MENEMUKAN AKAR PERSAMAAN POLINOMIAL MENGGUNAKAN PARTICLE SWARM OPTIMIZATION

[1]Oei,EdwinWicaksonoDarmawan, [2]Suyanto Edward Antonius, Ir., M.Sc
[1,2]Program Studi Teknik Informatika Universitas Katolik Soegijapranata
[1]edwinwicaksonodarmawan@gmail.com, [2]seantonius@unika.ac.id

Abstract

Polynomial is a mathematic function which involves multiplication, exponent, and variable. There are some methods to find the root of the function. The method will be more complicated if the highest exponent of the function is 3 or greater. This project will solve the problem using Particle Swarm Optimization algorithm to find the root with an accurate number.

Keywords: Particle Swarm Optimization, Polynomial

Pendahuluan

Polinomial adalah salah satu dari fungsi matematika yang melibatkan perkalian, perpangkatan, dan nilai variable. Untuk mencari akar persamaan dari fungsi tersebut, diperlukan untuk mencari nilai variabel yang mempunyai nilai fungsi 0 (f(x) = 0). Ada beberapa metode untuk mencari akar persamaan tersebut. Hanya saja, untuk mencari akar persamaan pada persamaan polinomial dengan pangkat tertinggi 3 atau lebih dibutuhkan metode lain yang lebih rumit.

Projek ini akan menggunakan Algoritma Particle Swarm Optimization untuk mencari akar tersebut. Algoritma ini adalah salah satu algoritma optimasi yang berfungsi untuk mencai solusi paling optimal. Algoritma ini menggunakan angka real untuk pengoperasiannya sehingga lebih mudah dalam implementasi. Diharapkan algoritma ini dapat mecari akar persamaan yang paling optimal atau paling dekat dengan hasil.

Landasan Teori

Polinomial

Polinomial adalah salah satu persamaan dalam matematika yang melibatkan perkalian, pangkat, dan nilai variabel. Berikut adalah bentuk dasar dari polinomial :

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \ldots + a_0$$

dengan adalah koefisien, x adalah variabel yang akan dicari, n adalah pangkat tertinggi, f(x) adalah nilai fungsi dari persamaan tersebut, dan adalah konstanta. n adalah bilangan cacah.

Akar persamaan polinomial adalah nilai dari x variabel yang mempunyai nilai f(x) = 0. Jadi, bila nilai x dimasukkan dalam persamaan, hasilnya adalah nol.

Particle Swarm Optimization

Particle Swarm Optimization adalah algoritma optimasi yang digunakan untuk pencariaan solusi sehingga mendapatkan hasil yang optimal. Algoritma ini meniru sifat segerombolan binatang dalam mencari makan seperti burung dan ikan. Bila ada salah satu burung yang lokasinya paling dekat dengan sumber makanan, maka burung lain akan pergi menuju ke sumber makanan dengan mengikuti burung yang paling dekat.

Burung dilambangkan dengan partikel dalam algoritma. Setiap partikel mempunyai 2 karakteristik, yaitu lokasi dan kecepatan. Setiap iterasi, partikel tersebut akan bergerak mengikuti partikel lain yang mempunyai solusi yang lebih baik dengan cara memperbaharui kecepatan dan lokasinya. Berikut adalah rumus untuk memperbaharui kecepatan dan lokasi:

$$v_n = w \cdot v_{n-1} + c_1 \cdot r_1 \cdot \left(Pbest - x_{n-1}\right) + c_1 \cdot r_1 \cdot \left(Gbest - x_{n-1}\right)$$

$$x_n = x_{n-1} + v_n$$

V = kecepatan

X = lokasi partikel

R1 = r2 = angka acak dari 1-0

W = Bobot Inersia

C1 = learning rates (partikel)

C2 = learning rates (global)

Selama partikel bergerak di setiap iterasi, mereka juga menyimpan nilai terbaik yang selama ini ia dapat dan nilai terbaik dari semua partikel juga. Nilai terbaik setiap partikel dinamakan Particle Best dan nilai terbaik dari semua partikel adalah Global Best. Setiap partikel bergerak dan berpindah tempat dengan mempertimbangkan kedua hal ini.

