موازی‌سازی الگوریتمهای ژنتیکی با استفاده از اسکلتهای الگوریتمی

حسین دلداری* و تکتم غفاریان**

گروه مهندسی کامپیوتر، دانشگاه فردوسی مشهد

(دریافت مقاله: ۸۰/۱۲/۲۵ – دریافت نسخه نهایی: ۸۲/٤/۳۰)

چکیده – اسکلتهای الگوریتمی[1] به‌عنوان شئـوه‌ای برای برنامه نویسی موازی درسالهـای اخیرمورد توجه محققان قرارگرفته است. با استفاده ازاین شئوه ، برنامه نویس می تواند با بهره‌گیری از یک سری الگوهای پیش ساخته، الگوریتم خود را پیاده سازی کند. دراین مقاله مجموعه‌ای از الگـوهای پیش ساخته با الهام از این شئوه برای پیاده سازی الگوریتمهای ژنتیکی موازی طراحی شده است . برای هریک از الگو های پیشنهادی، مدل کارایی[2] محاسبه شده است. برنامه نویس با استفاده از این مدل ، قادر به مقایسه الگوها وانتخاب بهترین الگو برای کاربرد مورد نظرش است. ضـمناً بـرنامه نویس می‌تواند در هر الگو توپولوژی مجازی پردازشها را تعیین کند. این عمل ضمن افزایش کارایی الگو، کار جدیدی در تحقیقات اسکلتهای الگوریتمی است. از دیگر دستاوردهای این تحقیق قابلیت ترکیب الگوهاست که تنها در معدودی از تحقیقات قبلی دیده شده است.

واژگان کلیدی : اسکلت الگوریتمی ، الگوریتم ژنتیکی موازی ، مدل کارایی ، توپولوژی مجازی ، مهاجرت

Parallel Genetic Algorithm Using Algorithmic Skeleton

H. Deldari, T. Ghafarian
Computer Department, Ferdowsi University of Mashhad

Abstract: *Algorithmic skeleton has received attention as an efficient method of parallel programming in recent years. Using the method, the programmer can implement parallel programs easily. In this study, a set of efficient algorithmic skeletons is introduced for use in implementing parallel genetic algorithm (PGA).A performance model is derived for each skeleton that makes the comparison of skeletons possible in order to select the best one for the application. The performance of the selected skeleton can be increased by specifying the virtual topology required by the application.This is a novel approach with no precedent. Nesting of skeletons used here is another novelty of the study which has been employed only in few previous studies.*

Keywords: *Algorithmic Skeleton, Parallel Genetic Algorithm, Performance Model, Virtual Topology, Migration*

---

* – استادیار ** – کارشناسی ارشد

۱- مقدمه

محاسبات علمی قرن حاضر نیاز به استفاده از رایانه‌هایی با کارایی بالا دارد. با توجه به محدودیت در افزایش سرعت رایانه‌های سری، چنین رایانه‌هایی می توانند رایانه‌های موازی باشند. رایانه‌های موازی امروزه دارای چندین هزار پردازشگراند. این رایانه‌ها دارای تکنولوژی بالایی بوده و از دسترس ما خارج اند. به‌عنوان نمونه، اخیراً شرکت آی ـ بی ـ ام[۳] رایانه موازی بلوجنی[٤] را که دارای یک میلیون پردازشگر پروتیینی است در دست ساخت دارد. از طرفی می توان تعدادی از رایانه‌های شخصی را از طریق یک شبکه قیاس پذیر[٥] به‌هم وصل نمود و از آن به‌عنوان یک رایانه موازی استفاده کرد [۱] . این دو نوع رایانه موازی تحت نرم افزارهایی نظیر ام ـ پی ـ آی[٦] و پی ـ و ـ ام[۷] قابل برنامه نویسی‌اند. ولی مشکل برنامه نویسی با ام ـ پی ـ آی آن است که برنامه نویس با جزییات موازی‌سازی ازقبیل نحوه توزیع داده‌ها برروی پردازشگرها[۸] ، تکنیکهای موازنه کردن میزان پردازش برروی هر یک از پردازشگرها[۹] ، هماهنگی بین پردازشها[۱۰] وارتباط بین پردازشها[۱۱] درگیر می‌شود [۲] که برنامه نویسی را مشکل می‌کنند.

در این تحقیق، برای موازی‌سازی الگوریتمهای ژنتیکی مجموعه‌ای از اسکلتها طراحی شده است. برنامه نویس با استفاده از رابط کاربر طراحی شده برای اسکلتهای مختلف، می‌تواند به سادگی از آنها استفاده کند بدون آنکه از جزییات موازی سازی مطلع شود. ضمناً برای افزایش کارایی می‌توان اسکلتها را با هم ترکیب کرد.

در این مقاله، ابتدا مقدمه‌ای بر اسکلتهای الگوریتمی آورده شده است. سپس در بخش سوم عملکرد یک الگوریتم ژنتیکی ساده مورد بحث قرارگرفته است. در بخش چهارم به انواع روشهای موازی سازی الگوریتمهای ژنتیکی به‌همراه تحقیقاتی که تا به‌حال در این زمینه انجام شده است، اشاره می‌شود. در بخش پنجم الگوهای پیشنهادی برای موازی‌سازی الگوریتمهای ژنتیکی به تفصیل بحث شده است. سپس در بخش ششم، محاسبهٔ مدل کارایی برای پیشگویی زمان اجرای هر یک از اسکلتهای پیشنهادی آورده شده است. در بخش هفتم نتایج حاصل از پیاده سازی الگوهای پیشنهادی برروی شبکه‌ای از رایانه‌های شخصی آمده است و درنهایت مقاله با نتیجه‌گیری و پیشنهاداتی برای تحقیقات آتی خاتمه می‌یابد.

۲- اسکلت الگوریتمی

اسکلت الگوریتمی، یک مدل کلی برای برنامه‌نویسی موازی است که توسط مری کل[۱۲] درسال ۱۹۸۹معرفی شده است [۳]. هدف ازآن ایجاد زبانی است که ضمن برنامه‌نویسی آسان و مستقل از معماری ماشین، برنامه نوشته شده نیز کارایی بالایی داشته باشد. اسکلت، یک الگوی الگوریتمی متداول در برنامه های موازی است [۳]. اسکلتها معمولاً در داخل یک زبان سری میزبان[۱۳] قرار می‌گیرند وتنها منبع موازی سازی در آن زبان محسوب می شوند. به‌عنوان مثال اسکلت *نگاشت*[۱٤]، تابعی را برروی تمام عضوهای یک لیست به‌طور موازی اجرا می‌کند. اسکلت مزرعه[۱٥] ساختار ارباب – بنده[۱٦]را به‌طور موازی پیاده‌سازی می‌کند و اسکلت *تقسیم و حل*[۱۷] یک مسئله را با تقسیمات متوالی بازگشتی، به‌طور موازی حل می‌کند [٤].

مهمترین ویژگی یک اسکلت، کلی بودن آن است که عبارت از قابلیت استفاده از آن اسکلت درکاربردهای متفاوت است. از این رو در اغلب اسکلتها، مجموعه‌ای از توابع وجود دارد که باید توسط استفاده کنندهٔ اسکلت تعریف شود. پس از تعریف این توابع توسط کاربر، با انجام یک پیش پردازش این توابع در مکان مشخصی از اسکلت درج وسپس اجرا می‌شود. از جمله تحقیقات انجام شده در زمینه اسکلتهای الگوریتمی می‌توان به کارهای دارلینگتون[۱۸] [٤]، پلاگاتی[۱۹][٥] و رابهی[۲۰] [٦] اشاره کرد.

معمولاً هر اسکلت دارای یک مدل کارایی است که به‌صورت یک فرمول ریاضی ارایه می‌شود و کاربر با درج پارامترهای مربوط به مسئله خود دراین مدل، قادر به پیشگویی زمان اجرای برنامه‌اش خواهد بود.

۳- الگوریتمهای ژنتیکی

الگوریتمهای ژنتیکی از تئوری تکامل داروین الهام گرفته‌اند. درواقع این الگوریتمها یک مدل محاسباتی از پدیده تکامل بیولوژیکی‌اند، که اولین بار در اوایل دهه هفتاد توسط جان هلند معرفی شدند [۷]. این الگوریتمها را می توان برای حل مسائل بهینه‌سازی یا مدلسازی سیستمهای تکاملی، به‌کار برد [۸و۹].

یک الگوریتم ژنتیکی ساده مطابق شکل (۱) شامل مراحل زیر است: ابتدا پارامترهای مسئله به یک رشته رمزی تبدیل می‌شوند که به این رشته اصطلاحاً کروموزوم می گویند (یک کروموزوم ازمجموعه‌ای از ژنها تشکیل شده است). درواقع هر کروموزوم نسخهٔ به رمز درآورده شده‌ای از راه حلهای مسئله است. سپس جمعیتی از کروموزومها به‌طور تصادفی ایجاد می‌شود و این جمعیت به‌صورت تدریجی از طریق چهار عملگر زیر تکامل پیدا می‌کند.

• ارزیابی[۲۱]: این عملگر از یک تابع ارزیابی برای تعیین برازندگی[۲۲] هر کروموزوم نسبت به سایر کروموزمها استفاده می‌کند. در واقع این عملگر میزان مناسب بودن کروموزوم را برای حل مسئله نشان می دهد.

• انتخاب[۲۳] : این عملگر برازنده ترین کروموزومها را ازجمعیت برای نسل بعدی انتخاب می‌کند.

تقاطع[۲۴] : این عملگر ژنهای تصادفی از کروموزوم های تصادفی را با هم تعویض می کند. این عملگر بر اساس احتمال مشخصی برروی جمعیت اعمال می شود که این احتمال معمولاً بالاست.

• جهش[۲۵]: این عملگر ژنهای تصادفی از کروموزومهای تصادفی را تغییر می‌دهد. احتمال وقوع آن برروی جمعیت کم بوده وهدف از آن حفظ تنوع ژنتیکی جمعیت در جهت میل به سمت راه حل بهینه است.

