دانشگاه اصفهان
دانشکده‌ی فنی و مهندسی
گروه کامپیوتر

مروری بر الگوریتم‌های ژنتیک موازی†

نویسنده: اریک کَنتوپاز[1]
مترجم: محسن مؤمنی[2]

چکیده

الگوریتم‌های ژنتیک، روش‌های پرقدرتی برای جست‌وجو هستند که با موفّقیت برای حل مسائل در زمینه‌های مختلفی به‌کار گرفته‌شده‌اند.

الگوریتم‌های ژنتیک موازی، علی‌الخصوص، به راحتی پیاده‌سازی می شوند و کارائی پایداری را تأمین‌می‌کنند. در این زمینه تلاش‌های فراوانی صورت‌گرفته‌است. در این مرور تلاش‌کرده ایم تا بیش‌ترین نمایندگان شاخص منتشرشده از الگوریتم‌های ژنتیک موازی را، در یک روند مشخّص، جمع‌آوری، سازمان‌دهی و ارائه‌کنیم.

برای سازمان‌دهی مقاله، در این‌جا یک دسته بندی از روش‌های مورد استفاده در موازی‌سازی الگوریتم‌های ژنتیک ارائه‌شده و برای هر یک مثالی نشان‌داده‌شده‌است. درهرحال، به خاطر آن‌که عمده‌ی تحقیقات این زمینه، برروی این تکنیک‌ها، متمرکز بوده‌است، در این مرور بر این انواع خاص تمرکزشده‌است.

همچنین مقاله به توصیف مهم‌ترین مسائل در مدل‌سازی و طرّاحی ***PGA***های چندجمعیتی پرداخته و برخی پیش‌رفت‌های اخیر را ارائه‌کرده‌است.

کلمات کلیدی: *الگوریتم‌های ژنتیک موازی، الگوریتم‌های ژنتیک پایه-پیرو، الگوریتم‌های ژنتیک سلسله‌مراتبی، چنددسته‌ای‌ها.*

---

† مشخصات کتابشناختی این مقاله چنین است:

Eric Cantú-Paz, "A Survey of Parallel Genetic Algorithms", 1997, Illinois at Urbana-Champaign.

[1] cantupaz@illigal.gc.uiuc.edu
[2] mohsenmomeni@yahoo.com

۱) مقدّمه

الگوریتم‌های ژنتیک روش‌های کارائی برای جست‌وجو هستند، که برپایه ی مفهوم انتخاب طبیعی و ژنتیک، استوارشده‌اند. این الگوریتم‌ها در حل مسائل فراوانی در تجارت، مهندسی و علم، به شکل موفقیت‌آمیزی به کارگرفته‌شده اند. ***[GOL94]***

الگوریتم‌های ژنتیک در بیش‌تر موارد قادرند در زمان قابل‌قبولی جواب‌های مطلوبی بیابند، امّا در به‌کارگیری آن‌ها، در مسائل بزرگ‌تر و مشکل‌تر، زمان مورد نیاز برای یافتن جواب مقبول، افزایش‌می‌یابد.

چندین روش برای افزایش سرعت الگوریتم‌های ژنتیک وجوددارد و یکی از به‌ترین انتخاب‌ها در این زمینه، پیاده‌سازی موازی آن‌هاست.

مأموریت این مقاله، جمع‌اوری، سازمان‌دهی و ارائه‌ی مهم‌ترین روش‌های مرتبط با موازی‌سازی الگوریتم‌های ژنتیک است. به منظور سازمان‌دهی مقالات چشم‌گیر و روبه‌فزونی دراین‌زمینه، این مقاله دسته‌بندی‌ای از انواع مختلف پیاده‌سازی‌های موازیِ الگوریتم‌های ژنتیک را ارئه‌کرده‌است. محبوب‌ترین موازی‌سازیِ الگوریتم‌های ژنتیک شامل چندین جمعیت است که جداگانه تکامل‌می‌یابند و گه‌گاه افراد خود را با جمعیت دیگری تعویض‌می‌کنند. این دسته از الگوریتم‌های ژنتیک موازی، چنددسته‌ای‌ها نامیده‌می‌شوند.دانه‌درشت یا توزیع‌شده نیز به آن‌ها می‌گویند. این مقاله بیش‌تر بر این دسته‌ی خاص متمرکزمی‌شود؛ امّا به‌هرحال دسته های بزرگ دیگر نیز توصیف می‌شوند و برخی مثال‌ها به طور خلاصه موردبحث‌قرارمی‌گیرند.

مثال‌های فراوانی از کاربرد الگوریتم‌های ژنتیک در یک مسئله‌ی خاص وجوددارد، امّا توجه این مقاله به همه‌ی مثال‌هایی که در آن‌ها الگوریتم‌های ژنتیک موازی در ارائه‌ی راه حل موفق‌بوده‌اند، نیست و درعوض، برای درشت‌نمائی مقالاتی است که الگوریتم های ژنتیک را در بعضی روندها رشد داده‌اند.

خلاصه آن‌که، ما مقالاتی را که راهی جدید یا تلاشی نو را برای به‌کارگیری الگوریتم‌های ژنتیک معرّفی‌کرده‌اند با جزئیات بیش‌تری مورد بررسی قرارمی‌دهیم. ولی درعین‌حال چند مثال برای نشان‌دادن فایده ی الگوریتم‌های ژنتیک در «جهان واقعی» و این‌که این الگوریتم‌ها تنها جنبه‌ی آکادمیک ندارند، ارائه‌می‌شود.

مقاله به صورت زیر سامان‌دهی شده‌است:

بخش بعد، به معرّفی کوتاه الگوریتم های ژنتیک می‌پردازد و بخش ۳، یک دسته بندی کلّی از الگوریتم های ژنتیک موازی را ارائه‌می‌دهد. مرور الگوریتم های ژنتیک موازی از بخش ۴ با ارائه ی نخستین یافته‌ها، آغازمی‌شود. بخش ۵ روش پایه-پیرو در موازی‌سازی الگوریتم‌های ژنتیک را ارائه می‌دهد. بخش‌های ۶ و ۷ مقالاتی که درباره‌ی الگوریتم‌های ژنتیک موازی روی جمعیت وسیع بوده‌اند

را برمی‌رسد و بخش ۸ مروری بر تحقیقات انجام‌شده روی الگوریتم‌های ژنتیک موازی دانه‌ریز دارد. بخش ۹ روش‌های آمیخته برای موازی‌سازی را برمی‌رسد و در بخش ۱۰ بعضی مسائل حل‌شده در مدل‌سازی و طرّاحی مطرح‌می‌شود و پیش‌رفت‌های اخیر بررسی‌می‌گردد. و بالأخره گزارش با ارائه ی خلاصه و جمع بندی این پروژه به پایان‌می‌رسد.

۲) الگوریتم‌های ژنتیک- مقدّمه‌ای کوتاه

در این بخش پیش‌زمینه‌ی لازم در مورد الگوریتم‌های ژنتیک ارائه می‌شود. بعضی واژه‌ها تعریف‌می‌شود و نشان‌می‌دهیم یک الگوریتم ژنتیک ساده چگونه کارمی‌کند. امّا یک مرور کامل صورت‌نمی‌گیرد. خوانندگان علاقه‌مند می‌توانند با مراجعه به کتاب گلدبرگ ***[GOL 89a]***، برای یافتن پیش‌زمینه‌ی بیش‌تر و اطّلاعات جزئی‌تر در مورد الگوریتم‌های ژنتیک اقدام کنند.

الگوریتم‌های ژنتیک، الگوریتم‌های جست وجوی تصادفی[1]ای هستند که برپایه‌ی مفهوم انتخاب طبیعی و بازترکیب استوارند. در آن‌ها تلاش‌می‌شود که حل بهینه‌ای برای مسئله یافته‌شود. این کار با استفاده از ایجاد تغییرات در جمعیتی از راه‌حل‌های برگزیده صورت‌می‌گیرد. جمعیت ارزیابی‌می‌شود و به‌ترین حل‌ها برای بازتولید و ایجاد نسل جدید به‌وسیله‌ی آمیزش بایک‌دیگر، انتخاب‌می‌شوند. پس از گذر چند نسل، خصیصه‌های خوب بر جمعیت حاکم می‌شود، که در نتیجه‌ی آن کیفیت حل‌های یافته‌شده بهبود می‌یابد.

مکانیسم پایه در الگوریتم‌های ژنتیک تکامل داروینی است: خصیصه‌های بد از جمعیت حذف می‌شوند، زیرا در افرادی به چشم می‌خورند که پس از فرآیند انتخاب دیگر در جمعیت باقی نخواهند ماند. خصیصه‌های خوب باقی می‌مانند و با استفاده از بازترکیب آمیخته می‌شوند تا افراد به‌تری را بازآفرینند. جهش نیز در الگوریتم‌های ژنتیک وجوددارد، امّا به عنوان عمل‌گر دوم محسوب‌می شود. این برای آن‌است که مطمئن شویم گونه‌گونی جمعیت ازمیان‌نمی‌رود؛ و بنابراین الگوریتم ژنتیک می‌تواند به جست‌وجوی خود ادامه‌دهد.

اندیشه‌ی خصیصه‌های "خوب" با استفاده از مفهوم بلوک‌های ساختمانی (***BBs***)، که قالب‌های رشته‌ای هستند، که قطعه‌ای کوچک از افراد را می‌سازند و به عنوان یک قسمت تأثیرگذار بر برازندگی افراد عمل می‌کنند. تئوری شایع پیشنهاد می‌کند، که الگوریتم ژنتیک، با ***BBs*** منتشرشونده با استفاده از انتخاب و آمیزش کارکند. ما به دنباله‌روی از گلدبرگ، دِب و تیرنس ***[GOL,93b]*** مفهوم بلوک‌های ساختمانی را به کوچک‌ترین شمای سازنده‌ی بهینه‌ی سراسری محدود می‌کنیم. در این دید، مجاورت

[1] ***stochastic***

دو ***BBs*** از ***O(k)*** در یک رشته‌ی خاص، به ساخت ***BB***یی از ***O(2k)*** منجر نمی‌شود، بلکه به جای آن به دو ***BB*** جداگانه منجر می‌گردد.

اغلب، افراد از یک رشته‌ی دودوئی با طول ثابت ***l*** تشکیل می‌شوند و بدین‌سان الگوریتم‌های ژنتیک، فضای جست‌وجویی متشکل از 2^l نقطه را جست‌وجو می‌کنند. در ابتدا، جمعیت شامل چند نقطه‌ی تصادفی انتخاب شده است، مگر اینکه از راه‌های اکتشافی[1] برای ایجاد حل‌های ابتدائی تقریباً خوب در آن دامنه استفاده کرده باشیم. در مورد اخیر، بخشی از جمعیت کماکان تصادفی ایجاد می‌شود، تا از وجود گوناگونی در میان حل‌ها اطمینان یابیم.

اندازه‌ی جمعیت از اهمیت فراوانی برخوردار است. زیرا بر توانایی دست یابی الگوریتم‌ژنتیک به پاسخ مناسب و زمان دست‌یابی به این پاسخ، اثرمی‌گذارد.***[GOL92,HAR97]*** اگر جمعیت خیلی کوچک باشد، شاید عرصه‌ی مناسبی برای ***BBs*** نباشد و مشکل می‌توان به حل خوب دست یافت. اگر جمعیت خیلی بزرگ باشد، ***GA*** منابع محاسباتی را بی‌جهت تلف می‌کند. این ایجاد تعادل میان کیفیت یک راه حل و زمانی که ***GA*** برای یافتن نیاز دارد، همچنان برای ***GA***های موازی نیز وجود دارد، و در قسمت‌های بعدی می‌بینیم که چگونه بر طراحی آن‌ها تأثیر می‌گذارد.

هر فرد در جمعیت، یک مقدار برازندگی دارد. در حالت کلی، برازندگی یک اندازه‌ی نتیجه‌ی نهایی است که بستگی به این امر دارد که فرد تا چه حد توانایی حل مشکل را دارد. در حالت خاص، ***GA***ها اغلب به عنوان بهینه‌ساز عمل می‌کنند و برازندگی یک فرد مقداری‌ست که تابع مأموریت در نقطه به وسیله‌ی رشته‌ی دودوئی ارائه می‌دهد. بخش انتخاب، میزان برازندگی را برای تعیین افرادی که باز تولید و در آمیزش مشارکت داده می‌شوند، تا جمعیت نسل بعدی را شکل‌دهند، به‌کار می‌گیرد.

GAهای ساده از دو عملگر که تاحدودی بر پایه‌ی ژنتیک طبیعی هستند، برای کاوش فضای جست‌وجو استفاده می‌کند: آمیزش[2] و جهش[3]. **آمیزش** یک مکانیزم ابتدائی کاوش در ***GA***هاست. این عملگر دو فرد را، از میان آن‌هایی که برای تشکیل نسل بعدی انتخاب شده‌اند، تصادفاً بر می‌گزیند و به شکلی تصادفی زیررشته‌هایی از آن‌ها را با یکدیگر جایگزین می‌کند. برای مثال فرض کنید دو رشته‌ی ***A1*** و ***A2*** با طول ۸ را داریم:

A1 = 0 1 1 0 | 1 1 1 1
A2= 1 1 1 0 | 0 0 0 0

l-1 نقطه‌ی آمیزش در رشته‌ای به طول ***l*** وجود دارد. در این مثال، یک نقطه‌ی تکی به عنوان نقطه‌ی آمیزش برگزیده‌ایم، تصادفاً نقطه‌ی ۴ام (که در بالا با نماد | نشان‌اش داده‌ایم.) تعویض زیررشته‌ها حول نقطه‌ی آمیزش دو رشته‌ی جدید را در نسل بعدی ایجاد می‌کند:

---

[1] ***heuristic***
[2] ***crossover***
[3] ***Mutation***

A1' = 0 1 1 0 0 0 0 0
A2' = 1 1 1 0 1 1 1 1

عملگر بازترکیب می‌تواند اشکال متفاوتی داشته باشد. مثال بالا از ***1-point crossover*** استفاده کرده، اما می‌توان از ***2-point***، ***N-point*** یا ***Uniform*** نیز استفاده کرد.

جهش معمولاً به عنوان عملگر کاوش دوم در نظر گرفته می‌شود. کار آن این است که گوناگونی گونه‌ها را، که ممکن است در عملیات تکراری انتخاب و آمیزش از دست رفته باشد، باز یابد.

این عملگر به طور تصادفی برخی از مقادیر رشته را دگرگون می‌کند. برای مثال رشته‌ی

A1 = 1 1 0 1 1

را در نظر بگیرید و فرض کنید که نقطه‌ی ۳ (به تصادف) به عنوان نقطه‌ی جهش برگزیده شده است. رشته‌ی جدید، پس از اعمال این عملگر

A1' = 1 1 1 1 1

خواهد بود. همانند آنچه در طبیعت روی می‌دهد، احتمال روی دادن جهش، در ***GA***ها بسیار پایین است، اما احتمال آمیزش معمولاً بالاست.

چندین راه برای متوقف کردن ***GA*** وجود دارد. یک روش آن است که پس از تولید تعداد معینی نسل، کار متوقف شود. یا ارزیابی‌های تابع زمان پایان‌یافتن الگوریتم را مشخص کند. راه دیگر این است که زمانی کار را متوقف کنیم که میانگین کیفیت جمعیت پس از تعدادی (یک تعداد مشخص) نسل، بهبود نیابد. یک راه جایگزین معمول دیگر آن است که ***GA*** را زمانی متوقف کنیم که همه‌ی افراد همسان شده باشند، که این حالت تنها در صورتی ممکن است رخ دهد، که از عملگر جهش استفاده نشده باشد.

۳) طبقه‌بندی الگوریتم‌های ژنتیک موازی

ایده‌ی ابتدایی در پس بیش‌تر برنامه‌های موازی آن است که کار را به تکه‌های بزرگی[1] تقسیم‌می‌کنند و این تکه‌های بزرگ را هم‌زمان با استفاده از چند-پردازنده‌ها حل‌می‌نمایند. این راهبرد تقسیم‌وفتح[2] می‌تواند در الگوریتم‌های ژنتیک به روش‌های گوناگونی به‌کار گرفته‌شود. در مقالات[3] نمونه‌های فراوانی از پیاده‌سازی موازی موفق الگوریتم‌های ژنتیک می‌توان دید. بعضی روش‌های موازی‌سازی از یک تک‌جمعیت بهره‌می‌گیرند، حال آن‌که بقیه جمعیت را به چندین زیر-جمعیت[4] نسبتاً مجزا[5] تقسیم‌می‌کنند. بعضی روش‌ها می‌توانند، با استفاده از معماری کامپیوترهای موازی توده‌ای، به‌شیوه‌ای

---

[1] ***chunk***
[2] ***Divide and conquer***
[3] ***literature***
[4] ***subpopulation***
[5] ***isolated***

توده‌ای[1] رفتار کنند. دیگر الگوریتم‌ها برای چند-کامپیوترهای با تعداد عناصر پردازش کم‌تر، مناسب‌اند.

رده‌بندی الگوریتم‌های ژنتیک موازی در این قسمت هم‌سان دیگر رده‌بندی‌های انجام‌گرفته است. ***[LIN94, GOR93, ADA94]*** اما در این‌جا رده‌بندی را توسعه‌داده و یک رده‌ی دیگر بر طبقه‌بندی‌های پیشین افزوده‌ایم.

