درس ۱۶:

الگوریتم فراابتکاری انجماد تدریجی

تهیه شده توسط گروه بهینه‌یاب

www.behinehyab.com

مقدمه

محاسبه راه حل بهینه *Optimal solution* برای اکثر مسایل بهینه سازی کـه در خیلـی از زمینـه‌هـای کاربردی و عملی مشاهده می‌گردند، کار دشوار و سختی است. در عمل، معمولا به راه حل‌های خـوب کـه از الگوریتم‌های هیوریستیک *Heuristic* یا متاهیوریستیک (همان فراابتکاری)*Metaheuristic* بدست مـی‌آیـد، اکتفا می‌گردد. متاهیوریستیک‌ها مجموعه ای از فنون بهینه سـازی تقریبـی *Approximate optimization techniques* را که عمدتا در طول دو دهـه گذشـته شـهرت پیـدا کـرده انـد، در بـر مـی‌گیرنـد. روش‌هـای فراابتکاری راه حل‌های قابل قبول در زمان معقول را برای مسایل پیچیده و سخت، در زمینه‌های مهندسی و علوم پایه ارایه می نمایند. بـرخلاف الگـوریتم‌هـای بهینـه سـازی دقیـق *Exact optimization algoritms*، فراابتکاری ها بهینه بودن جواب‌های بدست آمده را ضمانت نمی نمایند.

کلمه متاهیوریستیک از کلمه یونانی *"Heuriskein"* به معنای هنر کشف قواعد جدید برای حل مسـایل گرفته شده است. پیشوند "متا" نیز از یک کلمه یونانی به معنای "متدولوژی" سطح بالا گرفته شـده اسـت. واژه " متاهیوریستیک " اولین بار توسط گلوور در سال ۱۹۸۶ ارایه گردید. روش جستجوی متاهیوریستیک را می توان به صورت متدولوژی‌های عمومی سطح بالایی که می توانند به عنوان یـک اسـتراتژی راهنمـا در طراحی هیورستیک‌های اختصاصی برای حل مسایل بهینه سازی تخصصی به کار روند، تعریف کرد.

برخلاف روش‌های دقیق، متاهیوریستیک‌ها (فراابتکاری‌ها) برای مسایل با اندازه‌های بزرگ کاربرد دشوار دارد و راه حل‌هایی راضی کننده ای در زمان معقولی ارایه می نمایند. در این الگوریتم‌ها، هیچ گونه ضمانتی برای یافتن جواب بهینه سراسری یا حدودی از آن وجود ندارد. متاهیوریستیک‌ها (فراابتکـاری‌هـا) در طـول بیست سال گذشته شهرت زیادی پیدا کرده اند. کاربرد و استفاده از آن‌ها در خیلی از مسایل، کـارایی و اثـر بخشی آن‌ها را برای حل مسایل پیچیده و بزرگ نشان می دهد. فراابتکاری‌ها در خیلی از زمینه‌هـا از قبیـل موارد زیر کاربرد دارند:

- ✓ طراحی مهندسی، بهینه سـازی توپولـوژی، بهینـه سـازی مسـایل الکترونیـک، آئرودینامیـک، دینامک سیالات، مخابرات، رباتیک

- ✓ یادگیری ماشین و کاوش داده‌ها
- ✓ مدل سازی سیستم‌ها، شبیه سازی و تحقیق در شیمی، فیزیک، بیولوژی، کنترل، سیگنال، و پردازش تصویر
- ✓ مسایل مسیردهی و برنامه ریزی، برنامه ریزی ربات، مسایل تولید و زمان بندی، حمل و نقل و لجستیک، مدیریت زنجیره تامین

روش‌های مختلف الگوریتم‌های فراابتکاری یا متاهیوریستیک تا به حال پیشنهاد شده است که به صورت زیر است:

- ✓ بهینه سازی کلونی مورچگان *Ant colony optimization*
- ✓ بهینه سازی کلونی زنبوران *Bee colony*
- ✓ الگوریتم‌های ترتیبی *cultural algorithms*
- ✓ الگوریتم‌های با هم تکاملی *Co-evolutionary algorithms*
- ✓ الگوریتم ژنتیک *Genetic algorithm*
- ✓ جستجوی محلی تکراری *Iterated local search*
- ✓ بهینه سازی توده ذرات *particle swarm intelligent*
- ✓ انجماد تدریجی *Simulated Annealing*
- ✓ جستجوی ممنوع *Taboo search*
- ✓ جستجوی همسایگی متغیر *Variable neighboar search*

در طراحی یک فراابتکاری، دو معیار متناقض شامل کاوش *Exploration* و در فضای جستجو (گوناگونی و تنوع) و تبعیت (*Exploition*) از بهترین راه حل‌های پیدا شده، باید در نظر گرفته شوند. در کاوش در ناحیه‌های جستجو نشده بررسی صورت می‌گیرد. در تبعیت، در ناحیه‌های امید بخش که تا به حال در آن ناحیه یک جواب خوب پیدا شده است بررسی بیشتر صورت می‌گیرد. در صورتیکه به کاوش اهمیت بیشتری داده شود، الگوریتم رفتار تصادفی بیشتری خواهد داشت و به سمت رفتار تصادفی میل می کند و در صورتی

که به رفتار تبعیت توجه بیشتری شود، الگوریتم از رفتار تصادفی فاصله می‌گیرد و جستجو تنها در محدوده راه حل‌های خوب به جستجو می پردازد.

معیارهای طبقه بندی زیادی ممکن است برای طبقه بندی فراابتکاری‌ها استفاده شود که در زیر به بعضی از آن‌ها اشاره می‌شود:

الهام گرفته از طبیعت در مقابل عدم الهام از طبیعت

خیلی از الگوریتم‌های فراابتکاری از فرایندهای طبیعی الهام گرفته شده اند: از قبیل الگوریتم‌های اجتماع مورچگان و زنبور عسل که از هوش توده ای از گونه‌های مختلف مورچگان و زنبوران استفاده می کنند.

نحوه استفاده از حافظه

بعضی از الگوریتم‌های فراابتکاری از قبیل انجماد تدریجی بدون حافظه هستند و حرکت‌های قبلی را در جایی ذخیره نمی کنند. در مقابل الگوریتم انجماد تدریجی از یک حافظه که بعضی از اطلاعات را در طول جستجو بدست می‌آورد، ذخیره می کند.

قطعی در مقابل احتمالی

یک الگوریتم قطعی، یک مسئله بهینه سازی را از طریق تصمیم گیری قطعی حل می نماید(برای مثال الگوریتم جستجوی محلی و جستجوی ممنوع). در الگوریتم‌های فراابتکاری احتمالی، بعضی از قواعد احتمالی در فرایند جستجو مورد استفاده قرار می‌گیرد که می توان به الگوریتم انجماد تدریجی و الگوریتم ژنتیک اشاره کرد. در الگوریتم‌های قطعی، با داشتن یک راه حل اولیه و اجراهای متفاوت، تنها یک جواب نهایی بدست می‌آید و در حالی که در الگوریتم‌های تصادفی، با داشتن یک راه حل اولیه و اجراهای متفاوت، ممکن است که جواب‌های نهایی متفاوتی بدست آید.

تکراری در مقابل حریصانه

در الگوریتم‌های تکراری، الگوریتم با یک راه حل کامل شروع شده و در هر تکرار راه حل یا راه حل‌ها تغییر پیدا می کنند. در الگوریتم‌های حریصانه یک راه حل کامل در اختیار نبوده بلکه با یک راه حل ساخته نشده شروع شده و در هر مرحله، یک متغیر تصمیم از مسئله یک قسمت از راه حل را می سازد. اغلب الگوریتم‌های فراابتکاری، الگوریتم‌های تکراری هستند.