٤- الگوریتمهای ژنتیکی موازی

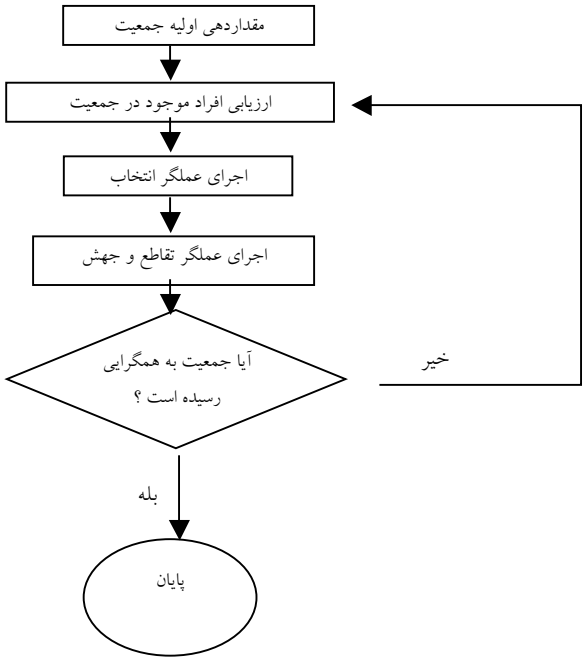
درالگوریتمهای ژنتیکی برای حل یک مسئله، تابع ارزیابی ممکن است صدها یا هزاران بار اجرا شود، بنابراین بسته به زمان لازم برای اجرای تابع ارزیابی، یک الگوریتم ژنتیکی ممکن است ساعتها، روزها یا حتی ماهها برای پیدا کردن یک راه حل قابل قبول طول بکشد. بنابراین این الگوریتمها با توجه به طبیعت موازی جستجوی ژنتیکی و زمان اجرای طولانی، خصوصاً درحل مسائل پیچیده کاربردی، کاندیدای بسیار خوبی برای موازی‌سازی خواهند بود. یک الگوریتم ژنتیکی موازی علاوه بر دارا بودن مزایای یک الگوریتم ژنتیکی ساده از آن سریعتر است. همچنین کمتر مستعد پیدا کردن راه حلهای شبه بهینه است و نیز می توان درطول اجرای آن از سایرتکنیکهای جستجو نیز به طورموازی بهره گرفت [۱۰و۱۱]. در بخشهای بعد شئوه‌های متفاوت موازی‌سازی الگوریتم های ژنتیکی به‌همراه خلاصه‌ای از تحقیقات انجام شده آورده شده است.

٤-۱- الگوریتم ژنتیکی موازی ارباب- بنده[۲۶]

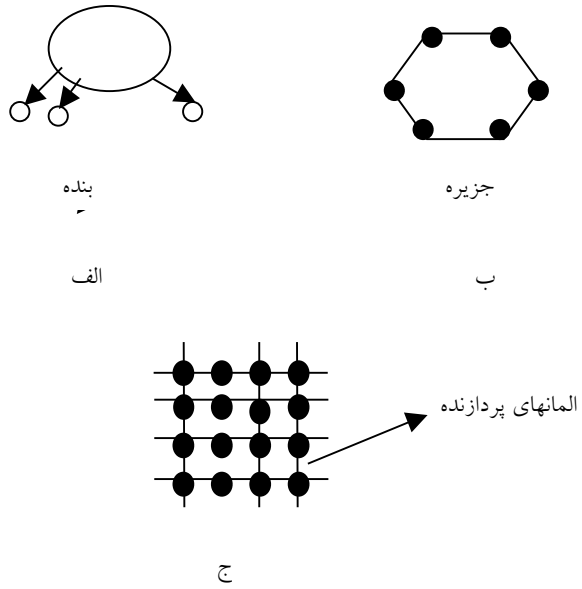
این الگوریتم شامل یک پردازش ارباب و تعدادی پردازش بنده است، شکل (۲- الف). پردازش ارباب الگوریتم ژنتیکی سری را اجرا می‌کند و ضمن اجرای آن، تابع ارزیابی ویا در برخی موارد عملگرهای تقاطع و جهش را به پردازشهای بنده واگذار کرده وسپس نتایج را از آنها دریافت می‌کند [۱۱و ۱۲] .

٤-۲- الگوریتم ژنتیکی موازی جزیره ای[۲۷]

این الگوریتم شامل تعدادی پردازش است که هرکدام یک الگوریتم ژنتیکی سری را به‌طور مستقل اجرا می‌کند. این پردازشها را جزیره می نامند، شکل (۲ ب). این جزایر درفواصل زمانی مناسب، با هم تبادل اطلاعات می نمایند که این تبادل اطلاعات مها جرت[۲۸] نامیده می‌شود. در طی فرایند مهاجرت، تعداد نسبتاً کمی از کروموزومهای هر جزیره در فواصل زمانی مشخص[۲۹] بر اساس توپولوژی خاصی به جزیره دیگر ارسال می‌شوند که این کروموزومها را ، کروموزومهای مهاجر نامند [۱۱و۱۲] . کارهای بسیار زیادی در زمینه انتخاب بهترین توپولوژی مهاجرت [۱۳ - ۱٦] انجام شده است، اما به نظر می‌رسد توپولوژی ابر مکعب[۳۰] بهترین توپولوژی باشد.



شکل۱- نمودار اجرایی یک الگورتم ژنیتیکی ساده



شکل ۲ – الگوی شماتیک الگوریتمهای ژنتیکی موازی

٤-۳- الگوریتم ژنتیکی موازی دانهٔ ریز[۳۱]

معمولاً این الگوریتم بر روی ماشینهای موازی بزرگ با معماری توری[۳۲] دوبعدی پیاده سازی می‌شود، شکل (۲- ج). در این الگوریتم یک یا چند کروموزوم از جمعیت برروی هر پردازشگر قرار گرفته و سپس هر پردازشگر یک الگوریتم ژنتیکی سری را اجرا می‌کند، اما در اجرای الگوریتم ضمن اجرای عملگر انتخاب، بهترین کروموزومها را از میان همسایگان خود (شمال – جنوب –شرق – غرب) انتخاب کرده وسپس عملگرهای تقاطع و جهش را بر روی جمعیت برگزیده اجرا می کند. کروموزومهای فرزند در صورتی‌که بهتر از کروموزومهای موجود در پردازشگر باشند جایگزین آنها می‌شوند [۱۱ و ۱۲]. در این الگوریتم به دلیل همپوشانی بین پردازشگرهای همسایه ، امکان گسترش راه حلهای خوب درکل جمعیت وجود دارد.

٤-٤- الگوریتم ژنتیکی موازی سلسله مراتبی [۳۳]

این الگوریتم از دوالگوریتم ژنتیکی موازی به‌صورت سلسله مراتبی تشکیل می‌شود. در این الگوریتم در بالاترین سطح، یک الگوریتم ژنتیکی موازی جزیره‌ای قرار دارد و هر یک ازگره‌ها به‌طورمستقل یکی از الگوریتمهای ژنتیکی موازی ارباب – بنده، جزیره ای یا دانه ریز را به‌طورمستقل اجرا می‌کند [۱۱ و ۱۲]. درحل مسائل پیچیده، این الگوریتم معمولاً نتایج بهتری را نسبت به سایر الگوریتمهای ژنتیکی موازی به‌دست می‌دهد.

جدول (۱) برخی از تحقیقات انجام شده در زمینه الگوریتمهای ژنتیکی موازی را به طور خلاصه نشان می‌دهد.

۵- الگوهای پیشنهادی برای الگوریتمهای ژنتیکی موازی

اغلب تحقیقات اولیه انجام شده درزمینه اسکلتهای الگوریتمی ، بر ارائهٔ ساختارهای متداول برنامه نویسی موازی متمرکز شده اند که ازآن جمله به الگوهای *تقسیم و حل*، *پردازش لوله‌ای*[۳٤] و *نگاشت* [٦،٤،۳] می‌توان اشاره کرد. درحالی‌که تحقیقات محدودی در زمینه‌های کاربردی نظیر پردازش تصویر، مدل‌سازی جامدات وغیره انجام شده است. از این رو دراین مقاله با توجه به محبوبیت الگوریتمهای ژنتیکی

جدول ۱- چند پیاده سازی از الگوریتمهای ژنتیکی موازی

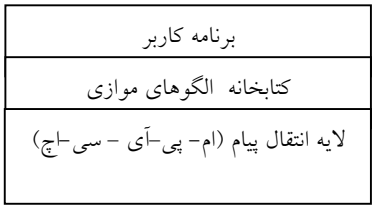
الگوریتم ژنتیکی موازی	مرجع	نوع موازی سازی	توپولوژی	کاربرد
ASPARAGOS	[۱۷]	الگوریتم دانه ریز	نردبانی*	فروشنده دوره گرد
CoPDEB	[۱۸]	الگوریتم جزیره ای	کاملاً متصل	بهینه سازی توابع
DGENESIS 1.0	[۱۶]	الگوریتم جزیره ای	دلخواه	بهینه سازی توابع
ECO-GA	[۱۹]	الگوریتم دانه ریز	توری	بهینه سازی توابع
EnGENEer	[۲۰]	الگوریتم ارباب – بنده	ارباب – بنده	متنوع
GALOPPS 3.1	[۲۱]	الگوریتم جزیره ای	دلخواه	بهینه سازی توابع و حل مسئله حمل و نقل
GAMES	[۲۲]	الگوریتم جزیره ای	------	بهینه سازی توابع و ...
GDGA	[۲۳]	الگوریتم جزیره ای	----	بهینه سازی توابع
GENITOR II	[۲٤]	الگوریتم جزیره ای	حلقه	بهینه سازی توابع
HSDGA	[۲۵]	الگوریتمهای جزیره ای - دانه ریز- سلسله مراتبی	حلقه ، درخت ، ستاره** و...	بهینه سازی توابع
PeGAsuS	[۲٦]	الگوریتمهای جزیره ای – دانه ریز	دلخواه	بهینه سازی توابع.
PGAPack	[۲۷]	الگوریتم ارباب – بنده	ارباب – بنده	بهینه سازی توابع
RPL2	[۲۸]	الگوریتمهای ارباب – بنده و دانه ریز	دلخواه	مسائل تحقیقی و تجاری
SGA-Cube	[۲۹]	الگوریتم ارباب - بنده از ماشین ان-کیوب ۲*** استفاده شده است.	ابر مکعب	بهینه سازی توابع

*Lader **star ***nCUBE2

در میان الگوریتمهای بهینه سازی، مجموعه‌ای از الگوها برای موازی‌سازی الگوریتمهای ژنتیکی به‌صورت کتابخانه‌ای از کلاسها طراحی شده است .

