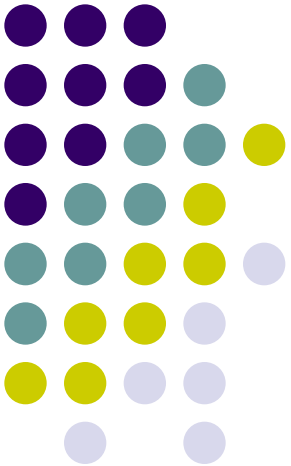
الگوریتم های ژنتیک

Instructor : Saeed Shiry



📖 Mitchell Ch. 9

الگوریتم ژنتیک

- الگوریتم ژنتیک روش یادگیری بر پایه تکامل بیولوژیک است.
- این روش در سال 1970 توسط John Holland معرفی گردید
- این روشها با نام Evolutionary Algorithms نیز خوانده میشوند.

ایده کلی

- یک GA برای حل یک مسئله مجموعه بسیار بزرگی از راه حلهای ممکن را تولید میکند.
- هر یک از این راه حلها با استفاده از یک " تابع تناسب "مورد ارزیابی قرار میگیرد.
- آنگاه تعدادی از بهترین راه حلها باعث تولید راه حلهای جدیدی میشوند . که اینکار باعث تکامل راه حلها میگردد.
- بدین ترتیب فضای جستجو در جهتی تکامل پیدا میکند که به راه حل مطلوب برسد
- در صورت انتخاب صحیح پارامترها، این روش میتواند بسیار موثر عمل نماید.

فضای فرضیه

- الگوریتم ژنتیک بجای جستجوی فرضیه های general-to specificو یا simple to complex فرضیه ها ی جدید را با تغییر و ترکیب متوالی اجزا بهترین فرضیه های موجود بدست میاورد.
- در هرمرحله مجموعه ای از فرضیه ها که جمعیت (population)نامیده میشوند از طریق جایگزینی بخشی از جمعیت فعلی با فرزندانی که از بهترین فرضیه های موجود حاصل شده اند بدست میآید.

ویژگیها

الگوریتم های ژنتیک در مسائلی که فضای جستجوی بزرگی داشته باشند میتواند بکار گرفته شود.

- همچنین در مسایلی با فضای فرضیه پیچیده که تاثیر اجزا آن در فرضیه کلی ناشناخته باشند میتوان از GA برای جستجو استفاده نمود.
- برایdiscrete optimization بسیار مورد استفاده قرار میگیرد.
- الگوریتم های ژنتیک را میتوان براحتی بصورت موازی اجرا نمود از اینرو میتوان کامپیوترهای ارزان قیمت تری را بصورت موازی مورد استفاده قرار داد.
- امکان به تله افتادن این الگوریتم در مینیمم محلی کمتر از سایر روشهاست.
- از لحاظ محاسباتی پرهزینه هستند.
- تضمینی برای رسیدن به جواب بهینه وجود ندارد.

Parallelization of Genetic Programming

- در سال 1999 شرکت Genetic Programming Inc. یک کامپیوتر موازی با 1000 گره هر یک شامل کامپیوتر های P2, 350 MHZبرای پیاده سازی روش های ژنتیک را مورد استفاده قرار داد.

کاربر دها

- کاربرد الگوریتم های ژنتیک بسیار زیاد میباشد

- optimization,
- automatic programming,
- machine learning,
- economics,
- operations research,
- ecology,
- studies of evolution and learning, and
- social systems

زیر شاخه های EA

روش های EA به دو نوع مرتبط به هم ولی مجزا دسته بندی میشوند:

۱. **Genetic Algorithms (GAs)**

در این روش راه حل یک مسئله بصورت یک bit string نشان داده میشود .

۲. **Genetic Programming (GP)**

این روش به تولید expression trees که در زبانهای برنامه نویسی مثل lisp مورد استفاده هستند میپردازد بدین ترتیب میتوان برنامه هائی ساخت که قابل اجرا باشند.

الگوریتم های ژنتیک

- روش متداول پیاده سازی الگوریتم ژنتیک بدین ترتیب است که:
- مجموعه ای از فرضیه ها که population نامیده میشود تولید وبطور متناوب با فرضیه های جدیدی جایگزین میگردد.
- در هر بار تکرارتمامی فرضیه ها با استفاده از یک تابع تناسب یا Fitnessمورد ارزیابی قرار داده میشوند .آنگاه تعدادی از بهترین فرضیه ها با استفاده از یک تابع احتمال انتخاب شده و جمعیت جدید را تشکیل میدهند.
- تعدادی از این فرضیه های انتخاب شده به همان صورت مورد استفاده واقع شده و مابقی با استفاده از اپراتورهای ژنتیکی نظیر Crossover وMutation برای تولید فرزندان بکار میروند.

پارامترهای GA

یک الگوریتم GA دارای پارامترهای زیر است:

GA(Fitness,Fitness_threshold,p,r,m)

- Fitness :تابعی برای ارزیابی یک فرضیه که مقداری عددی به هر فرضیه نسبت میدهد
- Fitness_threshold :مقدار آستانه که شرط پایان را معین میکند
- p :تعداد فرضیه هائی که باید در جمعیت در نظر گرفته شوند
- r:در صدی از جمعیت که در هر مرحله توسط الگوریتم crossover جایگزین میشوند
- m:نرخ mutation

الگورتیم

- Initialize :جمعیت را با تعداد p فرضیه بطور تصادفی مقدار دهی اولیه کنید.
- Evaluate :برای هر فرضیه h در p مقدار تابع Fitness(h)را محاسبه نمائید.
- تا زمانیکه $[\max_h \text{Fitness}(h)] < \text{Fitness\_threshold}$ یک جمعیت جدید ایجاد کنید.
- فرضیه ای که دارای بیشترین مقدار Fitness است را برگردانید.

نحوه ایجاد جمعیت جدید

مراحل ایجاد یک جمعیت جدید بصورت زیر است:

۱. select :تعداد (1-r)pفرضیه از میان P انتخاب و به P_sاضافه کنید .احتمال انتخاب یک فرضیه h_iاز میان Pعبارت است از:

$$P(hi) = Fitness (hi) / \Sigma j\ Fitness (hj)$$

هر چه تناسب فرضیه ای بیشتر باشد احتمال انتخاب آن بیشتر است .این احتمال همچنین با مقدار تناسب فرضیه های دیگر نسبت عکس دارد.

۲. Crossover :با استفاده از احتمال بدست آمده توسط رابطه فوق، تعداد (rp)/2زوج فرضیه از میان P انتخاب و با استفاده از اپراتور Crossoverدو فرزند از آنان ایجاد کنید .فرزندان را به P_sاضافه کنید.

