پژوهش‌های نوین در تصمیم‌گیری
دوره ۳، شماره ۳، پاییز ۱۳۹۷

حل ابتکاری مسئله مکان‌یابی تسهیلات ظرفیت‌دار و راهکارهای تسریع آن

احمد مرادی[۱*]، علی ولی‌نژاد[۲]

۱- استادیار، گروه علوم کامپیوتر، دانشکده علوم ریاضی، دانشگاه مازندران، بابلسر، ایران
۲- استادیار، گروه علوم کامپیوتر، دانشکده علوم ریاضی، دانشگاه مازندران، بابلسر، ایران

دریافت: ۱۳۹۷/۲/۴ پذیرش: ۱۳۹۷/۴/۱۹

چکیده

این پژوهش به حل ابتکاری مسئله مکان‌یابی تسهیلات ظرفیت‌دار تک-منبع اختصاص یافته است. در این نوشتار، ابتدا یک روش جستجوی محلی برای حل مسئله معرفی شده و سپس سازوکار حریصانه آن با دیدگاه تبرید شبیه‌سازی‌شده ترکیب می‌گردد تا توان گریز از بهینگی موضعی را ایجاد کند. در ادامه، سازوکارهای هوشمندانه‌ای برای تسریع آن اتخاذ می‌شود تا فرآیند جستجو را کارآمد سازد. سازوکارهای تسریع یادشده شامل الگوریتم‌هایی به‌منظور تمرکز جستجو بر بخش‌هایی از فضای جواب و نیز الگوریتم‌هایی به جهت موازی‌سازی فرآیند جستجو است. به‌کارگیری هر یک از این سازوکارها صورت کارآمدی از الگوریتم مذکور با ویژگی‌های محاسباتی متفاوت به دست خواهد داد. نتایج به‌دست‌آمده از به‌کارگیری روش فوق به روی نمونه‌های معتبر از مسئله مکان‌یابی تسهیلات تک-منبع، نه‌تنها از موفقیت آن در مقایسه با بهترین روش‌های حل موجود حکایت دارد، بلکه آن را به‌عنوان ابزاری عملی برای به‌کارگیری روی نمونه‌های دنیای واقعی مطرح خواهد ساخت.

واژگان کلیدی: مکان‌یابی تسهیلات؛ روش‌های ابتکاری؛ الگوریتم تبرید شبیه‌سازی‌شده؛ موازی‌سازی؛ ساختار همسایگی.

---

* نویسنده مسئول مقاله: E-mail: A.moradi@umz.ac.ir

۱- مقدمه

مسائل مکان‌یابی تسهیلات ازجمله مسائل مهم در بهینه‌سازی ترکیباتی هستند که کاربردهای بسیاری در صنایع مختلف یافته‌اند. در این رده از مسائل، اغلب مجموعه‌ای از مکان‌های مشخص برای نصب (بازگشایی) تسهیلات و همچنین مجموعه‌ای مشخص از مشتریان مفروض‌اند. خدمات‌دهی به مشتریان در این مسائل نتیجه دو سطح تصمیم‌گیری است. ابتدا در سطح تسهیلات، زیرمجموعه‌ای از مکان‌ها برای نصب تسهیلات انتخاب می‌شود و پس از بازگشایی تسهیلات در این مکان‌ها، در سطح مشتریان، هر مشتری به تسهیلات مشخصی تخصیص می‌یابد. در این رده از مسائل، متأثر از نوع کاربرد، محدودیت‌هایی به مسئله افزوده می‌شود. چنانچه روی هر یک از تسهیلات، ظرفیت محدودی در خدمات‌دهی به تقاضای مشتریان در نظر گرفته شود، مسئله، مکان‌یابی تسهیلات ظرفیت‌دار نامیده می‌شود. علاوه بر اعمال محدودیت ظرفیت، اگر هر مشتری مجبور به برآورده‌سازی تقاضای خود تنها از طریق یکی از تسهیلات شود، صورت مهمی از مسئله مکان‌یابی تسهیلات ظرفیت‌دار پدید می‌آید که مکان‌یابی تسهیلات ظرفیت‌دار تک-منبع نامیده می‌شود.

مسائل مکان‌یابی تسهیلات به جهت کاربردهای گسترده، توجه بسیاری از مجامع پژوهشی را به خود معطوف ساخته‌اند. دامنه‌ی کاربردهای این مسائل حوزه‌های مختلفی چون صنایع مخابرات، حمل‌ونقل، توزیع و حتی صنعت نفت را در بر گرفته است. خواننده علاقه‌مند می‌تواند فهرست کامل‌تری از این کاربردها را در مراجع [۵، ۴، ۳، ۲، ۱] بیابد. این پژوهش به مطالعه‌ی مسئله مکان‌یابی تسهیلات ظرفیت‌دار تک-منبع اختصاص یافته است. ازاین‌رو در ادامه، با تمرکز بر این مسئله به‌مرور مبانی نظری موجود می‌پردازیم.

مسئله مکان‌یابی ظرفیت‌دار تک‌منبع در گروه مسائل سخت است [۶]. از پژوهش‌های انجام‌شده به‌منظور حل دقیق مسئله می‌توان به منابع [۸، ۷، ۶] اشاره کرد. مقاله [۶] از یک چهارچوب شاخه و کران و مقاله [۷] از یک روش تولید ستون برای حل مسئله بهره جسته‌اند. مقاله [۸] به مطالعه‌ی تأثیر افزودن برش‌هایی خاص به مدل مسئله در یک چهارچوب برش و حل پرداخته است. از ایده‌های ابتکاری موفق برای حل مسئله می‌توان به ترکیب روش آزادسازی لاگرانژ و روش کلونی مورچگان یا ترکیب آن با روش جستجوی ممنوعه اشاره کرد. برای مروری کامل بر چنین

روش‌هایی خواننده به منبع [۹] ارجاع داده می‌شود. در مقاله‌ی [۱۰] یک ساختار همسایگی گسترده در یک الگوریتم جستجوی محلی بکار گرفته شده است. در [۱۱] روش جستجوی ممنوعه و ایده‌های متأثر از آن، ایفاگر نقش اصلی هستند. مهم‌تر آنکه برتری محاسباتی چه ازنظر زمان اجرا و چه ازنظر کیفیت جواب‌های به‌دست‌آمده، برای الگوریتمی موسوم به جستجوی ممنوعه تکراری گزارش شده است [۱۲]. این الگوریتم به‌عنوان مرجع مقایسه‌ها در نتایج به‌دست‌آمده از پژوهش حاضر در نظر گرفته شده است. روش جستجوی ممنوعه تکراری، به‌اختصار ITS، در هر تکرار از اجرای خود الگوریتم جستجوی ممنوعه را با یک ساختار همسایگی ترکیبی به خدمت می‌گیرد. جواب‌های جستجو شده در این روش لزوماً شدنی نبوده و درنتیجه، در محاسبه کیفیت آن‌ها اندازه‌ی نشدنی‌بودن در نظر گرفته می‌شود.