در ادامه الگوریتم انجماد تدریجی که الگوریتمی است که از طبیعت الهام گرفته است و در رده الگوریتم‌های احتمالی، و تکراری بدون حافظه قرار می‌گیرد صحبت می‌شود.

معرفی الگوریتم انجماد تدریجی

الگوریتم انجماد تدریجی یا **الگوریتم گرم و سرد کردن شبیه سازی شده** یک روش تصادفی است که مکانیزم آماری جهت یافتن جواب مسایل بهینه سازی استفاده می کند. این الگوریتم که به اختصار با ***SA*** یا ***Simulate Annealing*** نمایش داده می شود بر مبنای دو قاعده از فیزیک آماری عمل می نماید.

قاعده اول بر این اساس است که وقتی تعادل ترمودینامیکی به یک دمای مشخص رسید، احتمال یک سیستم فیزیک برای داشتن سطح انرژی E با فاکتور بالتزمن *Boltzmann Factor* که رابطه آن $e^{\frac{-E}{k_B T}}$ که در آن k_B توزیع بالتزمن در دمای مربوطه باشد، متناسب است. براساس این قاعده، احتمال پذیرش جواب ها در دماهای مختلف متفاوت بوده و متناسب با تابع معرفی شده می باشد.

قاعده دوم نحوه رسیدن به تعادل ترمودینامیکی در یک دمای مشخص را بیان می کند. اگر یک تغییر شکل منجر به افزایش تابع هدف یا انرژی به اندازه ΔE بشود، این تغییر با احتمال $e^{\frac{-\Delta E}{T}}$ پذیرفته می شود. این پذیرش از طریق تولید یک عدد تصادفی با توزیع یکنواخت در فاصله بین ۰ و یک و مقایسه آن با تابع تعریف شده انجام می گردد. در صورتی که مقدار عدد به دست آمده از تابع توزیع یکنواخت کوچکتر از تابع تعریف شده باشد، تغییر شکل پذیرفته می شود و در غیر این صورت، تغییر شکل پذیرفته نمی شود.

در دمای بالا و مراحل اولیه الگوریتم، مقدار $e^{\frac{-\Delta E}{T}}$ نزدیک به یک بوده و در نتیجه اکثر حرکت ها پذیرفته می شود و الگوریتم رفتاری شبیه به یک جستجوی تصادفی را خواهد داشت. در دماهای پایین و اواخر الگوریتم، مقدار $e^{\frac{-\Delta E}{T}}$ به صفر نزدیک می شود و اکثر جواب های بدتر رد می شوند و تنها جواب های خوب پذیرفته می شوند و شانس جابه جایی یک جواب خوب با یک جواب بدتر کاهش می یابد. این تغییر رویه الگوریتم، خروج الگوریتم از جواب بهینه محلی را تضمین می کند و از طرف دیگر، کاهش احتمال پذیرش جواب بدتر با کاهش دما، موجب تضمین همگرایی ***SA*** می شود.

در این الگوریتم، x_1 معرف یک راه حل اولیه امکان پذیر است که محدودیت های مدل را رعایت نموده است. $F(x)$ مقدار تابع هدف به ازای x بوده و x_c یک راه حل جاری در هر مرحله می باشد. $V(x_c)$ مجموعه ای است که برای نشان دادن همسایه های x_c به کار می رود. P یک عدد تصادفی می باشد که در زمان برخورد با جواب بدتر به منظور بررسی پذیرش یا عدم پذیرش آن جواب به کار رود.

در **قدم اول**، تنظیمات مربوط به الگوریتم انجام می شود. این تنظیمات شامل دمای اولیه و تعیین پارامترهای کاهش دما است.

در **قدم دوم** الگوریتم، یک راه حل اولیه ایجاد و مقدار تابع هدف آن محاسبه می گردد. در این قدم، راه حل اولیه به عنوان راه حل جاری x_c و بهترین راه حل x^* نیز در نظر گرفته می شود. البته، این دو راه حل در طول الگوریتم تغییر خواهد یافت. همچنین در ابتدای الگوریتم، مقدار تابع هدف راه حل اولیه، به عنوان مقدار تابع هدف راه حل جاری $F(x_c)$ و بهترین راه حل $F(x^*)$ در نظر گرفته می شود. در این قدم دمای اولیه T_0 به عنوان دمای جاری یا همان T_c در نظر گرفته می شود.

در **قدم سوم**، برای هر راه حل جاری، یک راه حل همسایه که به صورت $x_{c'}$ در همسایگی آن پیدا شده و مقدار تابع هدف آن به صورت $F(x_{c'})$ محاسبه می گردد.

در **قدم چهارم**، از تابع بولتزمن برای حرکت به همسایگی استفاده می شود. در صورتی که همسایه جدید مقدار تابع هدف جاری را بهبود بدهد یا برابر آن باشد، راه حل همسایه به عنوان راه حل جاری پذیرفته می شود که در این صورت تغییرات $x_c \leftarrow x_{c'}$ و $F(x_c) \leftarrow F(x_{c'})$ انجام می شود. در غیر این صورت، اگر راه حل همسایه منجر به بدتر شدن تابع هدف جاری گردد، یعنی $F(x_c) < F(x_{c'})$، همسایه با تابع احتمال بولتزمن $e^{\frac{-\Delta F}{T_c}}$ که در آن $\Delta F = F(x_{c'}) - F(x_c)$ پذیرفته می شود. برای این منظور مقدار تصادفی $p \sim U(0,1)$ تولید شده و سپس با تابع احتمال $e^{\frac{-\Delta F}{T_c}}$ مقایسه می گردد. در صورتیکه مقدار P کمتر از تابع احتمال باشد، راه حل همسایه به عنوان راه حل جاری پذیرفته می شود و در غیر این صورت راه حل جاری تغییر نمی کند.

بعد از یافتن هر راه حل جدید که منجر به بهبود راه حل جاری گردد، آن راه حل با بهترین راه حل مقایسه می گردد. اگر این راه حل از بهترین راه حلی که تا به حال پیدا شده بهتر باشد، $F(x_{c'}) \le F(x^*)$ (به دنبال کمینه سازی هستیم)، جایگزین بهترین راه حل $x_{c'} \rightarrow x^*$ و $F(x_{c'}) \rightarrow F(x^*)$ می گردد. این قدم تا زمانی ادامه می یابد تعداد مشخصی از جستجوها انجام شود.

در **قدم پنجم**، بعد از اتمام یک زنجیره(شامل تعدادی مشخص جستجو است که معمولا برابر با یک در نظر گرفته می شود)، کاهش دما اتفاق خواهد افتاد. معمولا برای کاهش دما از یک ضریب ثابت $0 < r < 1$ استفاده می شود. رابطه کاهش دما در این حالب به صورت $T \leftarrow rT$ تعریف می شود.

در **قدم ششم**، شرط توقف الگوریتم بررسی می گردد. در صورتی که شرط توقف برآورده شود، الگوریتم متوقف و در غیر این صورت به **قدم سوم** بر می گردد. شرط توقف های مختلف می توان تعریف کرد. یک شرط توقف رایج، رسیدن دمای الگوریتم به یک دمای انتهایی یا همان T_f می باشد که به صورت $T \le T_f$ تعریف می گردد.

