ششمین کنفرانس ملّی
محاسبات توزیعی و پردازش داده های بزرگ
۱۳ اسفند ۱۳۹۹
6th National Conference on Distributed Computing and Big Data Processing
دانشگاه شهید مدنی آذربایجان
ISC

# کاهش makespan در زمانبندی اینترنت اشیاء مبتنی بر ابر با روش بازپخت شبیه‌سازی شده*

زهرا محمدبیگی دهقی[۱]، عبدالرضا رسولی کناری[۲]، محبوبه شمسی[۳]

[۱] دانشجو کارشناسی ارشد مهندسی کامپیوتر(نرم افزار)، گروه برق و کامپیوتر دانشگاه صنعتی قم
mohammadbeigi.z@qut.ac.ir

[۲] استادیار دانشکده برق و کامپیوتر دانشگاه صنعتی قم
rasouli@qut.ac.ir

[۳] استادیار دانشکده برق و کامپیوتر دانشگاه صنعتی قم
shamsi@qut.ac.ir

## چکیده

**اینترنت اشیا در دنیای امروز پیشرفت فزاینده‌ای داشته. همچنین در کنار آن فناوری‌هایی رشد کرده که باعث تحمیل بار محاسباتی زیاد بر سرورهای ابر می‌کند بنابراین استفاده از محاسبات مِه افزایش یافته است. در نتیجه برنامه‌ها به بخش‌های کوچک‌تر خود تقسیم می‌شوند و ارتباطات بین آن‌ها زمان‌بندی این کارها را چالش‌برانگیز کرده است. از این رو نویسندگان بر این شدند تا یک روش بهینه برای زمان‌بندی کارها مبتنی بر اینترنت اشیا را معرفی کنند. روش پیشنهادی، الگوریتم بازپخت شبیه‌سازی‌شده است که یک روش فراابتکاری برگرفته از عملیاتی در متالورژی است. الگوریتم‌های فراابتکاری راه حل‌های مناسبی برای مسائل بهینه‌سازی ارائه می‌دهند که بهترین پاسخ یا پاسخی نزدیک به بهترین را ارائه می‌دهند. الگوریتم بازپخت شبیه‌سازی شده روی چندین گراف تصادفی اعمال شده و کارها روی دو پردازنده شبیه‌سازی شده قرار گرفتند. نتایج در زمان اجرای کل برنامه (makespan) در این الگوریتم در مقایسه با الگوریتم MMAS کاهش ۱.۷ درصدی را نشان می‌دهد. همچنین زمان اجرای برنامه (execution time) کمتر از الگوریتم بازپخت MMAS است.**

## کلمات کلیدی

**اینترنت اشیا، ابر، مِه، زمان‌بندی، بازپخت شبیه‌سازی‌شده،** simulated annealing، scheduling

## ۱- مقدمه

امروزه برنامه‌های کاربردی روزبه‌روز در حال بزرگ شدن هستند. استفاده روزافزون از شبکه‌های عصبی و یا رشد مفهوم کلان‌داده‌ها نمونه‌ای از این داده‌های بزرگ هستند. از این رو اجرای آنها با ماشین‌های تک‌پردازنده‌ی معمولی کاری سخت و زمان‌بر به نظر می‌رسد. بنابراین استفاده از ماشین‌های چندپردازنده‌ای و یا پردازنده‌های چندهسته‌ای رو به گسترش هستند. اینترنت اشیاء[□] نیز در حال گسترش است. اینترنت اشیاء شبکه ای از دستگاه های فیزیکی، خودروها، وسایل کاربردی خانگی، سیستمهای نهفته، نرم افزارها، سنسورها و تجهیزات ارتباطی است که به این وسایل در ارتباط با یکدیگر، اتصال و جمع آوری داده کمک میکند. این شبکه ی وسایل

---

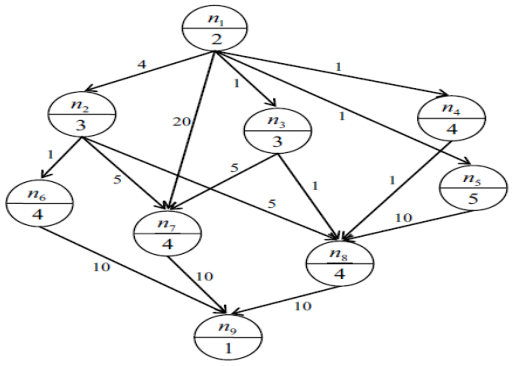
* Simulated Annealing(SA)

