Router: Jurnal Teknik Informatika dan Terapan
Volume. 4 Nomor. 2 Juni 2026
e-ISSN: 3032-3312; p-ISSN: 3026-3611, Hal. 81-97
DOI: https://doi.org/10.62951/router.v4i2.888
Tersedia: https://journal.aptii.or.id/index.php/Router

Analisis *Trade-off* antara Efisiensi Komputasi dan Solusi pada Optimasi Kombinasi Portofolio Saham Menggunakan Algoritma *Brute force* dan Algoritma *Greedy*

Valentino Juanito[1*], Manuele Jocelyn Blasius[2], Tinaliah[3]
[1-3]Fakultas Ilmu Komputer dan Rekayasa, Universitas Multi Data Palembang, Indonesia
E-mail: valentinojuanito_2327250020@mhs.mdp.ac.id[1],
manuelejocelynblasius_2327250087@mhs.mdp.ac.id[2], tinaliah@mdp.ac.id[3]

**Penulis Korespondensi:* valentinojuanito_2327250020@mhs.mdp.ac.id

Abstract. *In stock investment, determining a portfolio that offers attractive returns while maintaining acceptable risk remains a major challenge. Portfolio optimization algorithms can assist in identifying suitable stock combinations; however, the optimization process becomes increasingly complex as the number of possible combinations grows. This study analyzes the Trade-off between computational efficiency and portfolio solution quality using the Brute force and Greedy algorithms. The dataset consists of historical daily adjusted closing prices from 472 S&P 500 companies during the 2025-2026 period. The methodology includes data preprocessing, annualized return and volatility calculation, covariance matrix construction, and portfolio evaluation using the Sharpe ratio. The results show that the Brute force algorithm consistently achieves higher Sharpe ratios, reaching 4.3619 compared to 3.6470 obtained by the Greedy algorithm for four stock portfolios. However, its runtime increases substantially from 0.0425 seconds for two-stock combinations to 670.7102 seconds for four stock combinations. In contrast, the Greedy algorithm maintains a stable runtime of approximately 0.0015 seconds while producing solutions that remain relatively close to the optimal results. These findings demonstrate a clear Trade-off between computational efficiency and solution quality in portfolio optimization.*

Keywords: *Brute force Algorithm; Greedy Algorithm; Portfolio Optimization; Sharpe Ratio; Trade-Off.*

Abstrak. Dalam investasi saham, tantangannya adalah menentukan kombinasi portofolio yang mampu memberikan tingkat keuntungan yang baik dengan risiko yang terkendali. Oleh karena itu, diperlukan optimasi portofolio berbasis algoritma untuk membantu proses pencarian kombinasi saham yang sesuai. Namun, proses optimasi menjadi semakin kompleks ketika jumlah kombinasi portofolio meningkat karena ruang pencarian solusi bertambah secara kombinatorial sehingga memengaruhi efisiensi komputasi. Penelitian ini bertujuan menganalisis *Trade-off* antara efisiensi komputasi dan kualitas solusi portofolio menggunakan algoritma *Brute force* dan algoritma *Greedy*. *Dataset* yang digunakan berupa data historis 472 emiten S&P 500 berdasarkan harga penutupan harian (*Adjusted Close*) periode 2025-2026. Tahapan penelitian meliputi *preprocessing* data, perhitungan *annualized return*, *annualized volatility*, *covariance matrix*, serta evaluasi performa portofolio menggunakan *Sharpe ratio*. Hasil penelitian menunjukkan bahwa algoritma *Brute force* konsisten menghasilkan *Sharpe ratio* lebih tinggi, mencapai 4,3619 dibandingkan algoritma *Greedy* yang sebesar 3,6470 pada kombinasi empat saham. Namun, *runtime* algoritma *Brute force* melonjak drastis seiring bertambahnya kombinasi, dari 0,0425 detik pada kombinasi dua saham menjadi 670,7102 detik pada kombinasi empat saham. Sebaliknya, algoritma *Greedy* menghasilkan solusi dengan *runtime* yang stabil sekitar 0,0015 detik walaupun *Sharpe Ratio* yang dihasilkan di bawah *Brute force*, kualitas solusi tetap relatif mendekati solusi optimal.

Kata kunci: *Brute force;* Algoritma *Greedy;* Optimasi Portofolio; *Sharpe Ratio; Trade-Off.*

1. LATAR BELAKANG

Seiring berkembangnya teknologi informasi, peningkatan jumlah data transaksi mengalami lonjakan yang sangat besar pada berbagai bidang, salah satunya adalah sektor keuangan dan investasi. Investasi saham menjadi salah satu instrumen keuangan yang banyak diminati karena memiliki potensi dalam memberikan tingkat keuntungan yang relatif tinggi dibandingkan bentuk investasi lainnya. Namun, dalam praktiknya investor sering menghadapi tantangan dalam melakukan pemilihan ataupun menentukan kombinasi saham untuk portofolio

Naskah Masuk: 02 Maret 2026; Revisi: 04 April 2026; Diterima: 10 Mei 2026; Terbit: 29 Juni 2026

yang mampu memberikan keuntungan yang baik dengan tingkat risiko yang stabil. Pemilihan portofolio tidak hanya berorientasi pada return yang tinggi, tetapi juga harus mempertimbangkan *Trade-off* antara return dan risiko (*risk-return trade-off*), karena peningkatan keuntungan hampir selalu diikuti oleh peningkatan tingkat volatilitas investasi. Permasalahan pemilihan kombinasi saham tersebut juga menjadi semakin kompleks karena banyaknya jumlah emiten yang harus dianalisis. Oleh karena itu, diperlukan pendekatan berupa optimasi portofolio yang berbasis algoritma untuk membantu proses pencarian kombinasi portofolio yang efisien sekaligus tetap mempertimbangkan kualitas solusi investasi.

Optimasi portofolio merupakan salah satu solusi yang tepat. Menurut Setiawan dan Dewi (2021), pembentukan portofolio optimal bertujuan memperoleh kombinasi aset yang memberikan tingkat pengembalian terbaik dengan risiko yang lebih efisien sehingga dapat membantu investor dalam pengambilan keputusan investasi. Akan tetapi, proses pencarian kombinasi dalam menentukan portofolio yang optimal, menjadi semakin kompleks ketika jumlah emiten yang digunakan sangat besar karena kemungkinan kombinasi yang harus dievaluasi juga bertambah secara signifikan (Seru & Saputro, 2024). Kondisi ini menyebabkan meningkatnya kebutuhan waktu komputasi dan sumber daya pemrosesan, terutama ketika pencarian solusi dilakukan secara menyeluruh (*exhaustive search*) terhadap seluruh ruang solusi yang tersedia.