عوامل مختلفی در عملکرد الگوریتم انجماد تدریجی موثر است که می توان به موارد زیر اشاره کرد.

۱- دمای اولیه

۲- قاعده کاهش دما

۳- قاعده توقف

در ادامه هر یک از این موارد مورد بررسی بیشتر قرار می گیرد.

دمای اولیه

استفاده از دمای اولیه بالا موجب می گردد که الگوریتم رفتاری شبیه به یک جستجوی تصادفی به خود بگیرید و استفاده از دماهای پایین برای دمای اولیه، موجب می گردد که الگوریتم تبدیل به یک الگوریتم جستجوی محلی گردد. در الگوریتم انجماد تدریجی در هنگام تعیین دمای اولیه، باید دمای اولیه به نحوی طراحی گردد که موازنه ای بین دو حالت فوق اتفاق بیافتد. دو استراتژی مهم در هنگام تنظیم این پارامتر وجود دارد.

پذیرش کلیه راه حل ها

در این حالت مقدار دمای اولیه به اندازه ای بزرگ در نظر گرفته می شود که کلیه راه حل ها در طول فاز اولیه ای از الگوریتم پذیرفته شود. مشکل این روش هزینه بالای محاسباتی آن است.

وابسته به انحراف معیار

در این استراتژی، دمای اولیه برابر با $k\sigma$ براساس یک سری ازمایشات اولیه، محاسبه می گردد که در آن σ نشان دهنده انحراف معیار استاندارد توابع هدف جواب های بدست امده در آزمایشات اولیه بوده و $k=-3/\ln(p)$ که احتمال پذیرش p مرتبط با ناحیه 3σ در تابع توزیع احتمال می باشد.

قاعده کاهش دما

به طور کلی، بین کیفیت نتایج و سرعت انجماد رابطه محکمی وجود دارد. مقدار دما همواره مثبت بوده و زمانی که تعداد تکرارها به سمت بی نهایت میل کند، مقدار دما نیز به سمت صفر میل می کند. با رعایت این اصول، قواعد مختلفی برای کاهش دما وجود دارد که در زیر به بعضی از آن ها اشاره شده است:

قاعده خطی

در این قاعده، مقدار دما به صورت خطی کاهش پیدا می کند. فرمول زیر رابطه کاهش دما به صورت خطی را بیان می کند.

$$T_i = T_0 - i \times \beta$$

که در آن T_0 دمای اولیه، i شماره تکرار کاهش دما و β ضریب ثابتی بین صفر و یک می باشد.

قاعده هندسی

یکی از قواعد مشهور کاهش دما، قاعده هندسی کاهش دما می باشد که به صورت $T_{k+1} = \alpha T_k$ معرفی می گردد و در آن $0 < \alpha < 1$ و α ضریب ثابتی است. این قاعده به دلیل سادگی در بسیاری از موارد استفاده می شود.

قاعده لگاریتمی

این قاعده به صورت تابع زیر تعریف می شود. این قاعده نسبت به دو قاعده دیگر، سرعت کاهش دمای آن آرامتر بوده و باعث همگرایی بیشتری به سمت بهینه سراسری می شود.

$$T_i = \frac{T_0}{\log(i)}$$

قاعده توقف

قاعده توقف نیز یکی از اجزای الگوریتم انجماد تدریجی است که بر کیفیت الگوریتم تاثیر زیادی دارد. معمولا قاعده توقف به نحوی باید تعیین گردد که احتمال پذیرفتن جواب های بد به عنوان جواب بهینه نهایی به شدت کاهش یابد. قواعد زیر نمونه ای از قواعد توقف رایج می باشند.

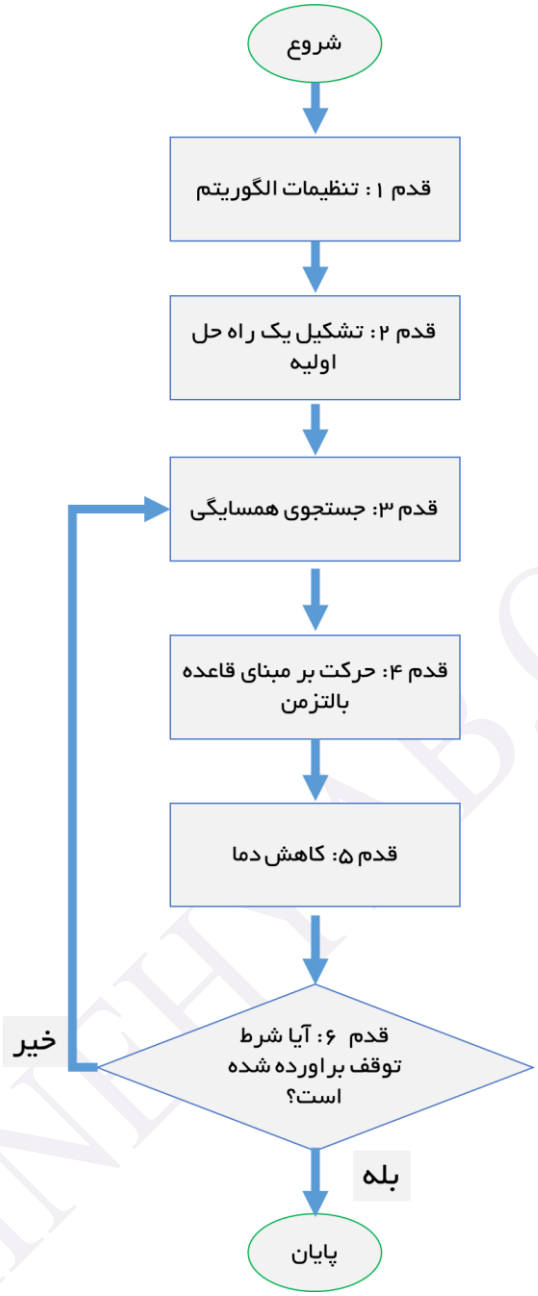
رسیدن به یک دمای نهایی

رسیدن دمای الگوریتم به یک دمای انتهایی معروف ترین قاعده توقف می باشد. برای مثال ممکن است رسیدن دمای الگوریتم به دمای $T_f = 0.01$ به عنوان قاعده توقف تعیین گردد.

عدم بهبود در جواب نهایی

رسیدن تعداد حرکت های بدون بهبود در بهترین حرکت نیز یکی از قواعد توقف مشهور می باشد. در این حالت، شمارنده ای برای شمارش تعداد تکرار اختصاص داده می شود. هر گاه مقدار بهترین جوابی که توسط الگوریتم یافته شده است، بهبود پیدا کند این شمارنده صفر می گردد. قاعده توقف فوق به صورت رسیدن شمارنده به یک حد بالایی از قبل تعیین شده، تعریف می گردد. در این حالت، فرض بر این است که اگر الگوریتم تا حد معینی به جستجوی خود ادامه دهد و فایده ای نداشته باشد، بهتر است الگوریتم متوقف شود زیرا احتمال یافتن نقطه ای بهتر از بهترین نقطه، خیلی کم است.

در ادامه مراحل الگوریتم انجماد تدریجی را به صورت فلوچارت بیان می کنیم.

