισμός "look-back" είναι ισχυρός, αλλά ενέχει τον κίνδυνο να οδηγήσει σε έναν τεράστιο αριθμό συναλλαγών με αμελητέο κέρδος, σπαταλώντας άσκοπα το όριο των διαθέσιμων κινήσεων. Για την αντιμετώπιση αυτού του φαινομένου, εξετάστηκαν και υλοποιήθηκαν δύο μηχανισμοί δυναμικού ελέγχου.

Ο πρώτος μηχανισμός αφορούσε τον περιορισμό του **πλήθους** των διορθωτικών κινήσεων. Υλοποιήθηκε η συνάρτηση _dynamic_max_pairs στο module src/trading_engine.py, η οποία ρύθμιζε δυναμικά τον μέγιστο επιτρεπόμενο αριθμό παρελθοντικών κινήσεων ανάλογα με το έτος. Η ιδέα ήταν να επιτρέπονται περισσότερες διορθωτικές κινήσεις στα πιο πρόσφατα έτη, όπου οι ευκαιρίες ήταν δυνητικά περισσότερες.

Ο δεύτερος, και τελικά πιο κρίσιμος, μηχανισμός αφορά τον περιορισμό του **κέρδους** των διορθωτικών κινήσεων. Υλοποιήθηκε η συνάρτηση dynamic_minimum_profit στο src/config.py, η οποία θέτει ένα **δυναμικό κατώφλι ελάχιστου κέρδους** για κάθε διορθωτική κίνηση. Το κατώφλι αυτό αυξάνεται ανάλογα με το διαθέσιμο κεφάλαιο: όσο μεγαλύτερο το κεφάλαιο, τόσο μεγαλύτερο το απαιτούμενο κέρδος για να θεωρηθεί μια κίνηση "άξια" εκτέλεσης. Έτσι, ο

αλγόριθμος αποτρέπεται από το να πραγματοποιεί χιλιάδες κινήσεις για κέρδη της τάξης των λίγων δολαρίων όταν το κεφάλαιό του ανέρχεται σε εκατομμύρια.

Μετά από πειραματισμούς, διαπιστώθηκε ότι ο έλεγχος μέσω του ελάχιστου δυναμικού κέρδους (`min_profit`) ήταν πολύ πιο αποτελεσματικός και αποδοτικός στη διαχείριση του ορίου των συναλλαγών. Ως εκ τούτου, στην τελική στρατηγική που παρουσιάζεται, ο μηχανισμός περιορισμού του πλήθους των κινήσεων (`max_past_pairs`) δεν ενεργοποιήθηκε, και ο έλεγχος βασίστηκε αποκλειστικά στο κατώφλι του κέρδους.