با استفاده از الگوهای طراحی شده، برنامه نویس می‌تواند از محیط برنامه نویسی ویژوال C++ به‌عنوان یک محیط برنامه‌نویسی موازی استفاده کند و با فراخوانی شئ مربوطه، یک الگوی موازی برای برنامه خود انتخاب کرده و آن را با توابع مربوط به مسئله اش پر کند، در این صورت لازم نیست از زیر ساخت الگوی انتخابی و مسائل مربوط به اجرای موازی آن مطلع باشد. البته برنامه نویس می تواند با تعیین توپولوژی مجازی پردازشها درتعیین ارتباط منطقی پردازشها و در نتیجه در زمان اجرای اسکلت نقش داشته باشد که در سایر تحقیقات در زمینه اسکلتهای الگوریتمی نظیر کار دارلینگتون [٤] و پلاگاتی [۵]وجود ندارد. همچنین برای هر یک از اسکلتهای طراحی شده می‌توان یک مدل کارایی به‌دست آورد. در نتیجه کارایی اسکلتها قابل پیشگویی است. این اسکلتها کارایی خوبی دارند درحالی‌که در تحقیقات دیگر [۵ و ۳۰ - ۳۲] چون اسکلتها به برنامه نویسی تابعی پیاده‌سازی شده‌اند پیشگویی کارایی آنها میسر نیست. ضمناً کارایی پایینی دارند زیرا کامپایلرهای تابعی عمل نگاشت پردازشها را نمی‌توانند به بهترین وجه انجام دهند. در این الگوها پردازشهای موازی از طریق تبادل پیام با هم ارتباط برقرار می‌کنند. اما برنامه نویس در استفاده از آنها مستقیماً درگیر توابع مربوط به تبادل پیامها نمی‌شود، بلکه این توابع در کد مربوط به پیاده سازی الگوها گنجانده شده است. بنابراین مطابق شکل (۳) این کتابخانه به‌صورت لایه‌ای بین دستورات سطح پایین برنامه نویسی موازی و برنامه کاربر قرار می‌گیرد.

در این پژوهش، خوشه[۳۵] ای از رایانه‌های شخصی از



شکل ۳- معماری لایه ای یک برنامه کاربردی با استفاده از کتابخانه الگوهای پیشنهادی

طریق پرتکل ایترنت[۳۶] به‌هم متصل شده و به‌عنوان ماشئن موازی استفاده شده است. چنانچه این شبکه ارتباطی قیاس پذیر باشد این نوع ماشئن موازی از کارایی مناسبی برخوردار است. این نوع ماشئن به علت سادگی و نیز ارزانی اخیراً مورد توجه قرار گرفته است. نصب یک مدل انتقال پیام استاندارد از قبیل ام_ پی _ آی یا پی _ و _ ام این شبکه را به صورت یک ماشئن موازی واحد در اختیار کاربر می گذارد. الگوهای انتخاب شده برروی ماشئن موازی فوق پیاده‌سازی شده است ونتایج عملی به‌دست آمده، نتایج واقعی است و هیچ گونه شبیه‌سازی در این پژوهش انجام نشده است. برخی الگوها ترکیبی از الگوهای دیگر هستند که دستاورد جدیدی در تحقیقات انجام شده درزمینه اسکلتهای الگوریتمی است ودر بسیاری از تحقیقات انجام شده در این زمینه از جمله سیستمهایALWAN [۳۳] , HDC [۳٤] انجام نشده است. قبل از معرفی الگوها به اختصار کتابخانه های استفاده شده در این پژوهش آورده شده است.

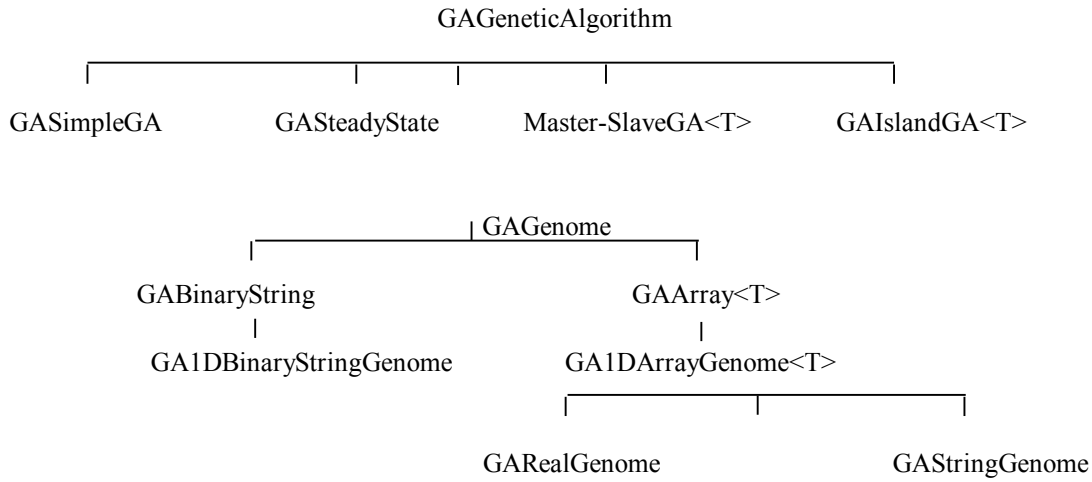
الف- توضیحی بر کتابخانه‌های استفاده شده

کتابخانه الگوهای پیشنهادی ، توسعه یافتهٔ کتابخانه GAlib نسخه۲.٤.٤ [۳۵] است. این کتابخانه در دانشگاه ماساچوست طراحی شده و شامل مجموعه ای از اشئا به زبان C++ برای طراحی الگوریتمهای ژنتیکی سری است ودر آن ابزارهایی برای استفاده از الگوریتمهای ژنتیکی در حل مسائل بهینه سازی در هر برنامه C++ ای فراهم شده است.

قبل از معرفی مجموعه الگوهای پیشنهادی، ابتدا سلسله مراتب کلاسها را شکل (٤) در آنها به اختصار شرح می‌دهیم. در این مجموعه دو شئ پایه به نامهای GAGeneticAlgorithm و GAGenomeداریم .GAGeneticAlgorithm یک کلاس مجرد است که تمام الگوهای پیشنهادی از آن ارث بری می کنند. این شئ سیر تکاملی الگوریتم ونیز مجموعه‌ای از پارامتر ها را نظیر نرخ جهش ، نرخ تقاطع و معیار خاتمه الگوریتم و... را مشخص می کند. دوشئ GASimpleGA وGASteadyState از کلاس GAGeneticAlgorithm ارث بری می کنند وبرای پیاده‌سازی یک الگوریتم ژنتیکی سری می‌توان از آنها استفاده کرد. تفاوت این دو در آن است که در شئ GASteadyState در هر نسل جمعیت جدید به جمعیت قبلی اضافه شده و سپس از جمعیت حاصل بدترین افراد حذف می‌شوند تا اندازه جمعیت ثابت بماند در حالی‌که در شئ GASimpleGA در هر نسل، جمعیت جدیدجایگزین جمعیت قبلی می‌شود. از دو شئ GAIslandGA, Master-SlaveGA برای پیاده سازی یک الگوریتم ژنتیکی موازی می‌توان استفاده کرد. این دو شئ به‌صورت قالب[۳۷] طراحی شده‌اند. به این مفهوم که می‌توان هریک از کلاسهای GASimpleGA و GASteadyState را به‌عنوان الگوریتم ژنتیکی سری به درون آنها ارسال کرد. البته در کلاس GAIslandGA می‌توان خود این کلاس و یا کلاس Master-SlaveGA را نیز درون آن ارسال کرد ویک الگوریتم ژنتیکی سلسله مراتبی ساخت.

برای استفاده از یک الگوریتم ژنتیکی سری باید بتوان راه حل مسئله را به‌صورت ساختمان داده‌ای نمونه‌ای نمایش داد تا بر اساس آن جمعیتی از راه حلها تولید شود. این ساختمان داده نمونه توسط شئ GAGenome تعریف می‌شود و به آن کروموزوم نیز می‌گویند. این کتابخانه شامل دو نوع کروموزوم GABinaryStringGenome وGAArrayGenome است. این کروموزومها از کلاس پایه GAGenome و کلاس مربوط به ساختمان داده شان ارث بری می کنند. به‌عنوان مثال کلاس GABinaryStringGenome از دو شئ GABinaryString و GAGenome مشتق می‌شود.

شئ GABinaryString یک پیاده‌سازی ساده از یک رشته



شکل ٤- سلسله مراتب کلاسها

بیتی است . شئ GA1DBinaryStringGenome کروموزومی را بـه‌صورت یک رشته دودویی تعریف می‌کند این رشته دودویی ممکن است طول ثابت یا متغییر داشته باشد.

