حل پازل سودوکو با الگوریتم های ژنتیک

علیرضا کشاورز (a. keshavarz@khu. ac. ir)

پروژه یادگیری ماشین-ترم اول ۱۴۰۱

عنوان مقاله:

Solving, Rating and Generating Sudoku Puzzles with GA

Timo Mantere and Janne Koljonen

دانشکده ریاضی و کامپیوتر-دانشگاه خوارزمی

۱.معرفی

سودوکو یک معما مبتنی بر منطق است که به طور گسترده به عنوان بازی و سرگرمی در روزنامه ها و مجلات وجود دارد. این بازی در ابتدا در دهه هشتاد در ژاپن رواج یافت، اما امروز یک بازی محبوب در سراسر جهان و یک معیار مفید برای آزمایش تکنیک های حل هوش مصنوعی است.

یک معمای سودوکو شامل پر کردن یک تابلوی ۹*۹ است که به زیر شبکه هایی با اندازه ۳*۳ تقسیم شده است. به طوری که هر سطر، ستون و زیر شبکه شامل ارقام مختلف از ۱ تا ۹ است. یک مشکل سودوکو، سلول های پر شده است یعنی سلول های داده شده ای که قابل تغییر و جابجایی نیست. مطمئنا مقدار سلول های پر شده در میزان دشواری مساله تاثیر محدودی دارند یا هیچ تاثیری ندارند. دشواری بیشتر به موقعیت سلول های پر شده در طول پازل بستگی دارد. یک طبقه بندی مفید برای

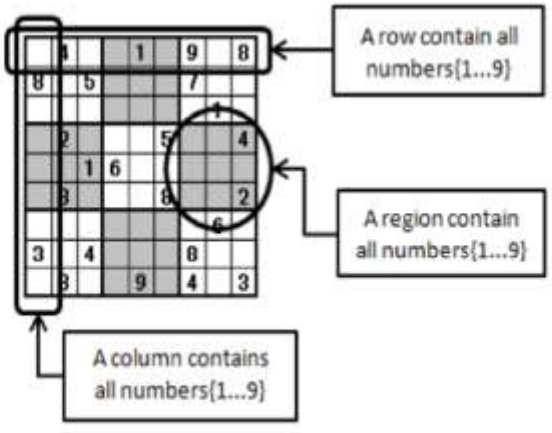


Fig. An example of Sudoku puzzles, 24 positions contain a given number, the other position should be solved.

دشواری شامل دشواری ساده، متوسط و سخت است که توسط مانتره و کولجنن ارائه شده است.

درجه دشواری سودوکو با تعداد سلول های پر شده (اعداد داده شده ابتدایی) و قرارگیری آن ها متفاوت است. اساسا، تعداد کمتری از سلول های پر شده به معنای تعداد ترکیب های بیشتری است که باید از بین آنها راه حل را پیدا کرد و این موضوع درجه دشواری را افزایش می دهد. اما حدود ۱۵ تا ۲۰ عامل وجود دارد که در درجه بندی دشواری تاثیر دارند.

سودکو انواع مختلفی دارد. بعضی از آن ها پازل هایی با اندازه های بزرگتر مانند ۱۶*۱۶ و ۲۵*۲۵ هستند. برخی دیگر محدودیت های جدیدی را اعمال می کنند مانند اینکه اجازه نمی دهند یک عدد بیش از یکبار در قطرها یا زیر شبکه های ۹ سلولی دارای رنگ یکسان ظاهر شود. برای حل معماهای بزرگتر مانند ۱۶*۱۶ یا ۲۵*۲۵، الگوریتم ژنتیک یا روش های تصادفی دیگر ممکن است موثر واقع شوند. همچنین روش های سرعت بخشیدن به محاسبات تکاملی از طریق پیاده سازی در واحدهای پردازش گرافیک (GPU) نیز ممکن است موثر باشد.

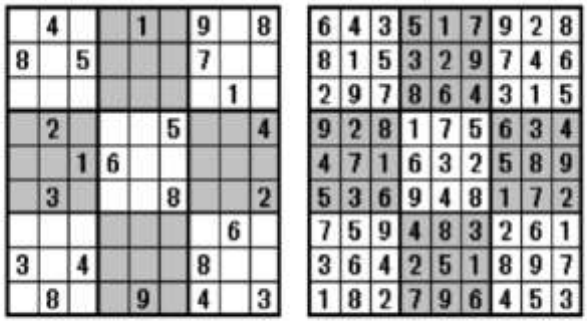


Fig. A solution for the Sudoku puzzles given in fig 1. The given numbers marked in bold-face.

سودوکو را میتوان به عنوان یک مشکل رضایت محدودیت در نظر گرفت. وقتی با الگوریتم های ژنتیک حل شود، میتوان آن را به عنوان یک مساله بهینه سازی چند هدفه کنترل کرد.

این بدان معنی است که از جهش ها یا همبری هایی استفاده نمی شود که میتوانند موقعیت های غیر قانونی

ایجاد کنند. موقعیت هایی مانند: ردیف ها، ستون ها و زیر شبکه هایی که میتوانند شامل برخی از اعداد ۱ تا ۹ بیش از یک مرتبه باشند یا اینکه شامل بعضی از اعداد بین ۱ تا ۹ نباشند. علاوه بر این عملگرهای ژنتیک مجاز به جابجایی سلول های پر شده توسط اعداد ثابت که در ابتدای مساله آورده شده اند، نیستند.

برای حل این معماهای اساسی روش هایی مانند ردیابی مجدد، که همه ترکیبات احتمالی را شمارش میکند و رویکرد فراابتکاری موثرند. همچنین برخی الگوریتم های موثرتر و سریعتر از الگوریتم ژنتیک برای حل این مسائل وجود دارند.