□ Internet of Things

معمولا از طریق دستگاه های استانداردی مانند رایانه های دسکتاپ، لپ تاپ ها، تلفن های هوشمند و تبلت ها کنترل میشوند. داده‌های حسگر[1] اینترنت اشیاء نمی‌توانند به راحتی و به طور مستقیم در سرورهای راه دور بارگذاری شوند. این مشکل به علت محدودیت در زیرساخت‌های شبکه به وجود می‌آید[1]. از این رو با گسترش رایانش مِه[2]، استفاده از مراکز داده‌ی کوچک محلی برای بارگذاری داده‌های اینترنت اشیاء رو به رشد است. بنابراین در چنین ساختاری برنامه‌ها به بلاک‌های سازنده‌ی خود تقسیم می‌شوند که وظیفه[3] نامیده می‌شوند. سپس یک گراف جهت‌دار بدون دور[4] (DAG) شامل وظایف به عنوان گره‌ها و اولویت بین وظایف به عنوان لبه‌ها ساخته می‌شود. با توجه به وابستگی‌های زمانی میان این وظایف، زمانبندی انجام این وظایف به صورت بهینه و به دست آوردن کمترین زمان اجرا به یک چالش بدل شده است.[۲] مسئله بهینه‌سازی زمانبندی وظایف به عنوان یک مسئله‌ی NP-Hard شناخته می‌شود، بنابراین برای حل این‌گونه مسائل به سراغ روش‌های اکتشافی[5] یا فرااکتشافی[6] می‌رویم و تلاش می‌کنیم تا بهینه‌ترین راه حل را برای این مسائل بیابیم. Wu و Dajski در سال ۱۹۹۰ الگوریتم MCP( مسیر بحرانی اصلاح‌شده) را معرفی نمودند. ALAPگره به عنوان معیار محاسبه میشود. لیست آماده به ترتیب صعودی ALAP مرتب می‌شوند. الگوریتم MCP به ترتیب گره‌ها را از لیست خارج کرده و به پردازنده تحویل می‌دهد.[۳] پیچیدگی زمانی این الگوریتم $O(mn2)$ است. الگوریتم ISH (اکتشاف زمانبندی درجی) در سال ۱۹۸۷ توسط Kruatrachue و Lewis معرفی گردید. این الگوریتم از تخصیص وظیفه مبتنی بر درج در اسلات‌های خالی زمان بیکاری پردازنده استفاده می‌کند. برای مرتب سازی وظایف در لیست آماده از ویژگی SLevel استفاده می‌شود.[۴] پیچیدگی زمانی این الگوریتم $O(mn2)$ است. فام و همکاران روشی مبتنی بر makespan و هزینه معرفی کردند تا بالاترین مقدار trade off برای makespan و هزینه را پیدا کنند. آنها از یک فاکتور منطقی برای خطی کردن trade off استفاده کردند.[۵] در کاری دیگر زِنگ و همکاران به صورت همزمان تصاویر کارها و استراتژی های زمانبندی را جای‌گذاری کردند تا زمان تکمیل سرویس‌ها را به حداقل برسانند. همچنین با ایجاد توازن در بارگذاری کارها زمان تأخیر را کاهش دادند.[۶] بویری و همکاران در سال ۲۰۱۸ از سیستم Max-Min-Ant-System (MMAS) استفاده کردند.[۲] طاهری و همکاران یک روش تلفیقی دوفازی پیشنهاد کردند که مبتنی بر تجزیه‌ی گراف ورودی و بر مبنای پارتیشن‌بندی طیفی است. این روش هر بخش از گراف کار را به یک پردازنده کم‌توان به منظور کاهش توان مصرفی نسبت می‌دهد.[۷]