۳. Mutate :تعداد m درصد از اعضا P_sرا با احتمال یکنواخت انتخاب و یک بیت از هر یک آنها را بصورت تصادفی معکوس کنید

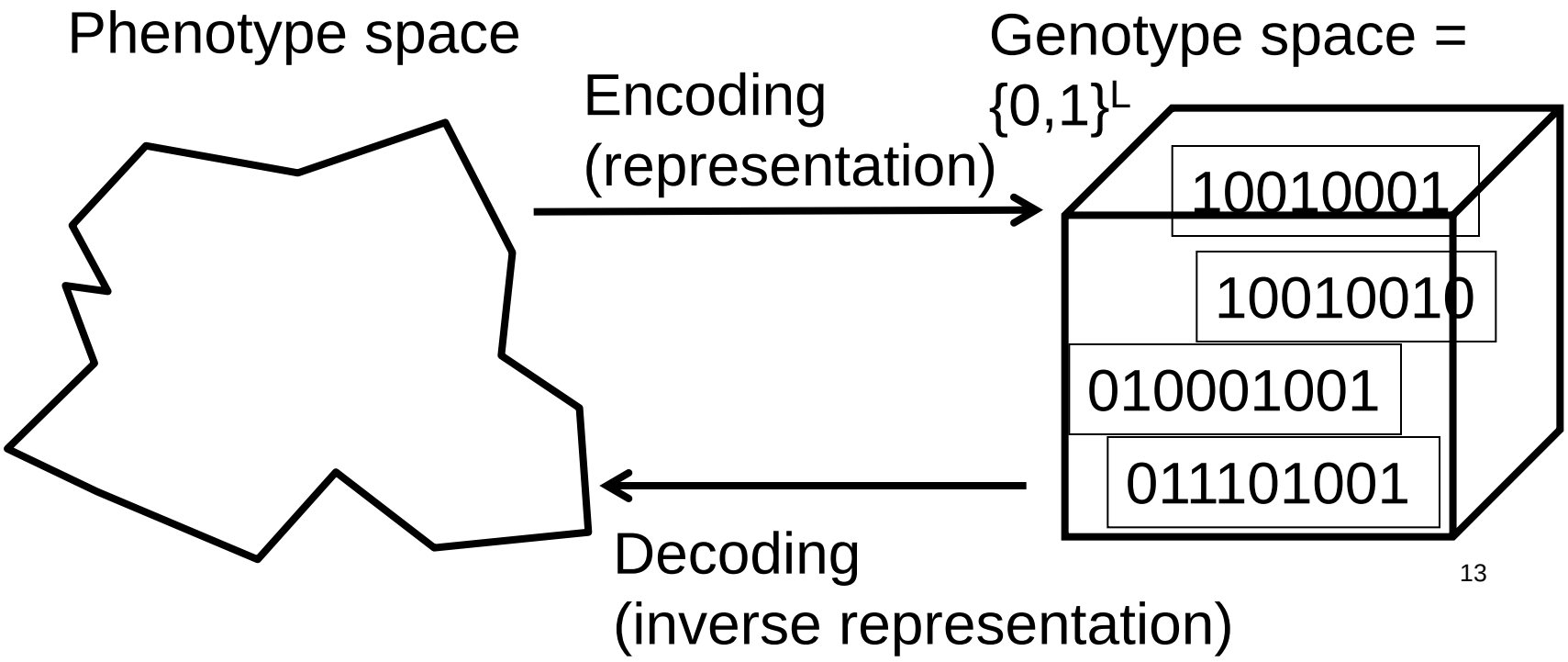
۴. Update: $P \leftarrow P_s$

۵. برای هر فرضیه h در P مقدار تابع Fitness را محاسبه کنید

نمایش فرضیه ها

در الگوریتم ژنتیک معمولا فرضیه ها بصورت رشته ای از بیت ها نشان داده میشوند تا اعمال اپراتورهای ژنتیکی برروی آنها ساده تر باشد.

- Phenotype :به مقادیر یا راه حلهای واقعی گفته میشود.
- Genotype :به مقادیر انکد شده یا کروموزم ها گفته میشود که مورد استفاده GA قرار میگیرند.
- باید راهی برای تبدیل این دو نحوه نمایش به یکدیگر بدست آورده شود.



مثال :نمایش قوانین If-then rules

- برای نمایش مقادیر یک ویژگی نظیر Outlook که دارای سه مقدار Sunny, Overcast ,Rain است میتوان از رشته ای با طول 3 بیت استفاده نمود

100 -> *Outlook = Sunny*

011-> *Outlook = Overcast* $\vee$ *Rain*

برای نمایش تر کیب ویژگی ها رشته بیت های هر یک را پشت سر هم قرار میدهیم:

Outlook	*Wind*
011	10

به همین ترتیب کل یک قانون if- then را میتوان با پشت سر هم قرار دادن بیت های قسمت های شرط و نتیجه ایجاد نمود:

IF *Wind = Strong* THEN *PlayTennis = No*

Outlook	*Wind*	*PlayTennis*
111	10	0

$\Rightarrow$ bit string: 111100

نمایش فرضیه ها :ملاحظات

ممکن است ترکیب بعضی از بیت ها منجر به فرضیه های بی معنی گردد .برای پرهیز از چنین وضعیتی:

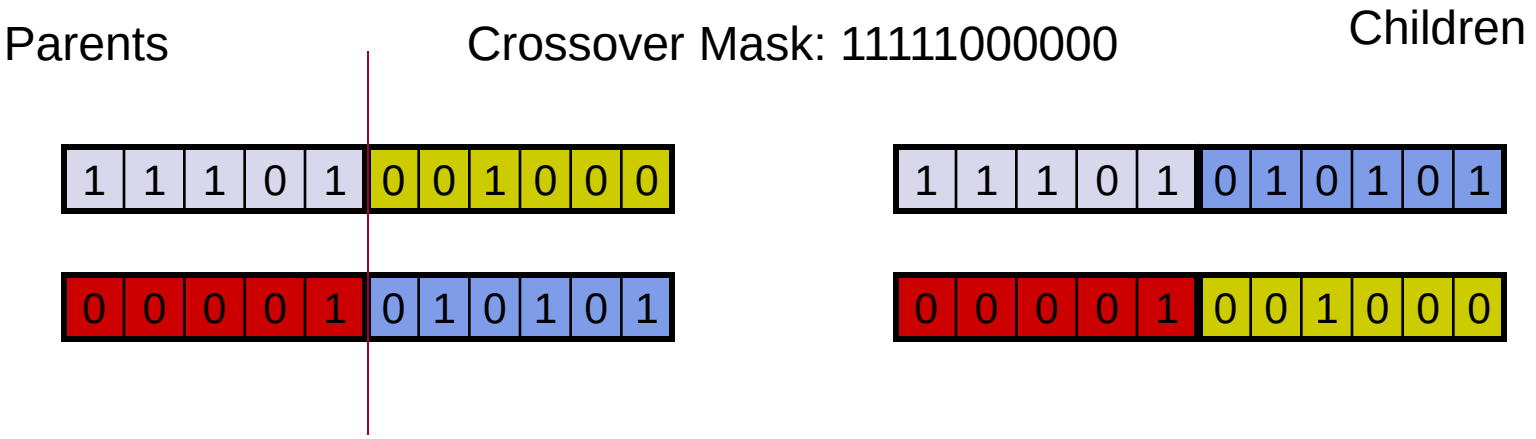
- میتوان از روش انکدینگ دیگری استفاده نمود.
- اپراتورهای ژنتیکی را طوری تعیین نمود که چنین حالتهائی را حذف نمایند
- میتوان به این فرضیه ها مقدار fitness خیلی کمی نسبت داد.

