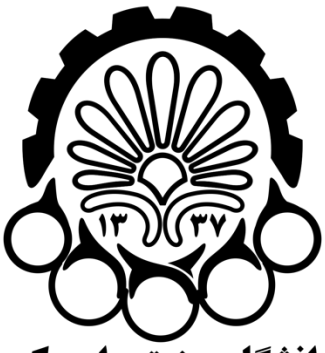


دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

گزارش کارآموزی

محل کارآموزی : پژوهشگاه دانش های بنیادی (IPM)

نام استاد کارآموزی : دکتر حامد فربه

نام دانشجو : محمد رضا جبلی حاجی آبادی

شماره دانشجویی : ۹۴۳۱۰۳۵

تابستان ۱۳۹۷

چکیده

هدف اصلی از انجام این دوره کارآموزی، کسب توانایی‌های لازم برای مطالعه بر روی مقالات متعدد و همچنین پیاده‌سازی الگوریتم‌های بهینه‌سازی جدید برای پروژه‌ی مطرح در محل کارآموزی است. این دوره در پژوهشگاه دانش‌های بنیادی[۱] و در مرکز تورین محاسباتی ملی[۲] زیر نظر دکتر کامران لطفی انجام گرفت.

پروژه‌ی اصلی این مرکز در دوره کارآموزی مسئله‌ی جایابی ماشین‌های مجازی[۳] بر روی سرورها است. با توجه به افزایش درخواست‌های کاربران و اهمیت پاسخ سریع‌تر و مناسب به کاربران در امر محاسبات ابری[۴]، نیاز به ایجاد تغییراتی در ساختار الگوریتم‌های کنونی مورد استفاده که بتواند کاربر را راضی نگه‌دارد، حس می‌شود. از این رو و به طور مشخص، طراحی و پیاده‌سازی الگوریتم‌های بهینه‌سازی مختلف و مقایسه آن‌ها با یکدیگر از لحاظ انرژی مصرفی هدف اصلی پروژه است تا پس از بررسی و آزمایش‌های دقیق، یک الگوریتم به عنوان الگوریتم اصلی برای این مسئله مورد استفاده قرار گیرد.

---

[۱] Institute for Research in Fundamental Sciences (IPM)

[۲] Grid Computing

[۳] Virtual Machine Placement

[۴] Cloud Computing

فهرست مطالب

فصل اول : مقدمه

همانگونه که ذکر شد، هدف از انجام این کارآموزی کسب مهارت و تجربه در زمینه طراحی و پیاده‌سازی الگوریتم‌های بهینه‌سازی جدید برای مسائل مطرح شده در دنیای امروز از جمله مسئله محاسبات ابری است که به سبب افزایش کاربران این تکنولوژی جدید و اهمیت پاسخ‌گویی در زمانی کم و با بالاترین کیفیت به آن‌ها، به یک موضوع مهم و ضروری در سازمان‌هایی که چنین امکاناتی را در اختیار کاربران خود قرار می‌دهند، تبدیل شده است.

علاوه بر این یکی دیگر از اهداف این دوره، یادگیری شیوه‌ی مطالعه مقالات است که بتوانیم آن‌ها را به درستی درک کرده و پیاده‌سازی کنیم.

در طول مدت انجام پروژه این مرکز، آشنایی با زبان برنامه‌نویسی پایتون[۵] نیز حاصل شد که جزو پرکاربردترین زبان‌های برنامه‌نویسی دنیا است و یارگیری آنن می‌تواند در آینده به دلیل امکانات و ابزارهای قدرتمندی که در زمینه یادگیری ماشین[۶]، کلان‌داده‌ها[۷] و ... در اختیار ما قرار می‌دهد، سودمند باشد.

همانطور که پیش از این اشاره شد، پروژه این مرکز طراحی و پیاده‌سازی الگوریتم‌های بهینه‌سازی برای مسئله جایابی ماشین مجازی است و در مدت ۲ ماه طول کارآموزی، چندین الگوریتم از جمله الگوریتم‌های ACO[۸]، ABC[۹]، FO[۱۰] و Tabu[۱۱] پیاده‌سازی و مورد بررسی قرار گرفت که در ادامه گزارش کارآموزی به‌طور دقیق به توضیح هر یک از آن‌ها پرداخته خواهد شد.

در ادامه این گزارش پس از معرفی محل کارآموزی، تجربیات و فعالیت‌های انجام شده در طول مدت کارآموزی به‌طور مفصل توضیح داده خواهد شد و در نهایت نتایج حاصل از کارآموزی عنوان می‌شود و در انتها مراجع اصلی کار معرفی خواهد شد.