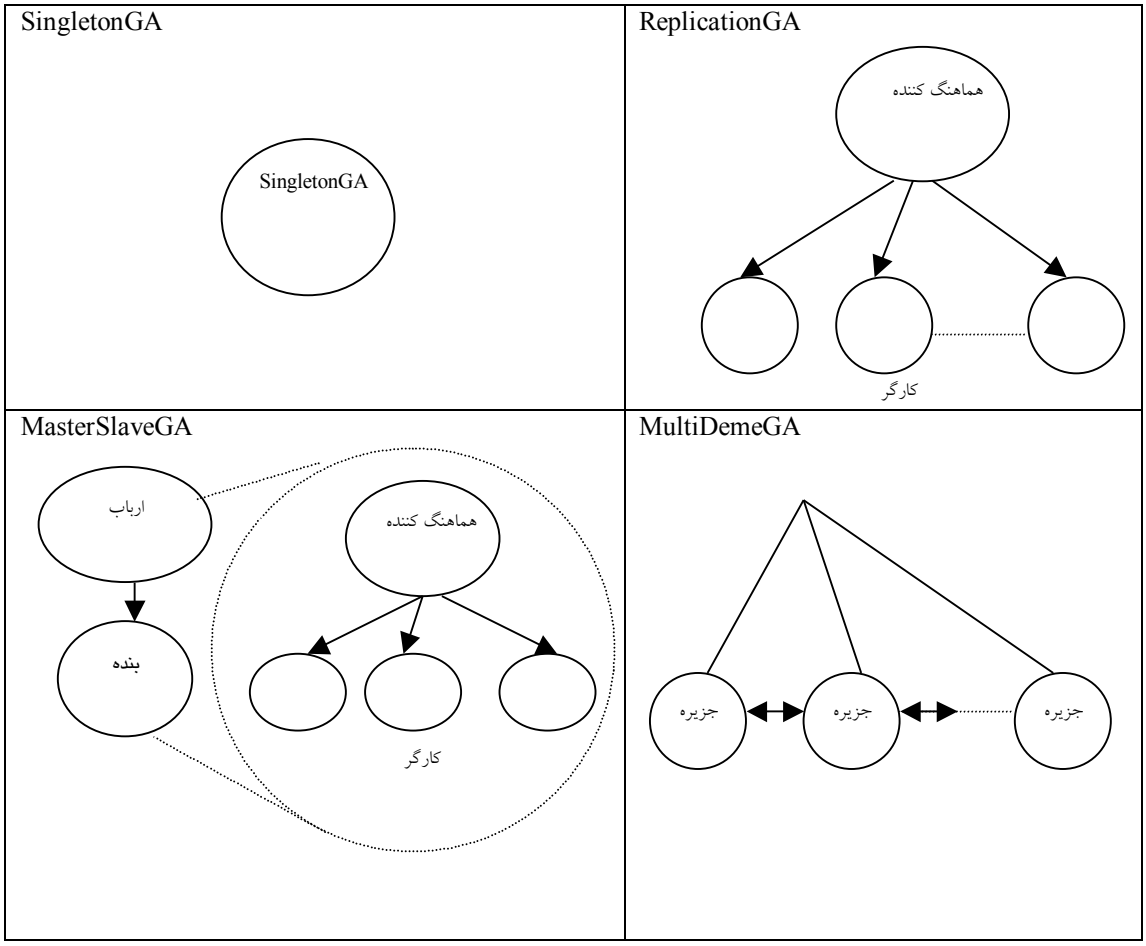
شـئ GAArray<T> یـک پـیاده‌سازی از یک آرایه است و به‌صورت قالب پیاده سازی شده است. هر یک از عضوهای این آرایـــه مـــی‌تـــواند از هـــر نوعـــی باشـــد. شـــئ GA1DArrayGenome<T> کروموزومی را به‌صورت یک آرایه یـک بعـدی تعـریف مـی‌کـند و از دو کلاس GAArray<T> و GAGenome مشـتق مـی‌شود. هر عضو آرایه یک ژن است که نوع آن توسط T در این قالب مشخص می‌شود. به‌عنوان مثال اگر T از نـوع صحیح باشد کروموزومی به‌صورت رشته‌ای از اعداد صـحیح تعـریف مـی‌شـود. مـی‌تـوان بـا استفاده از این کلاس کروموزومهای اعشاری و کاراکتری نیز ساخت اما به‌دلیل کثرت اســتفاده از آنهــا دو کــلاس مجــزا بــه‌صــورت دو شــئ GARealGenome و GAStringGenome در نظـر گـرفته شده است.

در ایـن تحقیق برای ایجاد پردازشهای موازی از کتابخانه ام ـ پـی ـ آی ـسـی ـ اچ[38] نسـخه ۲ .۲. ۱ [۳٦] اسـتفاده شده اسـت. ایـن کتابخانه نسخه‌ای از مدل انتقال پیام ام ـ پی ـ آی است که در دانشگاه میسیسیپی طراحی شده است و می‌توان آن را برروی شبکه‌ای از ایستگاههای کاری اجرا کرد و از آن شبکه بـه‌عـنوان یـک ماشـئن مـوازی واحد استفاده کرد. این کتابخانه اجـازه مـی‌دهـد پردازشـهای روی ماشـئنهای مـتفاوت ازطریق پُـرتکل TCP/IP و یا حافظهٔ اشتراکی با هم ارتباط برقرار کنند. ایـن نسخـه بر روی سیسـتم عامل ویندوز NT / 2000 کارایـی بسیار خوبی دارد و محیطی را برای اجرا از راه دور به‌طور کامل فراهم می کند. در بخش زیر الگوهای پیشنهادی به تفصیل مورد بحث قرار می‌گیرند.

ب- SingletonGA

ایـن اسـکلت، یـک الگـوی تـک پردازشـئ اسـت که یک الگوریـتم ژنتیکـی سـری را اجـرا مـی‌کـند، شکل (۵). دو نوع الگوریـتم ژنتیکـی سـری در ایـن کـتابخانه تعبـیه شـده است: الگوریـتم ژنتیکـی ساده[39] و الگوریتم ژنتیکی حالت پایدار[40] که بـه ترتیب با دو کلاس GASimpleGA و GASteadyStateGA مشـخص شـده و بـه‌عنوان هستهٔ مرکزی سایر الگوها محسوب می‌شوند. واسط برنامه نویسی این الگو به‌صورت زیر است:

```
GASimpleGA ( const GAGenome & g )
GASteadyStateGA ( const GAGenome & g )
```



شکل ۵- اسکلت های پیشنهادی

پارامتر ورودی، متغییری از کلاس GAGenome است. این کلاس ، یک کلاس پایه برای تعیین نوع کروموزوم است و چهار کلاس GA1DBinaryStringGenome,GARealGenome, GAStringGenome , GA1DArrayGenome < int > از آن ارث بری می‌کنند که به‌ترتیب چهار نوع کروموزوم دودویی، اعشاری، کاراکتری و عددی را مشخص می‌کنند. بنابراین می‌توان برای هر کاربردی کروموزوم مناسب با آن را انتخاب کرد.

پارامترهایی نظیر سرعت تقاطع، سرعت جهش، اندازهٔ جمعیت وتعداد نسلها بافراخوانی توابع موجود دراین کلاس مقدار دهی می‌شوند. همچنین بااستفاده ازاین کتابخانه می‌توان نوع عملگر تقاطع را که شامل تک نقطه‌ای[41] ، دونقطه‌ای[42] و یکنواخت[43] است، تعیین کرد. علاوه برآن نوع عملگر انتخاب را که شامل چرخ رولت[44]، تورنومنت و غیره است، نیز می‌توان در الگوریتم ژنتیکی خود تعیین کرد.

ج- ReplicationGA

این الگو شامل یک هماهنگ کننده[45] ومجموعه‌ای از کارگرها[46] است شکل (۵). پردازش هماهنگ کننده، جمعیتی از کروموزومها را ازورودی می گیرد وبسته به تعداد کارگرها، زیر مجموعه‌ای از جمعیت ورودی را به هر یک از کارگرها می‌فرستد. کارگرها به محض دریافت زیر جمعیت، شروع به اجرای عملگر ژنتیکی می‌کنند. این عملگر ممکن است یکی از توابع ارزیابی، تقاطع یا جهش باشد که به‌عنوان پارامتر به این الگو ارسال شده است، سپس نتیجه را به هماهنگ کننده

برمی‌گردانند. پردازش هماهنگ کننده پس از دریافت نتایج از تمام کارگرها، آنها را با هم ترکیب کرده و به‌عنوان خروجی الگو برمی‌گرداند. واسط برنامه نویسی این الگو به‌صورت زیر است:

ReplicationGA(GAPopulation& pop ,int nprocess , char operation , char topology)

• پارامتراول متغیری از کلاس GAPopulation است. این متغیر جمعیتی از کروموزومها را مشخص می‌کند که باید بر روی آنها، یکی از عملگرهای ژنتیکی (ارزیابی ، تقاطع ، جهش) توسط این الگو اجرا شود.

• پارامتر دوم (nprocess)، تعداد کارگرها دراین الگو را مشخص می کند .

• پارامتر سوم (operation)، مشخص می‌کند که کدام یک از عملگرهای ژنتیکی باید توسط کارگرها بر روی جمعیت ورودی اجرا شود. این پارامتر سه کاراکتر E و C و M را به‌ترتیب برای سه عملگر ارزیابی ،تقاطع و جهش می‌گیرد.

• پارامتر چهارم (topology)، توپولوژی مجازی پردازشها را دراین الگو مشخص می کند. توپولوژی مجازی می تواند به صورت ابر مکعب ، توری ، حلقه[47] ویا به‌صورت پیش فرض درخت باشد.

تعیین توپولوژی مجازی یکی ازنقاط قوت این الگو محسوب می‌شود، زیرا دراغلب اسکلتهای معرفی شده تا به‌حال برنامه‌نویس هیچ نقشی درزمان اجرای اسکلت نداشته است، درحالی‌که در این الگو کاربرمی‌تواند باانتخاب توپولوژی مجازی پردازشها در زمان اجرای آن نقش داشته باشد.

د- Master-SlaveGA

این الگو مرکب از دو الگوی ReplicationGA, SingletonGA است، شکل (۵). در این الگو پردازش ارباب درواقع یک SingletonGA ناقص است درحالی‌که بنده یک ReplicationGA است. پردازش ارباب ابتدا جمعیتی از کروموزومها را به‌صورت تصادفی تولید کرده سپس یک الگوریتم ژنتیکی سری را اجرا می‌کند اما در اجرای الگوریتم ژنتیکی، اجرای برخی عملگرها را به پردازش بنده که درواقع نمونه‌ای ازالگوی ReplicationGA است واگذار می‌کند. این عملگرها توسط پارامترهای ورودی الگو مشخص می‌شوند. واسط برنامه نویسی این الگو به‌صورت زیر است:

MasterSlaveGA(const GAGenome& g ,int nprocess ,char opr1,opr2 ,char topology)

• پارامتر اول متغیری ازکلاسGAGenome است. این پارامتر، نوع کروموزوم را در الگوی SingletonGA درون این الگو مشخص می‌کند.

• پارامتر دوم (nprocess)، تعداد کارگرها را در الگوی ReplicationGA درون این الگومشخص می‌کند .

• پارامتر سوم و چهارم (opr1,opr2) عملگرهای ژنتیکی‌ای را که باید توسط الگوی ReplicationGA درون این الگوبه‌طور موازی اجرا شوند، مشخص می‌کند. حداکثر دو عملگر از بین عملگرهای ژنتیکی (ارزیابی ،تقاطع، جهش) را می‌توان در این الگو به بنده واگذار کرد.

• پارامتر چهارم (topology) توپولوژی مجازی پردازشهارا در این الگومشخص می‌کند ومانند پارامترمشابه‌اش در الگوی ReplicationGA مقدار می گیرد

ازجمله نقاط قوت این الگو می‌توان به ترکیب دو الگوی SingletonGA و ReplicationGA اشاره کرد. از دیگر نقاط قوت این الگو تعیین توپولوژی مجازی پردازشها توسط برنامه نویس ودر نتیجه نقش داشتن او در زمان اجرای الگوست.