سه نوع اصلی الگوریتم ژنتیک موازی وجود دارد:

۱) الگوریتم‌های ژنتیک پایه-پیرو با یک تک‌جمعیت سراسری[2]

۲) الگوریتم‌های ژنتیک ریز-دانه[3] با یک تک‌جمعیت

۳) الگوریتم‌های ژنتیک درشت-دانه[4] چندجمعیتی

در یک الگوریتم ژنتیک پایه-پیرو، یک تک‌جمعیتِ ***panmictic***[5] وجوددارد. امّا ارزیابی برازندگی[6] میان چندین پردازنده توزیع‌می‌گردد. (شکل ۱ را ببینید.)

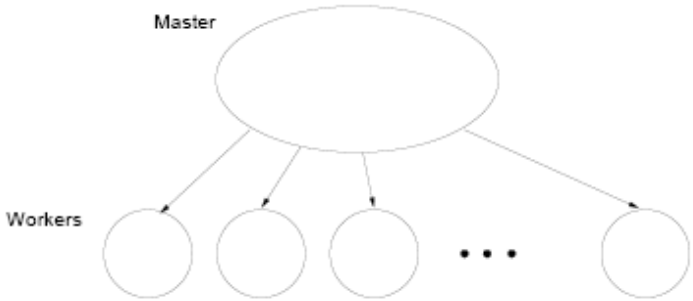


Figure 1. *A schematic of a master-slave parallel GA. The master stores the population, executes GA operations, and distributes individuals to the slaves. The slaves only evaluate the fitness of the individuals.*

از آن جا که در این نوع الگوریتم ژنتیک موازی، انتخاب و آمیزش[7] تمام جمعیت را شامل می‌شود، این نوع با نام الگوریتم ژنتیک موازی سراسری[8] شناخته‌شده‌است.

الگوریتم‌های ژنتیک موازی دانه ریز، برای کامپیوترهای موازی توده‌ای مناسب‌اند و شامل یک جمعیت ساختار-فاصله‌ای[9] هستند. انتخاب و آمیزش[1]، به یک همسایگی کوچک محدود می‌گردد، امّا هم‌پوشانی[2] همسایگی‌ها اجازه‌می‌دهد، تقابل‌هایی بین همه ی افراد به‌وجودآید. (شکل ۲ را ببینید.)

---

[1] ***massively***
[2] ***global***
[3] ***fine-grained***
[4] ***coarse- grained***
[5] مترجم معادل مناسبی برای این واژه نیافت. معنای ***panmictic*** با توجه به متن به تقریب قابل وصول است.
[6] ***Fitness evaluation***
[7] ***Crossover***
[8] ***global pGA***
[9] ***Spatially-structured***

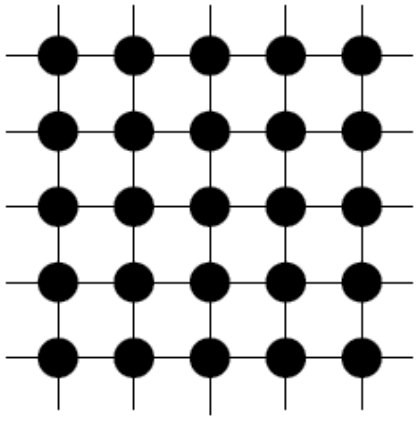


Figure 2. *A schematic of a fine-grained parallel GA. This class of parallel GAs has one spatially-distributed population, and it can be implemented very efficiently on massively parallel computers.*

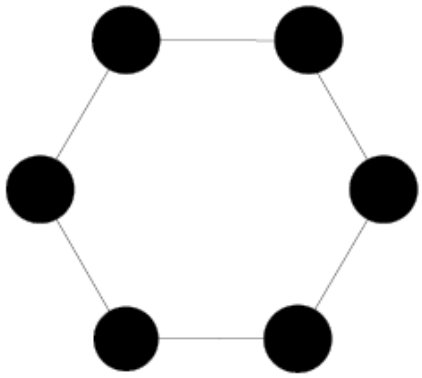


Figure 3. *A schematic of a multiple-population parallel GA. Each process is a simple GA, and there is (infrequent) communication between the populations.*

حالت ایده‌آل این است که تنها یک فرد برای هر عنصر پردازش فراهم[۳] باشد.

الگوریتم‌های ژنتیک جمعیت چند-گانه[۴] یا چندبخشی[۱] حتا از این هم پیچیده‌تر[۲] است. زیرا این دسته شامل چندین زیرجمعیت است که افرادشان را گاه‌وبی‌گاه[۳] مبادله‌می‌کنند. (شکل ۳ را ببینید.)

---

[1] ***Mating***، واژه‌ی آمیزش در این مقاله گاه در مقابل واژه‌ی ***crossover*** و گاه در مقابل واژه‌ی ***mating*** به‌کاررفته‌است. در این متن تفاوت آشکاری میان معنای برآینده از این دو واژه مشاهده نمی‌شود و هر دو واژه به یک مفهوم اشاره‌دارند. البته در صورت توجه دقیق‌تر می‌توانیم تفاوت‌هایی نیز در کارکرد دو واژه شناسایی‌کنیم، که در فهم کلی مسئله قابل چشم‌پوشی است.

[2] ***overlap***

[3] ***available***

[4] ***multiple-population***

این جایگزینی افراد، مهاجرت[4] نامیده‌می‌شود. و آن‌گونه که در فصل‌های بعد خواهیم‌دیدع به وسیله‌ی چندین پارامتر کنترل می‌شود.

الگوریتم‌های ژنتیک چند-بخشی، از همه‌ی انواع دیگر رایج‌ترند. در همین حال این رده از الگوریتم‌های ژنتیک موازی از دیگر رده‌ها برای فهم مشکل‌تر و پیچیده‌ترند، زیرا تأثیرات مهاجرت کاملاً فهمیده نمی‌شود.

الگوریتم‌های ژنتیک چند-بخشی تغییرات بنیادینی را در عملیات الگوریتم‌های ژنتیک مطرح[5] می‌کنند و رفتار متفاوتی نسبت به الگوریتم‌های ژنتیک ساده دارند.

با نام‌های متفاوتی شناخته‌می‌شوند. گاهی با عنوان الگوریتم های ژنتیکِ توزیع‌شده[6] از آن‌ها نام‌می‌برند، زیرا معمولاً در کامپیوترهای با حافظه‌ی توزیع‌شده‌ی ***MIMD*** پیاده‌سازی می‌شوند. از آن جا که محاسبات، نسبت به نرخ ارتباط بالاست، این‌ها گه‌گاه، درشت-دانه نامیده می‌شوند. و درنهایت از این بابت که یک مدل جزیره‌ای[7] را در جمعیت ژنتیک تداعی‌می‌کند[8]، که در آن بخش‌ها نسبتاً منزوی هستند،« الگوریتم ژنتیک موازی جزیره‌ای[9]» نیز نامیده‌می‌شود.

از آن جا که اندازه‌ی بخش‌ها کوچک‌تر از جمعیت مورد استفاده‌ی الگوریتم ژنتیک ترتیبی است، می‌توانیم انتظار داشته‌باشیم که الگوریتم ژنتیک موازی سریع‌تر همگرا[10] شود. اما باید در مقایسه‌ی کارائی الگوریتم‌های ژنتیک ترتیبی و موازی، کیفیت حل یافت‌شده در هر مورد را نیز مدنظر داشته‌باشیم. بنابراین، با وجود این‌که هم‌گرایی سریع‌تر بخش‌های کوچک‌تر نکته‌ی صحیحی است، این‌که کیفیت حل‌های یافت‌شده با استفاده از بخش‌های کوچک‌تر ممکن است بدتر[11] و نامرغوب‌تر باشد نیز امری صحیح است.

بخش ۱۱، یک تئوری جدید را که کیفیت حل را برای بعضی مواردِ افراطی[12] پیش‌بینی می‌کند -و به ما امکان قضاوت منصفانه‌تر در مورد آن‌ها و الگوریتم‌های ژنتیک ترتیبی می‌دهد- ارائه‌شده‌است.

توجه به این نکته حائز اهمیت است که، در حالی‌که الگوریتم ژنتیک موازی پایه-پیرو روی رفتار الگوریتم تأثیرگذار نیست، دو روش دیگر راهکار الگوریتم ژنتیک را تغییرمی‌دهند.

---

[1] ***multiple-deme***
[2] ***sophisticated***
[3] ***occasionally***
[4] ***migration***
[5] ***introduce***
[6] ***distributed***
[7] ***island model***
[8] ***resemble***
[9] ***island pGA***
[10] ***converge***
[11] ***poorer***
[12] ***extreme***

برای مثال، در الگوریتم‌های ژنتیک موازی پایه-پیرو، مکانیسم انتخاب همه‌ی جمعیت را به‌حساب می‌آورد، امّا در دو روش دیگر، انتخاب تنها در زیرمجموعه‌ای از جمعیت صورت‌می‌پذیرد. هم‌چنین، در مدل پایه-پیرو، هر دو فرد موجود در جمعیت امکان آمیزش با یک‌دیگر را دارند، امّا در روش‌های دیگر، آمیزش در یک زیرمجموعه از افراد محدود است.

روش پایانی برای موازی‌سازی الگوریتم‌های ژنتیک، چند-بخشی را با پایه-پیرو یا ریز-دانه ترکیب‌می‌کند. ما این رده از الگوریتم‌ها را، الگوریتم‌های ژنتیک موازی سلسله‌مراتبی می‌نامیم، زیرا در سطح بالاتر چند-بخشی، و در سطح پایین‌تر تک-جمعیتی –آن‌گونه که در پایه-پیرو و ریز-دانه دیده‌ایم– هستند.

یک الگوریتم ژنتیک موازی سلسله‌مراتبی، مؤلفه‌های سودمند هر یک از راهبردها را ترکیب‌می‌کند و نوید کارائی به‌تر از هرکدام از آن‌ها به‌تنهایی را می‌دهد.

بخش ۱۰ مروری بر نتایج پیاده‌سازی ***hPGA***ها دارد.

۴) مطالعات اوّلیه

به‌رغم همه‌ی رشد توجهاتی که روی محاسبات موازی در ۱۵ سال اخیر، می‌بینیم، ایده‌ی ساختن کامپیوترهای موازی توده‌ای، عمر بیش‌تری دارد.

برای مثال در اواخر دهه‌ی ۵۰ میلادی جان هلند، یک معماری برای کامپیوترهای موازی پیشنهاد کرد که می‌توانست تعداد نامعینی از برنامه‌ها را هم‌روند اجرا کند. ***[HOL59,HOL60]***

سخت‌افزار کامپیوتر هلند متشکل از ماژول‌های به‌هم‌پیوسته‌ی نسبتاً ساده‌ای بود و برنامه‌ای که روی آن اجرا می‌شد می‌توانست آن‌ها را چنان‌که اجراشونده بودند، تغییر دهد...

یکی ز کاربردهایی که هلند، برای این کامپیوتر در نظر داشت، شبیه‌سازی تکامل جمعیت‌های طبیعی بود. «کامپیوترهای مبتنی بر مدار تکرارشونده»، به این اسم هرگز ساخته نشدند، امّا بعضی سخت‌افزارهای کامپیوتر موازی توده‌ای در دهه‌ی ۱۹۸۰ (مثلاً***Connectio Machinels***) از مفهوم ماژولار یکسانی با کامپیوترهای هلند بهره گرفتند. نرم‌افزار در هر حال، در کامپیوترهای اخیر، تا حد زیادی انعطاف‌پذیر نیست.

یکی از نخستین مطالعات در چگونگی موازی‌سازی الگوریتم‌های ژنتیک به‌وسیله‌ی بتکه ***[BET76]*** هدایت شد. او پیاده‌سازی موازی (سراسری)ای از یک الگوریتم ژنتیک ***conventional***(پیچشی) توصیف‌کرد و در مورد الگوریتم ژنتیک با یک گپ نسلی (توضیح، این یکی تنها قسمتی از جمعیت‌اش را در هر نسل جایگزین می‌نمود.)، و به تحلیل کارایی[1] دو الگوریتم پرداخت. تحلیل‌های او نشان داد که

[1] ***efficiency***

pga سراسری در کامپیوترهای ***SIMD*** می‌تواند کارایی ٪۱۰۰ داشته‌باشد. او همچنین به تحلیل بهینه‌سازهای مبتنی بر شیب[1] پرداخت و برخی گلوگاه‌ها، که کارایی الگوریتم موازی را محدود می‌ساخت، را مشخص کرد.

دیگر تحقیق اولیه بر چگونگی تصویر نمودن، الگوریتم‌های ژنتیک روی معماری‌های کامپیوتر موجود به‌وسیله‌ی گرِفنْسْتِت ***[GRE81]*** انجام گرفت. او چهار پیش‌ساخت[2] برای ***pga***ها پیشنهاد کرد. سه تای نخست انواعی از شمای پایه-پیرو بودند و چهارمی یک ***m-pga*** بود. در پیش‌ساخت نخست، پردازنده‌ی پایه[3] جمعیت را ذخیره‌می‌کند و والدین نسل بعدی را برمی‌گزیند و آمیزش[4] و جهش را اعمال می‌کند. افراد برای ارزیابی به پردازنده‌های پیرو فرستاده می‌شوند و در پایان هر نسل به پردازنده‌ی پایه برمی گردند. پیش‌ساخت دوم بسیار شبیه اولی است، اما تقسیم واضحی بین نسل‌ها وجود ندارد. هنگامی که هر پروسسور ارزیابی خود را به پایان می‌رساند، پردازنده‌ی پیرو، فردِ[5] خود را به پردازنده‌ی پایه برمی گرداند و فرد دیگری دریافت می‌کند. این شِما نیاز به همگام‌سازی را در هر نسل نادیده‌می‌گیرد و می‌تواند بهره‌وری[6] را در سطح بالایی برای پردازنده‌ها تأمین کند. (پردازنده‌ها را در سطح بالایی از بهره‌وری نگه‌دارد. —م)حتا اگر پردازنده‌های پیرو با سرعت‌های متفاوتی کارکنند.

در پیش‌ساخت سوم جمعیت در یک حافظه‌ی اشتراکی ذخیره می‌شود، که می‌تواند به وسیله‌ی پیروها ارزیابی شود[7]، که افراد را ارزیابی می‌کنند و عملگرهای ژنتیکی را مستقل از یک‌دیگر اعمال می‌کنند.

پیش‌ساخت چهارم گرفنستت امّا یک ***m-pga*** است که به‌ترین افراد در هر پردازنده در هر نسل، به همه‌ی پردازنده‌های دیگر، پراکنده (***broadcast***) می‌شوند. پیچیدگی ***multiple-deme pga*** از این نخستین پیشنهاد آشکار شد و گرفنستت چندین پرسش جالب درباره‌ی بسامد[8] مهاجرت، مقصد مهاجران، توپولوژی و تذثیر مهاجرت بر پیش‌گیری از هم‌گرایی زودرس[9] مطرح‌نمود (آن‌ها را برجسته نمود. —م). این تنها پیش‌ساختی بود که چند وقت بعد پیاده‌سازی شد. ***[PET 87a, PET 87b]*** و ما مروری بر برخی نتایج آن در بخش ۶ خواهیم داشت. گرفنستت هم‌چنین به امکان ترکیب پیش‌ساخت چهارم، با هر یک از سه پیش‌ساخت دیگر به اجمال اشاره‌کرده‌بود؛ آن چه موجب خلق ***pga***ِ سلسله‌مراتبی می‌گردد.

---

[1] ***Gradient-based optimizer***
[2] ***prototype***
[3] ***Master proccessor***
[4] ***Cross-over***
[5] ***individual***
[6] ***utilization***
[7] ***assess***
[8] ***frequency***
[9] ***Preventing premature convergence***

۵) موازی‌سازی پایه-پیرو

این بخش به مرور روش موازی‌سازی پایه‌پیرو (یا سراسری) می‌پردازد. الگوریتم در این روش از یک تک جمعیت بهره‌می‌گیرد و ارزیابی افراد و/یا کارکرد عملگرهای ژنتیکی موازی انجام‌می‌شود.

همانند آن چه که در الگوریتم ژنتیک سریال شاهدیم، در اینجا نیز، امکان رقابت[1] و آمیزش[2] هر فرد با هر یک از افراد دیگر وجود دارد. (این گزینش و آمیزش سراسری است.)

الگوریتم‌های ژنتیک موازی سراسری[3] معمولاً به صورت برنامه‌های پایه-پیرو پیاده‌سازی می‌گردند، که در آن‌ها پایه جمعیت را ذخیره می‌کند و پیروها ارزیابی برازندگی[4] را به عهده دارد.

بیش‌تر عملیات معمولی که موازی‌سازی می‌شود، ارزیابی افراد است. زیرا برازندگی افراد، مستقل از باقیمانده‌ی جمعیت است و هیچ نیازی به ارتباط در این فاز وجود ندارد.

ارزیابی افراد، با انتساب یک کسر[5] از جمعیت به هر یک از پردازنده‌های دردسترس، موازی‌سازی می‌شود. ارتباط تنهایی در زمان دریافت زیرمجموعه‌ی افراد توسط هر پیرو برای ارزیابی و برگشت‌دادن مقدار برازندگی[6] آن‌ها، اتفاق می‌افتد.