Beberapa penelitian sebelumnya yang telah membahas optimasi portofolio saham menggunakan berbagai pendekatan. Azim et al. (2021) menerapkan algoritma genetika berbasis pembobot (*weight*) untuk memperoleh kombinasi portofolio saham yang optimal. Penelitian tersebut menunjukkan bahwa pendekatan heuristik mampu menghasilkan kombinasi investasi yang baik melalui proses pencarian adaptif, namun lebih berfokus pada kualitas solusi portofolio tanpa membahas efisiensi komputasi algoritma yang digunakan. Selain itu, Fadhila dan Zuliana (2023) menggunakan model Markowitz berbasis prediksi harga saham untuk menentukan portofolio optimal berdasarkan keseimbangan antara *return* dan risiko. Penelitian tersebut menegaskan pentingnya evaluasi hubungan *risk-return*, tetapi belum membandingkan performa algoritma optimasi dari sisi kualitas solusi maupun *runtime* komputasi.

Pada sisi lain, penelitian terkait perbandingan algoritma heuristik juga telah dilakukan pada domain pencarian solusi. Syuhada dan Sudirman (2018) membandingkan *Greedy search* dan *Depth-First Search* pada permasalahan pencarian graf untuk menganalisis efisiensi pencarian solusi. Hasil penelitian menunjukkan bahwa pendekatan *Greedy* memiliki *runtime* lebih cepat dibandingkan metode pencarian lain, meskipun tidak selalu menghasilkan solusi global optimum. Dima et al. (2025) juga menjelaskan bahwa algoritma *Greedy* memiliki

keunggulan pada kecepatan proses pencarian karena hanya memilih kandidat solusi terbaik pada setiap langkah, tetapi pendekatan tersebut tetap memiliki keterbatasan dalam menjamin optimalitas solusi.

Berdasarkan penelitian terdahulu, dapat diketahui bahwa penelitian terkait optimasi portofolio saham umumnya berfokus pada kualitas kombinasi portofolio yang dihasilkan, sedangkan penelitian terkait algoritma *Greedy* lebih banyak diterapkan pada domain pencarian jalur atau optimasi umum. Hingga saat ini masih terbatas penelitian yang secara spesifik menganalisis *Trade-off* antara efisiensi komputasi dan optimalitas solusi pada optimasi kombinasi portofolio saham dengan menggunakan algoritma *Greedy* dibandingkan dengan solusi optimal global sebagai *baseline* .

Oleh karena itu, penelitian ini menawarkan analisis *Trade-off* antara efisiensi komputasi dan optimalitas solusi pada optimasi kombinasi portofolio saham terhadap algoritma *Brute force* dan algoritma *Greedy*. Pada penelitian ini, algoritma *Brute force* digunakan sebagai *baseline* atau acuan optimalitas karena mengevaluasi seluruh kemungkinan kombinasi portofolio untuk memperoleh solusi global optimum, sedangkan algoritma *Greedy* digunakan sebagai pendekatan heuristik yang diukur tingkat kedekatan hasilnya terhadap *baseline* tersebut.

Penelitian menggunakan data historis saham sebanyak 472 emiten S&P 500 yang didapat dari kaggle dengan parameter *annualized return* dan *annualized volatility* untuk mengevaluasi performa portofolio menggunakan *Sharpe ratio*. Selain kualitas solusi, penelitian ini juga mengevaluasi *runtime* komputasi, dan *sharpe difference* untuk mengukur hubungan antara optimalitas solusi dan efisiensi waktu komputasi. Hasil penelitian diharapkan dapat memberikan pemahaman mengenai dampak kompleksitas komputasi terhadap kualitas solusi optimasi portofolio saham serta menjadi referensi pengembangan sistem pendukung keputusan investasi berbasis komputasi.

2. KAJIAN TEORITIS

Bagian ini menguraikan teori-teori relevan yang mendasari topik penelitian atau perancangan yang dilakukan.

Optimasi Portofolio Saham

Portofolio saham adalah kombinasi beberapa aset investasi yang disusun untuk menghasilkan keuntungan optimal dengan risiko tertentu. Sebagaimana mestinya investor tidak hanya mengejar return tinggi, tapi juga perlu mempertimbangkan risiko yang muncul saat harga pasar yang penuh volatilitas.

Seperti pada penelitian Fadhila dan Zuliana (2023) menunjukkan bahwa optimasi portofolio bertujuan menemukan kombinasi saham terbaik dengan menyeimbangkan return dan risiko untuk membantu investor mengambil keputusan. Penelitian lain oleh Abdjul et al. (2023) menunjukkan bahwa pembentukan portofolio optimal juga dapat dilakukan menggunakan Model Indeks Tunggal dengan mempertimbangkan hubungan antara return dan risiko investasi. Hasil penelitian tersebut berhasil mengidentifikasi kombinasi saham optimal serta mengukur tingkat risiko portofolio menggunakan pendekatan *Value at Risk* (VaR). Adapula, Nisardi et al. (2024) melakukan penelitian dengan model Markowitz terhadap optimasi portofolio. Model ini mempertimbangkan hubungan antara *expected return* dan risiko portofolio melalui diversifikasi. Diversifikasi berarti menggabungkan saham dengan karakteristik berbeda untuk mengurangi risiko keseluruhan. Pendekatan lainnya adalah pada penelitian Desvi et al. (2024) yang berbasis *historical return* dengan melihat data historis harga saham untuk menentukan kombinasi yang lebih efisien.

Penelitian ini sendiri berdasar pada data historis harga saham dan menghitung *annualized return* serta *annualized volatility* untuk setiap emiten. *Return* menunjukkan keuntungan investasi per tahun, dan *volatility* menunjukkan seberapa besar risiko. Untuk mengevaluasi kombinasi saham, penelitian ini menggunakan *Sharpe ratio*, yang mengukur seberapa baik portofolio dengan cara menyeimbangkan *return* yang didapatkan terhadap risiko.

Algoritma *Brute force*

Brute force merupakan algoritma dengan metode pencarian *exhaustive search* yang cara kerjanya adalah mengevaluasi seluruh kemungkinan solusi yang ada untuk memperoleh hasil optimal. Pendekatan ini menjamin solusi terbaik karena setiap kombinasi diperiksa secara menyeluruh, namun memiliki kelemahan berupa kompleksitas komputasi yang tinggi.

