JURNAL DEVICE, VOL. 11 NO 1, 25-33
ISSN: 0216-9185 | eISSN: 2746-8984

ALGORITMA EVOLUSI GENETIKA SEBAGAI FUNGSI OPTIMASI JARINGAN SARAF BUATAN

Muqorrobien Ma'rufi [1], Muhamad Fuat Asnawi [2]

[1) 2)] Universitas Sains Al-Qur'an
Email : obin.mf@gmail.com [1)], fuatasnawi@unsiq.ac.id [2)]

ABSTRAK

Reinforcement Learning (RL) pada Machine Learning memiliki masalah yang dikenal dengan istilah *exploitation and exploration problem*, yang membuat RL sering kesulitan mendapatkan konvergensi dalam proses trainingnya. Untuk itu penulis coba mengambil pendekatan lainnya dengan cara menggunakan Algoritma Evolusi Genetika sebagai fungsi optimasinya. Kesimpulan dari penelitian ini adalah algoritma evolusi genetika sangat efektif diterapkan sebagai fungsi optimasi pada Reinforcement Learning untuk membangun sistem trading saham berbasis kecerdasan buatan.

Kata Kunci : algoritma, evolusi, genetika

ABSTRACT

Reinforcement Learning (RL) in Machine Learning has a problem known as exploitation and exploration problems, which makes it difficult for RL to find convergence in the training process. For that the authors try to take another approach by using the Genetic Evolution Algorithm as an optimization function. The conclusion of this research is that genetic evolution algorithm is very effective to be applied as an optimization function in Reinforcement Learning to build a stock trading system based on artificial intelligence.

Keywords: algorithm, evolution, genetic

1. PENDAHULUAN

Perkembangan sistem berbasis kecerdasan buatan (*Artificial Intelligence*) saat ini sangatlah cepat, apalagi sejak ditemukannya metode *Deep Learning* yang menggunakan konsep jaringan syaraf buatan (*Artificial Neural Network*) yang mana berdasarkan hasil studi memiliki performa jauh lebih baik dari metode *Machine Learning* tradisional terutama ketika berhadapan dengan dataset yang sangat variatif. Hal inilah yang membuat penulis memilih jaringan syaraf buatan sebagai bahan penelitian.

Pada *Machine Learning* salah satu fungsi yang sangat penting adalah fungsi optimasi, fungsi ini memastikan proses belajar mesin berjalan dengan optimal.

Algoritma evolusi genetika penulis pilih sebagai fungsi optimasi karena masih sangat jarang ditemukan publikasi ilmiah yang menguji pendekatan ini terutama pada jaringan syaraf buatan, sementara algoritma genetika sudah terkenal akan kemampuannya yang baik dalam hal optimasi.

Untuk studi kasus penulis akan menggunakan sistem trading saham sebagai masalah yang harus dipecahkan oleh mesin dengan target mendapatkan profit sebanyak-banyaknya dan kerugian sekecil-kecilnya, untuk itu penulis akan menggunakan *Reinforcement Learning* sebagai metode pembelajarannya.

Reinforcement Learning (RL) penulis pilih karena telah sangat sukses dalam perkembangan kecerdasan buatan (*Artificial Intelligence*) untuk membangun mesin yang mampu bermain game secara mandiri dan melebihi kemampuan pemain game profesional, bahkan mampu mengalahkan juara Go dunia Lee Se-dol, dimana permainan Go jauh lebih kompleks dari permainan catur dengan perbandingan jumlah kombinasi kemungkinan yang mencapai miliaran. Keberhasilan tersebut membuat penulis berhipotesis bahwa RL seharusnya bisa juga digunakan untuk berkompetisi dengan manusia dalam trading di pasar saham, karena apabila *Reinforcement Learning* telah berhasil mengalahkan juara game dunia maka seharusnya *RL* juga mampu untuk mengalahkan trader profesional dalam dunia trading (Van Otterlo, 2012).

Namun masalah klasik dari RL adalah adanya *exploitation and exploration problem*, yang membuat RL sering kesulitan mendapatkan konvergensi dalam proses trainingnya. Untuk itu penulis coba mengambil pendekatan lainnya dengan cara menggunakan AEG sebagai fungsi optimasinya.