اگر الگوریتم متوقف شود و منتظر دریافت مقادیر قابلیت برای همه‌ی جمعیت بماند، پیش از اینکه به نسل بعدی روی آورد، الگوریتم همگام است.

یک الگوریتم ***M-S PGA*** همگام خصوصیات دقیقاً مشابهی با ***GA*** ساده دارد و تنها در سرعت با آن متفاوت است. در هر حال امکان پیاده‌سازی یک ***M-S PGA*** ناهمگام نیز وجود دارد،که در آن الگوریتم در انتظار پردازنده‌های کُند نمی‌ماند و متوقف نمی‌شود، اما عملکرد آن دقیقاً شبیه ***GA*** ساده نیست.

بیش‌تر ***gpGA***ها، همگام پیاده‌سازی شده‌اند و در کل این مقاله فرض بر این است که ***gpGA*** دقیقاً همانند ***simple GA*** کار می‌کند. مدل موازی‌سازی سراسری هیچ پیش‌فرضی در مورد معماری کامپیوتر ندارد، و می‌تواند به صورت مؤثر روی مدل‌های حافظه‌ی اشتراکی و حافظه‌ی توزیع‌شده پیاده‌سازی شود. روی چند پردازنده‌ی حافظه اشتراکی، جمعیت می‌تواند در حافظه‌ی اشتراکی ذخیره شود و هر پردازنده می‌تواند افرادی که منتسب به آن هستند را بخواند و نتیجه‌ی ارزیابی را باز بنویسد، بدون اینکه کشمکشی[7] پیش آید.

روی یک کامپوتر با حافظه‌ی توزیع‌شده، جمعیت می‌تواند در یک پردازنده ذخیره شود . این «پردازنده‌ی پایه» مسئول آن است که جداگانه افراد را به پردازنده‌های پیرو برای ارزیابی ارسال کند،

---

[1] ***compete***
[2] ***mate***
[3] ***Global PGAs***
[4] ***fitness***
[5] ***fraction***
[6] ***Fitness value***
[7] ***conflict***

نتایج را جمع‌آوری نماید و عملگرهای ژنتیکی را اعمال کند تا نسل جدید تولید شود. تعداد افراد منتسب به هر پردازنده می‌تواند ثابت باشد، اما در بعضی موراد (مثل محیط چند کاربره که کارایی هر پردازنده متغیر است) شاید لازم باشد که تعادل بار محاسباتی بین پردازنده‌ها با استفاده از یک الگوریتم زمان‌بندی پویا ایجاد گردد. (مثلاً ***Guided self Scheduling***)

فوگارتی و هُآنگ ***[FOG91]*** تلاش کرده‌اند، مجموعه‌ای از قوانین، برای برنامه‌های کاربردی متعادل‌سازی قطب، که زمان قابل ملاحظه‌ای در شبیه‌سازی صرف آن می‌شود، را استنتاج نمایند. آن‌ها از شبکه‌ای از ***transputer***ها که ریزپردازنده‌هایی با طراحی خاص برای محاسبات موازی هستند، بهره گرفتند. یک ***transputer*** می‌تواند مستقیماً به تنهاً چهار ***transputer*** دیگر متصل گردد و ارتباط بین گره‌های دلخواه، در آن‌ها با استفاده از بازارسال[1] پیام‌ها، میسر می‌گردد. این امر باعث ایجاد سربار زیادی در ارتباط‌ها می‌گردد، و در تلاش برای کمینه کردن آن، فوگارتی و هآنگ ***transputer***ها را با توپولوژی‌های مختلف به یکدیگر متصل نمودند. آن‌ها بدین نتیجه رسیدند که پیکربندی شبکه، تغییر قابل ملاحظه‌ای ایجاد نمی‌کند. آن‌ها به تسریع معقولی دست یافتند، اما رشد زیاد سربار ارتباطی را به عنوان مانعی در مقابل بهبود بیش‌تر سرعت، تشخیص دادند.

آبرامسون و ابلا ***[ABR 92]*** الگوریتم ژنتیکی را روی یک کامپیوتر حافظه-اشتراکی[2] (یک ***EncoreMultimax*** با ۱۶ پردازنده) برای جست‌وجوی تقویم زمانی مؤثری برای مدارس پیاده‌سازی نمودند. گزارش آن‌ها حاکی از تسریعی محدود بود، و عیب اصلی در آن این بود، که برخی از بخش‌های سریال، که در مسیر بحرانی[3] برنامه قرار داشت، موجب این نتیجه می‌گردید. کمی بعدتر، آبرامسون، میلز و پرگینز ***[ABR93]*** یک ماشین با حافظه‌ی توزیع شده (یک ***Fujitsu AP1000*** با ۱۲۸ پردازنده) را به آزمایشات خود افزودند و برنامه‌ی کاربردی خود را متناسب با آن، برای آماده‌سازی ***timetable***ها تغییر دادند و کد آن را اصلاح نمودند. آن‌ها تسریع قابل ملاحظه (و تقریباً مشخصی) را تا حدود ۱۶ پردازنده، روی دو کامپیوتر، گزارش داده‌اند، اما تسریع با افزوده شدن بیش‌تر پردازنده‌ها به طرز قابل ملاحظه‌ای افت می‌کند، که بیش از هر چیز به خاطر افزایش ارتباط‌هاست.

اخیراً یک پیاده‌سازی از الگوریتم ژنتیک سراسری، توسط هاوسر و مانر ***[HAU94]*** انجام گرفته است. آن‌ها از سه کامپیوتر موازی مختلف، اما حائز تسریع خوب (تسریع ۵ با استفاده از ۶ پردازنده) بر روی یک ریزپردازنده‌ی ***NERV*** که سربار ارتباطی پایینی دارد، استفاده کرده‌اند. آن‌ها توضیح داده‌اند، که کارائی پایین و ضعیف دیگر سیستم‌های استفاده شده (یک ***SparcServer*** و یک ***KSRI***) در برنامه‌ریزی نامناسب نخ‌های محاسباتی به وسیله‌ی سیستم بوده است.

---

[1] ***retransmit***
[2] ***Shared-Memory***
[3] ***Critical-path***

جنبه‌ی دیگر الگوریتم‌های ژنتیک که می‌تواند موازی‌سازی شود، به کار گیری عملگرهای ژنتیکی است. آمیزش و جهش می‌تواند با استفاده از ایده‌ای مشابه در بخش‌بندی جمعیت و توزیع کار میان پردازنده‌های چندگانه موازی‌سازی شود. به هر حال، این چنان ساده‌اند، که احتمالاً زمان مورد نیاز برای ارسال افراد و بازگشت آن‌ها، هر دست‌اوردی را در کارائی، بازپس می‌زند.

سربار ارتباطی، به علاوه مانع از موازی‌سازی انتخاب می‌گردد. بیش‌تر به این خاطر که اغلب اطلاعات مورد نیاز برای انتخاب، در مورد همه‌ی جمعیت است و بنابراین نیاز به ارتباطاتی گسترده دارد. اخیراً، برانک ***[BRA97]*** انواع گونه‌گونی از انتخاب سراسری را بر روی یک ***grid*** دوبُعدی از پردازنده‌ها موازی‌سازی نموده است و نشان داده که الگوریتم‌های آن‌ها، برای توپولوژی‌ای که استفاده کرده‌اند، بهینه است. (الگوریتم $O(\sqrt{n})$ گام زمانی روی یک ***grid*** $\sqrt{n} * \sqrt{n}$ وقت می‌گیرد.)

در مجموع، پیاده‌سازی الگوریتم‌های ژنتیک موازی پایه-پیرو، آسان است و وقتی که ارزیابی نیاز به محاسبات قابل توجهی دارد، این روش می‌تواند روشی بسیار مؤثر برای موازی‌سازی باشد. به‌علاوه، این روش، مزیت دیگری دارد و آن عدم ایجاد تغییر در رفتار جست‌وجوی الگوریتم ژنتیک است، که به همین خاطر می‌توانیم همه‌ی تئوری‌های در دسترس برای الگوریتم ژنتیک ساده را در مورد آن نیز به کار بریم.

۶) الگوریتم‌های ژنتیک موازی چنددسته‌ای- نسل اوّل

مهم‌ترین خصوصیت‌های الگوریتم‌های ژنتیک موازی چندجمعیتی، استفاده از زیرجمعیت‌های نسبتاً بزرگ و مهاجرت است. الگوریتم‌های ژنتیک چندجمعیتی، یکی از معمول‌ترین روش‌های موازی‌سازی هستند، و مقالات فراوانی در توصیف جنبه‌های گوناگون و جزئیات پیاده‌سازی آن‌ها نوشته شده است. مرور همه‌ی این مقالات، به وضوح، غیرممکن است و بنابراین در این قسمت، تنها مهم‌ترین و تأثیرگذارترین مقالات و چند مثال کوچک از پیاده‌سازی‌ها و کاربردهای خاص آن‌ها، مورد بررسی قرار می‌گیرد.

شاید نخستین مطالعه‌ی سیستماتیک الگوریتم‌های ژنتیک موازی چندجمعیتی، مطالعه‌ی گروسو ***[GRO85]*** باشد. هدف او شبیه‌سازی تعامل چندین زیرمؤلفه‌ی موازی از یک جمعیت در حال تکامل بود. گروسو افراد دیپلوئیدی (دو لا) را شبیه‌سازی نمود. (که در آن‌ها برای هر ژن، دو زیر مؤلفه وجود داشت) و جمعیت به ۵ قسمت، تقسیم شده بود. هر قسمت افراد را با همه‌ی قسمت‌های دیگر با نرخ مهاجرت ثابتی مبادله می‌نمود.

با آزمایش‌های کنترل شده، گروسو دریافت که میانگین برازندگی جمعیت در چند قسمت کوچک‌تر، بهبود قابل ملاحظه‌ای نسبت به یک جمعیت بزرگ ***panmictic*** (تقسیم‌نشده) دارد.

این امر مفهوم مهمی را در جمعیت‌های ژنتیکی بیان می‌نمود: هنگامیکه قسمت‌ها کوچک باشند، ویژگی‌های مطلوب سریع‌تر رشد می‌کنند، تا وقتی که قسمت‌ها بزرگ باشند. اما او این را هم دید که هنگامی‌که قسمت‌ها ایزوله باشند، رشد سریع برازندگی در یک ارزش برازندگی پایین‌تر (نسبت به جمعیت بزرگ) متوقف می‌شود. به عبارت دیگر، کیفیت حل یافت‌شده، پس از تقاطع[1] در مورد جمعیت ایزوله بدتر از زمانی است که یک تک‌جمعیت داشته باشیم.

با یک نرخ مهاجرت پایین، قسمت‌ها هنوز رفتار مستقلی دارند و نواحی متفاوتی را می‌کاوند. مهاجران اثر مهمی بر جمعیت ندارند و کیفیت حل همانند زمانی است که قسمت‌ها ایزوله هستند. اما در مهاجرت‌های با نرخ متوسط، جمعیت‌های تقسیم شده، حل‌هایی شبیه آنچه در جمعیت ***panmictic*** یافت می‌شود، می‌یابند. این مشاهدات نشان می‌دهد که یک کف نرخ مهاجرت بحرانی وجود دارد، که پایین‌تر از آن، کارائی الگوریتم، به وسیله‌ی ایزوله‌سازی قسمت‌ها، پیشگیری می شود، و سقفی که جمعیت‌های بخش‌بندی شده حل‌هایی هم‌کیفیت با جمعیت ***panmictic*** می‌یابند.

در یک الگوریتمِ ژنتیکِ موازی مشابه، پتی، لوز و گرفنست ***[PET87a]*** یک کپی از به‌ترین فرد یافته شده در هر قسمت را، پس از تولید هر نسل، به همه‌ی همسایه‌ها می‌فرستادند. هدف از این ارتباطات ثابت، اطمینان از این بود که آمیختگی خوبی از افراد به وجود آید. همانند گروسر، نویسندگان این مقاله نیز دریافتند که الگوریتم‌های ژنتیک موازی با چنین سطح بالایی از ارتباطات، حل‌هایی هم‌کیفیت با الگوریتم ژنتیک ترتیبی‌ای که جمعیتی ***panmictic*** و بزرگ داشت را ارائه می‌دهد.

این مشاهدات سؤالاتی را پیش آورد، که تا امروز حل‌نشده باقی مانده‌اند:

۱) چه سطحی از ارتباط برای هم‌رفتار نمودن ***PGA*** با ***panmicticGA*** لازم است؟

۲) هزینه‌ی چنین ارتباطی چقدر است؟

۳) آیا هزینه‌ی ارتباط آنقدر اندک هست که این روش را به آلترناتیو مناسبی برای طراحی الگوریتم ژنتیک موازی بدل کند؟

برای آنکه بتوانیم به چنین سؤالاتی پاسخ دهیم، تحقیقات بیش‌تری برای فهم آثار مهاجرت بر کیفیت جست‌وجو در ***PGA*** لازم است.

جالب این است که چنین مشاهدات مهمی بسیار پیش از این انجام شده، در حالی که در همان زمان، مطالعات سیستماتیک بر الگوریتم‌های ژنتیک موازی، بدون سبک بوده است.

مثلاً تانس ***[TAN87]*** الگوریتم ژنتیک موازی‌ای با دسته‌های متصل در یک توپولوژی اَبَرمکعب[2] چهاربُعدی، پیشنهاد می‌دهد.در الگوریتم تانش، مهاجرت در ورودی‌های ثابتی میان پردازنده‌ها در یک بُعد از ابرمکعب روی می‌دهد. مهاجران به شکل احتمالی، تصادفاً، از میان به‌ترین افراد یک زیرجمعیت،

---

[1] ***convergence***
[2] ***Hyper cube***

برگزیده شده و با به‌ترین افرادی که در یک قسمت یافته می‌شوند، جایگزین می‌گردند. تانس، سه مجموعه آزمایش انجام داد. در نخستین مجموعه، فاصله‌ی زمانی[1] بین مهاجرت‌ها پنج نسل قرار داده شد و تعداد پردازنده‌ها متفاوت بود. در تست‌هایی با مهاجرت با سرعتی دو برابر و متغیر بودن تعداد پردازنده‌ها، الگوریتم ژنتیک موازی، نتایجی با کیفیت مشابه الگوریتم سریال یافت. اما فهمیدن اینکه آیا الگوریتم موازی جواب‌ها را سریع‌تر از الگوریتم سریال می‌یابد، در این آزمایش‌ها مشکل بود، زیرا دامنه‌ی زمانی بسیار بزرگ بود. در مجموعه‌ی دوم آزمایش‌ها، تانس نرخ امیزش و جهش در دسته‌ها را تغییر داد، تا مقادیر پارامترها را در تعادل کاوش[2] و استخراج[3] بیابد. مجموعه‌ی سوم آزمایش‌ها به مطالعه‌ی اثر بسامد تبادل بر جست‌وجو می‌پرداخت و نتایج نشان دادند که مهاجرت با بسامد بسیار بالا یا بسیار پایین، کارایی الگوریتم را کاهش می‌دهد.

همچنین در همان سال، کُهون، هج، مارتین و ریچاردز، تشابهات دقیقی میان تکامل حل‌ها در ***pGA*** و در تئوری ***punctuated equilibria [COH87]*** را یاداوری کرده‌اند. این تئوری توسط اِلرِج و گولد برای توضیح حلقه‌ی گم‌شده در آثار فسیلی پیشنهاد شده است. و نشان می‌دهد که در بیش‌ترین اوقات زمان، جمعیت‌ها در تعادل بوده‌اند. (برای مثال، در ترکیب ژنتیکی آن تغییر مهمی وجود ندارد.) امّا این تغییرات بر محیط می‌تواند توسط یک تغییر سریع تکاملی شروع شود. یک مؤلفه‌ی مهم برای محیط یک جمعیت ترکیب خود آن است. زیرا افراد در جمعیت باید برای منابع با دیگر اعضا به رقابت بپردازند. بنابراین، زنده‌ماندن افراد از دیگر جمعیت‌ها می‌تواند منجر به تعادل شود و باعث تغییرات تکاملی گردد. در آزمایشات آنان با ***pGA***ها، کُهون و دیگران، نشان دادند که نسبتاً تغییرات اندکی در زمان مهاجرت‌ها وجود دارد، اما حل‌های جدید، مدت کوتاهی پس از مبادله‌ی افراد، سروکله شان پیدا می‌شود.

کُهون و دیگران الگوریتم‌ها را با یک مسئله‌ی خطی کاریابی محک زدند و با اتصال دسته‌ها با استفاده از توپولوژی مِش، آزمایش را انجام دادند. به هر حال انان دریافتند که انتخاب توپولوژی، در قیاس با "اتصال شدید و قطر پایین برای اطمینان از مناسب بودن نرخ آمیزش[4] در پیشروی زمان"، تقریباً در کارائی الگوریتم، اهمیتی ندارد. توجه داشته باشید که توپولوژی‌های منتخب آنان چگال-متصل‌شده[5] بود و همچنین قطر کمی داشت (***o(√r)***، که ***r*** تعداد دسته‌هاست.) این کار سپس به وسیله‌ی همان گروه با استفاده از کاربردهای ***VLSI***، در یک توپولوژی ابر-مکعب چهار بُعدی، در مورد مسئله‌ی بخش‌بندی گراف، پی گرفته شد. ***[COH91a, COH91b]***. همانند توپولوژی مِش، گره‌ها در ابر-مکعب ۴ همسایه دارند، اما قطر ابر-مکعب کم‌تر از مش است. (***lg_2 r***).