نتایج حاصل از آزمایش‌های گسترده در این پژوهش، به برتری سازوکار جستجوی روش تبرید شبیه‌سازی‌شده در زمان اجرا، کیفیت جواب‌های شدنی یافته‌شده و حتی در قابلیت اطمینان بیشتر به این جواب‌ها، در مقایسه با نتایج حاصل از روش ITS، اشاره دارد.

نوآوری‌های انجام‌گرفته در این پژوهش به‌اختصار به شرح زیر است:

۱- به‌کارگیری سازوکار آشفته‌سازی الگوریتم تبرید شبیه‌سازی‌شده بر روی یک ساختار جستجوی محلی موجود از مسئله و تحلیل محاسباتی عملکرد آن در مقایسه با الگوریتمی مبتنی بر ترکیب جستجوی محلی یادشده و ایده جستجوی ممنوعه تکراری؛

۲- به‌کارگیری سازوکارهای مبتنی بر الف) تمرکز جستجو و ب) موازی‌سازی جستجو روی الگوریتم‌های فوق به‌منظور تسریع زمان اجرای آن؛

۳- تحلیل محاسباتی گسترده از الگوریتم‌های یاد شده و صورت‌های متمرکز و موازی آن‌ها به روی نمونه‌های استاندارد از مسئله.

در ادامه این نوشتار و در بخش دوم، به بیان دقیق مسئله مکان‌یابی تسهیلات تک-منبع در قالب یک مدل برنامه‌ریزی صحیح خواهیم پرداخت. بخش سوم، به معرفی روش حل مسئله بر مبنای تبرید شبیه‌سازی‌شده، روش‌های تسریع و بحث در جزئیات پیاده‌سازی آن پرداخته است و بخش چهارم، به شرح محاسبات انجام‌شده و نتایج حاصل از آن در مقایسه با دو الگوریتم یادشده اختصاص یافته است. سرانجام،

بخش آخر به نتیجه‌گیری و بیان پیشنهادهای آتی پرداخته است.

۲- فرمول‌بندی مسئله

برای مسئله‌ی مکان‌یابی ظرفیت‌دار تک-منبع، مجموعه مشتریان را با نماد I و مجموعه مکان‌های ممکن برای نصب و بازگشایی تسهیلات را با نماد J نشان دهید. در این صورت برای هر مشتری چون $i \in I$، مقدار تقاضا با مولفه نامنفی d_i بیان می‌شود. برای مکان $j \in J$، مولفه‌های نامنفی f_j و s_j، به ترتیب هزینه‌ی بازگشایی تسهیلات و ظرفیت آن در این مکان را نشان می‌دهند. علاوه بر موارد فوق، تخصیص مشتری i به تسهیلات بازگشایی‌شده در مکان j هزینه‌ی c_{ij} را به دنبال خواهد داشت.

مسئله‌ی مکان‌یابی ظرفیت‌دار تک-منبع می‌تواند در قالب مدل برنامه‌ریزی خطی دودویی زیر بیان شود:

$$\min \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{i\in I} f_i y_i \qquad (۱)$$

$$s.t \quad \sum_{j\in J} d_j x_{ij} \le s_i y_i, \quad \forall i \in I \qquad (۲)$$

$$\sum_{i\in I} x_{ij} = 1, \qquad \forall j \in J \qquad (۳)$$

$$y_i \in \{0,1\}, \qquad \forall i \in I \qquad (۴)$$

$$x_{ij} \in \{0,1\}, \qquad \forall i \in I, \forall j \in J \qquad (۵)$$

که در آن متغیر دودویی x_{ij} درصورتی‌که مشتری i به مکان j تخصیص یابد

مقدار یک و در غیر این صورت، مقدار صفر خواهد داشت. به‌طور مشابه، متغیر دودویی y_j درصورتی‌که مکان j برای نصب تسهیلات بازگشایی شده باشد، مقدار یک و در غیر این صورت، مقدار صفر خواهد داشت.

تابع هدف در این مدل به‌صورت مجموع هزینه‌های اتصال مشتریان به مکان‌ها و نیز هزینه‌های بازگشایی تسهیلات در مکان‌های ممکن، تعریف شده است. قید ۲ بیانگر محدودیت ظرفیت روی مکان jام و قید ۳ بیانگر محدودیت خدمات‌دهی به مشتری i ام تنها از یک مکان هستند. درنهایت، قیدهای ۴ و ۵ شرط دودویی بودن متغیرهای تصمیم را بیان می‌دارند.

در دوتایی (x, y)، بردارهای $y \in \{0,1\}^{|J|}$ و $x \in \{0,1\}^{|I|\times|J|}$ را در نظر بگیرید. با توجه به مدل، (x, y) یک جواب شدنی از مسئله خواهد بود، اگر و فقط اگر در قیدهای ۲ و ۳ صدق کند. درصورتی‌که دوتایی (x, y) تنها قید ۳ را برآورد، یک شبه‌جواب از مسئله نامیده می‌شود. بدیهی است که یک شبه‌جواب، تخصیصی از مشتریان به مکان‌هاست که ممکن است محدودیت ظرفیت روی مکان‌ها را نقض کند.

۳- جستجو در فضای جواب

ساخت یک روش برای جستجوی محلی در فضای جواب مسئله، پیش از هر چیز به تعریف یک ساختار همسایگی مناسب و تعیین سازوکار حرکت از یک جواب به جواب همسایه، نیازمند است. مفاهیم زیر را که در بسیاری از تحقیقات مرتبط با مسئله و ازجمله در مرجع [۱۲] بکار گرفته‌شده‌اند، در نظر بگیرید.