---

[۵] Python

[۶] Machine Learning

[۷] Big Data

[۸] Ant Colony Optimization

[۹] Artificial Bee Colony Optimization

[۱۰] Firefly Optimization

[۱۱] Tabu Search

فصل دوم : معرفی محل کارآموزی

تاریخچه

پژوهشگاه دانش‌های بنیادی موسسه‌ای وابسته به وزارت علوم، تحقیقات و فناوری است که در سال ۱۳۶۸ با نام "مرکز تحقیقات فیزیک نظری و ریاضیات" تاسیس شد و هدف اولیه آن پیشبرد پژوهش و نوآوری در این دو رشته و ضمنا فراهم آوردن الگویی بود که به ترویج اعتلای فرهنگ پژوهش در سطح کشور کمک کند. مرکز فعالیت خود را با ۳ هسته تحقیقاتی در فیزیک نظری و ۳ هسته تحقیقاتی در ریاضیات و با امکاناتی اندک آغاز کرد ولی به تدریج با توسعه امکانات و جذب دانشورانی از رشته‌های دیگر، فعالیت آن به حیطه‌ای دیگری گسترش یافت و در سال ۱۳۷۶ نام آن به "پژوهشگاه دانش‌های بنیادی" تغییر کرد. این پژوهشگاه در حال حاضر با ۸ پژوهشکده در زمینه‌های گوناگون علوم بنیادی و برخورداری از زیرساخت‌ها و امکانات لازم (شبکه الکترونیکی، کامپیوتر، آزمایشگاه‌ها و کتابخانه مجهز و روزآمد) که دائما هم رو به توسعه است، حضور فعالی در جریان پژوهشی کشور در این دانش‌ها دارد. شرح وظایف و نوع فعالیت‌های پژوهشگاه در سطور آینده خواهد آمد لی در یک نگاه کلی به تجربه بیست ساله فعالیت پژوهشگاه سه ویژگی بارز در این تجربه مشهود است: اول، کمیت و کیفیت تحقیقات انجام شده در این نهاد، یعنی کثرت تعداد مقاله‌ای پژوهشی چاپ شده آن در مجله‌های علمی معتبر و تعداد استنادها به آن‌ها؛ دوم، نوعی مدیریت پیویا در امیر پژوهش که می‌تواند الگویی برای موسسات تحقیقاتی باشد یعنی مدیریتی براساس "محوریت محقق"، "استقلال مدیریتی واحدهای پژوهشی" و "انعطاف‌پذیری در تاسیس و انحلال آن‌ها"؛ سوم، نگرش "ملی" پژوهشگاه در ایجاد شبکه ارتباطی الکترونیکی (به نام شبکه علمی-تحقیقاتی ایران یا ایرانت) در سال ۱۳۷۱، که به تدریج و برای اولین بار مراکز علمی-تحقیقی و دانشگاه‌های ایران را به یکدیگر و به جهان علم در خارج مربوط ساخت، و اهتمام پژوهشگاه به اجرای طرح‌های ملی، مانند رصدخانه ملی ایران و شتابگر ملی، از جلوه‌ها و ثمرات این نگرش ملی است.

اهداف و فعالیت‌ها

۱. انجام تحقیقات در زمینه‌های مرتبط با موضوع تاسیس پژوهشگاه به‌طور مستقل و یا با هکاری مراکز علمی و پژوهشی داخل و خارج کشور؛

۲. ایجاد ارتباط فعال و سازنده با سایر موسسات و جوامع علمی و پژوهشی در داخل و خارج از کشور از طریق برگزاری انواع همایش‌ها، مبادله محقق و اجرای طرح‌های مشترک؛

۳. همکاری با دانشگاه‌ها و مراکز آموزش عالی و موسسات پژوهشی کشور و سایر نهادها در راستای پیشبرد اهداف موضوع تاسیس پژوهشگاه از طریق ارائه تسهیلات مختلف، پذیرش طرح‌های تحقیقاتی، ایجاد امکان گذراندن فرصت‌های مطالعاتی در پژوهشگاه؛

۴. ایجاد زمینه‌های مناسب برای جذب دانشمندان و پژوهشگران ایرانی؛

۵. کمک به پرورش محقق در زمینه‌های موضوع تاسیس از طریق دایر کردن دوره‌های تحصیلات تکمیلی و اعطای کمک هزینه تحصیلی؛

۶. نشر و ترویج یافته‌های علمی در زمینه‌های فعالیت پژوهشگاه از طریق انتشار کتب و نشریات و تشکیل تجمعات پژوهشی و آموزشی؛

۷. ارائه خدمات علمی و فنی در چارچوب فعالیت‌های پژوهشگاه

۸. بررسی و شناسایی نیازهای پژوهشی در زمینه دانش‌های بنیادی

۹. تاسیس مرکز خدمات شبکه‌ای با هدف برقراری ارتباطات شبکه‌ای جهت ارائه خدمات به پژوهشگاه و سایر مراکز علمی و پژوهشی و متقاضیان دیگر و همچنین تلاش برای توسعه فنون مربوط به شبکه در کشور

ارکان پژوهشگاه

ارکان پژوهشگاه عبارت‌اند از هیئت امنا، رئیس و شورای پژوهشگاه. ریاست پژوهشگاه از آغاز تاسیس تاکنون بر عهده دکتر محمد جواد ا. لاریجانی بود است.

ساختار پژوهشی

بخش‌های تحقیقاتی پژوهشگاه متشکل از پژوهشکده‌ها و مراکز تحقیقاتی وابسته است. در حال حاضر، پژوهشگاه شامل ۹ پژوهشکده است.

پژوهشکده ذرات و شتابگرها

پژوهشکده ریاضیات

پژوهشکده علوم زیستی

پژوهشکده علوم شناختی

پژوهشکده علوم کامپیوتر

پژوهشکده علوم نانو

پژوهشکده فلسفه تحلیلی

پژوهشکده فیزیک

پژوهشکده نجوم

این پژوهشکده‌ها طبق اساسنامه پژوهشگاه از استقلال داخلی برخوردارند.

طرح‌های ملی

این طرح‌ها پروژه‌های تحقیقاتی در مقیاس ملی هستند که اجرای آن‌ها به پژوهشگاه دانش‌های بنیادی واگذار شده و از حمایت مالی دولت برخوردارند. این طرح‌ها در حال حاضر عبارت‌اند از:

رصدخانه ملی ایران

همکاری با سرن (CERN)

گرید (Grid) برای محاسبات گسترده علمی

شتابگر خطی Mev۱۰

چشمه نور ایران

شبکه علمی دانشگاه‌ها و دامنه کشوری

مرکز محاسبات ملی با توان بالا

تورین محاسباتی ملی

تورین محاسباتی ملی، شبکه‌ای از مراکز محاسباتی قدرتمند کشور است که توسط یک میان‌افزار بومی و ایرانی، به کاربران امکان بهره‌مندی از این منابع می‌دهد. تورین محاسباتی ملی، سرویس محاسباتی را با ویژگی‌هایی همانند قیمت مناسب، امنیت، نظارت و کیفیت ارائه می‌دهد.

سرویس‌های محاسباتی که این مرکز ارائه می‌دهد عبارت‌اند از:

منابع محاسباتی

ماشین و سرور مجازی

مراکز داده ابری

دوره ۲ ماهه کارآموزی در این مرکز و با تمرکز بر بخش بهینه‌سازی الگوریتم‌های سرویس ماشین و سرور مجازی انجام گرفت.