### ۱-۱- گراف جهت‌دار بدون دور

گراف جهت‌دار بدون دور که با نام گراف کار شناخته می‌شود برای مدل-سازی مسئله زمان‌بندی در محیط‌های چندپردازنده به کار می‌رود و به شکل مقابل تعریف می‌شود. $G = \{N, E, W, C\}$ شامل $N = \{n1, n2, \dots nn\}$ یعنی تعدادی رأس یا گره، $E = \{(ni, nj)|(ni, nj) \in E\}$شامل لبه‌هایی که دو رأس را به صورت جهت دار به هم متصل می‌کنند و $(ni, nj) \in E$به این معناست که ni باید پیش از nj اجرا شود که ni پدر و nj فرزند نامیده می‌شوند. $W = \{w1, w2, \dots, wn\}$ وزن هر گره است. وزن هر گره زمان لازم برای




**شکل ۱ نمونه‌ای از یک گراف جهت‌دار بدون دور**

اجرای آن وظیفه است. $C = \{c(ni, nj)|\ (ni, nj) \in E\}$که وزن هر یال است. وزن یال زمان لازم برای انتقال داده از کار i به کار j است. اگر هر دو گره پدر و فرزند روی یک پردازنده اجرا شوند، زمان انتقال برابر با صفر در نظر گرفته می‌شود. چرا که داده‌های مورد نیاز گره فرزند روی حافظه محلی موجود است. گره‌ای که پدر ندارد، گره ورودی و گره‌ای که فرزند ندارد، گره خروجی نامیده می‌شود.

## ۲- روش تخصیص کارها به پردازنده‌ها

بیشتر الگوریتم‌های زمان‌بندی از روش زمان‌بندی لیست به عنوان روش پس زمینه استفاده می‌کنند. اصول کار به این صورت است که یک لیست آماده از کارهای آماده ایجاد شود. کارهای آماده کارهایی هستند که پدر ندارند یا تمامی پدرهای آنها از پیش زمان‌بندی شده‌اند. سپس کاری که اولویت بیشتری دارد انتخاب شده و به پردازنده‌ای که اجازه‌ی EST می‌دهد تخصیص داده می‌شود. هر زمان که نوبت به اجرای وظیفه‌ای می‌رسد، پارامتری به نام $EST(n_i, p_j)$ برای کار $n_i$ و پردازنده‌ی $p_j$ محاسبه می‌شود. این عبارت بیانگر زودترین زمانی است که کار $n_i$ روی پردازنده‌ی $p_j$ قابل اجرا است. این پارامتر برای تمام پردازنده‌ها محاسبه می‌شود. سپس پردازنده‌ای که کمترین میزان EST را برای $n_i$ داشته باشد، به عنوان پردازنده‌ی میزبان این کار انتخاب می‌گردد.

برای محاسبه‌ی EST قواعدی وجود دارد:

اگر تمام پیشینیان کار روی همین پردازنده اجرا شده‌اند $EST(n_i, p_j)$ برابر خواهد بود با $Avail(p_i)$. پارامتر Avail نشان‌دهنده‌ی زودترین زمانی است که $p_j$ آماده‌ی اجرای یک کار است.

در غیر این صورت، EST با فرمول زیر محاسبه می‌گردد:

$$EST(n_i, p_i) = \begin{cases} 0, & if\ n_i = entry - node \\ max \begin{cases} (AFT(n_k)), & if\ processor(n_k) = p_j \\ (AFT(n_k) + c(n_k, n_i)), & else \end{cases} \\ n_k \in Parent(n_i) \end{cases}$$

که:

$AFT(n_k) = AST(n_k) + w_k$ زمان پایان واقعی کار $n_k$ و $parents(n_i)$ مجموعه همه‌ی پدران $n_i$ است.