---

[1] ***interval***
[2] ***exploration***
[3] ***exploitation***
[4] ***mixing***
[5] ***Densely-connected***

خانم تانس کار خود را با مطالعات آزمایشگاهی بسیار خسته‌کننده‌ای، درباره‌ی بسامد مهاجرت و تعداد و افراد ردوبدل شده در هر زمان، ادامه داد. ***[TAN89a]*** او کارائی ***GA*** سریال را در برابر ***pGA*** (با و بدون داشتن ارتباط) مقایسه نمود. همه‌ی الگوریتم‌ها اندازه‌ی جمعیت مشابهی داشتند (۲۵۶ فرد) و در ۵۰۰ نسل اجرا می‌شدند. تانس دریافت که یک ***m-dGA*** (الگوریتم ژنتیک چندجمعیتی) بدون ارتباط و در ***r*** قسمت که هر یک ***n*** فرد دارند، می‌تواند-در برخی نقاط در طول اجرا- حل‌هایی هم‌کیفیت با ***GA*** سریال تک‌جمعیتی‌ای که از جمعیت ***r*n*** فرد برخوردار است، بیابد. به هر حال کیفیت متوسط جمعیت نهایی در ***ga***های ایزوله‌ی موازی بسیار پایین‌تر است. با استفاده از مهاجرت، کیفیت جمعیت نهایی، بهبود قابل ملاحظه‌ای داشته و در برخی موارد به تر از کیفیت حل‌های یافته شده در ***GA*** سریال بوده است. اما در هر حال این نتیجه باید به دقت مورد تفسیر قرار گیرد، زیرا ***GA*** سریال، در پایان ۵۰۰ نسل کاملاً متشابه نشده است و بهبودهای بیشتر همچنان امکانپذیر است. جزئیات بیش‌تر درباره‌ی آزمایشات و نتایج آن را در مقاله‌ی تانس ***[TAN89b]*** می‌توانید بیابید. یک مقاله اخیراً به وسیله‌ی بلدینگ ***[BEL95]*** یافته‌های تانس را با استفاده از توابع ***royal road*** تایید نموده است. مهاجران به مقصدی تصادفی فرستاده می‌شدند، اما نویسنده مدعی است که آزمایشات با بهره‌گیری از ابر-مکعب نتایج مشابهی داشته است.

در بیش‌تر موارد بهینه‌ی سراسری اغلب زمانی یافته می‌شود که به جای مورد کاملاً ایزوله، از مهاجرت استفاده شود.

حتا در اینجا که ما تنها مشتی از خروار مثال‌های الگوریتم‌های ژنتیک موازی چنددسته‌ای را مرور کردیم، شاید بتوان چندین دلیل، که موجب محبوب و معمول شدن این روش گردیده، بیان داریم:

- ✓ الگوریتم‌های ژنتیک چنددسته‌ای، توسعه ی ساده‌ای از الگوریتم‌های ژنتیک سریال به نظر می‌رسند. دستورالعمل ساده است: چند الگوریتم ژنتیک سنتی (سریال) را گرفته و هر یک را در یک گره از کامپیوتری موازی اجرا کنید و در زمان‌های از پیش مشخص‌شده‌ای تعداد اندکی از افراد را مبادله نمایید.
- ✓ نسبتاً تلاش اضافه‌ی کم‌تری برای تبدیل الگوریتم ژنتیک سریال به یک الگوریتم ژنتیک موازی چنددسته‌ای نیاز است. بیش‌تر برنامه‌ی الگوریتم ژنتیک سریال دست‌نخورده باقی می‌ماند و تنها توابع اندکی برای پیاده‌سازی مهاجرت به آن اضافه می‌شود.
- ✓ کامپیوترهای موازی دانه درشت به سادگی در دسترس‌اند و حتا هنگامیکه نیستند، شبیه‌سازی آن‌ها با یک شبکه از ایستگاه‌های کاری یا حتا پردازنده‌ی تنها با استفاده از برخی نرم‌افزارهای ***free*** که برای این کار طراحی شده‌اند، به آسانی امکانپذیر است. (مانند ***MPI*** و ***PVM***)

این بخش به مرور برخی از مقالاتی که تحقیقاتی نظام‌مند بر روی الگوریتم ژنتیک موازی با استفاده از چند جمعیت را آغاز کرده‌اند و همچنین پرسش‌هایی که در این زمینه هنوز با آن‌ها روبه‌رو هستیم،

پرداخت. بخش بعد به مرور مقالات بیش‌تری درباره‌ی الگوریتم ژنتیک موازی چنددسته‌ای می‌پردازد و نشان می‌دهد که چگونه این زمینه به بلوغ رسید و شروع به تحقیق دیگر جنبه‌های این الگوریتم‌ها می‌کند.

۷) الگوریتم‌های ژنتیک موازی دانه‌درشت-نسل دوم

با توجه به مقالاتی که در بخش قبل بررسی شد، می‌توان دریافت برخی ایده‌های مهم، آشکار هستند. برای مثال ***pGA***ها در ترم‌های رسیدن به کارائی امیدبخش هستند. همچنین، ***pGA***ها بسیار پیچیده‌تر از همتایان سریال خود هستند؛ به خصوص مهاجرت افراد از یک قسمت به دیگر قسمت‌ها، که با چندین شاخصه (پارامتر) کنترل می‌گردد. از جمله: (***a***) توپولوژی‌ای که اتصالات میان زیرجمعیت‌ها را تعریف می‌کند. (***b***) نرخ مهاجرت، که تعداد مهاجران را کنترل می‌کند. (***c***) فاصله‌ی زمانی بین مهاجرت‌ها که بر فرکانس مهاجرت تأثیر می‌گذارد.

در اواخر دهه‌ی ۸۰ و اوایل دهه‌ی ۹۰، تحقیقات بر روی ***pGA***ها آغاز به کاوش درباره‌ی جایگزینی کردند که ***pGA***را سریع‌تر کند و اینکه چگونگی کارکرد آن‌ها را بتوان به‌تر فهمید. تا زمان کنونی نخستین مطالعات تئوری بر روی ***pGA***ها آشکار گردیده و تحقیقات آزمایشی برای تشخیص شاخصه‌های مناسب پیگیری می‌شود. این بخش به مرور برخی از نخستین کارهای تئوری و مطالعات آزمایشگاهی درباره‌ی مهاجرت و توپولوژی‌ها می‌پردازد. همچنین در این دوره، بیش‌تر محققان آغاز به استفاده از ***GA***های چندجمعیتی برای حل مسائل کاربردی نموده‌اند، و این بخش با مروری کوتاه بر کار آن‌ها خاتمه می‌یابد.

یکی از مسیرهایی که در تکمیل این زمینه اهمیت دارد، آزمایش ***pGA***ها به همراه توابع آزمون بسیار بزرگ و سخت است، که تلاش‌هایی در این زمینه شروع شده است. یک مثال قابل توجه کاری است که توسط مولنبین، شومیخ و برن ***[MUH 91a]*** انجام گرفته و به توصیف ***pGA***هایی می‌پردازد، که بهینه‌ی سراسری چندین تابع مورد استفاده برای محک الگوریتم‌های بهینه‌سازی را پیدا می‌کند. بعدتر، توابع مورد استفاده در این مقاله، توسط دیگر محققان نیز به عنوان محکی برای آزمایش آنان، پذیرفته شد. (مثلاً، ***[CHE93, GOR93]***) مولنبین، شومیخ و برن، از بهینه‌ساز محلی در الگورتم استفاده نمودند، و چون آن‌ها برنامه‌ای مؤثر و قدرتمند طراحی نمودند، روشن نیست، که نتایج حاصل شده، برای جمعیت توزیع شده، اعتبار دارد و یا برای بهینه‌ساز محلی.

مولنبین عقیده دارد که ***pGA***ها می‌تواند در مطالعه‌ی تکامل جمعیت‌های طبیعی سودمند باشد. ***[MUH 89a, MUH 89b, MUH 92, MUH 91a]***. این دید در تمام کسانی که درک کرده‌اند، یک جمعیت چندبخش‌شده، با جایگزینی گاه‌به‌گاه افراد، همانندی نزدیک‌تری به فرآیند طبیعی تکامل، در قیاس با یک تک جمعیت دارد؛ مشترک است. اما، با وجود اینکه، این نکته که، در یک الگوریتم ژنتیک

موازی می‌توانیم برخی پدیده‌های، مستقیماً همتای طبیعت (مثلاً اینکه، تغییر مناسب انتشار یک کروموزوم، در دسته‌های کوچک‌تر نسبت به دسته‌های بزرگ تر سریع‌تر است) را ببینیم، درست است؛ قصد اصلی اغلب محققانِ این زمینه، تنها طراحی سریع‌تر الگوریتم‌های جست‌وجوست.

۷-۱) نظریه‌پردازی‌های اولیه

در جمع‌بندی همه‌ی مقالاتی که، تا کنون غالباً بر پایه‌ی نتایج تجربی انتشار یافته‌اند، اما پس از نخستین موج انتشار به توصیف پیاده‌سازی‌های موفق پرداخته‌اند، به نظر می‌رسد که مطالعات تئوریک اندکی انجام گرفته است. یک مطالعه‌ی نظری درباره‌ی الگوریتم‌های ژنتیک (موازی) نیاز به برگرفتن فهمی عمیق‌تر از این الگوریتم‌ها دارد، تا بتوانیم آن‌ها را در پتانسیل کامل‌شان، به کار گیریم.

شاید نخستین تلاشی که پایه‌ای نظری برای کارائی یک ***pGA*** فراهم آورد، مقاله‌ی نوشته شده توسط پتی و لوئیز ***[PET 98]*** باشد. در این مقاله‌ی آن‌ها استنتاجاتی درباره‌ی شمای قضیه‌ای برای آن دسته از ***pGA***هایی که تصادفاً افراد را برمی‌گزینند تا در هر نسل پراکنده[1]شان کنند، توصیف گردیده است. محاسبات آن‌ها نشان داد که تعداد مورد انتظار دنباله‌های تخصیص داده شده بر شما می‌تواند به وسیله‌ی یک تابع تجربی محدوده‌بندی[2] گردد، دقیقاً همانگونه که در مورد الگوریتم‌های ژنتیک سریال انجام می‌گیرد.

یک سؤال تئوریک بسیار مهم تعیین آن است که آیا (وتحت چه شرایطی) یک ***pGA*** می‌تواند حل به‌تری نسبت به ***sGA*** بیابد؟ استارک‌وثر، وایتلی و مَتیاس ***[STA 91]*** دو مشاهده‌ی مهم را در قبال این پرسش از سر گذراندند. نخست آنکه، دسته‌های نسبتاً ایزوله، تمایل به همگرایی به حل‌های متفاوتی دارند، و مهاجرت و بازترکیب ممکن است حل‌های جزئی را ترکیب کند. استارک‌وثر و دیگران احتمال دادند، که به همین دلیل الگوریتم ژنتیک موازی با مهاجرت اندک، ممکن است حل‌های به‌تری نسبت به توابع مجزا ارائه دهد. دومین مشاهده‌ی آن‌هاع آن بود که، اگر بازترکیب نتایج حل‌های جزئی در افرادی با برازندگی پایین انجام شود (شاید بدین خاطر که تابع مجزا نبوده است)، الگوریتم ژنتیک سریال ممکن است بر آن مزیت داشته باشد.

۷-۲) مهاجرت

یکی از علائم بلوغ این زمینه آن است که تحقیقات بر جنبه‌های خاصی از الگوریتم‌های ژنتیک موازی تمرکز نموده‌اند. این بخش به مرور برخی مقالاتی که به کاوش درباره‌ی تدبیرهای مهاجرتی جایگزین، در جهت کاراتر نمودن الگوریتم‌های ژنتیک موازی پرداخته‌اند، می‌پردازد.

---

[1] ***broadcast***
[2] ***bounded***

در اکثر الگوریتم‌های ژنتیک موازی چنددسته‌ای، مهاجرت هم‌گام بدین معنی‌است که این امر در خلال زمان‌های ازپیش‌تعیین‌شده‌ی ثابتی روی‌می‌دهد. مهاجرت می‌تواند ناهم‌گام نیز باشد که بنابراین دسته‌ها تنها پس از روی‌دادن برخی رخدادها ارتباط برقرارمی‌کنند. مثالی برای مهاجرت ناهم‌گام را می‌توان در رساله‌ی گروسو که آزمایش‌های او با سامانه‌ی مهاجرت "تاخیری"، که مهاجرت را تاقبل از همگرایی تقریباً کامل دسته اجازه‌می‌دهد، یافت.

بران ***[BRA 90]*** چنین ایده‌ی را مورد استفاده قرارداد و الگوریتمی ارائه‌نمود که در آن مهاجرت پس از همگرایی کامل روی‌می‌دهد. (نویسنده واژه‌ی "انحطاط" را برای این مفهوم مورد استفاده قرارداده‌است.) با این هدف که گوناگونی در دسته‌ها بازاندوزی گشته و از همگرایی زودرس حل در کیفیت پایین جلوگیری گردد. استراتژی مهاجرت یکسانی پس از آن توسط مونه‌تامو، تاکی و ساتو ***[MUN 93]***، استفاده‌گردید و اخیراً کانت‌پاز و گلدبرگ ***[CAN97b]*** مدل‌های نظری‌ای را که –در محیط کاملاً متصل– به پیش‌بینی کیفیت حل‌ها می‌پردازد ارائه‌نموده‌است.

سؤال جالب دیگری در این‌جا به ذهن خطورمی‌کند: چه زمانی، زمان شایسته‌ی مهاجرت است؟ به نظر می‌رسد که اگر مهاجرت در طول زمان اجرا بسیار زود رخ‌دهد، ممکن است تعداد بلوک‌های خوش‌ساختمان مهاجران اندک‌تر از آن باشد که بتواند تأثیری بر جست‌وجوی درمسیر صحیح داشته‌باشد، و ارتباطات گران اتلاف منابع را موجب‌گردد.

راهبرد متفاوتی برای مهاجرت که توسط مارین، ترلس–سالازار و سنداوال ***[MAR 94b]*** توسعه‌داده‌شد. آن‌ها راهبردی متمرکز را پیشنهادنمودند که در آن فرآیندهای پیرو یک الگوریتم ژنتیک را در جمعیت خود اجرا نموده و متناوباً به‌ترین نتایج جزئی را به فرآیند پایه ارسال می‌نمایند. سپس، فرآیند پایه برازنده‌ترین افراد یافته‌شده –توسط هر فرآیند– را برگزیده و آن‌ها را به دیگر پیروها اشاعه‌می‌دهد. نتایج آزمایشگاهی در یک شبکه‌ی کامپیوتری نشان‌داد که با استفاده از تعداد اندکی گره –حدود ۶ گره– تسریع غیرخطی حاصل‌می‌گردد و نویسندگان مدعی شده‌بودند که این راهبرد می‌تواند گستره‌پذیری[1]ای تا حدود شبکه‌های بزرگ داشته‌باشد، زیرا ارتباط تکرارشونده نیست.

۷-۳) توپولوژی‌های ارتباطی

یک مؤلفه‌ی مغفول در مورد الگوریتم‌های ژنتیک موازی در گذشته، توپولوژی ارتباطات داخلی میان دسته‌ها بوده است. توپولوژی فاکتور مهمی در کارائی الگوریتم ژنتیک موازی است، زیرا سرعت (یا کندی) توزیع یک حل خوب را به دیگر دسته ها تعیین می‌کند. اگر توپولوژی اتصالات گسترده و چگالی داشته باشد (یا قطر کم، یا هر دو)، حل‌های خوب، سریع به دیگر دسته ها منتشر خواهند شد، و احتمالاً به سرعت بر جمعیت مسلط خواهند شد. از طرف دیگر، اگر توپولوژی اتصالات خلوت و کمی

[1] ***Scale up***

را دارا باشد (یا قطر زیادی داشته باشد)، حل‌ها آهسته‌تر منتشر می‌شوند و دسته ها از یک دیگر ایزوله تر خواهند بودع که اجازه‌ی وقوع حل‌های متفاوت را می‌دهد. این حل‌ها ممکن است در زمان بعدتری جمع‌آوری شوند، و برای ساخت افراد بالقوه به‌تر، بازترکیب شوند.

توپولوژی ارتباطیع همچنین به خاطر آنکه، فاکتور مهمی در هزینه ی مهاجرت به حساب می‌ایدع از اهمیت برخوردار است. برای مثال، یک توپولوژی با اتصالات چگال، آمیختگی به‌تری از افراد، عرضه می‌کند، اما مستلزم هزینهی ارتباطی افزون‌تری نیز هست.

روند عمومی الگوریتم‌های ژنتیک موازی چند-دسته‌ایع استفاده از توپولوژی‌های ایستایی‌است که در ابتدای اجرا مشخص شده اند و تا پایان بی‌تغییر می‌مانند. عمده‌ی پیاده‌سازی‌های الگوریتم‌های ژنتیک موازی چند-دسته‌ای که توپولوژی‌های ایستایی دارند، از توپولوژی طبیعی کامپیوتر در دسترس‌شان برای تحقیق بهره می‌گیرند. برای مثال، پیاده‌سازی‌ها بر روی ابرمکعب‌ها ***[COH91a, TAN89a, TAN89b]*** و حلقه ها ***[GOR93]*** معمول هستند.