Metodologi Penelitian

Berikut adalah langkah yang dirperlukan untuk menyelesaikan proyek ini:

1. Mencari referensi dari berbagai sumber

   Langkah ini adalah mencari referensi dari berbagai sumber seperti buku, penelitian sebelumnya, jurnal, dan lainnya. Langkah ini berfungsi untuk mempelajari algoritma yang dipakai dan masalah yang akan diselesaikan.

2. Menganalisa dan membuat desain program dari permasalahan dan algoritma yang digunakan.

   Langkah ini menganalisa masalah dan algoritma sehingga algoritma tersebut dapat diimplementasikan untuk menyelesaikan masalah. Lalu, membuat rancangan program sehingga lebih mudah dalam implementasi.

3. Mengimplementasi algoritma ke dalam program.

   Langkah ini adalah mengimplementasikan algoritma ke dalam program. Program yang dibuat akan berfungsi untuk menyelesaikan masalah yang ada dengan algoritma.

4. Melakukan percobaan .

   Langkah ini adalah melakukan percobaan – percobaan dari program yang sudah dibuat. Percobaan ini bisa berfungsi untuk memperbaiki program agar bisa lancar dalam penggunaan dan juga untuk mempelajari algoritma lebih lanjut sehingga bisa mengetahui karakteristik algoritma.

Hasil dan Pembahasan

Berikut ini adalah beberapa hasil percobaan saat mengganti nilai setiap parameter dalam algoritma. Parameter tersebut adalah jumlah partikel, iterasi maksimum, bobot inersia, nilai random (r1 dan r2), dan learning rates (c1 dan c2). Percobaan ini dilakukan untuk mencari parameter paling cepat dengan tingkat kesuksesan tinggi dalam mendapatkan solusi. Percobaan ini dilakukan dengan cara melakukan 10 kali percobaan pada setiap nilai parameter. Hasil percobaan dikatakan buruk bila iterasi yang diperlukan terlalu besar, tingkat kesuksesan terlalu kecil, atau nilai 0 pada grafik yang berarti tidak berhasil mendapatkan solusi.

Persamaan polinomial pertama adalah $x^3-4x^2-7x+10$ dan ersamaan polinomial kedua adalah : $5x^5+x^6-x^4+3x^7-x^3-4x^2-7x+1000$. Pertama, percobaan akan menggunakan jumlah partikel 1000, maksimum iterasi 1000, r1

dan r2 dengan angka random, bobot inersia dengan rumus $w_{max} - (\frac{w_{max} - w_{min}}{i_{max}})i$. w_{max} adalah 0.9 dan w_{min} adalah 0.4, dan Learning rates c1 dan c2 dengan nilai 2.0. Percobaan kedua akan menggunakan nilai 0.1 untuk semua parameter kecuali jumlah partikel dan iterasi maksimum.

Ini adalah percobaan saat mengganti jumlah partikel. Dapat dilihat bahwa semakin banyak jumlah partikel, maka iterasi yang dibutuhkan untuk mencari solusi juga semakin sedikit. Tingkat kesuksesan tidak terlalu dipengaruhi oleh jumlah partikel. Perbedaan permasalahan juga tidak terlalu mempengaruhi perbedaan hasil. Hanya lebih lama dari pertama.

Tabel 1: Tabel Hasil saat Mengganti Jumlah Partikel

Polinomial	1		2	
Jumlah partikel	iterasi	persentase	iterasi	persentase
10	430	100	502.4	100
100	383.2	100	477.6	100
1000	203.4	100	448.6	100
10000	201.6	100	499.6	100

Saat mengganti nilai iterasi maksimum, algoritma bisa mendapatkan hasil lebih sering dengan iterasi maksimum yang besar. Tetapi iterasi yang dibutuhkan juga semakin banyak sehingga akan semakin lama dalam mencari hasilnya. Algoritma gagal dalam mendapatkan hasil saat iterasi maksimum hanya 10 pada persamaan pertama. Tetapi pada persamaan kedua, algoritma gagal pada iterasi maksimum 100.