راه‌های ارتباطی با مرکز تورین محاسباتی ملی

آدرس: تهران، ابتدای بزرگراه ارتش، مقابل اراج، پژوهشگاه دانش‌های بنیادی، جنب باغ لارک، (پروژه تورین ملی)

شماره تماس: ۲۶۱۱۳۲۷۷-۲۱-۹۸+

آدرس ایمیل: gcg@ipm.ir

فصل سوم : فعالیت ها و تجربیات کارآموزی

در این قسمت به بررسی اقدامات و تحقیقات انجام گرفته در طول فرایند کارآموزی میپردازم. هدف از کارهای انجام گرفته مطالعه‌ی تفاوت‌های الگوریتم‌های بهینه‌سازی مختلف و اجرای آنهای روی مسئله ی جایگذاری ماشین مجازی میباشد.

فعالیت اصلی در این دوره کارآموزی، مطالعه مقالات مختلف در زمینه الگوریتم‌های بهینه‌سازی، آشنایی با مسئله جایابی ماشین مجازی، کاهش انرژی مصرفی سرورها در محاسبات ابری و همچنین پیاده‌سازی و نگاشت الگوریتم‌های مطالعه شده به مسئله موردنظر است.

ابتدا لازم است که با مسئله مربوطه در دوره کارآموزی بیشتر آشنا شویم. می‌دانیم که هر ماشین مجازی، میزان خاصی هسته [۱۲] و همچنین حافظه[۱۳] نیاز دارد. حال قرار است تعدادی از این ماشین‌ها را بر روی گروهی از سرورها (که هر کدام میزان هسته و حافظه مشخصی دارند) قرار دهیم به‌گونه‌ای که بتوانیم به هر یک از این ماشین‌ها با سرعت و کیفیت مناسبی سرویس دهیم و همچنین از نظر توان مصرفی نیز بهینه باشد. لذا ورودی مسئله لیستی از ماشین‌های مجازی و همچنین سرورهای سازمان است و خروجی آن یک راه‌حل و ترکیب خاصی از جایگذاری این ماشین‌ها بر روی سرورهاست؛ به شرطی که این جواب، یک جواب بهینه باشد. قبل از معرفی الگوریتم‌ها لازم است اشاره کنیم که در حل مسئله، دو شاخص مورد بررسی ما که تحلیل نتایج هم به آن‌ها وابسته هستند، میزان هسته و حافظه مصرفی هر ماشین مجازی می‌باشد.

مسئله بدین صورت طراحی شده است که تعدادی سرور با تعدادی هسته ی پردازشی و مقداری حافظه وجود دارد. همچنین از طرف دیگر تعدادی درخواست از طرف کاربران وجود دارد که هر کدام تعدادی هسته و مقداری حافظه درخواست میکنند.

هدف از الگوریتم های بهینه سازی تعیین نحوه‌ی جایگذاری کارها در سرورها میباشد به گونه ای که تابع هدف را بیشینه کند. معیار مقدار دهی ما به هر جواب ممکن بدین گونه است که تلاش بر این بوده که تعداد سرورهای بیکار بیشینه شود. چرا که روشن شدن هر سرور باعث از بین رفتن انرژی می‌شود و توان قابل ملاحظه ای مصرف می‌کند. پس اگر مجبور به استفاده از یک سرور می‌شویم بهتر است که تا حد ممکن آن سرور را پر کنیم.

همانطور که قبلا اشاره شد پیاده‌سازی این الگوریتم‌ها به زبان پایتون انجام گرفت و در ادامه این فصل به توضیحات دقیق در مورد معرفی الگوریتم‌های مطالعه شده و پیاده‌سازی آن‌ها برای مسئله موردنظر پرداخته خواهد شد. این الگوریتم‌ها عبارت‌اند از: FFD[۱۴]، BFD[۱۵]، ACO، ABC، Firefly و Tabu Search

[۱۲] Core

[۱۳] Memory

[۱۴] First Fit Decreasing

[۱۵] Best Fit Decreasing

۳-۱- توضیح تابع معیار

تابع معیار یک تابع است که یک راه حل را به عنوان ورودی دریافت می‌کند و به آن یک عدد نسبت می‌دهد بر این اساس که چه مقدار آن راه حل مفید است.

با استفاده از راه حل دریافت شده که نحوه جایگیری هر کدام از درخواست ها در آن تعیین شده، مقدار قسمت خالی و پر هر سرور بدست می‌آید. با استفاده از این اطلاعات میزان بهره برداری از هر سرور محاسبه می‌شود. بدین صورت که نسبت تعداد پردازنده های استفاده شده و تخصیص یافته شده به کل تعداد پردازنده های موجود در هر سرور نشان دهنده ی درصد بهره برداری از آن سرور است. حال به ازای تمامی سرور ها این مقدار محاسبه میشود و مجموع مربع آنها خروجی این تابع می باشد. با یک حساب سر انگشتی با کم کردن تعداد سرور های درگیر مقدار خروجی تابع معیار بزرگتر می‌شود.

$$f = \sum Util(C_i)^2$$

فرمول ۱

۳-۲- الگوریتم بهینه سازی ACO

الگوریتم بهینه سازی لانه‌ی مورچه ها یک الگوریتم بهینه سازی بر اساس جمعیت است که برای ارائه‌ی راه حل‌های حریصانه برای مسائل سخت مطرح می‌شود. در این الگوریتم، مجموعه ای از مورچه های مصنوعی به دنبال راه حل های بهتر برای یک مسئله‌ی مشخص جستجو می‌کنند. با استفاده از ماده‌ی ترشح شده توسط مورچه ها الگوریتم پیش می‌رود به گونه ای که مسیری که مورچه‌ی بیشتری از آنجا عبور کند ماده ی بیشتری دارد و مورچه ها تصمیم می‌گیرند که از آن مسیر حرکت کنند.