هـ- Multi-DemeGA

این الگو شامل تعدادی الگوهای کوچک SingletonGA است که به هر یک، زیر جمعیت[48] یا جزیره اطلاق می‌شود، شکل (۵). زیرجمعیتها در شرایط خاصی با هم ارتباط برقرار می‌کنند که این ارتباط، شامل ارسال تعدادی کروموزوم از یک زیرجمعیت به دیگری بوده که مهاجرت نامیده می‌شود. دراین الگو دو نوع مهاجرت (همگام[49] وغیر همگام[50]) درنظر گرفته شده است که درنوع غیرهمگام، مهاجرت بین زیر جمعیت‌ها پس از آنکه هر یک از زیر جمعیتها کاملاً همگرا شدند، اتفاق می‌افتد[۳۷,۱۲] . درحالی‌که درنوع همگام مهاجرت هر چند

نسـل، یکـبار اتفـاق می‌افتد[۳۸]. دراین اسکلت الگوی انتخاب شده برای هر مهاجرت به‌صورت زیر است:

با فرض اینکه زیر جمعیتها به‌طور مجازی و به‌صورت حلقه مـرتب شـده‌انـد، دراولـین مهاجـرت، هـر زیـر جمعـیت، کـروموزومهای مهاجـرش را بـه اولـین همسـایه سـمت چپش مـی‌فرستد. دردومین مهاجرت، هر زیر جمعیت، کروموزومهای مهاجـرش را بـه دومـین همسایه سمت چپش می فرستد و این عمـل تکرار می‌شود تا یک زیر جمعیت طی مهاجرتهای متعدد، کـروموزومهای مهـا جـر خـود را به سایر زیر جمعیت‌ها (بجز خودش) ارسال کند. سپس این چرخه مجدداً تکرار می‌شود. در ایـن الگـو، کروموزومهای مهاجر از بهترین کروموزومها ی زیر جمعـیت انـتخاب شـده و جایگزین بدترین کروموزومهای زیر جمعـیت مقصـد مـی‌شـوند. واسـط بـرنامه نویسی این الگو به صورت زیر است:

GAIslandGA(const GAGenome& g , int nprocess , int migtype , char topology)

• پارامتـر اول متغیـری ازکلاسGAGenome است. این پارامتر، نـوع کـروموزوم را در الگـوی SingletonGA درون ایـن الگـو مشخص می‌کند.

• پارامتـر دوم (nprocess) ، تعـداد زیر جمعیتها (جزایر) را در این الگو مشخص می‌کند.

• پارامتـر سـوم (migtype)، نـوع مهاجـرت درالگو را مشخص می‌کند که ممکن است همگام یا غیر همگام باشد.

• پارامتـر چهـارم (topology)، توپولـوژی مجـازی پردازشها را مشابه سایر الگوها مشخص می‌کند.

ازجملـه نقـاط قـوت این الگو، می‌توان به ترکیب الگوهای کـوچک SingletonGA وتعیـین توپولـوژی مجـازی پردازشـها توسط برنامه نویس اشاره کرد. پارامترهایی نظیر تعداد نسلها بین دو مهاجـرت و تعـداد مهاجـرین از طـریق فراخوانـی روالهای موجود دراین الگو، قابل مقداردهی هستند.

و- تکنیک به رمز درآوری

جمعـیت در هـر الـگـو آرایـه‌ای ازاشـاره گرهـای کـلاس GAGenome است، لذا برای تبادل جمعیت یا بخشئ از آن بین پردازشـگرها، بـا آرایه‌ای ازاشاره گرها سروکارداریم. اما از آنجا کـه مـبادلهٔ اشـاره گرها سبب ایجاد نتایج نادرست می‌شود، لذا بایـد این آرایه یابخشئ ازآن قبل از ارسال به‌صورت رشته‌ای به رمز درآورده شود. این عمل باارسال جمعیت به تابع Encode از کلاس CommunicateGenome انجام می‌گیرد. برای رمزگشایی نیز تابع Decode در این کلاس در نظرگرفته شده است. لازم به توضیح است که فراخوانی این توابع در هر الگو به‌طور خودکار قبل از هر تبادل اطلاعات انجام می‌گیرد.

۵-۱ مثال

بـرای تولـید یـک بـرنامه مـوازی با استفاده ازاین کتابخانه باید مراحل زیر طی شود:

الف- انتخاب کروموزوم مناسب باتوجه به نوع مسئله

ب- انتخاب الگوی مناسب ازکتابخانه

ج- مقداردهی الگو با توجه به مسئله

د- اجرای الگوی انتخابی

در بـرنامهٔ زیـر، مـی‌خواهـیم بـا اسـتفاده از الگـوی GAIslandGA کـروموزوم دودویـی را پـیدا نمایـیم کـه تعداد یـک‌هـای آن ماکـزیمم باشـد. برای این منظور تابع OneMax بـه‌عـنوان تابـع برازندگی تعریف شده است. این تابع بر اساس تعداد یک‌های موجود در کروموزوم یک مقدار برازندگی به آن نسـبت مـی دهـد. (توضـیح: در تابـع OneMax، توابـع genome.length(),genome.gene() بـه ترتیب مقدار ژن i ام در کـروموزوم و طـول کـروموزوم را برمـی‌گردانند.) در برنامه ابـتدا یـک کـروموزوم دودویـی از کـلاس GA1DBinaryStringGenome بنام genome گرفته شده است. طول کروموزوم ۳۲ بیت و تابع OneMax نیز برای تعیین مقدار برازندگـی آن در نظر گرفته شده است، خط (۱). سپس الگوی GAIslandGA از کتابخانه الگوها انتخاب شده است، خط (۲). بـا تـوجه بـه نحـوهٔ پیاده‌سازی این الگو به‌صورت قالب کلاس GASimpleGA بـه‌عـنوان الگوریـتم ژنتیکی سری به درون آن ارسال شده است.سپس با توجه به واسط برنامه نویسی این الگو

کـه اشـاره شد، پارامتر genome به‌عنوان نوع کروموزوم، تعداد زیـر جمعیـتها برابـر پنج، الگوی مهاجرت همگام و توپولوژی مجـازی پردازشـها نیز حلقه انتخاب شده است . در خط (۳) با فراخوانـی تابـع ()nInterval، تعـداد نسـلها بـین دو مهاجـرت متوالـی برابـر پـنج ودر خـط (٤) نیـز بـا فراخوانـی تابـع ()nMigration تعـداد مهاجرین ده کروموزوم تعیین شده است. سـایر پارامترها نظیر تعدادنسلها، اندازهٔ جمعیت، احتمال تقاطع واحتمال جهش از طریق فراخوانی سایر توابع ، قابل مقدار دهی هسـتند. در خط (۵) پارامترهای موجود در این الگو مقداردهی اولیه شده و در خط (٦)، اسکلت اجرا می‌شود.

```
1. GA1DBinaryStringGenome genome(32,OneMax);
2. GAIslandGA<GASimpleGA> ga (genome,5,
   synchronous,ring);
3. ga.nInterval(5);
4. ga.nMigration(10);
5. ga.initialize(seed);
6. ga.RunSkeleton();
```

```
float OneMax( GAGenome & g)
{
double count=0;
GA1D binary String Genome & genome
= ( GA1DBinaryStringGenome &) g;
Float score=0.0;
For (int I=0 ;I<genome.length();I++)
Score+=genome.gene(I);
Score/=genome.length();
return(score);
}
```

٦- مدل کارایی اسکلتهای پیشنهادی

در بخـش زیـر به تفکیک مدل کارایی هر یک از اسکلتهای پیشنهادی آمده است.

الف- ReplicationGA

بـرای محاسبهٔ زمان اجرای این الگو، سه فرض زیر در نظر گـرفته شـده است: ۱- هر پردازش برروی یک پردازشگر اجرا مـی‌شود ۲- تعداد افرادی که در هر مرحله به یک کارگر نسبت داده می شود ثابت است. ۳-زمان اجرای تابع ارزیابی برای تمام کروموزومهای موجود در جمعیت ، ثابت فرض شده است.

زمـان اجرای این الگو شامل دو بخش می‌باشد: T_{comp} که در واقـع زمـان صرف شده برای اجرای عملگرژنتیکی ، توسط هـریک از کارگرهاسـت و T_{comm} کـه شـامل زمـان لازم برای ارسـال بخشـئ از جمعیـت از هماهـنگ کـننده بـه هـر یک از کارگرها ودریافت نتایج از آنهاست. برای محاسبه T_{comp} فرض شده است که فقط تابع ارزیابی به کارگر ها واگذار می‌شود و از دو عملگـر تقاطـع و جهش به دلیل سادگی و کوتاه بودن زمان اجرای شان در مقابل تابع ارزیابی صرف نظر شده است.

اگر n تعداد کل کروموزومهای موجود درجمعیت، T_f زمان لازم بـرای اجرای تابع ارزیابی برروی یک کروموزوم، p تعداد پردازشـگرها ودر نتـیجه $p-1$ تعـداد کارگـرها باشـد در آن صورت :

$$T_{comp} = \frac{nT_f}{p-1}$$

بـرای محاسـبهٔ T_{comm} چـنانچه ارسـال هرزیـرجمعیت از هماهـنگ‌کـننده بـه هـر یـک از کارگرها T_c واحد زمانی طول بکشد، درآن صورت ارسال تمام زیر جمعیتها $(p-1)T_c$ واحد زمانـی طـول خواهدکشـئد. پـس از ار سال زیر جمعیتها مطابق شکل (٦) ، هماهنگ کننده قبل از اتمام کار آخرین کارگر، نتایج کارگـرهای قبلـی را جمع آوری می‌کند سپس T_c واحد زمانی طـول مـی‌کشـد تـا نـتایج آخـرین کارگـر را دریافت کند. پس

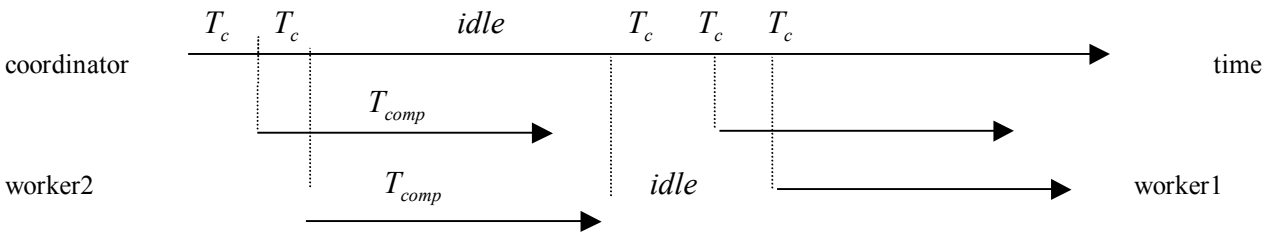
$$T_{comm} = (p-1)T_c + T_c = pT_c$$

$$T_p = T_{comp} + T_{comm} = \frac{nT_f}{p-1} + pT_c \qquad (۱)$$

چنانچه از T_p نسبت به p مشتق بگیریم می توانیم تعداد بهینه کارگر ها را به‌دست آوریم.