۲.حل سودوکو با الگوریتم های ژنتیک

۲,۱ .الگوریتم ژنتیک

همه الگوریتم های ژنتیک روش های بهینه سازی رایانه ای هستند که از تکامل داروینی طبیعت به عنوان الگو و الهام استفاده می کنند. پایه حل یک مشکل به صورت فردی رمزگذاری می شود که کروموزوم هایی متشکل از چندین ژن هستند.

در مقابل افراد در وراثت طبیعی (طبیعت)، کروموزوم ها در الگوریتم ژنتیک قرار دارند.

الگوریتم ژنتیک با استفاده از عملگرهای همبری و جهش، افراد جدیدی را ایجاد می کند. با استفاده از همبری ، قادر خواهیم بود ژن ها را با تمرین های از قبل تعیین شده از والدین خود برای یک کروموزوم جدید انتخاب کنیم. برخی از این تمرین ها عبارتند از همبری تک نقطه ای، دو نقطه ای و یکنواخت. با عملگر جهش، ژن های تصادفی کروموزوم را به طور تصادفی یا با استفاده از برخی استراتژی های از پیش تعریف شده، تغییر می دهیم.

استراتژی الگوریتم ژنتیک غالبا نخبه گرایانه است و بنابراین از اصول "بقای اصلح" تکامل داروین پیروی می کند.

۲,۲.دقت الگوریتم ژنتیک در حل سودوکو

مطالعات اندکی در مورد کاربرد الگوریتم ژنتیک در حل سودوکو انجام شده است و از همین رو در این زمینه تعداد مقالات علمی بسیار کمی وجود دارد. در روش هایی که در این مقالات بررسی شده اند، به راحتی راه حل های بهینه برای معماهای ساده یافت می شود، اما راه حل های بهینه برای معماهای دشواری که نقطه شروع آن ها داده های کمی دارند، اغلب در زمان واقعی قابل دستیابی نیستند. این مساله منجر به رویکردهایی شد که الگوریتم ژنتیک با Cultural Algorithms و Grammatical Evolution ترکیب شود. این ترکیب را به اختصار GAuGE (الگوریتم های ژنتیک با استفاده از تکامل گرامری) می نامند. GAuGE توالی عملیات منطقی را بهینه می کند که بعد از آن برای یافتن راه حل اعمال می شود.

محققین معتقدند دلیل شکست الگوریتم ژنتیک در معماهای دشوار، عملگر همبری است که تمایل دارد برنامه های کاملا متناسب (Building Blocks) را از بین ببرد. با این حال در این مطالعه تمرکز ما بر حل سودوکو با روش های ابتکاری است. دلیل اصلی حل سودوکو با الگوریتم ژنتیک، یادگیری بیشتر در مورد قابلیت های الگوریتم ژنتیک در مواجه با مشکلات ترکیبی محدود است.

۲,۳.روش پیشنهادی برای حل معما

۲,۳,۱.ساخت جمعیت آغازین (نسل اول)

جمعیت آغازین را می توانیم به صورت تصادفی ایجاد کنیم. یعنی در هر خانه از یک نمونه از جدول مورد نظر یک عدد تصادفی بین ۱ تا ۹ قرار دهیم. اگر بتوانیم جمعیت آغازین را به گونه ای انتخاب کنیم که در عین تصادفی بودن، قسمت هایی از شروط را نیز در خود داشته باشد، مطمئنا بسیار سریعتر به جواب می رسیم؛ چراکه هرچه جمعیت آغازین بهتر باشد، احتمال دسترسی سریع به جواب بیشتر است.

با توجه به این ایده میتوانیم جمعیت آغازین را به گونه ای طراحی کنیم که با اینکه در هر خانه یک عدد تصادفی

قرار داشته باشد، اما در هر سطر و ستون هیچ عدد تکراری نداشته باشیم.

شایان ذکر است در حل این مساله، کروموزوم ها (مجموعه های ۹ عنصری) که می توانند یک ردیف، یک ستون و یا یک زیر شبکه از پازل باشند را، به صورت یک متغیر گسسته در نظر گرفته ایم که می توانند بیانگر اعداد یا حالت های ۱ تا ۹ باشد.

۲,۳,۲.تابع هدف (ارزش)

در مسائل ترکیبی اغلب طراحی یک تابع ارزش که به جستجوی الگوریتم ژنتیک کمک کند، دشوار است. به همین خاطر در حل این مساله از یک تابع ارزش نسبتا پیچیده استفاده کردیم.

نیازهایی وجود دارد که در تعیین تابع ارزش باید آنها را برآورده ساخت. اولین نیاز این است که مجموع هر سطر و ستون باید برابر با ۴۵ باشد. دومین خواسته برابر بودن هر محصول ردیف و ستون با !۹ است. سومین نیاز از تئوری مجموعه گرفته می شود و مستلزم این است که هر سطر (x_i) و ستون (x_j) برابر با مجموعه ای در نظر گرفته شود که برابر با اعداد صحیح از ۱ تا ۹ است.

این روش تا حدودی خوب کار می کند اما با حذف قسمت هایی از تابع ارزش به این نتیجه رسیدیم که عملکرد تابع ارزش بسیار ساده تر نیز به همان خوبی انجام می شود.

این قسمت ها شامل شروط زیر هستند:
۱) شرطی که هر زیر شبکه شامل اعداد صحیح از ۱ تا ۹ باشد.
۲) وجود و عدم تغییر و جابجایی سلول های داده شده (اعداد ثابت)
این شروط ذاتا تضمین شده هستند و نیازی به آن ها نیست.