۳-۲-۱- فرمول ها :

فرمول ۲

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij}$$

فرمول ۳

$$\Delta\tau_{ij} = \sum_{k=1}^{l} \Delta\tau_{ij}^{k}$$

فرمول ۴

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k & \text{if ant } k \text{ travels on edge } (i,j) \\ 0 & \text{otherwise} \end{cases}$$

فرمول ۵

$$p_{ij}^{k} = \begin{cases} \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{s \in allowed_k} [\tau_{is}]^{\alpha} \cdot [\eta_{is}]^{\beta}} & j \in allowed_k \\ 0 & \text{otherwise} \end{cases}$$

$\Delta\tau_{ij}$: مقدار کل فرمون اضافه شده به مسیر i و j

$\Delta\tau^{k}_{ij}$: مقدار کل فرمون اضافه شده به مسیر i و j توسط مورچه‌ی k ام

L_k : طول مسیر طی شده توسط مورچه‌ی k ام

Q : عدد ثابت

ρ : نرخ تبخیر فرمون

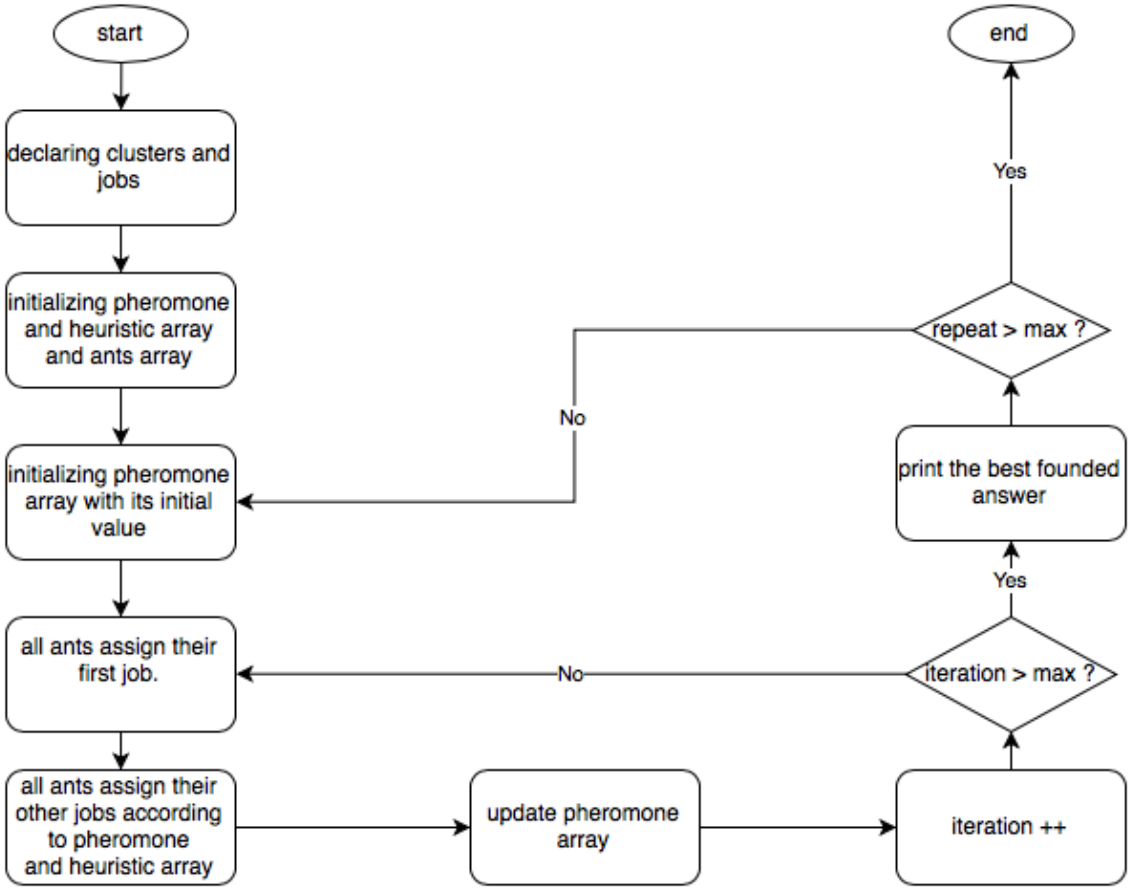
τ_{ij} : مقدار فرمون روی یال i و j

۳-۲-۲- نگاشت الگوریتم برای مسئله جایابی ماشین مجازی

هر مورچه یک راه حل است. هر راه حل با یک آرایه یک بعدی مدل شده است که نشان می دهد هر کار به کدام سرور اختصاص یافته است. به عنوان مثال اگر خانه ی i ام از این آرایه برابر j باشد نشان می دهد که کار i ام به سرور j ام اختصاص یافته است. کارها از ۰ تا (تعداد کارها - ۱) و سرورها از ۰ تا (تعداد سرورها - ۱) نامگذاری شده اند. بنابراین طول آرایه به تعداد کارها است و هر سلول دارای مقدار بین ۰ تا تعداد خوشه است. اگر یک خانه از این آرایه مقدار تعداد سرورها را داشته باشد بدان معنی است که کار i ام به هیچ یک از سرورها اختصاص نیافته است.

در اولین بار که الگوریتم اجرا می شود و مورچه ها می خواهند راه حل خود را پیدا کنند، آنها فقط بر اساس آرایه ی حریصانه حرکت می‌کنند که برای این منظور طراحی شده است که مورچه ها تلاش بیشتری در جادادن تمامی کارها انجام دهند و کاری بدون سرور نماند. پس از آنکه مورچه ها راه حل خود را پیدا کردند، با توجه به راه حل ها، ما شروع به تغییر در آرایه فرومون می کنیم. این بدان معنی است که برای هر راه حل به هر کار نگاه می کنیم که اگر شغل i به خوشه j اختصاص داده شود، باید یک عدد ثابت به سلول فرومون [j] [i] اضافه کنیم. پس از آن هر مورچه شروع به یافتن راه حل دیگری با توجه به فرومون و آرایه حریصانه می کند و این دنباله تکرار می شود و ما به یک حالت می رسیم که تمام مورچه ها همگرا می شوند.