حرکت نوع اول؛ یک جواب از مسئله و مشتری i_1 از آن را در نظر بگیرید. فرض کنید مشتری i_1 در این جواب به مکان j_1 تخصیص‌یافته است. تخصیص i_1 به مکان متفاوت j_2، یک حرکت از نوع اول روی این جواب تعریف می‌کند. این حرکت می‌تواند در صورت لزوم به بازگشایی تسهیلات در مکان j_2 یا بستن تسهیلات در j_1 بیانجامد. توجه می‌دهیم که اعمال یک حرکت از نوع اول، لزوماً قید محدودیت را حفظ نمی‌کند و همسایه به‌دست‌آمده از این روش یک شبه‌جواب خواهد بود.

حرکت نوع دوم؛ یک جواب از مسئله و دو مشتری i_1 و i_2 از آن را که به ترتیب به مکان‌های متفاوت j_1 و j_2 تخصیص یافته‌اند، در نظر بگیرید. تخصیص مشتری i_1 به j_2 و مشتری i_2 به j_1 یک حرکت از نوع دوم روی این جواب تعریف می‌کند.

توجه می‌دهیم که اعمال یک حرکت از نوع دوم لزوماً قید محدودیت را حفظ نمی‌کند و همسایه به‌دست‌آمده یک شبه‌جواب خواهد بود.

اکنون برای یک شبه‌جواب داده‌شده (x,y)، مجموعه شبه‌جواب‌های حاصل از انجام یک حرکت از نوع اول روی (x,y) را همسایگی نوع اول آن بنامید و آن را با $N_1(x,y)$ نشان دهید. به‌طور مشابه، مجموعه شبه‌جواب‌های حاصل از انجام یک حرکت از نوع دوم روی (x,y) را همسایگی نوع دوم آن بنامید و آن را با $N_2(x,y)$ نشان دهید. درصورتی‌که (x,y) یک جواب شدنی از مسئله باشد، می‌توان تنها حرکت‌هایی از نوع اول یا نوع دوم را در نظر گرفت که محدودیت ظرفیت را حفظ کنند. در این صورت همسایگی‌های حاصل را به ترتیب با $F_1(x,y)$ و $F_2(x,y)$ نشان دهید. بدیهی است که $F_1(x,y)$ و $F_2(x,y)$ زیرمجموعه‌هایی از فضای شدنی مسئله هستند. در ادامه این پژوهش، واژه‌های «جواب» و «شبه‌جواب» را به‌طور معادل بکار خواهیم گرفت. در مقابل، واژه «جواب شدنی» را برای تأکید بر شدنی‌بودن یک جواب بکار خواهیم برد.

فرض کنید برای یک جواب چون (x,y) حرکت به یک همسایه چون (x',y')، تغییری معادل Δ در تابع هدف ایجاد کند. تغییر ایجادشده در تابع هدف تنها زمانی یک معیار کارآمد در ارزیابی جواب‌های همسایه است که حرکت‌های انجام‌شده محدود به جستجو در $F_1(.)$ و $F_2(.)$ گردد. درصورتی‌که جستجو در فضای شبه‌جواب‌ها انجام شود، یک ارزیابی کارآمد می‌بایست میزان نشدنی بودن شبه‌جواب‌ها را نیز در نظر بگیرد. برای شبه‌جواب (x,y)، میزان نشدنی بودن را با توجه به قید محدودیت ظرفیت برابر با $\sum_{j\in J}\max\left(0,-s_j y_j+\sum_{i\in I} d_i x_{ij}\right)$ تعریف کنید. اکنون فرض کنید که حرکت از (x,y) به همسایه (x',y')، تغییری معادل δ در میزان نشدنی‌بودن ایجاد کند. یک ارزیابی کارآمد از جواب (x',y') با محاسبه‌ی کمیت $\Delta+\alpha\delta$ به دست خواهد آمد. در محاسبه این کمیت، مولفه α، اهمیت عامل δ را کنترل خواهد کرد. این کمیت را مطلوبیت حرکت از (x,y) به (x',y') بنامید. اگر مطلوبیت حرکت منفی باشد، $\Delta+\alpha\delta<0$، حرکت از (x,y) به (x',y')، مطلوب و در غیر این صورت، حرکتی نامطلوب ارزیابی خواهد شد.

با وجود یک ساختار همسایگی و یک روش ارزیابی مناسب از حرکت، روش جستجوی محلی زیر بلافاصله قابل‌طرح خواهد بود:

الگوریتم۱: جستجوی محلی

گام ۱- یک جواب آغازین یافته و آن را به‌عنوان جواب جاری، (x,y)، در نظر بگیرید.

گام۲- اگر حرکتی مطلوب در $N_1(x,y)\cup N_2(x,y)$ یافت شود،

(x,y) را با اعمال این حرکت به‌روز کنید و به ابتدای گام ۲ بروید.

در غیر این صورت،

با معرفی جواب جاری به جستجو خاتمه دهید.

زمان اجرای یک روش جستجوی محلی معمولاً متأثر از سایز همسایگی آن است. در عمل، به‌منظور کنترل زمان اجرا، تعداد حرکت‌های جستجوشده در این روش را به مقدار مشخصی چون L محدود می‌کنند. درهرصورت، روند حریصانه بکار گرفته‌شده در انتخاب حرکت‌های مطلوب، به‌ناچار آن را به جواب‌های بهینه محلی همگرا خواهد ساخت. در ادامه این پژوهش، ایده تبرید شبیه‌سازی‌شده را برای خروج از جواب‌های بهینه‌های محلی به کار خواهیم برد.

۳-۱- روش تبرید شبیه‌سازی‌شده

شبیه‌سازی فرآیند تبرید در علم ترمودینامیک بر روی یک مسئله بهینه‌سازی ترکیباتی، به ابداع روشی توانمند موسوم به تبرید شبیه‌سازی‌شده انجامیده است [۱۴، ۱۳]. در این شبیه‌سازی، معمولاً مفاهیم آرایش مولکولی ماده و میزان پایداری آن با مفهوم جواب یک مسئله بهینه‌سازی ترکیباتی و میزان مطلوبیت آن هم‌ارز گرفته می‌شود. علاوه بر آن، تأثیر دما در فرآیند تبرید، با معرفی یک معیار احتمالی در پذیرش حرکت‌های جستجوشده، شبیه‌سازی می‌شود. در ادامه، روش تبرید شبیه‌سازی‌شده را برای حل مسئله مکان‌یابی تسهیلات ظرفیت‌دار تک-منبع بکار خواهیم گرفت.