$$S^* = p-1 = \sqrt{\frac{nT_f}{T_c}} \qquad (۲)$$

دراین الگو T_c را می‌توان به‌صورت رابطهٔ خطی $T_c = Bx + L$ نمـایش داد که درآن x حجم اطلاعات، B عکس پهنای باند و L زمـان لازم بـرای افـزودن سرامد[۵۱] به هر پیام است و بستگی به سیستم عامل، محیط برنامه نویسی و سخت افزار دارد. لذا زمان لازم بـــرای ارســـال کـــروموزومها بـــه هرکارگـــر



شکل ٦- الگوی شماتیک از زمان اجرای الگوی ReplicationGA

برابر $T_{send} = B\frac{nl_i}{p-1} + L$ و زمان لازم برای دریافت مقدار برازندگی هر زیرجمعیت $T_{rec} = B\frac{nl_f}{p-1} + L$ خواهد بود که درآنها l_i طول هرکروموزوم و l_f طول مقدار برازندگی کروموزوم است .بنابراین زمان تخمین زده شده درهرنسل به‌طور دقیقتر برابر معادلهٔ زیر خواهد بود:

$$T_p = (p-1)T_{send} + T_{rec} + \frac{nT_f}{p-1}$$

ب- Master-SlaveGA

برای محاسبهٔ زمان اجرای این الگو فرض شده است که هر پردازش برروی یک پردازشگر قراردارد و فقط تابع ارزیابی در این الگو توسط بنده که نمونه‌ای از الگوی ReplicationGA است به‌طور موازی انجام می شود .ضمناً" عملگرهای انتخاب، تقاطع وجهش به‌دلیل سادگی و کوتاه بودن زمان اجرای آنها به‌طور سری توسط ارباب انجام می‌شوند. بنابراین T_{comp} دراین الگو برابر معادلهٔ زیر خواهد بود:

$$T_{comp} = T_{selection} + T_{crossover} +$$
$$T_{mutation} + T'_{comp} = nt_{selection} +$$
$$np_{crossover}t_{crossover} + np_{mutation}t_{mutation}$$

که در آن n تعداد کروموزومها ، $p_{crossover}, p_{mutation}$ به ترتیب احتمال جهش واحتمال تقاطع و $t_{mutation}, t_{crossover}, t_{selection}$ به ترتیب زمان اجرای عملگرهای انتخاب ، تقاطع وجهش بوده و T'_{comp} نیز برابر T_{comp} در الگوی ReplicationGA است. در این معادله به‌دلیل کوچک بودن زمان اجرای عملگرهای انتخاب ، تقاطع وجهش در مقابل زمان اجرای تابع ارزیابی می‌توان از این زمانها در مقابل T_f صرف نظر کرد بنابراین

$$T_{comp} \cong \frac{nT_f}{p-1}$$

T_{comm} نیزدر این الگو برابر T_{comm} درپردازش بنده ودر نتیجه در الگوی ReplicationGA بوده و برابر pT_cاست، بنابراین

$$T_p = \frac{nT_f}{p-1} + pT_c$$

در این الگو Speedup که برابر خارج قسمت زمان اجرای الگوریتم ژنتیکی سری، به زمان اجرای الگوریتم موازی است به صورت زیر خواهد بود:

$$Speedup = \frac{nT_f}{\frac{nT_f}{p-1} + pT_c}$$

استفاده از این الگو زمانی مفید است که تابع ارزیابی، یک تابع زمانبرباشد، درصورتی که زمان اجرای تابع ارزیابی کوتاه باشد ممکن است زمان اجرای این الگو از زمان اجرای الگوریتم سری بیشتر شود . در این الگو چنانچه $T_f >> T_c$ و اندازه جمعیت نیز بزرگ باشد می‌توان Speedup ای نزدیک Speedup خطی به‌دست آورد.

ج- Multi-DemeGA

برای محاسبهٔ زمان اجرای این الگو فرضهای زیر در نظر گرفته شده است : ۱- هر زیر جمعیت برروی یک پردازشگر اجرا می‌شود. ۲- اندازهٔ تمام زیر جمعیتها برابر و مساوی n/p است که n تعداد کل کروموزومها و p تعداد زیر جمعیتها یا به

عبارت دیگر تعداد پردازشگرهاست. در این الگو چون هر زیر جمعیت یک SingletonGA است لذا T_{comp} در یک نسل برابر T_{comp} در یک نسل از یک الگوریتم ژنتیکی سری است و برابر $\frac{n}{p}T_f$ است (البته در محاسبهٔ زمان اجرای الگوریتم ژنتیکی سری از $T_{mutation}$ ، $T_{crossover}$ درمقابل T_f صرف نظر شده است).

همان‌طور که گفته شد، دراین الگو در هر بار مهاجرت ، هر یک از زیر جمعیتها به‌طور همزمان درصدی از بهترین کروموزومهای خود را به یکی از زیر جمعیتهای همسایه می‌فرستند بنابراین چنانچه T_c زمان لازم برای ارسال کروموزومها ازیک زیر جمعیت به دیگری باشد در آن صورت $T_{comm} = T_c$ خواهد بود. البته دراین الگو برای جلوگیری از بن بست[۵۲] ، پردازشها به دو گروه زوج و فرد تقسیم می‌شوند به‌طوری‌که ابتدا پردازشها ی زوج وسپس پردازشهای فرد، کروموزومهای مهاجر را ارسال می کنند. بنابراین $T_{comm} = 2T_c$ خواهد شد.

زمان اجرای این الگو پس از وقوع اولین مهاجرت برابر است با

$$T_p = g_c \frac{n}{p} T_f + 2T_c \qquad (۳)$$

که در آن g_c برابر است با تعداد نسلها بین دو مهاجرت، اگر مهاجرت از نوع همگام باشد ویا برابر است با تعداد نسلها تا زمانی که زیر جمعیت به همگرایی برسد، اگر مهاجرت از نوع غیرهمگام باشد.

زمان اجرای کلی این اسکلت برابر است با

$$T_p = k(g_c \frac{n}{p} T_f + 2T_c)$$

که در آن k برابر تعداد دفعاتی است که باید مهاجرت انجام شود تا کیفیت راه حل به‌دست آمده در این الگو با یک الگوریتم ژنتیکی سری هم ارز شود. در این الگو Speedup نیز برابر معادلهٔ زیر است:

$$\text{Speedup} = \frac{gnT_f}{k(g_c \frac{n}{p} T_f + 2T_c)}$$

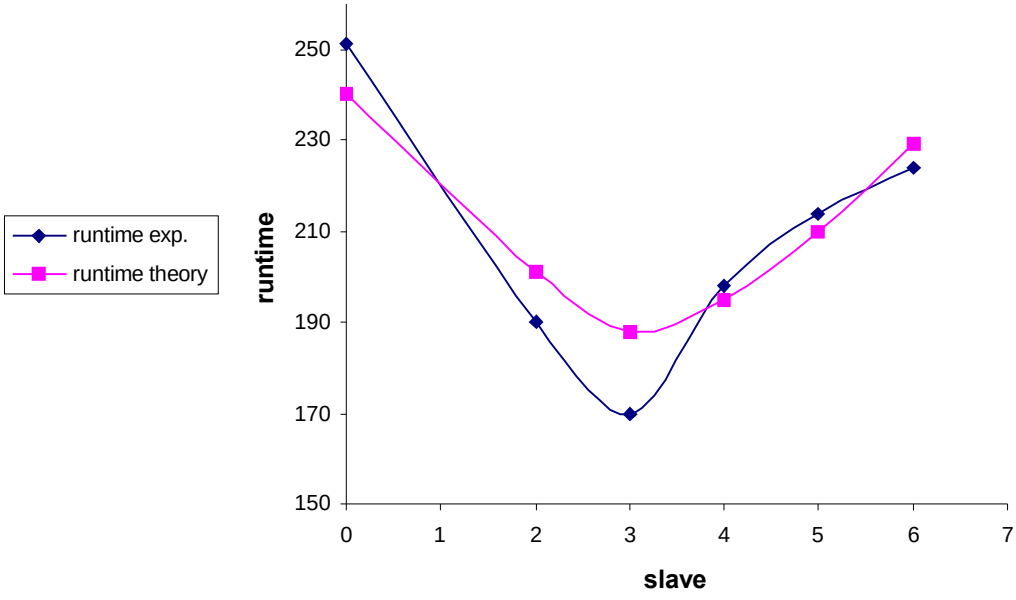
gتعداد نسلها در یک الگوریتم ژنتیکی سری است.

۷- آزمایشهای انجام شده

همان‌طور که در بخش ۵ اشاره شد در این تحقیق، شبکه‌ای از رایانه‌های شخصی به‌عنوان ماشین موازی استفاده شده است واجرای برنامه کامپایل شده بر روی این شبکه و تمامی ارتباطات بین پردازشهای موجود بر روی پردازشگرهای متفاوت در سطح پایین از طریق نرم افزار ام ــ پی ــ آی ــ سی ــ اچ انجام گرفته است. با توجه به کندی نسبی محیط ارتباطی برای آزمون الگوهای پیشنهادی ونشان دادن مزیت استفاده از الگوریتمهای ژنتیکی موازی در حل مسائل زمانبر، استفاده از توابع آزمون ساده نظیر مثال مطرح شده در بخش (۵)، هیچ بهبودی در زمان اجرا نسبت به حالت سری ایجاد نمی‌کند. به همین دلیل در مجموعه آزمایشات انجام شده تاکید بر آن بوده است که تابع ارزیابی یک تابع زمانبر بوده و بار محاسباتی زیادی داشته باشد لذا مسئله‌ای با یک تابع ارزیابی فرضی در نظر گرفته شده است که زمان اجرای آن قابل تغییر است و یکبار زمان اجرا را ۲ میلی ثانیه وبار دیگر ٤ میلی ثانیه گرفته و الگوها را آزموده‌ایم. در آزمونهای انجام شده برای آنکه بتوانیم به لحاظ زمانی، مقایسه درستی بین این الگو و یک الگوریتم ژنتیکی سری داشته باشیم سعی شده است کیفیت راه حل به‌دست آمده توسط هر دو تقریبا" یکسان باشد همان‌طور که در تحقیقات سایرین نیز این مسئله مد نظر قرار گرفته است [۱۲].