اپراتورهای ژنتیکی: Crossover

- اپراتور Crossover با استفاده از دو رشته والد دو رشته فرزند بوجود میآورد .
- برای اینکار قسمتی از بیتهای والدین در بیتهای فرزندان کپی میشود .
- انتخاب بیت هائی که باید از هر یک از والدین کپی شوند به روشهای مختلف انجام میشود
  - single-point crossover
  - Two-point crossover
  - Uniform crossover
- برای تعیین محل بیتهای کپی شونده از یک رشته به نام Crossover استفاده میشود.Mask

Single-point crossover

- یک نقطه تصادفی در طول رشته انتخاب میشود .
- والدین در این نقطه به دوقسمت میشوند.
- هر فرزند با انتخاب تکه اول از یکی از والدین و تکه دوم از والد دیگر بوجود میاید.

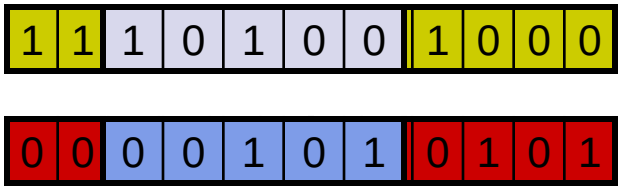


روشهای دیگرCrossover

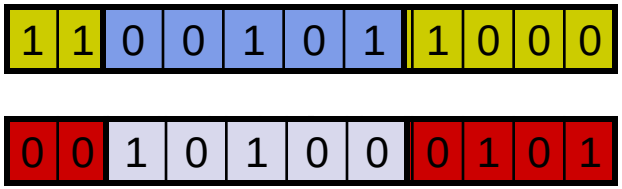
- Two-point crossover

Crossover Mask: 00111110000

Parents



Children



- Uniform crossover

Parents

Crossover Mask: 10011010011

Children

بیتها بصورت یکنواخت از والدین انتخاب میشوند

اپراتورهای ژنتیکی : Mutation

- اپراتور mutation برای بوجود آوردن فرزند فقط از یک والد استفاده میکند .اینکار با انجام تغییرات کوچکی در رشته اولیه بوقوع میپیوندد.
- با استفاده از یک توزیع یکنواخت یک بیت بصورت تصادفی انتخاب و مقدار آن تغییر پیدا میکند.
- معمولا mutation بعد از انجام crossover اعمال میشود.

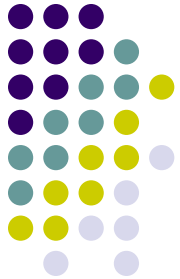
Parent

1	1	1	0	1	0	0	1	0	0	0

Child

1	1	1	0	1	1	0	1	0	0	0

Crossover OR mutation?

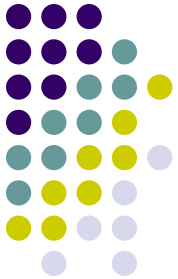


- این سوال ها سالها مطرح بوده است:

کدامیک بهتر است؟ کدامیک لازم است؟ کدامیک اصلی است؟

- پاسخی که تاکنون بیشتر از بقیه پاسخها مورد قبول بوده:
  - بستگی به صورت مسئله دارد
  - در حالت کلی بهتر است از هر دو استفاده شود
  - هر کدام نقش مخصوص خود را دارد
  - میتوان الگوریتمی داشت که فقط از mutation استفاده کند ولی الگوریتمی که فقط از crossoverاستفاده کند کار نخواهد کرد

Crossover OR mutation?

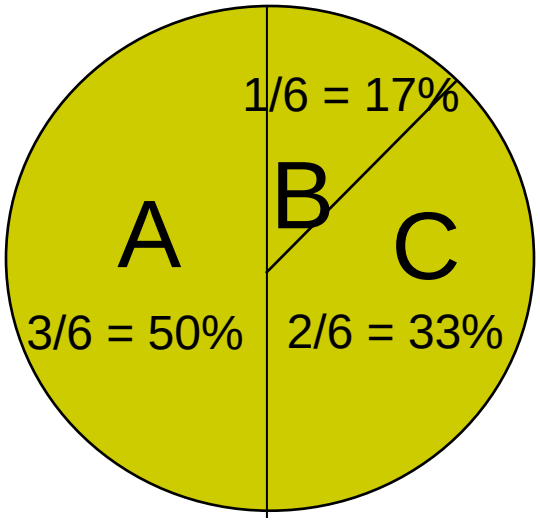


- Crossoverخاصیت جستجوگرانه و یا explorative دارد .میتواندبا انجام پرشهای بزرگ به محل هائی دربین والدین رفته و نواحی جدیدی را کشف نماید .
- Mutationخاصیت گسترشی و یا exploitive دارد .میتواند با انجام تغییرات کوچک تصادفی به نواحی کشف شده وسعت ببخشد.
- Crossove اطلاعات والدین را ترکیب میکند درحالیکه mutation میتواند اطلاعات جدیدی اضافه نماید .
- برای رسیدن به یک پاسخ بهینه یک خوش شانسی در mutation لازم است.

تابع تناسب

- تابع fitness معیاری برای رتبه بندی فرضیه هاست که کمک میکند تا فرضیه های برتر برای نسل بعدی جمعیت انتخاب شوند .نحوه انتخاب این تابع بسته به کاربر مورد نظر دارد:
- classification :در این نوع مسایل تابع تناسب معمولا برابر است با دقت قانون در دسته بندی مثالهای آموزشی.

انتخاب فرضیه ها



fitness(A) = 3

fitness(B) = 1

fitness(C) = 2

- Roulette Wheel selection

در روش معرفی شده در الگوریتم ساده GA احتمال انتخاب یک فرضیه برای استفاده در جمعیت بعدی بستگی به نسبت fitness آن به fitness بقیه اعضا دارد .این روش Roulette Wheel selectionنامیده میشود.

$$P(h_i) = Fitness(h_i) / \Sigma_j Fitness(h_j)$$

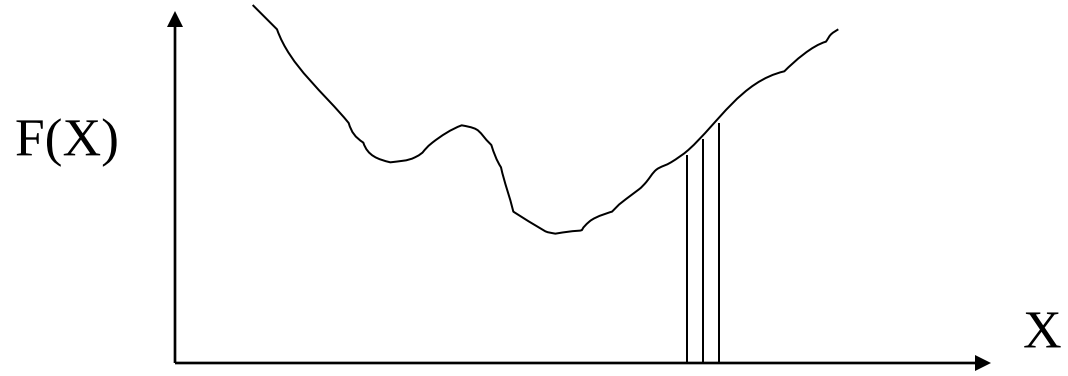
- روشهای دیگر:
  - tournament selection
  - rank selection

