

ارائه یک روش جدید برای افزایش هوشمندی و سرعت الگوریتم کرم شب تاب*

فریبا سلیمانی^۱

نرگس صالح پور^۲

محمد نظری فرخی^۳

چکیده

امروزه اغلب مسائل مهم در صنعت از نوع مسائل بهینه‌سازی غیرخطی و چند پارامتری محسوب می‌شوند. از طرفی، جذابیت رفتار و تعامل جانوران در طبیعت باعث شده است تا دانشمندان علوم رایانه با الهام از این تعاملات، الگوریتم‌هایی برای مسائل بهینه‌سازی ایجاد نمایند که در خیلی از موارد راه‌حل‌های سریع و قابل قبولی برای مسائل پیچیده به همراه دارند. یکی از الگوریتم‌های هوش ازدحامی، الگوریتم کرم شب تاب است که بر اساس نوردهی کرم‌های شب تاب و جذب آن‌ها به سمت نور بیشتر شکل گرفته است. ایراد اصلی الگوریتم‌هایی مانند کرم شب تاب این است که برای همگرا شدن به جواب‌های مورد نظر، نیاز به زمان زیاد دارد. بنابراین، در صورتی که تعداد کرم‌های شب تاب بیش از ۱۲۸ باشد، زمان اجرای آن‌ها با استفاده از سی پی یو ۲/۵۸۲۰ میلی ثانیه اما با استفاده از جی پی یو ۱/۵۰۹۰ میلی ثانیه است. در این مقاله قصد داریم با استفاده از امکانات واحد پردازش کارت گرافیک، نسخه‌ای از الگوریتم کرم شب تاب را ارائه دهیم که همگام با حفظ دقت، با سرعت بیشتری به جواب‌های مورد نظر همگرا شود.

واژه‌های کلیدی: ازدحام ذرات و بهینه‌سازی، جی پی یو، سی پی یو، کرم شب تاب، هوش ازدحامی

* تاریخ دریافت: ۱۳۹۷/۱۲/۱۹؛ تاریخ پذیرش: ۱۳۹۷/۰۴/۲۲.

^۱ کارشناس ارشد مهندسی کامپیوتر، دانشکده فنی مهندسی، دانشگاه آزاد، لرستان، خرم آباد. fariba.solaimani92@yahoo.com

^۲ کارشناس ارشد مهندسی کامپیوتر، دانشکده فنی مهندسی، دانشگاه آزاد، لرستان، خرم آباد. (نویسنده مسئول)

Salehpour_narges@yahoo.com

^۳ دانشجوی دکتری مدیریت فناوری اطلاعات، دانشگاه آزاد اسلامی، واحد علوم و تحقیقات تهران. m_kasitman@yahoo.com

مقدمه

تلاش انسان برای حل مسائل پیرامون خود را می‌توان به نوعی جستجو در جهت تکامل راه‌حل‌های موجود دانست، که در آن با یک تعریف کلی می‌توان تکامل را نوعی جستجو برای به دست آوردن بهترین‌ها تعبیر کرد. مشاهدات انسان از محیط پیرامون باعث شده است که انسان برای زیستن بهتر، دست به درک و شبیه‌سازی رفتار حیوانات، گیاهان و حتی اشیاء بی‌جان بزند. از این نظر رفتارهای اجتماعی بین جانداران به خصوص رفتارهای اجتماعی بین حشراتی مانند مورچه‌ها و کرم شب‌تاب بیشتر مورد توجه انسان بوده است. معمولاً حل مسائل بهینه‌سازی صنعتی با روش‌های ریاضی، فرآیندی زمان‌گیر است. در حالی که الگوریتم‌های هوش ازدحامی معمولاً سریع‌تر عمل می‌کنند. الگوریتم‌های هوش ازدحامی با استفاده از فرآیندهای تصادفی و البته هوشمندانه سعی می‌کنند تا به بهترین جواب‌های یک مسئله بهینه‌سازی برسند. نتایج بررسی‌ها و تحقیقات مختلف نشان می‌دهد، الگوریتم‌های هوش ازدحامی در بیشتر مواقع با بهینه‌های سراسری^۱ همگرا^۲ می‌شوند. ولی با این وجود این احتمال نیز وجود دارد که در بهینه محلی^۳ گرفتار شوند. هر الگوریتم تکاملی به علت غیرقطعی^۴ بودن ماهیت آن‌ها، مستعد گرفتار شدن در بهینه‌های محلی می‌باشد. با وجود این که افزایش تکرار و جمعیت اولیه الگوریتم‌های تکاملی^۵، شانس یافتن بهینه سراسری را افزایش می‌دهد، زمان اجرای الگوریتم تکاملی نیز به شکل ناخوشایندی افزایش می‌یابد. به‌طور طبیعی تکرار زیاد این گونه الگوریتم‌ها بخصوص در جمعیت‌های انبوه، زمان اجرای الگوریتم را افزایش می‌دهد (ایکس. یانگ، ۲۰۱۰). در این مقاله با استفاده از امکانات واحد پردازش کارت گرافیک، نسخه‌ای از الگوریتم کرم شب‌تاب را ارائه خواهیم داد که در حین حفظ دقت با سرعت بیشتری به جواب‌های مورد نظر همگرا می‌شود. الگوریتم پیشنهادی در این تحقیق با الگوریتم‌هایی مانند الگوریتم

¹ Swarm intelligence algorithms

² Global Optimization

³ Convergent

⁴ Local optimization

⁵ non-deterministic

⁶ Evolutionary Algorithms

ذرات، ژنتیک و الگوریتم استاندارد کرم شب تاب از نظر زمان اجرا و دقت مقایسه شده است. نتایج نشان می دهد که الگوریتم پیشنهادی موفق و کارآمد است.

پیشینه پژوهش

با توجه به این که اجرای الگوریتم های هوش ازدحامی برای حل مسائل بهینه سازی بر روی سی پی یو^۱ به شدت زمان گیر و هزینه بر است، این الگوریتم ها برای همگرا شدن به جواب های مطلوب و با دقت مناسب، نیاز به تکرار زیادی دارند. الگوریتم کرم شب تاب یک الگوریتم هوش ازدحامی موفق (ایکس.اس، یانگ ۲۰۱۳) و دارای دقت مناسبی برای رسیدن به جواب های بهینه است اما برای رسیدن به جواب های بهینه و دقیق نیاز است تا تعداد جمعیت اولیه و تکرارهای الگوریتم افزایش یابد. افزایش جمعیت اولیه و تعداد تکرارها در الگوریتم کرم شب تاب فرآیندی بسیار زمان بر است. واحد پردازش کارت گرافیک دارای مزایای بسیار زیادی است. تعداد هسته های فراوان این پردازنده ها و همچنین پهنای باند زیاد حافظه کارت های گرافیکی، قابلیت اجرای تعداد نخ های موازی زیادی را در مقایسه با سی پی یو به این پردازنده داده است. یکی از انگیزه های این تحقیق استفاده از سخت افزارهای جانبی کامپیوتر در پردازش موازی و بهبود دقت و سرعت الگوریتم کرم شب تاب است.

هوش ازدحامی حاصل جمع یا برآیند حاصل از هوش تعدادی عامل ساده می باشد (اس. سی. چو، ۲۰۱۱). هوش ازدحامی، در برگیرنده گستره وسیعی است که سعی در حل مسائل با تقلید از شیوه رفتار جمعیت های موجود در طبیعت را دارد. در دسته هایی که دارای هوش ازدحامی هستند، اطلاعات زیادی وجود دارد که ناشی از تجمع هوش گروه است و این اطلاعات، گروه را قادر می سازد تا مشکلات خود را حل کنند. هوش ازدحامی، منبع الهام بخشی برای محققان و الگوی جدیدی برای حل مسئله است. گروه ها، برای حل مسائل از عوامل متعددی استفاده می کنند. رفتار گروه مرتبط با یکدیگر و مشابه

به هم است و همین ویژگی‌ها، گروه را قادر به حل مشکلاتی می‌کند که حل بهینه آن برای روش‌های سنتی، در بازه زمانی مشخص مشکل است. از جمله الگوریتم‌های مطرح در این زمینه می‌توان به الگوریتم‌های بهینه‌سازی ازدحام ذرات اشاره کرد (ای. شوکلا، ۲۰۱۰). این دسته از الگوریتم‌ها دارای دو مؤلفه اصلی می‌باشند:

- ۱- خود ترتیبی^۱، به معنی تعریف یکسری قوانین و توابع که تمامی اعضای گروه ملزم به رعایت و اجرای این قوانین می‌باشند.
 - ۲- جریان اطلاعاتی^۲، در هر مرحله پس از اجرای قوانین توسط اعضای گروه، اطلاعات حاصل می‌بایست بین تمامی اعضای گروه مبادله شود.
- مهم‌ترین ویژگی الگوریتم هوش ازدحامی به تولید یک سیستم مستقل، تطبیقی، مقیاس‌پذیر، انعطاف‌پذیر، خودسازمانده، مقرون‌به‌صرفه، مبتنی بر جمعیت و برای بهینه‌سازی غیرخطی می‌توان اشاره کرد (اس. ایکس. وو، ۲۰۱۰).

