ISSN: 2745-3673
http://www.Rcsj.ir

مجله‌ی علمی پژوهش در علوم رایانه (سال دوم)
شماره ۴ / زمستان ۱۳۹۵ / ص ۵۷- ۵۰

الگوریتم بهینه‌سازی کد برنامه ها جهت کاهش توان مصرفی با ترکیب الگوریتم های ژنتیک و K-means

سجاد بحرکاظمی[۱]، اسدالله شاه بهرامی [۲]
[۱] کارشناسی ارشد مهندسی کامپیوتر دانشگاه گیلان
[۲] دانشیار گروه مهندسی کامپیوتر دانشگاه گیلان

نام و نشانی ایمیل نویسنده مسئول:
سجاد بحر کاظمی
sbahrekazemi@ymail.com

چکیده

توان مصرفی در سیستم‌های کامپیوتری موضوعی مهم است. به دلیل کاهش ذخایر انرژی، اهمیت ذخیره توان مصرفی روزبه‌روز بیشتر احساس می‌شود. امروزه این موضوع در علوم کامپیوتر بخصوص برنامه‌های کامپیوتری بسیار بارزتر و آشکارتر شده است. توان مصرفی یک چالش مهم در برنامه‌هاست بنابراین الگوریتم‌های بهینه‌سازی کد با بهینه نمودن کد برنامه‌ها باعث می‌شوند که پیمایش برنامه‌ها زودتر انجام گیرد و پردازش برنامه‌ها بهبود یابد. بنابراین هدف الگوریتم‌های بهینه‌سازی کد، بهبود عملکرد برنامه‌ها در جهت کاهش توان مصرفی آن‌هاست. امروزه تلاش‌های زیادی در جهت کاهش توان مصرفی انجام می‌گیرد. در این مقاله، هدف کاهش مصرف توان برنامه‌ها است. پس توان مصرفی به چالش مهمی در سیستم‌های کامپیوتری تبدیل‌شده است. در اینجا الگوریتم‌های ژنتیک و K-means ترکیب شده و الگوریتمی به‌منظور بهینه‌سازی کد پیشنهاد گردیده است که با استفاده از تکنیک بهینه‌سازی حلقه Loop interchange می‌تواند توان مصرفی برنامه را کاهش دهد. نتایج نشان دهنده کاهش مصرف توان در اجرای اندازه‌های مختلف ماتریس است.

واژگان کلیدی: الگوریتم‌های بهینه سازی کد- کاهش توان مصرفی- الگوریتم ژنتیک- الگوریتم K-means

مجلـه‌ی علمی پژوهش در علوم رایانه ، شماره ۴ ،زمستان ۱۳۹۵، ص ۵۷ – ۵۰

ISSN: 2745-3673
http://www.Rcsj.ir

مقدمه

درگذشته مسئله توان مصرفی بیشتر در سخت‌افزارها مورد بحث و بررسی بود، اما امروزه در نرم‌افزارها بیشتر کارایی دارد و قابل‌قبول‌تر و بهتر است[۱]. بنابراین توان مصرفی در برنامه‌های کامپیوتری نقش بسزایی دارد و باعث می‌شود قسمت‌هایی از برنامه‌ها که توان بیشتری مصرف می‌کنند، بیشتر مورد ارزیابی قرار گیرند[۱۸]. به‌عنوان نمونه در زمان اجرای برنامه در CPU، هرچقدر برنامه طولانی‌تر باشد، CPU بیشتر درگیر می‌شود و انرژی بیشتری مصرف می‌کند، پس نقش توان مصرفی پررنگ‌تر می‌شود.

همچنین هدف از بهینه‌سازی، کاهش حجم و توان و انرژی مصرفی در برنامه‌ها است[۲]. اکثر مسائل مهم بهینه‌سازی در کامپایلرها از نوع غیرقطعی هستند. پژوهشگران کامپایلر در تلاشند تا راه‌های مؤثری را برای کاهش توان مصرفی در برنامه‌های کامپیوتری بیابند. استفاده از تکنیک‌های بهینه‌سازی کامپایلر می‌تواند در حل این مشکل مفید باشد.

