حل, رتبه بندی و تولید سودوکو بر اساس سطح بازی کاربر با استفاده از الگوریتم GA

محمد ایزدی
کارشناسی دانشگاه کاشان

رضا مظاهری
کارشناسی دانشگاه کاشان

محمد عماد مالکی
کارشناسی دانشگاه کاشان

استاد راهنما : جواد سلیمی سرتختی .

چکیده – در چند دهه ی اخیر سودوکو محبوبیت خاصی در میان مردم و به خصوص جامعه ی محققان و دانشمندان پیدا کرده است . شاید این محبوبیت بخاطر سادگی بازی و قوانین آن ولی در با وجود این سادگی چالش بر انگیز بودن آن هم برای کسانی که از این بازی لذت میبرند و هم برای محققان جهت آزمایش و پیاده سازی روش های حل, تولید و درجه بندی سختی آن که در این مقاله تمرکز ما بر روی الگوریتم GA و استفاده آن در موارد ذکر شده و بررسی و محاسبه سطح بازی کاربر و ارائه سودوکو مناسب و در عین حال چالش برانگیز برای کاربر می باشد.

کلیدواژه – حل سودوکو, تولید سودوکو, رتبه بندی سودوکو, الگوریتم GA, پردازش سطح بازی کاربر

1 – مقدمه

بر طبق گفته های ویکی پدیا [1] سودوکو در ابتدا یک بازی ژاپنی میباشد که محبوبیت خاصی در اروپا و امریکای شمالی یافته است . با این وجود اولین پازل در امریکا و در سال 1979 ساخته شد و پس از آن به ژاپن راه پیدا کرد و در این اواخر شاهد آن در غرب هستیم. محبوبیت سودوکو بخاطر چالش برانگیز بودن آن همراه با قوانین ساده می باشد. از آنجایی که پازل سودوکو نمونه خوبی از رویکردهای مختلف استدلال در زمینه هوش مصنوعی است، توجه زیادی را از جامعه هوش محاسباتی جلب کرده است.

پازل سودوکو یک مربع 9x9 از مربع های کوچکتر و شامل 9 مربع 3x3 می باشد (این توضیحات شکل ساده و اصلی سودوکو بود, سودوکو شکل های متفاوت در اندازه های گوناگون دیگری دارد).

حل سودوکو به این صورت می باشد که هر سطر, ستون و هر کدام از مربع های 3x3 باید شامل تمام اعداد 1 تا 9 بدون تکرار باشد. در اول بازی تعدادی از خانه ها پر شده اند و باید تا آخر بازی دست نخورده باقی بمانند(شکل 1-a).

تعداد اعداد داده شده تاثیر چندانی برروی سختی سودوکو ندارد[3][2] ولی 15 الی 20 عامل برای تعیین درجه سختی سودوکو وجود دارند[4].

در ابتدای این پژوهش پس از توضیح مختصری درباره تحقیقات قبلی انجام شده در این زمینه و توضیح الگوریتم ژنتیک, به بررسی چگونگی حل سودوکو بوسیله الگوریتم ژنتیک می پردازیم[5].

پس از آن به بررسی کارایی الگوریتم ژنتیک در مبحث محاسبه درجه سختی سودوکو و در مرحله سوم به تولید سودوکوی جدید بوسیله الگوریتم ژنتیک و محاسبه درجه سختی آنها و در آخر مبحث پردازش و محاسبه سطح بازی کاربر میپردازیم.

2- تحقیقات مشابه

الگوریتم GA برای حل سودوکو مناسب است ولی توانایی حل تمامی سودوکو ها را ندارد. در این مقاله تمرکز ما برروی استفاده از الگوریتم های تکاملی میباشد. دلیل اصلی برای حل سودوکو با GA، کسب اطلاعات بیشتر در مورد قابلیت های GA در مسائل ترکیبی و محدود است و امیدواریم بتوانیم ترفندهای جدید را برای کارآیی آن در این حوزه مشکلات نیز یاد بگیریم. چند مقاله درباره حل سودوکو با EA وجود دارند. Moraglio [3] و همکارانش سودوکو را بوسیله الگوریتم ژنتیک و بازترکیبی های هندسی حل کردند. آنها ادعا کردند که روششان بهتر از الگوریتم hill-climbing و استفاده از جهش تنها عمل میکند.

الگوریتم آنها بطور موثری سودوکوهای آسان را حل می کند[5]، ولی در حل سودوکوهای سخت تر با مشکل روبرو می شود. آنها همچنین تصدیق کردند که الگوریتم های تکاملی کارآمدترین روش برای حل سودوکو نیستند، اما سودوکو یک مورد مطالعه جالب برای توسعه الگوریتم است.