نحوه جستجو در فضای فرضیه

- روش جستجوی GA با روشهای دیگر مثل شبکه های عصبی تفاوت دارد:
- در شبکه عصبی روش Gradient descent بصورت هموار از فرضیه ای به فرضیه مشابه دیگری حرکت میکند در حالیکه GA ممکن است بصورت ناگهانی فرضیه والد را با فرزندی جایگزین نماید که تفاوت اساسی با والد آن داشته باشد.از اینرو احتمال گیر افتادن GA در مینیمم محلی کاهش می یابد.
- با این وجود GA با مشکل دیگری روبروست که crowding نامیده میشود.

Crowding

- crowdingپدیده ای است که در آن عضوی که سازگاری بسیاربیشتری از بقیه افراد جمعیت دارد بطور مرتب تولید نسل کرده و با تولید اعضای مشابه درصد عمده ای از جمعیت را اشغال میکند.
- اینکار باعث کاهش پراکندگی جمعیت شده و سرعت GA را کم میکند.



راه حل رفع مشکل Crowding

- استفاده از ranking برای انتخاب نمونه ها :با اختصاص رتبه به فرضیه ای که بسیار بهتر از بقیه عمل میکند مقدار این برتری نشان داده نخواهد شد.
- Fitness sharing : مقدار Fitness یک عضودر صورتیکه اعضا مشابهی در جمعیت وجود داشته باشند کاهش می یابد.

چرا GA کار میکند؟

- سئوالی که ممکن است برای تازه واردین به روشهای ژنتیکی ایجاد شود این است که آیا این روش واقعا میتواند کار مفیدی انجام دهد؟

ارزیابی جمعیت و قضیه Schema

- آیا میتوان تکامل در جمعیت در طی زمان را بصورت ریاضی مدل نمود؟
- قضیه Schema میتواند مشخصه پدیده تکامل در GA را بیان نماید.
- یک Schema مجموعه ای از رشته بیت ها را توصیف میکند.یک Schemaهر رشته ای از 0 و1و *است .مثل 10*0 که * حالت dont careاست.
- یک رشته بیت را میتوان نماینده هر یک ازSchema های متفاوتی دانست که با آن تطابق دارند .مثلا 0010 را میتوان نماینده 24 Schemaمختلف دانست : ****,0*10,**00وغیره

قضیه Schema

- قضیه Schema بیان میکند که چگونه یک Schema در طول زمان در جمعیت تکامل پیدا خواهد کرد.
- فرض کنید که در لحظه t تعداد نمونه هائی که نماینده یک Schemaمثل s هستند برابر با m(s,t) باشد .این قضیه مقدار مورد انتظار m(s,t+1) را مشخص میکند.
- قبلا دیدیم که احتمال انتخاب یک فرضیه برابر بود با:

$$P(h_i) = Fitness(h_i) / \Sigma_j Fitness(h_j)$$

- این مقدار احتمال را میتوان بصورت زیر نیز نشان داد:

$$P(h_i) = f(h_i) / n f'(t_i)$$

مقدار متوسط fitness برای تمامی فرضیه ها

قضیه Schema

- اگر عضوی از این جمعیت انتخاب شود احتمال اینکه این عضو نماینده S باشد برابر است با:

$$p(h \in s) = \sum_{h \in s \cap p_s} \frac{f(h)}{n\,\bar{f}(t)} = \frac{u(s,t)}{n\,\bar{f}(t)} m(s,t)$$

- که در آن مقدار u(s,t) برابر است با مقدار میانگین fitness اعضای s

$$u(s,t) = \frac{\sum_{h \in s \cap p_s} f(h)}{m(s,t)}$$

- از اینرو مقدار مورد انتظار برای نمونه هائی از s که از n مرحله انتخاب مستقل حاصل خواهند شد برابر است با:

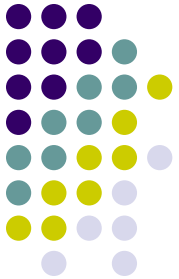
$$E[m(s,t+1)] = \frac{u(s,t)}{\bar{f}(t)} m(s,t)$$

قضیه Schema

- رابطه فوق به این معناست که تعداد Schema های مورد انتظار در لحظه t+1 متناسب با مقدار میانگین u(s,t) بوده و با مقدار fitness سایر اعضا نسبت عکس دارد.
- برای بدست آوردن رابطه فوق فقط اثر مرحله انتخاب نمونه ها در نظر گرفته شده است .با در نظر گرفتن اثر crossover و Mutation به رابطه زیر خواهیم رسید:

$$E[m(s,t+1)] \geq \frac{u(s,t)}{\bar{f}(t)} m(s,t)\left(1-p_c \frac{d(s)}{l-1}\right)\left(1-p_m\right)^{o(s)}$$

Schema Theorem



- Theorem

$$E[m(s,t+1)] \geq \frac{\hat{u}(s,t)}{\bar{f}(t)} \cdot m(s,t) \cdot \left(1 - p_c \frac{d_s}{l-1}\right) \cdot (1-p_m)^{o(s)}$$

  - $m(s, t)$ ≡ number of instances of schema s in population at time t
  - $\bar{f}(t)$ ≡ average fitness of population at time t
  - $\hat{u}(s,t)$ ≡ average fitness of instances of schema s at time t
  - p_c ≡ probability of single point crossover operator
  - p_m ≡ probability of mutation operator
  - l ≡ length of individual bit strings
  - $o(s)$ ≡ number of defined (non "*") bits in s
  - $d(s)$ ≡ distance between rightmost, leftmost defined bits in s
- Intuitive Meaning
  - "The expected number of instances of a schema in the population tends toward its relative fitness"

خلاصه

- یک Schema اطلاعات مفید و امید بخش موجود در جمعیت را کد میکند.
- از آنجائیکه همواره رشته هائی که سازگارترند شانس بیشتری برای انتخاب شدن دارند، بتدریج مثالهای بیشتری به بهترین Schema ها اختصاص می یابند.
- عمل crossover باعث قطع رشته ها در نقاط تصادفی میشود .با این وجود در صورتیکه اینکار باعث قطع Schema نشده باشد آنرا تغییر نخواهد داد.در حالت کلی Schema های با طول کوتاه کمتر تغییر میکنند.
- عمل mutaion در حالت کلی باعث تغییرات موثر در Schema نمیگردد.