هر چند تکنیک‌ها، روش‌ها و الگوریتم‌های متعددی برای کاهش توان مصرفی ارائه شده است، اما باید تکنیک‌های هوشمندی را بکار ببریم که راه‌حل تقریبی مناسب را برای برنامه‌های کامپیوتری پیدا کنند. با استفاده از روش‌های بهینه‌سازی کامپایلر پیشرفت هایی در برنامه‌های کامپیوتری به‌دست‌آمده است. برای رسیدن به نتیجه مطلوب، انتخاب روش‌های مناسب از اهمیت بسیاری برخوردار است.

۱- بیان مسئله

در سال‌های اخیر توجه عمده بر بهبود عملکرد و تکنیک‌ها بی‌توجه به توان بود. در کنفرانس‌ها و همایش‌ها موضوع توان به‌عنوان یک مسئله حاشیه‌ای تلقی می‌شد ولی به مرور به دلیل اهمیت ذخیره‌سازی مصرف توان و انرژی در سیستم‌های کامپیوتری و جلوگیری از هدر رفتن توان و انرژی مصرف‌شده، مسئله توان مورد اهمیت قرار گرفت و در مجموعه‌ای از معماری‌ها و سیستم‌های کامپیوتری نمایان گردید. امروزه توان را نمی‌توان در هیچ معماری نادیده گرفت[۱۵]. همچنین بهینه‌سازی کد از بخش‌های مهم برنامه‌های کامپیوتری است که هدف آن بهبود عملکرد است و در سیستم‌های نرم‌افزاری کاربرد فراوانی دارد و به همراه خود، کاهش توان مصرفی را به دنبال دارد. این موضوع در برنامه‌های کامپیوتری مورد اهمیت قرار دارد، به‌نحوی‌که بین سال‌های ۲۰۱۱ و ۲۰۱۲ در امریکا، توان مصرف شده در کامپیوترها ۱۰ درصد شد[۳و۴]. در اواخر دهه ۱۹۹۰، توان در سطح جهانی توسط طراحان سیستم به محدودیت مرتبه اول در طراحی سیستم‌ها شناخته شد. پس توان را نمی‌توان بی‌اهمیت دانست[۱۵]. بنابراین روش‌ها و شیوه‌هایی که برای کاهش توان مصرفی است، باید بررسی و ارزیابی قرار گیرد. در این میان بهینه‌سازی کد برنامه‌ها، یکی از این راه‌هاست که باید به‌طورجدی موردمطالعه قرار گیرد.
صفحه اول مقاله باید کاملاً مشابه صفحه اول این مقاله باشد. در صفحه اول از نوشتن سایر موارد خودداری کنید. همچنین تمام موارد صفحه اول باید در همان صفحه آماده و نوشته شوند.

۲- روش بکار گرفته شده

در این مقاله الگوریتم‌های ژنتیک و K-means ترکیب شده و الگوریتمی به‌منظور بهینه‌سازی کد پیشنهاد گردیده است که با استفاده از تکنیک بهینه سازی حلقه Loop interchange می‌تواند توان مصرفی برنامه مورد نظر(ماتریس n*n) را در اجرای اندازه‌های مختلف ماتریس کاهش دهد. دلیل انتخاب این برنامه، آن است که حلقه ها بهترین گزینه برای بهینه سازی هستند، چرا که بیشتر وقت برنامه‌ها در حلقه‌ها صرف می‌شود [۱۶]. بنابراین هدف اصلی این پژوهش، کاهش مصرف توان است.

```
for (x = 1; x < n; x ++)
  for (y = 1; y < n; y ++)
    a[x][y] = a[x-1][y] + a[x][y-1]
```