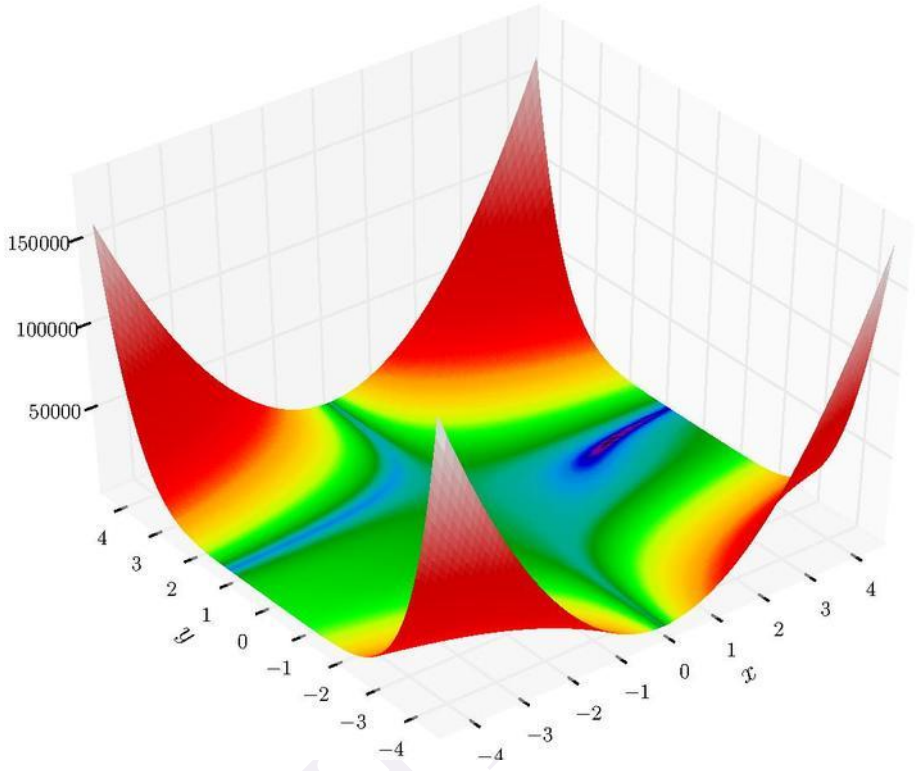


مثال

در این قسمت می خواهیم یک مسئله بهینه سازی بدون محدودیت را بـا اسـتفاده از روش جسـتجوی ممنوع حل نماییم. در این مثال به دنبال کمینه کردن تابع هدف زیر هستیم.

$$f(x,y)=(1.5-x+xy)^2+(2.25-x+xy^2)^2+(2.625-x+xy^3)^2$$

مقدار بهینه این مدل برابر با صفر و مقدار متغیرهای برابر با $(x^*, y^*) = (3, 0.5)$ هستند. نمـایش سـه بعدی این تابع به صورت زیر است.



برای یافتن جواب بهینه این تابع مراحل زیر باید طی شود.

قدم ۱: تنظیمات الگوریتم

در این قدم نتظیمات اولیه برای اجرای الگوریتم آورده می شود. دمای اولیه الگوریتم برابر با $T_0 = 1000$ در نظر گرفته می شود. از قاعده کاهش دمای هندسی استفاده می شود و نرخ کاهش دما برابر با $\alpha = 0.99$ لحاظ می گردد. بازه ای تغییرات دو متغیر $x \in [0,10]$ و $y \in [0,10]$ است.

قدم ۲: تشکیل یک راه حل اولیه

برای نمایش جواب ها و ساخت جواب های همسایگی از رویکرد مبنای دو استفاده شد. هر عدد صـحیح در مبنای ده را می توان به صورت یک عدد در مبنای ۲ نمایش داد.

$$(x')_{10} = (a_0, a_1, a_2, ..., a_{n-1}, a_n) \rightarrow x' = \sum_{i=0}^{n} a_i\, 2^i$$

در این تمرین، در صورت استفاده از نمایش دودویی هر جواب، دقت جواب بدون رقم اعشار در نظر گرفته می شود و این برای محاسبه جواب بهینه کافی نیست. برای افزایش دقت و تولید اعداد بین صفر تا ۱۰، اعدادی بر مبنای دو تولید می شود که بین صفر تا ۱۰۲۳ است و سپس عدد تولید شده بر عدد ۱۰۰ تقسیم می شود. برای نمونه مثال زیر را در نظر بگیرید.

$$(x'')_2 = (0,0,1,1,0,0,1,1,0,0)$$
$$\rightarrow (x')_{10} = 0\times 2^0 + 0\times 2^1 + 1\times 2^2 + 1\times 2^3 + 0\times 2^4 + 0\times 2^5 + 1\times 2^6 + 1\times 2^7 + 0\times 2^8 + 0\times 2^9$$
$$= 4 + 8 + 64 + 128 = 204$$

برای تولید یک مقدار برای متغیر x تنها کافی است مقدار تولید شده در بالا را بر ۱۰۰ تقسیم نماییم. در این صورت مقدار تولید شده برای x برابر با 2.04 می شود.

در انتهای این قدم، مقدار تابع هدف براساس دو متغیر x و y با توجه به تابع هدف زیر محاسبه می شود.

$$f(x,y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2$$

قدم ۳: جستجوی همسایگی

برای ایجاد یک جواب در همسایگی جواب فعلی، روند زیر انتخاب شده است.

۱- یک عدد تصادفی پیوسته بین صفر و یک تولید شود.

۲- اگر عدد تصادفی تولید شده کمتر از ۰.۵ بود آنگاه متغیر x تغییر می کند و در غیر این صورت متغیر y تغییر می کند.

۳- یک عدد تصادفی گسسته بین صفر و نه تولید می شود. عدد انتخاب شده را i می نامیم. در این صورت مولفه a_i اگر صفر است به یک و اگر یک است به صفر تغییر می دهیم.

برای مثال همسایگی جواب $(x,y)=(2.04,7.91)$ را تولید کنید.

$$x=2.91\rightarrow(x')_{10}=291\rightarrow(x'')_2=(11001100)_2$$
$$y=7.91\rightarrow(y')_{10}=791\rightarrow(y'')_2=(1100010111)_2$$

مقدار عدد تصادفی برای انتخاب یکی از دو متغیر برابر ۰.۵۷ در نظر گرفته می شود. در این صورت متغیر y انتخاب می شود. یک عدد تصادفی بین صفر و نه تولید می شود. مقدار این عدد برابر ۵ می شود. در این صورت مولفه a_5 یا $(1100\underline{1}0111)_2$ که برابر ۱ است انتخاب می شود. در این صورت مقدار ۱ به مقدار صفر تغییر می کند و مقدار جدید برابر با $(1100\underline{0}0111)_2$ می شود. در این صورت مقدار y به صورت زیر محاسبه می شود.

$$(y'')_2=(1100000111)_2\rightarrow(y')_{10}=775\rightarrow y=7.75$$

در این قدم، جواب کنونی $(x,y)=(2.04,7.91)$ به جواب همسایگی $(x_n,y_n)=(2.04,7.75)$ می رویم.

در انتهای این قدم مقدار تابع هدف جواب جدید محاسبه می شود و آن را f_n می نامیم.

قدم ۴: حرکت بر مبنای قاعده بالتزمن

در این قدم اگر $f_n<f$ باشد در این صورت جواب فعلی برابر با $(x_c,y_c)=(x_n,y_n)$ می شود. اگر $f_n\geq f$ باشد مقدار $e^{-\frac{\Delta f}{T}}$ که در آن $\Delta f=f_n-f$ است محاسبه می شود. یک عدد تصادفی بین صفر و یک ، r ، تولید می شود. اگر $r<e^{-\frac{\Delta f}{T}}$ باشد آنگاه $(x_c,y_c)=(x_n,y_n)$ می شود و این به این معنا است که با احتمال r به جواب بدتر حرکت می کنیم. و در غیر این صورت جواب کنونی تغییر نمی کند.