توابع (۱)، بیانگر تابع هدف (ارزش) مساله هستند. تابع ارزش بر اساس قانون عدم وجود هر عدد بیش از یک بار در هر ردیف و ستون است. امتیاز ردیف (g_i) برابر با تعداد اعداد مختلف در یک ردیف، و امتیاز ستون (h_i) برابر با تعداد اعداد مختلف در یک ستون است. نمره افراد (مقدار تابع ارزش) برابر با مجموع امتیازات ردیف و ستون است.

$$f(x) = \sum_{i=1}^{9} g_i(x) + \sum_{j=1}^{9} h_j(x)$$
$$g_i(x) = |x_i| \qquad (1)$$
$$h_j(x) = |x_j|$$

۲,۳,۳.عملگر انتخاب

در پیاده سازی این مساله از انتخاب تورنمنت (مسابقه) استفاده می کنیم. روش انتخاب تورنمنت سعی در کنترل فشار انتخاب به نحوی مطلوب و مستقل از اندازه شایستگی افراد دارد. ضمن اینکه این روش از نظر پیچیدگی محاسباتی بسیار کم هزینه تر از سایر روش های انتخاب بوده و با سخت افزارهای موازی قابل پیاده سازی است و همین موضوع دلیل استفاده از این روش در حل مساله مورد نظر است.

در حل این مساله اندازه تورنمنت را ۳ در نظر گرفته ایم.

۲,۳,۴.عملگر همبری

۲,۳,۴,۱.همبری پیشنهادی) در ادامه روشی با تاکید بر جلوگیری از، از بین رفتن طرحواره های متناسب بیان خواهد شد که هدف آن ایجاد تنوع در عملیات جستجو است.

وقتی دو فرزند از والدین خود تولید می شوند، برای هر سه ردیف که زیر شبکه های والدین را تشکیل می دهند، امتیاز کسب می شود و کودک اول از هر زیر شبکه، ردیفی را که بالاترین امتیاز را دارا می باشد، به ارث می برد.
سپس ستون های زیر شبکه به همین ترتیب مقایسه می شوند و کودک دیگر ستونی را که بالاترین امتیاز را دارد، به ارث می برد.

نمونه ای از این همبری در شکل نشان داده شده است. در این شکل فرض می شود، بالاترین امتیاز هر سلول (خانه) برابر با ۹ باشد. بنابراین بالاترین امتیاز هر سطر یا ستون در زیر شبکه ها برابر با ۲۷ خواهد بود.

فرزند ۱، اطلاعات ردیف را از والد ۱ و والد ۲ به ترتیب از بالا به پایین به ارث برده و فرزند ۲ به ترتیب از چپ به راست، اطلاعات والد ۱ و والد ۲ را به ارث می برد.

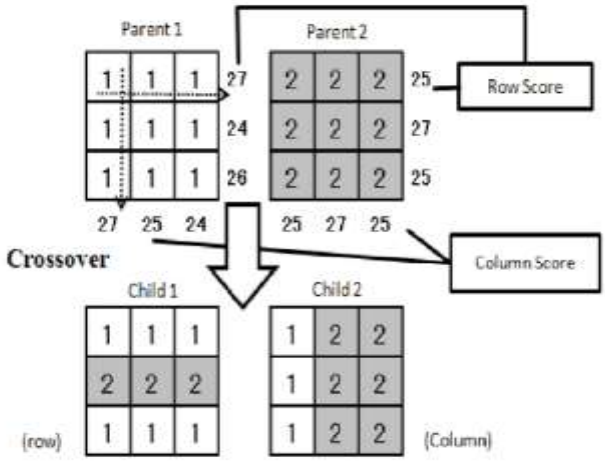


Fig. An example of the crossover considered the rows or the columns that constitute the sub-blocks.

همانطور که ملاحظه می کنید، ما در این حل این مساله، کروموزوم ها را به شکل آرایه های دو بعدی (زیر شبکه ها) در نظر گرفته ایم و همچنین در هنگام تولید فرزند، امتیازات ردیف ها و ستون ها در زیر شبکه ها را در نظر می گیریم، دلیل آن ها این است که از ایجاد یک طرح کاملا مناسب با طول زیاد و تخریب طرحواره های متناسب (Building Blocks) که شامل ردیف یا ستون سلول در زیر شبکه ها است، جلوگیری کنیم.

برای توضیح بهتر یک طرح معمولی (طرحی کاملا مناسب با طول زیاد) میتوان به نمونه ای از آن در شکل بعد، توجه کرد. در این شکل کروموزوم به عنوان یک آرایه تک بعدی از ۸۱ عدد تعریف شده که به ۹ زیر شبکه تقسیم شده است و نقاط همبری فقط می توانند بین زیر شبکه ها ظاهر شوند. از این شکل می توان فهمید که این طرح می تواند از قرار گرفتن نقاط همبری در زیر شبکه ها جلوگیری کند. از طرف دیگر همانطور که گفته شد، این طرح تولید کروموزوم ها، شامل یک طرحواره کاملا مناسب با طول زیاد است که از ردیف های سلول ها یا ستون ها در زیر شبکه ها ساخته می شود. و این طرحواره های بسیار مناسب (BBs) به راحتی با عملگر همبری قابل

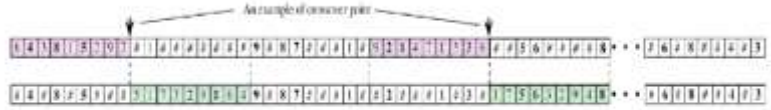


Fig. An example of conventional design of the chromosome and the crossover operation. The chromosome is defined as one-dimensional array of 81 numbers that is divided into nine sub blocks and the crossovers points can only appear between sub blocks.

تخریب هستند.

یکی از مشکلات این نوع طرح، ایجاد نوعی پدیده ی اتومبیلرانی (hitchhiking) است. به عنوان مثال، زیر شبکه هایی که امتیاز پایینی دارند، اما در همسایگی زیر شبکه ای هستند که امتیاز بالایی دارد، به همراه آن زیر شبکه دارای امتیاز بالا به ارث می رسند. بنابراین امتیاز کلی فرزند بهبود نمی یابد. به عبارت دیگر بررسی نمرات هر یک از سه ردیف تشکیل دهنده یک زیر شبکه و انتقال بالاترین امتیاز به فرزند، احتمال کاهش امتیاز بالای طرحواره مناسب (BB) به امتیاز پایین با استفاده از اتومبیلرانی را کاهش می دهد و همچنین انتظار تخریب کمتری از BB که شامل ردیف ها یا ستون های سلول است، نسبت به روش همبری مورد استفاده در مثال بالا، خواهیم داشت.