مطالعه‌ی آزمایشگاهی‌ای اخیراً نشان داده که الگوریتم‌های ژنتیک موازیع با توپولوژی چگال، حل سراسری را، با استفاده از ارزیابی توابع کم‌تری، نسبت به الگوریتم‌های ژنتیک دارای اتصالات خلوت، می‌یابند. ***[CAN94]*** این مطالعه توپولوژی‌های کاملاً متصل چهاربُعدی، ابرمکعب‌های چهاربُعدی، یک مِش حلقه‌مانند (***toroidal***) ۴×۴، و حلقه‌های بی‌جهت و دوجهته را در نظر می گیرد.

دیگر انتخاب مهم، استفاده از توپولوژی دینامیک است. در این روش، یک دسته به ارتباط با مجموعه‌ای ثابت از دسته‌ها، محدود نمی‌گردد، و به جای آن، مهاجران به دسته‌هایی ارسال می‌شوند، که برخی معیارها را برآورده سازند. انگیزه‌ی استفاده از توپولوژی دینامیک، تشخیص دسته‌هایی است، که مهاجران، احتمالاً می‌توانند تغییراتی در آن‌ها ایجاد کنند. نوعاً، معیار استفاده شده برای انتخاب دسته، به عنوان مقصد، شامل اندازه‌هایی از گوناگونی جمعیت است ***[MUN93]*** و یا اندازه‌ی فاصله‌ی ژنوتیپیک میان دو جمعیت ***[LIN94]***. (یا برخی افراد نماینده‌ی یک جمعیت، مثلاً به‌ترین آن‌ها)

۷-۴) کاربردها

بسیاری از پروژه‌های کاربردی که از الگوریتم ژنتیک موازی چندجمعیتی استفاده می‌کنند، تاکنون انتشار یافته‌اند. این بخش تنها به بررسی مثال‌های شاخص می‌پردازد.

مسئله‌ی بخش‌بندی گراف یکی از کاربردهای مشهور الگوریتم ژنتیک چنددسته‌ای است. (برای مثال کارهای انجام شده توسط بسیر و تالبی ***[BES91]*** و کُهون، مارتین و ریچاردز ***[COH 91c]***، تالبی و بسیر ***[TAL 91]***) کارِ "لوین" ***[LEV94]*** روی مسئله‌ی بخش‌بندی مجموعه برجسته است. الگوریتم او حل‌های سراسری را با ۳۶۶۹۹ و ۴۳۷۴۹ متغیر صحیح می‌یابد. او دریافت که کارائی اندازه‌گیری شده به عنوان

کیفیت حل و تکرار آنچه می‌یابد، با اضافه شدن دسته‌های بیش‌تر افزایش می‌یابد. همچنین لی و ماشفورد ***[LI90]*** و مولنبین ***[MUH89b]*** راه حل‌های قابل قبولی برای مسئله‌ی انتساب درجه‌ی دوم (که یک مسئله‌ی بهینه سازی دیگر ست) می‌یابند.

الگوریتم ژنتیک موازی دانه درشت برای یافتن حل‌هایی برای مسئله‌ی توزیع بار محاسباتی در گره‌های پردازشی کامپیوترهای ***MIMD***، توسط نیواس ***[NEv91]***، مونتین و تالبی ***[MUN91]*** و سردینسکی ***[SER94]*** نیز به‌کار گرفته شده‌اند.

یکی دیگر از کاربردهای بحث‌انگیز، سنتز مدارهای ***VLSI*** است. دیوس، لیو و الیاس ***[DAV94]*** انواع گوناگونی از الگوریتم ژنتیک موازی را در جست‌وجوی حل این مسائل به‌کار گرفته اند. دو گونه از الگوریتم ژنتیک موازی از جمعیت‌های چندگانه بهره گرفته‌اند (از میان چهار دسته) و گونه‌ی دیگر (سوم) الگوریتم ژنتیک سراسری است که ۱۶ فرآیند کارگر را برای ارزیابی جمعیت به کار می‌گیرد. در نخستین الگوریتم ژنتیک چنددسته‌ای، دسته‌ها ایزوله هستند و در دومین گونه ، دسته‌ها با برخی دسته‌های دیگر، (به صورت نامشخص) در زمانبندی قابل برنامه‌ریزی‌ای، ارتباط دارند. این مقاله بسیار جالب است، زیرا شامل دو عنصر غیرمعمول است: نخست اینکه، الگوریتم ژنتیک موازی با استفاده از ***Linda*** (یک زبان هماهنگ‌سازی موازی که معمولاً به خاطر کُند بودن‌اش مورد ملاحظه قرار می‌گیرد/نمی‌گیرد) پیاده‌سازی شده. و دوم اینکه، تسریع با استفاده از زمانی که برای یافتن یک حل، با کیفیتی ثابت و معین، به وسیله‌ی الگوریتم‌های ژنتیک سریال و موازی صرف می‌شود، اندازه‌گیری شده است. محاسبه‌ی تسریع بدین روش، اندازه‌ی کارائی قابل ملاحظه‌ای را برای الگوریتم ژنتیک موازی، نسبت به زمانی که زمان اجرای الگوریتم‌های ژنتیک را برای نسل‌های معینی مقایسه کنیم، نشان می‌دهد.

۸) الگوریتم‌های ژنتیک غیرسنّتی

همه‌ی الگوریتم‌های ژنتیک موازی چندجمعیتی از الگوریتم ژنتیک سنتی‌ای که در بخش دوم مرور گردید، استفاده نمی‌کنند. برخی الگوریتم‌هی ژنتیک جدید (غیرسنتی) نیز موازی‌سازی شده‌اند و بهبودهایی نسبت به حالت سریال در آن‌ها گزارش گردیده است. برای مثال، مارسکی ***[MAR 94a]*** از مدل ***Devidor's ECO [DAV 91]*** در گره‌ها برای الگوریتم توزیع‌شده‌ی خود استفاده نموده است و از شماهای مهاجرتی متفاوتی با پیچیدگی‌های متفاوت بهره می‌گیرد. ساده‌ترین شمای مهاجرتی این است که اصلاً هیچ مهاجرتی نداشته باشیم و فقط نتایج را از هر گره، در پایان یک اجرا، جمع کنیم. الگوریتم‌های پیچیده‌تر مستلزم مهاجرت در دسته‌های ***ECO*** کامل است. چندین سیاست جایگزینی مورد کاوش قرار گرفت، که تنها مزایای حاشیه‌ای داشت.

Genttor یک الگوریتم ژنتیک حالت-یکنواخت[1] است، که در آن تنها بخش کوچکی از افراد در هر نسل تغییر می‌کنند. در نسخه‌ی توزیع‌شده‌ی ***Genttor*** ***[WILL 90]***، افراد در فواصل زمانی ثابتی به همسایه‌ها مهاجرت می‌کنند و با بدترین افرادِ دسته‌ی مقصد جایگزین می‌شوند. در مقاله‌ی اصلی و همچنین در کاری که بعداً توسط گوردون و وایتلی ***[GOR 93]*** انجام گرفت، بهبود قابل ملاحظه‌ای نسبت به نسخه‌ی سریال، برای این الگوریتم گزارش شده است.

تلاش‌هایی برای موازی‌سازی الگوریتم‌های ژنتیک ***messy***[2] نیز صورت گرفته است. ***[GOL 89b]*** الگوریتم‌های ژنتیک ***messy*** دو فاز دارند. در فاز اولیه جمعیت ابتدائی با استفاده از شمارش جزئی[3] ایجاد می‌گردد، و با استفاده از انتخاب به شیوه‌ی تورنمنت از این تعداد کاسته می‌شود. در مرحله‌ی بعد (مرحله‌ی مجاورت[4])، حل‌های جزئی یافت‌شده در مرحله‌ی نخست، با یکدیگر آمیخته می‌شوند. از آنجا که مرحله‌ی نخست زمان اجرا را کنترل می‌کند، مرکل و لامونت ***[MER 93b]***، در تلاش برای بسط کار قبلی انجام شده توسط دایمک ***[DYM 92]***، از چندین استراتژی توزیع داده برای تسریع فاز نخست استفاده نموده‌اند. نتایج بدست آمده مشخص می‌کند که در حالی که تغییر قابل ملاحظه‌ای در کیفیت حل‌ها در استراتژی‌های گونه‌گون توزیعی پدید نیامده است، در زمینه‌ی زمان اجرا این استراتژی‌ها به دستاوردهای قابل ملاحظه‌ای رسیده‌اند.

مرکل، گیتز، لامونت و پَچِر یک الگوریتم ژنتیک ***messy*** سریع را موازی‌سازی نموده‌اند، ***[GOL 93a]*** تا صورت‌بندی بهینه‌ی مولکول‌های ***Met-Enkephalin*** را جست‌وجو کنند. ***[MER 93a]*** در این مسئله، الگوریتم، تسریع خوبی را برای ۳۲ پردازنده نشان داد، اما با پردازنده‌های بیش از ۳۲، تسریع دیگر به سرعت افزایش نمی‌یابد.

کوزا و اندره ***[KOZ 95]*** از شبکه‌ای از ***transputer***ها برای موازی‌سازی برنامه‌سازی ژنتیک بهره گرفتند. در این مقاله، آنان مطالعاتی درباره‌ی نیازمندی‌های حافظه و محاسباتی الگوریتم خود انجام داده و به این نتیجه رسیدند که ***transputer***ها با توجه به هزینه مؤثرترین راه برای پیاده‌سازی این الگوریتم هستند. ۶۴ دسته در پیاده‌سازی آنان در یک ***mesh*** دو بُعدی به همدیگر متصل گردیده و هر دسته دارای ۵۰۰ فرد است. نویسندگان نرخ های مهاجرت متفاوتی را آزمایش کرده‌اند و گزارش آنان حاکی از تسریع خطی فوق‌العاده[5] در تلاش‌های محاسباتی (و نه زمان کل)، با استفاده از نرخ مهاجرتِ معتدل (در حدود ۸٪) بوده است.

۹) الگوریتم‌های ژنتیک موازی دانه‌ریز

---

[1] ***Steady-state***

[2] شلوغ، آشفته

[3] ***Partial enumeration***

[4] ***Juxtapositional phase***

[5] ***Super linear***

این الگوریتم‌ها تنها یک جمعیت دارند، اما این جمعیت ساختاری فضایی[1] دارد، که تعامل بین افراد را محدود می‌کند. یک فرد تنها می‌تواند با همسایگان خود رقابت و آمیزش کند، اما همسایگانی که در حل‌های خوب همپوشانی دارند، ممکن است در تمام جمعیت توزیع شوند.

رابرتسون ***[ROB87]*** الگوریتم ژنتیک یک سیستم دسته‌بندی[2] را بر روی یک ***CM-1***[3] موازی‌سازی نمود. او انتخاب والدین، انتخاب سیستم‌های دسته‌بندی‌ای که باید جایگزین شوند، آمیزش و ***Crossover*** را موازی‌سازی نمود. زمان اجرای پیاده‌سازی او مستقل از تعداد سیستم‌های دسته‌بندی بود. (تا حدود ***16k***، تعداد عناصر پردازنده در ***CM-1***)

آسپاراگوس به وسیله‌ی جرج-شلتر ***[GOR 89a, GOR89b]*** و مولنبین ***[MUH 89a, MUH 89b]***، معرفی گردید. آسپاراگوس از ساختار جمعیتی‌ای استفاده می‌کند، که یک نردبان[4] را با پایان‌های بالایی و پایینی همانند می‌کند و با یکدیگر گره می‌زند. آسپاراگوس برای حل برخی مسائل ترکیبی بهینه‌سازی با موفقیت زیادی به کار گرفته شده است. بعدتر یک ساختار جمعیت خطی در ***[GOR 91]*** استفاده گردید و روش‌های مختلف آمیزش مقایسه شد. ***[GOR 92b]***

آسپاراگوس از الگوریتم کوهنوردی (***Hillclimbing***) محلی برای بهبود برازندگی افراد در جمعیت خود بهره می‌گیرد. این امر آن را برای جدا کردن سهم ساختار فضاییِ جمعیت در کیفیت جست‌وجو مشکل می‌سازد. به‌هر حال همه‌ی نتایج تجربی نشان از این دارند که آسپاراگوس ابزار بهینه‌سازی بسیار مؤثری است.

ماندریک و اسپیسنس ***[MAN89]*** نوعی الگوریتم ژنتیک موازی دانه-ریز را پیاده‌سازی نمودند، که جمعیت آن در یک ***grid*** دو بُعدی توزیع‌شده بود. انتخاب و آمیزش تنها با یک همسایه امکان‌پذیر بود و نویسندگان دریافتند که کارائی الگوریتم با افزایش اندازه‌ی همسایگی، پایین می‌آید. در نهایت، اگر سایز همسایگی به اندازه‌ی کافی بزرگ باشد، این الگوریتم ژنتیک موازی، همسان الگوریتمی با یک تک جمعیت عمل خواهد نمود. نویسندگان، این الگوریتم را به صورت راتئوریک در دو سال پیاپی ***[SPI 90, 91]*** تحلیل کرده‌اند و تحلیل‌های آنان نشان می‌دهد که برای سایز دسته‌ی ***s*** و طول رشته‌ی ***l***، پیچیدگی زمانی الگوریتم ***O(s+l)*** یا ***O(s(lg s) + l)*** گام زمانی است، بسته به اینکه چه تدبیری برای انتخاب در نظر گرفته شود.

اخیراً سارما و دژانگ ***[SAR96]*** اثرات اندازه و شکل[5] همسایگی را بر انتخاب مکانیسم بررسی نموده‌اند و نسبت شعاع همسایگی را به شعاع کل شبکه‌ی ***grid*** را پارامتری تعیین‌کننده یافته‌اند، که فشار انتخاب

---

[1] ***spatial***
[2] ***classifier***
[3] ***Connection Machine- 1***
[4] ***ladder***
[5] ***shape***

در تمام جمعیت را تعیین می‌کند. تحقیقات آنان به‌شکلی کمّی مشخص ساخت، که زمان انتشار حل خوب در کل جمعیت، با توجه به اندازه‌ی همسایگی متفاوت، تغییر می‌کند.

معمول است که افراد یک الگوریتم ژنتیک موازی دانه-ریز را در یک شبکه‌ی ***grid*** دوبُعدی قرار می‌دهند، زیرا در بسیاری از کامپیوترهای موازی توده‌ای، عناصر پردازش با این توپولوژی متصل شده‌اند. به هر حال، بسیاری از این کامپیوترها، یک مسیریاب سراسری دارند که می‌تواند پیام‌ها را به همه‌ی پردازنده‌های داخل شبکه بفرستد (با هزینه‌ای بالاتر)؛ دیگر توپولوژی‌ها نیز می‌تواند بر روی ***grid*** شبیه‌سازی گردد.شووِهم ***[SCH92]*** تذثیر استفاده از ساختارهای متفاوت را بر جمعیتِ الگوریتم ژنتیک موازی دانه-ریز مقایسه نموده است. در این مقاله، یک مسأله‌ی بخش‌بندی گراف به عنوان محک آزمونِ ساختارهای جمعیتی متفاوت بر روی کامپیوترِ ***MasPar MP-1*** شبیه‌سازی شده است. ساختارهای تست‌شده، یک حلقه، یک ***torus***، یک مکعب ۱۶×۸×۸، یک ابرمکعب ۴×۴×۴×۴ و یک ابرمکعب دودوئی ۱۰بُعدی بوده است. الگوریتم با ساختار ***torus***، زودتر از دیگر الگوریتم‌ها همگرا گردید، اما ذکری از کیفیت نتیجه در مقاله نیامده است.

اندرسون و فریس ***[AND90]*** نیز ساختارهای جمعیتی مختلف و الگوریتم‌های جایگزینی متفاوت را مورد بررسی قرار داده‌اند. آن‌ها ساختار دوحلقه‌ای، ابرمکعب، دو مش و یک ساختار "جزیره‌ای"، که در آن تنها یک فرد از دسته با دسته‌ای دیگر همپوسانی داشت، را مورد آزمایش قرار دادند. آن‌ها نتیجه گرفتند که برای مسئله‌ی خاصی که آن‌ها برای حل آن تلاش می‌کردند (تعادل خط مونتاژ) ساختارهای حلقه و "جزیره" به‌ترین هستند. بالوجا دو متغیر را بر یک ساختار خطی و یک ساختار مش دوبُعدی ***[BAL92, BAL93]*** آزمایش کرد و دریافت که مش به‌ترین نتایج را تقریباً در همه‌ی مسائلی که مورد آزمون قرار گرفته‌اند، به دست می‌دهد.