قدم ۵: کاهش دما

در این مثال از قاعده هندسی برای کاهش دما استفاده می شود. به این صورت که دما جدید از رابطه زیر محاسبه می شود.

$$T = 0.99 \times 1000 = 990$$

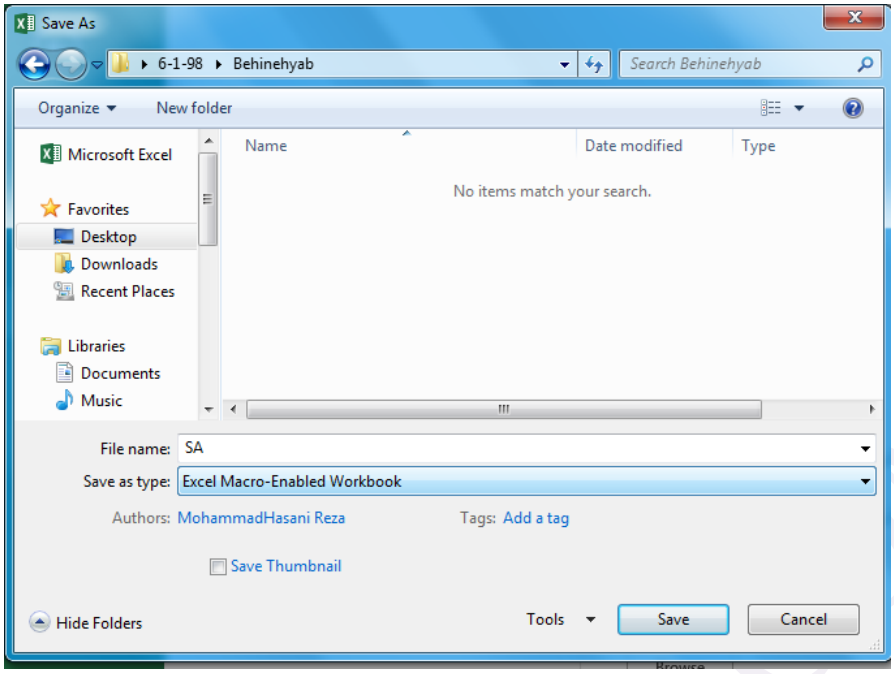
قدم ۶: آیا شرط توقف برآورده شده است؟

شرط های توقف مختلفی می توان ارایه کرد. در این مثال زمانی الگوریتم به پایان می رسد کـه دمـا از ۰.۰۱ کمتر شود.

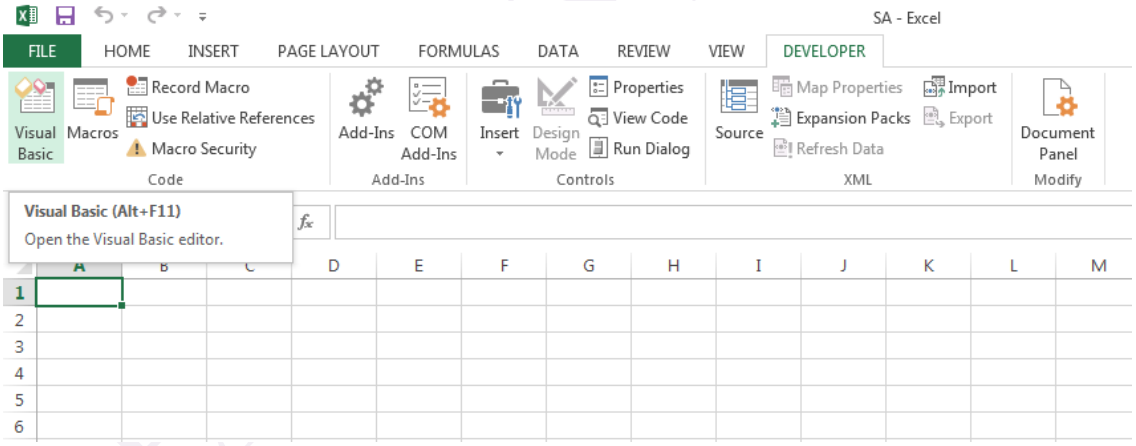
با توجه به موارد ذکر شده اکنون می توان جواب بهینه تابع *f(x,y)* را محاسبه کرد. بـرای پیـاده سـازی الگوریتم گرم و سرد کردن شبیه سازی شده از زبان *Visual Basic for Application* که زبان برنامه نویسی در نرم افزارهای *Office* است استفاده می کنیم. این زبان همان زبان برنامه نویسی *Visual Basic* است ولـی این امکان در آن فراهم شده است که از قابلیت های *Office* در این زبان بهره برد. در کنار این زبـان برنامـه نویسی از قابلیت های *Excel* برای نمایش جواب ها استفاده می شود.

در ادامه به صورت جزیی تمامی مراحل حل تابع *f(x,y)* با زبان *Visual Basic for Application* توضـیح داده می شود.

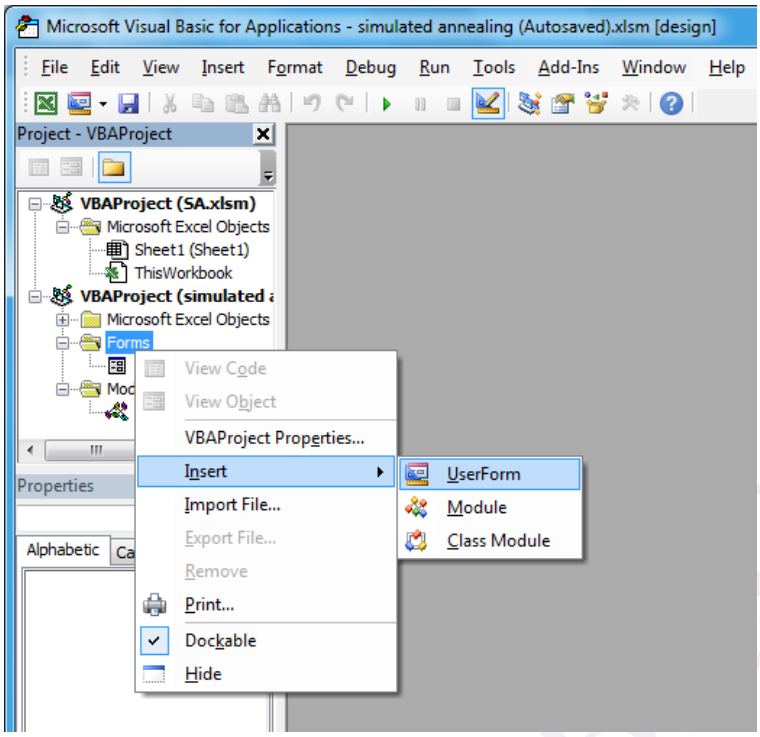
در ابتدا برنامه اکسل را باز می کنید و یک فایل با نام *SA* یـک فایـل اکسـل بـه فرمـت *Excel Macro-Enabled Workbook* ایجاد می کنیم. این فرمت به ما امکان می دهد که در محیط اکسل بتـوان بـه زبـان *Visual Basic for Application* برنامه نویسی کنیم.