4.3 Η Τελική Στρατηγική: Εκτέλεση ανά Μήνα

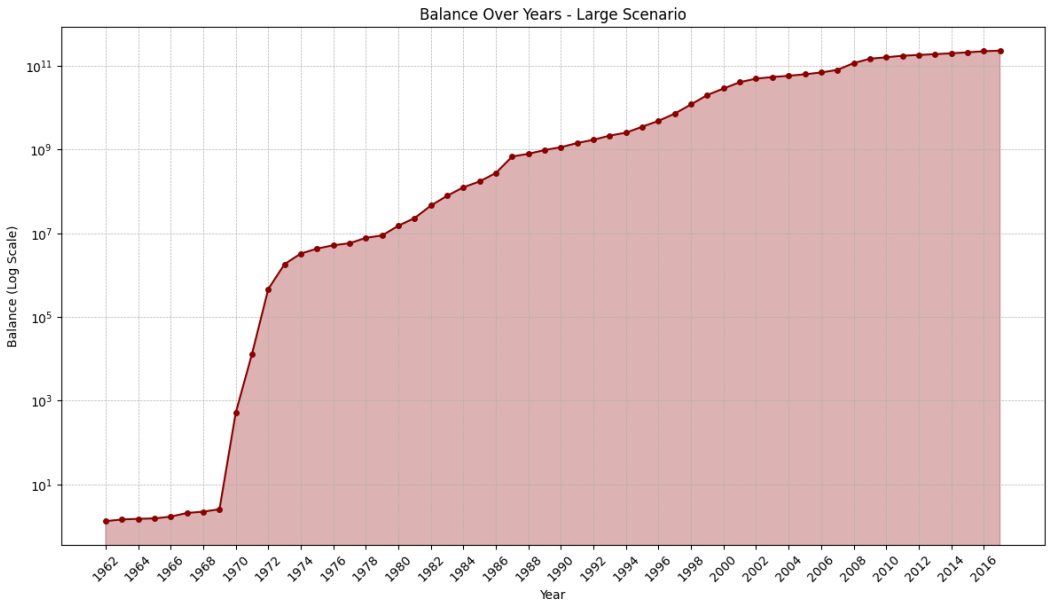
Για τη βέλτιστη δυνατή αξιοποίηση του αλγορίθμου `extra_greedy_trading_recursive`, η εκτέλεσή του γίνεται σε **μηνιαία βάση** μέσω της συνάρτησης `run_large_scenario`. Αυτή η τμηματοποίηση προσφέρει έναν ιδανικό συνδυασμό: παρέχει ένα αρκετά μεγάλο χρονικό παράθυρο για τον εντοπισμό καλών ευκαιριών, ενώ παράλληλα η συχνή επανεκκίνηση του αλγορίθμου διασφαλίζει ότι προσαρμόζεται συνεχώς στις μεταβαλλόμενες συνθήκες της αγοράς και του κεφαλαίου.

4.4 Αποτελέσματα Σεναρίου "Large"

Η τελική στρατηγική οδήγησε σε εντυπωσιακά αποτελέσματα, αξιοποιώντας σχεδόν πλήρως το όριο των διαθέσιμων κινήσεων.

- **Συνολικές Κινήσεις**: 848,208
- **Τελικό Κεφάλαιο**: $232,808,360,136.26

Το Σχήμα 2 απεικονίζει την πορεία του κεφαλαίου, η οποία, όπως και στο σενάριο "Small", ακολουθεί μια εκθετική πορεία ανάπτυξης.



Σχήμα 2: Εξέλιξη του κεφαλαίου σε λογαριθμική κλίμακα για το σενάριο "Large".

5 Υλοποίηση και Δομή Κώδικα

Η μετάβαση από τη θεωρητική στρατηγική σε μια λειτουργική και συντηρήσιμη εφαρμογή απαιτεί μια καλά δομημένη αρχιτεκτονική κώδικα. Το project αναπτύχθηκε με μια **modular προσέγγιση**, όπου κάθε τμήμα της λογικής απομονώνεται στο δικό του Python module. Αυτή η δομή, που παρουσιάζεται στο αποθετήριο GitHub, προσφέρει αυξημένη αναγνωσιμότητα, ευκολία στη συντήρηση και επεκτασιμότητα.

5.1 Αρχιτεκτονική του Project

Η δομή του project είναι οργανωμένη γύρω από έναν κεντρικό εκτελεστή, το `main.py`, και μια βιβλιοθήκη πηγαίου κώδικα, τον φάκελο `src/`.