و $AST(n_k)$ زمان شروع واقعی کار $n_k$ است و با فرمول زیر محاسبه می‌شود.

$$AST(n_k) = \min_{j=1}^{m}(max(Avail(p_i), EST(n_k, p_i)))$$

پس از تخصیص همه کارها به پردازنده‌های مورد نظر، پارامتر Makespan برای کل مجموعه کارها محاسبه می‌شود. این پارامتر نشان‌دهنده‌ی کل زمان لازم برای انجام کارها است و با فرمول زیر محاسبه می‌شود:

$$makespan = \max_{i=1}^{n}(FT(n_i))$$

## ۳- simulated annealing

بازپخت یا تبرید شبیه‌سازی یک الگوریتم فراابتکاری ساده و موثر برای حل مسائل بهینه‌سازی است. این الگوریتم از روشی با همین نام در رشته متالورژی الهام گرفته است. در متالورژی برای رسیدن به کریستال‌های مقاوم ابتدا ماده را تا حداکثر مقدار گرم کرده و سپس آن را به تدریج خنک می‌کنند تا انرژی آن به حداقل برسد و ذرات ماده یکدیگر را مرتب کنند و ساختار کریستالی تشکیل دهند. تبرید تدریجی را می‌توان اینگونه تعبیر کرد که در این روش احتمال انتخاب پاسخ‌های بد به تدریج کم می‌شود. در این روش یک آشفتگی تصادفی بین دو ذره به وجود می‌آید که این تغییر در الگوریتم SA به جابه‌جایی تصادفی دو عضو لیست تعبیر می‌شود. سپس میزان تغییر انرژی یا $\Delta E$ محاسبه می‌شود. اگر $\Delta E < 0$ یعنی انرژی کاهش داشته باشد، روند ادامه پیدا می‌کند، در غیر این صورت اگر $\Delta E \geq 0$ حالت جدید با یک احتمال غیرصفر پذیرفته می‌شود. این مراحل آنقدر تکرار می‌شود تا سیستم به یک حالت معقول با کمترین انرژی برسد. [۸] هر جایگشت از وظایف یک حالت نامیده می‌شود. هر حالت یک پاسخ محتمل است. همسایه‌ی یک پاسخ، جایگشت تازه‌ای است که از انتخاب دو وظیفه به صورت تصادفی و جابه‌جایی آن‌ها به دست می‌آید. الگوریتم‌های ابتکاری ساده از یک حالت شروع می‌کنند و به بهترین حالت از بین حالت‌های همسایه حرکت می‌کنند تا زمانی که حالت بهتری در بین حالت‌های همسایه وجود نداشته باشد. این عمل ممکن است ما را در تله‌ی یک جواب بهینه محلی گرفتار کند. اما الگوریتم‌های فراابتکاری مانند $SA$ جواب‌های دیگر را کامل رد نمی‌کنند بنابراین احتمال رسیدن به بهینه محلی از بین می‌رود.

## ۴- الگوریتم پیشنهادی

در این روش ابتدا یک جایگشت اولیه برای کارها در نظر گرفته می‌شود. کارهایی که پدر ندارند به صورت تصادفی در لیست قرار می‌گیرند سپس فرزندان غیرتکراری آن‌ها نیز به لیست اضافه می‌گردند. سپس عملیات بازپخت شبیه‌سازی شده روی این لیست اولیه اعمال می‌گردد.

انرژی حالت فعلی محاسبه می‌شود و حالت فعلی و انرژی آن نگهداری می‌شوند. سپس به تعداد گام‌های از پیش تعیین‌شده عملیات Move انجام می‌شود و هر بار $\Delta E$ را محاسبه می‌گردد که تفاوت makespanها است. اگر $\Delta E > 0$ ، با وجود اینکه انرژی محاسبه شده کمتر نشده، حالت جدید کاملاً رد نمیشود، بلکه احتمال پذیرش محاسبه میگردد، اگر احتمال از یک عدد تصادفی بزرگتر بود، حالت پذیرفته می‌شود. این کار از به دام افتادن در بهینه‌ی محلی جلوگیری می‌کند.

$$P(E, E', T) = exp(-\Delta E/T)$$

اگر $P(E, E', T) < random(0,1)$ به حالت قبل باز می‌گردیم.

این عملیات بارها تکرار شده و حالت با کمترین انرژی به دست می‌آید.

طبق الگوریتم بازپخت شبیه‌سازی شده دو پارامتر برای کار در نظر گرفته شده است.

*Move*: دو گره از لیست به صورت تصادفی انتخاب می‌شوند. سپس جای این دو گره عوض می‌شود. پس از جابه‌جایی، محدودیت‌های اولویت[۷] بررسی می‌شوند. این محدودیت‌ها به این موضوع اشاره دارند که بعضی کارها به خروجی کارهای دیگر برای اجرا نیاز دارند، بنابراین نمی‌توانند زودتر اجرا شوند. اگر محدودیت‌های اولویت رعایت نشده باشند، جابه‌جایی لغو شده و مجدداً انجام می‌گیرد.