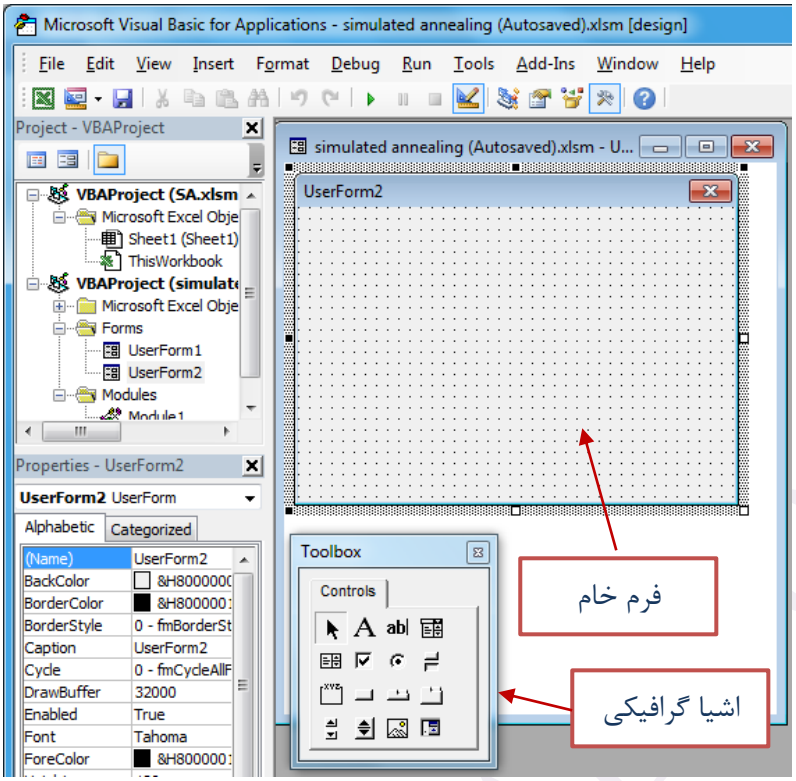
از تب *developer* بر روی دکمه *Visual Basic* کلیک می کنیم تا برنامه *Visual Basic for Application* باز شود.



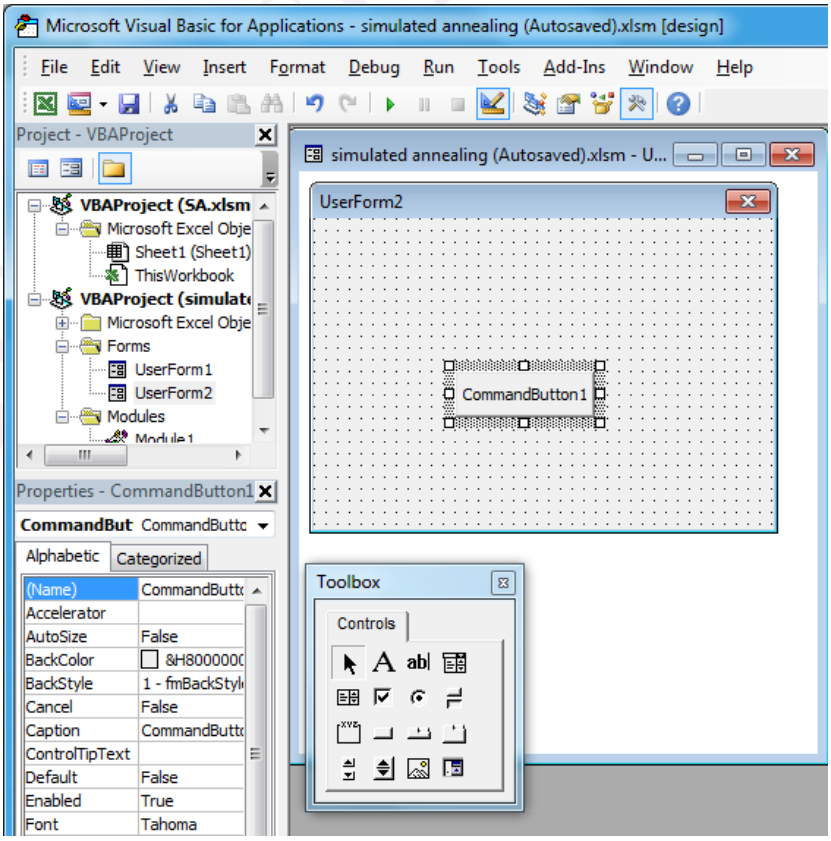
پس از باز کردن این برنامه وارد محیط برنامه نویسی برنامه اکسل می شوید. برای ایجاد یک پروژه می توانید از پنجره *VBA Project* استفاده کنید. در این تمرین با ایجاد یک فرم ساده کار را آغاز می کنیم.



فرم ایجاد شده این امکان را به شما می دهد تا شی های گرافیکی مختلفی را در فرم قرار دهید. البته باید به این نکته اشاره کرد که قابلیت های شما در این زبان برنامه نویسی در اکسل محدود است و در صورت نیاز به اشیا گرافیکی پیشرفته تر باید از سایر نرم افزارها مانند *Visual Studio* استفاده کنید. پس از باز کردن یک فرم جدید، نام آن را *Userform*۱ می نامیم و فرمی به صورت زیر باز می شود.

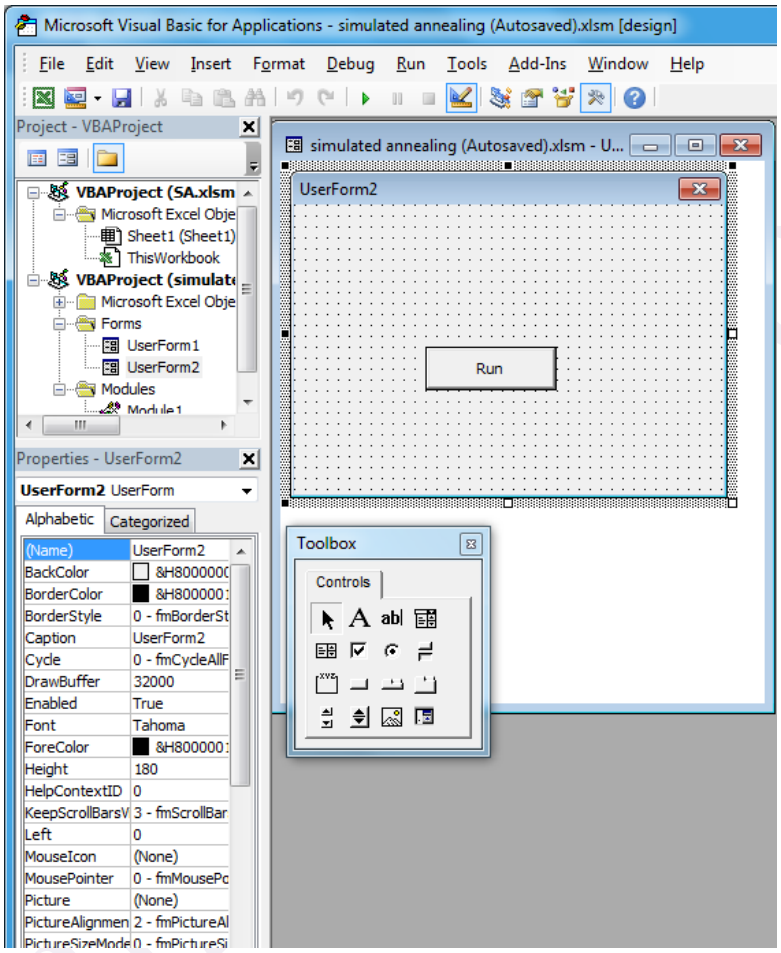


برای ایجاد برنامه تنها به یک دکمه یا *button* نیاز است که با درگ از *Toolbox* و گذاشتن در فرم خام ایجاد می شود. که در این صورت فرم شما به صورت زیر می شود.