نیکولاو و رایان [6] از روش کاملا متفاوت برای حل سودوکوس استفاده کرده اند: GAuGE آنها (الگوریتم های ژنتیکی با استفاده از

تکامل گرامری) دنباله ای از عملیات منطقی را به وجود می آورد که پس از آن برای یافتن راه حل مورد استفاده قرار می گیرد.

Gold [7] از الگوریتم ژنتیک برای تولید یک سودوکو جدید استفاده کرده است، ولی به نظر ناکارآمد می باشد زیرا در این مقاله و تحقیقات دیگری که در این زمینه انجام شد ما با تولید 101 نسل توانستیم یک سودوکو تولید کنیم ولی آنها این کار را با 35700 نسل انجام دادند.

		1			3			
			4		8			
				6	2	7		
						9		1
	8				6			4
	4	7	9					
							4	
6	9			7			8	2
5	1					3		

(a)

9	6	1	7	5	3	4	2	8
7	2	5	4	9	8	6	1	3
4	3	8	1	6	2	7	5	9
2	5	6	8	4	7	9	3	1
1	8	9	5	3	6	2	7	4
3	4	7	9	2	1	8	6	5
8	7	3	2	1	9	5	4	6
6	9	4	3	7	5	1	8	2
5	1	2	6	8	4	3	9	7

(b)

شکل1. (a) نشان دهنده اعداد اولیه می باشد و شکل(b) کامل شده ی سودوکو بدون تغییر اعداد اولیه داده شده.

قواعد سودوکو نشان می دهد که مجموع اعداد در هر ستون و ردیف برابر است. بنابراین، سودوکو به وضوح مربوط به مسئله مربع سحر و جادو باستان است (مربع لاتین)، جایی که اندازه داده شده مربع باید پر شود بنابراین مجموع هر ستون و هر ردیف برابر است. مربع جادویی قبلا بسیله GA حل شده است [8],[9] .

3- الگوریتم ژنتیک

الگوریتم ژنتیک روش های بهینه سازی کامپیوتری برگرفته از نظریه داروین درباره طبیعت می باشد.

پایه راه حل یک مشکل به صورت اجتماعی از کروموزوم های شامل چندین ژن کدگذاری میشود.

بر خلاف آنچه شاهد آن در طبیعت هستیم ,

جامعه تولید شده (phenotype) به طور قطع از کروموزوم ها نشات میگیرند(genotype).

عصر یا محیط فنوتیپ نسل تولید شده توسط GA را در طول عمرشان تغییر نمی دهد.

نسل های تولید شده بوسیله تابع شایستگی در برابر مشکل ارزیابی می شوند.

هر چه تابع شایستگی بهتر باشد شانس آنها برای تولید نسل بعدی بیشتر است و افرادی با شایستگی نامناسب حذف می شوند. الگوریتم ژنتیک با استفاده از عملیات های بازترکیبی و جهش نسل های بعدی را تولید میکند.

در عمل بازترکیبی, ژن های یک کروموزوم جدید با استفاده از ترکیب ژن های والدین آن

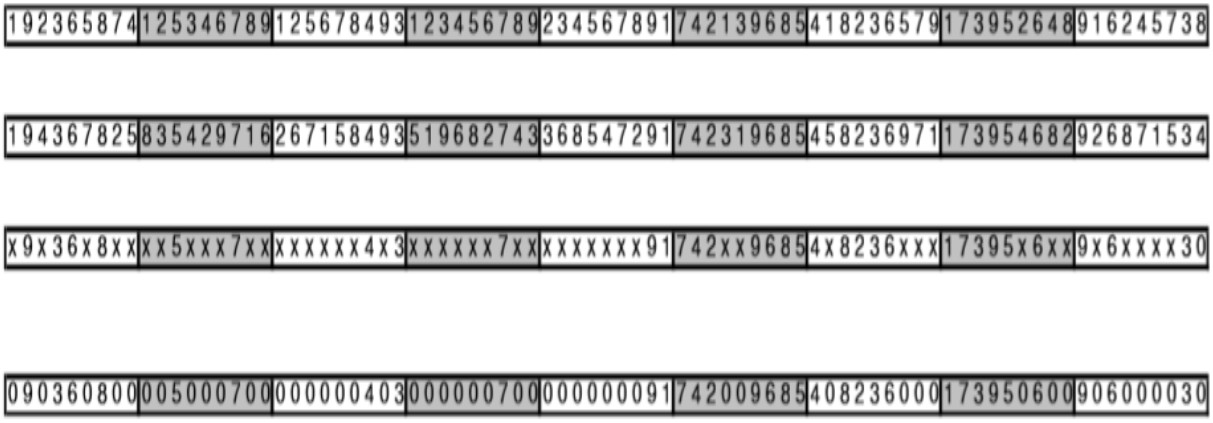
(point, or uniform crossover one-point, two) تشکیل می شوند. در جهش، ژنهای تصادفی یک کروموزوم را به صورت تصادفی یا با استفاده از یک استراتژی تغییر می دهیم. الگوریتم ژنتیگ با نگه داری اجتماع برتر کاملا شبیه نظریه داروین که گویای ماندگاری موجوداتی که توانایی سازگاری بیشتر را داشتن می باشد.

