بسم الله الرحمن الرحیم

مبانی و کاربردهای هوش مصنوعی – پروژه اول – گزارش سوال دو

پرهام رحیمی – ۹۵۳۱۰۳۱

تولید داده ورودی:

برای تولید داده ورودی، کلاس InputGenerator.java را به شکل زیر در کلاس Main.java صدا می زنیم.

```
InputGenerator inputGenerator = new InputGenerator();
inputGenerator.execute();
```

پس از اجرا فایل input.csv شامل ۱۰۰ نقطه در بازه x ۰ تا ۱۰ و با فاصله یکسان و y مطابق با خروجی y چندجمله ای درجه ۳ ایجاد می شود.

نحوه اجرای برنامه:

برای اجرای رگرسیون، کلاس Generic.java را به شکل زیر در کلاس Main.java صدا می زنیم.

```
Generic generic = new Generic();
generic.execute();
generic.printPopulation();
generic.printIndividual(generic.findBestCorrectedIndividual());
```

پس از اجرا الگوریتم ژنتیکی پیاده سازی شده به ترتیب به اجرای توابع خود می پردازد، ابتدا جمعیت اولیه تولید شده به صورت رندم را چاپ بعد از گذشتن تعداد نسل منظور، جمعیت نهایی چاپ می گردند. در صورتی که به تابع سازنده Generic ورودی ندهیم، اعداد پیش فرض به شرح زیر فرض می شوند:

```
public Generic() {
    this.numberOfGenerations = 1000;
    this.populationSize = 10000;
    this.tournamentSize = 2;
    this.mutationRate = 0.01;
    this.sigma = 0.2;
    random = new Random();
}
```

اگر بخواهیم خودمان به این متغیر ها مقدار بدهیم کافیست از سازنده زیر استفاده کنیم:

```
public Generic(int numberOfGenerations, int populationSize, int tournamentSize,
double mutationRate, double sigma) {
    this.numberOfGenerations = numberOfGenerations;
    this.populationSize = populationSize;
    this.tournamentSize = tournamentSize;
    this.mutationRate = mutationRate;
    this.sigma = sigma;
    random = new Random();
}
```

پس از آن در صورت نیاز به چاپ نمودار رگرسیون انجام شده از صدا زدن متد زیر استفاده می کنیم:

```
PredicationPlotting.run(args, generic.findBestCorrectedIndividual().genes);
```

و درصورت نیاز به چاپ فیتنس و مشاهده شرایط همگرایی از متد زیر استفاده می کنیم:

```
PredicationPlotting.run(args, generic.getBestFitnessInGenerations(),
generic.getWorstFitnessInGenerations(), generic.getAverageFitnessInGenerations());
```

خواندن از فایل:

فایل input.csv خوانده می شود و محتویاتش در یک Arraylist از double ذخیره می شود.

تولید جمعیت اولیه:

به تعداد populationSize که از پارامتر های قابل تنظیم مسئله است، individual با ژن های رندوم تولید می کنیم. این ژن ها که در واقع ضریب های چند جمله ای ما هستند، در حالت رندوم بین ۵۰۰۰- تا ۵۰۰۰+ تولید می شوند.

حلقه اصلی برنامه:

پس از ایجاد جمعیت اولیه وارد اجرای حلقه های اصلی برنامه می شویم. در هر iteration حلقه بیرونی، ابتدا حلقه درونی صدا زده می شود؛ در هر iteration حلقه درونی دو والد به روش tournament selection انتخاب می شوند. و تابع crossover برای آن دو صدا زده می شود و فرزند تولید می شود. پس از آن double رندومی تولید می شود که در صورتی که کمتر از

mutation rate باشد آن فرزند mutate می شود. این حلقه (درونی) به تعداد اندازه جمعیت اولیه تکرار می شود. و پس از پایان تابع remaining بر روی جامعه فرزندان + نسل قبل صدا زده می شود و نسل بعد را می سازد. پس از پایان حلقه درونی نیز بهترین و بدترین و متوسط شایستگی افراد جعیت موجود ذخیره می شوند. حلقه بیرونی به تعداد numberOfGenerations ادامه می یابد.

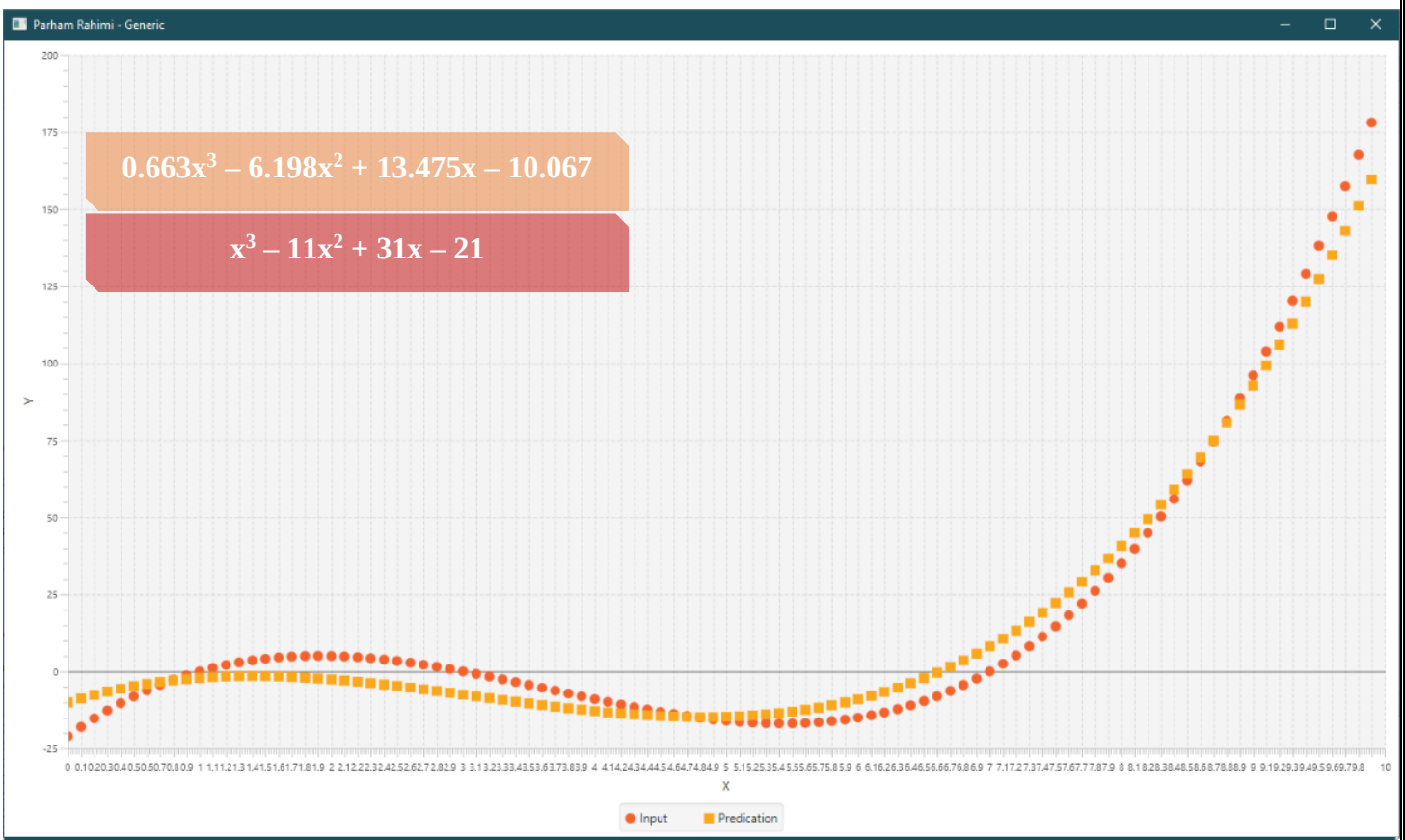
بررسی تاثیر پارامترها:

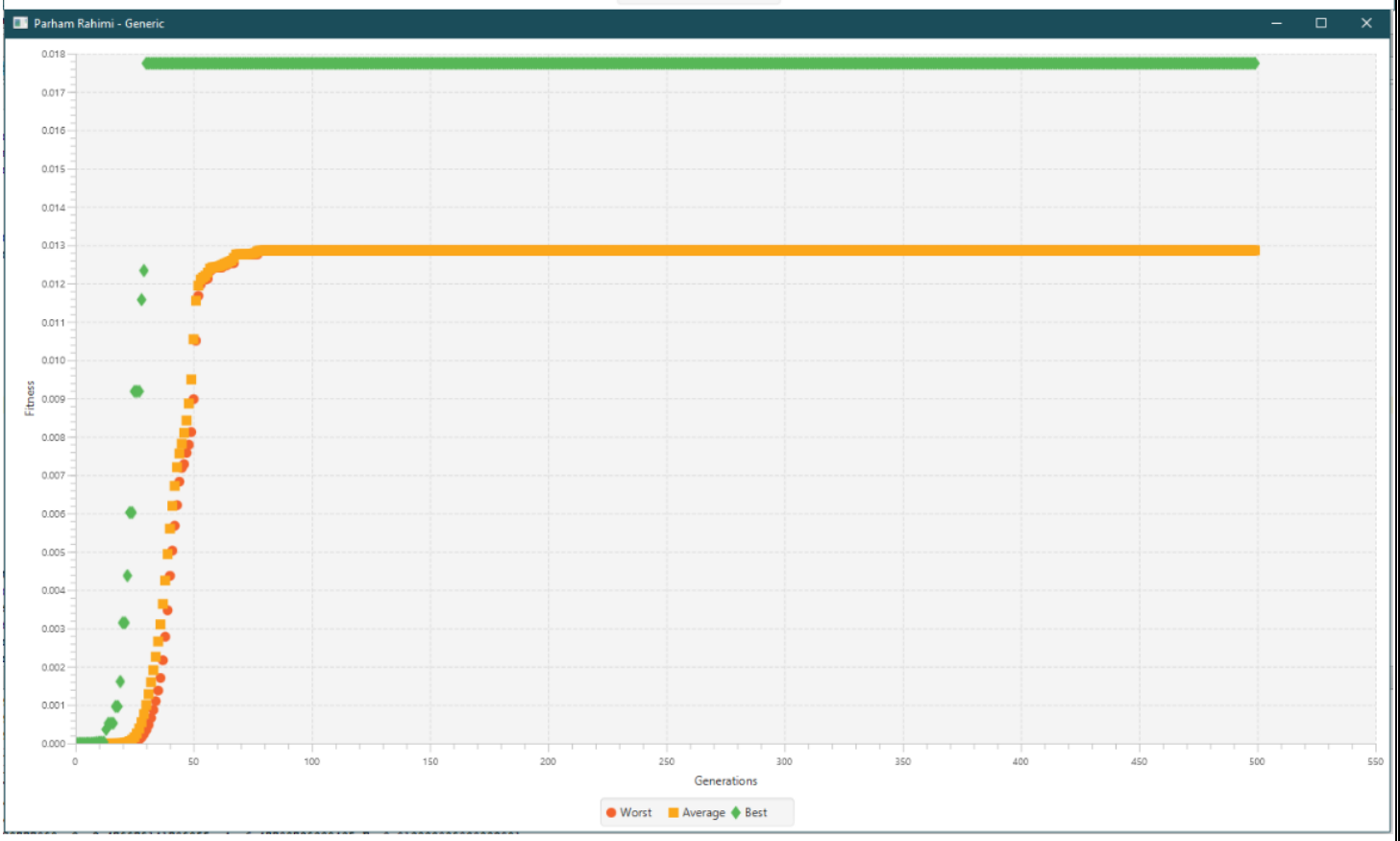
تاثیر تعداد نسل ها (numberOfGenerations):

معمول الگوریتم های یادگیری این است که با افزایش تعداد نسل ها از ۰ تا حد بهینه ای بهبود می یابند و پس از آن اصطلاحا Overfit می شوند به این معنا که برای تنها برای داده تربیت، شایستگیشان بهبود می یابد اما در کل برای داده های گسترده تر ضعیف تر عمل می کنند. این الگوریتم ژنتیکی نیز از این قاعده مستثنی نیست البته با توجه به اینکه در این مسئله ما داده تربیت و تست جداگانه نداریم، overfit شدن قابل مشاهده نیست و تنها underfit را می بینیم. هرچند با توجه به بررسی ژن های فرد برتر جمعیت نهایی می توان حدودا متوجه این اشتباه شد (به عنوان مثال اگر شایستگی قابل قبولی داشته باشد اما ضریب x^3 آن نزدیک به ضریب واقعی نباشد.

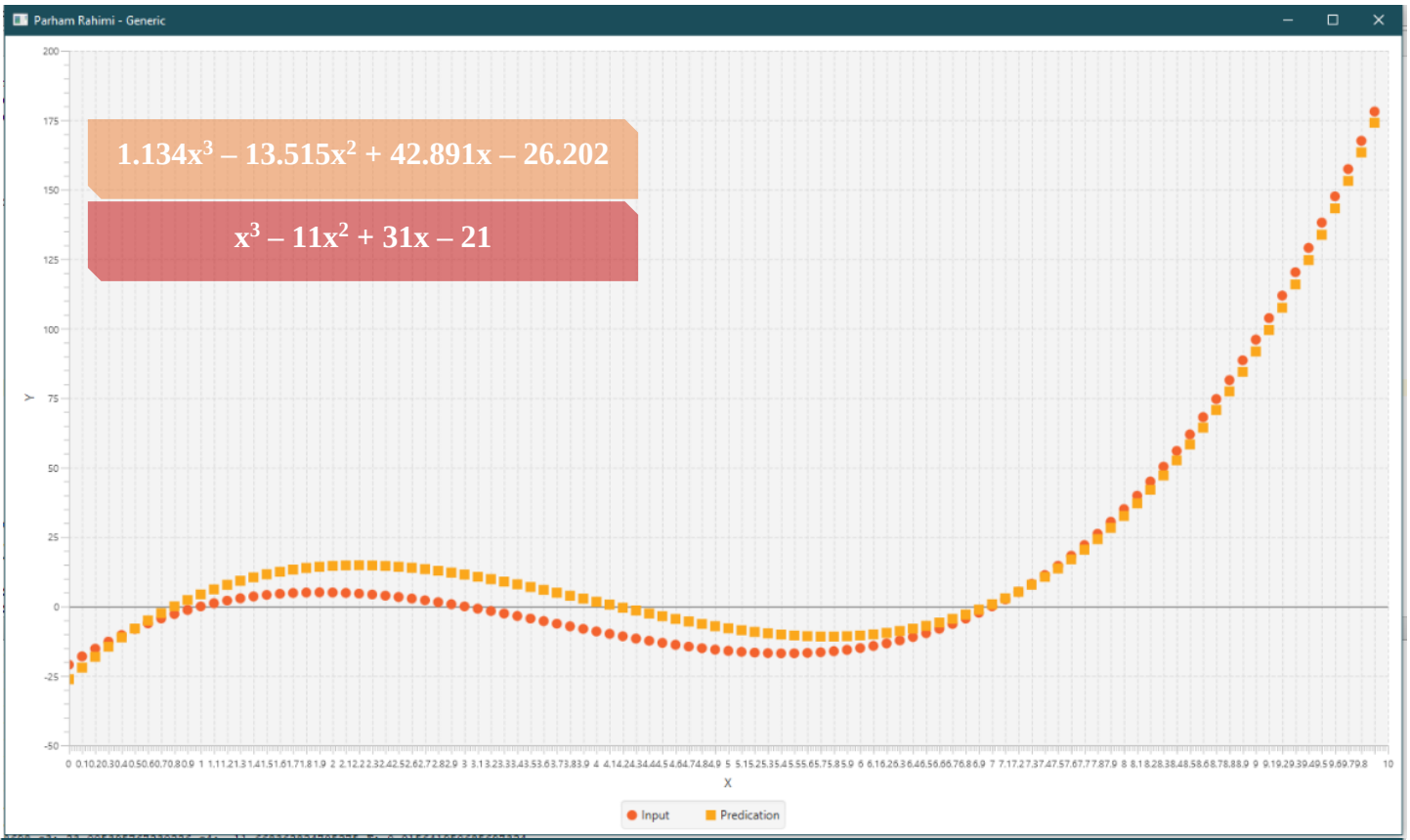
در آزمایش های من روی ورودی $x^3 - 11x^2 + 31x - 21$ و با پارامتر های (تعداد نسل: متغیر | اندازه جمعیت: ۱۰۰۰۰ | اندازه تورنومنت: ۲ | نرخ جهش: ۰,۰۱ | واریانس: ۰,۲) تعداد نسل بهینه ۱۰۰۰ به دست آمد.