Menurut Froese et al. (2022), peningkatan dimensi dan ruang pencarian solusi dapat menyebabkan kenaikan kompleksitas komputasi secara signifikan sehingga strategi *Brute force* memerlukan waktu komputasi yang semakin lama. Jumlah kombinasi yang dievaluasi pada penelitian ini dapat dihitung menggunakan persamaan kombinasi berikut:

$$C(n,k) = \frac{n!}{k!\,(n-k)!} \qquad \ldots (i)$$

Pada Persamaan 1, $C(n,k)$ menunjukkan total kombinasi yang dihasilkan, n merepresentasikan jumlah keseluruhan emiten saham yang tersedia, serta k menyatakan jumlah saham yang dipilih untuk menyusun portofolio. Setiap kombinasi kemudian dihitung nilai *annualized return*, volatilitas, dan *Sharpe ratio*-nya untuk memperoleh kualitas portofolio. Solusi optimal dicari menggunakan persamaan:

$$P^* = arg\ max\ _{P_i} \in C(n,k) Score(P_i) \quad \ldots (ii)$$

Pada Persamaan 2, P^* mewakili portofolio optimal global yang terpilih, P_i menunjukkan kombinasi portofolio ke-i yang merupakan bagian dari seluruh ruang kombinasi $C(n,k)$, dan $Score(P_i)$ menyatakan nilai evaluasi performa dari portofolio tersebut berdasarkan Sharpe ratio. Penggunaan *Brute force* sebagai *baseline* bertujuan untuk mengetahui sejauh mana algoritma *Greedy* mampu menghasilkan solusi yang mendekati optimal dengan waktu komputasi yang lebih efisien. Dengan demikian, hasil optimasi dari algoritma *Greedy* dapat dibandingkan langsung terhadap solusi optimal yang dihasilkan oleh algoritma *Brute force*.

Algoritma *Greedy*

Greedy merupakan algoritma heuristik yang bekerja dengan memilih solusi terbaik pada setiap langkah proses pencarian tanpa mengevaluasi seluruh kemungkinan solusi. Pendekatan ini bertujuan untuk memperoleh solusi secara cepat dengan kompleksitas komputasi yang lebih rendah dibandingkan metode *Brute force* (*exhaustive search)* (Dima et al., 2025).

Menurut Xu dan Xu (2024), algoritma *Greedy* memiliki efisiensi komputasi yang tinggi karena hanya memproses kandidat solusi yang dianggap paling optimal pada setiap iterasi. Akan tetapi, metode ini tidak selalu menjamin solusi global optimum karena keputusan lokal yang diambil pada tahap awal dapat memengaruhi hasil akhir optimasi. Oleh sebab itu, algoritma *Greedy* sering digunakan pada permasalahan optimasi yang membutuhkan *runtime* cepat dan efisien (Syuhada & Sudirman, 2018).

Pada penelitian ini, algoritma *Greedy* digunakan untuk memilih saham berdasarkan nilai *sharpe ratio* tertinggi secara bertahap hingga memenuhi jumlah kombinasi portofolio yang ditentukan pengguna. Pemilihan solusi pada setiap iterasi dapat direpresentasikan menggunakan persamaan berikut:

$$x_i = \arg\max_{S_i \in H} Score(S_i) \quad \ldots (iii)$$

Pada Persamaan 3, di mana x_i merupakan emiten saham yang dipilih pada iterasi ke-i , S_i mewakili saham kandidat yang dievaluasi dari himpunan saham yang tersedia H, dan $Score(S_i)$ menunjukkan nilai evaluasi performa dari saham kandidat tersebut berdasarkan *Sharpe ratio* individu. Portofolio kemudian dibentuk secara bertahap menggunakan himpunan:

$$P = \{x_1, x_2, \ldots, x_k\} \quad \ldots (iv)$$

Pada Persamaan 4, P mewakili portofolio hasil optimasi akhir dan $x_1, x_2, \dots, x_k$ menunjukkan sekumpulan emiten saham yang terpilih berdasarkan nilai *score* terbaik pada setiap tahapan iterasi.

Pendekatan tersebut memungkinkan proses optimasi dilakukan dengan waktu komputasi yang jauh lebih cepat dibandingkan *Brute force*. Penelitian ini membandingkan performa algoritma *Greedy* terhadap *baseline Brute force* untuk menganalisis *Trade-off* antara efisiensi komputasi dan optimalitas solusi pada optimasi kombinasi portofolio saham.

Annualized Return* dan *Volatility

Return merupakan tingkat keuntungan yang diperoleh investor dari suatu investasi saham selama periode tertentu. Dalam penelitian ini, return dihitung menggunakan *daily return* berdasarkan perubahan harga penutupan saham (*Adjusted Close*) setiap hari perdagangan. Nilai *daily return* kemudian dirata-ratakan dan dikonversi menjadi *annualized return* dengan mengalikan rata-rata *return* harian dengan 252 hari perdagangan saham dalam satu tahun. Secara matematis, *annualized return* dihitung menggunakan persamaan:

$$Anualized\ Return\ = \bar{R}_{daily}\ \times 252 \quad \dots (v)$$

Secara matematis, pada Persamaan 5, *Annualized Return* menunjukkan keuntungan investasi per tahun, $\bar{R}_{\text{daily}}$ merupakan rata-rata *daily return* yang diperoleh dari perubahan harga penutupan saham harian, dan angka 252 merepresentasikan asumsi jumlah rata-rata hari perdagangan saham aktif dalam kurun waktu satu tahun.

Sementara itu, volatilitas digunakan untuk mengukur tingkat risiko investasi. Untuk menghitung risiko keseluruhan dari sebuah portofolio saham gabungan (*Portfolio Volatility*), tidak cukup hanya dengan merata-ratakan risiko masing-masing saham. Diperlukan perhitungan yang mempertimbangkan korelasi antar-aset melalui Matriks Kovarians (*Covariance Matrix*).