۳-۲-۳- فلوچارت



۳-۲-۴- قسمت اصلی کد

```
while iteration < self.maxIterations:
    self.setupAnts()
    self.moveAnts()
    self.updateTrails()
    iteration += 1
```

۳-۳- الگوریتم بهینه سازی ABC

در مدل ABC ، کلنی متشکل از سه گروه زنبور عسل است: زنبورهای کارگر، تماشاچی و پیش آهنگ. فرض بر این است که تنها یک زنبور مصنوعی برای هر منبع غذایی وجود دارد. به عبارت دیگر، تعداد زنبورهای شاغل در کلنی برابر با تعداد منابع غذایی در اطراف کندو است. زنبورهای کارگر به منبع غذایی خود می روند و به کندو برمی‌گردند و به منطقه‌ی رقص می‌روند. زنبور کارگری که منبع غذایی آن از بین می‌رودن تبدیل به زنبور پیش آهنگ میشود و شروع به جستجو برای یافتن یک منبع غذایی جدید می کند. زنبورهای تماشاچی رقص زنبورهای کارگر را مشاهده کرده و غذای مورد نظر خود را انتخاب می‌کنند.

در ABC، الگوریتم مبتنی بر جمعیت، موقعیت یک منبع غذایی نشان دهنده یک راه حل ممکن برای مشکل بهینه سازی است و مقدار شهد یک منبع غذایی متناسب با ارزش آن راه حل است.

۳-۳-۱- گام های اصلی الگوریتم:

غذاهای اولیه برای تمام زنبورهای کارگر تولید می شود

- تکرار

- هر یک از زنبورهای کارگر به منبع غذایی در حافظه خود می‌رود و سپس نزدیکترین غذا به آن را تعیین می کند، سپس مقدار شهد آن را ارزیابی می کند، سپس به محل رقص در کندو برمیگردد

- هرزنبور تماشاگر، رقص زنبورهای کارگر را تماشا می کند و یکی از منابع را بسته به رقص انتخاب می‌کند و سپس به آن منبع می رود. پس از انتخاب یک همسایه اطراف آن، مقدار شهد خود را ارزیابی می کند.

- منابع غذای تمام شده تعیین می شوند و با منابع جدید که توسط زنبورهای پیش آهنگ کشف شده اند جایگزین می شوند.

- بهترین منبع غذایی یافت شده تا کنون ثبت می‌شود.

- تا (شرایط مورد نیاز)

۳-۳-۲- فرمول ها

$$p_i = \frac{\mathrm{fit}_i}{\sum_{n=1}^{\mathrm{SN}} \mathrm{fit}_n}, \qquad \text{فرمول ۶}$$

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}),$$ فرمول ۷

p_i : احتمال آنکه زنبور تماشاچی راه حل i ام را انتخاب کند.

fit_i : ارزش راه حل i ام

درباره ی فرمول ۷ :

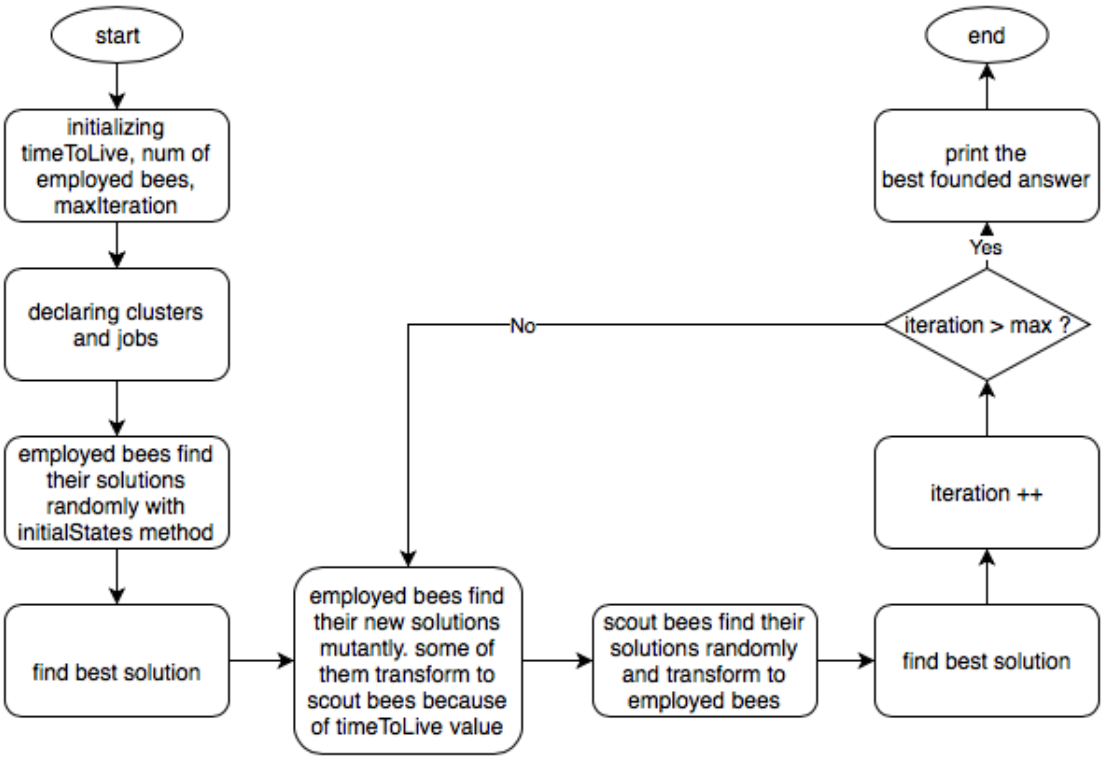
در هر چرخه، هر زنبور کارگر که به آخرین غذای در حافظه‌ی خود می رود، و از آنجا راه حل(غذا) ی دیگری با توجه به این فرمول پیدا میکند. اگر غذای دوم بهتر از غذای اول باشد زنبور کارگر آن را انتخاب می‌کند به جای غذای قبلی در غیر اینصورت غذای قبل را در حافظه نگه می‌دارد.