روش جستجوی محلی بیان‌شده در قسمت قبل و مولفه کنترلی دما (t) را در نظر بگیرید. همچنین فرض کنید (x,y) جواب جاری و حرکت از (x,y) به (x',y') دارای مطلوبیت $\Delta+\alpha\delta$ باشد. متأثر از مولفه t، این حرکت را با احتمال $exp\left\{-\frac{1}{t}max(0,\Delta+\alpha\delta)\right\}$ پذیرفته و جایگزین جواب جاری کنید. این قاعده‌ی احتمالی

همواره حرکت‌های مطلوب را می‌پذیرد و در مقابل، احتمال پذیرفتن یک حرکت نامطلوب را به کنترل مولفه t در می‌آورد. روند تکراری بالا وقتی به جستجوی L حرکت محدود شود، به‌اختصار حلقه‌ی جستجو در دمای t بنامید.

الگوریتم تبرید شبیه‌سازی‌شده با شروع از یک جواب آغازین، تکرارهای هوشمندانه‌ای از حلقه‌ی جستجو در دماهای متفاوت تشکیل می‌دهد. پس از یک‌بار اجرای حلقه جستجو، مولفه t طی سازوکاری موسوم به طرح‌ریزی دما کاهش می‌یابد و اجرایی دیگر از حلقه جستجو در دمای جدید آغاز می‌شود. با کاهش تدریجی دما، احتمال پذیرفتن حرکت‌های نامطلوب رفته‌رفته کاهش می‌یابد. به دنبال آن، الگوریتم با مشاهده مقادیر به‌اندازه کافی کوچک t، تکرار مجدد حلقه جستجو را بی‌فایده قلمداد کرده و خاتمه می‌یابد. طرح‌ریزی دمای بکار گرفته‌شده در این الگوریتم، یک طرح‌ریزی هندسی است؛ به این معنی که مقدار t با شروع از مقدار اولیه t_{max}، پس از هر بار اجرای حلقه جستجو به‌صورت $t := c_f t$ کاهش می‌یابد که در آن عامل تبرید c_f، عددی ثابت از بازه‌ی $(0,1)$ است. روند کاهش دما تا دست‌یافتن به مقدار مشخص t_{min} ادامه می‌یابد.

علاوه بر طرح‌ریزی دما، الگوریتم فوق از طرح‌هایی برای به‌روزرسانی جواب جاری، بلافاصله قبل از شروع یک اجرای جدید از حلقه جستجو، بهره می‌گیرد. یک طرح ساده از به‌روزرسانی جواب جاری می‌تواند انتخاب جواب حاصل از حلقه جستجوی ماقبل به‌عنوان جواب جاری جدید باشد. طرح دیگر در به‌روزرسانی، جایگزینی مطلوب‌ترین جواب سراسری یافته‌شده به‌جای جواب جاری است. طرح دیگر جایگزینی جواب جاری با صورت آشفته‌ساخته یکی از طرح‌های قبل است؛ به این معنی که قبل از جایگزینی یک جواب در جواب جاری، حرکت‌های محدودی برای آشفته‌سازی روی آن اعمال خواهد شد. طرح دیگر، به‌کارگیری ترکیبی تصادفی از طرح‌های قبل در به‌روزرسانی جواب جاری است.

توضیح درباره چگونگی یافتن جواب آغازین، روش محاسبه مقادیر t_{min} و t_{max}، و طرح بکارگرفته‌شده در به‌روزرسانی جواب جاری به بخش ۳-۳ موکول می‌شود.

الگوریتم تبرید شبیه‌سازی‌شده فوق برای حل مسئله مکان‌یابی تسهیلات ظرفیت‌دار را به‌اختصار SA بنامید. در ادامه این نوشتار، صورتی از الگوریتم SA که طی آن هر حلقه جستجو حرکت‌های ممکن از $N_1(.) \cup N_2(.)$ را در نظر می‌گیرد، با $SA\text{-}N$ و

صورتی از آنکه هر حلقه جستجو تنها حرکت‌های ممکن از $F_1(.) \cup F_2(.)$ را بیازماید، با $SA\text{-}F$ نشان داده خواهد شد. در بخش بعد سازوکارهایی را بررسی می‌کنیم که به بهبود زمان اجرای این الگوریتم در عمل می‌انجامند.

۳-۲- روش‌های تسریع سازوکار جستجو

آن‌چنان‌که پیش‌تر اشاره شد، تکرارهای فراوان حلقه جستجو که در الگوریتم SA پیش‌بینی‌شده، تأثیر نامطلوبی در زمان اجرای آن دارد. یک ایده اساسی در چنین شرایطی، هدفمند کردن اجرای حلقه‌های جستجوست. در عمل، درصورتی‌که بتوان اجرای یک حلقه جستجو را بر قسمت خاصی از فضای جواب متمرکز کرد، اغلب به تعداد کمتری حرکت برای یافتن جواب‌های مناسب در آن قسمت نیاز خواهد بود. با این هدف، مفاهیم آشفته‌سازی روی یک جواب چون (x,y) را به شرح زیر در نظر بگیرید.

آشفته‌سازی نوع اول؛ مکان jام که $y_j = 1$ و مجموعه همه مشتریان متصل به آن، $I_j = \{i \in I \mid x_{ij} = 1\}$ در نظر بگیرید. اکنون جواب (x',y') را از روی (x,y) با بستن مکان jام و تخصیص مشتریان آن (I_j) به دیگر مکان‌های بازگشایی‌شده بسازید.

آشفته‌سازی نوع دوم؛ مکان jام که $y_j = 0$ و زیرمجموعه دلخواهی از مشتریان چون I'_j را در نظر بگیرید. اکنون جواب (x',y') را از روی (x,y) با بازگشایی مکان jام و تخصیص مشتریان موجود در I'_j به مکان jام بسازید.

برای جواب جاری (x, y) و دمای داده‌شده‌ی t، حلقه جستجو را می‌توان به‌گونه‌ای خاص و با تمرکز بر روی آشفته‌سازی ایجادشده اجرا کرد. برای این منظور، قبل از شروع به اجرای حلقه جستجو، جواب (x',y') با آشفته‌سازی جواب جاری تولید و سپس حلقه جستجو را با شروع از (x',y') به شکل محدودشده زیر اجرا کنید:

– درصورتی‌که آشفته‌سازی اعمال‌شده در ساخت (x',y') از نوع اول باشد، حلقه جستجو تنها حرکت‌هایی را بررسی می‌کند که پس از اعمال آن‌ها مکان خاص j بسته بماند.