۲-۱- الگوریتم ژنتیک

الگوریتم ژنتیک از الگوریتم های تکاملی است و از تکنیک‌های وراثت و جهش استفاده می‌کنند. این الگوریتم بخشی از تحولات در رشته کامپیوتر است که توسط جان هلند در سال ۱۹۷۰ معرفی شد و تکنیک جستجویی در علوم کامپیوتر برای پیدا کردن راه‌حل تقریبی در بهینه‌سازی و مسائل جستجو است[۱۲ و۱۰]. این الگوریتم جایگاه معینی در حل مسائلی که فضای جستجوی آن‌ها بزرگ بوده و تخمین یک جواب نیز از هزینه بالایی برخوردار است را دارد[۱۱]. با نگاه دقیق به پروسه تکامل، یعنی پروسه ای که طبیعت برای حل مسائل خود از آن استفاده می‌کند، می‌توان به دستاوردهای جالب و قابل اجرا دست یافت.

ISSN: 2745-3673
http://www.Rcsj.ir

۲-۱-۱- مراحل کلی الگوریتم ژنتیک

مراحل الگوریتم ژنتیک بصورت زیر می‌باشد:

۱) تولید جمعیت اولیه: بصورت تصادفی N کروموزوم جهت تولید جمعیت اولیه P(0) ایجاد می‌شوند. مقدار حداکثر نسل‌ها (T) و شمارنده نسل برابر صفر (t=0) مقداردهی می‌شود، یعنی راه حل‌های مناسب برای مسئله تولید می‌شود.

۲) ارزیابی جمعیت: مقدار برازندگی هر کروموزوم در جمعیت P(t) توسط تابع تناسب محاسبه می‌شود.

۳) باز تولید یا انتخاب: این عملگر بر روی جمعیت P(t) اعمال می‌شود. (انتخاب فرزندان جدید و قرار دادن آن ها در جمعیت جدید)

۴) بازترکیب: فرزندان جدید را از ترکیب والدین تشکیل می‌دهد.

۵) جهش: بر روی جمعیت اعمال تا جمعیت جدید P(t+1) برای نسل t+1 تولید شود.

۶) تست پایان: زمانی که به یکی از شرط‌های پایانی مثل t=T رسیده باشیم، الگوریتم پایان می‌یابد و یک نمونه که بیشترین مقدار ارزش را بدست آورده، به عنوان راه حل بهینه ارائه می‌شود، در غیر این صورت t=t+1 قرار داده شده و به مرحله ارزیابی جمعیت بر می‌گردیم[۱۰].

۲-۲- الگوریتم K-means

این الگوریتم یکی از اصلی‌ترین الگوریتم‌های داده‌کاوی است که در سال ۱۹۷۶ توسط Mc Queen ارائه گردید[۱۹]. در این الگوریتم، داده‌ها بر اساس K ویژگی ذکر شده برای آنها به خوشه تقسیم می‌شوند. تمام اعضای داخل هر خوشه، بیشترین تشابه را با هم و بیشترین تفاوت را با اعضای خوشه‌های دیگر دارند. در این الگوریتم تعداد خوشه‌ها از قبل مشخص است و در مجموعه داده‌های بزرگ و زمانی که تعداد ویژگی داده‌ها زیاد باشد، استفاده می‌شوند.