الگوریتم ژنتیک موازی دانه-ریز برای حل بعضی مسائل کاربردی بسیار مشکل با موفقیت به‌کار گرفته شده‌اند. یک مسئله‌ی متداول، کوله‌پشتی دودوئی دوبُعدی است و مسائل حل رضایتمندی که به وسیله‌ی کروگر، شووِندرلینگ و وامبرگر ***[KRO91, KRO92, KRO93]*** الگوریتم‌های ژنتیک موازی دانه-ریزی برای حل آن‌ها یافت شده‌اند. مسئله‌ی زمانبدنی کار نیز مسئله‌ای متداول در الگوریتم ژنتیک است. تاماکی و نیشیکاوا ***[TAM92]*** الگوریتم‌های ژنتیک دانه-ریز را با موفقیت برای حل این مسئله به‌کاربرده‌اند.

شاپیرو و ناوتا ***[SHA94]*** از الگوریتم‌های ژنتیک دانه-ریز، جهت پیش‌بینی ساختار ثانویه‌ی ***RNA*** استفاده نموده‌اند. آن‌ها توپولوژی طبیعی ***X-Net*** کامپیوترِ ***MasPar-2*** را برای تعریف همسایگی هر فرد استفاده نموده‌اند. توپولوژی ***X-Net*** یک عنصر پردازشگر را به ۸ پردازنده متصل می‌کند. توپولوژی‌های منطقی مختلفی همراه الگوریتم ژنتیک آزمایش شده‌اند. توپولوژی‌های منطقی گوناگونی همراه الگوریتم

ژنتیک تست گردیدند: همه‌ی هشت همسایه‌ی نزدیک، چهار نزدیک ترین همسایه، و همه‌ی همسایه‌های دارای مشخصه‌ی فاصله‌ی *d*. نتایج برای *d>2* ضعیف‌ترین نتایج بوده است، و شمای ۴-همسایه همواره نسبت به هشت-همسایه به‌تر[1] بود.

مقالات اندکی به مقایسه‌ی الگوریتم‌های ژنتیک موازی دانه-ریز و دانه-درشت پرداخته‌اند. ***[BAL93, GOR 93]*** در این مقایسه‌ها گاهی دانه-ریزها پیش افتاده‌اند، اما گاهی نیز نتیجه عکس این بوده است. مشکل اینجاست که مقایسه‌ها نبایست در یک شرایط مطلق ترتیب داداه شوند، بلکه، باید مشخصاً روشن سازیم که ما چه چیزی را می‌خواهیم کمینه سازیم (برای مثال، زمان) یا قصد داریم چه چیزی را بیشینه سازیم (برای مثال، کیفیت حل‌ها را) و مقایسه‌ها را بر معیار خاص خود صورت دهیم.

گوردون، وایتلی و بوهم ***[GOr92a]*** نشان داده‌اند که مسیر بحرانی الگوریتم ژنتیک دانه-ریز کوتاه‌تر از الگوریتم ژنتیک چند-جمعیتی است. این بدان معنی است که اگر پردازنده‌های کافی در دسترس باشند، الگوریتم‌های ژنتیک موازی توده‌ای، صرف‌نظر از اندازه‌ی جمعیتی که دارند، به زمان کم‌تری برای به پایان رساندن اجرای خود نیازمندند. در هر حال، مهم است به این نکته توجه داشته باشیم که این یک مطالعه‌ی تئوریک است و شامل ملاحظاتی چون پهنای باند ارتباطی یا حافظه‌ی مورد نیاز نمی‌شود. گوردون ***[GOR94]*** به مطالعه‌ی محلی‌بودن فضایی، در ارجاعات حافظه‌ای، در مدل‌های مختلف الگوریتم‌های ژنتیک موازی، نیز پرداخته است. این تحلیل‌ها برای تعیین مناسب‌ترین خط مشی محاسباتی، برای هر مدل، سودمند است.

۱۰) الگوریتم‌های ژنتیک موازی سلسله مراتبی

تعداد کمی از محققان تلاش کرده‌اند دو روش موازی‌سازی الگوریتم‌های ژنتیک را ترکیب‌نموده و از این رهگذر الگوریتم‌های ژنتیک موازی سلسله‌مراتبی را به‌وجود آورده‌اند. برخی از این الگوریتم‌های دورگه‌ی جدید، درجه‌ای جدید از پیچیدگی را به چشم انداز پیچیده‌ی موجود افزوده‌اند. دیگر الگوریتم‌های دورگه مدیریت شده‌اند تا پیچیدگی‌ای همسان یکی از مؤلفه‌های خود داشته باشند.

[1] ***Outperformed?***

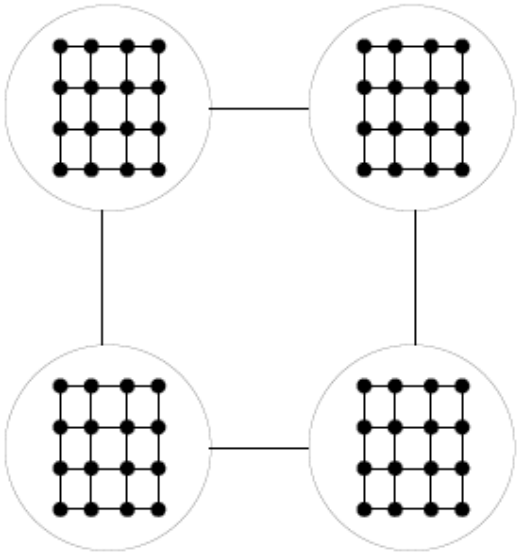


Figure 4. *This hierarchical GA combines a multi-deme GA (at the upper level) and a fine-grained GA (at the lower level).*

هنگامیکه دو روش موازی‌سازی الگوریتم‌های ژنتیک ترکیب شوند، یک سلسله‌مراتب را شکل می‌دهند. در سطح بالاتر، اغلب الگوریتم‌های ژنتیک موازی دورگه الگوریتم‌های چندجمعیتی هستند. برخی الگوریتم‌های ژنتیک موازی دورگه نیز الگوریتم‌های ژنتیک دانه-ریز را در سطح پایین‌تر خود دارند. (شکل ۴ را ببینید.)

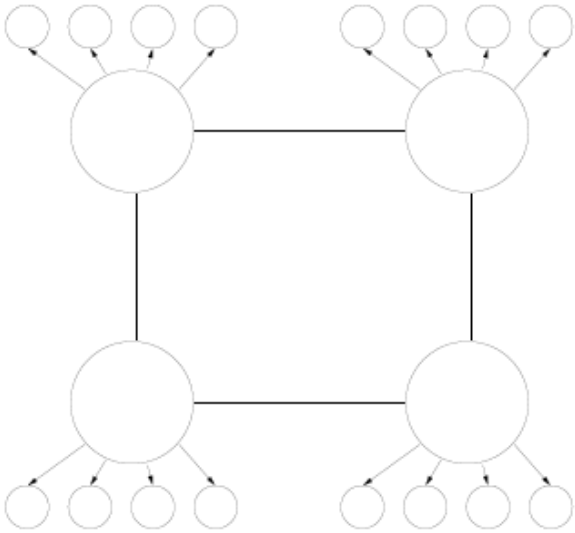


Figure 5. *A schematic of a hierarchical parallel GA. At the upper level this hybrid is a multi-deme parallel GA where each node is a master-slave GA.*

برای مثال گروا-او ***[GRU94]*** الگوریتم ژنتیک موازی "آمیخته"ای طراحی نمود. در الگوریتم او، جمعیت هر دسته در یک ***grid*** دوبُعدی قرار داده شده بود و دسته‌ها خود به شکل یک ***torus*** دوبُعدی

متصل شده بودند. مهاجرت در بین دسته‌ها در فواصل زمانی مشخصی اتفاق می‌افتاد و نتایج خوب یک طراحی شبکه‌ی عصبی جدید و برنامه‌ی کاربردی یادگیری گزارش می‌شدند.

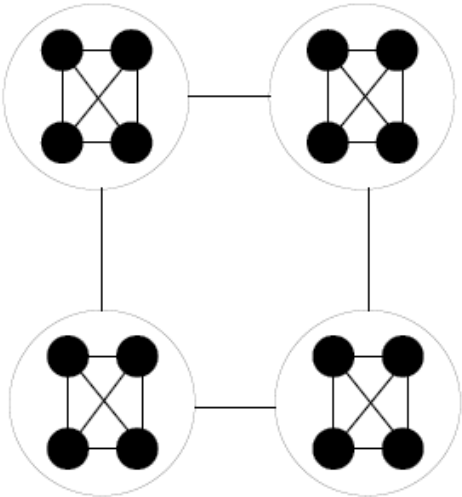


Figure 6. *This hybrid uses multi-deme GAs at both the upper and the lower levels. At the lower level the migration rate is faster and the communications topology is much denser than at the upper level.*

جدیداً در اسپاراگوس تجدید نظر شده ***[GOr97]***، و ساختار نردبانی آن با ساختاری حلقه‌ای جایگزین شده است، زیرا حلقه قطر بزرگ تری دارد و اجازه‌ی تفکیک به‌تر به افراد می‌دهد.نسخه‌ی جدید (آسپاراگوس ۹۶) چندین زیرجمعیت را نگه‌داری می‌نماید که خود را در ساختار حلقه‌ای سازماندهی کرده‌اند. مهاجرت میان زیرجمعیت‌ها امکانپذیر است، و هنگامی که یک دسته همگرا می‌شود، به‌ترین افراد دسته‌ای دیگر را دریافت می‌نماید. پس از آنکه همه‌ی جمعیت‌ها همگرا گردیدند، آسپاراگوس۹۶ به‌ترین و دومین فرد از هر جمعیت را می‌گیرد و از آن‌ها به عنوان جمعیت اولیه‌ی اجرای پایانی بهره می‌گیرد.

لین، گودمن و پانچ ***[LIN97]*** نیز از الگوریتم‌های ژنتیک چند-جمعیتی به همراه زیرجمعین‌های فضایی-ساختاریافته[1] بهره گرفته‌اند. قابل توجه اینکه آن‌ها نیز از توپولوژی حلقه -ای خلوت و قطور- برای اتصال زیرجمعیت‌ها استفاده کرده‌اند. اما هر دسته در آن به شکل یک ***torus*** ساختار یافته است. نویسندگان الگوریتم آمیخته‌ی خود را در مقابل الگوریتم‌های ژنتیک ساده، یک الگوریتم‌های ژنتیک دانه-ریز، دو الگوریتم‌های ژنتیک چند-جمعیتی با ساختار حلقه‌ای (یکی از پنج دسته استفاده می‌کند و از اندازه دسته های متفاوتی بهره می‌گیرد و دیگری سایز دسته‌ی ثابت ۵۰ فرد در هر دسته را دارد و

[1] ***Spatially-structured***

تعداد دسته در آن متغیر است) و یک الگوریتم‌های ژنتیک چند-دسته‌ای دیگر با دسته‌های متصل در یک ***torus***، مقایسه نموده‌اند. با استفاده از مسئله‌ی زمانبندی کار، به عنوان محک آزمایش، آن‌ها دریافتند که الگوریتم ژنتیک ساده، همانند دیگر الگوریتم‌ها حل‌های خوبی نمی‌یابد و اینکه اضافه کردن دسته‌های بیش‌تر به الگوریتم‌های ژنتیک چند-جمعیتی کارائی را بیش از افزودن افراد بیش‌تر و افزایش تعداد کل جمعیت، به‌تر می‌کند. الگوریتم آمیخته‌ی آن‌ها حل‌های به‌تری نسبت به دیگر الگوریتم‌ها یافت.

نوع دیگری از الگوریتم‌های ژنتیک موازی سلسله‌مراتبی از ساختار پایه-پیرو در هر دسته از یک الگوریتم ژنتیک چند-جمعیتی استفاده می‌کند. (شکل ۵ را ببینید) مهاجرت بین دسته‌ها اتفاق می‌افتد و ارزیابی افراد به صورت موازی انجام می‌گیرد. این راهبرد مسائل تحلیلی جدیدی مطرح نمی‌کند و می‌تواند در هنگام کار با کاربردهای پیچیده با توابع هدفی که نیاز به زمان محاسباتی قابل ملاحظه دارند مفید واقع شود. بیانچینی و براون ***[BIA93]*** مثالی از این روش موازی‌سازی الگوریتم‌های ژنتیک ارائه می‌کند و نشان می‌دهد که این روش می‌تواند حلی با کیفیت همسان الگوریتم‌های ژنتیک موازی پایه-پیرو یا یک الگوریتم ژنتیک چند-دسته‌ای، در زمانی کوتاه‌تر بیابد.

جالب است بدانید، مفهومی شبیه این توسط گلدبرگ ***[GOL89a]*** در بستر یک پیاده‌سازی شیء‌گرا از یک "مدل ارتباطی" الگوریتم ژنتیک موازی، طرح شده بود. در هر اجتماع[1] چندین خانه وجود دارد که والدین بازسازی می‌شوند و نتیجه‌ی زادوولد ارزیابی می‌شود. چندین اجتماع وجود دارد، و مهاجرت والدین به دیگر جاها امکانپذیر است.

روش سوم آمیختن الگوریتم‌های ژنتیک موازی، استفاده از الگوریتم‌های ژنتیک چند-جمعیتی در هر دوی سطوح بالاتر و پایین‌تر است. (شکل ۶ را ببینید) ایده‌ی اصلی این است که آمیختن ***panmictic*** را در سطح پایین‌تر با استفاده از نرخ مهاجرت بالا، و یک توپولوژی چگال، تقریباً اجباری کنیم، در حالی که در سطح بالاتر از نرخ مهاجرت اندکی بهره می‌بریم. ***[GOL, 96]*** پیچیدگی این الگوریتم آمیخته، احتمالاً همسان یک الگوریتم ژنتیک چند-دسته‌ای خواهد بود، اگر گروه زیرجمعیت‌های ***panmictic*** را به عنوان یک تک‌دسته در نظر گیریم. این روش هنوز پیاده‌سازی نگردیده است.

پیاده‌سازی‌های سلسله‌مراتبی می‌توانند زمان اجرا را، بیش از اجزای خود به تنهایی، کاهش دهند. برای مثال، در نظر بگیرید که در یک دامنه‌ی خاص، تسریع[2] بهینه‌ی یک الگوریتم ژنتیک پایه-پیرو Sp_{ms} باشد، و تسریع بهینه‌ی یک الگوریتم ژنتیک چند-دسته‌ای Sp_{md} باشد. تسریع (سردستی) یک الگوریتم ژنتیک سلسله‌مراتبی که از ترکیب این دو روش پدید آمده، $Sp_{ms}*Sp_{md}$ است.

---

[1] ***community***
[2] ***Speed-up***

۱۱) پیش‌رفت‌های اخیر

این بخش به مرور نتایج برخی پیشرفت‌های اخیر در مطالعات تئوریک الگوریتم‌های ژنتیک موازی می‌پردازد. ابتدا، این بخش به ارائه‌ی نتیجه‌ای درباره‌ی الگوریتم‌های ژنتیک پایه-پیرو می‌پردازد و سپس ارائه‌ای کوتاه از تئوری تعیین اندازه‌ی دسته‌ها در الگوریتم‌های ژنتیک چند-جمعیتی عرضه می‌گردد.

یکی از مسائل مهم درباره‌ی الگوریتم‌های ژنتیک پایه-پیرو آن است که، در این الگوریتم‌ها هر چه تعداد پردازنده‌های به‌کار گرفته شده بیش‌تر شود، زمان ارزیابی برازندگی جمعیت کاهش می‌یابد. اما همزمان، هزینه‌ی ارسال افراد به پیروها افزایش می‌یابد. این معاوضه‌ی میان نزول زمان محاسبات و فزونی زمان ارتباط موجب می‌شود، تعداد بهینه‌ای برای پیروها وجود داشته باشد، که کل زمان اجرا را کمینه نماید. اخیراً مطالعه‌ای ***[CAN 97a]*** به این جمع‌بندی رسیده است که مقدار بهینه $S^* = \sqrt{(nT_f/T_c)}$، که در آن n اندازه‌ی جمعیت، T_f زمان صرف شده برای یک ارزیابی تابع برازندگی، و T_c زمان ارتباط است. تسریع بهینه $0.5\ S^*$ است.

جست‌وجو درباره‌ی اندازه‌ی دسته‌ها در الگوریتم‌های ژنتیک موازی چند-جمعیتی، نقطه‌ی شروع طبیعی تحقیق در مورد آن‌هاست، زیرا اندازه‌ی جمعیت به احتمال فراوان تأثیرگذارترین پارامتر بر کیفیت حل‌های الگوریتم ژنتیک است. ***[GOL 91, GOL 92, HAR 97]*** اخیراً، کنتوپاز و گلدبرگ ***[CAN97b]*** یک مدل ساده‌ی برآورد اندازه‌ی جمعیت الگوریتم ژنتیک را بسط داده‌اند، تا در دو مورد تعیین‌کننده[1] در میان الگوریتم‌های ژنتیک دانه-درشت نیز به کار آید. این دو مورد تعیین‌کننده مجموعه‌ای از دسته‌های جداگانه (ایزوله) و مجموعه‌ای از دسته‌های کاملاً متصل بوده‌اند. در مورد دسته‌های متصل، نرخ مهاجرت به بالاترین مقدار ممکن مقداردهی گردید.