4- روش پیشنهادی حل سودوکو

اندازه کروموزوم GA برابر 81 می باشد, که به 9 زیر بلاک 9 عددی تقسیم می شود.

عملیات بازترکیبی یکنواخت تنها بین دو زیر بلاک و توالی جهش ها تنها در زیر بلاک ها اعمال می شوند.

احتمال جهش swap 0.6 بود (تعداد جهش های مورد آزمایش که با جهش های واقعی ساخته شده در این مورد برابر نیستند، جزئیات در ادامه جزئیات بیشتری را توضیح خواهیم داد).



شکل 2- یک نمایش از آرایه 81 تایی که به 9 زیر بلاک 9 عددی تقسیم شده. عملیات crossovers فقط در جاهایی که با خط عمودی نشان داده شده است قابل اجرا می باشند. اعدادی که مخالف صفر می باشند قابل تغییر نیستند و فقط اعدادی که با صفر شان داده شده اند قابلیت تغییر را دارند.

اندازه جمعیت 21(pop) و نسبت شایستگی 1(ELIT) می باشد.شایسته ترین این جمعیت با انتخاب x1 و x2 با الگوریتم ذیل انتخاب می شوند.

```
for( i=POP-1; i>=ELIT; i--){
  x1 = ( int )(i * Math.random() );
  x2 = ( int )(i * Math.random() );
  . . .
  }
```

همه افراد جدید از ابتدا با استفاده از crossover و سپس mutations برروی نتیجه crossover تولید شدند. با این وجود احتمال انتخاب x1 = x2 وجود دارد، جایی که تنها عملیات جهش، ژنوتیپ جدید فرد را تغییر می دهد. تنها شرط توقف پیدا کردن یک راه حل بود که ما موفق به یافتنش نشدیم.

علاوه بر قوانین اساسی سودوکو، در طول روند حل مسئله باید اعداد را مشاهده کرد. بنابراین راه حل سودوکو باید از چهار شرط ذیل پیروی کند:

1. هر سطر باید حاوی هر یک از اعداد صحیح از 1 تا 9 باشد.
2. هر ستون باید حاوی هر یک از اعداد صحیح از 1 تا 9 باشد.
3. هر زیر بلاک باید حاوی هر یک از اعداد صحیح از 1 تا 9 باشد.
4. اعداد داده شده باید در موقعیت های اصلی خود باقی بمانند.

ما تصمیم گرفتیم تا GA را برنامه ریزی کنیم به طوری که شرایط 3 و 4 به طور خودکار اجرا می شوند و تنها شرایط 1 و 2 بهینه می شوند. سودوکو دارای 9 ردیف و 9 ستون است، بنابراین ما 18 جای خالی بابر داریم که باید در زمان حل پر شوند. الگوریتم ژنتیک استفاده شده ر این مقاله در اصل نمونه تغییر یافته الگوریتم ژنتیکی است که برای حل مربع جادویی و مسائل ترکیبی دیگر استفاده میشده است.

این الگوریتم ژنتیک از crossover و mutation های مستقیم استفاده نمیکند زیرا میتوانند باعث ایجاد موقعیت های غیر مجاز برای زیر بلاک ها شود.برعکس چیزی که باید باشد شاهد تکرار اعداد در سطرها و ستون ها در شرایط نامطلوبی خواهیم بود.

اپراتورهای ژنتیکی مجاز به انتقال اعداد ثابتی که در ابتدای مشکل آمده است نیستند. این موضوع بوسیله آرایه کمکی تضمین میشود.

تصمیم گرفتیم یک راه حل پازل سودوکو را به عنوان یک آرایه عدد صحیح از 81 عدد ارائه کنیم.

آرایه به 9 زیر بلاک (بلاک های ساختمانی) از 9 عدد مربوط به زیر بلاک 3x3 از چپ به راست و از بالا به پایین تقسیم می شود. Crossovers تنها میان زیر بلاک ها اتفاق میافتد. بنابراین نقطه متقاطع نمی تواند درون یک زیربلاک باشد (شکل 3).