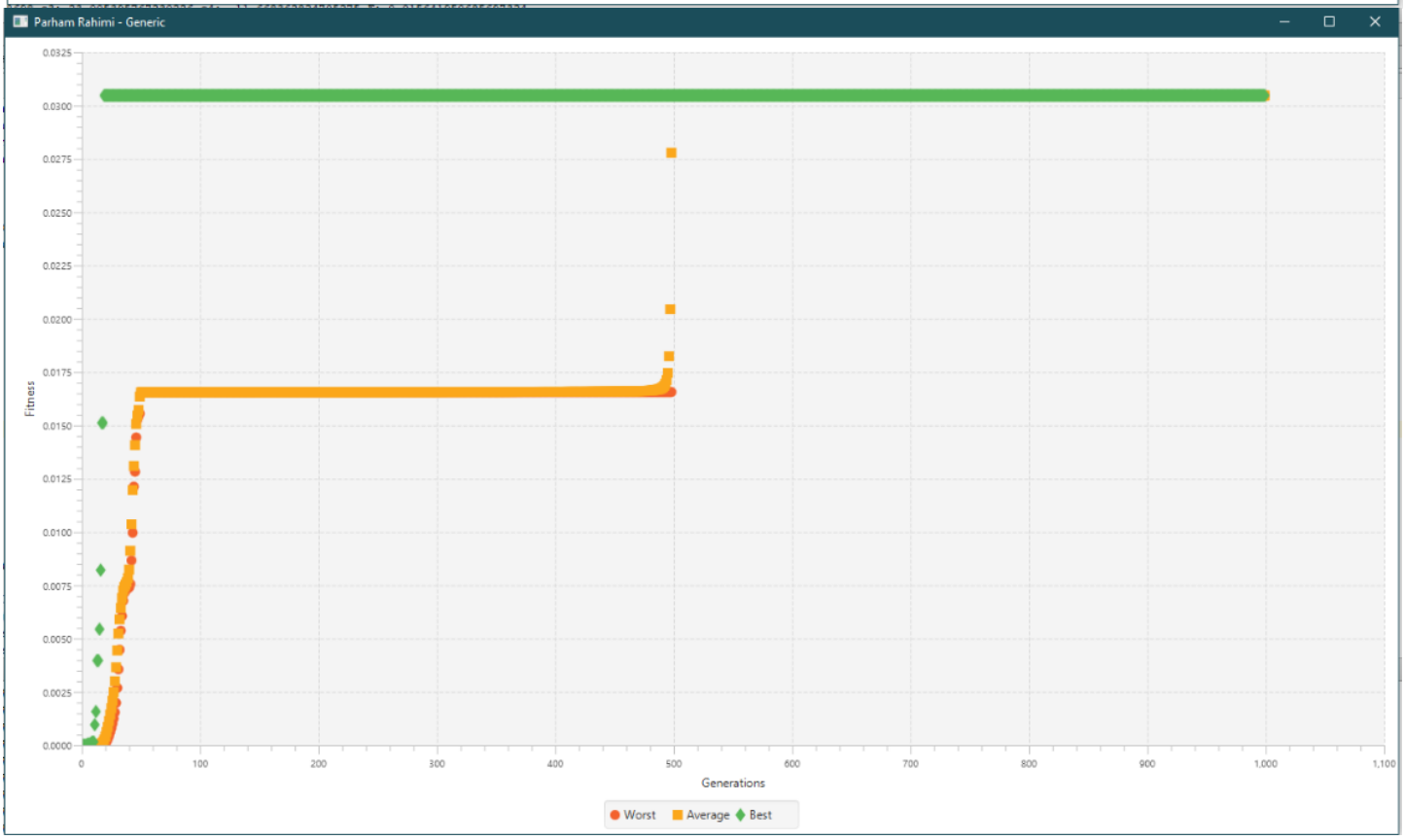
(تعداد نسل: ۵۰۰ | اندازه جمعیت: ۱۰۰۰۰ | اندازه تورنومنت: ۲ | نرخ جهش: ۰,۰۱ | واریانس: ۰,۲)



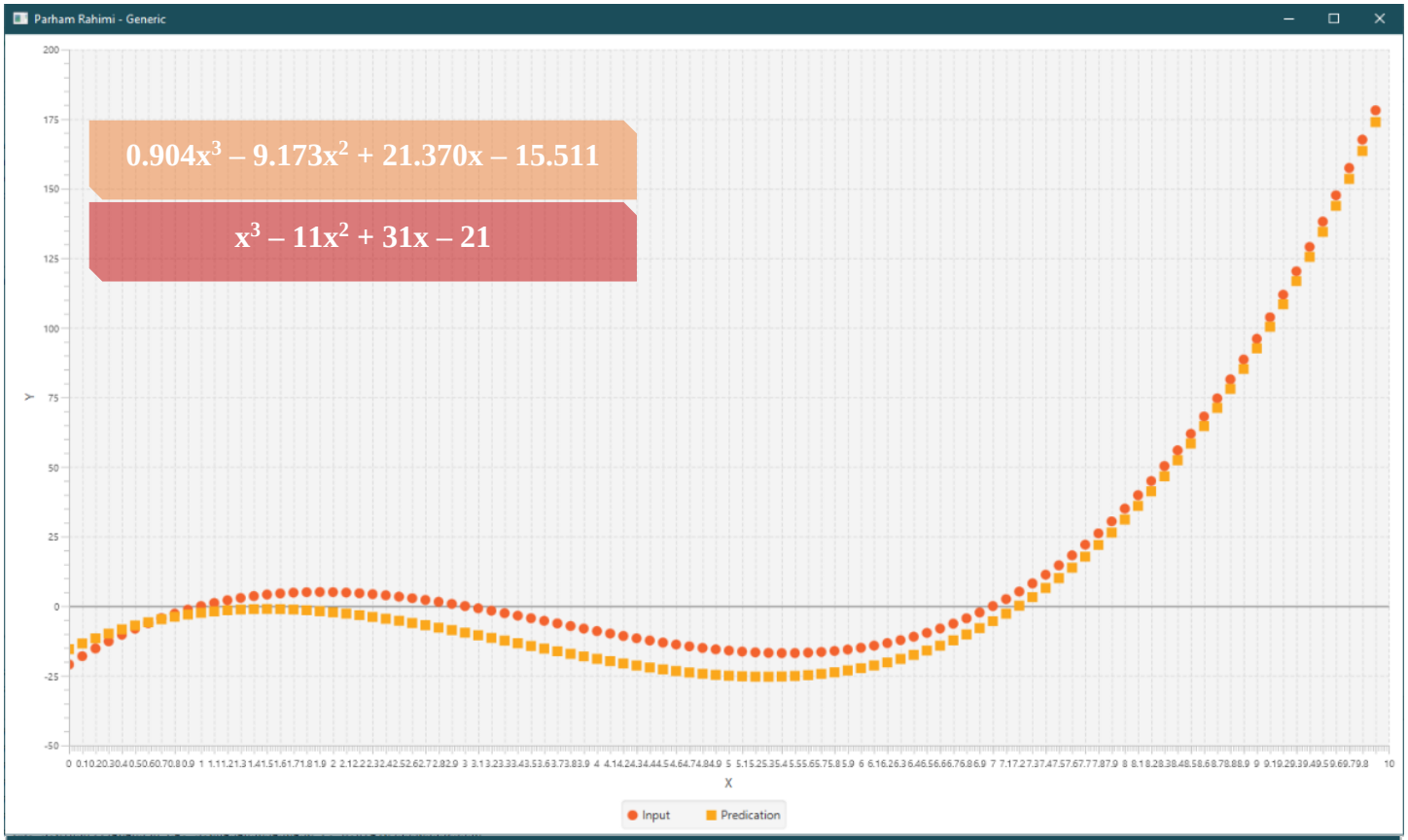


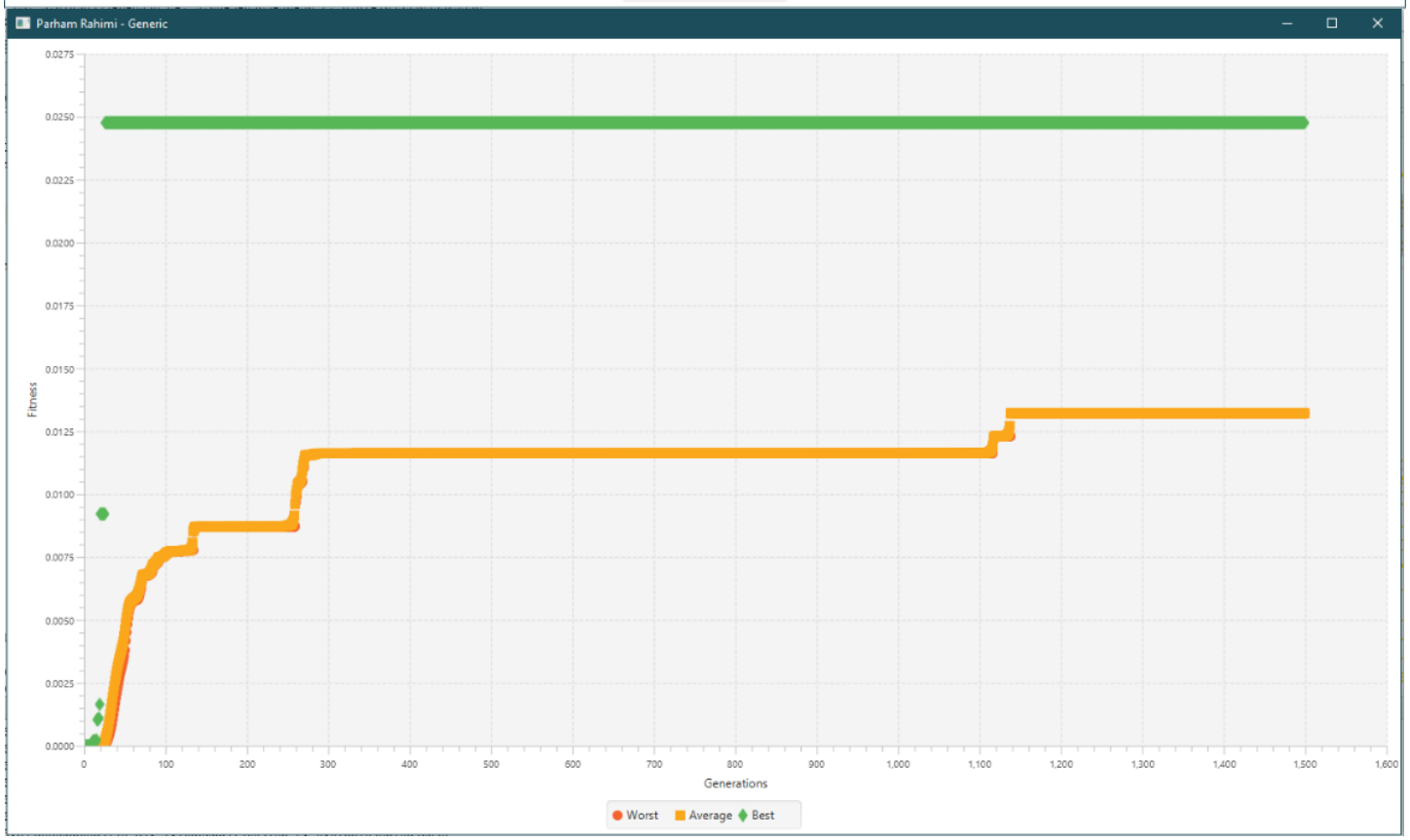
(تعداد نسل: ۱۰۰۰ | اندازه جمعیت: ۱۰۰۰۰ | اندازه تورنومنت: ۲ | نرخ جهش: ۰,۰۱ | واریانس: ۰,۲)





(تعداد نسل: ۱۵۰۰ | اندازه جمعیت: ۱۰۰۰۰ | اندازه تورنومنت: ۲ | نرخ جهش: ۰,۰۱ | واریانس: ۰,۲)



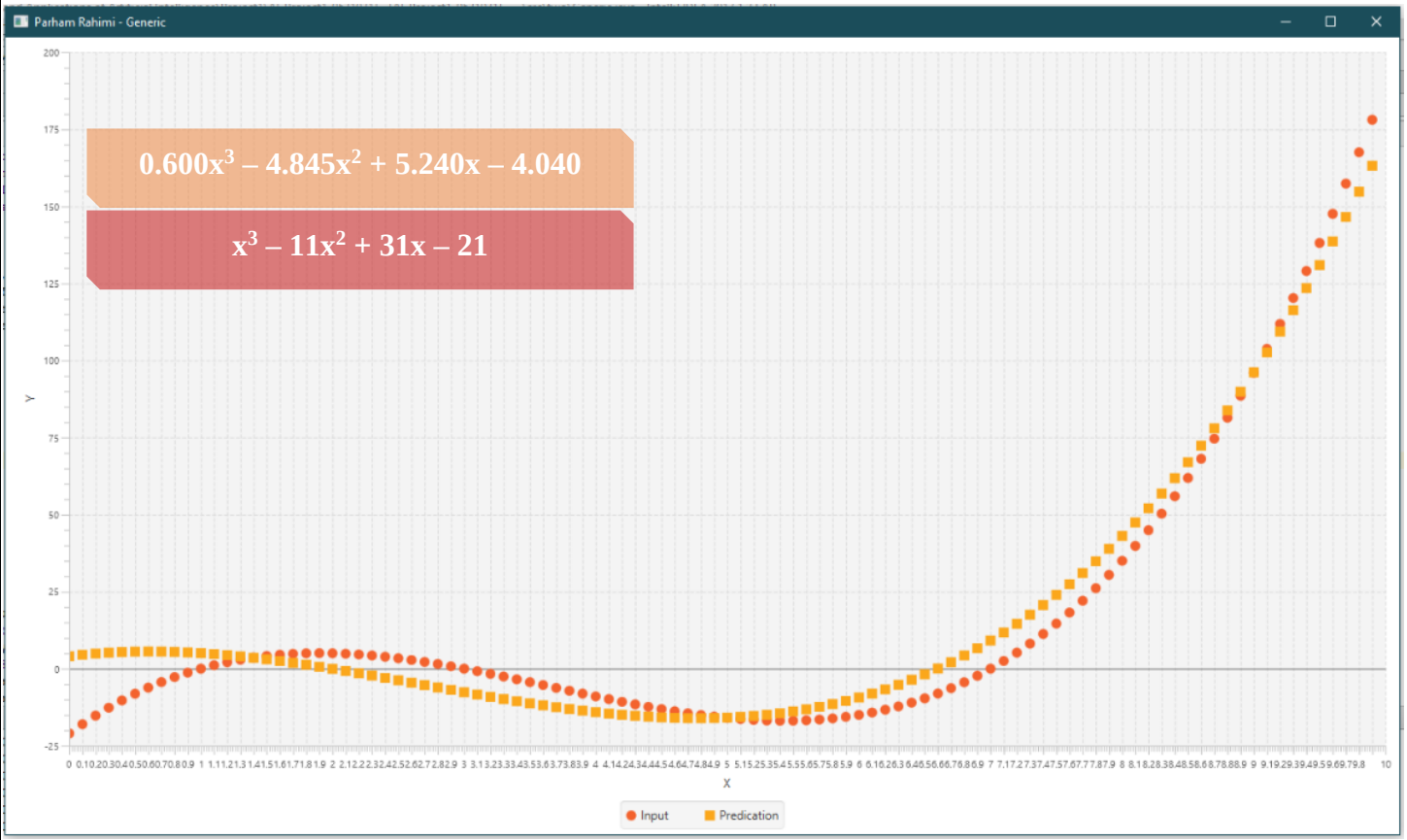


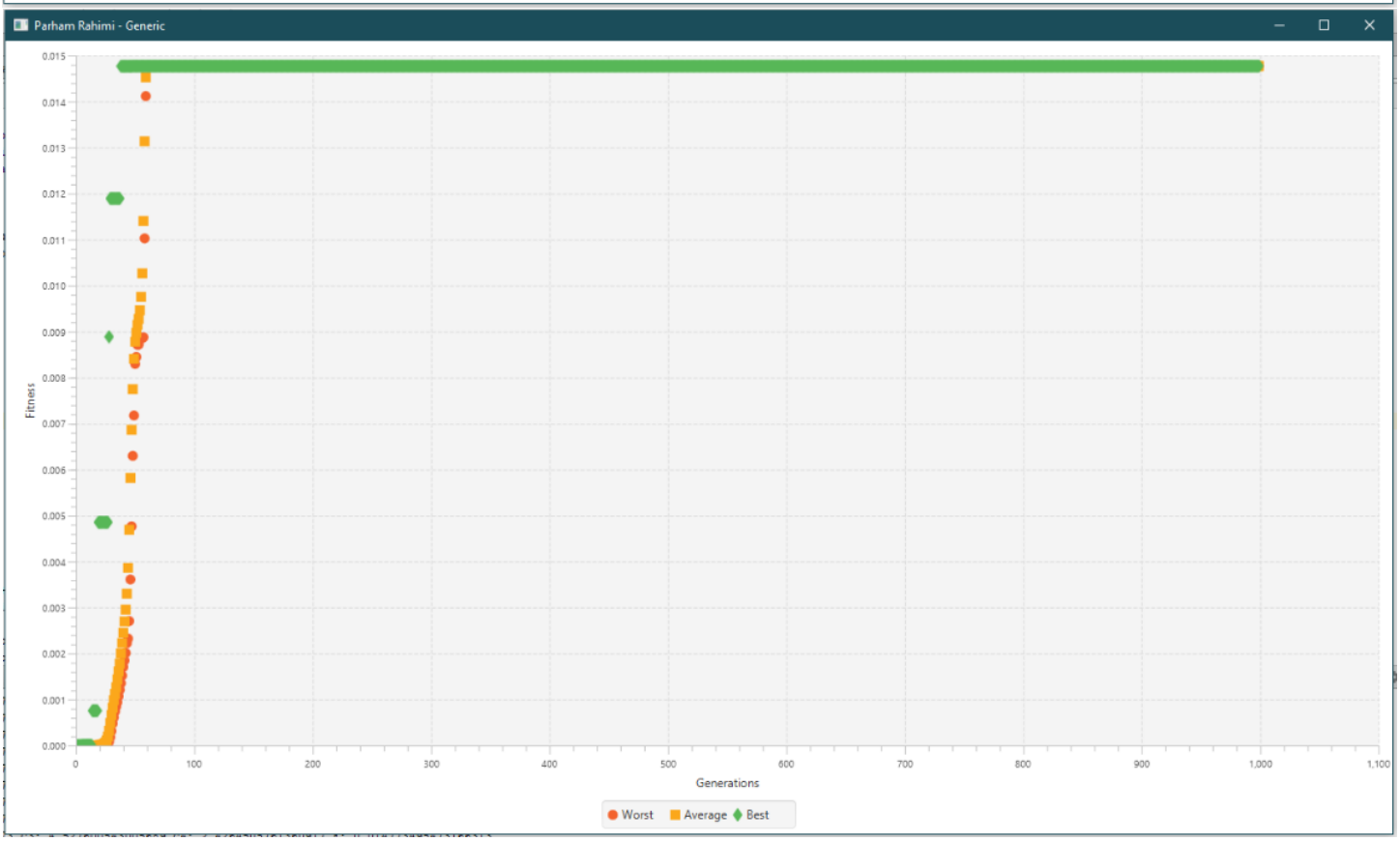
تاثیر اندازه جمعیت (populationSize):

انتظار من از تاثیر اندازه جمعیت این بود که با افزایش آن در تعداد نسل ثابت به شایستگی بهتری دست یابم که البته در طی آزمایش ها هم به همین نتیجه رسیدم. این رابطه به این علت است که با افزایش اندازه جمعیت تعداد جمعیت اولیه که با ژن های رندوم تولید می شوند بیشتر است در نتیجه احتمال آن که ژنی نزدیک به ضریب مد نظر ما تولید شود بیشتر است در نتیجه سریع تر(در تعداد نسل کمتر) و مطمئن تر به شایستگی مطلوب می رسیم.