Dari latar belakang tersebut di atas maka penulis sangat tertarik untuk mengangkat tema penelitian "Algoritma Evolusi Genetika Sebagai Fungsi Optimasi Jaringan Syaraf Buatan" dengan menggunakan sistem trading saham sebagai studi kasusnya.

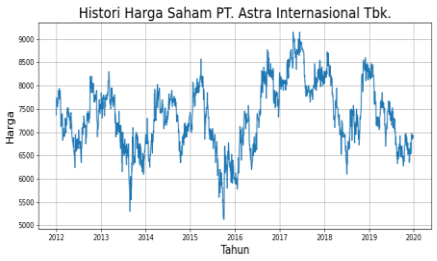
2. METODE

Penelitian ini merupakan penerapan algoritma evolusi genetika (AEG) sebagai fungsi optimasi pada jaringan saraf buatan untuk membangun sistem trading saham berbasis AI. Program yang dibuat akan menghasilkan output berupa sinyal kapan waktu beli/jual yang paling optimal, sehingga dapat meningkatkan profit. Pelaksanaan penelitian ini memiliki tahapan-tahapan sebagai berikut:

2.1 Pengumpulan Data

Pada tahap ini sebagai sampel data penulis akan menggunakan data histori harga saham PT. Astra Internasional Tbk. dengan kode saham ASII. Pemilihan ASII karena ASII memiliki tingkat volatilitas yang tinggi sehingga data lebih *stationary* tidak banyak memiliki kecenderungan trend turun atau naik, sehingga akan benar-benar menguji performa sistem.

Untuk mendapatkan data ASII penulis mengambil data histori dari Yahoo Finance, penulis menggunakan data dengan rentang waktu 8 tahun sejak 2012 hingga 2020 menggunakan data harian (daily timeframe).



Gambar 1. Pergerakan harga saham ASII

2.2 Alat Penelitian

Penelitian ini menggunakan alat berupa perangkat keras dan lunak agar bisa berjalan. Adapun perangkat keras yang digunakan adalah sebuah komputer dengan spesifikasi sebagai berikut:

1. Prosesor Intel Core i7-9750H
2. Grafis NVIDIA GeForce RTX 2060
3. Memori 32 GB
4. Storage 1TB SSD

Kemudian perangkat lunak yang digunakan adalah sebagai berikut:

1. Sistem operasi Windows 10 (sebagai host operating system).
2. WSL 2 (Windows Subsystem Linux 2).
3. Python 3.8.2
4. Numpy 1.18.0
5. Pandas 1.1.1
6. Backtesting.py 0.3.0

Sistem operasi yang penulis gunakan untuk melaksanakan penelitian ini adalah menggunakan Windows 10 dengan Windows Subsystem Linux 2 (WSL 2). WSL adalah sebuah subsistem berbasis Linux yang disediakan oleh Microsoft di Windows 10, WSL ini mempermudah penulis dalam menjalankan perintah-perintah UNIX di lingkungan Windows.

Python 3.8 penulis gunakan sebagai bahasa pemrograman dasar untuk membangun sistem dikarenakan Python memiliki banyak sekali pustaka (library) yang tersedia dan banyak mendukung pelaksanaan penelitian ini.

Numpy adalah pustaka Python yang penulis gunakan untuk membantu dalam melakukan operasi matematika terutama ketika melakukan kalkulasi pada vector, matrix dan tensor, yang mana akan sangat merepotkan tanpa bantuan Numpy (Zein, 2018).

Pandas adalah pustaka Python yang penulis gunakan untuk membaca data, mengolah data dan normalisasi data. Pandas bisa membaca data csv dan kompatibel dengan Numpy untuk melakukan operasi aritmatika di dalamnya (Vegari, 2020).

Untuk visualisasi dan analisis hasil penulis menggunakan Backtesting.py versi 0.3.0 yang tersedia sebagai pustaka Python. Backtesting ini bisa membuatkan visualisasi performa sistem dalam bentuk *chart* dan output seperti sharpe ratio dan SQN.