الگوریتم ازدحام ذرات

الگوریتم بهینه‌سازی ازدحام ذرات یک الگوریتم بهینه‌سازی فرااکتشافی با روش جستجوی جمعیت است که از حرکت گروهی پرندگان و دیگر حیواناتی که به شکل گروهی حرکت می‌کنند الگو گرفته است (اس. سی. چو، ۲۰۱۱).

در علوم کامپیوتر، بهینه‌سازی ازدحام ذرات یک روش محاسباتی است که سعی در بهینه‌سازی روش حل مسئله دارد. برای رسیدن به پاسخ بهینه، الگوریتم مورد نظر پس از تکرار چندین باره حل مسئله و به دست آوردن جواب‌های کاندید، پس از ارزیابی کیفی پاسخ‌های کاندید، بهترین پاسخ از میان راه‌حل‌های به دست آمده انتخاب می‌شود (اس. سی. چو، ۲۰۱۱، آر. پولی).

در الگوریتم ازدحام ذرات، هر پاسخ مسئله به صورت یک ذره می‌توان مدل کرد که دارای یک مقدار سرعت و همچنین میزان تناسب است. طبق ایده مطرح در الگوریتم

¹. Self-Organization

². Information Flow

ارائه یک روش جدید برای افزایش هوشمندی و سرعت الگوریتم کرم شب تاب // ۱۳

بهینه‌سازی ازدحام ذرات، موارد زیر تعریف می‌شوند (اس. سی. چو، ۲۰۱۱؛ آر. شوکلا، ۲۰۱۰؛ ام. ایمران، ۲۰۱۳؛ جی. کندی، ۱۹۹۵):

- هر عضو گروه یک ذره نامیده می‌شود. هر ذره در حال جستجو برای نقطه بهینه است.

- برای یافتن نقطه بهینه، هر ذره در حال جابجایی است.

- به دلیل جابه‌جایی ذرات، هر ذره دارای سرعت است.

- الگوریتم بهینه‌سازی ذرات، بر مبنای حرکت و هوش ذرات عمل می‌کند.

- الگوریتم بهینه‌سازی ذرات، مفهوم تعامل اجتماعی را برای حل مسائل بهینه‌سازی به کار می‌گیرد.

- ذرات در فضای جستجو حرکت می‌کنند.

الگوریتم کرم شب تاب

امروزه مسائل بهینه‌سازی یکی از مسائل پیشرو در صنعت است. مسائل بهینه‌سازی معمولاً با یک تابع غیرخطی نشان داده می‌شوند. هدف، پیدا کردن نقاط اکسترمم این تابع است. روش‌های ریاضی برای حل این مسائل معمولاً توصیه نمی‌شوند بدین دلیل که حل آن‌ها سخت و زمان‌بر است (جی. اگروال، ۲۰۱۳). الگوریتم کرم شب تاب، الگوریتم بهینه‌سازی است که از طبیعت الهام گرفته است. هدف اصلی، نور ساطع شده کرم شب تاب است که به عنوان یک سیستم سیگنال برای جذب کرم شب تاب دیگر عمل می‌کنند. در این الگوریتم، تنوع نور در شدت و تدوین جذابیت دو موضوع مهم هستند (ای. وی. هاسولمن، ۲۰۱۲).

فرموله کردن الگوریتم کرم شب تاب

در الگوریتم کرم شب تاب نیاز است که روشنایی کرم‌های شب تاب را در یک مکان مشخص با استفاده از یک رابطه ریاضی مدل کرد. رابطه (۱) حالت کلی سازی را نشان می‌دهد:

رابطه (۱) $I(x) \propto f(x)$

در رابطه (۱)، $I(x)$ مقدار شدت روشنایی کرم شب تاب در نقطه x و $f(x)$ تابعی است که روشنایی را مدل می کند.

در رابطه قبل فرض شده است که منبع نور کرم شب تاب نقطه ای باشد و توان نقطه ای بودن به صورت $m=2$ در نظر گرفته شده است. در حالت کلی منبع نور کرم های شب تاب کاملاً نقطه ای نیست، بلکه تا حدودی نقطه است برای اینکه رفتار نور پراکنی کرم های شب تاب کاملاً نقطه ای فرض نشود. همچنین توان را می توان بزرگ تر از ۲ در نظر گرفت. مقدار m می تواند بین مقادیر ۲ تا بی نهایت متغیر باشد. مقدار $m=2$ منبع نورهای کرم شب تاب را نقطه ای و $m=\infty$ منبع نورهای کرم شب تاب را از حالت نقطه ای خارج می کند و منبع نور را مسطح فرض می کند. در حالت کلی شدت جذب کرم های شب تاب در رابطه (۲) نشان داده شده است:

$$\beta(r) = \beta_0 e^{-\gamma r^m} \cong \frac{\beta_0}{1+\gamma r^m} \quad \text{رابطه (۲)}$$

در الگوریتم کرم شب تاب، برای جابه جایی کرم های شب تاب از نقطه ای با جذابیت کمتر به نقطه ای با جذابیت بیشتر از روابط استفاده می شود. در رابطه (۲) نحوه انتقال و حرکت یک کرم شب تاب بین دو نقطه خاص، نشان داده است (ایکس.یانگ، ۲۰۱۰):

$$=x_i^t + \beta \cdot \exp(x_i^{t+1}) \quad \text{رابطه (۳)}$$

$$(-\gamma r_y^2) \cdot (x_j^t - x_i^t) + \alpha \epsilon_t$$

در رابطه (۳)، ϵ_t پارامتر کنترل اندازه گام است که بر اساس توزیع گوسی یا یکنواخت تولید می شود. α_i را ضریب القا می گویند که باعث می شود حرکت کرم های شب تاب لزوماً روی یک خط راست صورت نگیرد.

مشکلات الگوریتم کرم شب تاب

الگوریتم های تکاملی و هوش ازدحامی از گرفتار شدن در نقاط بهینه محلی و زمان اجرای طولانی رنج می برند. الگوریتم کرم شب تاب نیز به عنوان یک الگوریتم هوش ازدحامی از این چالش مستثنی نیست. یکی از روش های بهبود دقت و فرار از نقاط بهینه

محلی، افزایش اندازه جمعیت اولیه کرم‌های شب تاب و تعداد تکرار الگوریتم کرم شب تاب می‌باشد. با افزایش تعداد کرم‌های شب تاب و پراکنش آن‌ها در فضای مسئله در واقع شانس یافتن بهینه‌های سراسری بیشتر می‌شود. تکرار زیاد نیز منجر به همگرا شدن جمعیت کرم‌های شب تاب به جواب‌های بهینه می‌شود. اعمال چنین روشی، زمان اجرای الگوریتم تکاملی را بیشتر می‌نماید با این تفاوت که در اکثر موارد قابل قبول نیست.

معماری کودا

امروزه جی‌پی‌یو^۱ به عنوان یک پردازنده با منابع محاسباتی بسیار زیاد مورد توجه قرار گرفته است (جی.دی. اوونس، ۲۰۰۸). تعداد هسته‌های فراوان این پردازنده و همچنین پهنای باند زیاد حافظه کارت‌های گرافیکی، قابلیت اجرای تعداد نخ‌های موازی بسیار زیادی را در مقایسه با سی‌پی‌یو به این پردازنده داده است. به طور کلی ساختار جی‌پی‌یو به گونه‌ای است که با پردازش دسته‌ای داده‌ها، بازده محاسبات را افزایش می‌دهد.

جی‌پی‌یوها در ابتدا به صورت سخت‌افزارهایی با کارکرد ثابت و برای محاسبات گرافیکی به وجود آمدند. تبدیل مسئله هدف به یک مسئله گرافیکی، خود موضوعی چالش برانگیز بود که استفاده از جی‌پی‌یوها را مشکل و در نتیجه محدود می‌کرد. در نهایت سازندگان این تراشه‌ها به فکر ارائه یک معماری عام به همراه یک زبان سطح بالای برنامه‌نویسی افتاده و مفهوم جی‌پی‌یو همه‌منظوره یا جی‌پی‌یو^۲ را معرفی کردند. از جمله مهم‌ترین این معماری‌ها کودا است که در آن هر جی‌پی‌یو شامل چندین واحد پردازشی مستقل به نام جریان‌های چندپردازنده‌ای^۳ است. هر یک از این واحدها نیز به نوبه خود حاوی چندین هسته اجرایی هستند (زد. یو، ۲۰۱۱).