۲،۳،۴،۲.همبری چند والدی) این نوع همبری تعمیم یافته همبری دو والدی است. روشی که در ادامه ارائه خواهد شد، قابلیت ایجاد فرزند با هر تعداد مشخص از والدین را دارند. اما نتایج بررسی ها در بسیاری از مسائل نشان داده است که همبری های با بیش از چهار والد، سود چندانی ندارد و باعث افزایش محسوس کارایی الگوریتم در قیاس با همبری های دو والدی نخواهد شد.

رویکرد ما برای حل این مساله، بر اساس سه والد است، به طوری که هر یک نشان دهنده یک ردیف از یک راه حل بالقوه است. برای کنترل ارتباط هر یک از والدین در تولید فرزند، با توجه به وزن W تصمیم گیری می کنیم. بدین معنی که تاثیر والدین توسط وزن ها در ماسکی که در روند کار استفاده می شود، نشان داده می شود. جایگزینی مقادیر در کروموزوم فرزند به مقدار ماسک بستگی دارد و با مبادله مقادیر مربوطه، انجام می شود. این فرایند در ادامه تشریح خواهد شد.

والد۱، نشان دهنده یک ردیف از بهترین راه حل همه نسل ها با وزن ۰,۵۵

والد۲، نشان دهنده یک ردیف از بهترین راه حل نسل فعلی با وزن ۰,۳۳

والد۳، نشان دهنده یک ردیف از راه حل فعلی با وزن ۰,۱۲

وزن های داده شده نمایانگر درصد ظاهر شدن والدین در ماسک است. به عنوان مثال، والد۱ پنج بار در ماسک، والد۲ سه بار، و والد۳ یک بار ظاهر می شود. کودک تولید شده به وسیله همبری مرتب سازی چند والدینی به شرح زیر ساخته شده است:

۱–ماسکی از طول والدین با توجه به وزن آن ها به طور تصادفی تولید می شود:

Mask:	3	1	2	1	1	1	2	2
Parent$_1$:	8	4	7	3	6	2	5	1	9
Parent$_2$:	9	1	2	3	4	5	6	7	8
Parent$_3$:	4	7	9	3	6	2	5	1	8

۲–اولین مقدار ماسک با والد۳ مطابقت دارد و سپس اولین مقدار والد۱ و والد۲ باید برابر با اولین مقدار والد۳ باشد که ۴ است. برای این منظور در والد۱ و والد۲، سلول اول با خانه ای که مقدار ۴ را دارد، عوض می شود:

Mask:	3	1	2	1	1	1	2	2	3
Parent$_1$:	4	8	7	3	6	2	5	1	9
Parent$_2$:	4	1	2	3	9	5	6	7	8
Parent$_3$:	**4**	7	9	3	6	2	5	1	8
Child	4								

۳– سپس، مقدار دوم از ماسک برابر با ۱ است. روند مبادله مشابه قبل است:

Mask:	3	**1**	2	1	1	1	2	2	3
Parent$_1$:	4	**8**	7	3	6	2	5	1	9
Parent$_2$:	4	8	2	3	9	5	6	7	1
Parent$_3$:	4	8	9	3	6	2	5	1	7
Child	4	8							

۴– طبق همان روش مراحل قبل، آخرین مرحله در زیر نشان داده شده است که طبق آن فرزند جدید به صورت ۴ ۸ ۲ ۳ ۶ ۷ ۹ ۵ ۱ خواهد بود:

Mask:	3	1	2	1	1	1	2	2	3
Parent$_1$:	4	8	2	3	6	7	9	5	1
Parent$_2$:	4	8	2	3	6	7	9	5	1
Parent$_3$:	4	8	2	3	6	7	9	5	1
Child	4	8	2	3	6	7	9	5	1

شایان ذکر است در حل این مساله، نرخ همبری ۰,۳ در نظر گرفته شده است.

۲,۳,۵.عملگر جهش

جهش مبادله) جهش ها در داخل یک زیر شبکه اعمال می شوند (با فرض نرخ جهش برابر با ۰,۶).

با بررسی سه استراتژی مختلف جهش که شامل

۱–جهش مبادله (swap mutation)،

۲–جهش ۳-مبادله (3-swap mutation)،

۳–جهش درجی (insertion mutation)

هستند و معمولا در بهینه سازی ترکیبی استفاده میشوند، به این نتیجه خواهیم رسید که جهش های ۳-مبادله و درجی در عمل فقط سلسله مراتب جهش مبادله (جهش ۲-مبادله) هستند، بنابراین آن ها را با مراحل جهش مبادله درون یک زیر شبکه جایگزین کردیم. این کار به این دلیل انجام شد که نتایج آزمایش ها نشان می دهند که الگوریتم های ژنتیک به تنهایی با جهش های مبادله عملکرد بهتری نسبت به عملکرد الگوریتم های ژنتیک با جهش های ۳-مبادله و درجی دارند.

در جهش مبادله، مقادیر دو موقعیت رد و بدل می شوند. هر بار که جهش درون یک زیر شبکه امتحان می شود، آرایه کمکی (Help array) از سلول های بدون تغییر (داده های اولیه ی بدون تغییر)، مورد بررسی قرار می گیرد. اگر تغییر موقعیتی که به طور تصادفی انتخاب شده است، غیر قانونی باشد، جهش حذف می شود. علاوه بر این، اگر مراحل جهش را به طور تصادفی ترسیم کرده باشیم، فقط تا جایی که هر دو موقعیت جهش مبادله قانونی باشد، جهش را انجام خواهیم داد. به همین خاطر، با وجود شانس برابر برای ترسیم سلسله مراتب ۱، ۲، ۳، ۴ یا ۵ مبادله ای، بیشتر اوقات سلسله مراتب قطع می شود. گاهی اوقات اولین تلاش برای مبادله، غیر قانونی است و جهشی انجام نمی شود. این بدان معنی است که نرخ جهش واقعی بسیار کمتر از مقداری است که در نظر گرفته ایم.