۲-۲-۱- مراحل کلی الگوریتم K-means

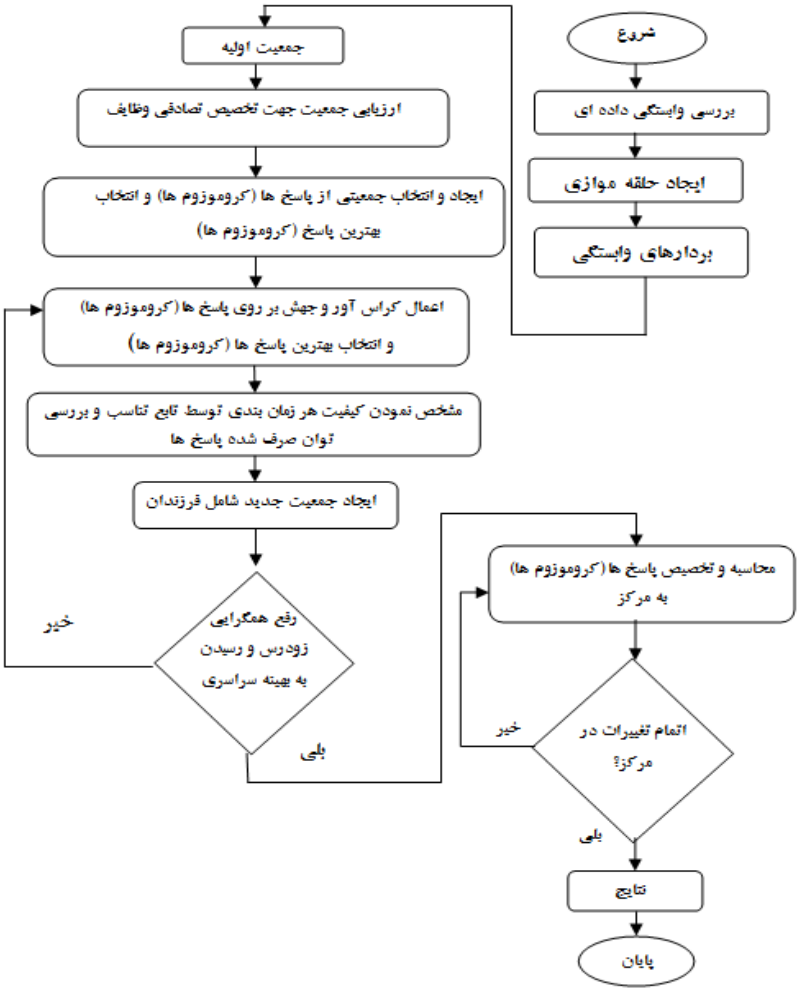
مراحل الگوریتم K-means بصورت زیر می‌باشد:

ابتدا برای نقاط مراکز خوشه‌ها K نقطه انتخاب می‌شوند، به خوشه‌ای که مرکز آن خوشه کمترین فاصله تا هر داده را داراست، نمونه داده نسبت داده می‌شود. پس از دادن تمام داده‌ها به یکی از خوشه‌ها، برای هر خوشه یک نقطه جدید به عنوان مرکز محاسبه انتخاب می‌شود. مرحله دوم و سوم تکرار می‌شوند تا وقتی که دیگر هیچ تغییری در مراکز خوشه‌ها نباشد[۱۹ و ۱۴].

۲-۳- الگوریتم پیشنهادی

بکارگیری الگوریتم‌های تکاملی در الگوریتم‌های داده‌کاوی موجب می‌شود مشکلات بسیاری از این الگوریتم‌ها حل شود. این الگوریتم، ترکیبی از دو الگوریتم ژنتیک و K-means است. ترکیب این دو الگوریتم منجر به حل مشکل همگرایی زودرس الگوریتم K-means می‌شود و از بهینه محلی رها و به بهینه سراسری می‌رسد. هدف این است که در کمترین توان صرف شده، کل عناصر ماتریس در اندازه‌های مختلف که در واقع کل برنامه مذکور(ماتریس n*n) را نشان می‌دهد، اجرا گردد. در این الگوریتم ابتدا بصورت تصادفی وظایف را به پردازنده‌ها تخصیص می‌دهیم. سپس جمعیتی از این پاسخ‌ها که بصورت تصادفی انتخاب شده‌اند، ایجاد و بهترین پاسخ‌ها را انتخاب می‌نماییم. بنابراین یک تابع هدف وجود دارد که کیفیت هر توان مصرفی را مشخص می‌نماید و در نتیجه تعدادی فرزند برای ایجاد نسل جدید انتخاب می‌شوند و با بکارگیری عملگر ترکیب، تکثیر و جهش نسل جدید تولید می‌گردد. این عمل آنقدر تکرار می‌شود تا اینکه در یک مسیر، بهترین راه حل (کروموزوم) ارائه شود. هدف الگوریتم پیشنهادی به حداقل رساندن توان مصرفی در مقداردهی عناصر ماتریس است که توسط حلقه تو در تو انجام می‌گیرد و شاخص کیفیت برای هر کروموزوم، زمان کامل شدن یا توان صرف شده کمتر برای مقداردهی عناصر ماتریس است که بوسیله کروموزوم مشخص شده است. برای اینکه این توان مصرف شده را کاهش دهیم، سعی می‌کنیم وظایفی را به یک پردازنده تخصیص دهیم که به یکدیگر وابسته‌اند.