برنامه‌ای که مطابق با معماری کودا نوشته شده باشد شامل دو جز سری و موازی است که قسمت سری آن بر روی سی‌پی‌یو و قسمت موازی آن بر روی جی‌پی‌یو در قالب توابع خاصی به نام کرنل اجرا می‌گردند. در حین اجرا، ابتدا برنامه توسط سی‌پی‌یو شروع

¹. GPU

². GPGPU

³. Streaming Multi processor

به کار کرده و به صورت سری و خط به خط جلو می‌رود. اگر یک خط خاص شامل یک فراخوانی کرنل باشد. در نهایت جی‌پی‌یو وارد عمل شده و تعداد زیادی نخ موازی را که تمامی آن‌ها تابع کرنل را اجرا می‌کنند؛ ایجاد می‌گردد. در معماری کودا فراخوانی یک تابع کرنل در قالب ایجاد یک تور از نخ‌ها صورت می‌گیرد که هر تور حاوی چندین بلوک بوده که هر بلوک نیز حاوی چندین نخ است.

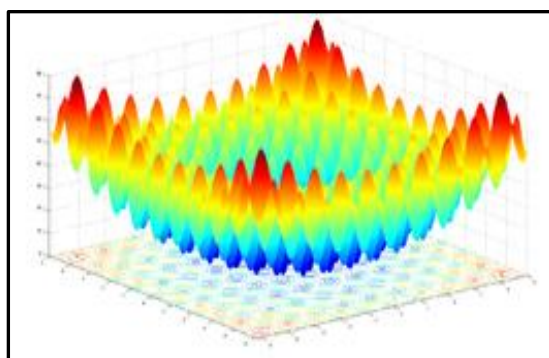
در طی اجرای این طرح، دستیابی به اهداف ذیل مورد انتظار است:

- ۱- افزایش سرعت حل مسائل بهینه‌سازی
- ۲- افزایش سرعت و دقت الگوریتم هوش دسته‌جمعی کرم شب‌تاب
- ۳- استفاده از سطح بالایی از موازی‌سازی پردازش‌ها از طریق اجرای هزاران رشته پردازشی سخت‌افزاری جهت حل مشکل زمان طولانی محاسبات به عنوان یکی از عوامل بازدارنده در حل مسائل بهینه‌سازی
- ۴- از آنجا که حافظه کارت‌های گرافیکی با سرعت زیادی افزایش می‌یابد، استفاده از حافظه آن‌ها به عنوان حافظه موقت، موجب کاهش ترافیک اطلاعات در گذرگاه کارت PC می‌شود و در نتیجه کارایی افزایش می‌یابد.
- ۵- فراهم نمودن امکان تبادل ناهمگام اطلاعات بر پایه پردازنده‌های گرافیکی

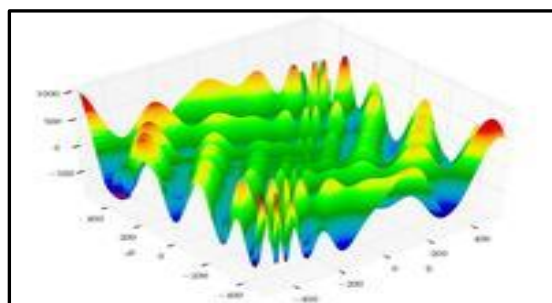
روش‌شناسی پژوهش

در این مقاله، نسخه الگوریتم پیشنهادی با الگوریتم‌های مانند ژنتیک، ذرات و الگوریتم کرم شب‌تاب از نظر دقت، سرعت اجرا و همگرایی مقایسه شده است. برای ارزیابی و تحلیل جواب‌ها از توابع بهینه‌سازی موسوم به توابع بنچمارک استفاده شده است. توابع بنچمارک، نوزده تابع ریاضی با اشکال پیچیده هستند. تابع بنچمارک با علائم F1 تا F19 مشخص شده‌اند و از این توابع برای سنجش همگرایی الگوریتم‌های تکاملی استفاده

می شود (جی.ژووا، ۲۰۱۰). در شکل (۱) تابع بنچمارک راستریجن^۱ و در شکل (۲) تابع بنچمارک اگهولدر^۲ نشان داده شده اند.



شکل ۱. تابع بنچمارک راستریجن در مسائل بهینه سازی



شکل ۱. تابع بنچمارک اگهولدر در مسائل بهینه سازی

برای محاسبه نمودارهای زمانی در الگوریتم پیشنهادی و رقیب، ابتدا جمعیت اولیه کم در نظر گرفته شده است و برای اجرا یک تابع دلخواه بنچمارک انتخاب شده است الگوریتم پیشنهادی در واحد پردازش کارت گرافیک اجرا می شود و زمان اجرا محاسبه و یادداشت می شود. همین تعداد جمعیت اولیه را به الگوریتم های ژنتیک، ذرات و کرم شب تاب غیرموازی تخصیص داده می شود. و زمان اجرا آنها در سی پی یو یادداشت می شود. این فرآیند را ۵۰ بار برای این جمعیت اولیه اجرا شده است و میانگین زمان

¹ Rastrigin

² Eggholder

الگوریتم‌ها محاسبه می‌شود. در مرحله بعد مقدار جمعیت اولیه را افزایش داده و مرحله قبلی مجدد طی می‌شود. این مراحل تا افزایش بیشتر جمعیت و اندازه داده‌ها مرتب انجام می‌شود و نمودار مقایسه‌ای بین الگوریتم پیشنهادی، ژنتیک، ذرات و کرم شب‌تاب استاندارد برحسب زمان - اندازه داده (یا جمعیت اولیه در هر آزمایش) رسم می‌شود این نمودار مربوط به یک تابع بنچمارک است و برای تعدادی از این توابع انجام می‌شود که یکی دیگر از روش‌های تحقیق ما محاسبه شتاب الگوریتم پیشنهادی نسبت به الگوریتم غیرموازی کرم شب‌تاب است.

روش‌هایی جهت بهبود الگوریتم کرم شب‌تاب

یکی از روش‌های اجرای الگوریتم‌های تکاملی با تعداد جمعیت اولیه انبوه و تکرار زیاد، اجرای آن‌ها به شکل موازی است. روش‌های موازی‌سازی امروزه بسیار متنوع هستند. از جمله آن‌ها می‌توان به محاسبات شبکه‌های گرید اشاره کرد. اخیراً امکان پردازش موازی در واحد پردازش کارت گرافیک که دارای صدها و هزاران هسته محاسباتی است، فراهم شده است. واحد پردازش کارت گرافیک در ابتدا برای پردازش تصاویر و عملیات گرافیکی تدارک دیده شده بود. اما اخیراً امکانات موازی‌سازی کودا باعث شده است تا برنامه‌نویسان از پردازنده‌های داخل کارت گرافیک جهت پردازش موازی از الگوریتم‌های خود استفاده نمایند. در این حالت، محاسبات پردازشی به‌جای اینکه روی سی‌پی‌یو اجرا شوند و کل سیستم مرکزی را کند نمایند، بر روی هسته‌های تقریباً بیکار جی‌پی‌یو اجرا می‌شوند. مزایای برنامه‌نویسی کودا بسیار زیاد است؛ از جمله نیاز به خریدن سخت‌افزار بسیار گران‌قیمت نیست، مهم‌تر این که از کارت گرافیک کاربر به‌عنوان یک ابررایانه کوچک در جهت پردازش‌های محاسباتی کاربران استفاده می‌شود و در مصرف انرژی صرفه‌جویی می‌شود.

تغییر پارامترهای الگوریتم کرم شب تاب به شکل پویا

در الگوریتم استاندارد کرم شب تاب، مقدار ϵ_t در همه شرایط یک عدد تصادفی در بازه صفر و یک است. اگر کرم‌های شب تاب، به نقاطی منتقل شوند که مقدار تابع هدف در آنجا از میزان فعلی کمتر باشند، باید این امکان را داشته باشند که بتوانند پرش و جهش بیشتری داشته باشند.

$$x_i^{t+1} = x_i^t + \beta \cdot \exp(-\gamma r_{ij}^2) \cdot (x_j^t - x_i^t) + (\alpha_i + T\alpha_j)\epsilon_t x_i^{t+1} \quad \text{(رابطه ۴)}$$

$$T=0 \quad \text{or} \quad T=1$$

در رابطه (۴) مقدار α_j و α_i دو مقدار ثابت عددی هستند و مقدار T معرف انتقال خوب یا بد خواهد بود. اگر کرم شب تابی حرکتی انجام دهد که نسبت به حرکت قبلی شایستگی آن را کاهش دهد، برای حرکت فعلی $T=1$ را در نظر می‌گیرد و در غیر این صورت مقدار $T=0$ منظور می‌شود. اگر در رابطه (۴) فرض شود که $\alpha_j = \alpha_i$ ، رابطه (۴) ساده‌تر شده و به رابطه (۵) حاصل خواهد شد.