– درصورتی‌که آشفته‌سازی اعمال‌شده در ساخت (x',y') از نوع دوم باشد، حلقه جستجو تنها حرکت‌هایی را بررسی می‌کند که پس از اعمال آن‌ها مکان خاص j

بازگشایی‌شده باقی بماند.

روند فوق، الگوریتم SA را قادر خواهد ساخت تا در یک اجرا از حلقه جستجو تنها مطلوبیت جواب‌هایی را بررسی کند که در مکان آشفته‌سازی‌شده ساختار یکسانی دارند. در عمل، آن‌چنان‌که در بخش ۴ روشن‌تر خواهد شد، با متمرکز ساختن یک حلقه جستجو، تعداد حرکت‌های ایجادشده تا جواب‌های مناسب کاهش می‌یابد. تأکید بر این نکته ضروری است که در صورت به‌کارگیری یک طرح به‌روزرسانی از جواب جاری، روند آشفته‌سازی فوق اعمال نخواهد شد. در این صورت، حلقه جستجو اجرایی عادی و نه متمرکز خواهد داشت. جزئیات بیشتر از طرح‌های به‌روزرسانی و روش‌های آشفته‌سازی جواب جاری در بخش ۳-۳ به بحث گذاشته خواهد شد.

ایده اساسی دوم در تسریع الگوریتم SA اجرای موازی یک حلقه جستجو از آن است. در این راستا و به فرض وجود n پردازنده، اجرای یک حلقه جستجو به طول L در دمای t و با شروع از جواب جاری (x,y)، می‌تواند روی هر پردازنده با روند زیر جایگزین شود:

- جواب (x',y') را با آشفته‌سازی (x,y) تولید کنید. سپس یک حلقه جستجو با طول محدودشده $\left\lfloor \frac{L}{n} \right\rfloor$ را با شروع از (x',y') و در دمای t اجرا کنید.

توجه می‌دهیم که در روند فوق، به‌منظور ارزیابی مستقل ایده موازی‌سازی، حلقه‌های جستجو روی هر پردازنده بدون اعمال تمرکز بر آشفتگی اجرا می‌شوند. بحث در جزئیات ایده موازی‌سازی به بخش ۳-۳ موکول می‌شود.

۳-۳- جزئیات پیاده‌سازی

در بخش‌های قبل، چارچوب کلی روش تبرید شبیه‌سازی در حل مسئله مکان‌یابی تسهیلات ظرفیت‌دار تک-منبع به بحث گذاشته شد. در این بخش جزئیاتی از قبیل روش انتخاب جواب آغازین، چگونگی تخمین مولفه‌های t_{min} و t_{max}، طرح‌های بکار گرفته‌شده در آشفته‌سازی یا به‌روزرسانی جواب جاری و روش موازی‌سازی حلقه جستجو موردبحث قرار خواهد گرفت.

در اجرای الگوریتم تبرید شبیه‌سازی‌شده، هنگامی‌که جستجو روی شبه‌جواب‌ها انجام می‌شود یک جواب آغازین مناسب، شبه‌جواب ساخته‌شده به روش زیر است:

– دنباله $\left(\frac{f_j}{s_j}\right)_{j\in J}$ را به ترتیب صعودی مرتب کنید. از روی ترتیب به‌دست‌آمده مکان‌ها را بازگشایی کنید تا جایی که مجموع ظرفیت بازگشایی‌شده از مجموع تقاضای مشتریان بیشتر شود. در ادامه، هر مشتری را به نزدیک‌ترین مکان بازگشایی‌شده تخصیص دهید.

معیار نزدیک‌بودن در بحث فوق، هزینه اتصال مشتری به یک مکان است. با به‌کارگیری روش فوق شبه‌جوابی حاصل می‌شود که هزینه آن، کران پایینی از بهینگی است. انتخاب جواب آغازین به این شکل، برای استفاده از الگوریتم ITS، در مرجع [۱۲] گزارش شده است. روند فوق بحث در چگونگی ساخت یک جواب شدنی آغازین را کامل می‌کند.

به‌منظور مقداردهی مناسب به مولفه‌های t_{min} و t_{max}، الگوریتم تبرید شبیه‌سازی‌شده تعداد محدودی جواب تصادفی از مسئله، تولید و طی آن متوسط مطلوبیت یک جواب را برآورد می‌کند. در ادامه، با استفاده از این برآورد، مولفه t_{max}، مقداری می‌یابد که احتمال پذیرفتن یک جواب با مطلوبیت متوسط، در ابتدای الگوریتم ($t \approx t_{max}$) نزدیک به ۱ باشد. به‌طور مشابه، مولفه t_{min} مقداری می‌یابد که احتمال پذیرفتن یک جواب با مطلوبیت متوسط، در انتهای الگوریتم ($t \approx t_{min}$) نزدیک به صفر باشد.

در الگوریتم SA-N، حرکت انتخاب‌شده در یک گام از حلقه جستجو، حرکتی است که به جوابی شدنی با هزینه کمتر از بهترین جواب شدنی ثبت‌شده منجر شود یا هر حرکتی که مطلوب ارزیابی شود. در صورت عدم موفقیت در یافتن چنین حرکتی، تکرار مذکور از حلقه جستجو، حرکتی که کمترین افزایش را به مطلوبیت جواب جاری تحمیل کند، برمی‌گزیند. درصورتی‌که الگوریتم SA-N، حلقه‌های جستجوی تمرکزیافته را اجرا کند، آشفته‌سازی اعمال‌شده قبل از اجرای یک حلقه جستجو، یکی از موارد زیر خواهد بود که به‌طور تصادفی انتخاب می‌شود. جواب جاری (x,y) را در نظر بگیرید،

– مکان بسته‌شده j در جواب جاری را به‌دلخواه باز کنید.

– مکان بازگشایی‌شده j که روی جواب جاری تنها به یک مشتری خدمت می‌دهد به‌دلخواه انتخاب کنید. مکان j را بسته و مشتری مذکور را به نزدیک‌ترین مکان بازگشایی شده و متفاوت با j تخصیص دهید.