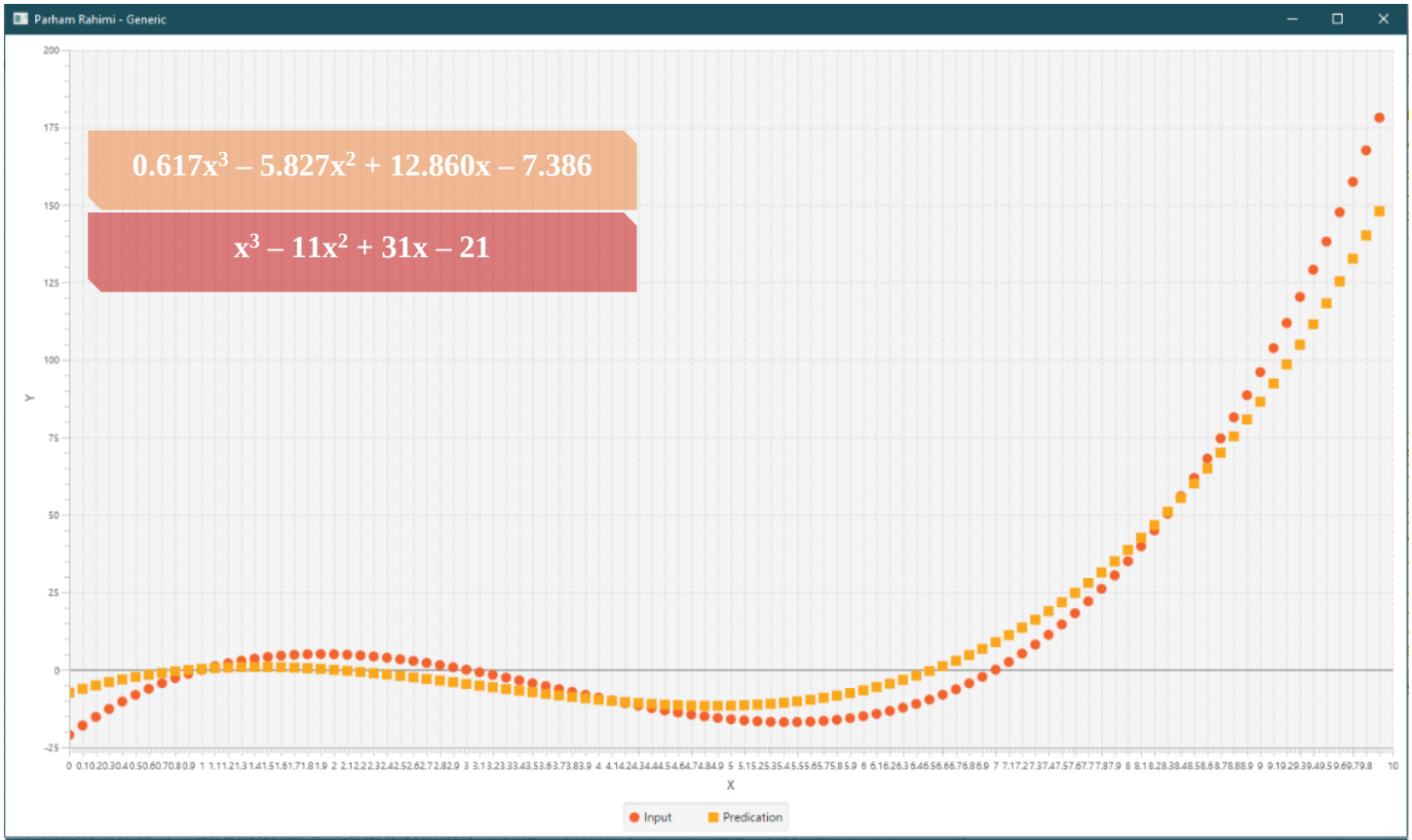
در آزمایش های من روی ورودی $x^3 - 11x^2 + 31x - 21$ و با پارامتر های (تعداد نسل: ۱۰۰۰ | اندازه جمعیت: متغیر | اندازه تورنومنت: ۲ | نرخ جهش: ۰,۰۲ | واریانس: ۰,۲) تعداد نسل بهینه بیشترین تعداد تست شده یعنی ۵۰۰۰۰ به دست آمد.

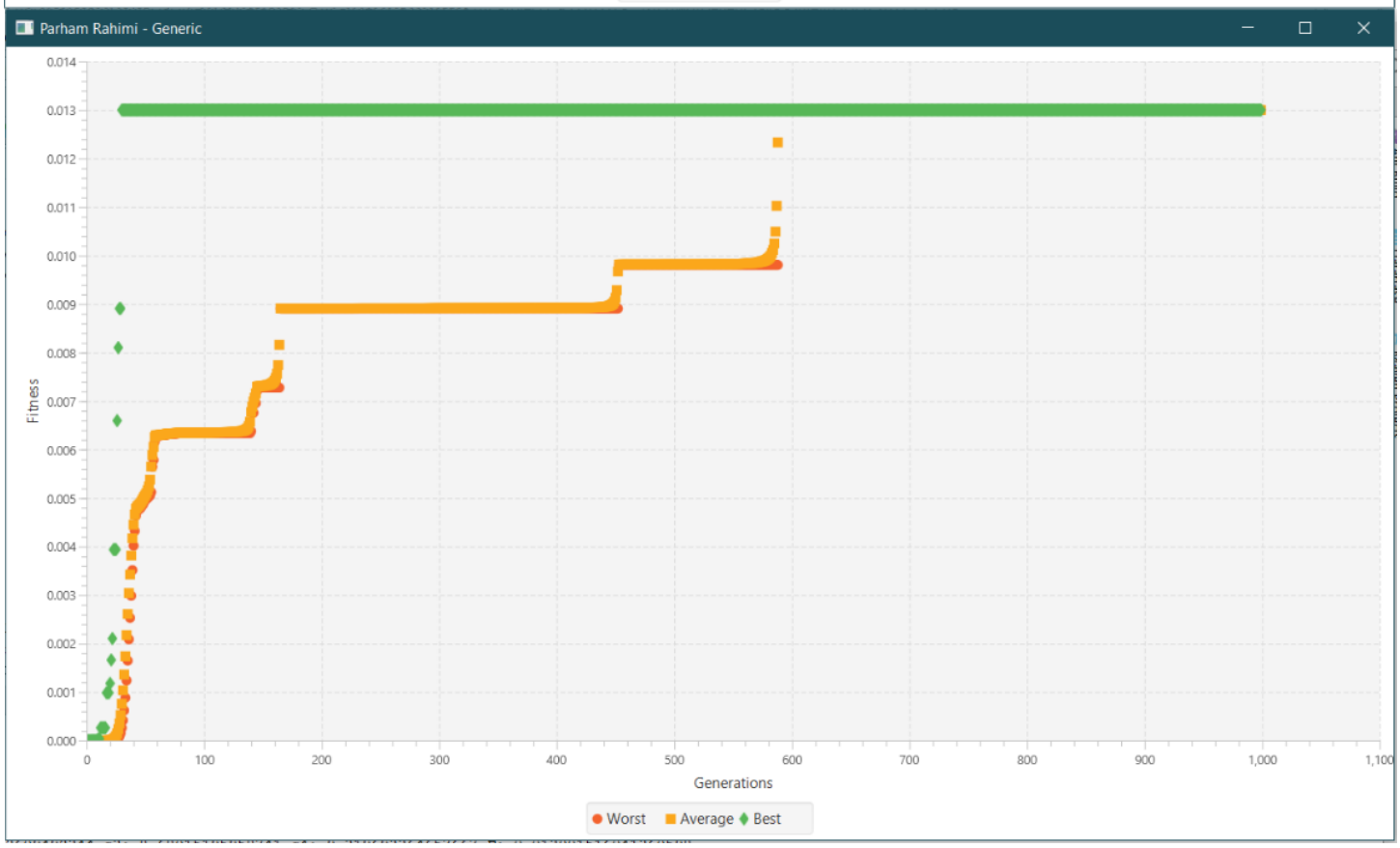
(تعداد نسل: ۱۰۰۰ | اندازه جمعیت: ۲۰۰۰ | اندازه تورنومنت: ۲ | نرخ جهش: ۰,۰۱ | واریانس: ۰,۲)



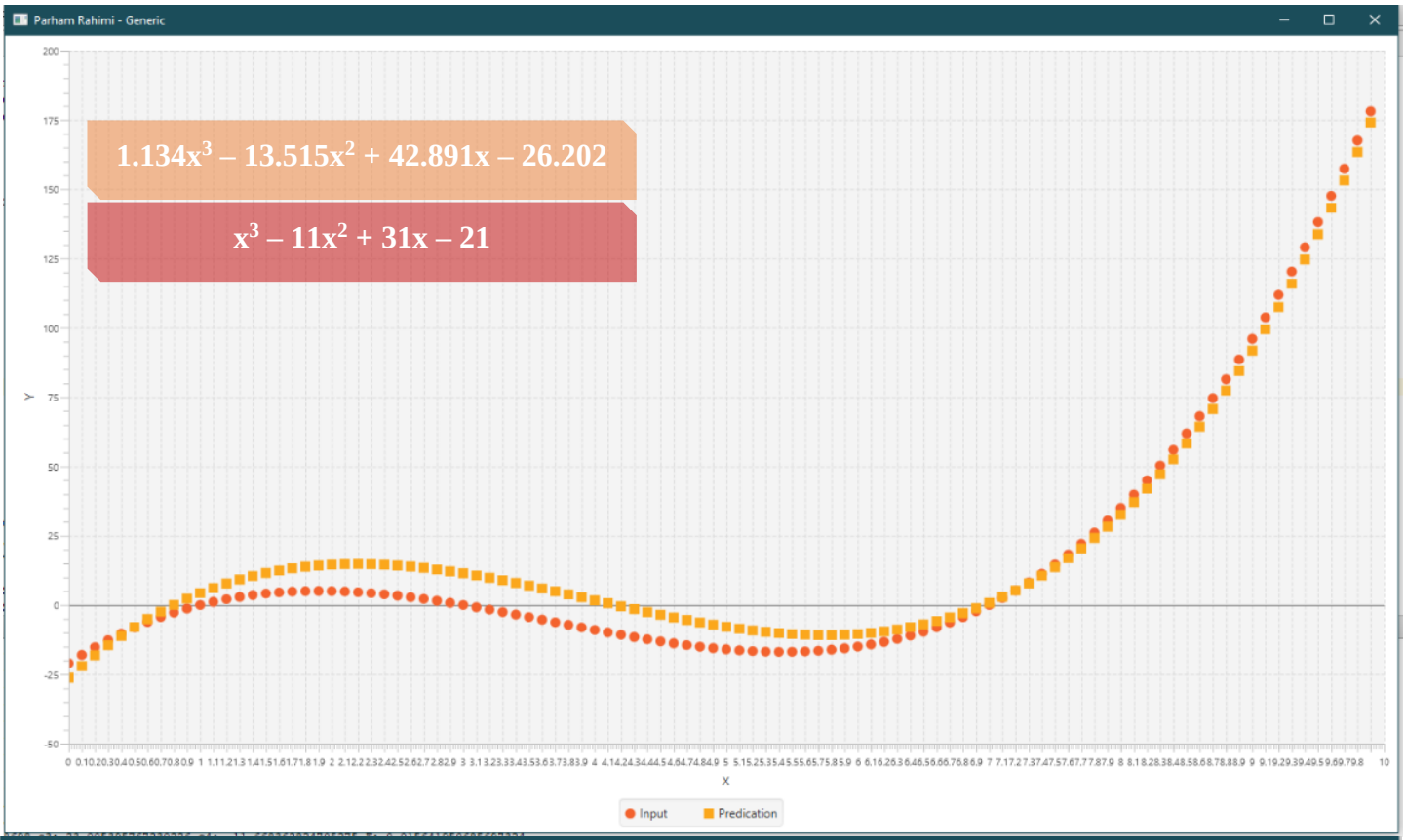


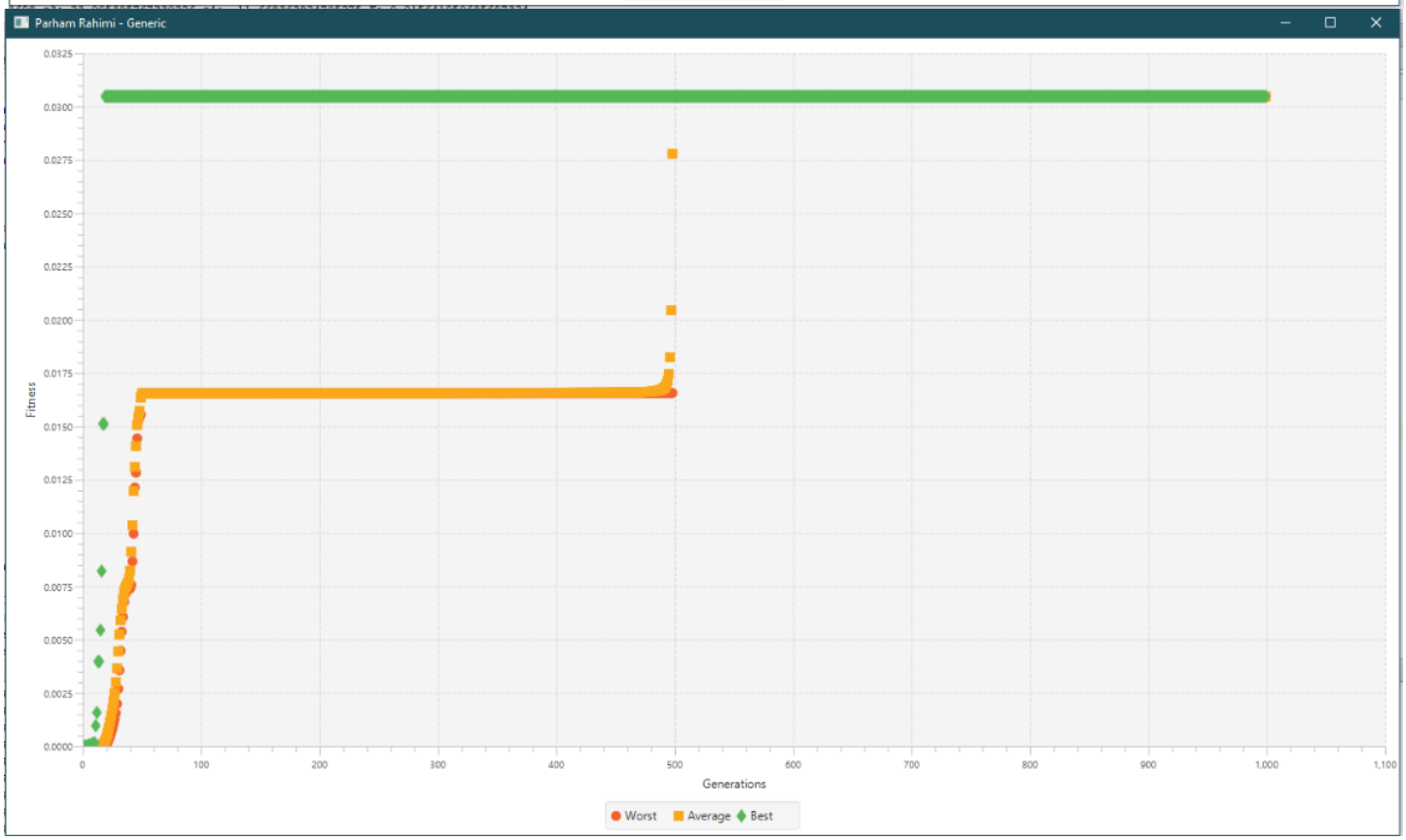
(تعداد نسل: ۱۰۰۰ | اندازه جمعیت: ۵۰۰۰ | اندازه تورنومنت: ۲ | نرخ جهش: ۰,۰۱ | واریانس: ۰,۲)