مجلـه‌ی علمی پژوهش در علوم رایانه ، شماره ۴ ،زمستان ۱۳۹۵، ص ۵۷ – ۵۰
ISSN: 2745-3673
http://www.Rcsj.ir



شکل۱- فلوچارت کلی الگوریتم پیشنهادی

۴-۲- روش SL

یادگیری با نظارت یکی از روش‌های کلی در یادگیری ماشین است. در این روش یک ناظر وجود دارد که بهترین کار در هر حالت را می‌داند. در این روش، سیستم سعی می‌کند، تابعی از ورودی را به خروجی انتقال دهد. یادگیری با نظارت، نیازمند تعدادی داده ورودی جهت آموزش سیستم است و ناظری وجود دارد که در هنگام آموزش، اطلاعاتی علاوه بر داده‌های آموزش در اختیار فراگیر قرار می‌دهد[۱۶].

شکل۲- شکل کلی روش یادگیری با نظارت

مجلـه‌ی علمی پژوهش در علوم رایانه ، شماره ۴ ،زمستان ۱۳۹۵، ص ۵۷ – ۵۰

ISSN: 2745-3673
http://www.Rcsj.ir

۳- پیاده سازی

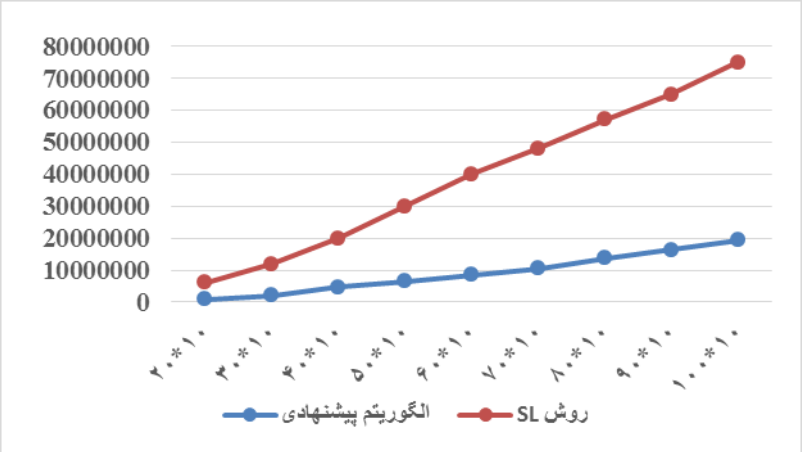
پیاده‌سازی در محیط نرم‌افزار نت بینز و با استفاده از زبان برنامه‌نویسی جاوا انجام گرفته است، برنامه مورد استفاده در این پژوهش با استفاده از تکنیک تعویض حلقه، به ارزیابی توان مصرفی در اندازه‌های مختلف ماتریس می‌پردازد.