– مکان بازشده j_1 و مکان بسته‌شده j_2 در جواب جاری را به‌دلخواه انتخاب کنید؛

به‌طوری‌که ظرفیت j_2 از مجموع تقاضای برآورده‌شده روی j_1 کمتر نباشد. در ادامه، مکان j_1 را بسته و مشتری‌های آن را پس از بازگشایی j_2 به آن تخصیص دهید.

– مکان بسته‌شده j_1 و مکان‌های بازگشایی‌شده j_2 و j_3 را به گونه انتخاب کنید که اولاً، پس از بازگشایی‌شدن j_1 و بستن j_2 و j_3 مجموع ظرفیت بازگشایی‌شده از مجموع تقاضای مشتریان کمتر نباشد. ثانیاً، تغییر ایجادشده در هزینه بازگشایی مکان‌ها پس از بازگشایی j_1 و بستن j_2 و j_3 کمینه گردد. در ادامه، مشتریان تخصیص‌یافته به j_2 و j_3 را به j_1 تخصیص دهید.

– مکان بازگشایی‌شده j_1 و مکان‌های بسته‌شده j_2 و j_3 را به‌گونه‌ای انتخاب کنید که اولاً، پس از بسته‌شدن j_1 و بازگشایی j_2 و j_3 مجموع ظرفیت بازگشایی‌شده از مجموع تقاضای مشتریان کمتر نباشد. ثانیاً، تغییر ایجادشده در هزینه بازگشایی مکان‌ها پس از بسته‌شدن j_1 و بازگشایی j_2 و j_3 کمینه گردد. در ادامه، مشتریان تخصیص‌یافته به j_1 را به مکان نزدیک‌تر از بین j_2 و j_3 تخصیص دهید.

هر یک از روش‌های فوق، حالتی خاص یا ترکیبی از آشفته‌سازی‌های نوع اول و دوم‌اند. آشفته‌سازی که به یکی از شیوه‌های فوق روی جواب جاری اعمال گردد، در اجرای بعدی از حلقه جستجو ثابت می‌ماند. الگوریتم SA-N علاوه بر اجرای متمرکز حلقه‌ی جستجو، در مواردی تصادفی به اجرای عادی آن نیز می‌پردازد. در این صورت، قبل از اجرای یک حلقه جستجو، طرح‌های آشفته‌سازی زیر روی جواب جاری بکار خواهند رفت. جواب جاری (x,y) را در نظر بگیرید،

– صرف‌نظر از تخصیص موجود و بدون ایجاد تغییر در مجموعه مکان‌های بازگشایی‌شده روی جواب جاری، هر مشتری را به‌طور تصادفی به یک مکانی بازگشایی‌شده تخصیص دهید.

– مجموعه مکان‌های بازگشایی‌شده روی جواب جاری را به‌تصادف به دو زیرمجموعه J_1 و J_2 به‌ترتیب از اندازه‌های $\left\lfloor \frac{|J|}{2} \right\rfloor$ و $\left\lceil \frac{|J|}{2} \right\rceil$ افراز کنید. مکان‌های موجود در J_2 را ببندید. سپس هر مشتری تخصیص‌یافته به مکانی از J_2 را به‌تصادف به مکانی از j_1 تخصیص دهید که ظرفیت کافی برای خدمت‌دهی به مشتری مذکور داشته باشد. درصورتی‌که چنین مکانی در J_1 یافت نشود، مشتری

مذکور به مکانی از J_1 که کمترین هزینه بازگشایی را داراست، تخصیص می‌یابد.

توجه به این نکته ضروری است که پس از به‌کارگیری یکی از طرح‌های به‌روزرسانی فوق، الگوریتم حلقه جستجو را به‌طور عادی اجرا خواهد کرد. روش‌های آشفته‌سازی و طرح‌های به‌روزرسانی بکار گرفته‌شده روی الگوریتم SA-N مشابه با روش ITS انتخاب شده‌اند.

در الگوریتم SA-F، حرکت انتخاب‌شده در یک گام از حلقه جستجو به‌تصادف حرکتی دلخواه از نوع اول یا دوم، یا حرکتی است که کمترین تغییر از نوع خود را در هزینه جواب شدنی جاری ایجاد کند. این الگوریتم حلقه‌های جستجو را به‌صورت متمرکز اجرا نکرده و در مقابل، طرح به‌روزرسانی زیر از جواب شدنی جاری را قبل از هر اجرای عادی از حلقه جستجو بکار خواهد گرفت.

– به‌تصادف، جواب شدنی جاری را با بهترین جواب شدنی یافته‌شده یا با جواب شدنی حاصل از اجرای حلقه جستجوی قبل، جایگزین کنید.

اجرای موازی حلقه جستجو، آن‌چنان‌که در بخش ۲–۳ بیان آن گذشت، به هر دو مدل معروف *OpenMP* و *MPI* قابل پیاده‌سازی است [۱۶، ۱۵]. در این پژوهش، مدل‌های پیاده‌سازی فوق برای هر یک از الگوریتم‌های SA-N، SA-F و ITS در نظر گرفته‌شده و با افزودن پسوندهای OMP- و MPI- به انتهای نام آن‌ها ارجاع می‌شود.

۴- محاسبات

در این بخش، آزمایش‌ها و محاسبات انجام‌شده به‌منظور ارزیابی الگوریتم‌های مبتنی بر روش تبرید شبیه‌سازی‌شده گردآوری شده است. در این راستا، کمیت‌های زمان اجرا، کیفیت جواب و قابلیت اطمینان در دستیابی به آن، برای هریک از روش‌های تبرید شبیه‌سازی‌شده و الگوریتم ITS و بر روی نمونه‌های استاندارد از مسئله مکان‌یابی تسهیلات تک–منبع، محاسبه‌شده و در مقایسه با یکدیگر گزارش شده است.