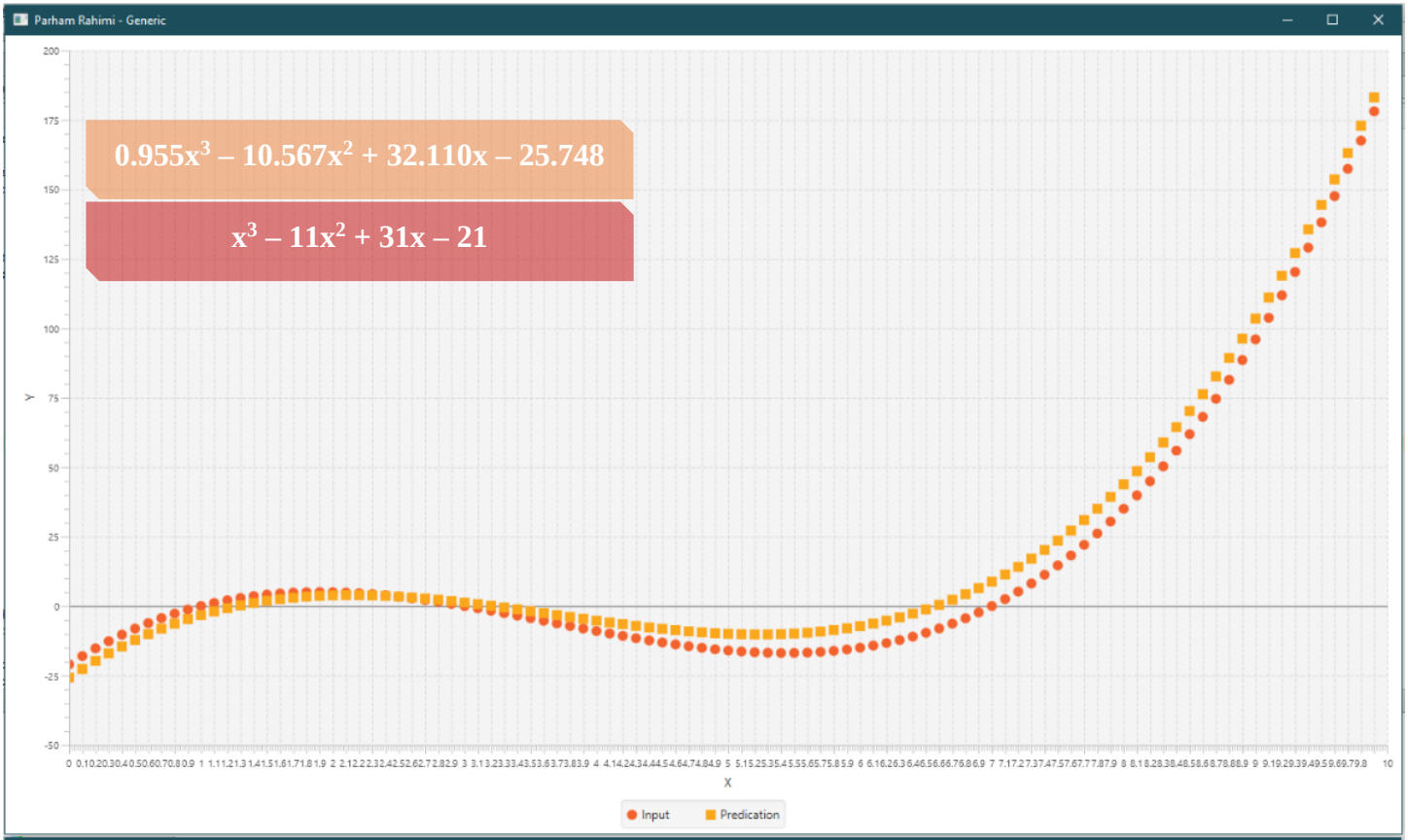


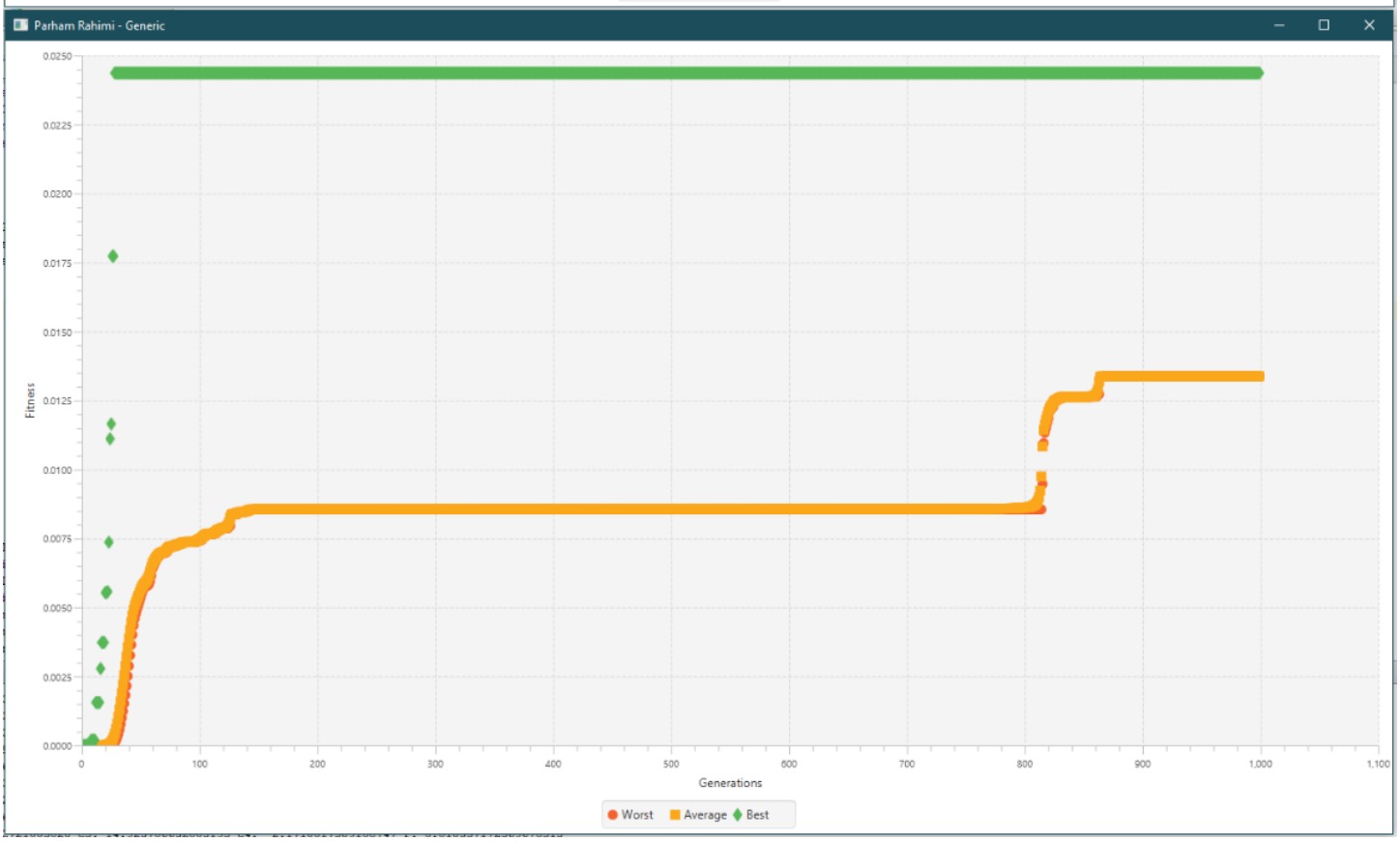
(تعداد نسل: ۱۰۰۰ | اندازه جمعیت: ۱۰۰۰۰ | اندازه تورنومنت: ۲ | نرخ جهش: ۰,۰۱ | واریانس: ۰,۲)





(تعداد نسل: ۱۰۰۰ | اندازه جمعیت: ۵۰۰۰۰ | اندازه تورنومنت: ۲ | نرخ جهش: ۰,۰۱ | واریانس: ۰,۲)



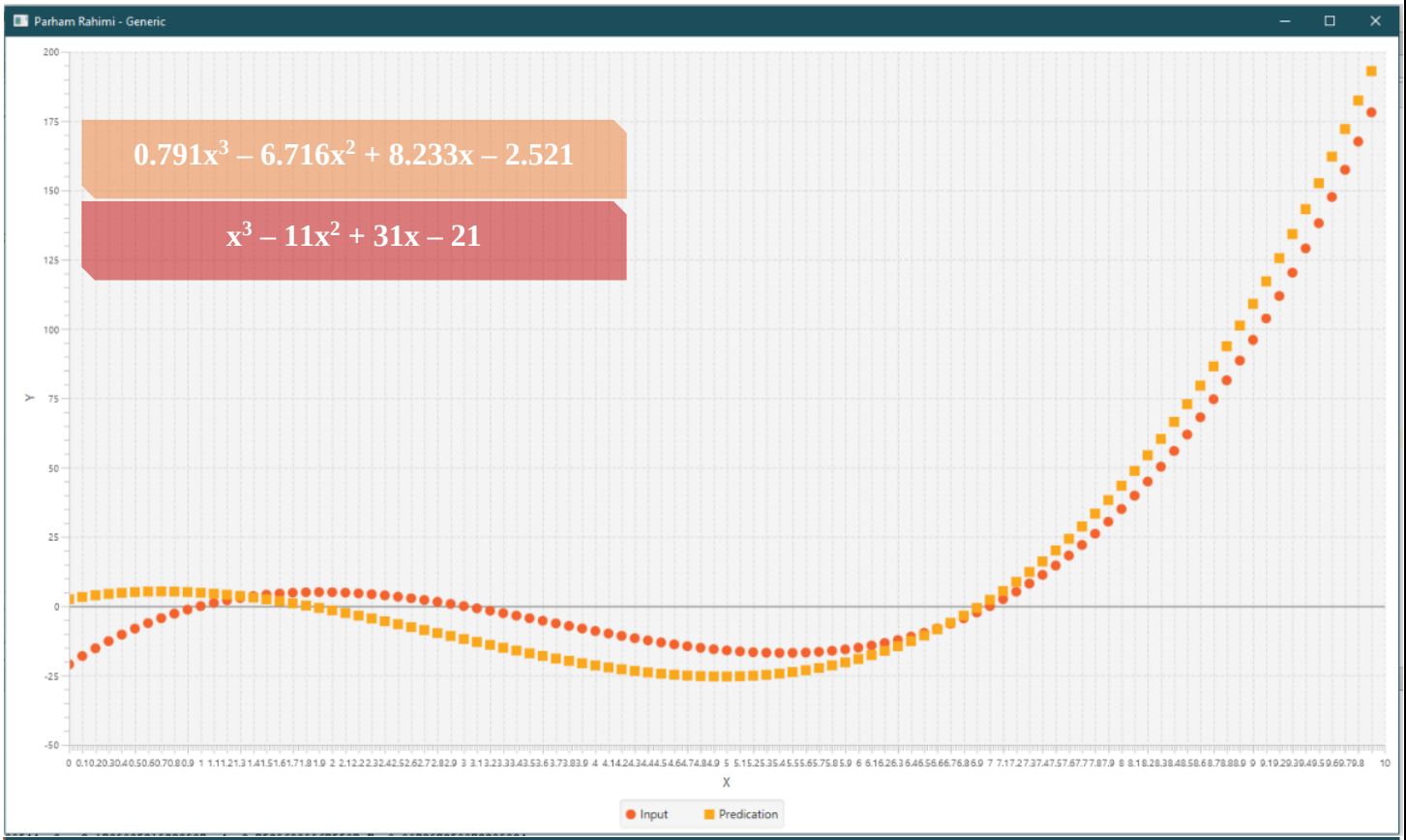


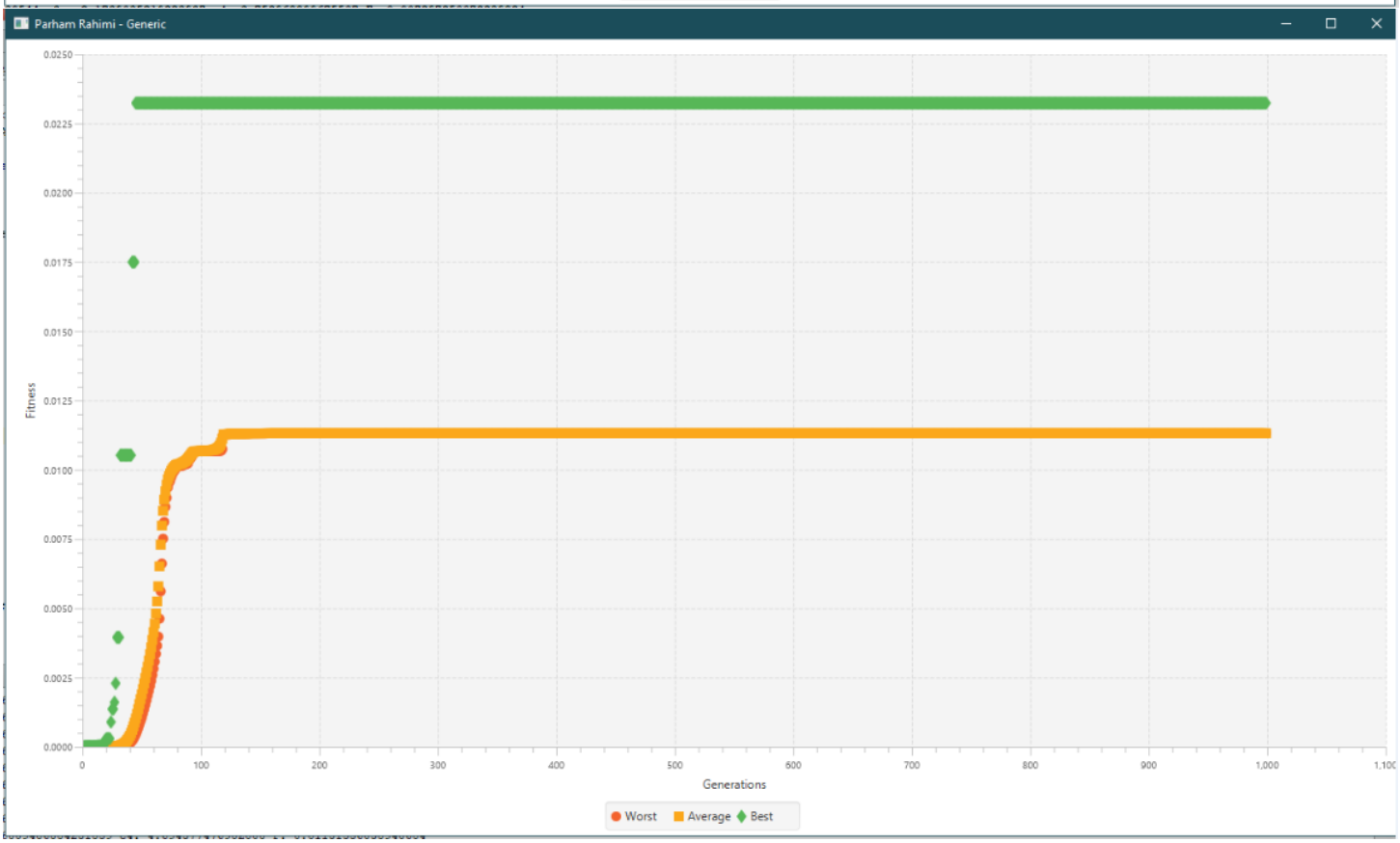
تاثیر اندازه تورنومنت (tournamentSize):

با توجه به آزمایش ها به نظر می آید که با افزایش اندازه تورنومنت از آنجا که اهمیت شایستگی را بیشتر از نگه داشتن تنوع می کنیم، سریعتر همگرا می شویم و سریعتر به شایستگی بالاتر دست می یابیم هرچند ممکن است به علت عدم تنوع زودرس به همگرایی زودرس برسیم و جواب نهاییمان شایستگی لازم را نداشته باشد.

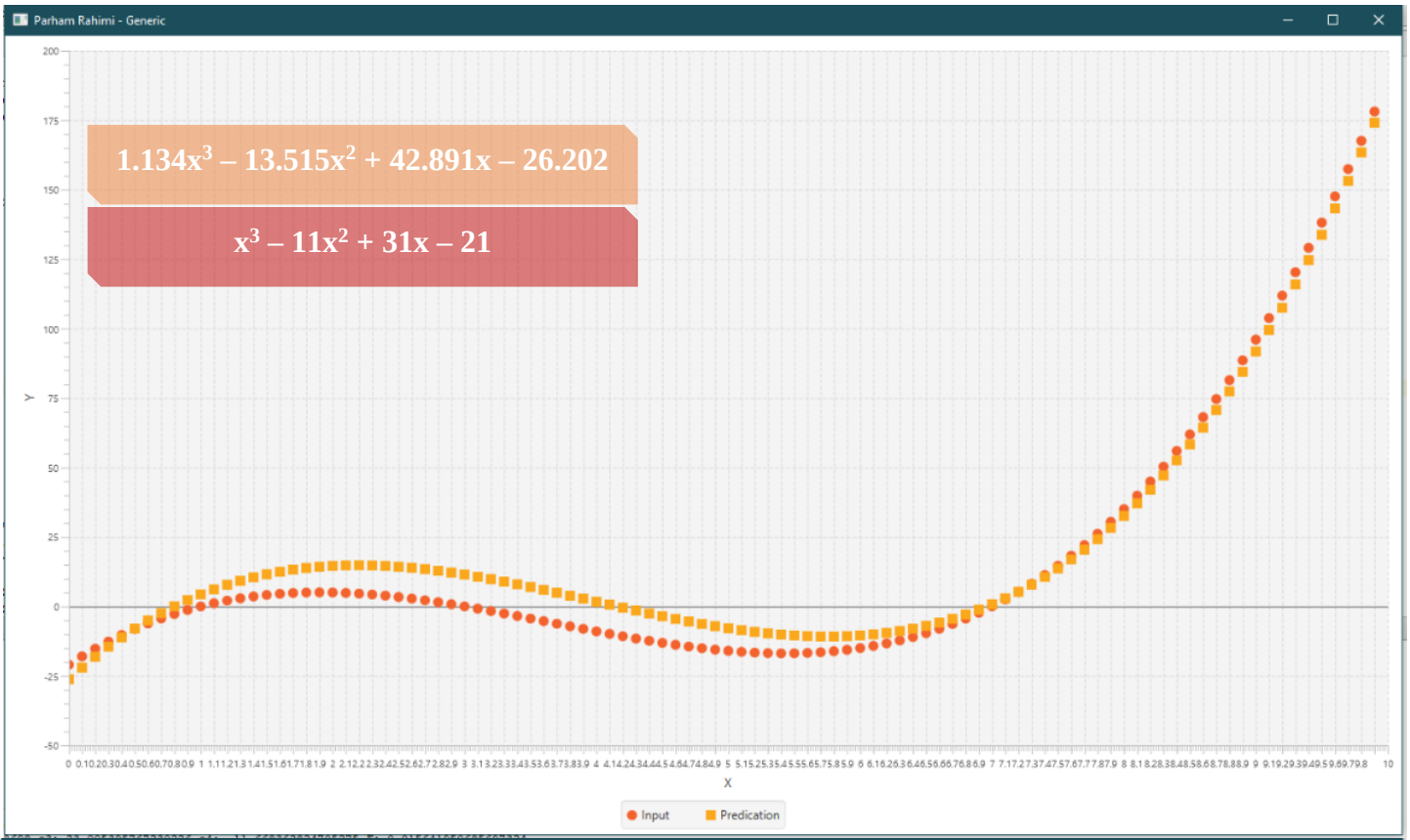
در آزمایش های من روی ورودی $x^3 - 11x^2 + 31x - 21$ و با پارامتر های (تعداد نسل: ۱۰۰۰ | اندازه جمعیت: ۱۰۰۰۰ | اندازه تورنومنت: متغیر | نرخ جهش: ۰,۰۲ | واریانس: ۰,۲)، بهترین اندازه تورنومنت ۴ به دست آمد.