بررسی ها با استفاده از تعداد متفاوتی مبادله های پی در پی نیز نشان داد که الگوریتم های ژنتیک تنها با یک جهش یک مبادله ای، به خوبی الگوریتم های ژنتیک با جهش های دارای سلسله مراتب ۱، ۲، ۳، ۴ یا ۵ مبادله ای عمل می کند. بنابراین این واقعیت که بیشتر سلسله مراتب های مبادله های طولانی تر، دچار عدم موفقیت نابهنگام می شوند، مضر نیست.

میتوان روند کار جهش مبادله را در شکل مشاهده کرد.

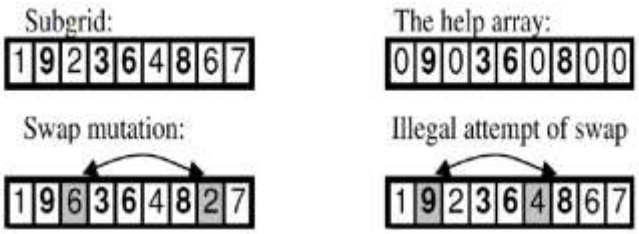


Fig. The swap mutation used in Sudoku optimization. Up left is one sub block and up right the static values of that sub block (6 and 7 are givens). The mutation is applied so that we randomly select positions inside the sub block, and then compare the selected positions to the help array in order to see if the positions are free to change.

قانون خاص دیگری وجود دارد که کنترل می کند جهش انجام شود یا خیر. در این قانون به یک جدول راهنما نیاز داریم که مشخص می کند هر رقم در هر سطر یا ستون چندبار ظاهر می شود. وقتی یک جهش انجام می شود، روی ۱یا ۲ ستون، ۱یا ۲ ردیف و به طور کلی روی ۳ یا ۴ بردار ردیف و ستون تاثیر می گذارد. در شرایط مطلوب، دو رقمی که مبادله (جابجا) می شوند، باید در کل دوبار در این بردارها ظاهر شوند.

در مثال زیر، جهش در هر سلول غیر از سلول ۴ که دارای مقدار ۳ است، مجاز است.

Sudoku problem row: - - - 3 - - - - -
Solution candidate row: 9 7 4 3 6 2 5 1 8

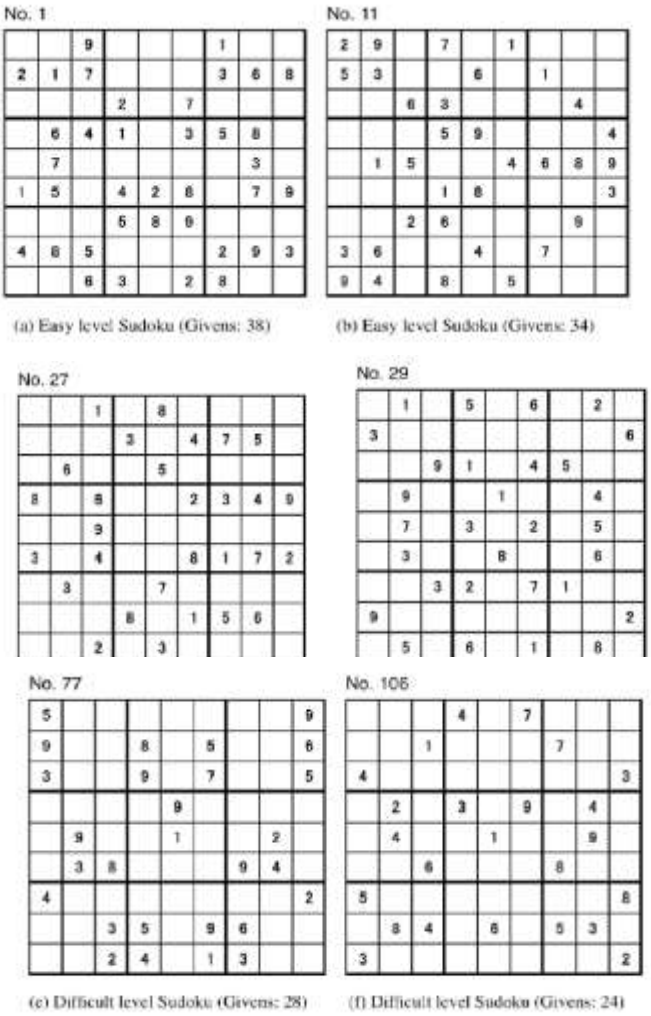
allowed mutation: 9 **6** 4 3 **7** 2 5 1 8
forbidden mutation: 9 7 4 **4** 6 2 5 1 8