- عملگر جهش

Mutations فقط در زیر بلاک ها اتفاق میافتند.

در اصل، ما از سه استراتژی جهش مختلف استفاده کردیم که معمولا در بهینه سازی ترکیبی استفاده می شود: swap mutation, 3-swap mutation, and insertion mutation.

با این حال، از آنجا که جهش های 3-swap و interstitial در واقع فقط توالی های جهش های swap (swap 2) هستند، ما آنها را با یک دنباله از 1 تا 5 swap mutations در یک زیر بلاک جابجا کردیم.

دلیل این کار این بود که ما فهمیدیم که الگوریتم ژنیک با swap mutation بهتر از 3-swap و interstitial عمل میکند.

در swap mutation دو مقدار در زیر بلاک با یکدیگر در شرایطی که قوانین را رعایت کنند جابجا می شوند.

اگر ما دنباله ای از جهش های متوالی رسم کنیم این دنباله تا جایی ادامه پیدا می کند که قوانین نقض نشوند.

گاهی اوقات ممکن است که اولین جهش یک قانون را نقض کند, بنابراین درصد جهش واقعی کمتر از 0.6 میشود. ممکن است GA با یک جهش بتواند جایگزین GA با 1 تا 5 جهش شود.

یک روش خاص دیگر که ما اضافه کردیم یک جهش مجدد بود [10]. در مسایل ترکیبی، بهینه‌سازی اغلب گیر افتاده و برای راه‌اندازی مجدد آن با جمعیت اولیه جدید، موثرتر از تلاش برای ادامه یافتن از وضعیت گیرافتاده است.

- تابع شایستگی

طراحی یک تابع شایستگی که به جستجوی GA کمک کند اغلب در مسایل ترکیبی دشوار است [11]. در این حالت، ما در ابتدا [12] از یک تابع شایستگی تا حدی پیچیده استفاده کردیم که تخلفات مختلف را به طور متفاوت، جریمه کرد. مجموع اعداد هر سطر و ستون باید برابر 45 شود و هر سطر xi و ستون xj باید برابر مجموعه A که شامل تمام اعداد 1 تا 9 باشد در غیر این صورت به تابع شایستگی جریمه مقدار مشخصی به عنوان جریمه اضافه میکنیم.

اما با حذف بخش هایی از تابع شایستگی، ما نتیجه گرفتیم که تابع شایستگی بسیار ساده تر نیز انجام می شود.

شرایطی که زیر بلاک ها باید شامل تمام اعداد صحیح 1 تا 9 باشند و اعداد اولیه جابجا یا حذف نشوند, در صورت نقض ای شرایط ثابت به تابع شایستگی جریمه اضافه می شود.

$$A = \{1,2,3,4,5,6,7,8,9\}$$
$$g_{i3}(x) = |A - x_i|$$
$$g_{j3}(x) = |A - x_j| \quad (1)$$

توابع (1) تعداد اعداد از دست رفته در سطر و ستون ها را حساب میکنند.

در وضعیت مطلوب، تمام ارقام در مجموعه سطر و ستون ظاهر می شوند و مقدار تابع شایستگی صفر می شود. در غیر این صورت مقدار مجاز | n-1 | به تابع شایستگی اضافه شده است. این بدان معنی است که اگر یک رقم از دست رفته باشد مقدار مجاز 1 اضافه می شود و اگر تعدادی عدد n بار تکرار شوند و n>1 باشد,

مقدار مجاز n-1 اضافه می شود. بنابراین هر اشتباه در تعدادی سطر یا ستون برابر حداقل مقدار مجاز 2 است. تابع شایستگی باید حداقل باشد.

```
If Best(generation_i) = Best(generation_{i-1})
   then Value(Best)+=1;
```

(2)

طبق شکل (2) ما یک جریمه دیگری اضافه کردیم. به این صورت که اگر راه حلی در نسل تولید شده بهترین باشد و همین موقعیت را (که بخاطر مقدار شایستگی آن است) در نسل های بعد ادامه بدهد هر بار تابع شایستگی آن را به مقدار 1 جریمه می کنیم. این راه حلیست برای جایگزینی نسل های جدید.