Tabel 2: Hasil saat Mengganti Iterasi Maksimum

Polinomial	1		2	
Iterasi Maksimum	iterasi	persentase	iterasi	persentase
10	0	0	0	0
100	86.75	80	0	0
1000	203.4	100	448.6	100
10000	1549.4	100	2100.2	100

Saat mengganti nilai bobot inersia, algoritma bisa mendapat hasil dengan iterasi yang sedikit saat nilai parameter sedikit seperti percobaan sebelumnya. Bila nilai di 0.9 ke atas, algoritma gagal mendapatkan hasilnya. Hal ini sama untuk kedua persamaan.

Tabel 3: Hasil saat Mengganti Bobot Inersia

Polinomial	1		2	
Bobot Inersia	iterasi	persentase	iterasi	persentase
with Formula	203.4	100	448.6	100
0.1	16.8	100	27.6	100
0.4	37	100	69.6	100
0.9	0	0	0	0
1	0	0	0	0
2	0	0	0	0

Saat mengganti nilai random (r1 dan r2), algoritma bisa mendapatkan hasil saat menggunakan nilai dibawah 1. Untuk persamaan kedua, algoritma bisa mendapatkan hasil bila menggunakan nilai dibawah 0.9. Algoritma juga semakin cepat dalam mencari dan mendapatkan solusi dengan nilai yang kecil karena iterasi yang dibutuhkan sedikit, tetapi tidak terlalu banyak perbedaan besar.

Tabel 4: Hasil saat Mengganti Angka Random

Polinomial	1		2	
r1 dan r2	iterasi	persentase	iterasi	persentase
random	203.4	100	448.6	100
0.1	132.2	100	233	100
0.5	129.6	100	213.6	100
0.9	163.2	100	0	0
1	0	0	0	0
2	0	0	0	0

Saat mengganti nilai learning rates, Algoritma bisa mendapatkan hasil dengan iterasi yang sedikit bila nilai yang digunakan juga kecil. Algoritma gagal menemukan hasil saat menggunakan nilai yang besar seperti 4. Bila ingin

membedakan nilai antara c1 dan c2, iterasi yang dibutuhkan akan semakin sedikit bila memberikan bobot lebih kepada c1. Iterasi yang dibutuhkan semakin sedikit.

Tabel 5: Hasil saat Mengganti Learning Rates

Polinomial		1		2	
Learning Rates		iterasi	persentase	iterasi	persentase
c1	c2				
4	4	0	0	0	0
2	2	203.4	100	448.6	100
1	1	169	100	291.6	100
0.1	0.1	142.4	100	229.6	100
1	3	496	100	601.2	100
3	1	333.8	100	535.4	100
0.1	3.9	851.6	100	828.6666667	60
3.9	0.1	714.2	100	788.2222222	90
0.1	1	180.4	100	277.2	100
1	0.1	163.4	100	246	100

Dari percobaan di atas, dapat dilihat bahwa bila salah 1 parameter kecil, maka iterasi yang dibutuhkan juga sedikit dan tingkat kesuksesan semakin tinggi. Tetapi bila memberikan nilai 0.1 pada semua parameter kecuali jumlah partikel dan iterasi maksimum, algoritma menjadi semakin buruk dalam mencari solusi. Iterasi yang dibutuhkan menjadi semakin banyak dan tingkat kesuksesan menurun. Perbedaan persamaan tidak terlalu mempengaruhi perbedaan hasil. Hanya saja semakin rumit persamaannya, maka iterasi yang dibutuhkan juga semakin banyak.

Di bawah adalah hasil percobaan bila semua nilai parameter adalah 0.1 dan menggunakan persamaan pertama. Dapat dilihat bahwa saat mengganti nilai bobot inersia, nilai yang tinggi semakin bagus kinerja algoritma. Iterasi yang dibutuhkan semakin sedikit dan tingkat kesuksesan meningkat.