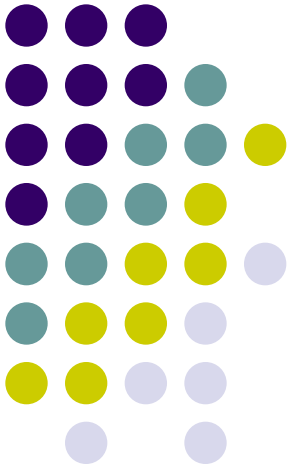
Highly-fit, short-defining-length schema (called *building blocks*) are propagated generation to generation by giving exponentially increasing samples to the observed best

تفاوت GA با سایر روشهای جستجو

- GAبجای کد کردن پارامترها مجموعه آنها را کد میکند
- GAبجای جستجو برای یک نقطه بدنبال جمعیتی از نقاط میگردد.
- GAبجای استفاده از مشتق و یا سایر اطلاعات کمکی مستقیما ازاطلاعات موجود در نتیجه بهره میگیرد.
- GAبجای قوانین قطعی از قوانین احتمال برای تغییر استفاده میکند.

مثالی از کاربرد الگوریتم ژنتیک

بهینه‌سازی چینش حروف فارسی بر روی
صفحه کلید با استفاده از الگوریتم‌های ژنتیکی



مقدمه

- بدست آوردن چینش بهینه حروف فارسی بر روی صفحه‌کلید در درازمدت برای کسانی که با تایپ کردن متون فارسی درگیر هستند, بسیار مفید خواهد بود.
- یک الگوریتم تکاملی می‌تواند با توجه به یک تابع تناسب که میزان راحتی تایپ کردن متون فارسی را برای یک چینش ارائه می‌دهد, در فضای چینش‌های حروف فارسی بر روی صفحه‌کلید جستجو کرده و چینش بهینه را بدست آورد.

چینش کنونی حروف فارسی بر روی صفحه‌کلید

مساله

- در این مساله هندسه صفحه‌کلید ثابت است و ما می‌خواهیم که تعداد ۳۳ نشانه که متشکل از ۳۲ حرف الفبای فارسی بعلاوه حرف همزه "ء" است را بر روی سه ردیف صفحه‌کلید که به ترتیب دارای ۱۲, ۱۱, و ۱۰ کلید هستند, قرار دهیم.
- هدف این مساله بدست آوردن چینشی از این نشانه‌ها بر روی این کلیدها است, به طوری که این چینش طوری باشد که کاربر هنگام استفاده از صفحه کلید برای تایپ حروف فارسی, احساس راحتی بیشتری نسبت به کار با بقیه چینش‌ها داشته باشد.

الگوریتم ژنتیک

- برای حل مساله از یک الگوریتم ژنتیک استفاده شده است.
- تابع تناسب موجود در این الگوریتم ژنتیک, میزان راحتی یا سختی استفاده از یک چینش را محاسبه می‌کند.
- در هر نسل, عملگرهای ژنتیکی بر روی جمعیت موجود که چینش‌های مختلفی از حروف فارسی بر روی صفحه‌کلید هستند, اعمال می‌شوند و جامعه به سمتی سوق داده می‌شود که مقدار تابع تناسب به ازای اعضای آن به کمینه مقدار خود برسند.
- میزان تناسب هر عضو از جامعه که در واقع یک چینش حروف فارسی بر روی صفحه‌کلید هستند, با اعمال تابع تناسب بر متنی که از مطالب چند سایت خبری فارسی زبان تهیه شده است, به دست می‌آید.

جمعیت

- اعضای جمعیت جایگشت‌های مختلف حروف فارسی روی صفحه‌کلید هستند. هر عضو جمعیت را می‌توان به صورت برداری از حروف فارسی در نظر گرفت که هر اندیس آن متناظر با یک کلید از صفحه کلید است.
- مثلاً هر بردار با طول۳۳ که شامل حروف فارسی بعلاوه حرف همزه "ء" باشد را میتوان به عنوان یک کروموزوم (یک عضو از جمعیت) در نظر گرفت که حرف iام از این بردار, متناظر با کلیدی از صفحه‌کلید است که برچسب شمارهٔ i بر روی آن زده شده است.

1	2	3	4	5	6	. . .	33
ض	ص	ث	ق	ف	غ	. . .	ب

تعداد چینش‌های مختلف !۳۳

تابع تناسب

- تعیین راحتی و سختی کار کردن با چینش حروف بر روی صفحه‌کلید یک مساله پیچیده ارگونومیک است.
- نورمن و رومل‌هارت چهار هدف را برای طراحی کارای یک صفحه‌کلید ارائه کرده‌اند:
  ۱. برابری کاری که دو دست انجام می‌دهند؛
  ۲. بیشترین تایپ حروف به صورت متناوب با دو دست؛
  ۳. کمترین تکرار تایپ دو حرف متوالی با یک انگشت؛ و
  ۴. بیشترین تایپ حروف بر روی کلیدهای پایه‌ای (کلیدهای ردیف وسط).

تابع تناسب

- . برای دو هدف اول می‌توان فاکتور اندازه‌گیری زیر را معرفی‌کرد:
- C_{hand}: هزینه مربوط به استفاده از یک دست برای تایپ کردن دو حرف پشت سر هم.
- برای هدف سوم, فاکتور اندازه‌گیری زیر معرفی می‌شود:
- C_{finger}: هزینه مربوط به استفاده از یک انگشت برای تایپ کردن دو حرف پشت سر هم.
- $C_{ergonomic}$: هزینه مربوط به تایپ کردن یک حرف با توجه به موقعیت آن حرف بر روی صفحه‌کلید.

تابع تناسب

- تابع تناسب برای هر کروموزوم از مجموع این سه فاکتور برای تمامی حروفی که در متن مورد استفاده برای آزمایش وجود دارند, بدست می‌آید:

$$\text{Fitness (layout)} = \sum_{w_i \in W} \sum_{l_j \in w_i} [C_{\text{hand}}(l_j, l_{j-1}) + C_{finger}(l_j, l_{j-1}) + C_{ergonomic}(l_j)]$$