Dalam optimasi ini, diasumsikan setiap saham di dalam portofolio memiliki bobot yang sama (*equal weight*) sebesar $w_i\ = \ 1/k$. Risiko portofolio $\{\sigma_p\}$ dihitung dengan mengalikan vektor bobot (w) dengan matriks kovarians tahunan (σ) , yang diformulasikan sebagai berikut:

$$\sigma_p\ = \sqrt{\{w^t \Sigma\ w\}} \qquad \dots (vi)$$

$$\sigma_p\ = \sqrt{\sum_{i=1}^{k} \sum_{j=i}^{k} w_i w_j \sigma_{ij}} \qquad \dots (vii)$$

Diformulasikan melalui Persamaan 6 dan Persamaan 7. Berdasarkan kedua persamaan tersebut, σ_p bertindak sebagai nilai risiko portofolio gabungan, w mewakili vektor bobot alokasi masing-masing saham di dalam portofolio, Σ menunjukkan matriks kovarians atau

covariance matrix tahunan antar-saham, dan σ_{ij} menyatakan nilai kovarians spesifik antara saham i dan saham j. Penggunaan matriks kovarians ini sangat penting untuk menangkap efek diversifikasi, Jika dua saham memiliki kovarians negatif atau korelasi yang berlawanan (Ismail & Pham, 2017). Secara singkat, parameter σ_p inilah yang kemudian dijadikan nilai pembagi dalam kalkulasi *sharpe ratio.*

Sharpe ratio

Untuk mengukur seberapa efisien keuntungan yang dihasilkan suatu investasi dibandingkan dengan risiko yang ditanggung, investor umumnya mengacu pada metode *sharpe ratio*. Secara prinsip, portofolio dengan skor *sharpe ratio* yang tinggi dianggap memiliki performa yang lebih baik karena mampu menghasilkan *excess return* yang lebih besar terhadap tingkat risiko investasi. Seperti pada penelitian Landete et al. (2020) menekankan bahwa *sharpe ratio* digunakan untuk membandingkan *excess return* portofolio terhadap volatilitas yang dihasilkan sehingga dapat menunjukkan kualitas *risk-adjusted return* suatu portofolio.

$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma_{\ p}} \qquad \dots (viii)$$

Pada Persamaan 8. Berdasarkan formulasi tersebut, R_p menunjukkan tingkat *return* portofolio tahunan, R_f sebagai *risk-free rate* yang diasumsikan konstan sebesar 2% per tahun, dan σ_p mewakili nilai standar deviasi atau volatilitas portofolio gabungan.

Dalam penelitian ini, proses evaluasi melibatkan beberapa parameter. Pertama, *risk-free rate* yang diasumsikan sebesar 2% per tahun, nilai tersebut dipilih sebagai asumsi konservatif yang digunakan secara konstan agar proses evaluasi performa portofolio pada kedua algoritma dapat dilakukan secara konsisten dan adil. Kedua, penentuan kualitas hasil kombinasi saham yang dihasilkan oleh algoritma *Brute force* maupun *Greedy* sepenuhnya bersandar pada nilai *sharpe ratio*.

Kompleksitas Komputasi

Algoritma dengan kompleksitas yang tinggi dan cenderung membutuhkan waktu eksekusi lebih lama ketika ukuran data meningkat. Menurut Ast et al. (2021), analisis performa algoritma sangat penting dalam menentukan efisiensi metode komputasi pada proses optimasi. Selain itu, Skala (2013) menjelaskan bahwa algoritma *exhaustive search* memiliki kompleksitas eksponensial $O(2^n)$ atau lebih tinggi tergantung jumlah kombinasi yang dievaluasi.

Pada penelitian ini, kompleksitas komputasi dianalisis melalui perbandingan *runtime* antara algoritma *Brute force* dan algoritma *Greedy*. *Runtime* digunakan untuk mengukur efisiensi waktu eksekusi algoritma dalam menghasilkan solusi optimasi kombinasi portofolio saham.

3. METODE PENELITIAN

Pada penelitian ini digunakan dataset sekunder berupa data historis harga saham dari 472 emiten yang tergabung dalam indeks S&P 500. Dataset ini diperoleh dari repositori publik Kaggle (*S&P 500 Historical Data*) dengan menggunakan parameter harga penutupan (*Adjusted Close Price*). Meskipun populasi data induk mencakup periode dari tahun 2000 hingga 2026, skenario pengujian pada penelitian ini pada periode tahun 2025 hingga 2026. Pemilihan periode ini didasari dan dilakukan untuk mengevaluasi performa algoritma pada kondisi pasar terkini, di mana fokus utama pengujian sendiri adalah untuk ditekankan pada evaluasi *Trade-off* akibat pertumbuhan kompleksitas jumlah kombinasi portofolio. Bentuk dari data mentah (*raw dataset*) harian yang digunakan sebelum tahap komputasi dapat dilihat pada Tabel 1.

Tabel 1. Sampel Dataset Awal.

Date	***Ticker***	***Adj Close***	***Daily Return***
2025-01-02	AAPL	242.53	NaN
2025-01-03	AAPL	242.04	-0.002020
2025-01-06	AAPL	243.67	0.006734
2025-01-02	AMZN	220.22	NaN
2025-01-03	AMZN	224.19	0.018027
2025-01-06	AMZN	227.61	0.015255
2025-01-02	GOOG	189.89	NaN
2025-01-03	GOOG	192.38	0.013113
2025-01-06	GOOG	197.19	0.025003
2025-01-02	MSFT	415.51	NaN
2025-01-03	MSFT	420.25	0.011408
2025-01-06	MSFT	424.72	0.010637
2025-01-02	TSLA	379.28	NaN
2025-01-03	TSLA	410.44	0.082156
2025-01-06	TSLA	411.05	0.001486

Tahapan penelitian diawali dengan proses pengumpulan data dan *preprocessing* data saham historis. *Preprocessing* dilakukan dengan menyeragamkan format tanggal, mengurutkan data berdasarkan *ticker* saham dan tanggal, serta menghitung nilai *daily return* menggunakan perubahan persentase harga penutupan harian. Mengingat tidak semua emiten melakukan *Initial Public Offering* (IPO) pada tahun yang sama, sistem melakukan pembersihan data kosong (*missing value handling*) menggunakan metode *listwise deletion* (penghapusan baris

NaN). Penanganan ini bersifat esensial agar perhitungan matriks kovarians tidak menghasilkan nilai tak terdefinisi (*undefined*) yang dapat menggagalkan keseluruhan proses komputasi risiko portofolio. Data yang telah dibersihkan kemudian diagregasi untuk mendapatkan *annualized return*, *annualized volatility*, dan *covariance matrix* sebagai parameter utama evaluasi. Cuplikan data hasil *preprocessing* yang telah dikonversi ke skala tahunan dan siap diproses oleh algoritma optimasi ditunjukkan pada Tabel 2.