*energy*: از آنجایی که هدف ما کاهش *makespan* است، پارامتر *makespan* را به عنوان انرژی انتخاب می‌کنیم. بنابراین هر بار که *move* صورت می‌گیرد، *makespan* محاسبه می‌گردد و با مقدار قبلی آن مقایسه می‌شود.

در پایان جایگشتی از کارها که هم محدودیت‌های اولویت را رعایت کند و هم کمترین *makespan* را داشته باشد به عنوان خروجی معرفی می‌گردد.

## ۵- شبیه‌سازی

برای شبیه‌سازی این الگوریتم از زبان python 3.8 استفاده شده است. پایتون با توجه به متن‌باز بودن، امکان توسعه‌ی کتابخانه‌های گوناگون را فراهم کرده است. از این رو کتابخانه‌ی قدرتمندی نیز برای شبیه‌سازی بازپخت شبیه‌سازی شده برای آن توسعه یافته است.

برای آزمودن این الگوریتم ۳۰ گراف جهت‌دار بدون دور به صورت تصادفی تولید شده است. این گراف‌ها تعداد گوناگونی گره دارند که به طور تصادفی از لیست {۳۲، ۶۴، ۱۲۸، ۲۵۶، ۵۱۲} انتخاب می‌شوند. این گراف‌ها با استفاده از زبان پایتون تولید شده و درپایگاه داده‌ی قدرتمند Neo4j که مختص گراف است، ذخیره می‌گردند.

تعداد گام‌های الگوریتم ۵۰۰۰ در نظر گرفته شده است. به این معنا که با تکرار ۵۰۰۰ باره‌ی الگوریتم روی گراف، به پاسخ مورد نظر خواهیم رسید. وزن گره‌ها و وزن یال‌ها از طریق توزیع یکنواخت و به طور تصادفی تولید می‌گردد، بنابراین آزمایش به طور استاندارد انجام می‌گیرد.

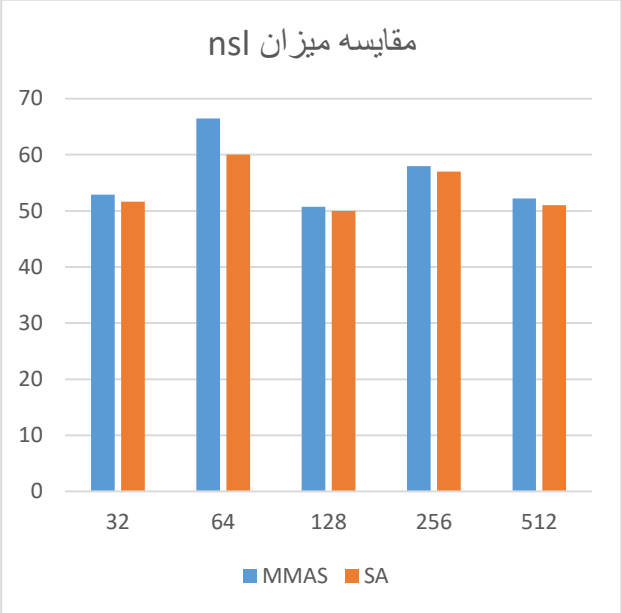
## ۶- نتایج اجرا

با شبیه‌سازی الگوریتم زمان‌بندی وظایف با روش بازپخت شبیه‌سازی‌شده و مقایسه‌ی آن با الگوریتم MMAS[8] به این نتیجه می‌رسیم که این الگوریتم همیشه جایگشتی از وظایف با کوتاهترین زمان اجرا یا makespan را پیدا می‌کند. در صورتی که MMAS معمولاً پاسخی نیمه‌بهینه را پیدا می‌کند و ممکن است در بهینه‌های محلی به تله افتد.

به دلیل متفاوت بودن اندازه‌ی گراف‌ها، از پارامتری به نام nsl[9] استفاده می‌گردد.