نمونه مسئله‌های در نظر گرفته‌شده در این بخش، نمونه‌هایی استاندارد برگرفته از مرجع [۱۲] هستند. این نمونه‌ها از دو مجموعه متفاوت تشکیل شده‌اند. مجموعه اول شامل ۵۷ نمونه مسئله است که تعداد مشتریان در آن‌ها از ۲۰ تا ۹۰ مشتری و تعداد مکان‌ها از ۱۰ تا ۳۰ مکان، متغیر است. روی این مجموعه، نمونه‌های یکسان ازلحاظ تعداد مکان و مشتری، در طبقه‌ای یکسان قرار گرفته و در طبقه‌های نمونه‌ای C_1،

C_2، ... و C_7 معرفی شده‌اند. مجموعه دوم شامل ۷۱ نمونه مسئله است که تعداد مشتریان در آن‌ها از ۵۰ تا ۲۰۰ مشتری و تعداد مکان‌ها از ۱۰ تا ۳۰ مکان، متغیر است. روی این مجموعه، به‌طور مشابه، نمونه‌های یکسان ازلحاظ تعداد مکان و مشتری، در طبقه‌ای یکسان قرار گرفته و در طبقه‌های نمونه‌ای C_8، C_9، C_{10} و C_{11} معرفی شده‌اند. بر روی هر نمونه مسئله، هر یک از الگوریتم‌های موردمقایسه، ۱۰ بار اجرا شده و سپس میانگین زمان اجرا، میانگین شکاف در هزینه بهترین جواب شدنی یافته‌شده از مقدار بهینه و انحراف معیار آن محاسبه و گزارش خواهد شد.

محاسبات این بخش بر روی یک پردازشگر از نوع Intel core i7 با سیستم‌عامل لینوکس با حافظه‌ی اصلی ۸ گیگابایت انجام شده است. در اجرای حلقه جستجو به‌صورت موازی، ۴ هسته این پردازشگر و در اجرای عادی، تنها یک هسته آن به کارگرفته شده است. الگوریتم ITS و الگوریتم‌های تبرید شبیه‌سازی‌شده و صورت‌های موازی‌سازی شده متناظر آن‌ها، همگی به زبان C پیاده‌سازی شده و در صورت درخواست از نویسندگان مقاله، در دسترس هستند.

آزمایش‌های انجام‌شده در این بخش در دو قسمت مطرح شده‌اند. در قسمت اول، الگوریتم‌های مذکور بدون اعمال روش‌های تسریع در سازوکار جستجو مورد مقایسه قرار گرفته‌اند. در مقابل در قسمت دوم، بررسی تأثیر روش‌های تسریع جستجو بر این الگوریتم‌ها، هدف اصلی خواهد بود.

۴-۱- جستجوی عادی

در این سری از آزمایش‌ها، حلقه جستجو از طول $L = 500$ برخوردار است. این مقدار، برای الگوریتم ITS که از مفهومی مشابه با حلقه جستجو در سازوکار جستجوی ممنوعه‌ی خود بهره می‌گیرد نیز اعمال خواهد شد. دیگر مولفه اثرگذار از این الگوریتم، تعداد حلقه‌های پیموده شده است. این مولفه در آزمایش‌های این بخش، مقدار ۲۰۰ را خواهد داشت. این مقدار دو برابر مقدار استاندارد آن در مرجع [۱۲] انتخاب شده است تا بر گستردگی جستجو در این الگوریتم تأکید شود. مولفه‌های دیگر بکار رفته در این الگوریتم، مولفه‌های استاندارد انتخاب‌شده در مرجع [۱۲] خواهند بود. در الگوریتم‌های تبرید شبیه‌سازی‌شده، برای مولفه‌های c_f، P_{min} و P_{max} به ترتیب مقادیر ۰/۹۹۹۸، ۰/۰۰۵ و ۰/۹۵ انتخاب شده‌اند. در این آزمایش‌ها، حلقه‌های

جستجوی عادی استفاده شده و تنها طرح‌های به‌روزرسانی، آن‌چنان‌که در بخش ۳-۲ موردبحث قرار گرفت، اعمال می‌گردد.

جدول ۱ نتایج اجرای عادی الگوریتم‌های ITS، SA-N و SA-F به تفکیک طبقه‌های نمونه‌ای

طبقه	SA-F			SA-N			ITS		
	Gap %	σ	*Time (sec.)*	*Gap %*	σ	*Time (sec.)*	*Gap %*	σ	*Time (sec.)*
C_1	۰/۳۳	۴/۳۲	۱۳/۶۰	۰/۳۵	۱۳/۸۲	۱۹/۵۷	۱/۶۲	۵۲/۳۴	۱۸/۸۵
C_2	۱/۰۷	۴۲/۸۰	۲۴/۸۱	۵/۸۶	۸۶/۹۷	۶۰/۸۶	۶/۶۷	۱۰۷/۱۶	۵۷/۴۸
C_3	۱/۵۷	۵۵/۷۶	۳۸/۳۱	۱۲/۳۵	۱۱۰/۱۱	۱۳۴/۲۵	۱۱/۴۴	۱۴۸/۳۹	۱۲۸/۴۲
C_4	۱/۸۸	۹۳/۸۹	۴۵/۸۲	۶/۰۲	۱۰۶/۸۲	۲۱۱/۴۹	۵/۸۳	۱۳۵/۹۱	۱۹۱/۶۷
C_5	۳/۹۲	۲۰۲/۸۷	۷۳/۴۰	۶/۹۹	۱۷۸/۱۰	۳۶۹/۲۷	۷/۲۵	۱۷۵/۰۲	۳۲۹/۸۲
C_6	۳/۰۵	۵۲/۰۵	۹۰/۰۲	۶/۱۳	۴۹۱/۲۳	۴۶۶/۲۵	۶/۱۸	۴۱۸/۶۸	۴۱۳/۱۴
C_7	۵/۸۲	۲۰۶/۳۲	۱۰۶/۲۳	۷/۶۱	۱۱۶/۹۴	۵۱۸/۶۷	۸/۰۱	۳۰/۹۰	۴۷۸/۰۳
C_8	۰/۲۶	۳۱/۱۰	۳۸/۳۵	۲/۶۱	۵۸/۶۵	۱۸۳/۲۱	۲/۹۲	۶۵/۴۲	۱۶۶/۶۶
C_9	۵/۳۸	۴۴۳/۸۶	۱۷۳/۹۵	۲۰/۱۲	۵۶۷/۸۰	۱۱۶۳/۲۶	۲۰/۸۲	۵۲۴/۰۰	۱۱۱۱/۴۹
C_{10}	۱۶/۴۱	۳۲۰/۵۸	۶۸/۶۸	۱۴/۸۹	۸۲/۴۵	۴۴۵/۲۲	۱۵/۲۰	۷۸/۸۱	۴۱۹/۷۷
C_{11}	۱۶/۰۸	۶۳۸/۸۳	۲۲۳/۱۷	۲۲/۶۸	۵۶۶/۹۲	۹۸۶/۵۹	۲۲/۶۳	۴۱۴/۷۷	۹۵۰/۳۷
میانگین	۵/۷۸	۲۲۰/۸۰	۸۹/۷۶	۱۰/۵۵	۲۳۳/۳۲	۴۶۷/۶۵	۱۰/۸۳	۲۰۷/۳۲	۴۴۰/۳۱