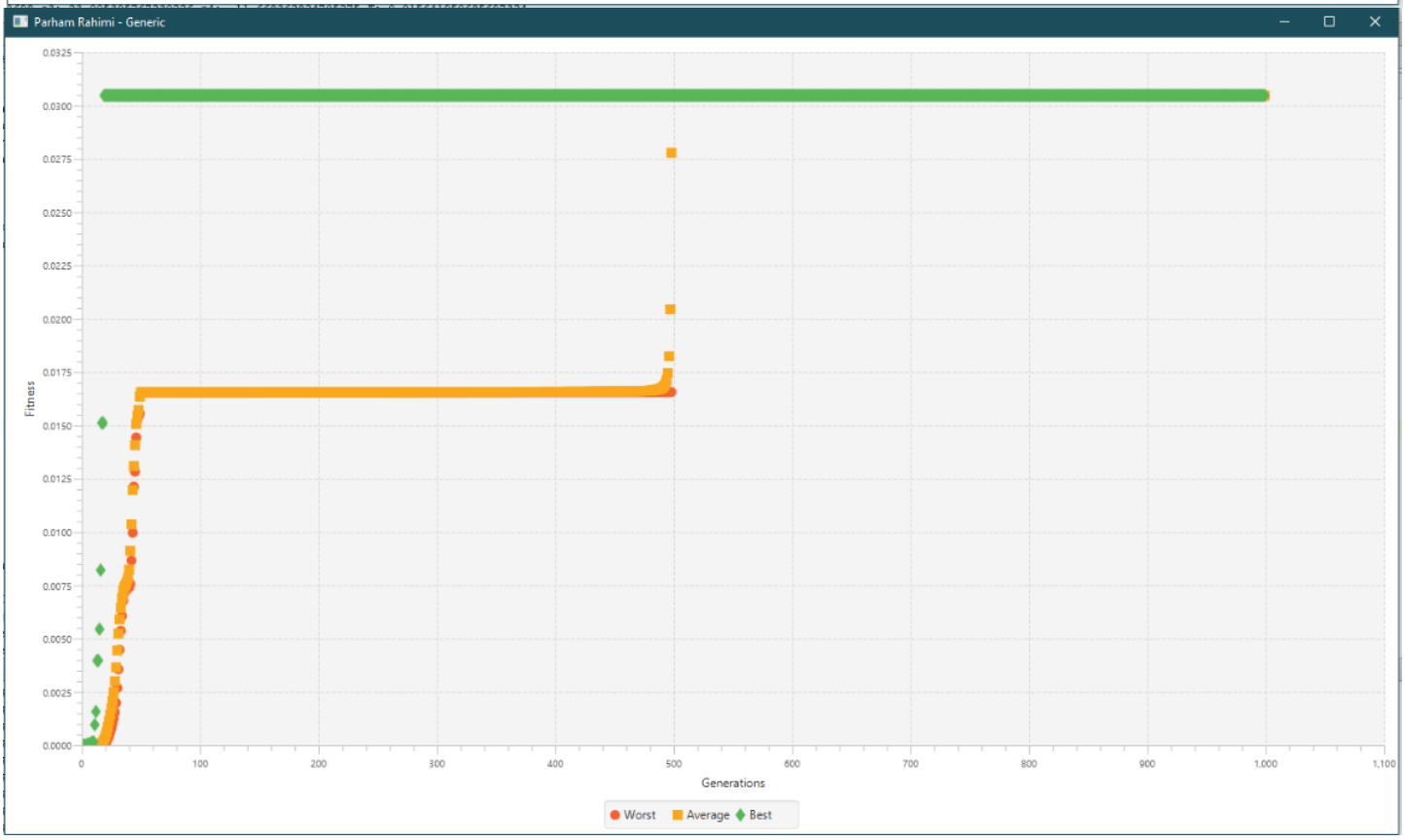
(تعداد نسل: ۱۰۰۰ | اندازه جمعیت: ۱۰۰۰۰ | اندازه تورنومنت: ۱ | نرخ جهش: ۰,۰۱ | واریانس: ۰,۲)



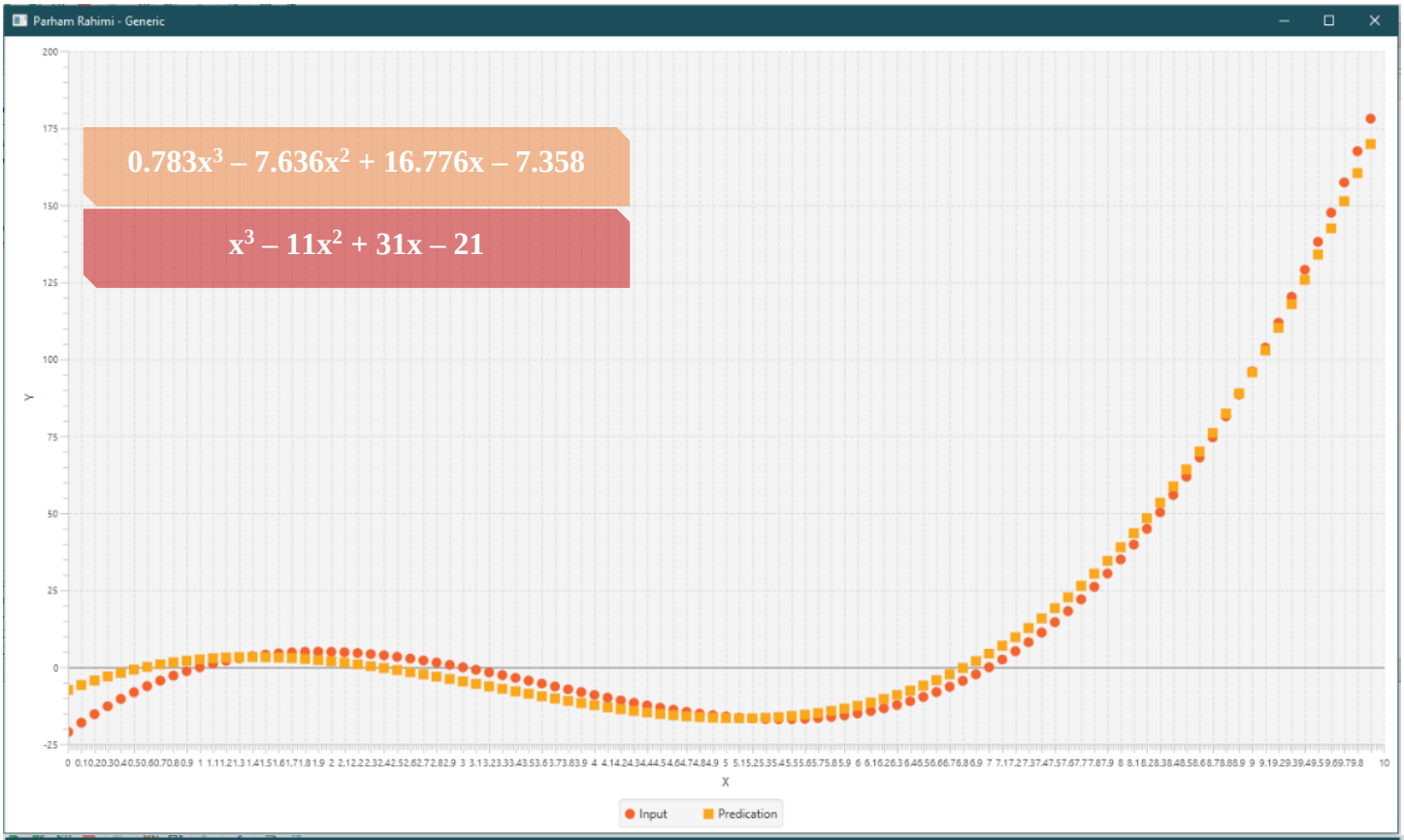


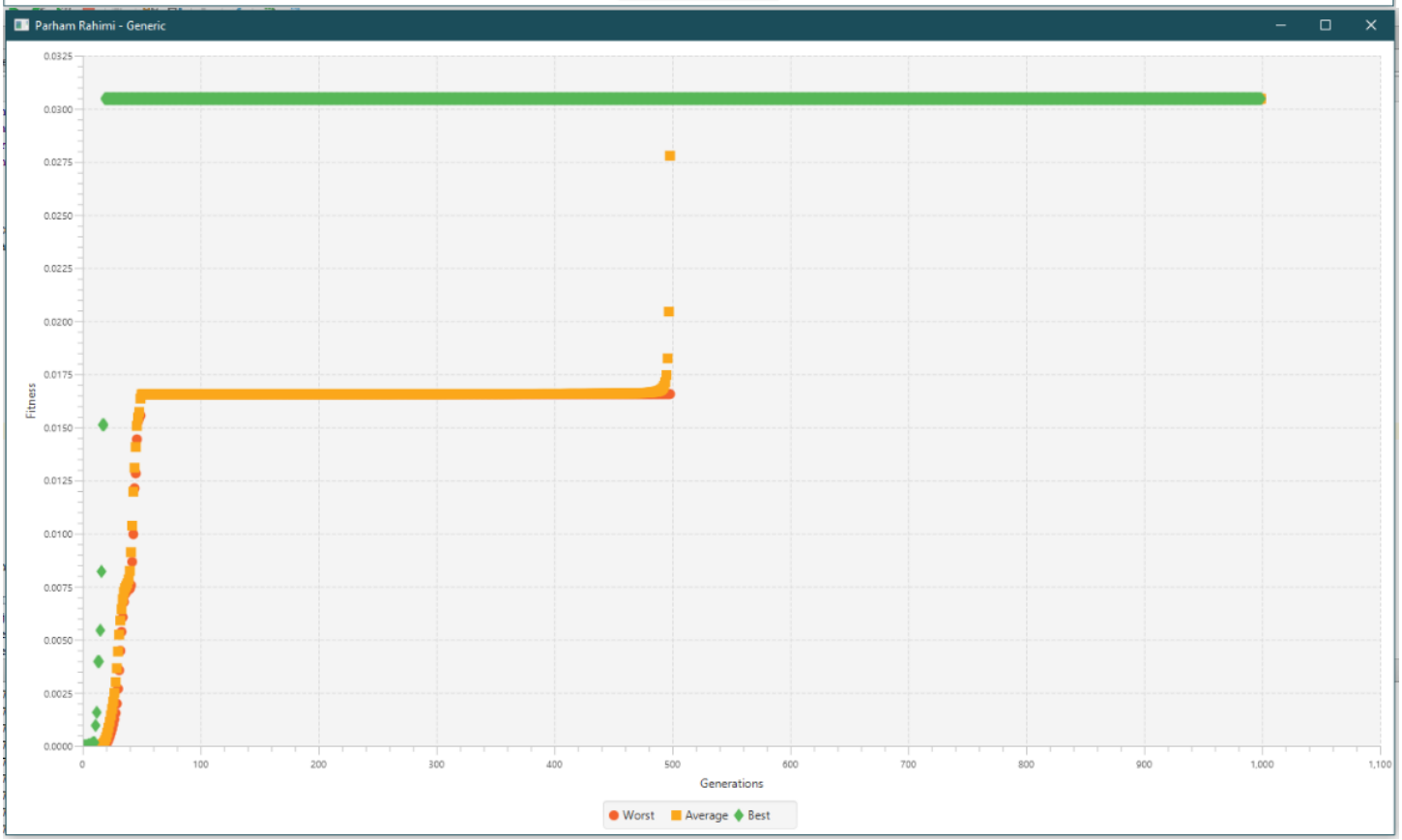
(تعداد نسل: ۱۰۰۰ | اندازه جمعیت: ۱۰۰۰۰ | اندازه تورنومنت: ۲ | نرخ جهش: ۰,۰۱ | واریانس: ۰,۲)





(تعداد نسل: ۱۰۰۰ | اندازه جمعیت: ۱۰۰۰۰ | اندازه تورنومنت: ۴ | نرخ جهش: ۰,۰۱ | واریانس: ۰,۲)



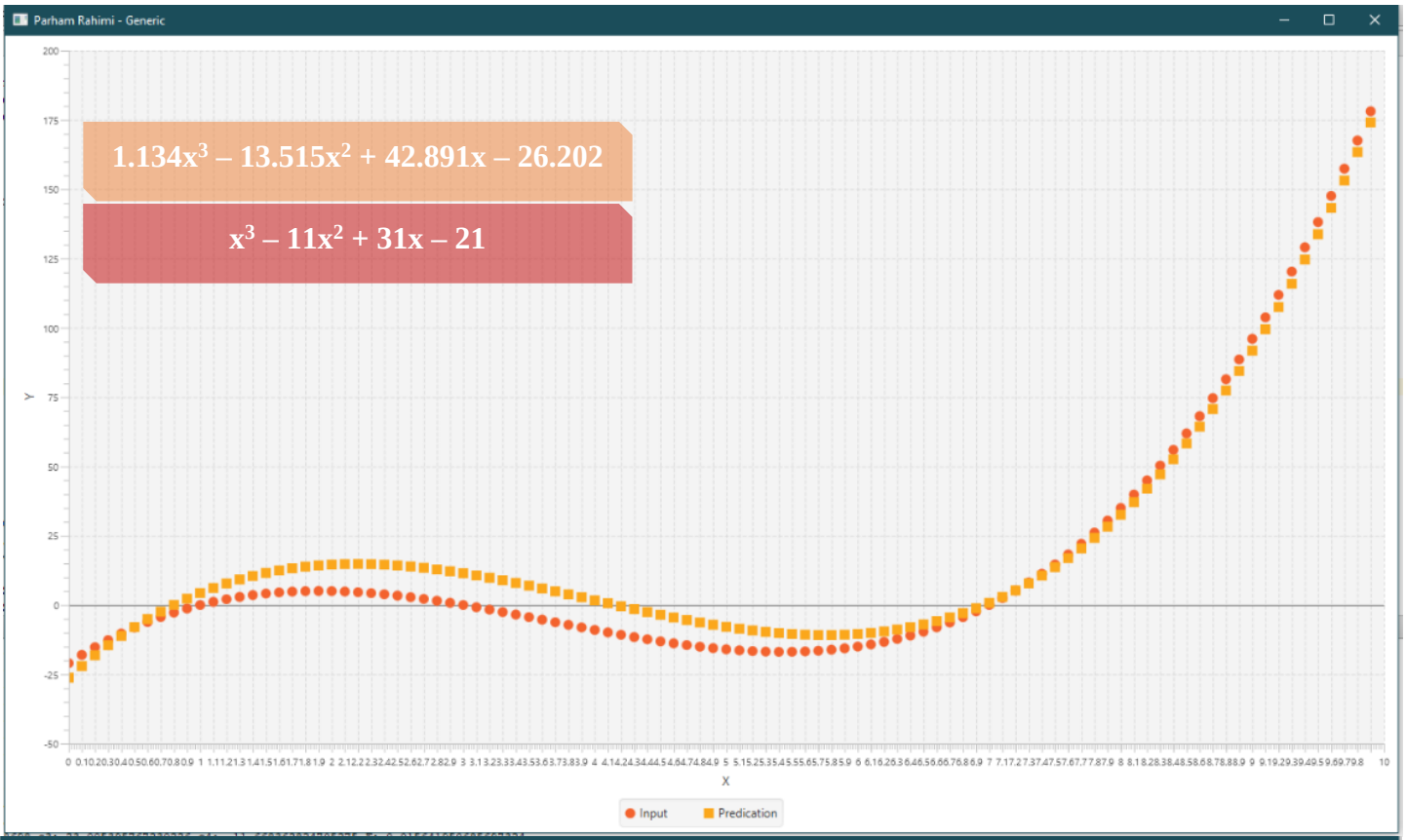


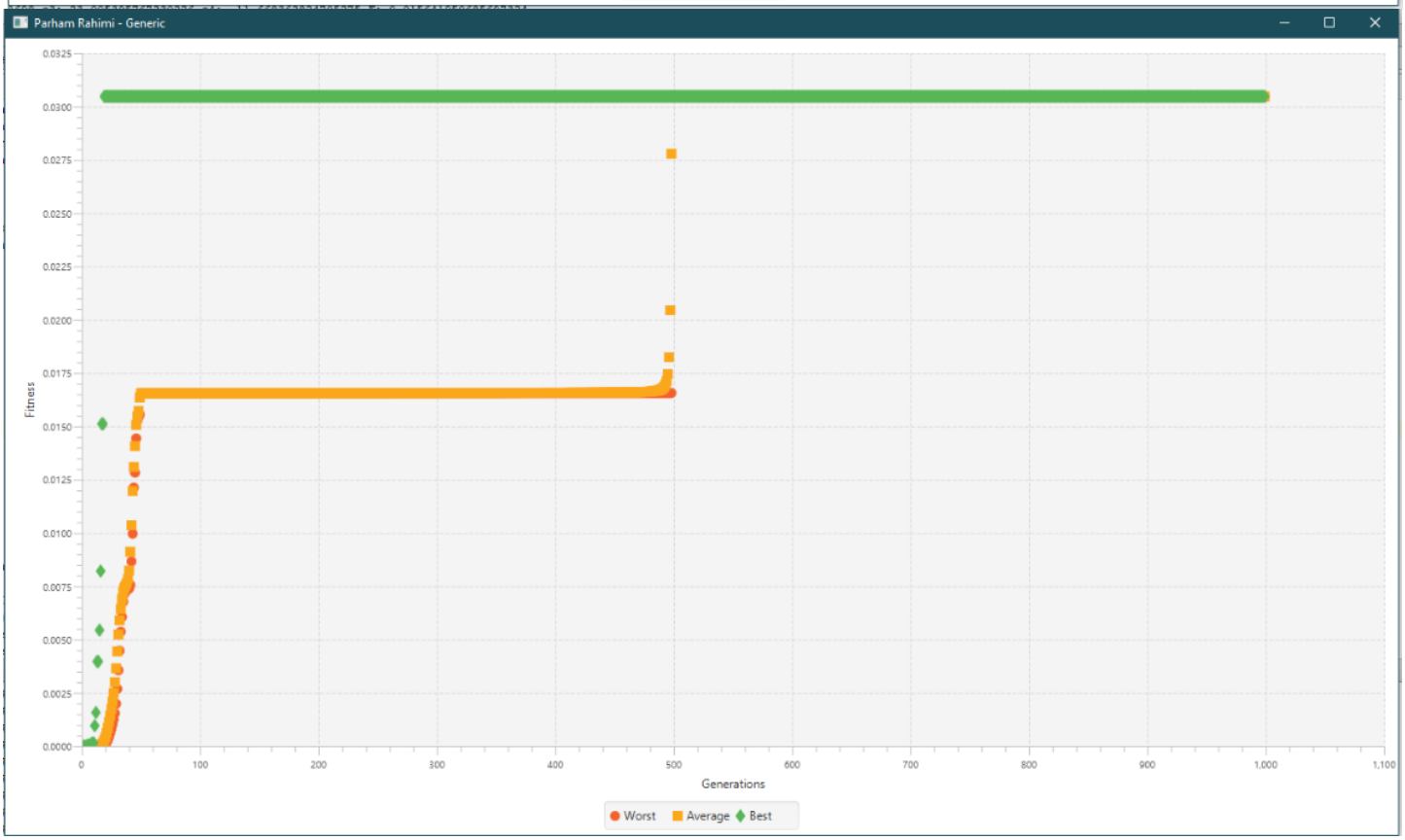
تاثیر نرخ جهش (mutationRate):

با توجه به آزمایش ها به نظر می آید که با افزایش نرخ جهش تاثیر خطی ای بر شایستگی جواب به دست آمده ندارد. اما بدیهیست که موجب افزایش تنوع و در نتیجه دیرتر همگرا شدن می شود.