5-بررسی سطح دشواری سودوکو

برای ایجاد معیارهای گسترده برای تعیین سطوح دشواری، برای اولین بار یک مدل ریاضی برای تعریف آن با یک فرمول قابل اجرا ایجاد می کنیم. هدف این بخش ارائه یک متریک کمی برای ارزیابی پیچیدگی بدون هیچ گونه بازی است. در روش سنتی، معیارهای سطح دشواری آن‌ها مشخص نشده است. اکثر سطوح دشواری تنها با توجه به تعداد اعداد اولیه داده شده مشخص می شود، که به این معنی است که هرچه تعداد معلومات بیشتر باشد، راه حل ساده‌تر است. اما در حقیقت عوامل بسیاری وجود دارند که بر این پازل تاثیرگذار هستند، از جمله تکنیک‌های مورد استفاده، تحلیل منطقی سطح دشواری و خانه های اولیه پر و خالی و موقیت آنها.

با این حال، سطح دشواری سودوکو عمدتا به توزیع اولیه داده ها بستگی دارد [13]. به این معنا است که توزیع اولیه داده ها عامل اصلی است که روی پازل سودوکو تاثیر می گذارد. در این قسمت ما در مورد چگونگی به دست آوردن سطح دشواری بر اساس توزیع اولیه صحبت خواهیم کرد. البته روش های

دیگری برای محاسبه سطح دشواری پس از حل سودوکو بوسیله الگوریتم GA نیز وجود دارد.

- آماده سازی

شانون [14] برای اولین بار تعریف آنتروپی را پیشنهاد کرد که می تواند عدم قطعیت سیگنال ارسال شده توسط یک منبع اطلاعاتی را ارزیابی کند. مفهوم "عدم قطعیت" شبیه عدم اطمینان داشتن به انتخاب شماره درست در سطرها است. بنابراین ما پیشنهاد می کنیم پیچیدگی حل پازل سودوکو را با استفاده از آنتروپی اطلاعات ارزیابی کنیم.

تعریف 2 [14] . فرض کنید که x یک متغییر تصادفی گسسته با مقدار احتمالی $\{p_1, p_2, \ldots, p_n\}$ میباشد, آنتروپی x برابر :

$$H(X) = -\sum_{i=1}^{n} p_i \log_b(p_i) \quad ,where \ (\sum_{i=1}^{n} p_i = 1) \qquad (1)$$

بطور کلی, b مقدار 2 خواهد بود و H(X) نشان دهنده عدم قطعیت سیگنال ارسال شده توسط X است.

مقدار H (X) یک عدد مثبت حقیقی است و هرچقدر مقدار آن بزرگتر باشد، عدم قطعیت بزرگتر است. بنابراین، این تعریف به عدم قطعیت پازل سودوکو تعمیم خواهد یافت.

- طراحی دشوار

تعریف 3. تعداد اعداد که می تواند در یک سلول خالی مشخص پر شود، درجه غیر متمرکز (با n مشخص می شود) از این سلول نامیده می شود.

در طی حل این معما، سلول‌های خالی مرحله به مرحله پر خواهند شد. در هر مرحله، تعدادی نامزد برای هر سلول خالی وجود خواهند داشت، به راحتی می توان فهمید که ما باید سلولی را انتخاب کنیم که حداقل درجه قطعی ((min(n)) داشته باشد و فضای خالی را با یک نامزد احتمالی پر کند. امکان انتخاب یکی از نامزدها : $1/\min(n) \quad (if \quad \min(n) \neq 0)$ می باشد.

با توجه به تعریف آنتروپی، عدم قطعیت ایجاد شده در این مرحله به شرح زیر است:

تعریف ۴. آنتروپی مرحله i به صورت زیر تعریف می‌شود:

$$H_i = -[1/\min(n) \log_2(1/\min(n))] * \min(n) = \log_2(\min(n))$$
$$(\text{where} \ \min(n) \neq 0) \qquad (2)$$

آنتروپی یعنی عدم قطعیت گام i. فرض کنید که برای اتمام این معما, N مرحله وجود داشته باشد, آنتروپی مجموع برای حل این معما برابر است با :

$$d = \sum_{i=1}^{N} H_i \qquad (3)$$

مقدار d عدم قطعیت را در کل فرآیند نشان می دهد. بزرگتر آنتروپی مجموع، بزرگتر عدم قطعیت از پازل است. بنابراین، مشکل پازل را می توان با انتروپی d جمع کرد.

- سطح دشواری

کار بعدی که باید انجام شود چگونگی محاسبه سطح دشواری D از d می باشد. رابطه D=f(d) باید از یک مقیاس به اعداد صحیح تنظیم شود, به طوری که هر پازل سودوکو در یک سطح دشواری خاص قرار گیرد.