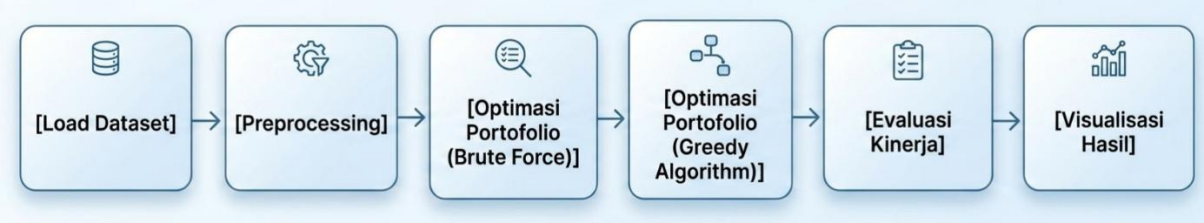
Tabel 2. Sampel *Dataset* Setelah *Preprocessing*.

Ticker	***Average Return***	***Volatility***	***Sharpe Ratio***
AAPL	18.40 %	27.86 %	0.6605
AMZN	11.55 %	35.05 %	0.3295
GOOG	30.19 %	30.79 %	0.9806
MSFT	15.99 %	26.14 %	0.6118
TSLA	28.30 %	60.21 %	0.4700

Penggunaan *covariance matrix* pada optimasi portofolio bertujuan untuk mengukur hubungan antar aset sehingga risiko portofolio dapat dihitung secara lebih realistis melalui efek diversifikasi (Ismail & Pham, 2017). Selanjutnya dilakukan perhitungan *Sharpe ratio* menggunakan selisih *annualized return* terhadap *risk-free rate* sebesar 2% yang dibagi dengan volatilitas portofolio.

Proses optimasi dilakukan menggunakan dua pendekatan algoritma, yaitu algoritma *Brute force* dan algoritma *Greedy*. *Brute force* digunakan sebagai *baseline* dengan mengevaluasi seluruh kemungkinan kombinasi portofolio berdasarkan jumlah saham yang dipilih pengguna. Sementara itu, algoritma *Greedy* menggunakan pendekatan heuristik dengan memilih saham berdasarkan nilai *Sharpe ratio* tertinggi secara bertahap tanpa mengevaluasi seluruh kombinasi solusi. Setelah proses optimasi selesai dilakukan, penelitian membandingkan performa kedua algoritma berdasarkan beberapa parameter evaluasi, yaitu *runtime*, *Sharpe ratio*, *annualized return*, *annualized volatility*, dan *sharpe difference*. Perbandingan tersebut digunakan untuk menganalisis *Trade-off* antara efisiensi komputasi dan optimalitas solusi pada optimasi kombinasi portofolio saham.

Untuk mendukung proses analisis, penelitian ini juga mengembangkan aplikasi GUI berbasis Python menggunakan Matplotlib. GUI digunakan untuk mempermudah simulasi optimasi portofolio dengan parameter yang dapat diatur secara dinamis oleh pengguna, seperti jumlah kombinasi portofolio, serta rentang tahun data saham. Hasil optimasi kemudian divisualisasikan dalam bentuk tabel dan grafik perbandingan performa algoritma. Adapun alur metodologi penelitian secara umum dapat dilihat pada Gambar 1.



Gambar 1. Alur Metodologi.

Ilustrasi Matematis Pencarian Solusi

Untuk memvalidasi cara kerja kedua algoritma, berikut adalah ilustrasi numerik berskala kecil. Diasumsikan terdapat $n = 3$ emiten (Saham A, B, C), dan sistem akan mencari kombinasi optimal berisi $k = 2$ saham dengan bobot seragam (*equal weight*) atau *w = 0.5. Risk-free rate* ditetapkan sebesar 0.02 (2%). Diketahui data performa individual (*Annualized Return dan Volatility*) masing-masing saham sebagai berikut: (1) Saham A: Return = 0.12, Volatilitas = 0.15 → Sharpe Individual = 0.667; (2) Saham B: Return = 0.10, Volatilitas = 0.10 → Sharpe Individual = 0.800; (3) Saham C: Return = 0.14, Volatilitas = 0.20 → Sharpe Individual = 0.600.

Matriks kovarians $\sum$ dari ketiga saham menunjukkan bahwa Saham A dan B memiliki korelasi positif yang tinggi, sedangkan Saham A dan C memiliki korelasi negatif (mendukung diversifikasi): (1) Kovarians(A, B) = 0.0120; (2) Kovarians(A, C) = -0.0150; (3) Kovarians(B, C) = 0.0000.

Evaluasi Menggunakan Algoritma Brute force

Algoritma ini melakukan pencarian menyeluruh (*exhaustive*) terhadap kombinasi $C(3,2) = 3$ kemungkinan portofolio dan memperhitungkan matriks kovarians pada setiap evaluasi. (1) Kombinasi (A, B): *Sharpe ratio* = 0.757; (2) Kombinasi (B, C): * Return Gabungan: 0.12; (3) Varians: $(0.5^2 \times 0.10^2) + (0.5^2 \times 0.20^2) + 0 = 0.0125$; (4) Risiko: $\sqrt{0.0125} = 0.1118$; (5) *Sharpe ratio*: $(0.12 - 0.02) \div 0.1118 = 0.894$; (6) Kombinasi (A, C): (7) Return Gabungan: 0.13; (8) Varians:$(0.5^2 \times 0.15^2) + (0.5^2 \times 0.20^2) + (2 \times 0.5 \times 0.5 \times -0.0150 = 0.0081$; (9) Risiko: $\sqrt{0.0081} = 0.0901$; (10) *Sharpe ratio*: $(0.13 - 0.02) \div 0.0901 = 1.220$).

Evaluasi Menggunakan Algoritma Greedy

Berdasarkan pendekatan ini, sistem langsung mengurutkan nilai *sharpe ratio* individual setiap saham secara menurun (*descending*). (1) Urutan prioritas: Saham B (0.800) → Saham A (0.667) → Saham C (0.600). (2) Karena sistem membutuhkan $k = 2$, algoritma memotong batas atas (top-k) dan memilih portofolio (A, B). (3) Evaluasi akhir portofolio (A, B). (4) Return Gabungan: $(0.12 + 0.10)/2 = 0.11$. (5) Varians: $\sigma_p^2 = (0.5^2 \times 0.15^2) + (0.5^2 \times$

$0.10^2) + (2 \times 0.5^2 \times 0.0120) = 0.0141$. (6)Risiko (Volatilitas): $\sqrt{0.0141} = 0.1188$. (7) *Sharpe ratio* Akhir (*Greedy*): $(0.11 - 0.02) \div 0.1188 = 0.757$.