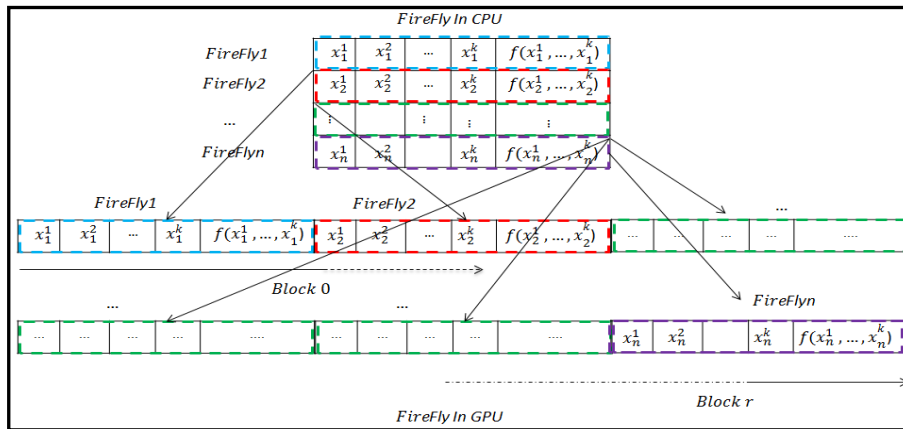
$$x_i^{t+1} = x_i^t + \beta \cdot \exp(-\gamma r_{ij}^2) \cdot (x_j^t - x_i^t) + \alpha_i(1 + T)\epsilon_t \quad \text{(رابطه ۵)}$$

$$T=0 \quad \text{or} \quad T=1$$

اختصاص هسته‌های واحد پردازش کارت گرافیک به کرم‌های شب تاب

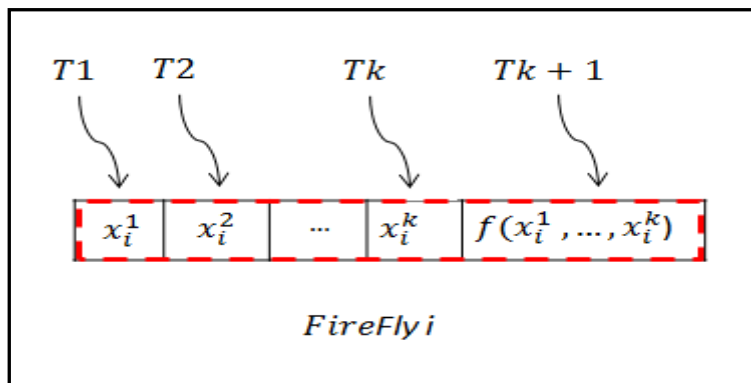
در الگوریتم کرم شب تاب، هر کرم شب تاب در واقع یک بردار به اندازه k می‌باشد که k ، اندازه بعد مسئله است. اگر تعداد کرم‌های شب تاب n باشد، روش مرسوم در برنامه‌نویسی، استفاده از یک ماتریس n در k می‌باشد که هر سطر معرف یک کرم شب تاب و هر ستون معرف یک بعد از مسئله است. در این روش برای پردازش ماتریس در سی‌پی‌یو باید ردیف‌های ماتریس به شکل سریالی پردازش شوند. در این فرآیند اگر تعداد ابعاد مسئله و جمعیت اولیه کرم‌های شب تاب زیاد باشد دارای زمان اجرای بسیار بالایی خواهد بود این مسئله زمانی حادتر خواهد شد که الگوریتم کرم شب تاب دارای تکرار

زیادی باشد. شکل (۳) نحوه قرار گرفتن کرم‌های شب‌تاب در سی‌پی‌یو و جی‌پی‌یو را نشان داده است.



شکل ۳. نحوه قرار گرفتن کرم‌های شب‌تاب در سی‌پی‌یو و جی‌پی‌یو

در این روش، هر نخ یک بعد از هر کرم شب‌تاب را زمان‌بندی می‌کند و مجموعه‌ای از نخ‌ها مسئول زمان‌بندی یک کرم شب‌تاب هستند. هر بلاک در روش پیشنهادی شامل تعدادی کرم شب‌تاب است. هم‌چنین تعداد بلاک‌ها توسط برنامه‌نویس تعیین می‌شود. شکل (۴) نحوه مدیریت یک کرم شب‌تاب توسط تعدادی نخ را نشان می‌دهد.



شکل ۴. مدیریت اجزاء یک کرم شب‌تاب توسط تعدادی نخ

شبه کد روش پیشنهادی

شکل (۵)، شبه کد روش پیشنهادی را نشان می دهد. ورودی های الگوریتم در شبه کد مواردی مانند آرایه ها، پارامترها، ثابت ها، تابع ارزیابی و شایستگی، تعداد تکرار، تعداد جمعیت اولیه کرم های شب تاب و غیره می باشند. در روش پیشنهادی ابتدا تمام متغیرها و آرایه ها در حافظه سراسری سیستم مرکزی مقداردهی اولیه می شوند و مکان اولیه کرم های شب تاب نیز به شکل تصادفی انتخاب می شوند. در مرحله ی بعد آرایه ها را از سی پی یو به جی پی یو انتقال داده می شوند. برای انتقال آرایه ها از سی پی یو به جی پی یو از توابع موجود کتابخانه کودا استفاده می شود. سپس کرنل در یک حلقه تکرار قرار گرفته می شود تا در هر گام حلقه، بهترین کرم شب تاب حاصل شود. بنابراین کرم شب تاب به دست آمده از حلقه ی تکرار را با بهترین کرم شب تاب سراسری مقایسه می کنیم. در صورتی که شایستگی بهترین کرم شب تاب فعلی از کرم شب تاب های سراسری بهتر بود آن را جایگزین کرم شب تاب سراسری می کنیم. در صورتی که کرم های شب تاب موجود در حلقه بخواهند مقدار تابع ارزیابی کمتری حاصل نمایند، $T=1$ می شود و در غیر این صورت $T=0$ خواهد شد. در نهایت بهترین کرم شب تاب سراسری از واحد پردازش کارت گرافیک خارج و در پردازنده مرکزی کپی می شود و نتایج به عنوان خروجی قابل دسترس است.

```

Algorithm GPU-FireFly
Input : arrays , parameter, const , fitness function , iterNum, NumFA, etc
Initialize the States of FAs randomly
Initialize the random arrays
Initialize the parameter of FA algorithms
Transfer data from CPU to GPU
for iter ← 1 to iterNum
do
    Calculate fitness values of all FAs by __device__ fitness
function
    Update the optimization result by
Kernel<<<BlockNum, ThreadNum>>>
    If Current fitness FA > Next fitness FA
    Set T ← 0
    elseif
        Set T ← 1
    Set Best FireFly
    If Best FA > Global FA : Set Best FA instead of Global FA
end for
Transfer device_best_global FA From GPU to CPU
Output : best global FA
    
```

شکل ۵. شبه کد الگوریتم پیشنهاد

ابزار شبیه‌سازی

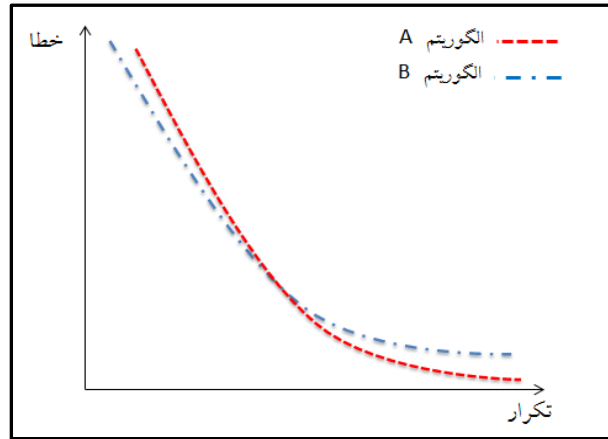
برای شبیه‌سازی و پیاده‌سازی الگوریتم پیشنهادی از سیستم عامل ویندوز ۷^۱ و با حافظه ۶۴ بیتی، کامپایلر ویژوال استادیو ۲۰۱۲^۲ محصول شرکت مایکروسافت و فریم ورک ۵/۵ محصول شرکت انویدیا^۳ استفاده شده است. زبان مورد استفاده در شبیه‌سازی‌ها ویژوال سی پلاس پلاس^۴ است. یکی از مزایای استفاده از زبان ویژوال سی پلاس پلاس دسترسی بالا به حافظه، سرعت بیشتر و وجود مستندات فراوان در مورد این زبان در برنامه‌نویسی کودا است. زبان ویژوال سی پلاس پلاس زبان استاندارد در برنامه‌نویسی موازی کودا است.