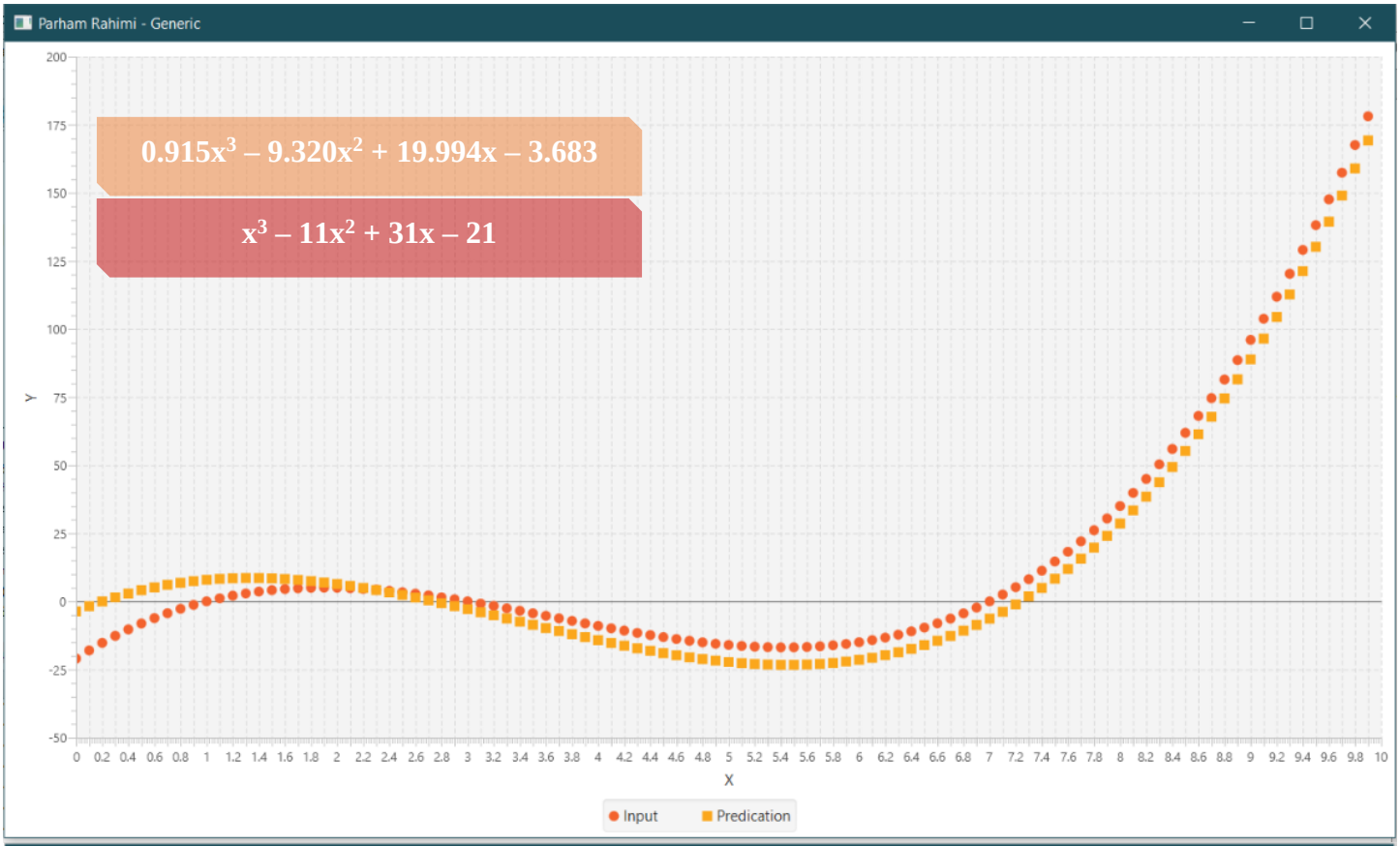
در آزمایش های من روی ورودی $x^3 - 11x^2 + 31x - 21$ و با پارامتر های (تعداد نسل: ۱۰۰۰ | اندازه جمعیت: ۱۰۰۰۰ | اندازه تورنومنت: ۲ | نرخ جهش: متغیر | واریانس: ۰,۲)، بهترین نرخ جهش ۰,۰۲ به دست آمد.

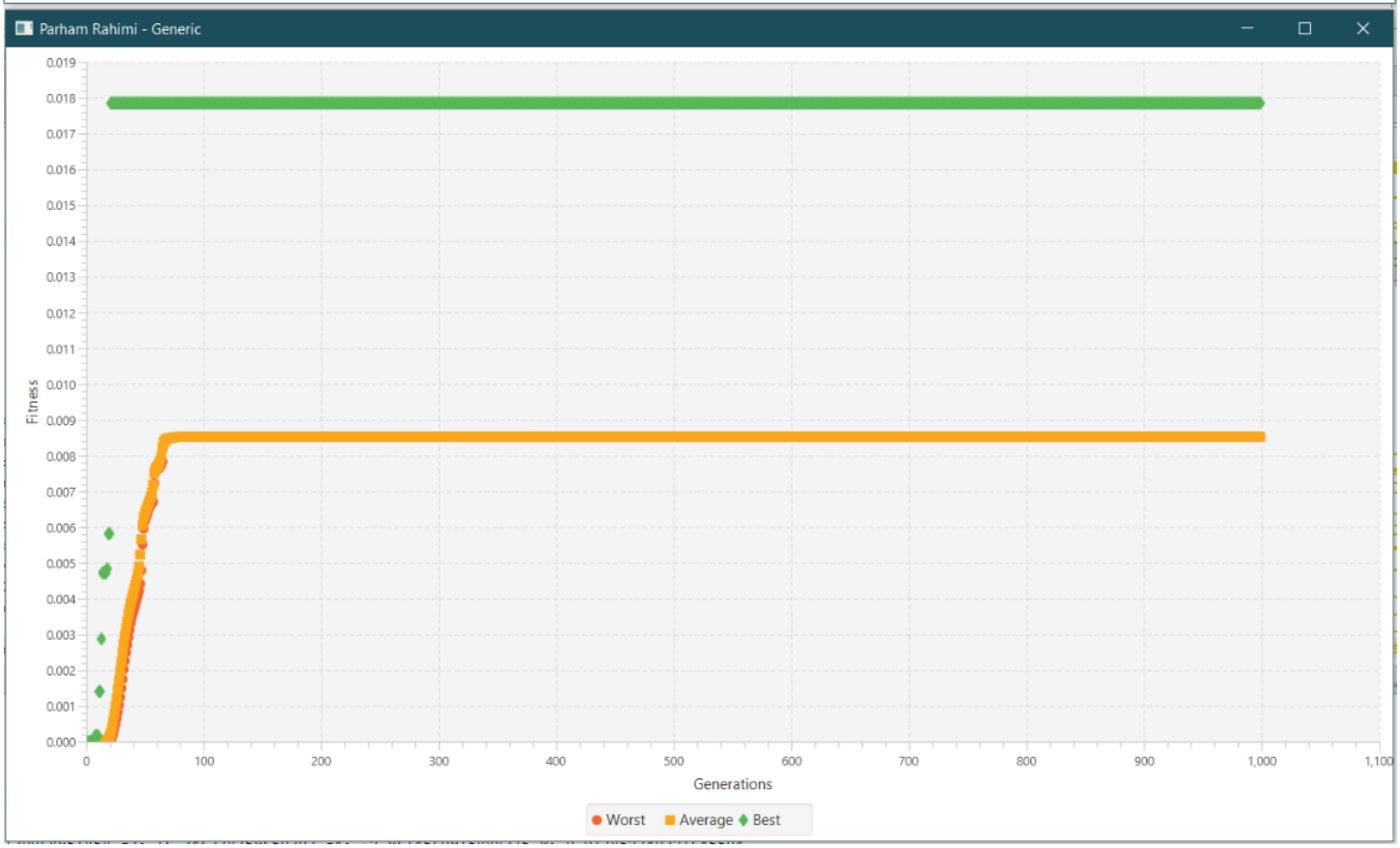
(تعداد نسل: ۱۰۰۰ | اندازه جمعیت: ۱۰۰۰۰ | اندازه تورنومنت: ۲ | نرخ جهش: ۰,۰۱ | واریانس: ۰,۲)



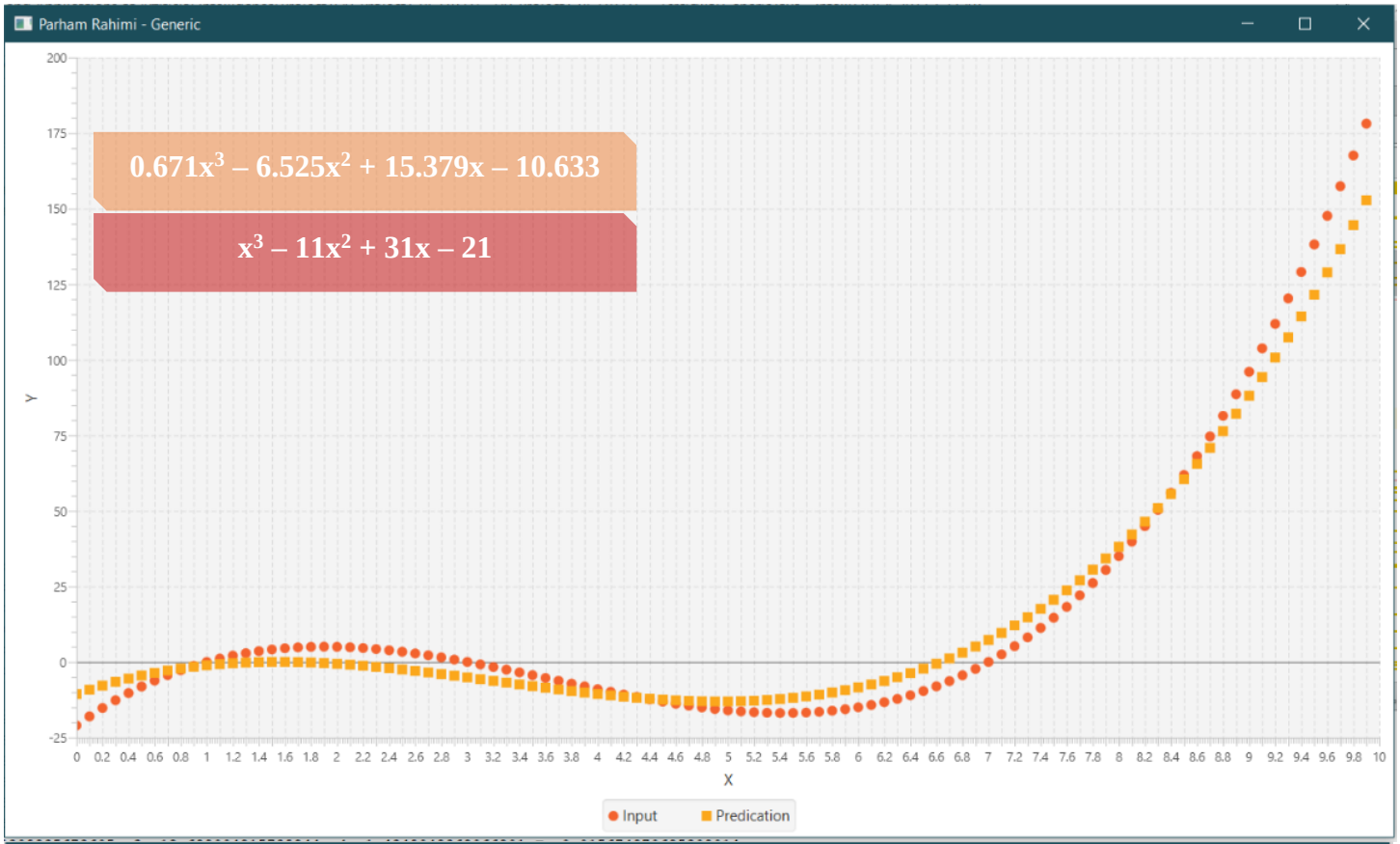


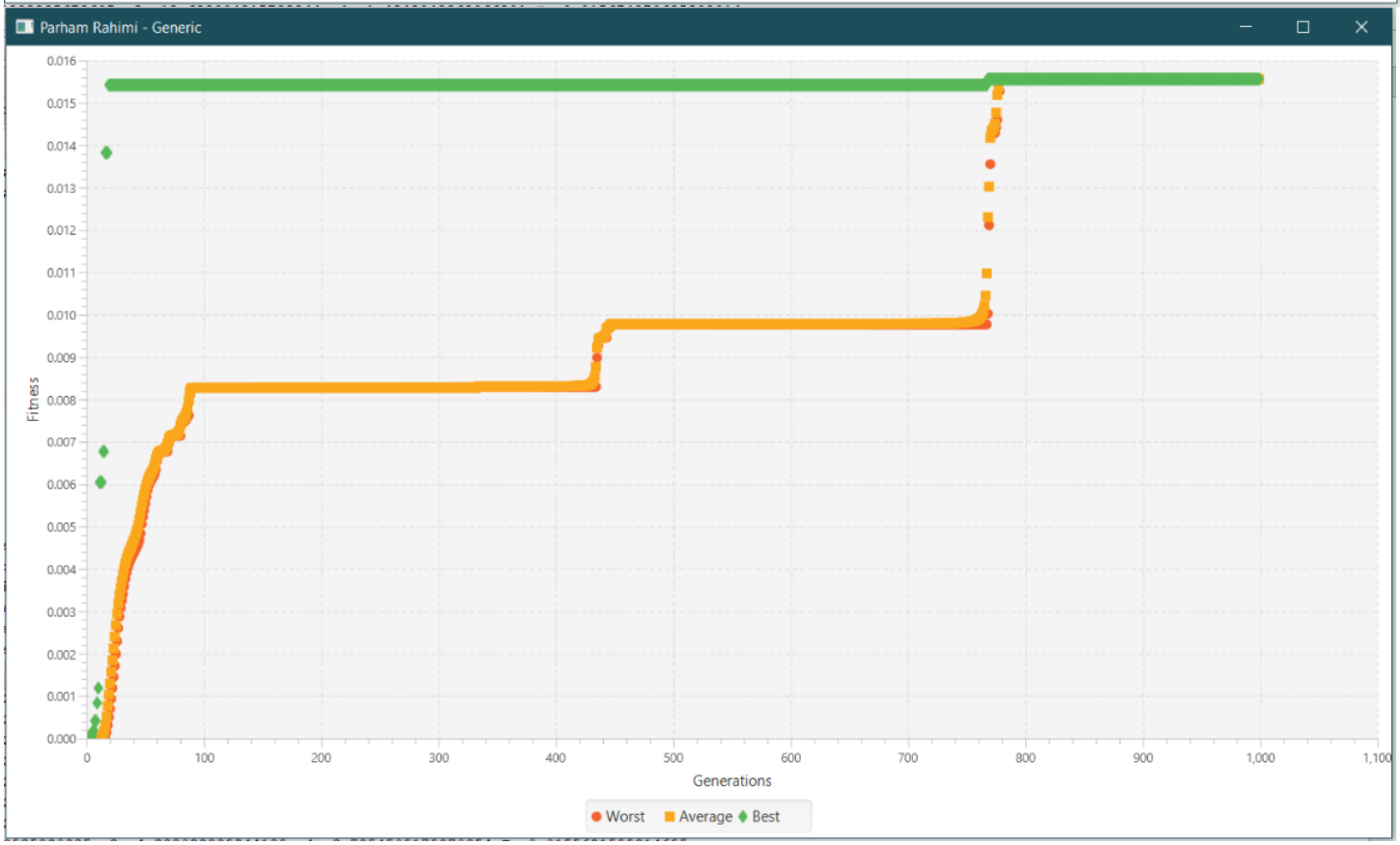
(تعداد نسل: ۱۰۰۰ | اندازه جمعیت: ۱۰۰۰۰ | اندازه تورنومنت: ۲ | نرخ جهش: ۰,۰۲ | واریانس: ۰,۲)