v_{ij} : یک تخصیص جدید برای i امین زنبور کارگر و j امین کار

x_{ij} : سرور قدیمی برای i امین زنبور و j امین کار

Φ : فاکتور تعیین کننده‌ی فاصله‌ی راه حل قدیم و جدید با استفاده از یادگیری ماشین

۳-۳-۳- فلوچارت



۳-۳-۴- قسمت اصلی کد

```
for i in range(maxIteration):
    empBeeNumBefore = len(employedBees)
    employedBees = problem.nextStates(employedBees)
    empBeeNumAfter = len(employedBees)
    scoutBees = empBeeNumAfter - empBeeNumBefore
    for j in range(scoutBees):
        employedBees.add(problem.randomState())
```

۳-۴- الگوریتم Firefly

هدف اولیه برای الگوریتم کرم شب تاب این است که هر کرم شب تاب به عنوان یک سیگنال برای جذب کرم شب تاب دیگر عمل می کند

این الگوریتم با فرض آن است که:

۱ .تمامی کرم های شب تاب یکدست هستند، به طوری که هر کرم شب تاب به تمام کرم شب تاب های دیگر جذب می شود

۲ .میزان جذب شدن متناسب با روشنایی آنها است، و برای هر دو کرم شب تاب، کرم کمرنگ تر به کرم روشن تر جذب می‌شود؛ با این حال، شدت (روشنایی ظاهری) با افزایش فاصله متقابل آنها کاهش می یابد

۳ .اگر هیچ کرم شب تاب روشن تر از یک کرم شب تاب وجود نداشته باشد، به طور تصادفی حرکت خواهد کرد. روشنایی باید با تابع هدف مرتبط باشد.

۳-۴-۱- گام ها ی اصلی الگوریتم

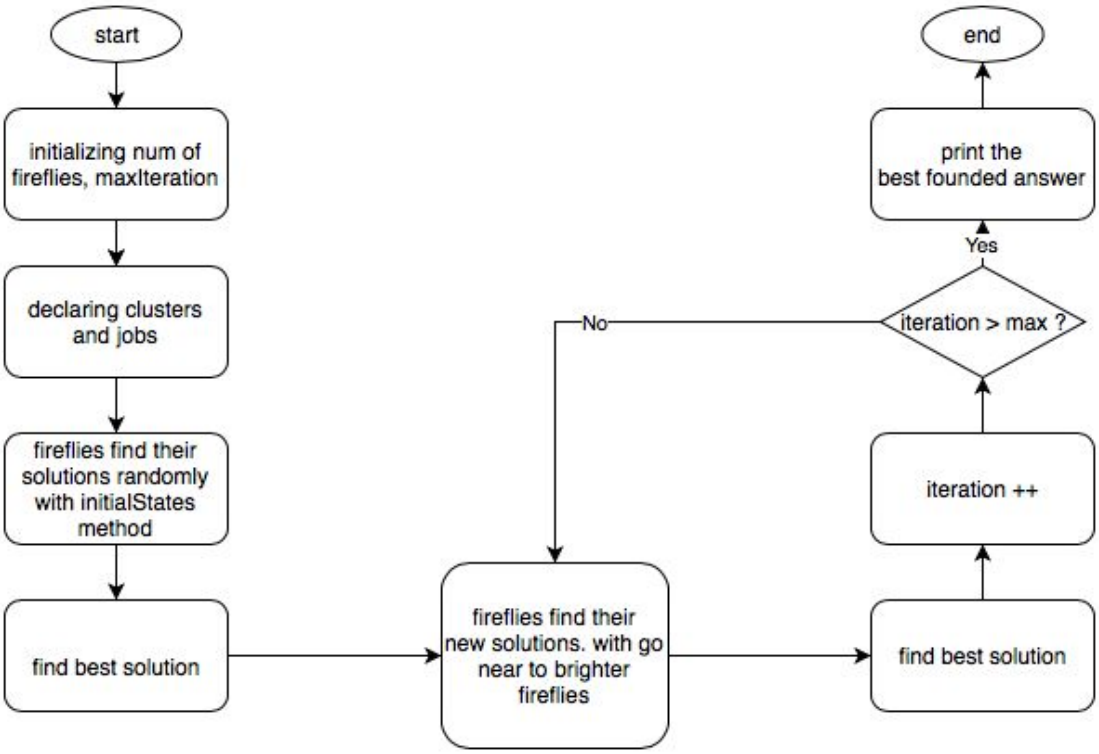
```
Begin
   1) Objective function:  f(x),   x = (x_1, x_2, ..., x_d) ;
   2) Generate an initial population of fireflies  x_i   (i = 1, 2, ..., n) ;.
   3) Formulate light intensity I so that it is associated with  f(x)
      (for example, for maximization problems,  I ∝ f(x)  or simply  I = f(x) ;)
   4) Define absorption coefficient γ

   While (t < MaxGeneration)
      for i = 1 : n (all n fireflies)
         for j = 1 : i (n fireflies)
            if (I_j > I_i),
               Vary attractiveness with distance r via exp(−γ r);
               move firefly i towards j;
               Evaluate new solutions and update light intensity;
            end if
         end for j
      end for i
      Rank fireflies and find the current best;
   end while

   Post-processing the results and visualization;

end
```

۳-۴-۲- فلوچارت



۳-۴-۳- قسمت اصلی کد

```
for k in range(maxIteration):
    for i in range(len(fireFlies)):
        for j in range(i):
            if problem.f(fireFlies[j]) > problem.f(fireFlies[i]):
                s = fireFlies[i].movefireFly(fireFlies[j], problem)
                fireFlies[i] = s
```

۳-۵- الگوریتم Tabu Search

جستجوی Tabu یک الگوریتم بهینه سازی است که شباهت زیادی به الگوریتم hill climbing دارد. اما دارای نکات کارآمدی است که نتیجه بهتری از الگوریتم hill climbing می‌دهد. به عنوان مثال مشکل hill climbing که در حالت‌های حداکثر محلی گیر می‌افتد را ندارد. در hill climbing ما وضعیت فعلی و همسایگان حالت فعلی را داریم. هر همسایه دارای ارزش است که با یک عدد مدل شده است، ما بهترین همسایه ای را انتخاب می کنیم که حداکثر مقدار ارزش را دارد. اما زمانی که تمام همسایگان ارزش بیشتری از حالت فعلی ندارند وضعیت فعلی را به عنوان بهترین راه حل ممکن باز می گردانیم. در صورتی که ممکن است که بهترین پاسخ در جای دیگر باشد.