Kesimpulan Evaluasi

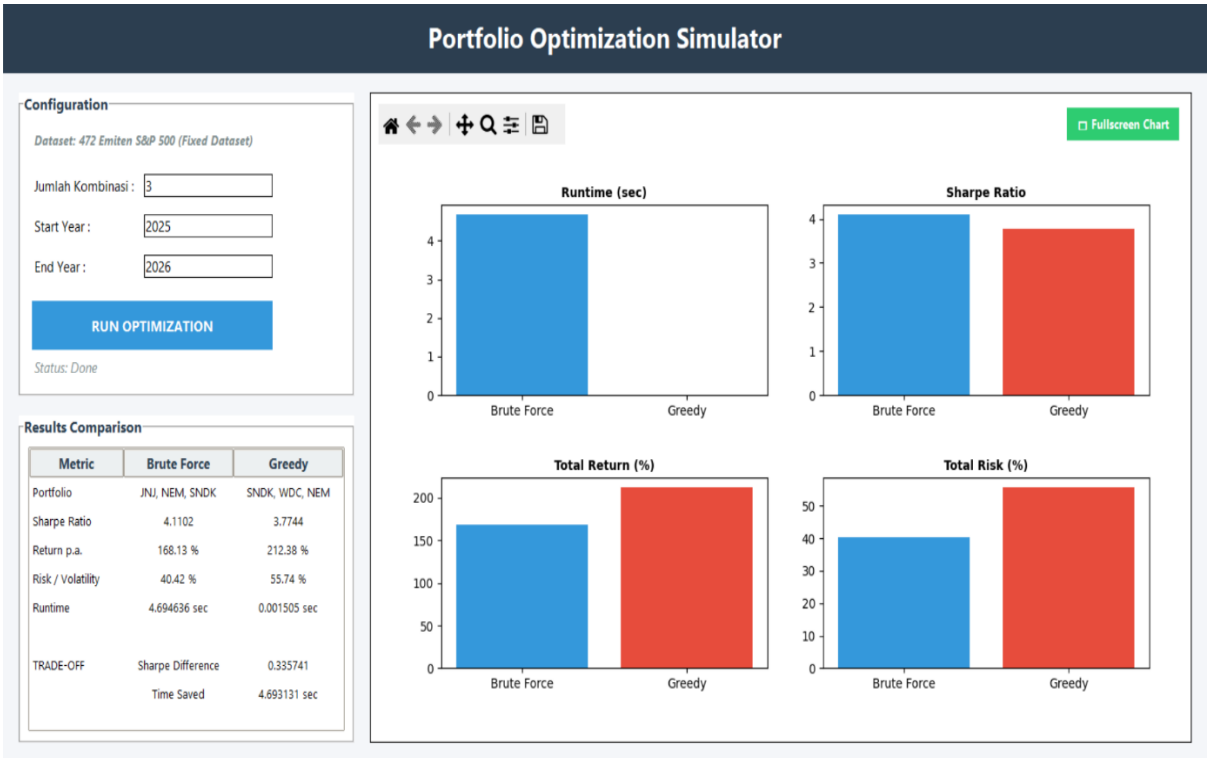
Brute force menemukan solusi global optimal pada portofolio (A, C) dengan *Sharpe ratio* 1.220. Sebaliknya, algoritma *Greedy* terjebak pada *local optimum* (A, B) dengan *Sharpe ratio* 0.757. Hal ini terjadi karena algoritma heuristik hanya mempertimbangkan performa individual (memilih Saham B karena risikonya kecil secara tunggal), dan gagal menangkap efek reduksi risiko (korelasi negatif) yang terjadi antara Saham A dan Saham C.

4. HASIL DAN PEMBAHASAN

Bagian ini memaparkan hasil pengujian dan analisis komprehensif dari simulasi optimasi kombinasi portofolio saham yang telah dilakukan. Pembahasan pada bab ini dibagi menjadi dua bagian utama. Bagian pertama menguraikan implementasi sistem antarmuka pengguna (GUI) beserta langkah eksekusi strategi komputasinya. Bagian kedua menyajikan data hasil pengujian komparatif antara algoritma *Brute force* dan *Greedy*.

Implementasi Sistem dan Langkah Eksekusi Strategi Algoritma

Untuk menguji dan membandingkan performa algoritma *Brute force* dan *Greedy*, penelitian ini mengimplementasikan sebuah sistem berbasis *Graphical User Interface* (GUI) menggunakan bahasa pemrograman *Python*. Secara operasional, eksekusi sistem diawali dengan konfigurasi parameter input, di mana pengguna menentukan jumlah kombinasi portofolio dan rentang waktu pengujian terhadap *fixed dataset* 472 emiten S&P 500. Saat proses optimasi dijalankan, sistem secara *backend* melakukan *preprocessing* dan pembentukan *covariance matrix*, lalu mengeksekusi algoritma *Brute force* dan *Greedy* secara paralel untuk mencari solusi portofolio dengan *sharpe ratio* tertinggi. Hasil dari kedua algoritma tersebut kemudian ditampilkan pada panel komparasi yang memuat detail emiten terpilih, metrik evaluasi komprehensif (*sharpe ratio*, *return*, volatilitas, dan *runtime*), beserta kalkulasi *trade-off*. Terakhir, seluruh hasil komputasi tersebut divisualisasikan ke dalam bentuk bar *chart* menggunakan *Matplotlib* pada sisi kanan antarmuka untuk mempermudah analisis komparatif. tampilan antarmuka tersebut dapat dilihat pada Gambar 2.



Gambar 2. Tampilan Implementasi GUI Optimasi Portofolio.

Pada Gambar 4, panel sisi kiri antarmuka menampilkan parameter input beserta hasil komputasi numerik secara mendetail. Mengambil contoh skenario pengujian untuk periode 2025–2026 dengan pencarian 3 kombinasi saham $k = 3$, sistem memperlihatkan bahwa algoritma *Brute force* merekomendasikan portofolio emiten JNJ, NEM, dan SNDK dengan nilai Sharpe ratio mencapai 4.1102 dalam waktu 4,694636 detik. Sebaliknya, *Greedy* Algorithm merekomendasikan emiten SNDK, WDC, dan NEM dengan Sharpe ratio 3.7744 dalam waktu eksekusi yang sangat efisien, yaitu 0,001505 detik. Panel ini juga secara eksplisit mengukur perhitungan trade-off, di mana algoritma heuristik mampu menghemat waktu pencarian (*time saved*) hingga 4,693131 detik dengan pengorbanan selisih skor (*sharpe difference*) sebesar 0.335741.