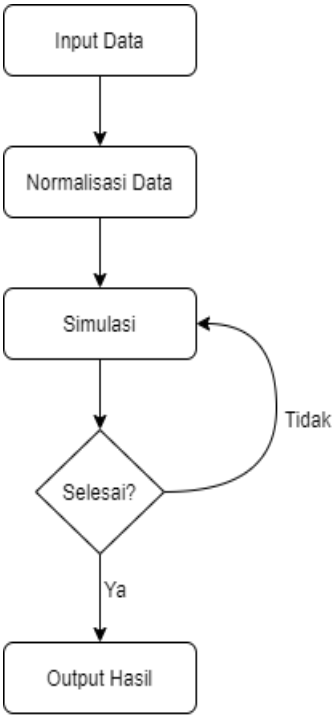
2.3 Pembuatan Program

Penulis membuat program menggunakan bahasa pemrograman Python. Adapun tahapan dalam pembuatan program adalah sebagai berikut:

a. Penyesuaian program agar dapat membaca format input data histori harga yang berasal dari Yahoo Finance berupa file csv. Penulis menggunakan Pandas Python library untuk melakukan proses ini.
b. Pembuatan fungsi normalisasi data.
c. Pembuatan simulator trading, pada tahapan ini program nantinya dapat mensimulasikan proses transaksi trading.
d. Pembuatan fungsi reward.
e. Pembuatan fungsi pelaporan hasil.

2.4 Cara Kerja Program

Program pada penelitian ini memiliki alur kerja sebagai berikut:



Gambar 2. Alur kerja program

3. HASIL DAN PEMBAHASAN

3.1 Proses Kerja Program

3.1.1 Input Data

Input data adalah titik awal dimana data pertama kali mulai dimasukkan ke dalam program untuk diproses. Data ini berupa data histori harga saham ASII dalam format csv OCHLVA (Open, Close, High, Low, Volume, Adj Close).

3.1.2 Normalisasi Data

Normalisasi Data adalah tahapan dimana dilakukan normalisasi data dari bentuk asli ke bentuk yang lebih mudah diproses oleh *machine learning* (Nasution, 2019), yakni dengan cara merubahnya menjadi *stationary*. Hal ini dibutuhkan karena untuk meminimalisir bias karena adanya *trend*, data yang *stationary* membuatnya tidak memiliki trend. Untuk membuat data yang *stationary* penulis cukup mentransformasikan data harga saat ini dengan harga sebelumnya: $C_t - C_{t-1}$

Date	High	Low	Open	Close	Volume	Adj Close
2012-01-02	7400.0	7350.0	7370.0	7370.0	2365000.0	4632.886230
2012-01-03	7500.0	7400.0	7500.0	7500.0	16420000.0	4714.605957
2012-01-04	7790.0	7595.0	7715.0	7715.0	25860000.0	4849.758301
2012-01-05	7860.0	7620.0	7745.0	7745.0	22700000.0	4868.616699
2012-01-06	7750.0	7650.0	7730.0	7730.0	9035000.0	4859.187500

Gambar 3. Format data dari Yahoo Finance

Tidak semua data pada kolom tersebut penulis gunakan, penulis hanya menggunakan data pembukaan (open), penutupan (close), dan volume saja yang akan diproses.

Data mentah tersebut perlu dilakukan normalisasi agar *stationary* dengan cara mengubahnya menjadi relatif terhadap hari sebelumnya. Ini bisa dilakukan dengan mengurangi harga hari saat itu dengan harga pada hari sebelumnya, maka akan didapatkan *delta*-nya.

Date	Open	Close	Volume
2012-01-03	130.0	130.0	14055000.0
2012-01-04	215.0	215.0	9440000.0
2012-01-05	30.0	30.0	-3160000.0
2012-01-06	-15.0	-15.0	-13665000.0
2012-01-09	-130.0	-130.0	12705000.0

Gambar 4. data setelah dinormalisasi

3.1.3 Simulasi

Simulasi adalah tahapan dimana proses transaksi terjadi, namun terjadi di dalam simulator, proses ini sering juga disebut sebagai backtesting.

Untuk simulasi, penulis menggunakan data 6 tahun awal dari tahun 2012 sampai 2018, sementara untuk pengujiannya penulis menggunakan *out sample* data 3 tahun dari tahun 2018 sampai tahun 2020.