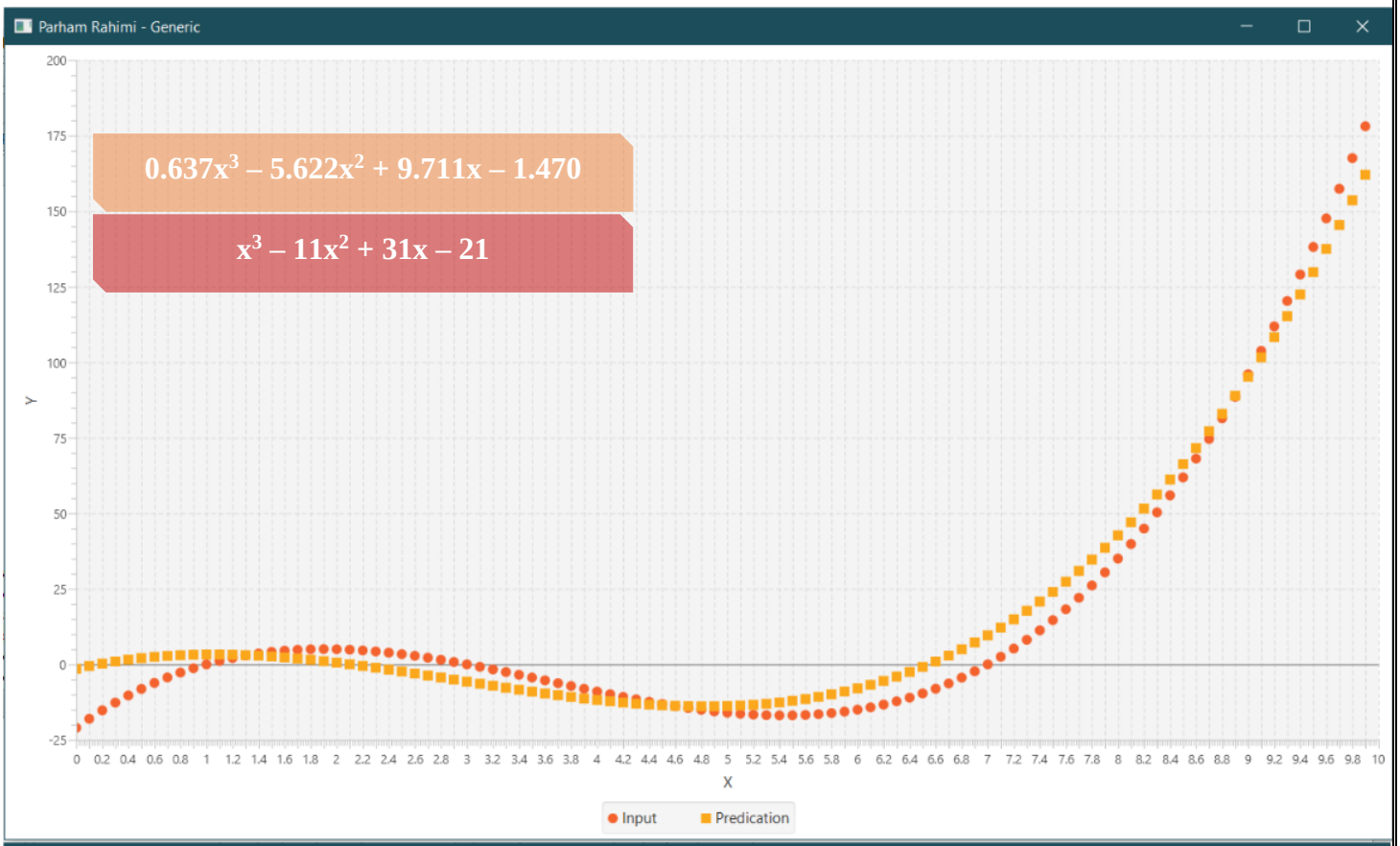


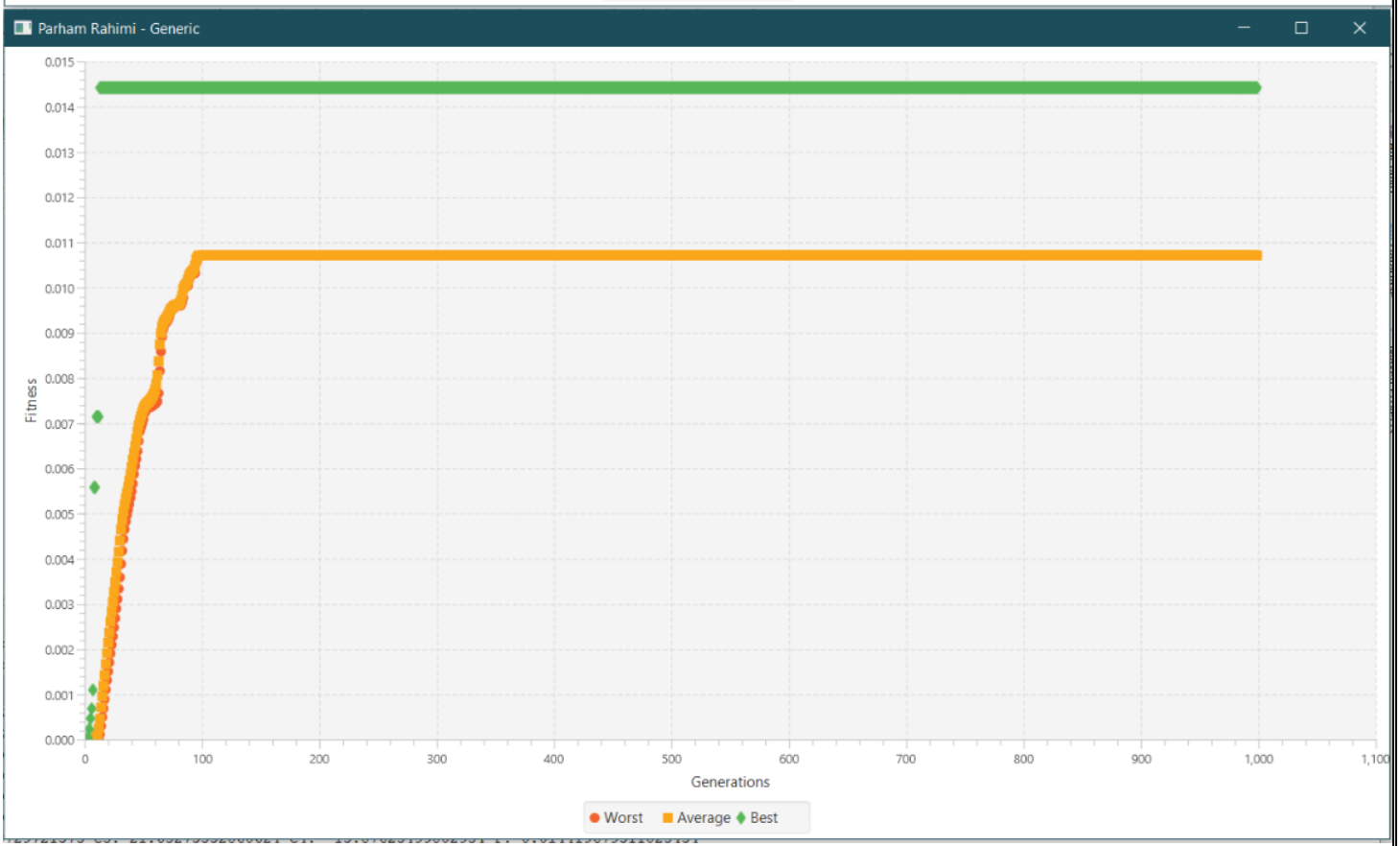
(تعداد نسل: ۱۰۰۰ | اندازه جمعیت: ۱۰۰۰۰ | اندازه تورنومنت: ۲ | نرخ جهش: ۰,۰۵ | واریانس: ۰,۲)





(تعداد نسل: ۱۰۰۰ | اندازه جمعیت: ۱۰۰۰۰ | اندازه تورنومنت: ۲ | نرخ جهش: ۰,۱ | واریانس: ۰,۲)



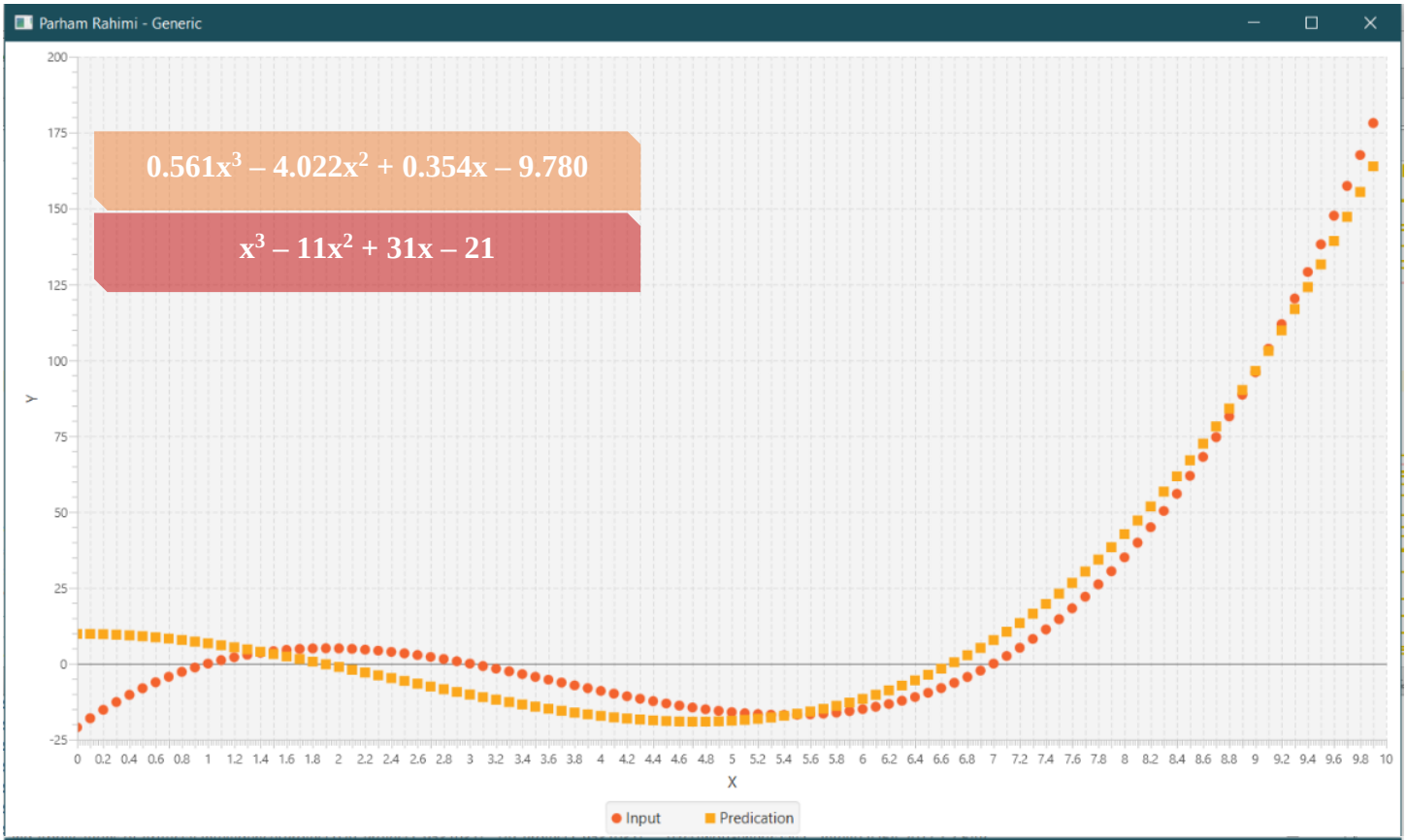


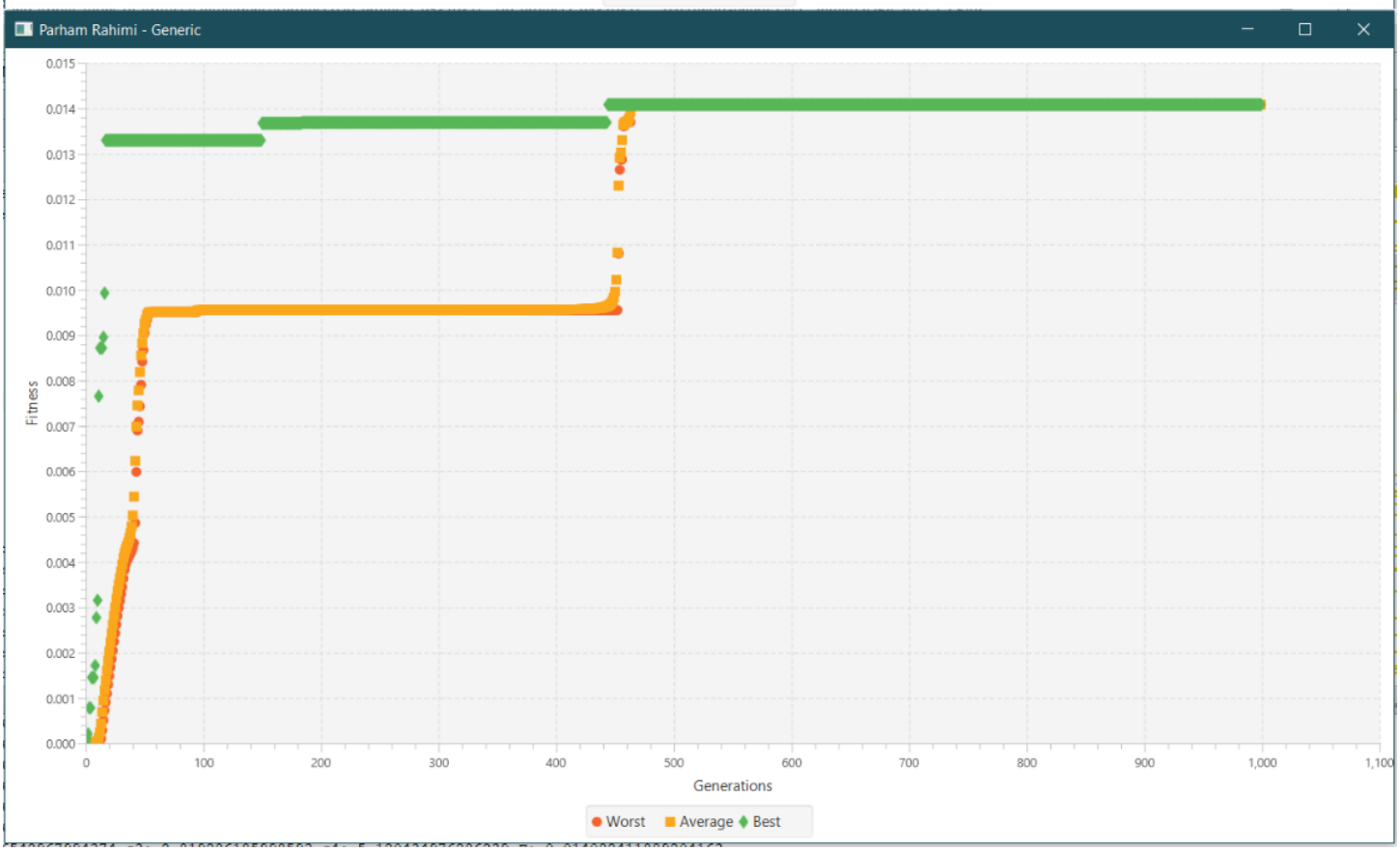
تاثیر واریانس (sigma):

با توجه به آزمایش ها به نظر می آید که با افزایش واریانس تاثیر خطی ای بر شایستگی جواب به دست آمده ندارد. اما بدیهیست که موجب افزایش تنوع و در نتیجه دیرتر همگرا شدن می شود. هرچند وقتی واریانس از حدی کمتر می شود موجب می شود تا جهش بسیار کم اثر شود و درنتیجه ژن جدید تولیدی به حداقل برسد همچنین وقتی که واریانس از حدی بیشتر می شود قابل مشاهده است که هیچگاه به همگرایی و حتی شایستگی حداقلی نمی رسیم چرا که تاثیر شایستگی در برابر تنوع بسیار کم شده است.

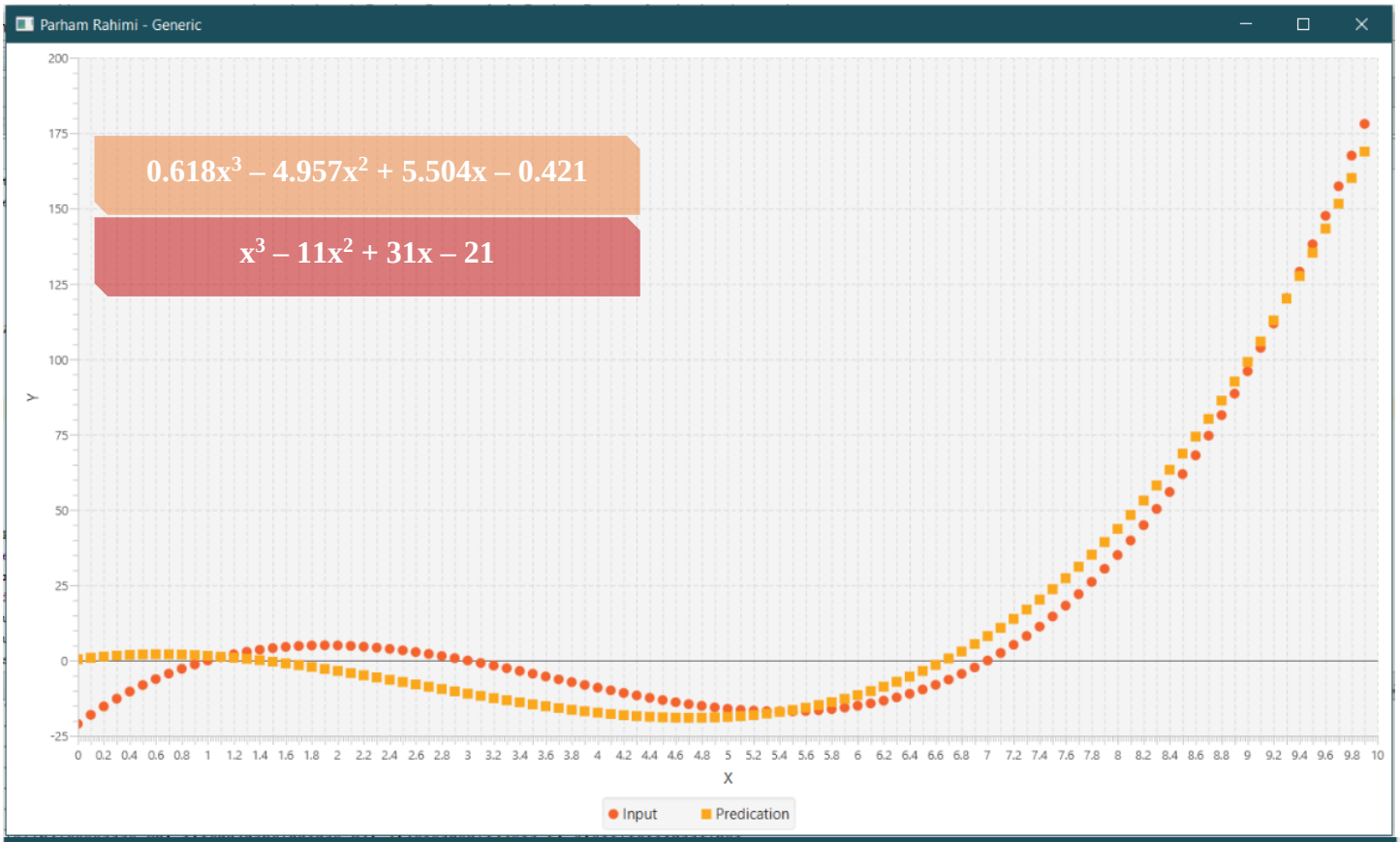
در آزمایش های من روی ورودی $x^3 - 11x^2 + 31x - 21$ و با پارامتر های (تعداد نسل: ۱۰۰۰ | اندازه جمعیت: ۱۰۰۰۰ | اندازه تورنومنت: ۲ | نرخ جهش: ۰,۰۱ | واریانس: متغیر)، بهترین واریانس، ۰,۱ به دست آمد.