تابع f ممکن است منحصر به فرد باشد. D مناسب است اگر f معقول است. برای مثال، اگر ما 4 سطح دشواری را انتخاب کنیم. یک روش این است:

$$\begin{aligned} &if \quad d \in [0,1); \\ &if \quad d \in [1,2); \\ &if \quad d \in [2,3); \\ &if \quad d \geq 3. \end{aligned} \tag{4}$$

البته، ما می‌توانیم D های متفاوتی را از f های متفاوت بگیریم. اما منطقی است که D بصورت یکنواخت در حال افزایش باشد. ما می‌توانیم از مقدار D برای تعریف سطح دشواری یک معمای Sudoku تصادفی استفاده کنیم.

6- تولید پازل

تعریف 5. نامزد، اعداد مناسب برای یک سلول خالی است.

پازل سودوکو می تواند به عنوان یک درخت به نام درخت پازل در نظر گرفته شود. گره ها در حین حل پازل داده شده خود پازل هستند. اگر سلول های خالی وجود داشته باشند که فقط یک نامزد برای پازل اصلی داشته باشند، سلول ها را با نامزد پر کنید. پازل دریافت شده ریشه درخت پازل است. با توجه به گره پدر، فرزندان (همان گره ها) پازل هایی هستند پس از پر شدن یک خانه خالی از گره بواسطه کاندیدا.

برای یک سطح دشواری داده شده D0, ما پیشنهاد الگوریتمی برای حل پازل سودوکو بر اساس درخت پازل را داریم.

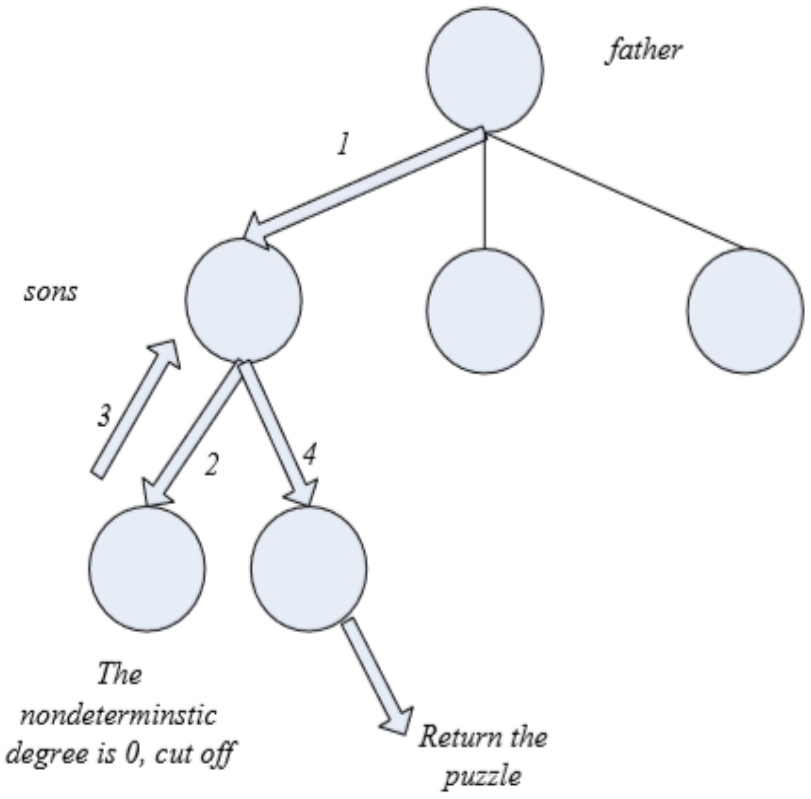
1. اگر پازلی در HASH MAP وجود دارد که سطح دشواری آن (D(D=f(d)) است, معمای مربوطه را به دست آورید و به پایان برسانید. به مرحله ۲ بروید.
2. یک پازل سودوکو را به صورت تصادفی ایجاد کنید
3. و d = 0 قرار دهید.
4. درجه غیرمتمرکز خانه های خالی ریشه را محاسبه کنید.
5. سلول خالی را انتخاب کنید که درجه غیرمتمرکز آن برابر min(N) و

$$d = d + \log_2(\min(n))$$

قرار دهید.