Dalam simulator terdapat iterasi sebanyak jumlah hari pada data ASII, setiap iterasi mewakili hari yang dilalui. Kemudian dibuatkan variabel saldo yang akan digunakan oleh agen untuk transaksi di dalamnya, dalam simulator terdapat fungsi untuk melakukan aksi beli atau jual saham dengan harga saham

yang tersedia pada hari itu (hari dalam simulator) sesuai data histori ASII.

Proses-proses yang terjadi dalam simulasi:

a. Pembentukan populasi baru (para agen).
b. Pelatihan para agen.
c. Pembentukan agen super.
d. Ujicoba agen super.

Pembentukan populasi baru adalah proses di mana simulator akan membangkitkan populasi baru berisi para agen yang jumlahnya penulis batasi sebanyak 30 individu. Para agen ini diinisialisasi dengan jaringan saraf bernilai *weight* acak. Karena acak inilah simulator bersifat non deterministik, pengulangan operasi simulasi tidak akan selalu menghasilkan output yang sama persis, namun cenderung mirip. Sebenarnya penulis bisa menyetel *seed randomness* yang konstan untuk menghindari angka acak yang berbeda setiap pengujian, namun penulis lebih suka acak karena keacakan ini akan memberikan gambaran umum kemampuan sistem dalam ketidakpastian (uncertainty).

Setelah populasi terbentuk setiap agen atau individu dalam populasi tersebut akan diumpan (feed) *state*, *state* ini adalah data harga saham ASII yang telah dinormalisasi tadi. *Feed* dalam bentuk *state* ini oleh agen diproses dalam jaringan saraf masing-masing agen yang pada akhirnya akan memberikan output berupa *vector* dalam format *categorical binary* yang masing-masing memiliki arti sebagai berikut:

a. [1, 0, 0] - yang artinya tahan (hold) jangan melakukan transaksi apapun.
b. [0, 1, 0] - yang artinya beli (buy).
c. [0, 0, 1] - yang artinya jual (sell).

Output dari jaringan saraf tersebutlah yang menentukan agen apakah akan memutuskan untuk membeli atau menjual saham. Sistem dalam simulator tidak mengijinkan agen melakukan penjualan apabila tidak memiliki saham untuk dijual, sehingga agen harus membeli terlebih dahulu untuk bisa menjual, sebagai contoh apabila jaringan saraf memberikan perintah menjual namun agen tidak memiliki saham untuk dijual maka agen tidak akan bisa melakukannya dan akan terekam sebagai aksi yang tidak memberikan *reward* apapun. Begitu juga dengan aksi beli apabila agen tidak memiliki cukup saldo untuk membeli maka aksi tersebut tidak akan bisa dilakukan, apabila tetap dilakukan tidak akan ada *reward* apapun untuk agen.

Proses simulasi ini akan dilakukan terus menerus hingga 1000 kali putaran, penulis hanya membatasi 1000 kali putaran saja karena dirasa cukup untuk mendapatkan hasil yang bisa dinilai. Dalam sekali putaran ada 30 agen yang melakukan aktivitas trading dalam simulator secara paralel.

Setiap sekali putaran para agen akan mendapatkan reward. Penulis menggunakan SQN sebagai reward-nya, SQN ini dikalkulasikan hanya pada in-sample data yang dalam hal ini adalah 6 tahun dari tahun 2012 hingga tahun 2018.

Setelah selesai satu kali putaran, setiap agen dalam populasi akan diambil sampel genetiknya untuk membentuk agen super, sampel genetik ini adalah *weight* yang ada pada jaringan saraf masing-masing agen yang kemudian dikombinasikan ke dalam *weight* jaringan saraf agen super. Jaringan saraf agen super ini akan terbentuk dengan pengaruh terkuat berasal dari agen paling berprestasi, yakni agen dengan *reward* tertinggi dalam populasi. Kemudian simulator akan mulai lagi ke putaran selanjutnya, yang mana akan dibentuk lagi populasi baru generasi selanjutnya dengan penurunan kode genetik dari agen super sehingga generasi kedua ini akan memiliki sifat bawaan dari agen super atau bisa dibilang keturunan (*offspring*) dari agen super. Agar para agen keturunan tidak melakukan hal yang sama persis dengan agen super (karena keturunannya secara langsung) maka perlu dilakukan mutasi genetik. Mutasi genetik ini dilakukan dengan cara merubah beberapa kode genetik para keturunan agen super dengan kode genetik acak. Dengan seperti ini para agen baru akan memiliki perbedaan karakter dengan super agen, sehingga memberikan kemungkinan lebih besar salah satu dari 30 agen baru ini melakukan explorasi lebih luas dan tak terduga