Tabel 6: Hasil saat semua parameter 0.1 dan mengganti nilai Bobot Inersia

Polinomial	1, semua parameter 0.1	
Bobot Inersia	iterasi	persentase
with Formula	137	100
0.1	962.3333333	30
0.4	534.8	100
0.9	211.2	100

Saat mengganti nilai r1 dan r2, Iterasi yang dibutuhkan juga semakin sedikit dan tingkat kesuksesan meningkat dengan nilai yang lebih besar, sama seperti percobaan sebelumnya.

Tabel 7: Hasil saat semua parameter 0.1 dan mengganti nilai Angka Random

Polinomial	1, semua parameter 0.1	
r1 dan r2	iterasi	persentase
random	199.4	100
0.1	962.3333333	30
0.5	206.4	100
0.9	94.8	100

Saat mengganti nilai learning rates, hasil yang didapat sama dnegan percobaan lain. Algoritma dapat mencari hasil dengan iterasi yang sedikit dan tingkat kesuksesan tinggi bila menggunakan nilai parameter yang kecil. Tetapi bila ingin membuat nilai c1 dan c2 berbeda, algoritma akan bekerja lebih baik bila memberikan bobot lebih kepada c2.

Tabel 8: Hasil saat semua parameter 0.1 dan mengganti nilai Learning Rates

Polinomial		1, semua parameter 0.1	
Learning Rates		iterasi	persentase
c1	c2		
0.1	0.1	962.3333333	30
1	1	99	100
2	2	45.8	100
4	4	12.6	100

0.1	3.9	19.6	100
3.9	0.1	644.8	100
1	3	28.8	100
3	1	90.4	100
4	0	0	0
0	4	16.4	100

Kesimpulan

Berikut adalah kesimpulan dari proyek:

1. Algoritma Particle Swarm Optimization bisa digunakan untuk mencari akar persamaan polinomial.
2. parameter yang digunakan dalam algoritma tidak bisa dimasukkan dengan sembarang. Parameter perlu disesuaikan masing – masing untuk bisa bekerja dengan baik dan maksimal dalam mencari solusi paling optimal.
3. angka acak yang digunakan dalam perhitungan kecepatan dalam algoritma tidak mempunyai efek terlalu besar dalam pencarian solusi optimal karena algoritma tetap bisa mendapatkan solusi optimal walaupun tanpa menggunakan angka acak.
4. Parameter terbaik yang didapat dari project ini:
   - Jumlah partikel = 1000
   - Iterasi masimum = 1000
   - bobot inersia = 0.1
   - angka random (r1 dan r2) = 0.1
   - learning rates (c1 dan c2) = 4

Daftar Pustaka

[1] Xiaohui Hu (2006). PSO Tutorial. Accessed on Sebtember 2016 from World Wide Web: http://www.swarmintelligence.org/tutorials.php

[2] Rumus Matematika (2013). Metode Pembagian Suku Banyak / Polinomial. Accessed on Sebtember 2016 from World Wide Web: http://rumus-matematika.com/suku-banyak-polinomial/

[3] MathsIsFun.com (2013). Polynomials. Accessed on Sebtember 2016 from World Wide Web: http://www.mathsisfun.com/algebra/polynomials.html

[4] Wati, D.A.R. & Rochman, Y.A. (2013). Model Penjadwalan Matakuliah Secara Otomatis Berbasis Algoritma Particle Swarm Optimization (PSO). *Jurnal Rekayasa Sistem Industri.* Vol. 2, No. 1, 22-31.

[5] Hoesen, D. (2012). Proses Pencarian Akar Persamaan Matematika Menggunakan Metode Bagi-Dua. *Strategi Algoritma***.** 1-6.

[6] Bansal, J. C. , Singh, P. K. , Saraswat, M. , Verma, A. , Jadon, S.S. , & Abraham, A. (2011).
Inertia Weight Strategies in Particle Swarm Optimization. *Third World Congress on Nature and Biologically Inspired Computing.* 640 – 647.