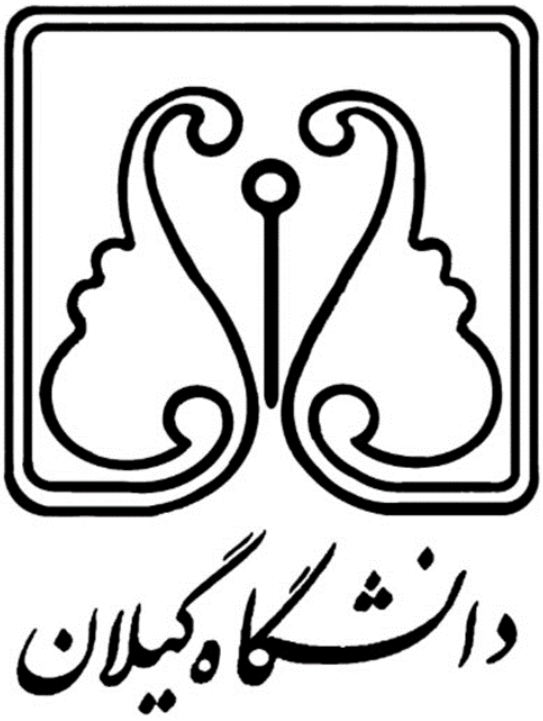

حل مسئله فروشنده دوره گرد با استفاده از الگوریتم خفاش

درس هوش محاسباتی

استاد: جناب آقای مهندس علی تورانی

اعضا:صبا اسکندری ، یاسمن خواجه ، کیمیا کشاورز ، فاطمه امیدمعاف

گروه مهندسی کامپیوتر دانشگاه گیلان

پاییز 98

حل مسئله فروشنده دوره گرد با استفاده از الگوریتم خفاش

موضوع پروژه ی ما پیاده سازی (حل) مساله ی فروشنده ی دوره گرد با استفاده از الگوریتم خفاش می باشد. مساله ی فروشنده دوره گرد (tsp) یکی از مسائل بسیار مهم و پر کاربرد در علوم کامپیوتر و تحقیق در عملیات است .

3 روش کلی برای کدنویسی راه حل های این مساله ارائه شده است که در الگوریتم های مختلفی قابل استفاده است. این 3 روش کلی عبارتند از:

الف) نمایش جواب به صورت رشته گسسته جایگشتی که در الگوریتم جای ژنتیک (GA) و شبیه سازی تبرید (SA) و جست و جوی ممنوعه (TS) و جست و جوی همسایگی متغیر (VNS) و بهینه سازی کلونی مورچگان (ACO) و جست و جوی هارمونی (HS) قابل استفاده است.

ب) نمایش جواب به صورت کلیدهای تصادفی (randomkey) که در الگوریتم هایی مانند ژنتیک (GA) و بهینه سازی ازدحام ذرات(PSO) واستراتژی های تکاملی یا برنامه نویسی تکاملی (ES) و (EP) و سایر الگوریتم های بهینه سازی پیوسته قابل استفاده است.

پ)نمایش جواب به شکل ماتریس های شبیه فرومون بوده که توسط تحلیل الگوریتم های روش (ب) قابل استفاده است.

الگوریتم خفاش نیز یکی از الگوریتم های فرا تکاملی می باشد که نوعی از الگوریتم های تصادفی برای یافتن پاسخ بهینه هستند.

در ادامه توضیحات بیشتری در مورد مساله ی فروشنده دوره گرد و الگوریتم خفاش بیان خواهیم کرد.

مسئله فروشنده دورگرد

در مسئله ی فروشنده ی دوره گرد ، تعداد معینی شهر (گره) داریم ، که هزینه ی رفتن مستقیم مابین این شهر ها معین است. حال فروشنده یی دوره گرد قصد دارد اجناس خود را در تعدادی شهر معین و با مختصات مکانی ثابت به فروش برساند. او باید از محل خود سفر را شروع و از تمام شهر ها دقیقا یکبار عبور کند و به محل اولیه اش بازگردد. در این دوره ها ، فروشنده باید شرایط دور همیلتونی ، به شرح زیر را در نظر بگیرید :

1.هر شهر (گره) دقیقا 1 بار رویت شود

2.پایان و شروع دور باید یک گره ی یکسان باشد (دور باید بسته باشد).

الگوریتم خفاش

الگوریتم خفاش یکی از الگوریتم های فراابتکاری الهام گرفته از طبیعت است که در سال ۲۰۱۰ توسط آقای یانگ معرفی گردید. این الگوریتم بر اساس اصول زندگی خفاش ها طراحی شده است. خفاش ها تنها پستانداران با بال هستند که برای شکار طعمه از انعکاس صدا استفاده می کنند. تاکنون الگوریتم BA برای مسائل بهینه سازی دودویی و برای مسائل بهینه سازی چند هدفی به کار گرفته شده است. با این وجود، بسیاری از مسائل بهینه سازی گسسته وجود دارند که قابل حل با الگوریتم های فراابتکاری موجود نمی باشند. لذا در این مقاله هدف این است که نوع صحیح الگوریتم خفاش ارائه گردد. برای اطلاع از پیشرفت های دیگر الگوریتم BA خواننده به ارجاع داده می شود.

الگوریتم بهینه سازی خفاش الهامی از خصوصیات ردیابی خفاش های کوچک در جستجوی شکار می باشد. به طوری که خفاش های کوچک می توانند در تاریکی مطلق با انتشار صدا و دریافت آن به شکار طعمه های خود بپردازند. برای توسعه این الگوریتم از سه قانون آرمانی زیر استفاده می شود:

همه خفاش ها از انعکاس صدا برای تشخیص فاصله استفاده می کنند و تفاوت بین مواد غذایی و موانع پیشرو را می دانند .