dari leluhurnya sehingga akan memperbesar kemungkinan untuk menemukan optima baru. Setelah selesai lagi satu putaran, maka akan diambil lagi sampel kode genetik (weight) dari masing-masing jaringan saraf agen dan akan dilakukan rekombinasi ke jaringan saraf agen super lagi. Jadi agen super hanya ada satu yang selalu diperbaharui kode genetiknya dari para agen, sehingga agen super akan semakin cerdas. Demikian seterusnya dilakukan hingga 1000 kali putaran atau 1000 generasi.

Agen super akan diuji pada tiap akhir putaran simulasi pada keseluruhan data dari tahun 2012 hingga tahun 2020 (termasuk out-sample data), hasil uji inilah yang digunakan penulis sebagai tolok ukur kesuksesan agen super mencapai level tertinggi kecerdasannya.

Semakin banyak putaran tidak selalu membuat agen super semakin cerdas, terutama ketika berhadapan dengan *out-sample data* atau data yang belum pernah dilihat oleh agen. Ketika agen super memiliki performa yang sangat bagus pada tahun sebelum 2018 (in-sample data) sementara memiliki performa yang buruk di tahun setelah 2018 (out-sample data) itu artinya agen super telah mengalami *overfitting*, atau kondisi dimana agen super gagal mengeneralisir lingkungannya sehingga tidak mampu beradaptasi terhadap lingkungan baru.

Untuk menghindari terjadinya *overfitting* maka perlu dilakukan penghentian proses training/evolusi ketika skor SQN pada *out-sample data* menurun sementara SQN pada *in-sample data* meningkat. Atau tetap lakukan simulasi namun lakukan *snapshot* pada *weight* agen super setiap putarannya, sehingga nanti bisa kita pilih mana yang paling bagus rasio antara *SQN di in-sample data* dengan di *out-sample data*.

3.1.4 Output Hasil

Output hasil adalah tahapan dimana program telah menghasilkan hasil simulasi untuk setiap putarannya. Adapun output yang dihasilkan adalah sebagai berikut:

a. Total profit akhir (net worth).
b. Lama proses learning / simulasi.
c. Sharpe Ratio.
d. SQN (System Quality Number).

Total profit dikalkulasikan dengan rumus:

$$P = S - G$$

Di mana P adalah profit, S adalah saldo awal, dan G adalah gains atau pendapatan yang diperoleh agen dari transaksi yang dilakukan sejak tahun pertama sampai tahun kelima.

Lama proses learning / simulasi adalah perhitungan waktu yang dibutuhkan untuk menjalankan semua putaran simulasi dalam detik.

Sharpe Ratio adalah nilai perbandingan antara ROI dengan resikonya, semakin tinggi sharpe ratio-nya semakin bagus. Rumus dari sharpe ratio adalah:

$$SharpeRatio = \frac{R_p - R_f}{\sigma_p}$$

Dimana:

$$R_f = \text{risk-free rate}$$

$$\sigma = \text{standard deviation of the excess return}$$

$$R_p = Returns$$

SQN (*System Quality Number*) adalah sebuah penilaian scoring system yang diciptakan oleh Van K. Tharp (1998) untuk menilai seberapa berkualitas sebuah sistem atau strategy trading (Y Zhang, 2005). SQN memiliki formula sebagai berikut:

$$SQN = \sqrt{N} \cdot \frac{\mu_w}{\sigma_w}$$

Dimana:

$$N = \text{jumlah transaksi yang dilakukan}$$

$$\mu_w = \text{rata-rata profit dan loss}$$

$$\sigma_w = \text{standar deviasi profit dan loss}$$

SQN dianggap efektif apabila jumlah transaksi lebih dari 30. Nilai skor SQN apabila dikelompokkan dapat dibagi menjadi 6 level:

Tabel. 1 Level Skor SQN

SQN Score	System Quality
< 1	Buruk
1-2	Lumayan

2-3	Bagus
3-5	Bagus Sekali
7-7	Luar Biasa
>7	Sempurna

3.1.5 Arsitektur Jaringan Saraf Agen

Penulis mendesain arsitektur jaringan saraf menggunakan 3 *linear layer* yang masing-masing *layer* memiliki bentuk atau dimensi 27 kali 276. Masing-masing *layer* tidak saling terkoneksi namun hasil kalkulasi akan disatukan menggunakan rata-rata pada *layer* terakhir.