W مجموعه تمامی کلمات موجود در متن مورد استفاده برای آزمایش است؛

- w_i کلمه iام از مجموعه W است؛
- l_j حرف jام از کلمه w_i است؛

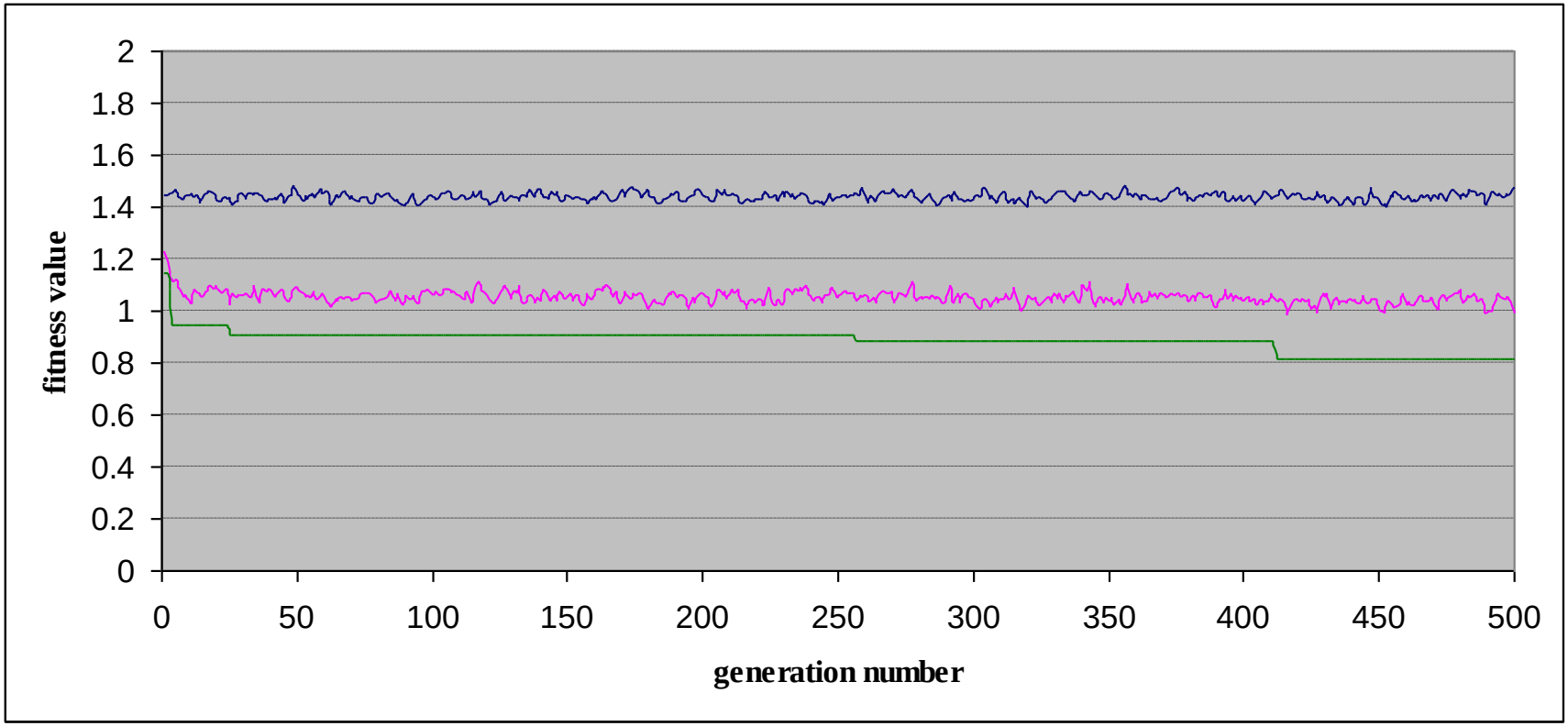
عملگرهای ژنتیکی

- در اینجا تنها از عملگر جهش استفاده شده است.
- دلیل عدم استفاده از عملگر دورگه این است که ساختار اعضای جمعیت طوری است که ترکیب کردن دو کروموزوم والد هزینهٔ زمانی بالایی دارد.
- عملگر جهش را به دو صورت برای اعضای مختلف جامعه به کار می‌بریم. در این مساله یک جامعه نخبگان انتخاب می‌کنیم که اعضای آن $\alpha\%$ تراز اول جمعیت از دید تابع تناسب را تشکیل می‌دهند.
- عملگر جهش برای هر عضو از جامعهٔ نخبگان, تنها محتویات چهار زوج ژن را به صورت تصادفی جابه‌جا می‌کند. در حالیکه برای افراد عادی جامعه این تعداد به ۱۲ جابه‌جایی افزایش می‌یابد.

کارایی

- الگوریتم ژنتیک با پارامترهای زیر اجرا کردیم:
- تعداد اعضای جمعیت ۱۰۰ کروموزوم که در نسل اول به صورت تصادفی تولید شده‌اند؛
- α درصد تشکیل‌دهنده جامعه نخبگان است, ۱۰% کل جمعیت ؛
- تعداد اعضایی که به صورت مستقیم و بدون اینکه عمگرهای ژنتیکی بر روی آن اعمال شود, به نسل بعدی می‌روند, ۳ عضو؛
- و تعداد کل نسل‌ها ۵۰۰ نسل.

کارایی



نمودارهای تناسب اعضای جامعه در طی نسل‌های مختلف. منحنی‌های نشان داده شده به ترتیب از بالا به پایین, متوسط مقادیر تناسب همه اعضای جامعه, متوسط مقادیر تناسب جامعه نخبگان, و بهترین تناسب هستند.

بهترین چینش

- بهترین چینشی که در نهایت این الگوریتم ژنتیک برای حروف فارسی ارائه داد, هزینه‌اش با توجه به تابع تناسب, ۸۱۵/۰ هزینه چینش کنونی حروف فارسی است .

مدلهای تکامل

- در سیستمهای طبیعی هر موجود زنده در طول زندگی خود یاد میگیرد که با شرائط سازگاری نماید .به همین ترتیب نسل های مختلف یک نمونه در طول زمان سازگاری های مختلفی را کسب میکنند:
- سوال:

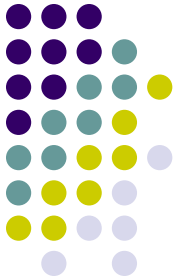
رابطه بین یادگیری یک موجود در طول زندگی شخصی و یادگیری نسل های یک نمونه در طول زمان چیست؟

Lamarckian evolution

- Lamarckدانشمند قرن نوزدهم فرضیه ای ارائه کرده که طبق آن تجربیات یک موجود زنده در ترکیب ژنتیکی فرزندان آن تاثیر میگذارد.
- برای مثال موجودی که یاد گرفته از غذای سمی پرهیز کند این ویژگی را بصورت ژنتیکی به فرزندان خود منتقل مینماید تا آنها دیگر مجبور به یادگیری این پدیده نباشند.
- اما شواهد تجربی این نظر را تائید نمیکنند :

یعنی تجربیات فردی هیچ تاثیری در ترکیب ژنتیکی فرزندان ندارد.

Baldwin Effect



نظریه دیگری وجود دارد که تاثیر یادگیری را بر تکامل توضیح میدهد .این نظریه که اثر Baldwin نامیده میشود بر مبنای مشاهدات زیر استوار است:

- اگر موجودی از طرف محیط متغیری تحت فشار قرار گرفته باشد، افرادی که توانائی یادگیری نحوه برخورد با شرایط را داشته باشند شانس بیشتری برای بقا دارند.
- موجوداتی که تحت شرایط جدید باقی میمانند جمعیتی با توانائی یادگیری را تشکیل میدهند که فرایندهای تکاملی در آنها سریعتر رخ میدهد و باعث میشود تا نسلی بوجود بیاید که نیازی به یادگیری مواجهه با شرایط جدید را نداشته باشند.

اجرای موازی الگوریتم های ژنتیک

- الگوریتم های ژنتیک از قابلیت خوبی برای پیاده سازی بصورت موازی برخوردارد هستند.
- در یک روش پیاده سازی موازی، جمعیت به گروههای کوچکتری با نام demes تقسیم شده و هرکدام در یک گره محاسباتی مورد پردازش قرار میگیرند .
- در هر گره یک الگوریتم استاندارد GA بر روی deme اجرا میشود .
- انتقال بین گره ها از طریق پدیده migration صورت میپذیرد .