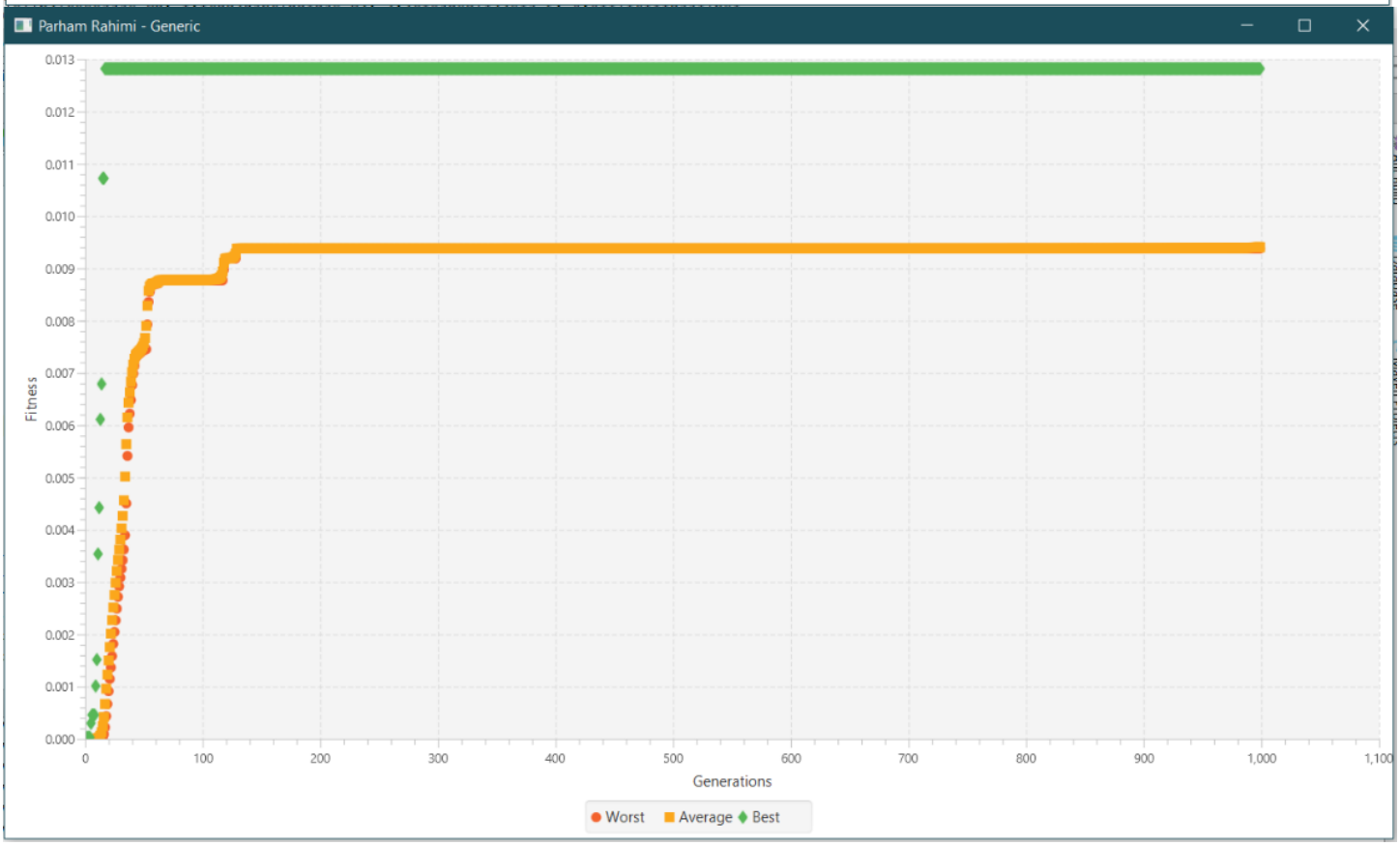
(تعداد نسل: ۱۰۰۰ | اندازه جمعیت: ۱۰۰۰۰ | اندازه تورنومنت: ۲ | نرخ جهش: ۰,۰۱ | واریانس: ۰,۰۰۱)



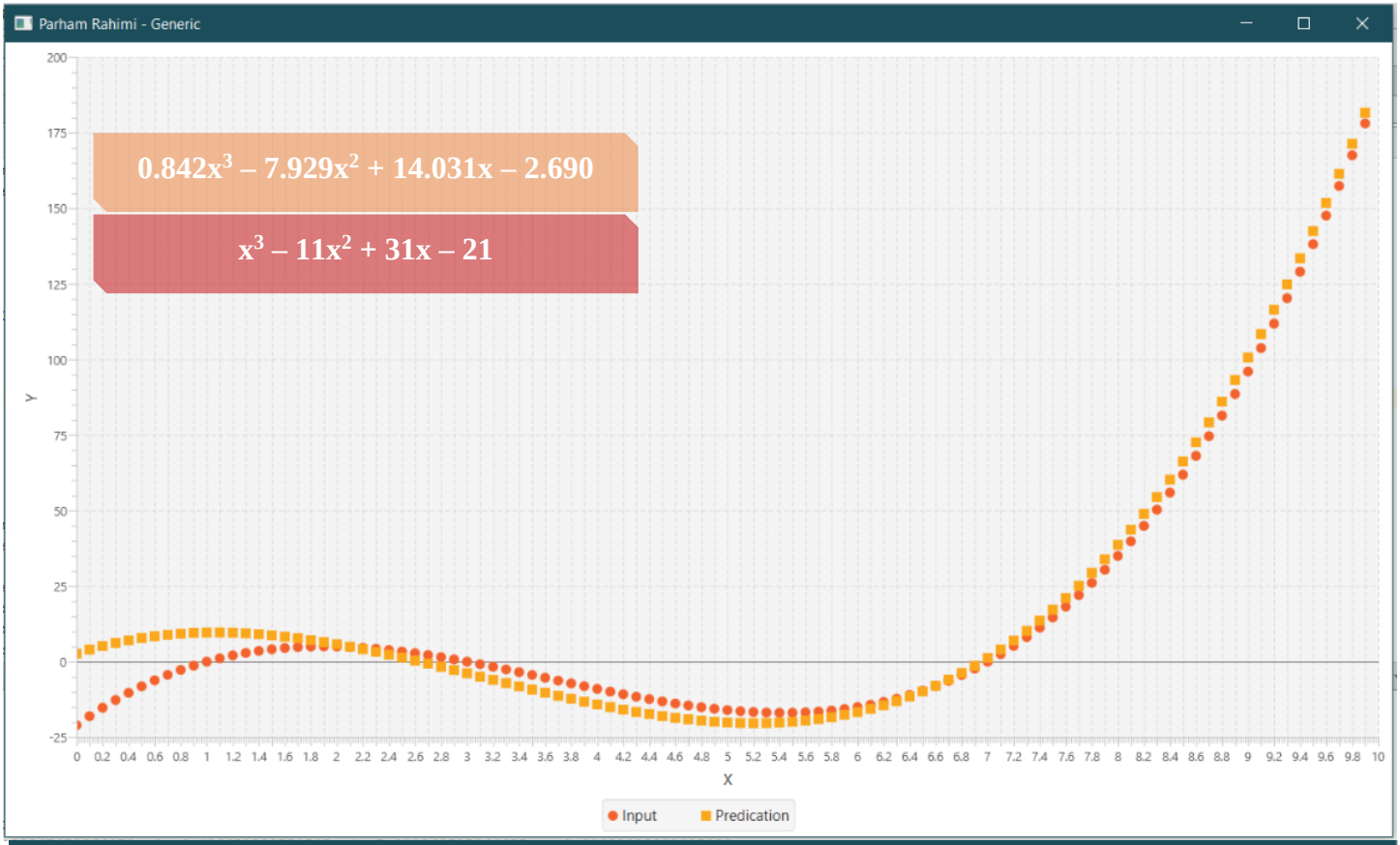


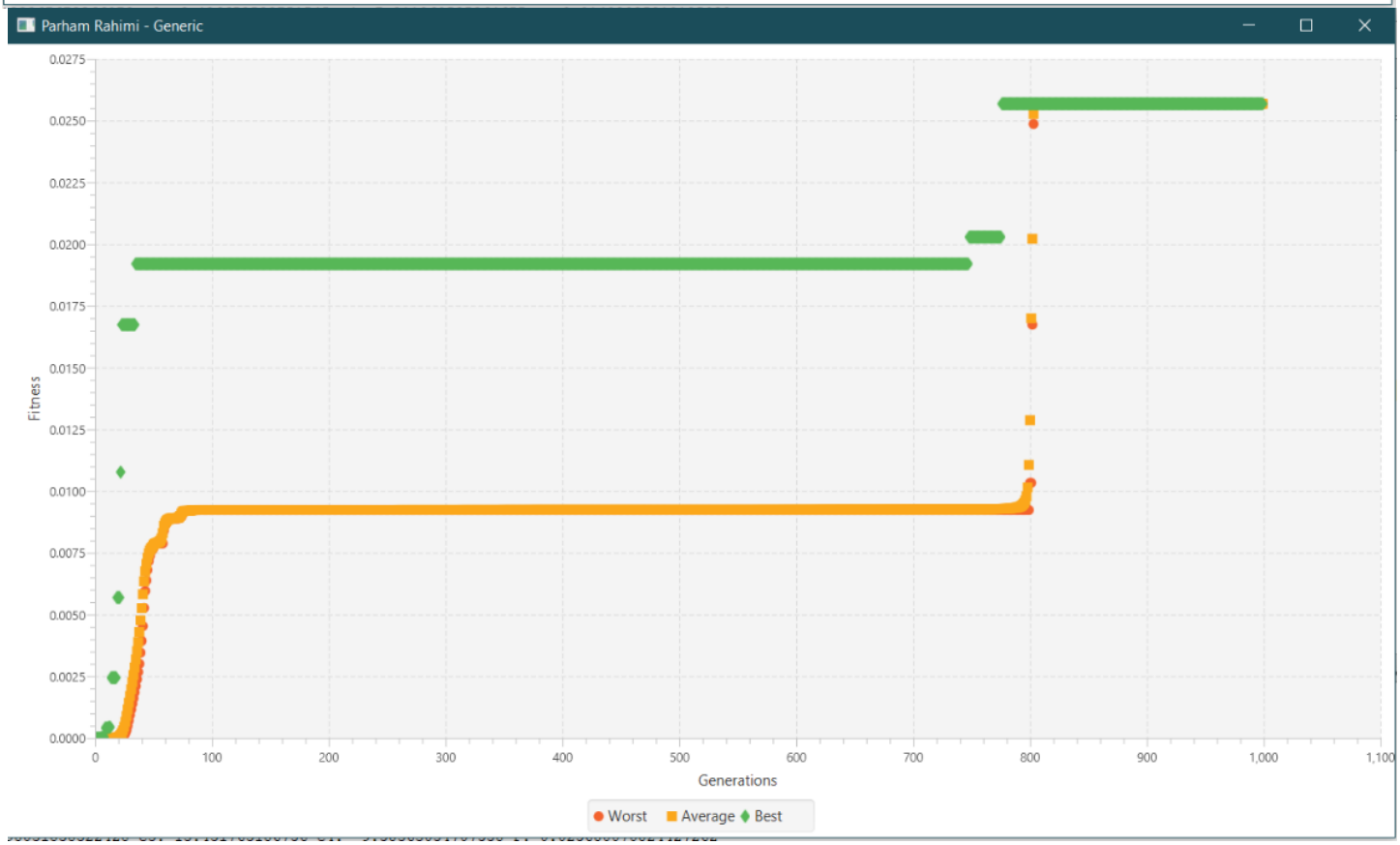
(تعداد نسل: ۱۰۰۰ | اندازه جمعیت: ۱۰۰۰۰ | اندازه تورنومنت: ۲ | نرخ جهش: ۰,۰۱ | واریانس: ۰,۰۱)



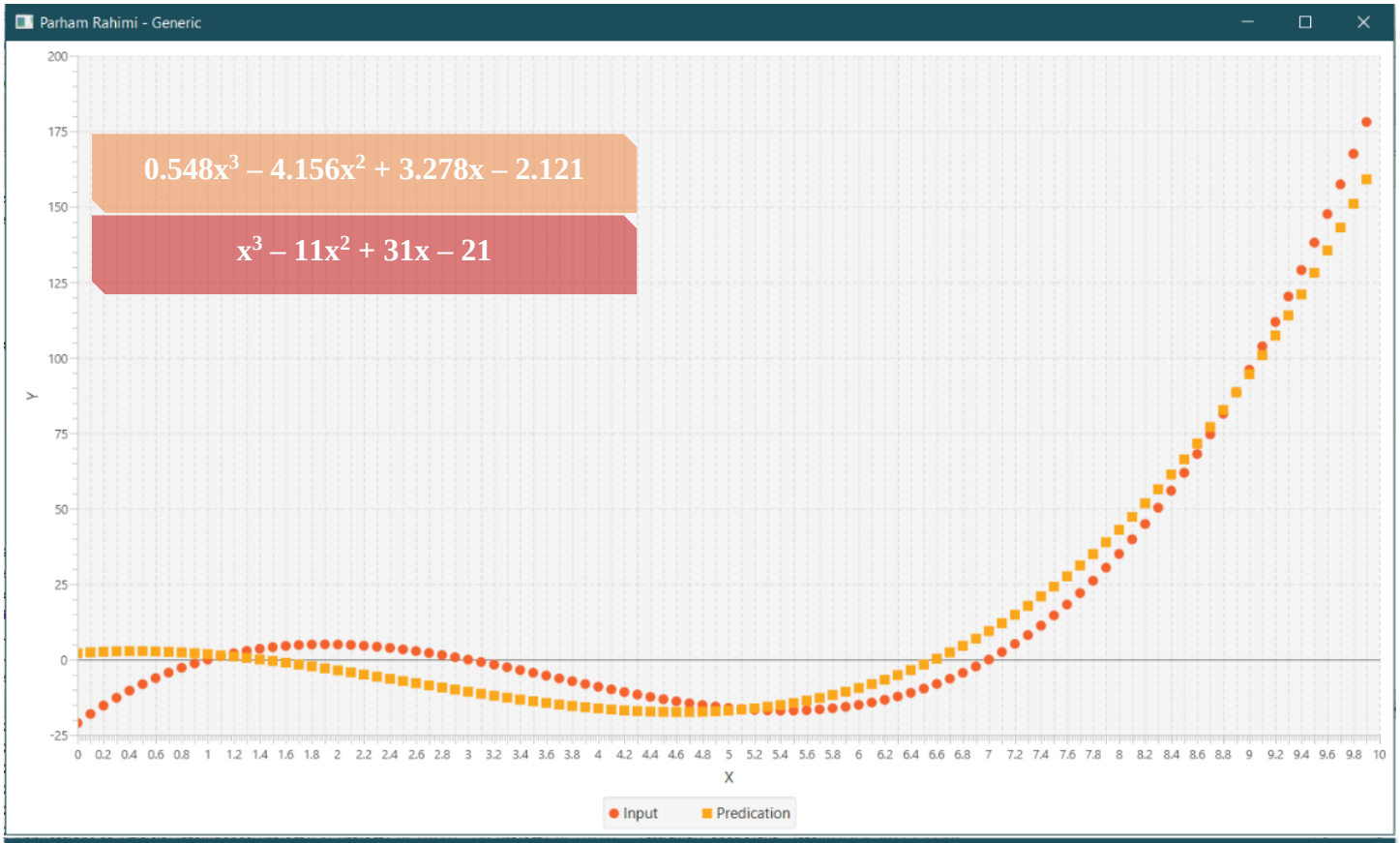


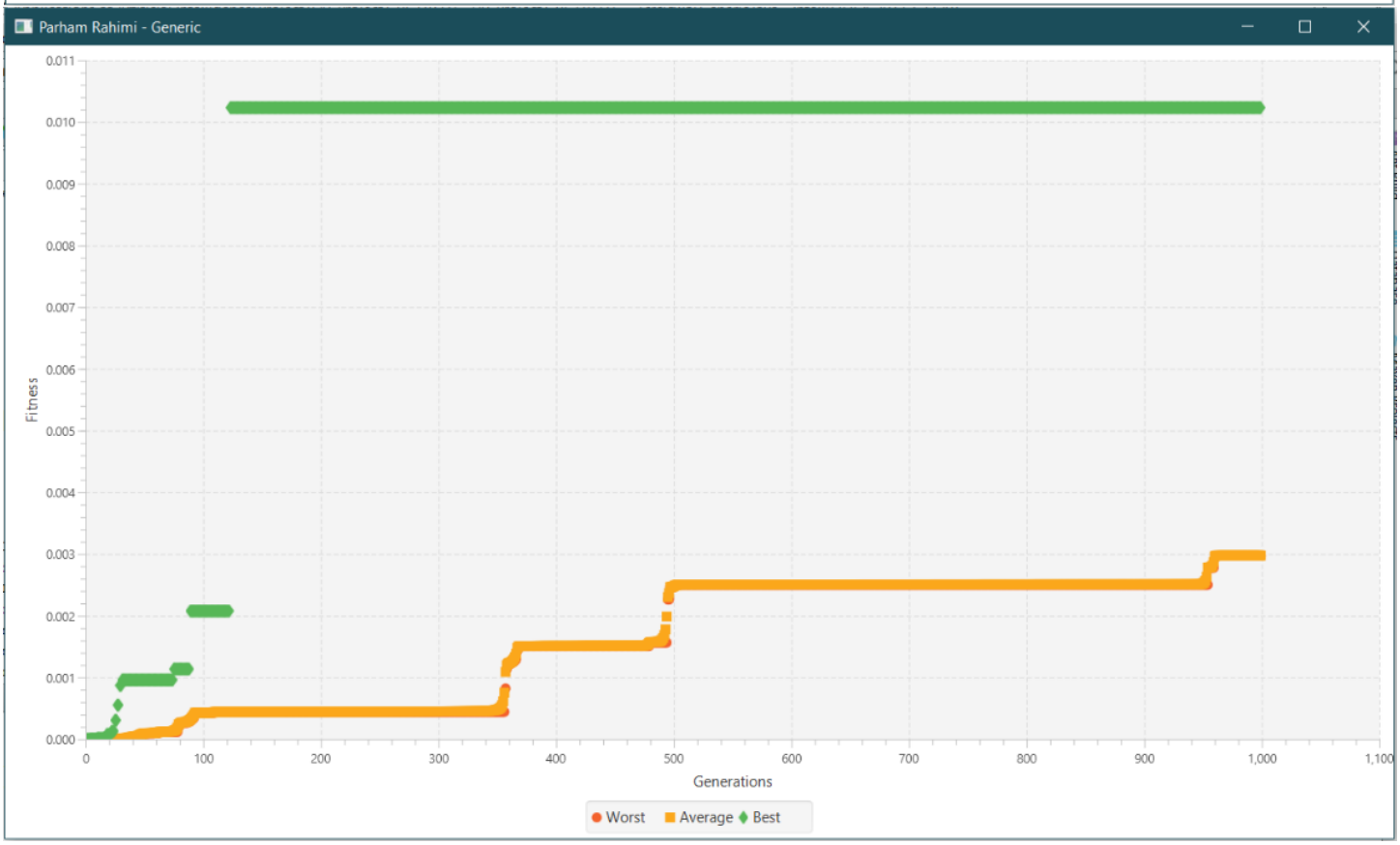
(تعداد نسل: ۱۰۰۰ | اندازه جمعیت: ۱۰۰۰۰ | اندازه تورنومنت: ۲ | نرخ جهش: ۰,۰۱ | واریانس: ۰,۱)





تعداد نسل: ۱۰۰۰ | اندازه جمعیت: ۱۰۰۰۰ | اندازه تورنومنت: ۲ | نرخ جهش: ۰,۰۱ | واریانس: ۱)





تعداد نسل: ۱۰۰۰ | اندازه جمعیت: ۱۰۰۰۰ | اندازه تورنومنت: ۲ | نرخ جهش: ۰,۰۱ | واریانس: ۱۰)