Tabu Search سعی دارد این مشکل را حل کند و برخی از قوانین را برای بالا بردن کارایی ایجاد کرده است:

۱) یک لیست بسته ی محدود نگه می دارد تا به حالت های دیده شده نرود.

۲) اگر در آرایه ی راه حل، ۱ بیت در t سیکل گذشته تغییر کرده باشد اجازه ی تغییر آن بیت را نخواهیم داشت.

۳) اگر همه حالت‌های همسایه بدتر از وضعیت فعلی باشند اما حالت غیر مجازی طبق قانون ۲ وجود دارد که اگر به آن برویم از بهترین حالت تا به حالمان نیز بهتر میشویم آن حرکت را مجاز اعلام می‌کنیم، در غیر این صورت باید به بهترین همسایه برویم حتی اگر بدتر از حالت فعلی باشد.

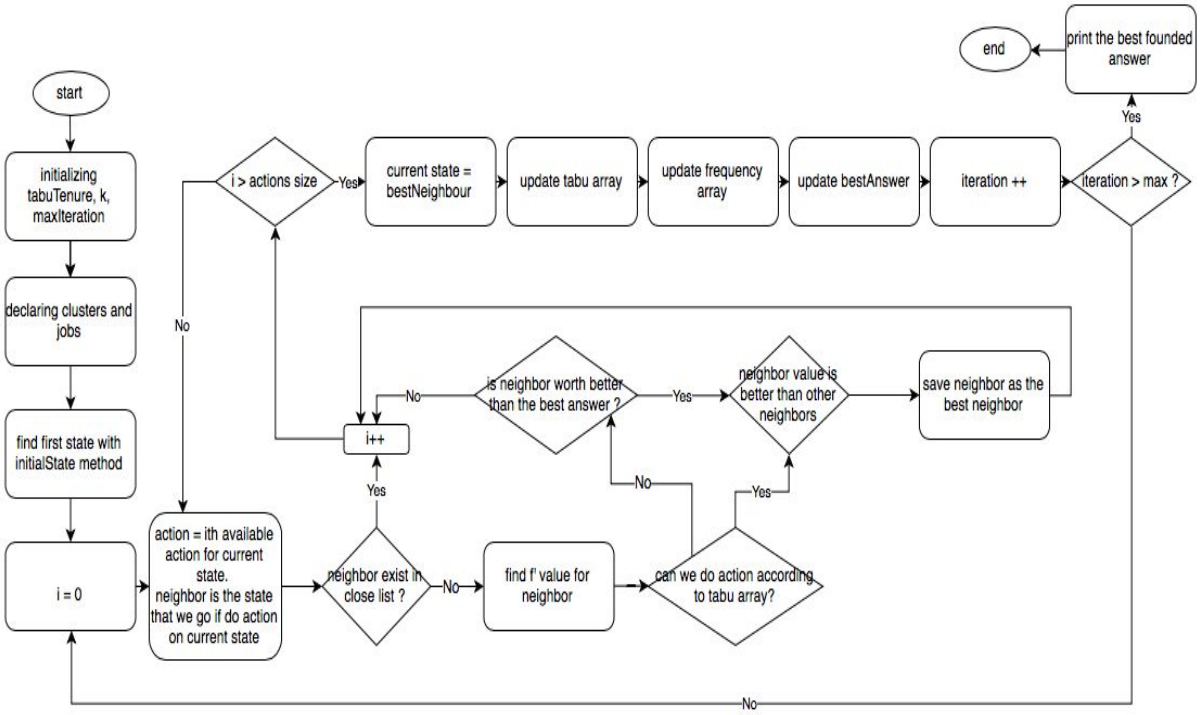
۴) یک نرخ فرکانس به الگوریتم اضافه می‌کنیم به طوری که هرچه یک بیت بیشتر تغییر کرده باشد احتمال تغییر آن را کمتر می‌کنیم. با استفاده از فرمول زیر :

$$f'(state) = f(state) - k * frequency[b_n]$$ فرمول ۸

که k یک عدد بین ۰ و ۱ است.

حالا به جای تابع f از تابع f’ برای ارزش گذاری راه حل ها استفاده می‌کنیم.

۳-۵-۱- فلوچارت



۳-۵-۲- قسمت اصلی کد

```
while maxIteration > 0:
    maxIteration -= 1
    actions = problem.actions(p)
    bestNeighbour = p
    bestAction = None
    maxCostFrequency = 0
    for i in range(len(actions)):
        neighbour = problem.result(p, actions[i])
        if problem.contains(myPath, neighbour):
            continue
        neighbourWorth = problem.f(neighbour)
        neighbourWorthFrequency = neighbourWorth - k * frequency[actions[i].jobIndex]
        observedNodes += 1
        if (tabu[actions[i].jobIndex] > 0 and neighbourWorth <= problem.f(best)):
            continue
        if (neighbourWorthFrequency > maxCostFrequency):
            maxCostFrequency = neighbourWorthFrequency
            bestNeighbour = neighbour
            bestAction = actions[i]

    if (bestNeighbour == p):
        break

    p = bestNeighbour
    tabu[bestAction.jobIndex] = tabuTenure
    for i in range(len(tabu)):
        if (tabu[i] > 0):
            tabu[i]-= 1
    frequency[bestAction.jobIndex] += 1
```

فصل چهارم : نتیجه گیری

پس از طراحی الگوریتم‌های بهینه‌سازی فوق، نیاز به آزمایش و تست دقیق الگوریتم‌ها با استفاده از داده‌های واقعی هستیم تا بتوانیم درستی و کیفیت عملکرد هر یک از آن‌ها را سنجیده، مقایسه و نهایتا الگوریتمی که بیشترین بازده را داراست به عنوان الگوریتم اصلی در پیاده‌سازی سرویس ارائه خدمت به ماشین مجازی توسط سرورهای موجود را انتخاب و جایگزین الگوریتم فعلی که بر پایه الگوریتم FFD است، کنیم.