یافته‌های پژوهش

برای ارزیابی الگوریتم پیشنهادی از نظر سرعت و دقت به معیارهای استاندارد برای ارزیابی نیاز است. از جمله معیارهای که برای سنجش الگوریتم‌های هوش ازدحامی در واحد پردازش کارت گرافیک بکار می‌رود می‌توان به موارد زیر اشاره کرد:

۱- سرعت هم‌گرایی: معیار سرعت هم‌گرایی نشان می‌دهد که کدام الگوریتم تمایل دارد سریع‌تر به جواب‌های موردنظر همگرا شود. در واقع، هم‌گرایی بدین معنا است که بعد از تعداد مشخصی تکرار، خطای بین جواب واقعی و تقریبی از یک مقدار آستانه کمتر است. در شکل (۶)، یک مقایسه بین هم‌گرایی الگوریتم‌های A و B صورت گرفته است. با توجه به این که، با ادامه تکرار میزان خطای نمودار A کمتر از نمودار B می‌باشد. بنابراین هم‌گرایی الگوریتم A در یافتن جواب‌های مسئله، بهتر از هم‌گرایی نمودار B است.

1. Windows7
2. VisualStudio 2012
3. Nvidia
4. VC++



شکل ۶. مقایسه همگرایی دو الگوریتم مختلف، به عنوان معیار ارزیابی دقت و سرعت

۲- شتاب کارت گرافیک: این معیار به صورت رابطه (۶) تعریف می شود:

$$SpeedUp = \frac{Time_{CPU}}{Time_{GPU}} \quad \text{رابطه ۶}$$

$Time_{GPU}$ و $Time_{CPU}$ در رابطه (۶) به ترتیب زمان اجرای الگوریتم در واحد پردازش مرکزی و واحد پردازش کارت گرافیک است. هر چه مقدار شتاب بیشتر باشد نشان می دهد الگوریتم موازی کارآمدتر بوده است.

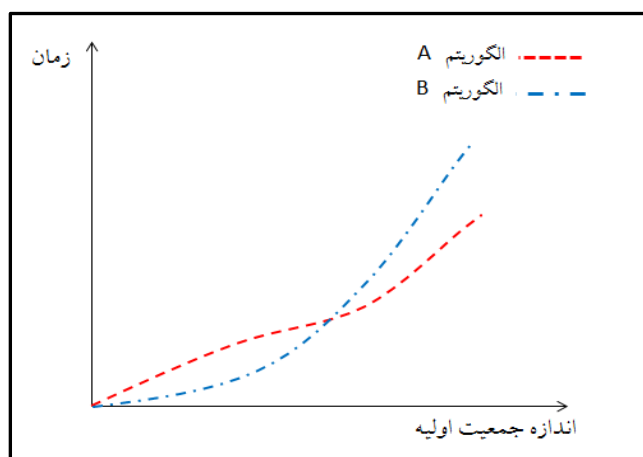
۳- تفاوت کارت های گرافیک: این معیار، برای بیان تفاوت سخت افزار و شتاب

به کار می رود در این ارزیابی الگوریتم را روی چند کارت گرافیک مختلف، از نظر توان پردازشی، اجرا می کنند و تأثیر توان پردازشی سخت افزارهای مختلف را بر روی سرعت و شتاب الگوریتم ها بررسی می کنند.

۴- زمان اجرای الگوریتم A و B: یکی دیگر از معیارهای مقایسه ای، مقایسه

زمان اجرای دو الگوریتم است برای مقایسه زمانی و سرعت دو الگوریتم روش های مختلفی وجود دارد در این تحقیق زمان اجرا بر حسب جمعیت اولیه در نظر گرفته شده است. به طور کلی جمعیت اولیه را به طور معمول توانی از عدد ۲ در نظر گرفته می شود. در شکل (۷)، یک مقایسه زمانی بین دو الگوریتم A و B نشان داده شده است. با توجه به شکل (۷) با افزایش جمعیت اولیه، الگوریتم A در زمان کمتری محاسبات خود را بر روی

این جمعیت انجام می‌دهد. بنابراین، نسبت به الگوریتم B سریع‌تر می‌باشد و زمان اجرای کمتری دارد. به طور کلی الگوریتم‌هایی که محاسبات آن‌ها در زمان کمتری نسبت به سایر الگوریتم‌ها انجام می‌شوند، دارای سرعت بیشتری می‌باشند.



شکل ۷. مقایسه زمان اجرای دو الگوریتم مختلف، به‌عنوان یک معیار ارزیابی سرعت

۵- توابع ارزیابی: توابعی که میزان دقت، سرعت هم‌گرایی الگوریتم‌های تکاملی با آن‌ها بررسی می‌شوند، توابع بنچمارک نامیده می‌شود. توابع بنچمارک توابعی با اشکال پیچیده‌ای هستند. در شکل (۸)، توابع بنچمارک به همراه رابطه ریاضی آن‌ها نشان داده شده است. این توابع در فضای یک‌بعدی تا بعدهای بیشتر را شامل می‌شود. در تحقیق از چهار عدد از این توابع بنچمارک برای ارزیابی استفاده شده است. در شکل (۸) متغیر D به ابعاد مسئله اشاره دارد.

Function Name	Equation	Bounds
Sphere	$\sum_{i=1}^D x_i^2$	$(-10,10)^D$
Rastrigin	$\sum_{i=1}^D [x_i^2 - 10 * \cos(2\pi x_i) + 10]$	$(-5,5)^D$
Griewangk	$\frac{1}{4000} \sum_{i=1}^D x_i^2 - \prod_{i=1}^D \cos(\frac{x_i}{\sqrt{i}}) + 1$	$(-10,10)^D$
Rosenbrock	$\sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$(-10,10)^D$

شکل ۸. توابع بنچمارکی که برای ارزیابی الگوریتم پیشنهادی با سایر الگوریتم‌ها در نظر گرفته شده است

ارزیابی و شبیه‌سازی

در این قسمت به ارزیابی و نتایج حاصل از شبیه‌سازی روش پیشنهادی و مقایسه آن با روش‌های دیگر پرداخته می‌شود.

مقایسه زمان اجرای الگوریتم پیشنهادی با سایر الگوریتم‌های تکاملی

در این قسمت الگوریتم کرم شب تاب پیشنهادی از نظر زمان اجرا با الگوریتم استاندارد کرم شب تاب و الگوریتم ذرات مقایسه شده است. برای مقایسه، اول از تابع اسپیر^۱ استفاده شده است. در این آزمایش از ۵۰ بار تکرار الگوریتم برای میانگین‌گیری استفاده شده است. تعداد تکرار الگوریتم پیشنهادی و کرم شب تاب استاندارد، ۱۰۰ در نظر گرفته شده است. مشخصه کارت گرافیک برای تمام شبیه‌سازی‌ها GeForce 710M است. در این شبیه‌سازی از تغییر جمعیت اولیه به عنوان یک پارامتر متغیر استفاده شده است. شبیه‌سازی و پیاده‌سازی‌ها روی این تابع بنچمارک با $D=3$ یا به عبارتی فضای سه‌بعدی در نظر گرفته شده است. در جدول (۱) الگوریتم پیشنهادی که در واحد پردازش کارت گرافیک اجرا می‌شود با الگوریتم کرم شب تاب استاندارد که در پردازنده مرکزی اجرا می‌شود مقایسه شده است. در این جدول اندازه جمعیت کرم شب تاب‌ها از ۱۶ تا ۱۰۲۴

¹ Sphere

تغییر کرده است. همچنین تأثیر اندازه جمعیت اولیه کرم‌های شب‌تاب روی زمان اجرای الگوریتم پیشنهادی در واحد پردازش کارت گرافیک و الگوریتم کرم شب‌تاب استاندارد بررسی شده است. اطلاعات جدول (۱) نشان می‌دهد اگر تعداد جمعیت اولیه کم باشد زمان اجرا در سی‌پی‌یو بهتر از زمان جی‌پی‌یو است. و اگر اندازه کرم‌های شب‌تاب بیشتر از ۱۲۸ باشد شتاب کارت گرافیک بیشتر از یک می‌شود.

جدول ۱. مقایسه الگوریتم پیشنهادی با الگوریتم کرم شب‌تاب استاندارد، با داده‌های مختلف و تابع اسپیر.