Evolving Neural Networks

از GA برای تکامل جنبه های مختلف NN استفاده زیادی بعمل آمده است.از جمله : وزنها، ساختارو تابع یادگیری.

- استفاده از GA برای یادگیری وزنهای یک شبکه عصبی میتواند بسیار سریعتر از روش استاندارد back propagation عمل نماید.
- استفاده از GA برای یادگیری ساختار شبکه عصبی مشکلتر میباشد . برای شبکه های کوچک با استفاده از یک ماتریس مشخص میشود که چه نرونی به چه نرونهای دیگری متصل است.آنگاه این ماتریس به ژن های الگوریتم ژنتیک تبدیل و ترکیبات مختلف آن بررسی میگردد.
- برای بدست آوردن تابع یادگیری یک شبکه عصبی راه حلهائی نظیر استفاده از GP مورد استفاده قرار گرفته اما عموما این روشها بسیار کند عمل کرده اند.

مراجع

[١]D. E. Glover, *Genetic Algorithm and Simulated Annealing*, page 12-31, Morgan Kaufmann, Los Altos, CA, 1987.

[٢]Lissa W. Light and Peter G. Anderson, “Typewriter keyboards via simulated annealing”, *AI Expert*, September 1993.

[٣]Peter Klausler, www.visi.com/~pmk/evovled.html, September 2005.

[٤]M. O. Wagner, B Yannou, S. Kehl, D. Feillet, and J. Eggers, Ergonomic Modeling and optimization of keyboard arrangement with an ant colony algorithm”, *European Journal of Operation research*, 2003.

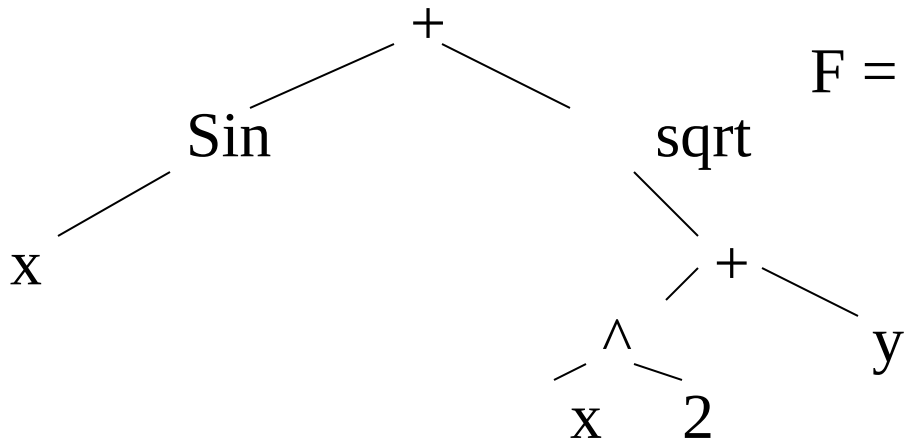
[٥]P. S. Deshwall and K. Deb, Design of an Optimal Hindi Keyboard for Convenient and Efficient Use. Technical Report on KanGAL Report No. 2003005, Indian Institute of Technology, Kanpur, 2003.

[٦]D. A. Norman and D. E. Rumelhart, *Cognitive Aspects of Skilled Typing*, Springer-Verlag, New York, 1983.

[٧]J. S. Goetti, A.W. Brugh, and B. A. Julstrom, “Arranging the Keyboard with a Permutaion-Coded Genetic Algorithm”, *Proc. of the 2005 SCM Symposium on Applied Computing*, Volume 2, pp. 947-951, 2005.

Genetic Programming

- GPتکنیکی است که کامپیوترها را قادر میسازد تا به حل مسائل بپردازند بدون آنکه بطور صریح برای آن برنامه ریزی شده باشند.
- GPروشی از الگوریتمهای تکاملی است که در آن هرعضو جمعیت یک برنامه کامپیوتری است.
- برنامه ها اغلب بتوسط یک درخت نمایش داده شده و اجرای برنامه برابر است با pars کردن درخت.



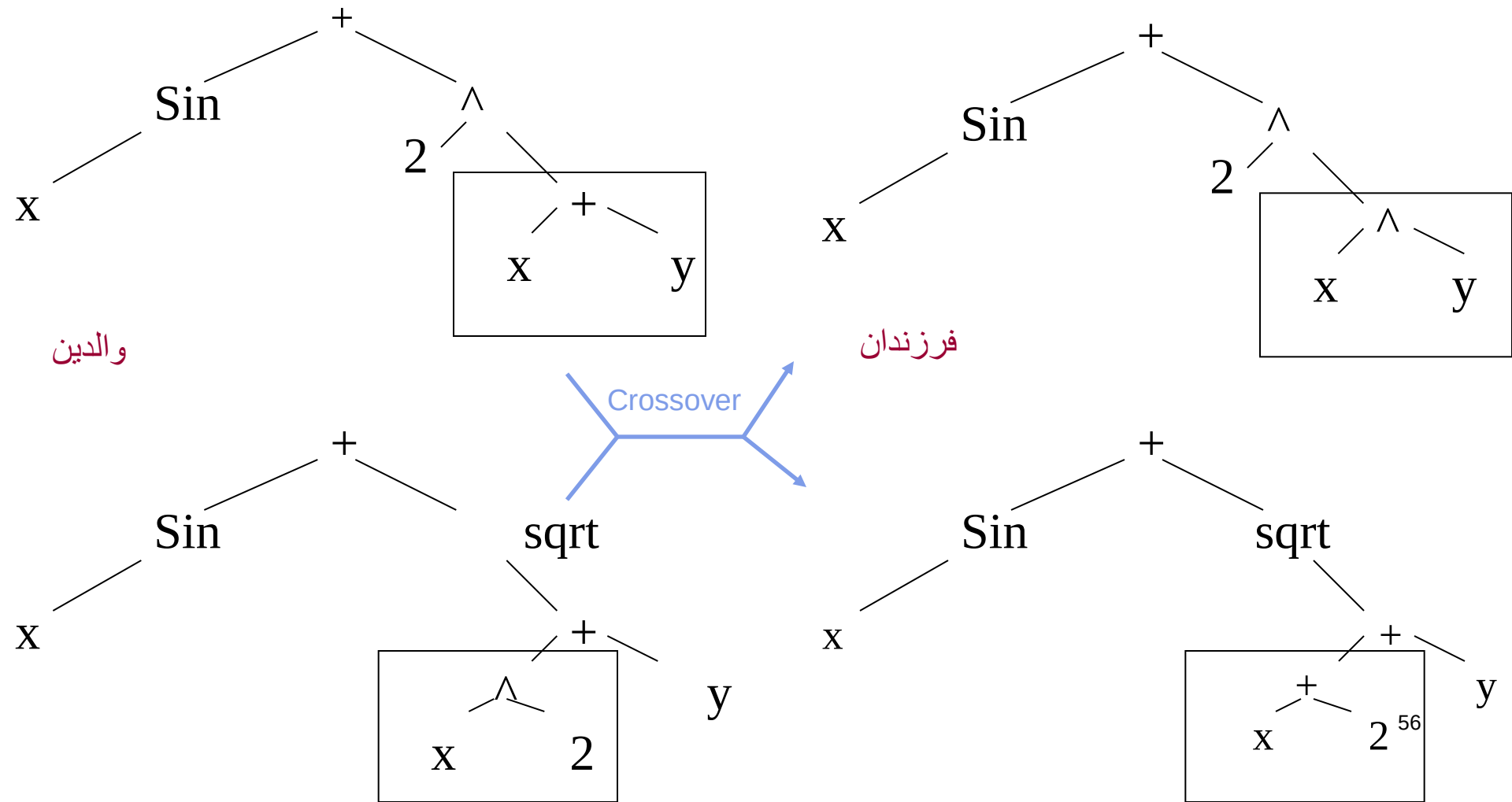
F = sin(x) + sqrt(x^2 + y)