۴- نتایج

جدول ۱ و شکل ۳، توان صرف شده الگوریتم پیشنهادی و روش یادگیری با نظارت در زمان‌های مختلف پاسخ با اندازه ثابت تعیین‌شده را نشان می‌دهد. محور افقی اندازه‌های ماتریس یا تعداد خانه‌های ماتریس را مشخص نموده و محور عمودی مشخص‌کننده زمان پاسخ (بر حسب میلی ثانیه) است، در شکل مذکور، در اندازه‌های کوچکتر الگوریتم پیشنهادی نسبت به روش یادگیری با نظارت مطلوب‌تر است ولی در اندازه‌های بزرگتر الگوریتم پیشنهادی نتیجه مطلوب‌تر و بهتری دارد. با توجه به جدول ۲ در اندازه‌های کوچکتر، زمان پاسخ نسبت به اندازه‌های بزرگتر، کمتر و کوچکتر است و در نتیجه توان مصرف شده کمتری بکار می رود.

جدول ۱- توان صرف شده در زمان‌های پاسخ (میلی ثانیه) برای الگوریتم پیشنهادی و روش یادگیری با نظارت به ازای اندازه‌های مشخص‌شده ماتریس

مقیاس ها	روش الگوریتم پیشنهادی	روش یادگیری با نظارت [۱۶]
۱۰×۲۰	۸۰۰۰۰۰	۶۰۰۰۰۰۰
۱۰×۳۰	۲۱۵۰۰۰۰	۱۲۰۰۰۰۰۰
۱۰×۴۰	۴۷۳۸۰۰۰	۲۰۰۰۰۰۰۰
۱۰×۵۰	۶۴۵۰۰۰۰	۳۰۰۰۰۰۰۰
۱۰×۶۰	۸۶۰۰۰۰۰	۴۰۰۰۰۰۰۰
۱۰×۷۰	۱۰۵۴۰۰۰۰	۴۸۰۰۰۰۰۰
۱۰×۸۰	۱۳۶۵۰۰۰۰	۵۷۰۰۰۰۰۰
۱۰×۹۰	۱۶۴۱۲۰۰۰	۶۵۰۰۰۰۰۰
۱۰×۱۰۰	۱۹۳۲۷۰۰۰	۷۵۰۰۰۰۰۰



شکل ۳: توان صرف شده برای الگوریتم پیشنهادی و روش یادگیری با نظارت در زمان‌های مختلف پاسخ

ISSN: 2745-3673
http://www.Rcsj.ir

۵- نتیجه‌گیری

در این مقاله، کل عناصر ماتریس در اندازه‌های مختلف که کل قطعه برنامه مذکور و نشان دهنده وابستگی بین قطعات کد است را بر روی پردازنده‌های موازی به اجرا درآوردیم. هدف ما این است که در کمترین توان مصرفی، کل ماتریس در مقیاس‌های مختلف که قطعه برنامه ذکر شده است، اجرا گردد. بدین ترتیب از تکنیک تعویض حلقه و ترکیب الگوریتم‌های ژنتیک وK-means استفاده نمودیم. از آنجایی‌که این مسئله، یک مسئله غیر قطعی سخت است، به این دلیل با بکارگیری ترکیب الگوریتم‌های تکاملی و خوشه‌بندی، مسئله مورد نظر را با الگوریتم پیشنهادی انجام دادیم که بررسی‌ها حاکی از بهبود نتایج در توان مصرف شده است.