Selanjutnya, pada sisi kanan antarmuka, metrik evaluasi dari kedua algoritma divisualisasikan dalam bentuk grafik *bar chart* komparatif yang mencakup perbandingan *runtime*, *sharpe Ratio*, *total return*, dan *total risk*. Melalui implementasi ini, sistem terbukti mampu melakukan simulasi optimasi secara interaktif dan dinamis, sehingga pengguna dapat dengan mudah menyesuaikan parameter rentang tahun maupun batas kombinasi saham untuk mengeksplorasi berbagai skenario pengujian performa algoritma.

Hasil Pengujian Algoritma

Hasil pengujian tersebut dapat dilihat pada Tabel 1 yang menunjukkan hasil perbandingan performa algoritma *Brute force* dan algoritma *Greedy* pada beberapa skenario pengujian.

Tabel 3. Perbandingan Hasil Optimasi Portofolio Menggunakan Algoritma *Brute force* dan Algoritma *Greedy.*

No	Kombinasi	Periode	Algoritma	*Sharpe ratio*	*Return p.a*	*Risk*	*Runtime*
1.	2	2025-2026	*Brute force*	3.8006	226.98%	59.20%	0.042529 detik
			Greedy	3.4404	260.31%	75.08%	0.001478 detik
2.	3	2025-2026	*Brute force*	4.1102	168.13%	40.42%	4.694636 detik
			Greedy	3.7744	212.38%	55.74%	0.001505 detik
3.	4	2025-2026	*Brute force*	4.3619	137.07%	30.97%	670.710229 detik
			Greedy	3.6470	198.69%	53.93%	0.001524 detik

Tabel 3 menunjukkan hasil perbandingan performa Algoritma *Brute force* dan algoritma *Greedy* pada beberapa skenario pengujian. Berdasarkan hasil tersebut terlihat bahwa algoritma *Brute force* secara konsisten menghasilkan *sharpe ratio* yang lebih tinggi dibandingkan algoritma *Greedy*. Hal tersebut menunjukkan bahwa pendekatan *exhaustive search* mampu menemukan kombinasi portofolio yang lebih optimal karena seluruh kemungkinan kombinasi saham dievaluasi secara menyeluruh.

Peningkatan jumlah kombinasi portofolio menyebabkan *runtime* algoritma *Brute force* meningkat secara sangat signifikan. Pada pengujian dengan kombinasi $k = 2$, *runtime Brute force* masih berada pada 0.042529 detik. Akan tetapi, ketika jumlah kombinasi ditingkatkan menjadi $k = 3$, *runtime* meningkat menjadi 4.694636 detik. Peningkatan paling signifikan terjadi pada pengujian $k = 4$ dengan *runtime* mencapai 670.710229 detik atau lebih dari 11 menit. Kondisi tersebut menunjukkan adanya pertumbuhan kompleksitas kombinatorial yang sangat besar seiring meningkatnya jumlah kombinasi portofolio yang harus dievaluasi.

Secara matematis, jumlah kombinasi portofolio yang dievaluasi dapat dihitung menggunakan persamaan kombinasi C(n,k). Dengan jumlah emiten tetap sebanyak 472 saham, maka jumlah kombinasi yang harus diproses meningkat secara drastis seiring bertambahnya nilai k.

Tabel 4. Pertumbuhan Jumlah Kombinasi Portofolio.

No	*Nilai k*	*Jumlah Kombinasi*
1.	2	111,156
2.	3	17,389,540
3.	4	2,039,656,190

Berdasarkan Tabel 4, terlihat bahwa peningkatan jumlah kombinasi portofolio menyebabkan pertumbuhan ruang pencarian secara eksponensial. Hal tersebut berdampak langsung terhadap *runtime* algoritma *Brute force* karena seluruh kombinasi harus dievaluasi satu per satu untuk memperoleh solusi optimal global.

Di sisi lain, algoritma *Greedy* menunjukkan *runtime* yang sangat stabil pada seluruh pengujian, yaitu sekitar 0.001 detik. Hal tersebut terjadi karena algoritma hanya melakukan proses *sorting* berdasarkan nilai *Sharpe ratio* tertinggi tanpa mengevaluasi seluruh kombinasi kemungkinan solusi. Dengan demikian, algoritma *Greedy* memiliki efisiensi komputasi yang jauh lebih tinggi dibandingkan *Brute force*, terutama ketika ukuran ruang pencarian meningkat secara besar.

Walaupun demikian, hasil pengujian juga menunjukkan adanya *sharpe difference* antara *Greedy* dan *Brute force* yang semakin meningkat pada nilai kombinasi yang lebih besar. Pada pengujian $k = 4$, perbedaan *sharpe ratio* mencapai 0.714893. Hal tersebut menunjukkan bahwa pendekatan heuristik *Greedy* tidak selalu mampu menemukan solusi global optimum karena keputusan lokal terbaik yang dipilih pada tahap awal dapat memengaruhi kualitas hasil akhir portofolio.

Secara keseluruhan, hasil penelitian menunjukkan adanya *Trade-off* yang jelas antara optimalitas solusi dan efisiensi komputasi. Algoritma *Brute force* sukses sebagai *baseline* karena dapat menghasilkan solusi optimal global namun memang memiliki *runtime* yang sangat tinggi, sedangkan algoritma *Greedy* unggul dalam efisiensi *runtime* dengan kualitas solusi yang relatif mendekati *baseline* .

5. KESIMPULAN DAN SARAN

Berdasarkan hasil penelitian yang telah dilakukan, dapat disimpulkan bahwa algoritma *Brute force* berhasil digunakan sebagai *baseline* optimal pada proses optimasi kombinasi portofolio saham. Pendekatan exhaustive search yang digunakan mampu mengevaluasi seluruh kemungkinan kombinasi portofolio sehingga menghasilkan solusi global optimum berdasarkan nilai *Sharpe ratio* terbaik. Akan tetapi, peningkatan jumlah kombinasi saham menyebabkan *runtime* algoritma meningkat secara sangat signifikan akibat pertumbuhan kompleksitas kombinatorial yang tinggi.

Pada pengujian menggunakan 472 emiten saham S&P 500 periode 2025-2026, *runtime* algoritma *Brute force* meningkat dari 0.042529 detik pada kombinasi k = 2 menjadi 670.710229 detik pada kombinasi k = 4. Hasil tersebut menunjukkan bahwa pendekatan exhaustive search memiliki keterbatasan efisiensi ketika ruang pencarian solusi semakin besar. Sementara itu, algoritma *Greedy* berhasil menghasilkan *runtime* yang sangat stabil dan jauh lebih cepat dibandingkan *Brute force* pada seluruh skenario pengujian. Pendekatan heuristik yang digunakan memungkinkan proses optimasi dilakukan tanpa mengevaluasi seluruh kemungkinan kombinasi portofolio. Walaupun nilai *Sharpe ratio* yang dihasilkan masih berada di bawah *baseline Brute force*, kualitas solusi yang diperoleh tetap relatif mendekati solusi optimal dengan efisiensi komputasi yang jauh lebih baik.