۲,۳,۶.تنظیم پارامترها

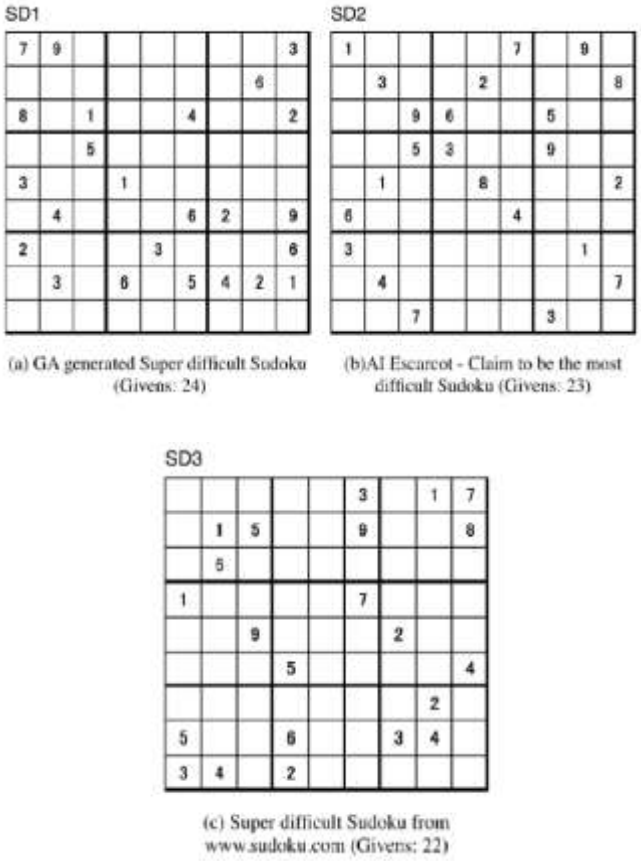
در این بخش به ارائه چندین پازل با سطوح دشواری مختلف و تنظیم پارامترهای مساله می پردازیم.

برای بررسی اثر بخشی عملیات ژنتیکی پیشنهاد شده، دو پازل از هر سطح دشواری مختلف را انتخاب کردیم: پازل های ۱ و ۱۱ از سطح آسان، پازل های ۲۹ و ۲۷ از سطح متوسط و پازل های ۷۷ و ۱۰۶ از سطح دشوار.

در مجموع ۶ پازل. نمونه ای از پازل های مورد استفاده در آزمایش در شکل زیر نشان داده شده است.



برای مقایسه کردن پازل ها با نمونه های مرسوم، میتوانیم از پازل های ویژه دشوار معرفی شده توسط تیمو مانتره هم استفاده کنیم. نمونه ای از این پازل ها را در شکل بعد می بینیم.



تنظیم پارامترهای مساله، نقشی اساسی را در نحوه رسیدن به جواب ایفا می کند، پس باید دقت کافی و لازم را برای تنظیم آن ها به خرج دهیم. پارامترهای این مساله به صورت زیر در نظر گرفته شده اند.

اندازه جمعیت [Population size] ۱۵۰

نرخ همبری [Crossover rate] ۰,۳

نرخ جهش [Mutation rate] ۰,۳

اندازه تورنمنت [Tournament size] ۳

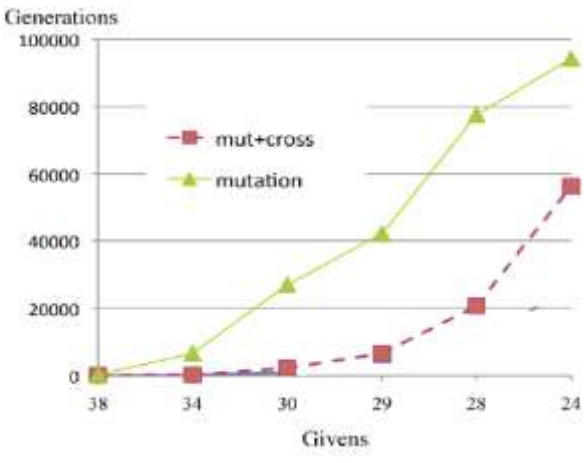
۲,۳,۷.نتایج تجربی

آزمون معیار: رابطه بین تعداد مقدارهای داده شده در نقطه شروع و تعداد نسل های مورد نیاز برای رسیدن به راه حل بهینه در جدول و شکل زیر نشان داده شده است. در این جدول و نمودار دو مورد بررسی می شوند که شامل موردی که فقط عملگر جهش اعمال می شود (نوعی از جستجوی تصادفی) و موردی که عملگر جهش و همبری

پیشنهادی اعمال می شود (mut+cross). آزمایشات ۱۰۰ مرتبه انجام شده و میانگین نتایج مقایسه شده است. نقطه پایان جستجو ۱۰۰۰۰۰ نسل بوده است. اگر راه حلی قبل از نقطه پایان جستجو به دست نیامده باشد، نتیجه ۱۰۰۰۰۰ نسل نمایش داده می شود. هنگامی که جستجو در ۱۰۰۰۰۰ نسل خاتمه می یابد، تناسب بدست آوردن یک راه حل بهینه برای یک پازل دشوار به وسیله افزودن تکنیک همبری پیشنهادی به عملگر جهش به طور واضحی بهبود یافته است. میانگین تعداد نسل ها تا زمان دستیابی به راه حل نیز کاهش می یابد.

Difficulty rating	Givens	mut+cross		Swap mutation	
		Count	Average	Count	Average
Easy (No. 1)	38	100	105	100	223
Easy (No. 11)	34	100	247	96	6627
Medium (No. 27)	30	100	2274	86	26961
Medium (No. 29)	29	100	6609	66	42141
Difficult (No. 77)	28	100	20658	35	77573
Difficult (No. 106)	24	74	56428	9	94314

جدول مقایسه می کند که چگونه الگوریتم های ژنتیک راه حل هایی برای پازل های سودوکو با دشواری های متفاوت پیدا می کند. Count نشان می دهد که چه تعداد از ۱۰۰ اجرای آزمایشی الگوریتم های ژنتیک راه حل را پیدا کرده اند و Average بیان می کند که برای تولید راه حل با ۱۰۰۰۰۰ کوشش، چند نسل از الگوریتم های ژنتیک مورد نیاز بوده است.



نمودار رابطه بین داده های ابتدایی مساله (Givens) و میانگین تعداد نسل های الگوریتم های ژنتیک (Generations) که برای یافتن راه حل مورد نیاز است را نشان می دهد (۱۰۰۰۰۰ کوشش، ۱۰۰ اجرای آزمایشی).