- **`main.py`**: Αποτελεί το κύριο σημείο εισόδου της εφαρμογής. Χρησιμοποιεί το module `argparse` της Python για να δέχεται παραμέτρους από τη γραμμή εντολών, επιτρέποντας στον χρήστη να επιλέξει το σενάριο (`small` ή `large`) που επιθυμεί να εκτελέσει. Ενορχηστρώνει τη ροή της εκτέλεσης, καλώντας με τη σειρά τα modules για την προεπεξεργασία, την εκτέλεση της στρατηγικής και την τελική επικύρωση.
- **Φάκελος `src/`**: Περιέχει τη βασική λογική του project, οργανωμένη στα παρακάτω modules:
  - `config.py`: Κεντρικό αρχείο ρυθμίσεων που περιέχει όλες τις σταθερές του project, όπως διαδρομές αρχείων, παραμέτρους συναλλαγών (π.χ., ποσοστό προμήθειας) και όρια για την προεπεξεργασία.
  - `data_preprocessor.py`: Υπεύθυνο για την ανάγνωση, τον καθαρισμό, το φιλτράρισμα και την προετοιμασία των ιστορικών δεδομένων.
  - `visualizer.py`: Περιέχει τις συναρτήσεις για τη δημιουργία και αποθήκευση των γραφημάτων.

5.2 Επεξήγηση Βασικών Modules Λογικής

Η καρδιά του project βρίσκεται σε δύο βασικά modules που υλοποιούν την αλγοριθμική λογική:

- **`strategies.py`**: Περιέχει τις "καθαρές" υλοποιήσεις των δύο βασικών αναδρομικών αλγορίθμων, `greedy_trading_recursive` και `extra_greedy_trading_recursive`. Αυτά τα modules δεν έχουν καμία γνώση για το από πού έρχονται τα δεδομένα ή πώς εκτελούνται· περιέχουν αποκλειστικά την αλγοριθμική στρατηγική.
- **`trading_engine.py`**: Λειτουργεί ως ο "εγκέφαλος" της εκτέλεσης. Περιέχει τις συναρτήσεις-«περιτυλίγματα» (`run_small_scenario` και `run_large_scenario`) που καλούν τις στρατηγικές του `strategies.py` με δομημένο τρόπο (ανά έτος ή ανά μήνα) και διαχειρίζονται τη ροή του κεφαλαίου σε βάθος χρόνου.

5.3 Διαδικασία Επαλήθευσης: `validator.py`

Για τη διασφάλιση της ορθότητας των αποτελεσμάτων, αναπτύχθηκε το module `validator.py`. Αυτό το module περιέχει τη συνάρτηση `validate_moves`, η οποία προσομοιώνει την εκτέλεση της λίστας των κινήσεων που παρήγαγε η στρατηγική, ελέγχοντας βήμα προς βήμα ότι όλοι οι κανόνες του προβλήματος τηρούνται αυστηρά:

1. Η χρονολογική σειρά των κινήσεων είναι ορθή.
2. Το διαθέσιμο κεφάλαιο επαρκεί για κάθε αγορά.
3. Ο περιορισμός του όγκου συναλλαγών δεν παραβιάζεται.

Στο τέλος, το τελικό κεφάλαιο που υπολογίζει ο `validator` συγκρίνεται με αυτό που υπολόγισε η στρατηγική. Μια επιτυχής ταυτοποίηση αποτελεί την τελική απόδειξη της ορθότητας της υλοποίησης.

6 Συμπεράσματα και Μελλοντικές Επεκτάσεις

Η παρούσα εργασία ανέδειξε τη δυνατότητα ανάπτυξης εξαιρετικά κερδοφόρων αλγοριθμικών στρατηγιών σε ένα περιβάλλον με τέλεια γνώση του παρελθόντος. Μέσα από μια εξελικτική διαδικασία, από έναν απλό άπληστο αλγόριθμο σε έναν προηγμένο μηχανισμό με "look-back", καταφέραμε να δημιουργήσουμε μια στιβαρή και αποτελεσματική λύση που σέβεται όλους τους περιορισμούς του προβλήματος.

6.1 Σύνοψη Αποτελεσμάτων

Η τελική μεθοδολογία αποδείχθηκε εξαιρετικά επιτυχής και στα δύο σενάρια του προβλήματος.