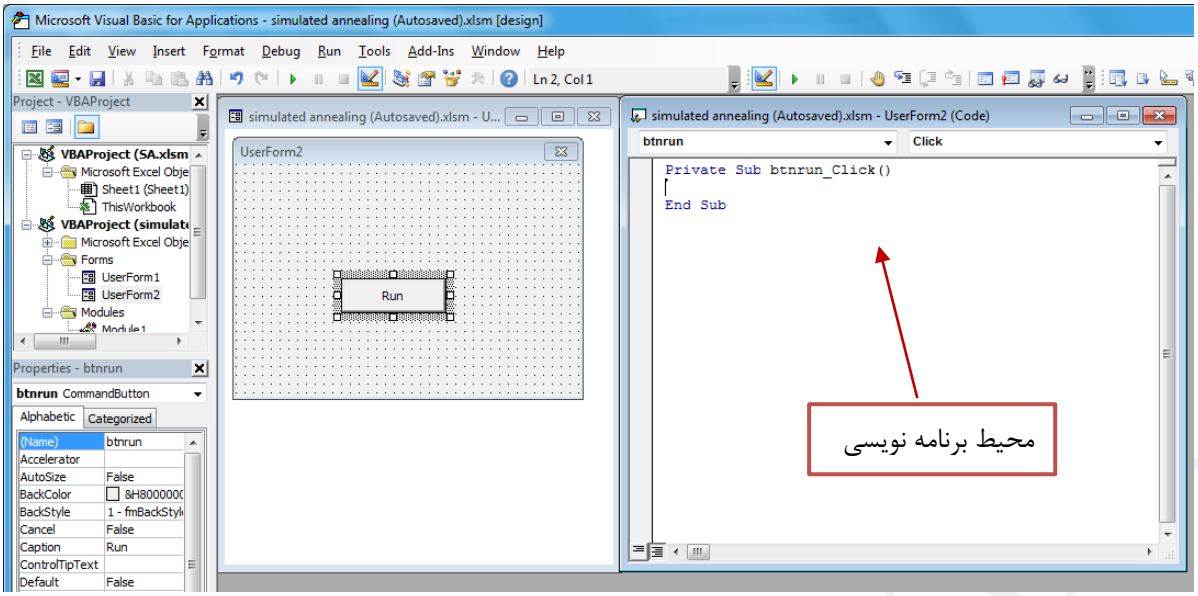


نام دکمه یا همان *Caption* را به *Run* و نام مشخصه دکمه یا *Name* را به *Btnrun* تغییر مـی دهـیم.

نام گذاری اشیا دارای استانداردی است که در صورت علاقه می توانید به کتاب های آموزش برنامـه نویسـی مراجعه کنید.پس از این تغییرات به فرم زیر می رسیم.



برای اضافه کردن برنامه باید روی دکمه *Run* دوبار کلیک کرد و به محیط برنامه نویسی وارد می شویم.



در محیط برنامه نویسی کد زیر را وارد می کنیم.

تعریف متغیرها به صورت زیر انجام می شود.

```
Dim x As Double 'first variable and its current value
Dim y As Double 'second variable and its current value
Dim xp As Double '=x*۱۰۰
Dim yp As Double '=y*۱۰۰
Dim xn As Double '=xnew
Dim yn As Double '=ynew
Dim T۰ As Double 'intial temparture
Dim T As Double 'current temperate
Dim alfa As Double  'decreasing rate
Dim Num As Integer    'base
Dim Ax(۱۰) As Double 'base ۲ value
Dim Ay(۱۰) As Double 'base ۲ value
Dim Abinary(۱۰) As Integer 'base
Dim Fxyc As Double 'current objective function
Dim Fxyn As Double ' new objective function
Dim random As Double 'random variable
```

Dim Bolzman As Double ' Value of Bolzman

Dim counter As Long ' Counter

Dim functionxy(۲۰۰۰۰۰) As Double ' objective functions for all iterations

برای این که مقدار غیر منطقی به پارامترها داده نشود توصیه می شود که برای هر متغیر تعریف شده یک مقدار اولیه در نظر گرفته شود که به صورت زیر می شود.

counter = ۰

xp = ۰

yp = ۰

x = ۰

y = ۰

T۰ = ۰

T = ۰

alfa = ۰

Num = ۱۰

Fxyc = ۰

Fxyn = ۰

random = ۰

Bolzman = ۰

'step ۱ initialization step

T۰ = ۱۰۰۰

T = T۰

```
alfa = ۰.۹۹

For i = ۰ To ۹

    Ax(i) = ۰

Next

For i = ۰ To ۹

    Ay(i) = ۰

Next

 Abinary(۰) = ۱

For i = ۱ To ۹

    Abinary(i) = Abinary(i - ۱) * ۲

Next
```

'step ۲ constructing step

```
For i = ۰ To ۹

    xp = Abinary(i) * Ax(i) + xp
    yp = Abinary(i) * Ay(i) + yp

Next

x = xp / ۱۰۰

y = yp / ۱۰۰

Fxyc = (۱.۵ - x + x * y) * (۱.۵ - x + x * y) + (۲.۲۵ - x + x * y * y) * (۲.۲۵ - x + x * y * y) +

(۲.۶۲۵ - x + x * y * y * y) * (۲.۶۲۵ - x + x * y * y * y(
```

'step ۳ calculating neighbour

```
While T > ۰.۰۰۱

counter = counter + ۱

Dim randstep۳ As Double

Dim First As Boolean
```

```
randstep۳ = Math.Rnd
If randstep۳ < ۰.۵ Then
    First = True
Else
    First = False
End If
Dim random۰_۹ As Double
If First = True Then
random۰_۹ = Fix(Math.Rnd * ۱۰)
Select Case random۰_۹
Case ۰
If Ax(random۰_۹) = ۰ Then
Ax(random۰_۹) = ۱
Else
Ax(random۰_۹) = ۰
End If
Case ۱
If Ax(random۰_۹) = ۰ Then
Ax(random۰_۹) = ۱
Else
Ax(random۰_۹) = ۰
End If
Case ۲
If Ax(random۰_۹) = ۰ Then
Ax(random۰_۹) = ۱
Else
Ax(random۰_۹) = ۰
```