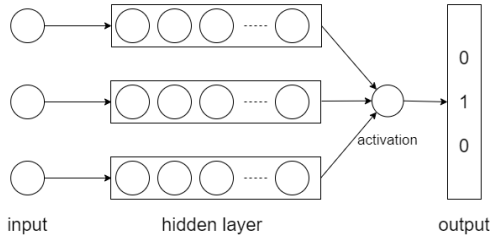
Cara kerja *linear layer* sama seperti *dense layer* yakni x dikalikan *weight* ditambah *bias* bedanya penulis tidak menggunakan *activation function* pada kalkulasi terakhirnya:

$$y = x \cdot w + b$$

Dense layer (non linier):

$$y = f(x \cdot w + b)$$

dimana f adalah activation function non-linier



Gambar 5. rancangan arsitektur jaringan saraf agen

State atau tiga input adalah data open, close, dan volume sepanjang 27 hari ke belakang yang telah dinormalisasi.

Penulis menggunakan *mean* (rata-rata) sebagai *activation function* akhir menggunakan *axis* 0 sehingga mengubah bentuk output dari dua dimensi menjadi satu dimensi. Hasil output tidak selalu absolut dalam bilangan biner 0 atau 1, pada kenyataannya hanya berupa probabilitas dengan range 0 - 1, contoh:

$$y = [0.172, 0.8, 0.3]$$

Sehingga perlu dilakukan argmax untuk mendapatkan posisi index dengan nilai tertinggi dalam vector:

$$\text{decision} = \text{argmax}(y)$$

Dalam contoh ini decision akan berisi 1 (index dihitung mulai dari 0) karena nilai tertinggi ada pada urutan ke dua (0.8). Nilai 1 artinya beli, 2 jual, dan 0 tahan. Inilah keputusan akhir yang diambil oleh agen untuk melakukan aksi.

3.2 Pembahasan Hasil

Pada bagian ini akan dibahas hasil penelitian dengan cara melakukan komparasi data output dari dua program seperti yang telah dibahas pada bagain sebelumnya.

3.2.1 Pembahasan

Berikut adalah penerapan algoritma AEG yang penulis gunakan sebagai fungsi optimasi pada Reinforcement Learning yang melingkupi proses (Ismail, 2012):

1. Pembentukan populasi
2. Mutasi
3. Rekombinasi / Persilangan

3.2.2 Pembentukan Populasi

Penulis set basis populasi konstan sebanyak 30, pembentukan populasi baru menggunakan nilai acak untuk setiap *weight* pada sel di *neural net-nya*. Pembentukan populasi baru memiliki formula sebagai berikut:

$$P = \{\text{random}(w_a \cdot w_b) : x \in (1, 2, 3, \cdots, 30)\}$$

3.2.3 3.1.2. Mutasi

Lalu untuk setiap individu dalam populasi baru tersebut akan dilakukan mutasi, dengan formula:

$$W = \{w + (\sigma \cdot x) : x \in P_i\}$$

Operasi mutasi tersebut menghasilkan *weight* baru, yang selanjutnya *weight* ini akan dimasukkan ke dalam simulator, di mana simulator ini penulis jadikan sebagai fungsi

reward-nya. Hasil dari simulasi adalah *rewards* (*R*), seperti dalam notasi berikut:

$$R = \{f(w) : w \in W\}$$

3.2.4 3.1.3. Rekombinasi / Persilangan

Proses rekombinasi / persilangan ini dilakukan dengan cara:

$$S = \{f(x) : x \in P\}$$

di mana:

$$f(x) = w + \frac{lr}{30 \cdot \sigma} \cdot (p^T \cdot R_s)^T$$

di mana:

$$R_s = \frac{\mu_r}{\sigma_r}$$

di mana:

w = weight setiap individu dalam populasi
lr = learning rate
σ = sigma
p^T = weight setiap populasi yang telah ditranspose
μ_r = rata-rata rewards keseluruhan populasi
σ_r = standar deviasi rewards keseluruhan populasi

Dari hasil rekombinasi tersebut maka didapatkan satu individu dengan genetika gabungan dari seluruh populasi dengan reward terbaik sebagai karakter yang paling kuat dan berpengaruh. Individu hasil rekombinasi inilah yang kemudian dilakukan simulasi untuk mendapatkan reward final dalam satu kali putaran (generasi).