- Για το σενάrio **"Small"**, η στρατηγική της ετήσιας επανεκκίνησης κατάφερε να μετατρέψει το αρχικό κεφάλαιο του $1 σε πάνω από **$4.1 δισεκατομμύρια**, πραγματοποιώντας 806 κινήσεις.
- Για το σενάριο **"Large"**, η προηγμένη στρατηγική του "look-back" με εκτέλεση ανά μήνα και δυναμικό έλεγχο κέρδους, εκτόξευσε το τελικό κεφάλαιο στα **$232.8 δισεκατομμύρια**, αξιοποιώντας πάνω από 848.000 συναλλαγές.

Η επιτυχής επαλήθευση των αποτελεσμάτων μέσω ενός ανεξάρτητου validator επιβεβαιώνει την ορθότητα και την αξιοπιστία της υλοποίησης. Η modular αρχιτεκτονική κώδικα όχι μόνο διευκόλυνε την ανάπτυξη και τον έλεγχο, αλλά καθιστά το project επεκτάσιμο και εύκολα συντηρήσιμο.

6.2 Πιθανές Βελτιώσεις και Ιδέες για το Μέλλον

Παρά τα εντυπωσιακά αποτελέσματα, υπάρχουν αρκετοί τομείς για περαιτέρω έρευνα και βελτίωση:

- **Εξερεύνηση Νέων Ειδών Συναλλαγών**: Η τρέχουσα υλοποίηση εστιάζει αποκλειστικά σε ενδοημερήσιες συναλλαγές. Μια προφανής επέκταση θα ήταν η ενσωμάτωση μακροπρόθεσμων στρατηγικών (π.χ., αγορά και διακράτηση για ημέρες, μήνες ή και χρόνια), οι οποίες θα μπορούσαν να ξεκλειδώσουν νέες πηγές κέρδους.
- **Βελτιστοποίηση Υπερ-παραμέτρων**: Οι τιμές για τις δυναμικές παραμέτρους, όπως το κατώφλι του ελάχιστου κέρδους, επιλέχθηκαν με βάση πειραματισμούς και ευριστικούς κανόνες. Θα μπορούσαν να εφαρμοστούν πιο εξελιγμένες τεχνικές βελτιστοποίησης (π.χ., Grid Search σε ένα υποσύνολο των δεδομένων) για την εύρεση των μαθηματικά βέλτιστων τιμών.
- **Ενσωμάτωση Περισσότερων Περιορισμών**: Για να γίνει το πρόβλημα ακόμα πιο ρεαλιστικό, θα μπορούσαν να προστεθούν νέοι περιορισμοί, όπως ο αντίκτυπος στην τιμή της μετοχής από μεγάλες συναλλαγές (market impact) ή διαφορετικά κόστη προμήθειας ανάλογα με τον όγκο.

Συνολικά, το project αυτό αποτελεί μια ισχυρή απόδειξη της δύναμης των αλγοριθμικών προσεγγίσεων στην επίλυση σύνθετων προβλημάτων βελτιστοποίησης και θέτει τις βάσεις για πολλές ενδιαφέρουσες μελλοντικές επεκτάσεις.

Α΄ Παράρτημα

Α΄.1 Πηγή Δεδομένων

Τα ιστορικά δεδομένα τιμών και όγκου συναλλαγών για τις μετοχές που χρησιμοποιήθηκαν σε αυτή την εργασία προέρχονται από το σύνολο δεδομένων "Price/Volume Data for all US Stocks & ETFs", το οποίο είναι διαθέσιμο δημόσια στην πλατφόρμα Kaggle.

Σύνδεσμος προς το Dataset: Kaggle: Price/Volume Data for all US Stocks & ETFs

Α΄.2 Αποθετήριο Κώδικα

Ο πλήρης πηγαίος κώδικας της υλοποίησης, συμπεριλαμβανομένων των scripts για την προεπεξεργασία, τις στρατηγικές και την επικύρωση των αποτελεσμάτων, είναι διαθέσιμος στο παρακάτω αποθετήριο (repository) στο GitHub.

Σύνδεσμος προς το Αποθετήριο: github.com/antonisbaro/Algorithmic-Trading-Time-Travel