سرعت	زمان جی پی یو کرم شب‌تاب (میلی ثانیه)	زمان سی پی یو کرم شب‌تاب (میلی ثانیه)	کرم‌های شب‌تاب
۰/۰۴۰	۰/۷۱۳۰	۰/۰۳۲۰	۱۶
۰/۱۸۰	۰/۸۰۲۰	۰/۱۴۴۰	۳۲
۰/۶۳۰	۰/۸۹۳۰	۰/۵۶۰۰	۶۴
۲/۰۱۰	۱/۰۷۹۰	۲/۱۶۸۰	۱۲۸
۶/۰۷۰	۱/۴۲۷۰	۸/۶۶۸۰	۲۵۶
۱۵/۶۲	۲/۱۴۶۰	۳۳/۵۱۰	۵۱۲
۲۰/۹۳	۶/۳۹۲۰	۱۳۳/۷۲	۱۰۲۴

در شبیه‌سازی دوم از تابع بنچمارک راستریجن، در شبیه‌سازی سوم از تابع بنچمارک گریوانک^۱ و در شبیه‌سازی چهارم از تابع بنچمارک رزنبرک استفاده شده است. در جداول (۲)، (۳) و (۴) نتایج این شبیه‌سازی‌ها نشان داده شده است.

جدول ۲. مقایسه الگوریتم پیشنهادی با الگوریتم کرم شب‌تاب استاندارد، با داده‌های مختلف و تابع راستریجن

سرعت	زمان جی پی یو کرم شب‌تاب (میلی ثانیه)	زمان سی پی یو کرم شب‌تاب (میلی ثانیه)	کرم‌های شب‌تاب
۰/۰۵۰	۰/۷۴۱۰	۰/۰۳۹۰	۱۶
۰/۱۸۰	۰/۹۲۴۰	۰/۱۶۲۰	۳۲
۰/۵۷۰	۱/۱۱۲۰	۰/۶۳۸۰	۶۴

¹ Griewangk

سرعت	زمان جی پی یو کرم شب تاب (میلی ثانیه)	زمان سی پی یو کرم شب تاب (میلی ثانیه)	کرم های شب تاب
۱/۷۱۰	۱/۵۰۹۰	۲/۵۸۲۰	۱۲۸
۴/۳۹۰	۲/۳۱۲۰	۱۰/۱۴۸	۲۵۶
۱۰/۶۳	۳/۷۷۶۰	۴۰/۱۴۹	۵۱۲
۱۲/۴۳	۱۲/۸۱۸	۱۵۸/۱۴	۱۰۲۴

جدول ۳. مقایسه الگوریتم پیشنهادی با الگوریتم کرم شب تاب استاندارد، با داده های مختلف و تابع گریوانک

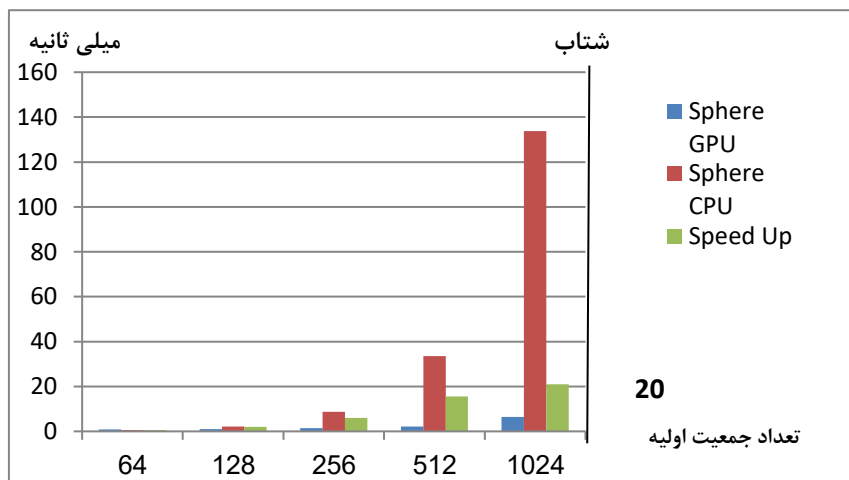
سرعت	زمان جی پی یو کرم شب تاب (میلی ثانیه)	زمان سی پی یو کرم شب تاب (میلی ثانیه)	کرم های شب تاب
۰/۰۶۰	۰/۷۳۸۰	۰/۰۴۳۰	۱۶
۰/۱۹۰	۰/۹۵۰۰	۰/۱۷۸۰	۳۲
۰/۶۰۰	۱/۱۴۹۰	۰/۶۹۴۰	۶۴
۱/۸۷۰	۱/۵۴۰۰	۲/۸۷۷۰	۱۲۸
۴/۵۱۰	۲/۳۵۴۰	۱۰/۶۲۸	۲۵۶
۱۰/۵۷	۳/۹۸۱۰	۴۲/۰۷۸	۵۱۲
۱۲/۲۳	۱۳/۷۳۲	۱۶۷/۹۴	۱۰۲۴

جدول ۴. مقایسه الگوریتم پیشنهادی با الگوریتم کرم شب تاب استاندارد، با داده های مختلف و تابع رزنبوک

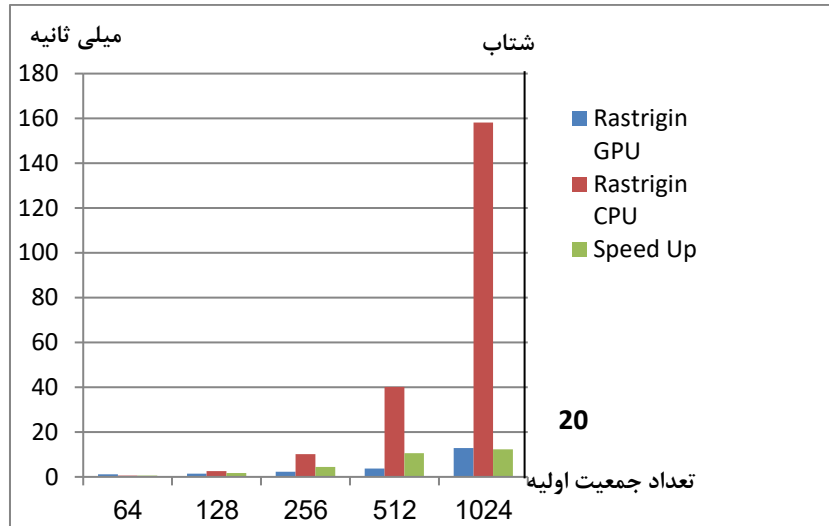
سرعت	زمان جی پی یو کرم شب تاب (میلی ثانیه)	زمان سی پی یو کرم شب تاب (میلی ثانیه)	کرم های شب تاب
۰/۱۰۰	۰/۷۰۷۰	۰/۰۶۸۰	۱۶
۰/۲۰۰	۰/۷۹۹۰	۰/۱۶۱۰	۳۲
۰/۵۷۰	۰/۹۸۶۰	۰/۵۶۱۰	۶۴
۱/۹۸۰	۱/۰۷۲۰	۲/۱۲۷۰	۱۲۸
۶/۴۴۰	۱/۳۵۳۰	۸/۷۱۷۰	۲۵۶
۱۶/۷۳	۲/۰۲۲۰	۳۳/۸۳۴	۵۱۲
۲۲/۷۶	۵/۹۲۷۰	۱۳۴/۸۹	۱۰۲۴

شکل های (۱۰)، (۱۱)، (۱۲) و (۱۳) میزان زمان اجرای الگوریتم پیشنهادی در واحد پردازش و الگوریتم استاندارد کرم شب تاب، به ترتیب با توابع بنچمارک اسپیر، راستریجن،