یکی از تست‌های انجام شده به شرح زیر بوده که در ادامه نتایج حاصل از آن نیز آورده شده است:

سرورهای موجود :

IDID	#Core	#Ram
۰	۱۰۰۰	۶۴
۱	۱۰۰۰	۶۴
۲	۱۰۰۰	۶۴
۳	۱۰۰۰	۶۴
۴	۱۰۰۰	۶۴

درخواست های کاربران :

IDID	#Core	#Ram
۰	۱۲۰	۱
۱	۱۲۰	۱
. .	۱۲۰	۱
۹	۱۲۰	۱
۱۰	۱۵۰	۱
. .	۱۵۰	۱
۱۹	۱۵۰	۱
۲۰	۱۷۰	۱
. .	۱۷۰	۱
۲۹	۱۷۰	۱

سایر مقداردهی‌های اولیه برای هر یک از متغیرهای الگوریتم‌ها در جدول زیر ذکر شده است:

ACO	ABC	Firefly	Tabu
maxIteration = ۱۰۰ pheremoneInitialValue = ۲ numOfAnts = ٤	maxIteration = ۱۰۰۰ timeToLive = ۱۰۰ numOfEmployedBees = ٥۰	maxIteration = ۱۰۰ numOfFireflies = ٥۰	**maxIteration** = ۱۰۰ **tabuTenure** = ٥ **k** = ۰,۰۱

نتایج حاصل از اجرای الگوریتم‌ها با مقادیر فوق در شکل زیر نشان داده شده است:

```
Ant Colony!
Best Solution Cost: 4.0047999999999995
Best Solution: job[0] -> 3  job[1] -> 2 job[2] -> 2 job[3] -> 0 job[4] -

Best Solution Cost: 4.0927999999999995
Best Solution: job[0] -> 4  job[1] -> 2 job[2] -> 1 job[3] -> 2 job[4] -

Best Solution Cost: 4.0927999999999995
Best Solution: job[0] -> 4  job[1] -> 2 job[2] -> 1 job[3] -> 2 job[4] -
```

```
Bee Colony!
4.1482
( 3  0  1  3  4  4  1  0  1  3  3  0  0  1  2  0  0  3  0  1  1  2  2  2  2  2  3  3  4  1  )
```

```
FireFlies!
4.1482
( 2  1  1  1  0  2  2  3  0  3  2  1  4  0  1  0  0  0  2  0  2  4  1  1  2  3  4  4  4  4  )
```

```
Tabu Search!
Number of observed nodes : 3374
Number of extended nodes : 100
Path cost : 100
Final state worth : 4.1482
Final Answer : ( 0  4  2  4  4  0  3  0  2  3  2  2  4  2  2  2  4  0  1  0  4  4  0  1  0  1  1  1  1  3  )
```

اینکه از ۱۰۰ درصد توان سرورها استفاده نکرده‌اند اما در یک زمان توانسته‌اند تعداد بسیار بیشتری از ماشین‌های مجازی را سرویس دهی کنند که این امر می‌تواند به افزایش رضایت کاربران از خدمت داده‌شده بیانجامد. علاوه‌بر این می‌دانیم که توان مصرفی در این شرایط بسیار کمتر بوده و هزینه‌های ناشی از نگهداری سرورها، قبض برق و ... بسیار کمتر از حالت قبلی خواهد بود..

لذا می‌توانیم نتیجه بگیریم که استفاده از الگوریتم‌های بهینه‌سازی جدید، بسیار مقرون به‌صرفه خواهد بود و همچنین می‌توان به تعداد کاربر بیشتری با کیفیت و سرعت بالاتر و بدون اختلال در سیستم، سرویس‌دهی کرد.

همچنین با توجه به مشاهدات انجام گرفته الگوریتم Tabu Search نسبت به تمامی الگوریتم ها از نظر زمان مصرفی و یافتن جواب بهینه بهتر عمل می‌کند.

فهرست مطالب

[۱] E. Barlaskar, Y. Jayanta Singh, B. Issac

“Enhanced cuckoo search algorithm for virtual machine placement in cloud data centres”

[۲] M. Abdel-Basset, L. Abdle-Fatah, A. Kumar Sangaiah,

“An improved Lévy based whale optimization algorithm for bandwidth-efficient virtual

machine placement in cloud computing environment”

[۳] S. Salcedo-Sanz, J. Del Ser, I. Landa-Torres, S. Gil-López, J. A. Portilla-Figueras,

“The Coral Reefs Optimization Algorithm: A Novel Metaheuristic for Efficiently Solving

Optimization Problems”

[٤] X. Fu, Q. Zhao, J. Wang, L. Zhang, and L. Qiao,

“Energy-Aware VM Initial Placement Strategy Based on BPSO in Cloud Computing”

[٥] H. Izakian, B. Tork Ladani, A. Abraham, and V. SÁŠEL,

“A DISCRETE PARTICLE SWARM OPTIMIZATION APPROACH FOR GRID JOB

SCHEDULING”

[٦] A. Beloglazov, and R. Buyya,

"Adaptive Threshold-Based Approach for Energy-Efficient Consolidation of Virtual Machines in Cloud Data Centers"

[٧] M. Mareli, and B. Twala

"An adaptive Cuckoo search algorithm for optimisation"

[٨]Wikiversity, "Whale Optimization Algorithm," ٢٠١٨. [Online]. Available:

https://en.wikiversity.org/wiki/Whale_Optimization_Algorithm.

[٩] swarmintelligence, "PSO Tutorial," ٢٠٠٦, [Online]. Available:

http://www.swarmintelligence.org/tutorials.php

[١٠] youtube, "Learn Particle Swarm Optimization (PSO) in ٢٠ minutes," [Online]. Available:

https://www.youtube.com/watch?v=JhgDMAm-imI