نمایش برنامه ها

- برای استفاده از GP در یک زمینه خاص،
  - میبایست توابع پایه ای که در آن زمینه مورد نیاز هستند نظیر sin, cos, sqrt, +, -, etc توسط کاربر تعریف شوند
  - همچنین ترمینالها نظیر متغیرها و ثوابت نیز باید مشخص شوند
- آنگاه الگوریتم GP در فضای بسیار بزرگ برنامه هائی که توسط این مقادیر اولیه قابل بیان هستند یک عمل جستجوی تکاملی را انجام خوهد داد.

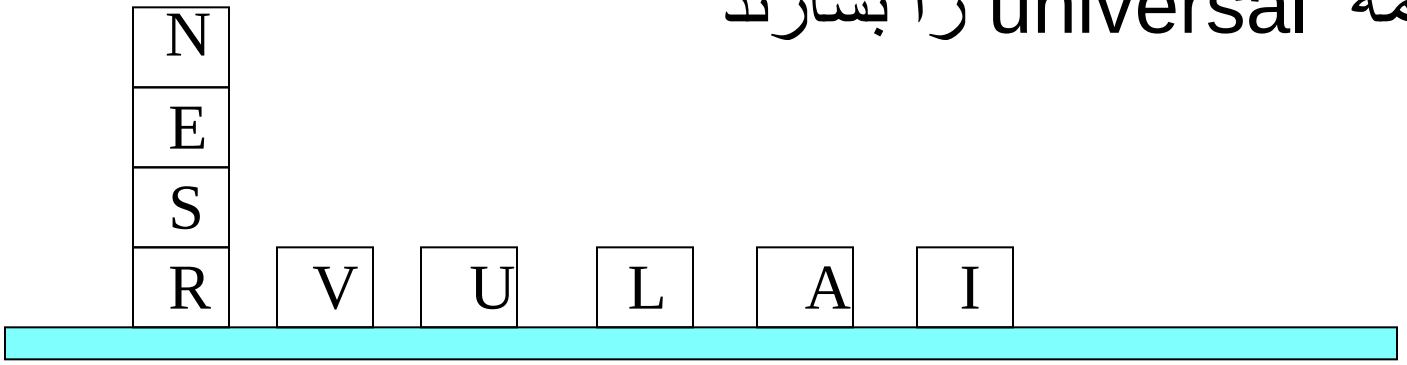
اپراتور crossover برای GP

- اپراتور crossover : شاخه هائی از یک درخت والد با شاخه هائی از درخت والد دیگر بطور تصادفی عوض میشوند



مثال

- در کتاب مثالی از Koza مطرح شده که در آن الگوریتمی یاد گرفته میشود که بتواند بلوک ها را در یک ستون روی هم بچیند
- هدف مسئله این است که بلوک ها طوری روی هم چیده شوند که کلمه universal را بسازند



مثال

- محدودیت :در هر مرحله فقط میتوان یک بلوک را جابجا نمود .در نتیجه تنها حرکت های ممکن عبارتند از :
  - بلوک آخر ستون را میتوان روی میز قرار داد و یا اینکه
  - یک بلوک را از میز به انتهای ستون منتقل نمود.
- توابع اولیه:
  - CS (current stack) :نام بلوک موجود در انتهای ستون را بر میگرداند (F برای حالتی که بلوکی وجود ندارد)
  - TP (top correct block)نام آخرین بلوکی را که همراه با بلوک های زیرینش ترتیب صحیح مورد نظر را دارند، برمیگرداند
  - NN (next necessary)نام بلوکی که باید در بالای TP قرار گیرد تا ترتیب universalدرست دربیاید.

مثال

- سایر توابع اولیه:

- (MS x) move block x to stack اگر بلوک x روی میزباشد این اپراتور آنرا به بالای ستون منتقل میکند .درغیر اینصورت مقدارF برمیگرداند.
- (MT x) move block x to table اگر بلوک x جائی روی ستون باشد این اپراتور بلوک بالای ستون را به میز منتقل میکند .درغیر اینصورت مقدار Fبرمیگرداند.
- (EQ x y) returns true if x = y.
- (NOT x) returns the complement of x.
- DU(x y) do x until expression y is true

مثال

مقدار تابع fitness

- در این آزمایش تعداد 166 مثال که هر یک در برگیرنده آرایش اولیه متفاوتی برای بلوک ها بودند تدارک دیده شده بود.
- تابع fitness یک برنامه برابر است با تعداد مثالهائی که برنامه قادر به حل آن است.
- برنامه با جمعیت اولیه ای برابر با 300برنامه تصادفی شروع بکار نموده و پس از تولید 10 نسل قادر میشود تا برنامه ای پیدا نماید که تمامی 166 مثال را حل نماید:

(EQ (DU (MT CS)(NOT CS)) (DU (MS NN)(NOT NN)))

مثال : طراحی فیلتر

صورت مسئله :برنامه ای که یک مدار ساده اولیه رابه مدار پیچیده مورد نیاز تبدیل نماید.

- توابع اولیه:
  - تابعی برای اضافه کردن قطعات و سیم بندی های مدار
  - تابعی برای حذف کردن قطعات و سیم بندی های مدار
- تابع fitness شبیه سازی مدار بدست آمده توسط نرم افزار SPICE برای مشخص نمودن میزان تطبیق آن با طرح مورد نظر .خطای مداربرای 101 فرکانس مختلف مورد بررسی قرار میگرفت.
- جمعیت اولیه : 640000
- نرخcrossover : 89 درصد
- نرخ mutation : 1 درصد

مثال : طراحی فیلتر

- سیستم برروی یک کامپیوتر موازی با 64 گره مورد آزمایش قرار گرفت.
- برای نسل اولیه ای که بصورت تصادفی ایجاد شدند در 98% مواقع حتی امکان شبیه سازی وجود نداشت.
- این نرخ بتدریج کاهش یافته و پس از تولید 137 نسل مداری حاصل شد که با مشخصات مورد نظر صدق میکرد.

در اغلب موارد کارائی الگوریتم GP بستگی به نحوه نمایش و همچنین تابع fitness دارد.