برای آزمون هر دو الگو از جمعیتی با ۱۲۰ کروموزم به‌طول ۸۰ استفاده شده است و زمان اجرای تابع ارزیابی بر روی هر کروموزوم یکبار ۲ میلی ثانیه و بار دیگر ٤ میلی ثانیه انتخاب شده است.

شکل (۷) نمودار زمان اجرای الگوی Master-SlaveGA را به‌صورت تجربی وتئوری به ازای $T_f = 2ms$ نشان می‌دهد. در این آزمایش $T_c \cong 27ms$ بوده و بنا به معادلهٔ (۲) تعداد

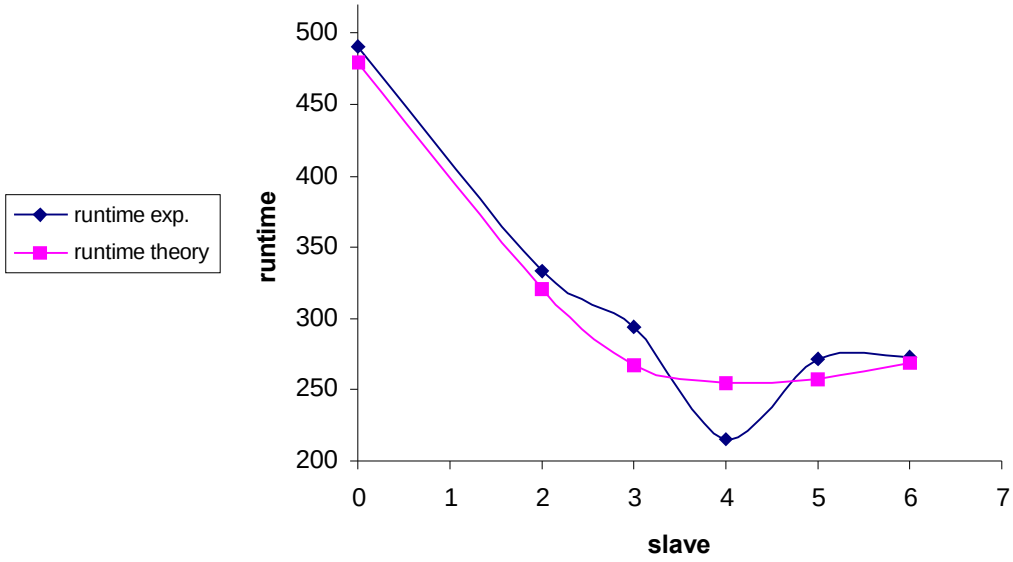


slave	runtime exp.	runtime theory
0	251	240
2	190	201
3	170	188
4	198	195
5	214	210
6	224	229

شکل ۷ –زمان اجرای الگوی Master-SlaveGA **به ازای** $T_f = 2ms$

بهینه بندهها برابر ۳ خواهد شد. به همین دلیل در نمودار زمان اجرای این الگوبا افزایش تعداد بندهها وثابت نگه داشتن اندازه مسئله، به ازای ۳ بنده مسئله کمترین زمان اجرا را خواهد داشت. شکل (۸) نمودار زمان اجرای همین الگو را بهصورت تجربی وتئوری به ازای $T_f = 4ms$ نشان میدهد. در این آزمایش بنا به معادلهٔ (۲) تعداد بهینه بندهها برابر ٤ خواهد شد و در این نقطه مسئله کمترین زمان اجرا را دارد. اما با افزایش تعداد بندهها پس از این نقطه به دلیل افزایش سربار ارتباطی بین پردازشها زمان اجرا افزایش مییابد. با مقایسهٔ دو شکل (۷) و (۸) بهطور تجربی نیز ثابت میشود رفتار الگوریتم با افزایش زمان تابع ارزیابی بهتر میشود همانطور که در بحث مدل کارایی الگو نیز بهطور تئوری به این نتیجه رسیدیم. برای محاسبه T_c دراین الگو با توجه به آنکه T_c زمان ارسال پیامها بین ارباب و بنده است و چون زمان ارسال یک پیام از ارباب به کارگرهای متفاوت است. لذا زمان ارسال اطلاعات بین ارباب و هر یک از کارگرها در بنده جداگانه اندازهگیری شده و سپس متوسط آنها محاسبه شده است. آنگاه آزمایش ۱۰۰ بار انجام شده و متوسط بین این متوسطها بهعنوان T_c انتخاب شده است. نتایج حاصل زمان واقعی اجرای الگوریتم در محیط انتقال پیام با استفاده از نرمافزار ام_پی_آی است و هیچگونه شبیه سازی انجام نشده است.

شکل (۹) نمودار زمان اجرای الگوی Multi-DemeGA را به ازای $T_f = 2ms$ نشان میدهد. دراین آزمایش الگوی

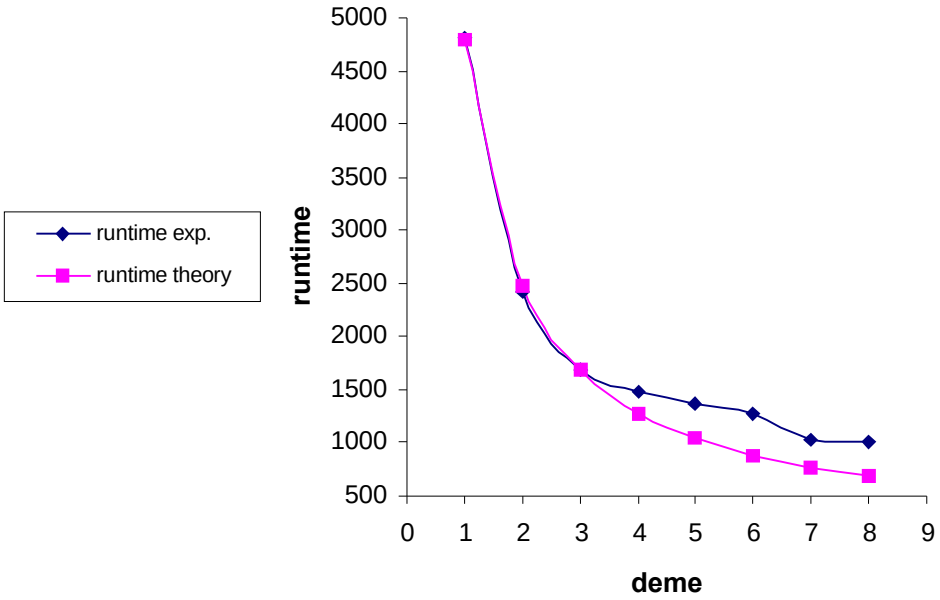


slave	runtime exp.	runtime theory
0	491	480
2	333	321
3	294	268
4	216	255
5	272	258
6	273	269

شکل ۸- زمان اجرای الگوی Master-SlaveGA **به ازای** $T_f = 4ms$

مهاجـرت ازنوع همگام است و مهاجرت هر ۱۰ نسل یکبار اتفـاق مـی‌افـتد. همچنـین درهر بار مهاجرت ۱۰ کروموزوم از بهتـرین کروموزومهای هرزیر جمعیت ، انتخاب می‌شوند. دلیل انتخاب الگوی مهاجرت همگام، همگامی با استفاده از همگامی سـدی کـه یک شئوهٔ آزاد از بن بست است انجام می‌گیرد. ولی در الگـوی مهاجـرت ناهمگـام کـه یـک الگوریتم موازی‌سازی کنترلـی اسـت الگوریـتم باید به نحوی آزاد از بن بست طراحی شود. این نوع همگامی را ام_ پی_ آی انجام نمی‌دهد. بلکه باید در سـطح طراحـی الگوریـتم انجـام گیـرد کـه در ایـن صورت طراحـی الگوریـتم مشـکل خـواهد شـد و کارایی الگوریتم نیز کـاهش مـی‌یابـد. همچنـین در الگـوی مهاجرت همگام برنامه نـویس مـی‌تـواند تعداد دفعات مهاجرت بین جزیره‌ها را تعیین کـند با استفاده از این امکان می‌توان تعداد مهاجرت را تغییر داد و اثـر آن را در کارایی پیدا کرد و بهترین الگوی مهاجرت را در زمـان اجـرا تعیـین کرد که از مزایای این اسکلت است. در این آزمـایش T_c زمـان ارسـال کروموزومها بین جزیره‌های مختلف اسـت و مشـابه مورد فوق اندازه‌گیری شده است و تقریبا" برابر ۲۰ میلی ثانیه است. چون زمان ارسال پیام بین دوجزیره با زمان ارسـال پیام بین ارباب و کارگرها در بنده متفاوت است لذا T_c های آنها نیز متفاوت است.

تـوجه شـود هدف ما در آزمایشات، نشان دادن بهبود نسبی زمـان اجـرای مسئله به‌صورت موازی با چند پردازشگر، نسبت بـه اجـرای آن بـه‌صورت سری با یک پردازشگر است ومقایسه الگوها با هم مد نظر نبوده است. بر روی هر نمودار محل تقاطع نمـودار بـا محـور عمـودی، زمان اجرای الگوریتم را به‌صورت سری، به ازای یک پردازشگر نشان می‌دهد. ملاحـظه می شود با



deme	runtime exp.	runtime theory
1	4810	4800
2	2420	2480
3	1680	1680
4	1470	1280
5	1360	1040
6	1270	880
7	1020	760
8	1010	680

شکل ۹ –زمان اجرای الگوی Multi-DemeGA به ازای $T_f = 2ms$ پس از وقوع دومین مهاجرت

افـزایش تعـداد پردازشـگرها، زمان اجرا تا تعداد پردازشگرهای مشخصـی در الگـوی Master-SlaveGA کـاهش مـی یابـد و سپس مجدداً افزایش می‌یابد. در شکل (۸) به ازای ٤ پردازشگر حـد عملـی از تئوری بهتر است که این امر نشان می‌دهد حجم سـربار ارتباطـی نسبت به حجم محاسبات بسیار کمتر بوده و با تقسیم بار محاسباتی بین ٤ پردازشگر که همان تعداد پردازشگر بهینه است نمودار زمان اجرا افت بیشتری داشته است.