پرواز خفاش ها به طور تصادفی با سرعت vi در مکان xi با فرکانس ثابت fmin و طول موج مختلف λ و بلندی صوتA0 به منظور شکار طعمه صورت می گیرد. همچنین آنها می توانند به طور خودکار امواج پخش شده و نرخ پالس های ارسالی خود را (r ∈ [۰ ,۱] با توجه به نزدیکی شکارشان تنظیم کنند .

با توجه به اینکه ممکن است بلندی صدا در بسیاری از جهات مختلف متفاوت باشد لذا فرض می کنیم که بلندی صدا ازR0 (بیشترین مقدار) تاRmin (کمترین مقدار) متغیر می باشد.

طبق قوانین بیان شده، مکان x_i^t و سرعت v_i^t برای هر خفاش مجازی iام در تکرار t و همچنین فرکانس fi به صورت زیر محاسبه می گردد:

$$f_i = f_{min} + (f_{max} - f_{min})\beta \quad (۱)$$

$$v_i^t = v_i^{t-۱} + (x_i^t - x^*) \quad (۲)$$

$$x_i^t = x_i^{t-۱} + v_i^t \quad (۳)$$

که در آن [۰ ,۱ β] ∈ یک بردار تصادفی با توزیع یکنواخت می باشد و x * بهترین مکان فعلی است که در هر تکرار پس از مقایسه با موقعیت خفاش های مجازی انتخاب می شود. معمولا فرکانس f را با fmin= 0 و fmax = 100 در نظر می گیرند. در هر تکرار، در جستجوی محلی یکی از جواب ها به عنوان بهترین جواب ها انتخاب شده و موقعیت جدید هر خفاش به طور محلی با گام تصادفی به صورت زیر به روز می شود: (4)

$$x_{new} = x_{old} + \epsilon A^t.$$

که در آن [۱ ,۱−] ∈ ϵ یک عدد تصادفی بوده و $A^t =< A_i^t >$ میانگین بلندی صدای خفاش ها در تکرار t می باشد. همچنین بلندی صدای Ai و نرخ پالس ارسالی r در هر تکرار به صورت زیر به روز می شود: (5)

$$A_i^{t+1} = \alpha A_i^t, \qquad r_i^{t+1} = r_i^\circ[1 - exp(-\gamma t)]$$

که در آن α و γ مقدار ثابت می باشند و برای هر ۱ > α >۰ و ۰ < r ،وقتی t → ∞ داریم :

$$A_i^{t+1} \to 0 \text{ و } r_i^{t+1} \to r_i^\circ$$

با توجه به مطالب ارائه شده در بالا، الگوریتم خفاش به صورت الگوریتم ۱ خلاصه می شود:

۱ .جمعیت اولیه خفاش ها xi ، i=1,2,3,..n را ایجاد کنید.

۲. به ازای i= 1,...,n سرعت vi ،فرکانس fi در xi ،نرخ پالس ri و بلندی صدای Ai را ایجاد کنید.

۳. قرار دهید t=1

۴. موقعیت ها را مقایسه کرده و جواب بهینه را بیابید.

۵. موقعیت های جدید موقتی با تنظیم فرکانس و به روز کردن سرعت همه خفاش ها ایجاد کنید(معادلات (۱)-(۳)).

۶. اگر ri > rand ،آنگاه یک جواب در میان بهترین جواب ها با گام تصادفی کرده و با استفاده از معادله (4)یک جواب محلی در اطراف بهترین جواب انتخاب شده ایجاد کنید.

۷. یک جواب جدید با پرواز تصادفی تولید کنید.

۸. اگر rand < Ai و f(xi) < f(xi*)،آنگاه جواب های جدید را بپذیرید و طبق معادله (۵)، riرا افزایش و Ai را کاهش دهید.

۹. خفاش ها را مرتب کرده و بهترین جواب x*را پیدا کنید.

۱۰. اگر t به ماکزیمم مقدار خود رسید الگوریتم را متوقف کنید در غیر این صورت قرار دهید t = t+ ۱ و به ۴ بروید.

توضیح عملکرد پروژه

برنامه در تابعی به نام objective که در تابع اصلی فراخوانی شده ابتدا از کاربر درخواست میکند که تعداد cities را وارد نماید سپس 2 تابع
enter_path_matrix(size,path_matrix);
print_path_matrix(size,path_matrix); برای گرفتن ورودی و نمایش ماتریس را فراخوانی میکند.

سپس با فراخوانی تابع random_path_generator(size,pattern); مسیر های تصادفی را پیدا کرده و با استفاده از city_num = new Random().nextInt() % size + 1; به صورت رندوم city num را محاسبه میکنیم و سپس هزینه هر random pathرا محاسبه میکنیم.

این خط کد freq[i] = (float)cost[i]/VAL;

در واقع فرکانس را برای هر i محاسبه میکند.

BatAlgorithm(size,10,freq,random_paths,cost,path_matrix,idx);

تابع بالا تابع فراخوانی الگوریتم bat است.

Algorithm bat in code : در این تابع ابتدا با مقایسه ی فرکانس ها ، فرکانس مینیمم را پیدا میکند و تابع sortرا فراخوانی کرده و با استفاده از تابع random path بهترین راه حل را می یابد و در آخر بهترین راه ها پیدا شده با بهترین هزینه را الگوریتم شناسایی میکند.