3.2.5 Hasil

Dari proses komputasi / training maka dihasilkan data sebagai berikut:

Tabel 2. Hasil Proses Komputasi

Total Profit (net worth)	Rp. 348.000.000,- (596%)
Lama Proses Training	3 menit 7 detik
Sharpe Ratio	0.895
SQN	2.17

Data tersebut merupakan hasil dari kalkulasi secara keseluruhan sampel + out sample data.

Bisa dilihat bahwa RL+AEG dapat menyelesaikannya hanya dalam waktu 3 menit 7 detik, dan bisa mendapatkan profit 596%.

Gambar 6 performa AEG

Pada gambar 6 bisa dilihat bahwa AEG mampu meningkatkan profit setiap tahunnya, dan yang lebih menarik lagi adalah AEG tetap memiliki performa yang baik pada *out sample data* (setelah 2018 ke atas), bahkan sistem berbasis AEG mampu melewati masa pandemi di awal-awal tahun 2020 dengan sempat merugi (max drawdown) -32% yang kemudian berhasil *recovery* dan profit kembali hingga awal desember 2020.

4. PENUTUP

4.1. Kesimpulan

Berdasarkan hasil penelitian yang telah penulis jabarkan sebelumnya, maka dapat diambil kesimpulan bahwa algoritma evolusi genetika sangat efektif diterapkan sebagai fungsi optimasi *pada Reinforcement Learning* untuk membangun sistem trading saham berbasis kecerdasan buatan.

4.2. Saran

Berdasarkan uraian hasil analisis terhadap penggunaan algoritma evolusi genetika

sebagai fungsi optimasi pada RL untuk membangun sistem trading berbasis kecerdasan buatan, maka penulis memberikan saran sebagai berikut:

1. Bagi para pengembang sistem kecerdasan buatan terutama yang berbasis RL, maka AEG bisa dijadikan alternatif yang bagus untuk menangani masalah pada domain yang reward-nya bersifat sparse.
2. Bagi para peneliti selanjutnya diharapkan mengadakan penelitian yang lebih mendalam tentang AEG dalam penerapannya di domain yang lain diluar sistem trading saham, sehingga dapat melengkapi bukti empiris dan mengetahui efektifitas AEG dalam menangani masalah yang lain.

5. DAFTAR PUSTAKA

Van Otterlo, M., & Wiering, M. (2012). Reinforcement learning and markov decision processes. *In Reinforcement learning* (pp. 3-42). Springer, Berlin, Heidelberg.

Zein, A. (2018, December). Menggabungkan Dua Wajah Dengan Metoda Ensemble Regression Trees Menggunakan Pustaka Dlib Dan Opencv Python. *In ESIT* (Vol. 13, No. 2, pp. 30-36).

Vegari, A., & Budi, S. (2020). Implementasi Exploratory Data Analysis Pada Dataset Video Trending Harian YouTube. *Jurnal STRATEGI-Jurnal Maranatha*, 2(2), 397-411.

Nasution, D. A., Khotimah, H. H., & Chamidah, N. (2019). Perbandingan Normalisasi Data untuk Klasifikasi Wine Menggunakan Algoritma K-NN. *CESS (Journal of Computer Engineering, System and Science)*, 4(1), 78-82.

Zhang, Y., Chen, W., Zhang, X., Wu, Y., & Yan, Q. (2005, January). Synthetic evaluation and neural-network prediction of laser cutting quality. *In Lasers in Material Processing and Manufacturing II* (Vol. 5629, pp. 237-246). International Society for Optics and Photonics.

Ismail, S. J. I. (2012). Analisis Aplikasi Pencarian Exploitable Bugs Secara Otomatis. *Jurnal Teknologi Informasi*, 1(3), 94-101.