[1] ***Bounding case***

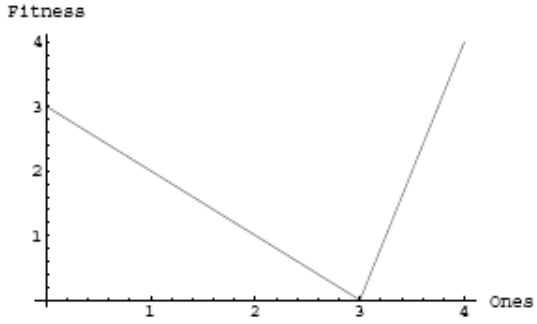


Figure 7. *A deceptive 4-bit trap function of unity. The horizontal axis is the number of bits set to 1 and the vertical axis is the fitness value. The difficulty of this function can be varied easily by using more bits in the basin of the deceptive optimum, or by reducing the fitness difference between the global and deceptive maxima. The first test problem was constructed by concatenating 20 copies of this function and the second test problem is formed with 20 copies of a similar deceptive function of 8 bits.*

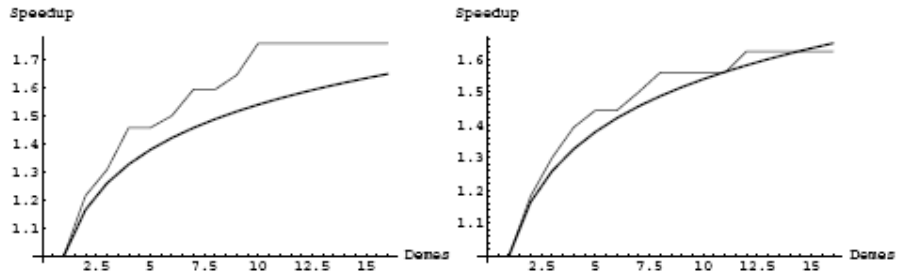
اندازه‌ی جمعیت نیز، فاکتوری اساسی در تعیین زمان مورد نیاز برای یافتن حل‌ها در الگوریتم ژنتیک است. بنابراین، مدل‌های تعیین اندازه‌ی دسته‌ها، می‌توانند برای پیش‌بینی زمان اجرای الگوریتم ژنتیک موازی به‌کار آیند، و در مقایسه‌ی آن با زمان مورد نیاز یک الگوریتم ژنتیک سریال برای دست‌یابی به همان کیفیت، به کار روند. کنتوپاز و گلدبرگ ***[CAN 97c]*** مدل‌های تعیین اندازه‌ی دسته‌هارا، به همراه مدلی برای محاسبه‌ی زمان ارتباط جمع‌آوری نموده و تسریع مورد انتظار برای این دو مورد را پیش‌بینی نمودند.

سه نتیجه‌ی اصلی از این مطالعات نظری فراهم آمده‌است. نخست آنکه، تسریع مورد انتظار هنگامیکه دسته‌هاایزوله هستند، چندان قابل ملاحظه نیست.(شکل ۷ و ۸ را ببینید) دوم، تسریع هنگامیکه دسته‌ها مرتبط باشند، بسیار به‌تر می‌شود. و در نهایت اینکه، تعداد بهینه‌ای برای دسته‌ها (و به تبع آن یک اندازه‌ی دسته‌ی بهینه) که تسریع را بیشینه می‌نماید، وجود دارد. (شکل ۹ را ببینید)

مدل‌های تعیین‌کننده‌ی ایده‌آل، به چند شیوه می‌توانند بسط داده شوند. برای مثال با در نظر گرفتن، نرخ مهاجرت‌های پایین‌تر یا توپولوژی‌های با اتصال اندک و خلوت. این واقعیتی است که تعداد بهینه‌ی دسته‌ها، تعداد پردازنده‌هایی را که می‌توانند برای کاهش زمان اجرا به‌کار آیند را، محدود می‌کند. استفاده از تعداد بیش‌تری دسته، نسبت به تعداد بهینه، تلف‌کننده است، و ممکن است به الگوریتمی کندتر منجر شود. الگوریتم‌های ژنتیک موازی سلسله‌مراتبی می توانند تعداد پردازنده‌ی افزون‌تری را، مؤثراً به کار گرفته و زمان اجرا را بیش از الگوریتم چند-دسته‌ای محض کاهش دهند.

الگوریتم‌های ژنتیک موازی بسیار پیچیده‌اند، و البته مسائل گوناگونی در آن‌ها هنوز حل نشده باقی مانده است. مثال‌هایی از این دسته (مسائل حل نشده) عبارتند از: (۱) تعیین نرخ مهاجرتی که باعث شود،

دسته‌های توزیع‌شده، همانند یک جمعیت پیوسته رفتار نمایند. (۲) تعیین توپولوژی ارتباطی مناسبی که اجازه‌ی آمیختگی حل‌های خوب را بدهد، اما هزینه‌های ارتباطی زیادی را تحمیل نکند، (۳) یافتن اینکه آیا تعداد بهینه‌ای برای دسته‌ها وجود دارد که قابلیت اطمینان را بیشینه نماید.



(a) 4-bit trap (b) 8-bit trap

Figure 8. *Projected and experimental speed-ups for test functions with 20 copies of 4-bit and 8-bit trap functions using from 1 to 16 isolated demes. The thin lines show the experimental results and the thick lines are the theoretical predictions. The quality demanded was to find at least 16 copies correct.*

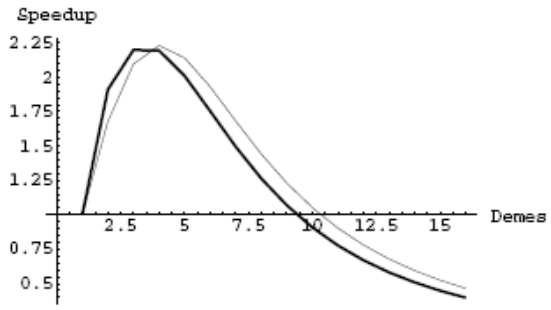
۱۲) خلاصه و نتیجه‌گیری

این مقاله به مرور برخی از مهم‌ترین مقالان منتخب درباره‌ی موضوع الگوریتم‌های ژنتیک موازی پرداخته است. مرور با دسته‌بندی کارهای انجام شده در این موضوع به چهار رسته‌ی: موازی‌سازی سراسری پایه-پیرو، الگوریتم‌های ژنتیک دانه-ریز، چند-دسته‌ای و الگوریتم‌های ژنتیک موازی سلسله‌مراتبی شروع شده است. برخی از مهم‌ترین مقالات در مورد هر یک از این رسته‌ها تحلیل گردید، و تلاش شد که موارد اثرگذار در طراحی و پیاده‌سازی هر رده از الگوریتم‌های ژنتیک موازی در کامپیوترهای موازی موجود، مشخص شود.

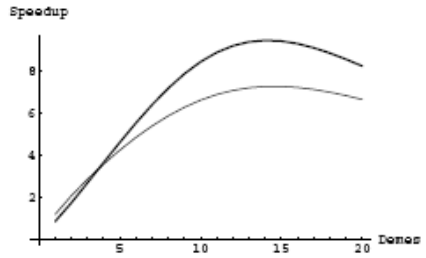
تحقیق درباره‌ی الگوریتم‌های ژنتیک موازی، بیش‌تر بر الگوریتم‌های چند-دسته‌ای متمرکز گردیده و به همین جهت، این مقاله نیز بر آن‌ها متمرکز گردید. بررسی الگوریتم‌های ژنتیک چند-جمعیتی آشکار ساخت که چندین سؤال پایه‌ای در این زمینهٔ تا امروز، سال‌ها پس از معرفی آن برای بار نخست، کماکان بی‌پاسخ مانده‌اند. این رده‌ی الگوریتم‌های ژنتیک موازی بسیار پیچیده است، رفتار آن از پارامترهای فراوانی تأثیر می‌پذیرد. به نظر می‌رسد، تنها راه دست‌یابی به فهمی وسیع‌تر درباره‌ی الگوریتم‌های ژنتیک موازی، مطالعه‌ی مستقل اشکال آن‌هاست. و دیدیم که برخی از تأثیرگذارترین مقالات درباره‌ی الگوریتم‌های ژنتیک موازی بر یک جنبه تمرکز نموده‌اند. (نرخ‌های مهاجرت،

توپولوژی ارتباطی، یا اندازه‌ی دسته‌ها) هر یک دیگر جنبه‌ها را نادیده‌گرفته‌اند یا فرض ساده‌ای درباره‌ی وضعیت آن‌ها نموده‌اند.

مقالاتی درباره‌ی الگوریتم‌های ژنتیک موازی پایه-پیرو و دانه-ریز را مرور نمودیم و دریافتیم که ترکیب استراتژی‌های گوناگون موازی‌سازی می‌تواند به ایجاد الگوریتم‌های سریع‌تری منجر شود. خصوصاً نظرکردن به آمیختگی تکنیک‌های موازی‌سازی در نتایج اخیر، که وجود تعداد دسته‌ی بهینه را پیش‌بینی می‌کنند، حائز اهمیت است.



(a) 4-bit trap



(b) 8-bit trap

Figure 9. *Projected and experimental speed-ups for test functions with 20 copies of 4-bit and 8-bit trap functions using from 1 to 16 fully connected demes. The quality requirement was to find at least 16 correct copies.*

وقتی الگوریتم‌های ژنتیک برای مسائل جست‌وجوی مشکل‌تر و بزرگ‌تر به‌کار گرفته می‌شوند، لازم می‌آید که الگوریتم‌های سریع‌تری طراحی گردند، که قابلیت یافت حل‌های قابل پذیرش را همچنان حفظ کنند. این مرور مثال‌های فراوانی را ارائه نموده، که نشانگر قابلیت الگوریتم‌های ژنتیک موازی در ترکیب سرعت و تأثیرگذاری هستند، و فهم به‌تری از الگوریتم‌های ژنتیک موازی برای ما فراهم نموده‌اند، که به کار گیری به‌تر آن‌ها در آینده را برای ما ممکن می‌سازد.

سپاس‌گزاری

در اینجا میل دارم از دیوید گلدبرگ، به خاطر ارائه‌ی نظرات‌اش درباره‌ی نسخه‌ی اولیه‌ی این مقاله تشکر نمایم. این مطالعه توسط ***Air Force Office of Scientific Research***، ***Air Force Material Command***، ***USAF***، تحت امتیاز شماره‌ی ***F49620-94-1-0103***، ***F49620-95-1-0338*** ، و ***F49620-97-1-0050*** پشتیبانی گردیده است. دولت امریکا مجاز به تکثیر و توزیع آن برای مقاصد دولتی می‌باشد، اما اجازه‌ی هیچگونه تغییر در آن وجود ندارد. دیدگاه‌ها و نتایج ارائه شده در آن نتایج و دیدگاه‌های نویسنده است و لزوماً نباید به عنوان ارائه‌کننده‌ی سیاست‌های رسمی یا صحه‌گذارده‌شده توسط دولت امریکا یا نیروی هوایی، به تصریح یاتلویحاً، تفسیر شود. اریک کنتوپاز توسط بورس تحصیلی ***Fullbright-Garcia Robles-CONCAY*** پشتیبانی گردیده است.

مراجع

[ABR 92] ***ABRAMSON D., ABELA J.,*** *« A parallel genetic algorithm for solving the schooltimetabling problem ».* In Proceedings of the Fifteenth Australian Computer Science Conference (ACSC-15), vol. 14, p. 1–11, 1992.

[ABR 93] ***ABRAMSON D., MILLS G., PERKINS S., «*** *Parallelisation of a genetic algorithm for the computation of efficient train schedules* ». Proceedings of the 1993 Parallel Computing and Transputers Conference, p. 139–149, 1993.

[ADA 94] ***ADAMIDIS P.,*** *« Review of parallel genetic algorithms bibliography ».* Tech. rep.version 1, Aristotle University of Thessaloniki, Thessaloniki, Greece, 1994.

[AND 90] ***ANDERSON E. J., FERRIS M. C.,*** *« A genetic algorithm for the assembly line balancing problem ».* In Integer Programming and Combinatorial Optimization : Proceedings of a 1990 Conference Held at the University of Waterloo, p. 7–18,Waterloo, ON, 1990. University of Waterloo Press.

[BAL 92] ***BALUJA S.,*** *« A massively distributed parallel genetic algorithm (mdpGA) ».* Tech. Rep. No. CMU-CS-92-196R, Carnegie Mellon University, Pittsburgh, PA, 1992.

[BAL 93] ***BALUJA S.,*** *« Structure and Performance of Fine-Grain Parallelism in Genetic Search ».* In FORREST S., Ed., Proceedings of the Fifth International Conference on Genetic Algorithms, p. 155–162, Morgan Kaufmann (San Mateo, CA),1993.

[BEL 95] ***BELDING T. C.,*** *« The distributed genetic algorithm revisited ».* In ESCHELMAN L., Ed., Proceedings of the Sixth International Conference on Genetic Algorithms, p. 114–121, Morgan Kaufmann (San Francisco, CA), 1995.

[BÉS 91] ***BÉSSIERE P., TALBI E.-G.,*** *« A parallel genetic algorithm for the graph partitioning problem ».* In ACM Int. Conf. on Supercomputing ICS91, Cologne, Germany, July 1991.

[BET 76] ***BETHKE A. D.,*** *« Comparison of Genetic Algorithms and Gradient-Based Optimizers on Parallel Processors: Efficiency of Use of Processing Capacity ».* Tech. Rep.No.197, University of Michigan, Logic of Computers Group, Ann Arbor, MI, 1976.

[BIA 93] ***BIANCHINI R., BROWN C. M.,*** *« Parallel genetic algorithms on distributed-memory architectures ».* In ATKINS S., WAGNER A. S., Eds., Transputer Research and Applications 6, p. 67–82, IOS Press (Amsterdam), 1993.

[BRA 90] ***BRAUN H. C.,*** *« On Solving Travelling salesman Problems by Genetic Algorithms».* In SCHWEFEL H.-P., MÄNNER R., Eds., Parallel Problem Solving from Nature, p. 129–133, Springer-Verlag (Berlin), 1990.

[BRA 97] ***BRANKE J., ANDERSEN H. C., SCHMECK H.,*** *« Parallelising Global Selection in Evolutionary Algorithms* ». Submitted to Journal of Parallel and Distributed Computing, January 1997.

[CAN 94] ***CANTÚ-PAZ E., MEJÍA-OLVERA M.,*** *« Experimental Results in Distributed Genetic Algorithms ».* In International Symposium on Applied Corporate Computing, p. 99– 108, Monterrey, Mexico, 1994.

[CAN 97a] ***CANTÚ-PAZ E.,*** *« Designing efficient master-slave parallel genetic algorithms ».* IlliGAL Report No. 97004, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, 1997.

[CAN 97b] ***CANTÚ-PAZ E., GOLDBERG D. E.,*** *« Modeling Idealized Bounding Cases of Parallel Genetic Algorithms ».* In KOZA J., DEB K., DORIGO M., FOGEL D., GARZON M., IBA H., RIOLO R., Eds., Genetic Programming 1997 : Proceedings of the Second Annual Conference, Morgan Kaufmann (San Francisco, CA), 1997.

[CAN 97c] ***CANTÚ-PAZ E., GOLDBERG D. E.,*** *« Predicting speedups of idealized bounding cases of parallel genetic algorithms ».* In BÄCK T., Ed., Proceedings of the Seventh International Conference on Genetic Algorithms, p. 113–121, Morgan Kaufmann (San Francisco), 1997.

[CHE 93] ***CHEN R.-J., MEYER R. R., YACKEL J.,*** *« A Genetic Algorithm for Diversity Minimization and Its Parallel Implementation ».* In FORREST S., Ed., Proceedings of the Fifth International Conference on Genetic Algorithms, p. 163–170, Morgan Kaufmann (San Mateo, CA), 1993.

[COH 87] ***COHOON J. P., HEGDE S. U., MARTINW. N., RICHARDS D.,*** *« Punctuated equilibria : A parallel genetic algorithm ».* In GREFENSTETTE J. J., Ed., Proceedings of the Second International Conference on Genetic Algorithms, p. 148–154, Lawrence Erlbaum Associates (Hillsdale, NJ), 1987.

[COH 91a] ***COHOON J. P., MARTIN W. N., RICHARDS D. S.,*** « *Genetic Algorithms and punctuated Equilibria in VLSI* ». In SCHWEFEL H.-P., MÄNNER R., Eds., Parallel Problem Solving from Nature, p. 134–144, Springer-Verlag (Berlin), 1991.

[COH 91b] ***COHOON J. P., MARTIN W. N., RICHARDS D. S.,*** « *A multi-population genetic algorithm for solving the K-partition problem on hyper-cubes* ». In BELEW R. K., BOOKER L. B., Eds., Proceedings of the Fourth International Conference on Genetic Algorithms, p. 244–248, Morgan Kaufmann (San Mateo, CA), 1991.

[COH 91c] ***COHOON J. P., MARTIN W. N., RICHARDS D. S.,*** *« A Multi-Population Genetic Algorithm for Solving the K-Partition Problem on Hyper-Cubes ».* In BELEW R. K., BOOKER L. B., Eds., Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann (San Mateo, CA), 1991.

[DAV 91] ***DAVIDOR Y.,*** *« A naturally occuring niche and species phenomenon : The model and first results ».* In BELEW R. K., BOOKER L. B., Eds., Proceedings of the Fourth International Conference on Genetic Algorithms, p. 257–263, Morgan Kaufmann (San Mateo, CA), 1991.