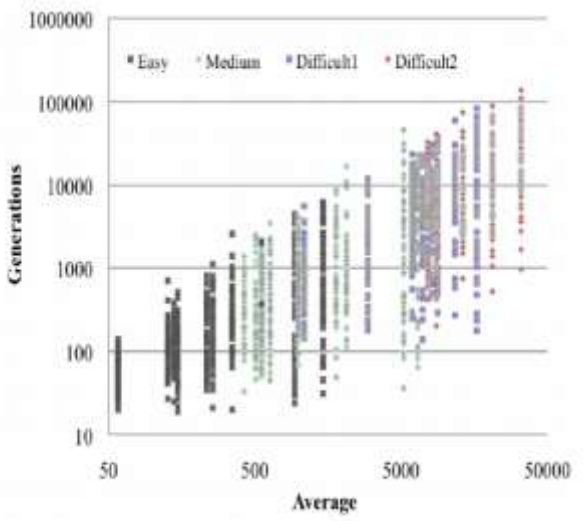
۲,۳,۸.بحث

از جدول و نمودار قبل در می یابیم که روش پیشنهادی همبری موثر است. به عنوان مثال هنگامی که نقطه شروع ۳۰ عدد را فراهم می کند، افزودن روش همبری پیشنهاد شده، تعداد نسل های مورد نیاز برای به دست آوردن راه حل بهینه را در مقایسه با استفاده از عملگر جهش به تنهایی ، تا حدود ۱۰ درصد کاهش می دهد. تفاوت در نتایج دو روش ارائه شده با کاهش تعداد داده های ارائه شده در نقطه شروع، کاهش می یابد. اما دلیل این تمایل این است که جستجو پس از فقط ۱۰۰۰۰۰ نسل خاتمه یافت تا زمان مورد نیاز برای آزمایش کاهش یابد.

دلیل کارآمد بودن این همبری این است که ۱) امکان پذیری مناطق را حفظ می کند، بنابراین فضای کمتری از راه حل ها را با مناطق امکان پذیر جستجو می کند. ۲) از ارزیابی جزئی ردیف ها و ستون ها به ظور حریصانه برای ایجاد فرزندان خوب، استفاده می کند، بنابراین از دانش اضافی مشکل در عملگر جستجو استفاده می کند. ۳) فرزند ۱ و فرزند ۲ به طور مکمل از بهره برداری جداگانه از ردیف ها و ستون ها ساخته می شوند، بنابراین از دو نوع ساختار حریصانه مکمل استفاده می شود.

نموداری که در صفحه بعد قرار دارد، رابطه ی میانگین تعداد نسل ها و پراکندگی را نشان می دهد. برای پازل هایی که تعداد داده های اولیه یکسانی دارند، وابستگی به مکان های داده شده وجود دارد و واریانس زیادی در میانگین تعداد نسل های مورد نیاز برای به دست آوردن راه حل بهینه دیده می شود. علاوه بر این، برای پازل های دشوار که مقادیر اولیه کمی را ارائه می دهند، مواردی وجود داشت که حتی وقتی نقطه پایان جستجو برای

۱۰۰۰۰۰ نسل تعیین شده بود، یک راه حل بدست نیامد. دلیل این نتیجه این است که دامنه جستجوی راه حل یک معمای دشوار زیاد است و راه حل های محلی بسیاری با امتیاز بالا وجود دارد که دور از راه حل مطلوب هستند. احتمال دیگر وجود پازل هایی است که دامنه جستجو برای آن ها بسیار گسترده است و وابستگی به مقادیر اولیه وجود دارد.



شکل نشان دهنده ترتیب دشواری سودوکوی آزمایش شده است که حداقل و حداکثر نسل های مورد نیاز برای حل هر سودوکو از ۱۰۰ اجرای آزمایشی به عنوان تابعی از نسل های مورد نیاز به تصویر کشیده شده است.

۲,۳,۹.ایده هایی برای بهبود بخشیدن به عملکرد الگوریتم

بسته به مسائل مختلف، نظرات و ایده های متفاوتی برای بهبود عملکرد الگوریتم های ژنتیک می توان ارائه داد. معمای سودوکو هم از این قاعده مستثنی نیست و برای بهبود کارایی و تسریع بخشیدن به حل آن می توان ایده هایی را اتخاذ کرد.

۱)پیشنهادی برای جستجوی محلی قدرتمندتر:

به طور کلی الگوریتم های ژنتیک در جستجوی محلی عملکرد خوبی ندارد. بنابراین، سعی ما بر تقویت عملکرد جستجوی محلی با استفاده از جهش برای تولید تعدادی فرزند (که یک عدد صحیح از تعداد والدین است)، بوده است. بعد از تولید فرزندان، تعدادی از آن ها را (به تعداد برابر با تعداد والدین) به ترتیب بالاترین امتیاز، انتخاب می کنیم. مفهوم بهبود قابلیت جستجوی محلی در شکل نشان داده شده است. شکل موردی را نشان می دهد که در هنگام تولید فرزند، فضای بسیار کوچکی در اطراف والدین از یک فضای جستجوی بزرگ استخراج می شود. در صورتی که یکی از والدها فرزندی را تولید کند، ممکن است فرزندی تولید شود که فاصله آن از راه حل بهینه حتی از والدین هم بیشتر باشد، حتی زمانی که والدین از قبل به راه حل بهینه نزدیک شده باشند. بنابراین ممکن است مسیر منتهی به راه حل بهینه، غیر مستقیم باشد. از طرف دیگر، اگر یکی از والدین چندین فرزند تولید کند، پراکندگی در جهت فرزندان نسبت به والد، احتمال کودکی را که به راه حل بهینه نزدیک تر از والد تولید شده است، افزایش می دهد.

انتخاب افراد با امتیاز بالا از بین فرزندان کاندید تولید شده، امکان کارایی خوب در جستجو برای راه حل بهینه را فراهم می کند.