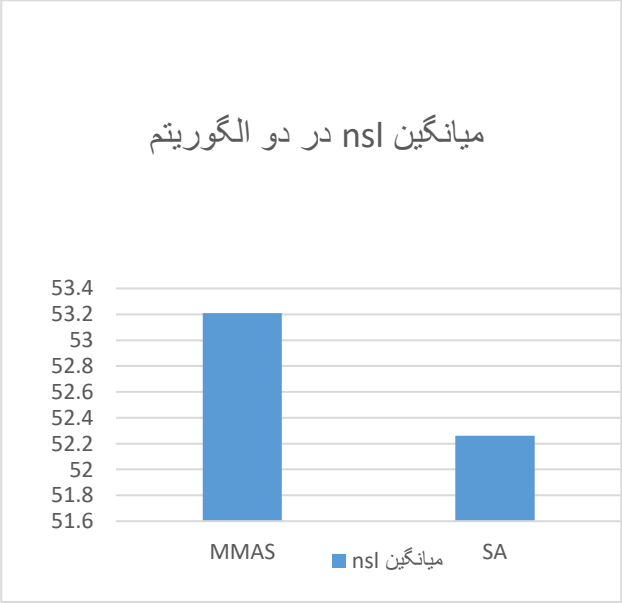
$$NSL = \frac{Scheduled\ Lingth(makespan)}{\sum n_i \in cp^{w_i}}$$

همچنین زمان اجرا[10]ی الگوریتم بازپخت شبیه‌سازی شده کمتر از الگوریتم MMAS است.




## ۷- نتیجه

در این پژوهش یک روش فراابتکاری برای زمان‌بندی وظایف در اینترنت اشیای مبتنی بر ابر ارائه شد. با توجه افزایش بار محاسباتی در شبکه‌های اینترنت اشیا نیاز به الگوریتم‌های زمان‌بندی بهتر به شدت




حس می‌شود که در این پژوهش از الگوریتم بازپخت شبیه‌سازی شده برای زمان‌بندی بهتر گره‌های شبکه بهره گرفته شد. ملاک ارزیابی

زمـان اجرای کـل یـا makespan بود و آزمـایش بر روی ۳۰ گراف تصـــادفی انجام شـــد که این الگوریتم در مقایســـه با الگوریتم مورچه کمینه-بیشـینه حدود ۱.۷ درصـد بهبود داشـت. همچنین زمان اجرای این الگوریتم نسبت به mmas مقداری کاهش نشان می‌دهد.

**۸- مراجع**


۱. Botta, A., et al., *Integration of cloud computing and internet of things: a survey.* Future generation computer systems, 2016. **56**: p. 684-700.

۲. Boveiri, H.R., et al., *An efficient Swarm-Intelligence approach for task scheduling in cloud-based internet of things applications.* Journal of Ambient Intelligence and Humanized Computing, 2019. **10**(9): p. 3469-3479.

۳. Wu, M.-Y. and D.D. Gajski, *Hypertool: A programming aid for message-passing systems.* IEEE transactions on parallel and distributed systems, 199۰. ۱(۳): p. 330-343.

۴. Kruatrachue, B., *Static task scheduling and grain packing in parallel processing systems.* 1987.

۵. Pham, X.-Q. and E.-N. Huh. *Towards task scheduling in a cloud-fog computing system*. in *2016 18th Asia-Pacific network operations and management symposium (APNOMS)*. 2016. IEEE.

۶. Zeng, D., et al., *Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system.* IEEE Transactions on Computers, 2016. **65**(12): p. 3702-3712.

۷. Taheri, G., et al., *A hybrid algorithm for task scheduling on heterogeneous multiprocessor embedded systems.* Applied Soft Computing, 2020. **91**: p. 106202.

۸. Van Laarhoven, P.J. and E.H. Aarts, *Simulated annealing*, in *Simulated annealing: Theory and applications*. ۱۹۸۷, Springer. p. 7-15.


## زیرنویس‌ها

۱ sensory data
۲ fog computing
۳ task
۴ Directed Acyclic Graph
5 heuristic
۶ meta heuristic
۷ predecence constrains
8 Max Min Ant System
9 Normalized scheduled length
۱۰ Execution time