[DAV 94] ***DAVIS M., LIU L., ELIAS J. G.,*** *« VLSI circuit synthesis using a parallel genetic algorithm ».* In Proceedings of the First IEEE Conference on Evolutionary Computation, vol. 1, p. 104–109, IEEE Press (Piscataway, NJ), 1994.

[DYM 92] ***DYMEK A.,*** *« An examination of hypercube implementations of genetic algorithms ».* umt, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, 1992.

[FOG 91] ***FOGARTY T. C., HUANG R.,*** *« Implementing the genetic Algorithm on Transputer Based Parallel Processing Systems ».* Parallel Problem Solving from Nature, p. 145–149, 1991.

[GOL 89a] ***GOLDBERG D. E.,*** *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley, Reading, MA, 1989.

[GOL 89b] ***GOLDBERG D. E., KORB B., DEB K.,*** *« Messy genetic algorithms : Motivation, analysis, and first results* ». Complex Systems, vol. 3, n U5, p. 493–530, 1989. (Also TCGA Report 89003).

[GOL 91] ***GOLDBERG D. E., DEB K.,*** *« A comparative analysis of selection schemes used in genetic algorithms ».* Foundations of Genetic Algorithms, vol. 1, p. 69–93, 1991. (Also TCGA Report 90007).

[GOL 92] ***GOLDBERG D. E., DEB K., CLARK J. H.,*** *« Genetic algorithms, noise, and the sizing of populations ».* Complex Systems, vol. 6, p. 333–362, 1992.

[GOL 93a] ***GOLDBERG D. E., DEB K., KARGUPTA H., HARIK G.,*** *« Rapid, accurate optimization of difficult problems using fast messy genetic algorithms ».* In FORREST S., Ed., Proceedings of the Fifth International Conference on Genetic Algorithms, p. 56–64, Morgan Kaufmann (San Mateo, CA), 1993.

[GOL 93b] ***GOLDBERG D. E., DEB K., THIERENS D.,*** *« Toward a better understanding of mixing in genetic algorithms ».* Journal of the Society of Instrument and Control Engineers, vol. 32, n U 1, p. 10–16, 1993.

[GOL 94] ***GOLDBERG D. E.,*** *« Genetic and evolutionary algorithms come of age ».* Communications of the ACM, vol. 37, nU 3, p. 113–119, 1994.

[GOL 96] GOLDBERG D. E., June 1996. Personal communication.

[Gor 89a] ***GORGES-SCHLEUTER M., ASPARAGOS*** *: A population genetics approach to genetic algorithms.* In VOIGT H.-M., MÜHLENBEIN H., SCHWEFEL H.-P., Eds., Evolution and Optimization '89, p. 86–94. Akademie-Verlag (Berlin), 1989.

[GOR 89b] ***GORGES-SCHLEUTER M.,*** *« ASPARAGOS : An asynchronous parallel genetic optimization strategy ».* In SCHAFFER J. D., Ed., Proceedings of the Third International Conference on Genetic Algorithms, p. 422–428, Morgan Kaufmann (San Mateo, CA), 1989.

[Gor 91] ***GORGES-SCHLEUTER M.,*** *« Explicit Parallelism of genetic Algorithms through Population Structures ».* In SCHWEFEL H.-P., MÄNNER R., Eds., Parallel Problem Solving from Nature, p. 150–159, Springer-Verlag (Berlin), 1991.

[GOR 92a] ***GORDON V. S., WHITLEY D., BÖHM A. P.W.,*** *« Dataflow parallelism in genetic algorithms ».* In MÄNNER R., MANDERICK B., Eds., Parallel Problem Solving from Nature, 2, p. 533–542, Elsevier Science (Amsterdam), 1992.

[Gor 92b] ***GORGES-SCHLEUTER M.,*** *« Comparison of local mating strategies in massively parallel genetic algorithms* ». In MÄNNER R., MANDERICK B., Eds., Parallel Problem Solving from Nature, 2, p. 553–562, Elsevier Science (Amsterdam), 1992.

[GOR 93] ***GORDON V. S., WHITLEY D.,*** *« Serial and Parallel Genetic Algorithms as Function Optimizers ».* In FORREST S., Ed., Proceedings of the Fifth International Conference on Genetic Algorithms, p. 177–183, Morgan Kaufmann (San Mateo, CA), 1993.

[GOR 94] ***GORDON V. S.,*** *« Locality in genetic algorithms ».* In Proceedings of the First IEEE Conference on Evolutionary Computation, vol. 1, p. 428–432, IEEE Press (Piscataway, NJ), 1994.

[GOR 97] ***GORGES-SCHLEUTERM.,*** *« Asparagos96 and the Traveling Salesman Problem ».* In BÄCK T., Ed., Proceedings of the Fourth International Conference on Evolutionary Computation, p. 171–174, IEEE Press (Piscataway, NJ), 1997.

[GRE 81] ***GREFENSTETTE J. J.,*** *« Parallel adaptive algorithms for function optimization».* Tech. Rep. No. CS-81-19, Vanderbilt University, Computer Science Department, Nashville, TN, 1981.

[GRO 85] ***GROSSO P. B.,*** *« Computer simulations of genetic adaptation : Parallel subcomponent interaction in a multilocus model ».* Unpublished doctoral dissertation, The University of Michigan, 1985. (University Microfilms No. 8520908).

[GRU 94] ***GRUAU F.,*** *« Neural network synthesis using cellular encoding and the genetic algorithm ».* Unpublished doctoral dissertation, L'Universite Claude Bernard-Lyon I, 1994.

[HAR 97] ***HARIK G., CANTÚ-PAZ E., GOLDBERG D. E., MILLER B. L.,*** *« The gambler's ruin problem, genetic algorithms, and the sizing of populations ».* In Proceedings of 1997 IEEE International Conference on Evolutionary Computation, p. 7–12, IEEE Press (Piscataway, NJ), 1997.

[HAU 94] ***HAUSER R., MÄNNER R.,*** *« Implementation of standard genetic algorithm on MIMD machines ».* In DAVIDOR Y., SCHWEFEL H.-P., MÄNNER R., Eds., Parallel Problem Solving fron Nature, PPSN III, p. 504–513, Springer-Verlag (Berlin), 1994.

[HOL 59] ***HOLLAND J. H.,*** *« A universal computer capable of executing an arbitrary number of sub-programs simultaneously ».* Proceedings of the 1959 Eastern Joint Computer Conference, p. 108–113, 1959.

[HOL 60] ***HOLLAND J. H.,*** *« Iterative circuit computers ».* In Proceedings of the 1960 Western Joint Computer Conference, p. 259–265, 1960.

[KOZ 95] ***KOZA J. R., ANDRE D.,*** *« Parallel genetic programming on a network of transputers ».* Tech. Rep. No. STAN-CS-TR-95-1542, Stanford University, Stanford, CA, 1995.

[KRÖ 91] ***KRÖGER B., SCHWENDERLING P. , VORNBERGER O.,*** *« Parallel genetic packing of rectangles ».* In SCHWEFEL H.-P., MÄNNER R., Eds., Parallel Problem Solving from Nature, p. 160–164, Springer-Verlag (Berlin), 1991.

[KRÖ 92] ***KRÖGER B., SCHWENDERLING P. , VORNBERGER O.,*** *« Massive parallel genetic packing ».* Transputing in Numerical and Neural Network Applications, p. 214–230, 1992.

[KRÖ 93] ***KRÖGER B., SCHWENDERLING P. , VORNBERGER O.,*** *Parallel genetic packing on transputers.* In STENDER J., Ed., Parallel Genetic Algorithms : Theory and Applications, p. 151–185. IOS Press, Amsterdam, 1993.

[LEV 94] ***LEVINE D.,*** *« A parallel genetic algorithm for the set partitioning problem ».* Tech. Rep. No. ANL-94/23, Argonne National Laboratory, Mathematics and Computer Science Division, Argonne, IL, 1994.

[LI 90] ***LI T., MASHFORD J.,*** *« A parallel genetic algorithm for quadratic assignment ».* In AMMAR R. A., Ed., Proceedings of the ISMM International Conference. Parallel and Distributed Computing and Systems, p. 391–394, Acta Press (Anaheim, CA), 1990.

[LIN 94] ***LIN S.-C., PUNCH W., GOODMAN E.,*** *« Coarse-Grain Parallel Genetic Algorithms : Categorization and New Approach ».* In Sixth IEEE Symposium on Parallel and Distributed Processing, IEEE Computer Society Press (Los Alamitos, CA), October 1994.

[LIN 97] ***LIN S.-H., GOODMAN E. D., PUNCH W. F.,*** *« Investigating Parallel Genetic Algorithms on Job Shop Scheduling Problem ».* In ANGELINE P., REYNOLDS R., J. M., EBERHART R., Eds., Sixth International Conference on Evolutionary Programming, p. 383–393, Springer-Verlag (Berlin), April 1997.

[MAN 89] ***MANDERICK B., SPIESSENS P.,*** *« Fine-grained parallel genetic algorithms ».* In SCHAFFER J. D., Ed., Proceedings of the Third International Conference on Genetic Algorithms, p. 428–433, Morgan Kaufmann (San Mateo, CA), 1989.

[MAR 94a] ***MARESKY J. G.,*** *« On effective communication in distributed genetic algorithms ».* Master's thesis, Hebrew University of Jerusalem, Israel, 1994.

[MAR 94b] ***MARIN F. J., TRELLES-SALAZAR O. , SANDOVAL F.,*** *« Genetic algorithms on LAN-message passing architectures using PVM : Application to the routing problem ».* In DAVIDOR Y., SCHWEFEL H.-P., MÄNNER R., Eds., Parallel Problem Solving fron Nature, PPSN III, p. 534–543, Springer-Verlag (Berlin), 1994.

[MER 93a] ***MERKLE L., GATES G., LAMONT G., PACHTER R.,*** *« Application of the Parallel Fast Messy Genetic Algorithm to the Protein Folding Problem ».* Rapport technique, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1993.

[MER 93b] ***MERKLE L. D., LAMONT G. B.,*** *« Comparison of parallel messy genetic algorithm data distribution strategies ».* In FORREST S., Ed., Proceedings of the Fifth International Conference on Genetic Algorithms, p. 191–198, Morgan Kaufmann (San Mateo,CA), 1993.

[MÜH 89a] ***MÜHLENBEIN H.,*** *Parallel genetic algorithms, population genetics, and combinatorial optimization.* In VOIGT H.-M., MÜHLENBEIN H., SCHWEFEL H.-P., Eds., Evolution and Optimization '89, p. 79–85. Akademie-Verlag (Berlin), 1989.

[MÜH 89b] ***MÜHLENBEIN H.,*** *« Parallel genetic algorithms, population genetics and combinatorial optimization ».* In SCHAFFER J. D., Ed., Proceedings of the Third International Conference on Genetic Algorithms, p. 416–421, Morgan Kaufmann (San Mateo, CA), 1989.

[MÜH 91a] ***MÜHLENBEIN H.,*** *« Evolution in time and space-The parallel genetic algorithm ».* In RAWLINS G. J. E., Ed., Foundations of Genetic Algorithms, p. 316–337, Morgan Kaufmann (San Mateo, CA), 1991.

[MÜH 91b] ***MÜHLENBEIN H., SCHOMISCHM. , BORN J.,*** *« The Parallel Genetic Algorithm as Function Optimizer ».* In BELEW R. K., BOOKER L. B., Eds., Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann (San Mateo, CA), 1991.

[MÜH 92] ***MÜHLENBEIN H.,*** *« Darwin's continent cycle theory and its simulation by the prisoner's dilemma ».* In VERELA F. J., BOURGINE P., Eds., Toward a Practice of Autonomous Systems : Proceedings of the First European Conference on Artifical Life, p. 236–244, MIT Press (Cambridge, MA), 1992.

[MUN 91] ***MUNTEAN T., TALBI E.-G.,*** *« A Parallel Genetic Algorithm for processprocessors mapping ».* In DURAND M., DABAGHI E., Eds., Proceedings of the Second Symposium II. High Performance Computing, p. 71–82, Montpellier, France, October, 7-9 1991.

[MUN 93] ***MUNETOMO M., TAKAI Y., SATO Y.,*** *« An efficient migration scheme for subpopulation-based asynchronously parallel genetic algorithms ».* In FORREST S., Ed., Proceedings of the Fifth International Conference on Genetic Algorithms, page 649, Morgan Kaufmann (San Mateo, CA), 1993.

[NEU 91] ***NEUHAUS P.,*** *« Solving the mapping Problem—Experiences with a Genetic Algorithm ».* In SCHWEFEL H.-P., MÄNNER R., Eds., Parallel Problem Solving from Nature, p. 170–175, Springer-Verlag (Berlin), 1991.

[PET 87a] ***PETTEY C. B., LEUZE M. R., GREFENSTETTE J. J.,*** *« A parallel genetic algorithm ».* In GREFENSTETTE J. J., Ed., Proceedings of the Second International Conference on Genetic Algorithms, p. 155–161, Lawrence Erlbaum Associates (Hillsdale, NJ), 1987.

[PET 87b] ***PETTEY C. C., LEUZE M., GREFENSTETTE J. J.,*** *« Genetic algorithms on a hypercube multiprocessor ».* Hypercube Multiprocessors 1987, p. 333–341, 1987.

[PET 89] ***PETTEY C. C., LEUZE M. R.,*** *« A theoretical investigation of a parallel genetic algorithm ».* In SCHAFFER J. D., Ed., Proceedings of the Third International Conference on Genetic Algorithms, p. 398–405, Morgan Kaufmann (San Mateo, CA), 1989.

[ROB 87] ***ROBERTSON G. G.,*** *« Parallel implementation of genetic algorithms in a classifier system ».* In GREFENSTETTE J. J., Ed., Proceedings of the Second International Conference on Genetic Algorithms, p. 140–147, Lawrence Erlbaum Associates (Hillsdale, NJ), 1987.

[SAR 96] ***SARMA J., JONG K. D.,*** *« An analysis of the effects of neighborhood size and shape on local selection algorithms ».* In Parallel Problem Solving from Nature IV, p. 236–244, Springer-Verlag (Berlin), 1996.

[SCH 92] ***SCHWEHM M.,*** *« Implementation of genetic algorithms on various interconnection networks ».* In VALERO M., ONATE E., JANE M., LARRIBA J. L. , SUAREZ B., Eds., Parallel Computing and Transputer Applications, p. 195–203, IOS Press (Amsterdam), 1992.

[SER 94] ***SEREDYNSKI F.,*** *« Dynamic mapping and load balancing with parallel genetic algorithms ».* In Proceedings of the First IEEE Conference on Evolutionary Computation, vol. 2, p. 834–839, IEEE Press (Piscataway, NJ), 1994.

[SHA 94] ***SHAPIRO B., NAVETTA J.,*** *« A massively parallel genetic algorithm for RNA secondary structure prediction ».* The Journal of Supercomputing, vol. 8, p. 195–207, 1994.

[SPI 90] ***SPIESSENS P., MANDERICK B.,*** *« A genetic algorithm for massively parallel computers ».* In ECKMILLER R., HARTMANN G., HAUSKE G., Eds., Parallel Processing in Neural Systems and Computers, Dusseldorf, Germany, p. 31–36, North Holland (Amsterdam), 1990.

[SPI 91] ***SPIESSENS P., MANDERICK B.,*** *« A massively parallel genetic algorithm : Implementation and first analysis ».* In BELEW R. K., BOOKER L. B., Eds., Proceedings of the Fourth International Conference on Genetic Algorithms, p. 279–286, Morgan Kaufmann (San Mateo, CA), 1991.

[STA 91] ***STARKWEATHER T., WHITLEY D. , MATHIAS K.,*** *« Optimization Using Distributed Genetic Algorithms ».* In SCHWEFEL H.-P., MÄNNER R., Eds., Parallel Problem Solving from Nature, p. 176–185, Springer-Verlag (Berlin), 1991.

[TAL 91] ***TALBI E.-G., BESSIÈRE P.,*** *« A Parallel Genetic Algorithm for the Graph Partitioning Problem ».* In Proc. of the International Conference on Supercomputing, Cologne, June 1991.

[TAM 92] ***TAMAKI H., NISHIKAWA Y.,*** *« A paralleled genetic algorihtm based on a neighborhood model and its application to the jobshop scheduling ».* In MÄNNER R., MANDERICK B., Eds., Parallel Problem Solving from Nature, 2, p. 573–582, Elsevier Science (Amsterdam), 1992.

[TAN 87] ***TANESE R.,*** *« Parallel genetic algorithm for a hypercube ».* In GREFENSTETTE J. J., Ed., Proceedings of the Second International Conference on Genetic Algorithms, p. 177–183, Lawrence Erlbaum Associates (Hillsdale, NJ), 1987.

[TAN 89a] ***TANESE R.,*** *« Distributed genetic algorithms ».* In SCHAFFER J. D., Ed., Proceedings of the Third International Conference on Genetic Algorithms, p. 434–439, Morgan Kaufmann (San Mateo, CA), 1989.

[TAN 89b] ***TANESE R.,*** *«Distributed genetic algorithms for function optimization».* Unpublished doctoral dissertation, University of Michigan, Ann Arbor, 1989.

[WHI 90] ***WHITLEY D., STARKWEATHER T.,*** *«Genitor II : A distributed genetic algorithm».* To appear in Journal of Experimental and Theoretical Artificial Intelligence, 1990.