Secara keseluruhan, penelitian ini berhasil menunjukkan adanya *Trade-off* antara optimalitas solusi dan efisiensi komputasi pada optimasi kombinasi portofolio saham. *Brute force* sukses karena unggul dalam menghasilkan solusi optimal global namun memang memiliki *runtime* yang sangat tinggi, sedangkan algoritma *Greedy* unggul dalam efisiensi *runtime* dengan kualitas solusi yang relatif mendekati *baseline* optimal. Selain itu, penelitian ini juga berhasil mengimplementasikan aplikasi berbasis *Graphical User Interface* (GUI) menggunakan Python untuk memvisualisasikan proses optimasi portofolio saham secara interaktif berdasarkan parameter kombinasi saham dan rentang periode data historis.

Untuk penelitian selanjutnya, metode optimasi dapat dikembangkan menggunakan pendekatan lain seperti *Genetic Algorithm, Particle Swarm Optimization* (PSO), maupun metode berbasis *Machine Learning* untuk memperoleh kualitas solusi yang lebih optimal dengan efisiensi komputasi yang tetap baik. Selain itu, penelitian berikutnya juga dapat mempertimbangkan penggunaan bobot portofolio dinamis *(non-equal weight*), transaction cost, serta data pasar secara *real-time* agar hasil optimasi lebih realistis terhadap kondisi investasi sebenarnya.

UCAPAN TERIMA KASIH

Penulis menyampaikan ucapan terima kasih yang sebesar-besarnya kepada Ibu Tinaliah, M.Kom., selaku dosen pembimbing yang telah memberikan arahan, bimbingan, waktu, serta ilmu yang sangat berharga selama proses penyusunan dan penyelesaian publikasi ilmiah ini.

DAFTAR REFERENSI

Abdjul, L., Resmawan, R., Nuha, A. R., Nurwan, Wungguli, D., & Nashar, L. O. (2023). Optimasi portofolio saham syariah menggunakan model indeks tunggal dan VaR berbasis GUI Matlab. *Jambura Journal of Mathematics, 5*(1), 243–253. https://doi.org/10.34312/jjom.v5i1.18570

Ast, J., Wasseghi, R., & Nyhuis, P. (2021). A comparison of methods for determining performance based employee deployment in production systems. *Production Engineering, 15*(3–4), 335–342. https://doi.org/10.1007/s11740-021-01019-5

Azim, M. F., Azizah, & Anggraini, D. (2021). Optimasi bobot portofolio menggunakan algoritma genetika. *Jurnal Sains Matematika dan Statistika, 7*(1), 58–64. https://doi.org/10.24014/jsms.v7i1.12190

Desvi, M. S., Saepudin, D., & Kurniawan, I. (2024). Optimasi portofolio saham menggunakan metode *Stock Network Portfolio Allocation* berbasis *return history* (SNPAr). *LOGIC: Jurnal Penelitian Informatika, 2*(1), 21–26. https://doi.org/10.25124/logic.v2i2.8809

Dima, J., Hamzah, M. S., Tallo, C. G., & Fallo, D. Y. (2025). Tinjauan literatur tentang pemanfaatan algoritma greedy untuk pencarian jalur terpendek. *Jurnal Kridatama Sains dan Teknologi, 7*(1), 519–528. https://doi.org/10.53863/kst.v7i01.1683

Fadhila, S. N., & Zuliana, S. U. (2023). Optimasi portofolio saham menggunakan model Markowitz berdasarkan prediksi harga saham. *Kaunia: Integration and Interconnection Islam and Science, 18*(1), 35–40. https://doi.org/10.14421/kaunia.3948

Froese, V., Hertrich, C., & Niedermeier, R. (2022). The computational complexity of ReLU network training parameterized by data dimensionality. *Journal of Artificial Intelligence Research, 74*, 1075–1116. https://doi.org/10.1613/jair.1.13547

Ismail, M., & Pham, H. (2017). *Covariance matrix estimation for portfolio optimization*. *arXiv*. https://doi.org/10.48550/arXiv.1610.06805

Landete, M., Monge, J. F., Ruiz, J. L., & Segura, J. V. (2020). Sharpe portfolio using a cross-efficiency evaluation. In V. Charles, J. Aparicio, & J. Zhu (Eds.), *Data science and productivity analytics* (pp. 415–439). Springer International Publishing. https://doi.org/10.1007/978-3-030-43384-0_15

Nisardi, M. R., Husain, H., Kusnaeni, & Resky, A. (2024). Penentuan portofolio saham optimal menggunakan metode Markowitz sebagai dasar keputusan investasi. *Square: Journal of Mathematics and Mathematics Education, 6*(1), 33–40. https://doi.org/10.21580/square.2024.6.1.20441

Seru, F., & Saputro, A. D. (2024). Pendekatan optimalisasi portofolio dengan *Capital Asset Pricing Model* dan model Markowitz sebagai strategi investasi cerdas bagi investor milenial. *Journal of Mathematics: Theory and Applications, 6*(2), 214–224. https://doi.org/10.31605/jomta.v6i2.4117

Setiawan, C. D., & Dewi, V. I. (2021). Analisis pembentukan portofolio saham optimal menggunakan pendekatan model indeks tunggal sebagai dasar keputusan investasi. *Valid Jurnal Ilmiah, 19*(1), 24–35. https://doi.org/10.55606/jebaku.v5i1.5461

Skala, V. (2013). Fast Oexpected(N) algorithm for finding exact maximum distance in E2 instead of O(N^2) or O(N log N). *AIP Conference Proceedings, 1558*, 2496–2499. https://doi.org/10.1063/1.4826047

Syuhada, M. Y., & Sudirman. (2018). Perbandingan algoritma greedy search dan algoritma *depth-first search* pada pencarian kota dengan *Graph Romania Problem*. *Journal of Informatics and Telecommunication Engineering (JITE), 1*(2), 58–60. https://doi.org/10.31289/jite.v1i2.1405

Xu, J., & Xu, X. (2024). *Efficient and provably convergent randomized greedy algorithms for neural network optimization*. *arXiv*. https://doi.org/10.48550/arXiv.2407.17763