در شـکل (۹) بـه‌دلـیل حجـم بسـیار کـم ارتـباطات بـین پردازشـهای مـوازی نسـبت بـه حجـم محاسباتی، زمان اجرا با افزایش تعداد پردازشگرها مرتباً کاهش می‌یابد.

۸– نتیجه‌گیری

با توجه به استفاده گسترده از الگوریتم های ژنتیکی در حل مسـائل بهیـنه سـازی و ضـرورت موازی‌سازی آنها با توجه به دلایـل یادشـده، درایـن مقالـه مجمـوعه‌ای از الگـوها بـرای الگوریـتمهای ژنتیکـی موازی معرفی شد. الگوهای معرفی شده در ایـن مقالـه کلـی است وبا مقداردهی مناسب پارامترهایشان بـرای انـواع کاربـردها قابل استفاده‌اند. با استفاده از این الگوها مـی‌توان برای بهینه سازی مسائل پیچیده و زمانبر، زمان اجرای بـرنامه را نسبت به استفاده از الگوریتم ژنتیکی سری برای حل همـان مسـئله بـه مقدار قابل توجهی کاهش داد. علاوه بر آن با

استفاده از مدل کارایی هر الگو، می‌توان زمان اجرای برنامه نوشته شده را قبل از اجرای آن تخمین زد. از دیگر ویژگیهای الگوهای معرفی شده ، نقش برنامه نویس در زمان اجرای برنامه موازی با تعیین توپولوژی مجازی پردازشها ونیز ترکیب الگوهای پیشنهادی است. البته راه برای انجام دادن مطالعات تئوری بیشتر در زمینه این الگوها باز است. به‌عنوان مثال در الگوی Multi-DemeGA در زمینه انتخاب مقادیر بهینه برای تعداد مهاجرین، فرکانس مهاجرت، استراتژی انتخاب وجایگزینی مهاجرین هنوز تحقیقی انجام نشده است.

واژه نامه

1. algorithmic skeleton
2. performance model
3. IBM
4. BLUE GENE
5. scable
6. MPI
7. PVM
8. data placement
9. load balancing
10. synchronization
11. communication
12. Murray Cole
13. host language
14. map
15. farm
16. master-slave
17. divide and conquer
18. Darlington
19. Pelagatti
20. Rabhi
21. evaluation
22. fittness
23. selection
24. crossover
25. mutation
26. master-slave parallel genetic algorithm
27. multi-deme parallel genetic algorithm / island parallel genetic algorithm
28. migration
29. migration interval
30. hypercube
31. fine-grain parallel genetic algorithm
32. mesh
33. hierarchical parallel genetic algorithm
34. pipeline
35. MPICH
36. 36.cluster
37. ethernet
38. simple genetic algori
39. steadystate genetic algorithm
40. one point crossover
41. two point crossover
42. uniform crossover
43. roulette wheel selection
44. coordinator
45. worker
46. ring
47. deme
48. synchronous
49. asynchronous
50. template
51. 51.header

مراجع

1. Sterling P., Becker B., Savarese D., et al, "BEOWULF: A Parallel Workstation for Scientific Computation," Proceeding of the 1995, International Conference on Parallel Processing (ICPP), Vol. 1, pp. 11-14, August 1995.
2. Monien B., Diekmann R., Feldmann R., Klasing R., luling R., Menzel K., RomkeT, and Schroeder U.-P., "Efficient Use of Parallel & Distributed Systems from Theory to practice,"*Computer Science Today*, Springer LNCS 1000, pp. 62-77, 1995.
3. Cole, M.I., *Algorithmic Skeletons: Structured Management of Parallel Computation*, Pitman/MIT, London, 1989 .
4. Darlingtons J., Guo Y, To H.W., and Yang J.,"Functional Skeletons for Parallel Coordination," proceeding of 1st EuroPar Conference, Stockholm, Sweden, pp. 55-66, Agust, 1995.
5. Gorlatch S., and Pelagatti, S., "A Transformational Framework for Skeleton Programs: Overview and Case Study," *Proceeding of HIP'99, a Satellite IPPS'99 Workshop*, San Juan, Puerto Rico, USA, LNCS 1586, pp. 123-137, April 1999.
6. Parsons, P.J., and Rabhi F.A.,"Generating Parallel Programs from Skeleton Based Speifications," Journal of Systems Architecture, Vol. 45,pp. 261-283, 1998.
7. Holland J., "Genetic Algorithms and the Optimal Allocation of Trials, " *SIAM Journal on Computing*, Vol. 2(2), pp. 88-105, 1973.
8. Forrest S.,"Genetic Algorithms," *Proceeding of the Fifth International Conference on Genetic Algorithms*, pp. 660, 1996.
9. Withley D, *Foundations of Genetic Algorithms* 2, pp. 45-62, Morgan Kaufmann, San Mateo, 1993.

10. Cantu-Paz E., "A Survey of Parallel Genetic Algorithm," *Calculateurs Paralleles, Reseaux et Systems Repartis*, Vol. 10(2), pp. 141-171, 1998.

11. Alba E., and Troya J.M., "A Survey of Parallel Distributed Genetic Algorithms", Internal T.R., University of Malaga, 1998.

12. Cantu-Paz E.,"Designing Accurate and Efficient Parallel Genetic Algorithm,"PhD Thesis, University of Illinois ,1999.

13. Adamidis P.,"Review of Parallel Genetic Algorithms Bibliograghy," Internal T.R., Aristotle University of Thessaloniki , November 1994.

14. Gordon V.S., and Whitley D.,"Serial and Parallel Genetic Algorithm as Function Optimizers," Proceeding of the 5th ICGA, pp. 177-183,1993.

15. Lin S., Punch W.F., and Goodman E.D.," Coarse-Grain Parallel Genetic Algorithms: Categorization and New Approach," Parallel & Distributed Processing, October 1994.

16. Mejia-Olvera M., and Cantu-Paz E.," DGENESIS-Software Roe Execution of Distributed Genetic Algorithms," Proceeding of XX Conferencia Latinoamericana de Informatica, pp. 935-946, Monterrey, Mexico ,1994.

17. Gorges–Schleuter M.," ASPARAGOS An Asynchronous Parallel Genetic Optimization Strategy," Proceeding of the 3rd ICGA, Schaffer J.D.(ed.) , Morgan Kaufmann,pp. 422-427,1989.

18. Adamidis P., and Petridis V., "Co-operating Populations with Different Evolution Behavior," Proceeding of the second IEEE Conference on Evolutionary Computing, pp. 188-191, 1996.

19. Davidor Y.," A Naturally Occurring Niche & Species Phenomenon: The Model And First Results," Proceeding of the 4th ICGA, Belew R.K., Booker L.B. (eds.),pp. 257-263, 1991.

20. Robbins G., "EnGENEer- The Evolution of Solutions, " Proceeding of the 5th Annual Seminar on Neural Networks and Genetic Algorithms, 1992.

21. Goodman E.D.," An Intoduction to GALOPPS v3.2," TR#96-07-01,GARAGe, I.S.Lab ,Dpt. Of C.S. and C.C.C.A.E.M., Michigan State University, East Lancing ,1996.

22. Potts J.C., Giddens T.D., and Yadav S.B., "The Development and Evaluation of an Improved Genetic Algorithm Based on Migration and Artifitial Selection," IEEE Transaction on Systems, Man, and Cybernetics, 24(1), pp. 73-86, January 1994.

23. Herrera F., and Lozano M., "Gradual Distributed Real-Coded Genetic Algorithms, " Technical Report # DECSAI-97-01-03, Fabruary, 1997. (Revisited Version 98).

24. Whitley, D., and Starkweather T., "GENITOR II: a Distributed Genetic Algorithm, " J. Expt. Theor. Artif. Intelligence 2, pp. 189-214, 1990.

25. Voigt H.M., Santibanez-Koref I., and Born J., "Hierarchically Structured Distributed Genetic Algorithms," Proceeding of the International Coference Parallel Problem Solving from Nature, North-Holland, Amsterdam , pp. 155-164, 1992.

26. Ribeiro Filho, J.L., Alippi C., and Treleaven P., " Genetic Algorithm Programming Environments," Parallel Genetic Algorithms: Theory & Applications, Stender J. (eds), IOS Press, 1993.

27. Levine D.," Users Guide to PGAPack Parallel Genetic Algorithm Library, "T.R. ANL-95/18, January 31, 1996.

28. Radcliffe N.J., and Surry P.D.," the Reproductive Plan Language RPL2: Motivation, Architecture and Applications," Genetic Algorithm in optimization, Simulation and Modeling, Stender J., Hillebrand E, Kingdon J. (eds.), IOS Press, 1994.

29. Erickson J.A., Smith R.E., and Goldberg D.E, SGA-Cube, "A Simple Genetic Algorithm for nCUBE 2 Hypercube Parallel Computers," TCGA Report No. 91005, the University of Alabama, 1991.

30. Michaelson G., Scaife N., Bristow P., and King P.,"Nested Algorithmic Skeletons from Higher order Functions," Submitted for parallel Algorithms and Applications, Special Issue on High Level Models and Languages for parallel processing, September 2000.

31. Scaife N., Bristo P., and Michaelson G.,"Engineering a Parallel Compiler for Standard ML ," in Proceeding of the 10th Intenational Workshop on Implementation of Functional Languages, IFL '98, University College London , pp. 213-225 , September 1998.

32. Skellicorn D.B. , and Pelagatti S. , " Building Programs in the Network of Task Model ," Proeeding of the ACM Symposium on Applied Computing (SAC2000) , Como, Italy , pp. 248-254 , March 2000.

33. Hachler G.," ALWAN: Design and Implementation of a parallel Coordination Language," PhD thesis, Department of Computer University, Basel University, 1997.

34. Herrmann C.A., Lengauer C., Gnz R.,Laitenberger J., and Schaller C.,“ A Compiler for HDC,” Technical Report MIP-9907,Fakultt fr Mathematik und Informatik , Universitt Passau, May 1999.

35. http://lancet.mit.edu/ga

36. ftp://ftp.mcs.anl.gov/pub/mpi/nt/mpich.nt

37. Munetomo, Taakai Y., and Sato Y.,“An efficient migration scheme for SubPopulation-Based Asynchrounously Parallel Genetic Algorithms,” Internal T.R., 1993.

38. Whitely D., Rana S., and Heckendorn R.B.,“Exploiting Separibility in Search: the Island Model Genetic Algorithm,”*Journal of Computing and Information Technology* , 1999.

[DOR: 20.1001.1.22287698.1382.22.2.1.2]