ISSN: 2745-3673
http://www.Rcsj.ir

منابع و مراجع

[1] M. Kandemir, S.W. Son, and G. Chen, “An Evaluation of Code and Data Optimizations in the Context of Disk Power Reduction” Procceding of the International Symposium on Low-Power Electronics and Design, PP: 209-214, San Diego, California, USA, 8-10 Aug, 2005.
[2] A. Oscar, J. Manuel, "A Survey of Software Optimization Techniques for Low-Power Consumption" Proceedings Computing Research Conference, PP:43 – 46, Estados Unidos, December, 2002.
[3] V, Sarkar, “Challenges in Code Optimization of Parallel Programs” 18 International Conference on Theory and Practice of Software, PP:1, Springer, Berlin, March, 2009.
[4] V, Sarkar, “Code Optimization of Parallel Programs: Evolutionary vs. Revolutionary Approaches” 6 th Annual ACM International Symposium on Code Generation and Optimization, PP:1, ACM, New York, USA, April, 2008.
[5] D. Saougkos. G. Manis, "Self adaptive run time scheduling for the automatic parallelization of loops with the C2µTC/SL compiler" Parallel Computing, Vol. 39, No. 10, PP: 603-614, October, 2013.
[6] C. Hsu, U. Kremer, “The design, implementation, and evaluation of a compiler algorithm for CPU energy reduction”, in Proc. of the ACM SIGPLAN Conf. on Programming Language Design and Implementation, San Diego, California, USA, PP: 38–48, June, 2003.
[7] R. Leupers. “Code Optimization Techniques for Embedded Processors: Methods, Algorithms, and Tools”, Kluwer Academic Publishers Norwell, MA, USA, 2000.
[8] N. Edward Johnson, “Code Size Optimization for Embedded Processors” PhD. thesis, University of Cambridge, May, 2004.
[9] P. A. Kulkarni, D. B. Whalley, G. S. Tyson, and J. W. Davidson, “Practical exhaustive optimization phase order exploration and evaluation” TACO, Vol. 6, No. 1, PP: 1-32, March, 2009.
[10] M. Mitchell, "an Introduction to Genetic Algorithms", MIT Press, Fifth printing, 1999.
[11] D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison – Wesley Publishing Company, 1st Edition, 1989.
[12] D. A. Coley, "an Introduction to Genetic Algorithms for Scientists and Engineers", World Scientific Publishing Company, River Edge, NJ, USA, 1998.
[13] G. Tzortzis, A. Likas, “The Global Kernel k-Means Clustering Algorithm” International Joint Conference on Neural Networks (IJCNN), PP: 1978-1985, July, 2008.
[14] R. Cordeiro de Amorim, “A survey on feature weighting based K-Means algorithms” Journal of Classification (Springer), Vol. 33, No.2, PP: 210–242, July, 2016.
[15] S. Kaxiras and M. Martonosi. “Computer Architecture Techniques for Power-Efficiency”, Morgan and Claypool Publishers, 2008.
[16] D. Saougkos, G. Manis. “Self-adaptive run time scheduling for the automatic parallelization of loops With the C2lTC/SL compiler” Parallel Computing, Vol 39, PP: 603–614, 2013.
[17] K. Cooper, T. Harvey, and T. Waterman, “An Adaptive Strategy for Inline Substitution” In Compiler Construction, ser. Lecture Notes in Computer Science, L. Hendren, Ed. Springer Berlin Heidelberg, Vol. 4959, PP: 69–84, April, 2008.
[18] T. Bostoen, S. Mullender, Y. Berbers, “Power reduction techniques for data center storage systems” ACM Computing Surveys (CSUR), Vol. 45, No. 3, June, 2013.
[19] J. Mac Queen, "Some Methods for classification and Analysis of Multivariate Observations” Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, Vol. 1, PP:281-297, 1967.
[20] R. Leupers, P. Marwedel, “Function inlining under code size constraints for embedded processors” In ICCAD ’99: Proceedings of the IEEE/ACM international conference on Computer aided design, Piscataway, IEEE Press, and PP: 253–256, California, USA, and Nov, 1999.
[21] K. Cooper, P. Schielke, D. Subramanian, “Optimizing for reduced code space using genetic algorithms” In Proceedings of the ACM SIGPLAN workshop on Languages compilers and tools for embedded systems, PP: 1–9, New York, USA, July, 1999.

ISSN: 2745-3673
http://www.Rcsj.ir

[22] D. A. Ortiz, N. G. Santiago, “Impact of Source Code Optimization on Power Consumption of Embedded Systems” 6th IEEE International Northeast Workshop on Circuits and Systems and TAISA Conference, PP: 133 – 136, Montreal, QC, June, 2008.