End If

Case ۳

If Ax(random۰_۹) = ۰ Then

Ax(random۰_۹) = ۱

Else

Ax(random۰_۹) = ۰

End If

Case ۴

If Ax(random۰_۹) = ۰ Then

Ax(random۰_۹) = ۱

Else

Ax(random۰_۹) = ۰

End If

Case ۵

If Ax(random۰_۹) = ۰ Then

Ax(random۰_۹) = ۱

Else

Ax(random۰_۹) = ۰

End If

Case ۶

If Ax(random۰_۹) = ۰ Then

Ax(random۰_۹) = ۱

Else

Ax(random۰_۹) = ۰

End If

Case ۷

```
If Ax(random۰_۹) = ۰ Then
Ax(random۰_۹) = ۱
Else
Ax(random۰_۹) = ۰
End If
Case ۸
If Ax(random۰_۹) = ۰ Then
Ax(random۰_۹) = ۱
Else
Ax(random۰_۹) = ۰
End If
Case ۹
If Ax(random۰_۹) = ۰ Then
Ax(random۰_۹) = ۱
Else
Ax(random۰_۹) = ۰
End If
End Select
Else
'Dim random۰_۹ As Double
random۰_۹ = Fix(Math.Rnd * ۱۰(
Select Case random۰_۹
Case ۰
If Ay(random۰_۹) = ۰ Then
Ay(random۰_۹) = ۱
Else
```

```
Ay(random۰_۹) = ۰
End If
Case ۱
If Ay(random۰_۹) = ۰ Then
Ay(random۰_۹) = ۱
Else
Ay(random۰_۹) = ۰
End If
Case ۲
If Ay(random۰_۹) = ۰ Then
Ay(random۰_۹) = ۱
Else
Ay(random۰_۹) = ۰
End If
Case ۳
If Ay(random۰_۹) = ۰ Then
Ay(random۰_۹) = ۱
Else
Ay(random۰_۹) = ۰
End If
Case ۴
If Ay(random۰_۹) = ۰ Then
Ay(random۰_۹) = ۱
Else
Ay(random۰_۹) = ۰
End If
```

```
Case ۵
If Ay(random۰_۹) = ۰ Then
Ay(random۰_۹) = ۱
Else
Ay(random۰_۹) = ۰
End If
Case ۶
If Ay(random۰_۹) = ۰ Then
Ay(random۰_۹) = ۱
Else
Ay(random۰_۹) = ۰
End If
Case ۷
If Ay(random۰_۹) = ۰ Then
Ay(random۰_۹) = ۱
Else
Ay(random۰_۹) = ۰
End If
Case ۸
If Ay(random۰_۹) = ۰ Then
Ay(random۰_۹) = ۱
Else
Ay(random۰_۹) = ۰
End If
Case ۹
If Ay(random۰_۹) = ۰ Then
```

```
Ay(random۰_۹) = ۱
Else
Ay(random۰_۹) = ۰
End If
End Select
End If
```

'step ۴ Boltzmann rule

```
xp = ۰
yp = ۰
For i = ۰ To ۹
    xp = Abinary(i) * Ax(i) + xp
    yp = Abinary(i) * Ay(i) + yp
Next
xn = xp / ۱۰۰
yn = yp / ۱۰۰
Fxyn = (۱.۵ - xn + xn * yn) * (۱.۵ - xn + xn * yn) + (۲.۲۵ - xn + xn * yn * yn) * (۲.۲۵ - xn +
xn * yn * yn) + (۲.۶۲۵ - xn + xn * yn * yn * yn) * (۲.۶۲۵ - xn + xn * yn * yn * yn(
Dim deltaf As Double
deltaf = Fxyn - Fxyc
If deltaf < ۰ Then
x = xn
y = yn
Fxyc = Fxyn
Else
Bolzman = Math.Exp(-deltaf / T(
Dim r As Double
r = Math.Rnd
```

```
If r < Bolzman Then
x = xn
y = yn
Fxyc = Fxyn
End If
End If
```

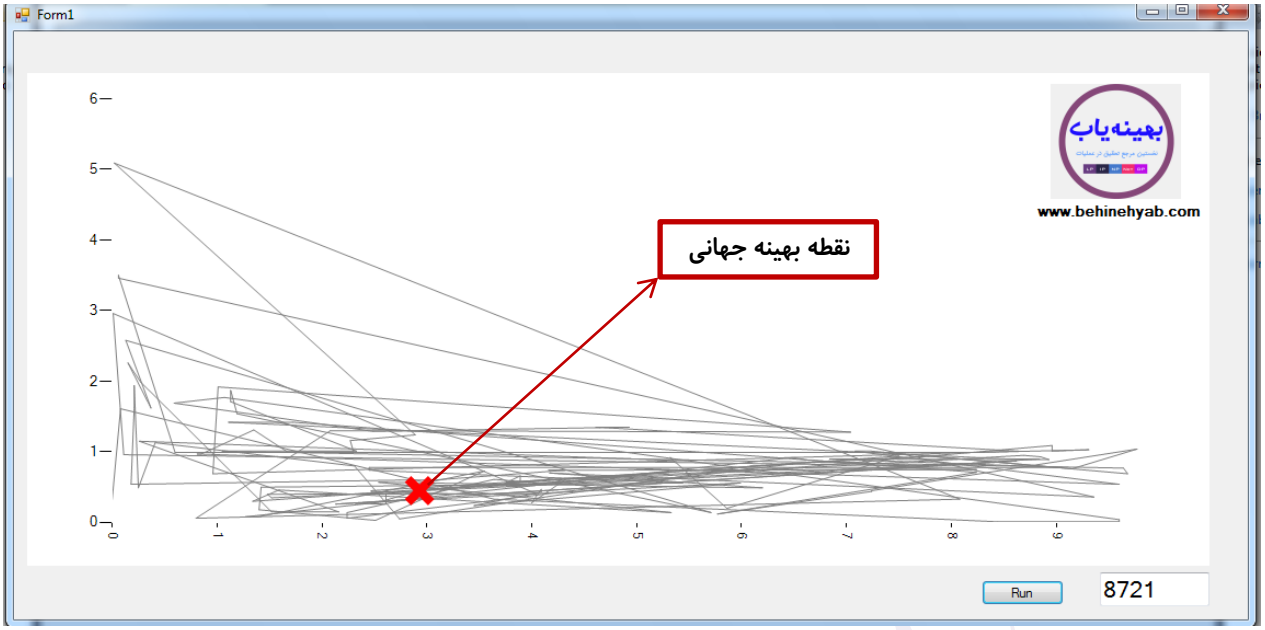
```
'step ۵ Updating tempature

Dim minglobal As Double
 = T * alfa
functionxy(counter) = Fxyc
Wend
```

```
'step ۶ Termination condition
```

زمانی که دما به کمتر از ۰.۰۱ رسید الگوریتم به پایان می رسد'

پس از تکمیل برنامه، می توان با اجرای برنامه جواب بهینه تابع هدف را بدست اورد. پس از اجرای برنامه جواب بهینه به دست امده برابر با $(x^*, y^*) = (0.53, 2.94)$ می شود که به بهینه جهانی $(x^*, y^*) = (0.5, 3)$ نزدیک است. برای این رفتار حرکت الگوریتم را متوجه شویم در شکل زیر می توانید نحوه جستجوی الگوریتم را مشاهده کنید.



همچنین فایل ویدیویی نحوه جستجوی الگوریتم به پیوست این جزوه در اختیار شما قرار می گیـرد. بـا دیدن این ویدیو می توانید به وضوح این نکته را مشاهده کنید که در ابتدا الگوریتم بیشتر به صورت تصادفی به دنبال جستجوی کل فضای امکان پذیر است و دامنه حرکت آن زیاد اسـت. هـر چـه بـه پایـان الگـوریتم نزدیک می شویم الگوریتم به سمت بهبود جواب در حوالی بهترین جواب بدست آمده می پردازد.

برای استفاده دانشجویان و محققان محترم فایل برنامه فوق به صورت فایل متنی به پیوست ایـن جـزوه در اختیار شما قرار می گیرد که می توانید از آن در پروژه های خود استفاده کنید.

برای دریافت بسته‌های آموزشی گروه **بهینه‌یاب** به وب سایت ما به نشانی www.behinehyab.com مراجعه کنید.

در صورت هر گونه سوال از طریق ایمیل به نشانی behinehyab@gmail.com و یا بخش تماس با ما وب سایت گروه **بهینه‌یاب** با ما در تماس باشید.

با تشکر از توجه شما

گروه آموزشی **بهینه‌یاب**