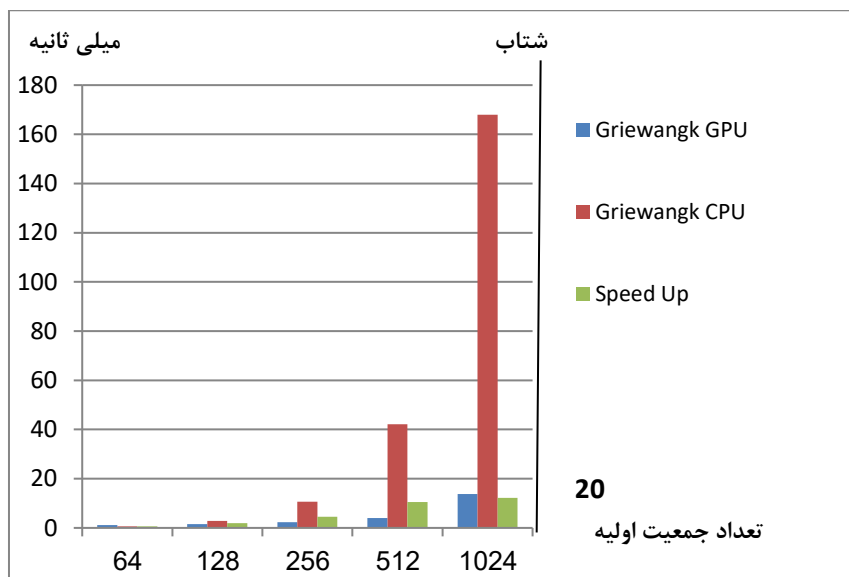
گریوانک و رزبرک بررسی شده است. در این نمودارها محور افقی اندازه جمعیت اولیه و محور عمودی دارای دو معیار زمان بر حسب میلی ثانیه و شتاب کارت گرافیک است. بررسی دقیق نمودارها نشان می دهد که اگر اندازه جمعیت اولیه کم باشد در همه موارد زمان اجرا در سی پی یو کمتر از زمان اجرا در جی پی یو خواهد بود. اما اگر تعداد جمعیت اولیه افزایش یابد زمان اجرا در جی پی یو کمتر از سی پی یو خواهد بود. شتاب کارت گرافیک در تمام نمودارها روندی صعودی دارد که با افزایش تعداد جمعیت اولیه کرم های شب تاب، اندازه شتاب بیشتر می شود. این نتایج نشان می دهد که افزایش اندازه داده ها تأثیر بیشتری روی شتاب کارت گرافیک دارد. در مواردی که برای به دست آوردن جواب های دقیق نیاز است تعداد جمعیت اولیه کرم های شب تاب و تکرار الگوریتم زیاد باشد. روش پیشنهادی دارای سرعتی تا ۱۶۰ برابر سی پی یو است. اما در حالی که تعداد جمعیت اولیه و تکرار کم باشد استفاده از سی پی یو توصیه می شود.



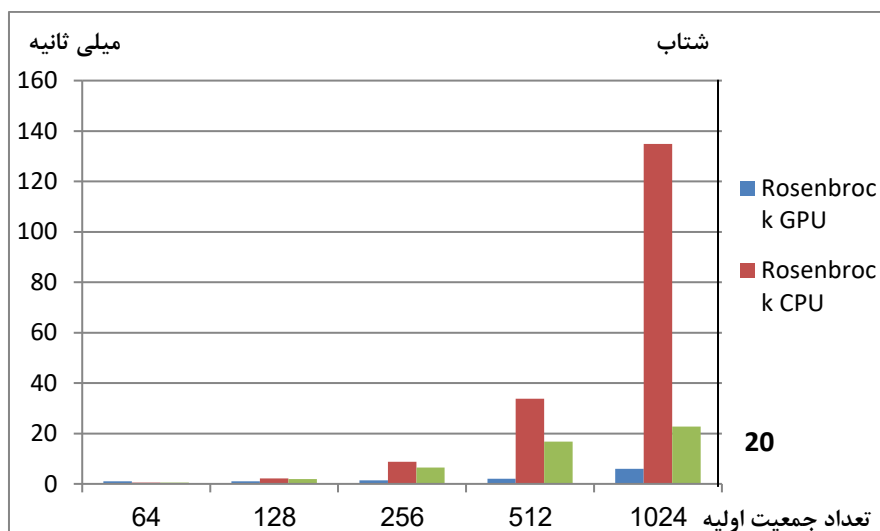
شکل ۱۰. مقایسه زمان و شتاب اجرای الگوریتم پیشنهادی، الگوریتم کرم شب تاب استاندارد با تابع ارزیابی اسپیر



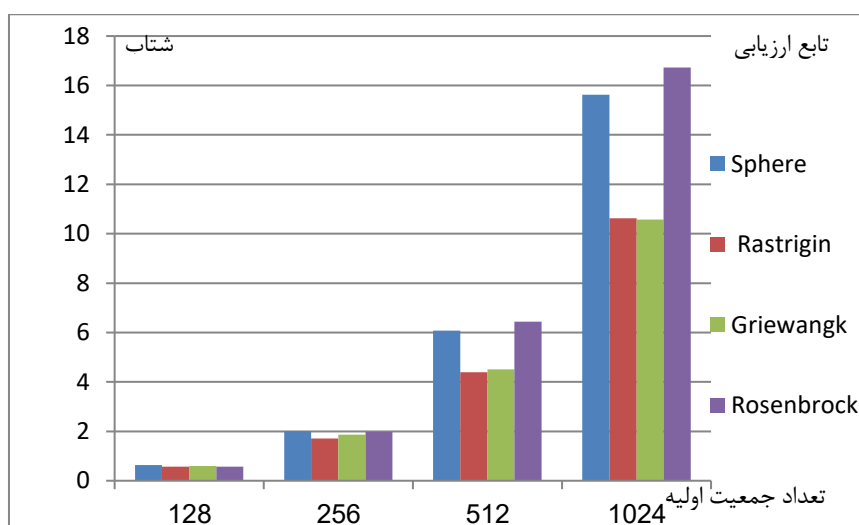
شکل ۱۱. مقایسه زمان و شتاب اجرای الگوریتم پیشنهادی، الگوریتم کرم شب تاب استاندارد با تابع ارزیابی راستریجن



شکل ۱۲. مقایسه زمان و شتاب اجرای الگوریتم پیشنهادی، الگوریتم کرم شب تاب استاندارد با تابع ارزیابی گریوانگ



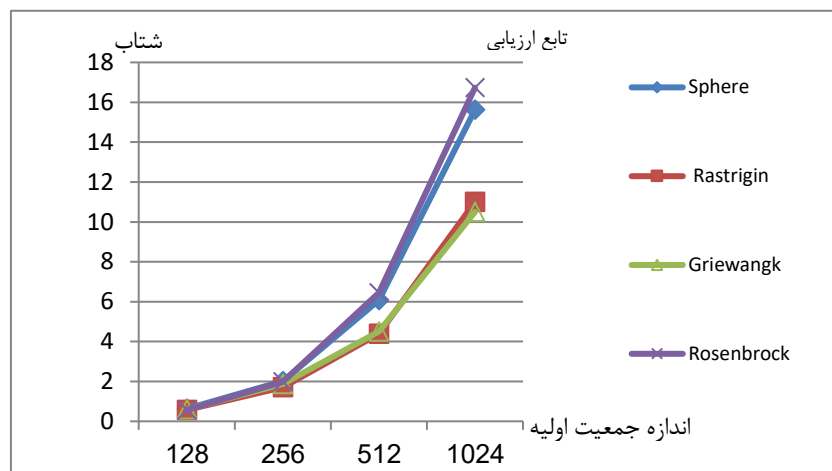
شکل ۱۳. مقایسه زمان و شتاب اجرای الگوریتم پیشنهادی، الگوریتم کرم شتاب استاندارد با تابع ارزیابی رزنبروک



شکل ۱۴. مقایسه (میله‌ای) شتاب اجرای الگوریتم با استفاده از تابع ارزیابی اسپیر، راستریجن، گریوانک، رزنبروک

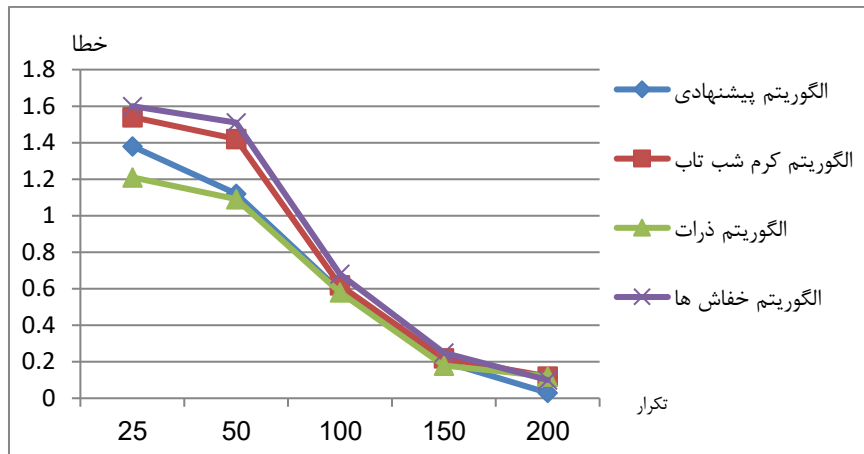
در نمودار (۱۳) روندی متفاوت از نمودارهای میله‌ای برای بررسی شتاب کارت گرافیک نشان داده شده است. در این نمودار مشخص است شتاب کارت گرافیک برای

توابع بنچمارک اسپیر و رزنبرک بیشتر از دو توابع بنچمارک راستریجن و گریوانک است. این نتیجه گیری قابل انتظار بود زیرا شکل توابع بنچمارک اسپیر و رزنبرک ساده تر از توابع بنچمارک راستریجن و گریوانک است به طور کلی شکل تابع بنچمارک معرف پیچیدگی بالای مسئله بهینه سازی مرتبط است.