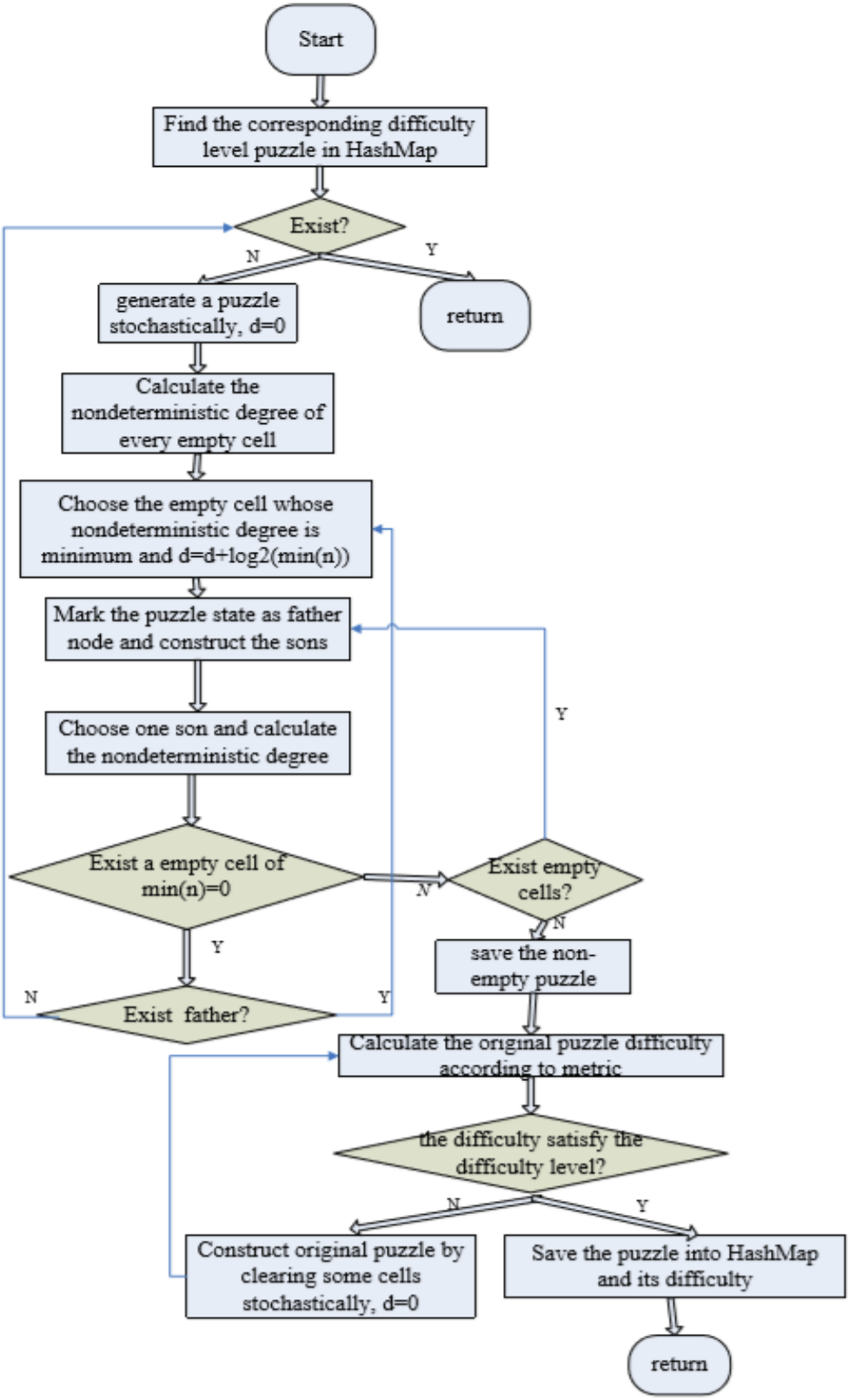
6. یکی از گره های فرزند را انتخاب کنید و درجه های غیرمتمرکز سلول های خالی پازل را که فرزند نشان می دهد محاسبه کنید.
7. اگر min(N)=0، شاخه را ببرید و به گره پدر برگردید .برو به فرزند بعدی .درجا ت غیر متمرکز سلول‌های خالی را محاسبه کرده و به مرحله ۴ بروید .در غیر این صورت اجازه دهید گره فعلی گره پدر باشد و به مرحله ۴ بروید.
8. هنگامی که تعداد سلول های خالی 0 است، پازل را ذخیره کنید.
9. محاسبه D با D=f(d) برابر است, پازل و آنتروپی مجموع را در hash map ذخیره کنید.
   آن را برگردانید و به پایان برسانید .سپس برخی از سلول‌ها را پاک کنید، d=0 قرار دهید و به مرحله ۳برگردیم.

10. یک پازل سودوکو را براساس یک سودوکو حل‌شده تولید کنید، و تنها کافیست تعدادی از اعداد را حذف کنید.HASHMAP یک جدول ویژه است که می‌تواند نتایج را ذخیره کند مسیر مرحله ۳ به مرحله ۶ یک نمودار درختی است که به شرح زیر توصیف می‌شود:



شکل 3-درخت پازل

در ادامه جزئیات الگوریتم را بوسیله flowchart نمایش می دهیم.