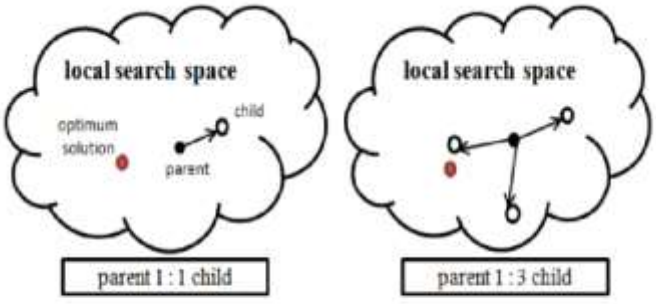


Fig. A concept for improving the local search capability.

حال می توانیم نمودارها و جداول را برزورسانی کنیم و تاثیر افزودن یک جستجوی محلی قدرتمند را در روند دستیابی به راه حل بهینه، مشاهده کنیم.

جدولی که در بخش ۲,۳,۷ ارائه شد، مقایسه می کرد که چگونه الگوریتم های ژنتیک راه حل هایی برای پازل های سودوکو با دشواری های متفاوت پیدا می کند. با افزودن بخش جستجوی محلی به این جدول، آن را دوباره مشاهده خواهیم کرد.

Count نشان دهنده آن است که چه تعداد از ۱۰۰ اجرای آزمایشی الگوریتم های ژنتیک، راه حل را پیدا کرده اند و Average بیان می کند که برای تولید راه حل با ۱۰۰۰۰۰ کوشش، چند نسل از الگوریتم های ژنتیک مورد نیاز بوده است.

و میتوانیم نمودار بروزرسانی رسانی شده بخش ۲,۳,۷ را مشاهده کنیم، در زمانی که معیار جستجوی محلی علاوه بر همبری و جهش (mut+cross+LS) اعمال شده است.

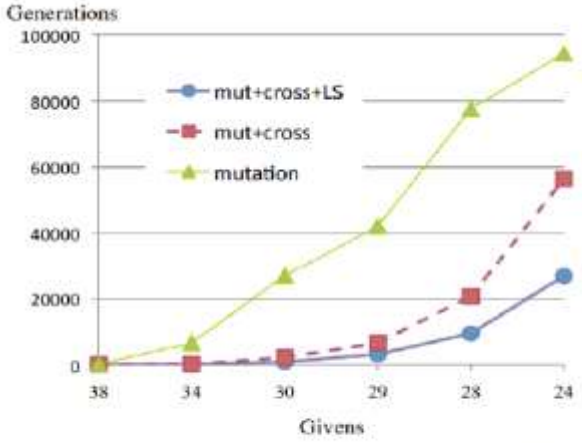


Fig. Relationship between givens and the average of how many GA generations were need to find the solution. (100,000 trials, 100 test runs).

هنگامی که جستجو در ۱۰۰۰۰۰ نسل خاتمه می یابد، با افزودن روش پیشنهادی همبری به عملگر جهش، به دست آوردن یک راه حل بهینه برای یک معمای دشوار به وضوح بهبود می یابد و با افزودن عملکرد جستجوی محلی، حتی بیشتر بهبود خواهد یافت. و همچنین میانگین تعداد نسل ها تا زمان دستیابی به راه حل بهینه نیز تا کمتر از ۵ درصد کاهش می یابد.

۲,۳,۱۰.نتیجه گیری

در این مقاله، بررسی کردیم که آیا می توان با یک الگوریتم ژنتیک ترکیبی، معماهای سودوکو را حل و تولید کرد. نتایج نشان می دهد الگوریتم ژنتیک می تواند

Difficulty rating	Givens	mut+cross+LS		mut+cross		Swap mutation	
		Count	Average	Count	Average	Count	Average
Easy (No. 1)	38	100	62	100	105	100	223
Easy (No. 11)	34	100	137	100	247	96	6627
Medium (No. 27)	30	100	910	100	2274	86	26961
Medium (No. 29)	29	100	3193	100	6609	66	42141
Difficult (No. 77)	28	100	9482	100	20658	35	77573
Difficult (No. 106)	24	96	26825	74	56428	9	94314

معماهای سودوکو را به طور موثر حل کند. با این حال الگوریتم های کارآمدتری برای حل معماهای سودوکو وجود دارد. در این مطالعه هدف این بود که آزمایش کنیم چقدر الگوریتم ژنتیک خالص کارآمد است، بدون بسیاری از قوانین خاص برای حل سودوکو. نتایج الگوریتم ژنتیک را می توان با افزودن قوانین مربوز به مساله افزایش داد. با این حال، اگر یک شخص منطق خاص مساله را بیش از حد به حل سودوکو اضافه کند، دیگر چیزی برای بهینه سازی وجود نخواهد داشت. بنابراین تصمیم گرفتیم بیشتر منطق خاص مساله را حذف کنیم و یک رویکرد بهینه سازی رو به جلوی الگوریتم ژنتیک را امتحان کنیم.

سودوکو آزمایشگاه خوبی برای طراحی الگوریتم هاست. این معما بر اساس یکی از سخت ترین مشکلات حل نشده در علوم کامپیوتر - NP-complete problems است. محققان بر این باورند که ممکن است شوق سودوکو حتی منجر به پیشرفت های چشمگیری در علوم کامپیوتر شود.

اثبات کردیم آن دسته از سودوکوهایی که درجه سختی بالاتری دارند، برای الگوریتم های ژنتیک دشوارتر هستند. این بدان معنی است که می توان از الگوریتم ژنتیک برای

درجه بندی دشواری یک معمای سودوکوی جدید استفاده کرد. شاید فکر کنید حل پازل سودوکو توسط انسان کار آنچنان دشواری نیست، پس برای رایانه هم نباید سخت باشد. اما حقیقت امر این است که حل چنین معماهایی با استفاده از الگوریتم ژنتیک که روشی مبتنی بر رایانه است، کاری دشوار است که ممکن است یک انسان لزوما دشواری آن را به همان روش تجربه نکند.