شکل ۱۵. مقایسه (نموداری) شتاب اجرای الگوریتم با استفاده از تابع ارزیابی اسپیر، راستریجن، گریوانک و رزنبرک

یکی از ارزیابی‌های موجود در این مقاله، بررسی دقت و سرعت همگرایی به جواب‌های بهینه است. در آخرین شبیه‌سازی، میزان میانگین خطا از جواب واقعی را در توابع بنچمارک اسپیر، راستریجن، گریوانک و رزنبرک در مورد الگوریتم پیشنهادی، الگوریتم کرم شب تاب استاندارد، الگوریتم ذرات و الگوریتم خفاش‌ها با ۱۰ بار آزمایش محاسبه و نتایج حاصل از آن در شکل (۱۶) نشان داده شده است. در نهایت، دقت الگوریتم پیشنهادی از الگوریتم استاندارد کرم شب تاب، الگوریتم ذرات و الگوریتم خفاش بیشتر شده است.



شکل ۱۶. مقایسه دقت الگوریتم پیشنهادی، کرم شب تاب، ذرات و الگوریتم خفاش

نتیجه گیری و پیشنهادها

در این مقاله از آزمایش‌های مختلفی برای ارزیابی الگوریتم پیشنهادی استفاده شد. یکی از معیارهای ارزیابی الگوریتم پیشنهادی مقایسه زمانی آن با نسخه استاندارد الگوریتم کرم شب تاب بود. برای مقایسه از توابع بنچمارک اسپیر، راستریجن، گریوانک و رزنبرک و جمعیت‌های اولیه متغیر استفاده شد. در این آزمایش‌ها از ۵۰ بار تکرار الگوریتم برای میانگین زمان اجرا استفاده شد و تعداد تکرار الگوریتم پیشنهادی و کرم شب تاب استاندارد ۱۰۰، در نظر گرفته شد. در تمام پیاده‌سازی‌ها از کارت گرافیکی به مشخصه GeForce 710M استفاده شد. نتایج نشان می‌دهد هرچه اندازه جمعیت اولیه کرم‌های شب تاب بیشتر باشد، اختلاف زمان اجرا بین سی‌پی‌یو و جی‌پی‌یو بیشتر و زمان اجرا در جی‌پی‌یو به مراتب کمتر از سی‌پی‌یو می‌شود و به همان نسبت شتاب کارت گرافیک برای الگوریتم پیشنهادی نیز بیشتر می‌شود از طرفی اگر اندازه داده‌ها یا جمعیت اولیه کم باشد، زمان اجرا در جی‌پی‌یو بیشتر از سی‌پی‌یو می‌شود و به همان نسبت شتاب کارت گرافیک برای الگوریتم پیشنهادی کمتر می‌گردد. نتایج نشان می‌دهد موازی‌سازی با استفاده واحد پردازش کارت گرافیک، زمانی توجیه پذیر است که داده‌ها به اندازه‌ی کافی بزرگ باشند. داده‌ها در شبیه‌سازی به عنوان جمعیت اولیه بودند. علت این امر این گونه توجیه شد که

سربار زمانی از داده‌های کم در انتقال از سی‌پی‌یو به جی‌پی‌یو در مقابل زمان اجرا الگوریتم پیشنهادی قابل توجه بود و هر چه داده‌ها بزرگ‌تر باشند این سربار در مقابل زمان اجرا در جی‌پی‌یو یونیز ناچیز خواهد بود. از دیگر نتایج این پژوهش، رابطه پیچیدگی مسئله بهینه‌سازی و شتاب الگوریتم پیشنهادی بود. نتایج نشان داد که شتاب کارت گرافیک برای توابع بنچمارک اسپیر و رزبرک بیشتر از دو تابع بنچمارک راستریجن و گریوانک است. این نتیجه‌گیری قابل انتظار بود، زیرا شکل توابع بنچمارک اسپیر و رزبرک ساده‌تر از توابع بنچمارک راستریجن و گریوانک بود و شتاب کارت گرافیک برای توابع بنچمارک راستریجن و گریوانک تقریباً یکسان بود. به طور کلی شکل توابع بنچمارک، معرف پیچیدگی بالای مسئله بهینه‌سازی مرتبط بود و هر چه مسئله ساده‌تر باشد شتاب کارت گرافیک برای الگوریتم پیشنهادی بیشتر خواهد شد. به طور کلی اگر اندازه داده‌ها به قدر کافی بزرگ باشد شتاب الگوریتم پیشنهادی همیشه از روش استاندارد بیشتر است. به عبارتی، زمان اجرای الگوریتم پیشنهادی از روش الگوریتم استاندارد کرم شب‌تاب کم‌تر شد. برای مقایسه دقت در روش پیشنهادی، این روش را با سه الگوریتم کرم شب‌تاب استاندارد، الگوریتم ذرات و الگوریتم خفاش‌ها مقایسه گردید. برای مقایسه دقت و سرعت همگرایی از میانگین نتایج حاصل از ۱۰ آزمایش با چهار تابع بنچمارک اسپیر، راستریجن، گریوانک و رزبرک استفاده شده است. در این آزمایش بعد از حدود ۲۰۰ تکرار به بعد، دقت الگوریتم پیشنهادی از الگوریتم کرم شب‌تاب استاندارد، الگوریتم ذرات و الگوریتم خفاش بیشتر شد.

در این مقاله، نشان داده شد که الگوریتم پیشنهادی نسبت به الگوریتم استاندارد کرم شب‌تاب دارای سرعت اجرای بالاتری است. همچنین سرعت و شتاب با افزایش داده‌ها و جمعیت اولیه بیشتر می‌شود. در پژوهش آینده قصد داریم به جای استفاده از یک کارت گرافیک از چند کارت گرافیک که به نام سخت‌افزار مالتی جی‌پی‌یو شناخته می‌شوند، استفاده نماییم زیرا سخت‌افزارها به جای یک واحد پردازش کارت گرافیک، از چند کارت گرافیک و چند واحد پردازش کارت گرافیک استفاده می‌کند. این مجموعه از واحدهای پردازش کارت گرافیک، توان محاسباتی و شتاب بسیار بالایی را به الگوریتم‌های مختلف می‌دهند.

منابع

- A.Shukla, R. a. (2010). Swarm Intelligence. *in Towards Hybrid and Adaptive Computing*, 187-207.
- A.V Husselmann, a. K. (2012). Parallel Parametric Optimisation with Firefly Algorithms on . *Graphical Processing Units*, 1-7.
- j.d.Owens, M. H. (2008). GPU Computing. *Proceedings of the IEEE*, 879-899.
- j.Kennedy, R. E. (1995). Particle swarm optimization. *in Neural Networks, Perth*, 1942-1948.
- G.Agarwal, S. (2013). Evaluation performance study of Firefly algorithm, particle swarm optimization and artificial bee colony algorithm for non-linear mathematical optimization functions. 1-8.
- G.Zhua, S. (2010). Gbest-guided artificial bee colony algorithm for numerical function optimization. pp. 3166-3173.
- M.Imran, R. H. (2013). An Overview of Particle Swarm Optimization Variants. *Procedia Engineering*, 491-496.
- p.Pospichal, j. J. (2010). Parallel Genetic Algorithm on the CUDA Architecture. *Applications of Evolutionary Computation*, 442-451.
- R. Poli, J. K. (2007). Particle swarm optimization. *Swarm Intelligence*, 33-57.
- S.X.Wu, a. W. (2010). The use of computational intelligence in intrusion detection systems A review. 1-35.
- S-C. Chu, H.-C. H.-S. (2011). Overview of Algorithms for Swarm Intelligence,". *in Computational Collective Intelligence*, 28-41.
- x.s.yang, x. H. (2013). Firefly algorithm: recent advances and applications. *International Journal of Swarm Intelligence*. pp. 35-50.
- x.Yang. (2010). *Engineering optimization: an introduction with metaheuristic applications*. John Wiley & Sons.
- X.S.Yang. (2010). A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization*. 65-74.
- X.Yang. (2013). *Bat Algorithm: Literature Review and Applications*. 141-149.
- Z.Yao, a. J. (2011). A quantitative performance analysis model for GPU architectures. *In High Performance Computer Architecture (HPCA)*, 382-393.

شناسه دیجیتال (DOI): 10.22091/jemsc.2018.753.1033

استناد به این مقاله:

سلیمانی، فریبا؛ صالح پور، نرگس؛ نظری فرخی، محمد. (۱۳۹۷). «ارائه یک روش جدید برای افزایش هوشمندی و سرعت الگوریتم کرم شب تاب». مدیریت مهندسی و رایانش نرم، ۷(۱)، ۳۴-۹.