شکل 4-نمودار جریان الگوریتم

جمع بندی

در این مقاله ما بررسی کردیم که آیا با الگوریتم ژنتیک میتوان سودوکو را حل, درجه بندی و تولید کرد یا خیر ؟ که برای هر مورد راه های پیشنهادی ارائه دادیم . برای حل سودوکو از الگوریتم های متفاوت زیادی می توان استفاده کرد و بعضی از آنها سودوکو را با سرعت بیشتری حل می کنند ولی باز هم در برخی از موارد قادر به حل سودوکو نبودند.

رتبه بندی سودوکو مسئله مشکلی است و تعداد افراد کمی به دنبال یافتن راه حلی برای آن میگردند.

در این مقاله، مدل ما یک معیار برای تعریف سطح دشواری براساس نظریه اطلاعات ارایه می‌دهد و با این روش به راحتی میتوان میزان سختی سودوکو را تعیین کرد. الگوریتم ما می‌تواند سودوکو را با توجه به سطح آن تولید کند و از آن میتوان به عنوان یک ماشین تولید استفاده کرد.

معمولا برای تولید پازل های جدید, ابتدا یک راه حل برای سودوکو پیدا میکنند و سپس تعدادی از اعداد آنهارا حذف می کنند.

مراجع

[1] Wikipedia: Sudoku. Available via WWW: http://en.wikipedia.org/wiki/Sudoku (cited 16.10.2006)

[2] Semeniuk, I.: Stuck on you. In NewScientist 24/31 December (2005) 45-47

[3] Moraglio, A., Togelius, J., Lucas, S.: Product geometric crossover for the sudoku puzzle. In 2006 IEEE Congress on Evolutionary Computation (CEC2006), Vancouver, BC, Canada, July 16-21 (2006) 470-476.

[4] Wikipedia: Sudoku. Available via WWW: http://en.wikipedia.org/wiki/Sudoku (cited 16.10.2006)

[5] Pappocom: Su|do|ku. Available via WWW: http://www.sudoku.com (cited 16.10.2006)

[6] Nicolau, M., Ryan, C.: Genetic operators and sequencing in the GAuGE system. In IEEE Congress on Evolutionary Computation CEC 2006, 16-21 July 2006, 1561 - 1568

[7] Gold, M.: Using Genetic Algorithms to Come up with Sudoku Puzzles. Sep 23, 2005. Available via WWW: http://www.csharpcorner.com/UploadFile/mgold/Sudoku 09232005003323AM/Sudoku.aspx?ArticleID=fba36449-ccf3-444fa435-a812535c45e5 (cited 16.10.2006)

[8] Ardel, D.H.: TOPE and magic squares, a simple GA approach to combinatorial optimization. In J.R. Koza (ed.) Genetic Algorithms in Stanford, Stanford bookstore, Stanford, CA (1994)

[9] Alander, J.T., Mantere T., Pyylampi, T.: Digital halftoning optimization via genetic algorithms for ink jet machine. In B.H.V. Topping (ed.), Developments in Computational mechanics with high performance computing, CIVIL-COMP Press, Edinburg, UK (1999) pp. 211-216

[10] Eshelman, L.J.: The CHC adaptive search algorithms: how to safe search when engaging in nontraditional genetic recombination. In G. Rawlinns (ed.) Foundations of Genetic Algorithms, Morgan Kaufmann (1991)

[11]Koljonen, J., Alander, J.T.: Solving the "urban horse" problem by backtracking and genetic algorithm – a comparison. In J. Alander, P. Ala-Siuru, and H. Hyötyniemi (eds.), Step 2004 – The 11th Finnish Artificial Intelligence Conference, Heureka, Vantaa, 1-3 September 2004, Vol. 3, Origin of Life and Genetic Algorithms (2004) 127-13.

[12] Mantere, T., Koljonen, J.: Solving and rating sudoku puzzles with genetic algorithms. In Finnish Artificial Intelligence

Conference (STeP 2006), October 26-27, Finnish Artificial Intelligence Society FAIS, Espoo, Finland (2006) 86-92

[13]I. Lynce and J. Ouaknine, “Sudoku as a SAT problem,” in Electronic Proceedings of the 9th International Symposium on Artificial Intelligence and Mathematics, 2006 .

[14]C.E. Shannon, “A Mathematical Theory of Communication,” The Bell System Technical Journal, 27, pp. 379–423, 623–656, 1948.