نتایج به‌دست‌آمده از این آزمایش‌ها در جدول ۱ گردآوری شده‌اند. در این جدول برای هر طبقه از نمونه مسائل و هر الگوریتم، میانگین درصد شکاف از مقدار بهینه (Gap)، میانگین انحراف معیار (σ) و نیز میانگین زمان صرف‌شده (Time) روی این طبقه از نمونه‌ها گزارش شده است. نتایج به‌دست‌آمده از جدول ۱ به برتری الگوریتم SA-F چه در کیفیت جواب و چه در زمان اجرا اشاره دارد. این الگوریتم روی اکثر طبقه‌های یادشده در زمان کوتاه‌تر به جواب‌های بهتری دست یافته است. روی این

طبقه‌ها، هر سه الگوریتم یادشده رفتار مشابهی ازنظر انحراف معیار نشان می‌دهند، اگرچه در این میان ITS از برتری ناچیزی برخوردار است.

۴-۲- تأثیر به‌کارگیری سازوکارهای تسریع

در این بخش، نتایج حاصل از به‌کارگیری ایده‌های موازی‌سازی و تمرکز حلقه‌های جستجو در تسریع هر یک از الگوریتم‌های ITS، SA-N و SA-F گردآوری شده است. نتایج به‌دست‌آمده از این آزمایش‌ها نشان خواهد داد که به‌کارگیری ایده‌های یادشده، بخشی اجتناب‌ناپذیر در دست‌یابی به یک جستجوی کارآمد روی مسئله و از طریق هر یک از الگوریتم‌های فوق است.

جدول ۲ نتایج حاصل از اجرای موازی الگوریتم‌های ITS، SA-N و SA-F بر اساس مدل OpenMP به تفکیک طبقه‌های نمونه‌ای

طبقه	SA-F-OMP			SA-N-OMP			ITS-OMP		
	Gap %	σ	*Time (sec.)*	*Gap %*	σ	*Time (sec.)*	*Gap %*	σ	*Time (sec.)*
C_1	۱/۲۱	۳۰/۵۸	۵/۶۰	۱/۳۹	۵۸/۳۸	۱/۵۷	۲/۵۶	۵۲/۷۷	۲/۲۱
C_2	۱/۷۶	۴۵/۵۸	۷/۱۱	۷/۷۴	۱۰۹/۷۷	۳/۵۳	۷/۵۰	۱۰۵/۶۳	۵/۵۳
C_3	۲/۷۱	۱۰۲/۶۰	۱۰/۶۶	۱۲/۹۶	۱۰۴/۶۴	۶/۴۸	۱۱/۵۳	۱۰۰/۶۵	۱۱/۲۰
C_4	۳/۰۱	۹۶/۷۰	۱۲/۹۸	۷/۲۸	۱۳۱/۷۲	۹/۲۳	۶/۸۷	۱۹۱/۹۸	۱۷/۳۶
C_5	۳/۵۰	۱۹۷/۴۴	۲۰/۱۲	۷/۲۲	۱۱۳/۲۹	۱۵/۰۷	۷/۹۱	۱۵۱/۹۱	۳۳/۵۴
C_6	۳/۱۲	۱۵۱/۰۷	۲۴/۳۳	۶/۲۴	۴۶۲/۵۷	۱۹/۱۹	۶/۰۱	۴۸۳/۱۵	۵۴/۲۷
C_7	۶/۰۷	۶۸۱/۴۸	۲۸/۹۰	۸/۲۴	۳۵۳/۸۳	۲۲/۶۶	۷/۹۲	۶۵/۴۹	۸۰/۹۲
C_8	۱/۳۸	۱۱۴/۹۶	۱۰/۹۸	۴/۰۵	۱۰۳/۲۴	۸/۱۳	۶/۸۰	۱۸۰/۳۱	۱۴/۹۶
C_9	۵/۱۰	۳۹۳/۶۹	۴۸/۰۳	۱۵/۰۸	۳۴۳/۱۱	۶۳/۷۹	۲۰/۹۵	۵۳۰/۵۲	۲۵۰/۱۰
C_{10}	۱۷/۱۰	۳۲۷/۹۳	۱۹/۶۹	۱۶/۰۴	۱۰۸/۶۶	۲۲/۴۲	۱۸/۰۲	۱۲۳/۹۵	۵۸/۲۸
C_{11}	۱۶/۴۳	۸۵۹/۰۷	۶۰/۳۰	۱۸/۴۵	۶۸۶/۶۷	۹۰/۱۹	۱۸/۱۴	۳۵۱/۴۲	۵۸۷/۹۳
میانگین	۶/۳۱	۲۹۸/۷۶	۲۴/۸۴	۱۰/۱۸	۲۴۵/۸۶	۲۸/۳۲	۱۱/۵۵	۲۳۲/۲۰	۱۲۷/۳۰

نتایج حاصل از موازی‌سازی هریک از الگوریتم‌های یادشده بر اساس مدل‌های

OpenMP و *MPI* به ترتیب در جدول‌های ۲ و ۳ گزارش شده است. مولفه‌های به‌کار گرفته‌شده در این آزمایش‌ها همانند با جستجوی عادی است؛ با این تفاوت که حلقه‌های جستجو در این آزمایش‌ها به‌صورت موازی اجرا شده‌اند. بهبود چشمگیر زمان اجرا، در مقایسه با جستجوی عادی، دستاوردی قابل‌انتظار از به‌کارگیری ایده‌های موازی‌سازی است. علاوه بر آن، نتایج به‌دست‌آمده از این جداول به برتری روش‌های مبتنی بر ایده تبرید شبیه‌سازی‌شده و به‌ویژه روش SA-F، چه ازنظر زمان اجرا و چه ازنظر کیفیت جواب، اشاره دارد. دراین‌بین، انحراف معیار به‌دست‌آمده برای هر الگوریتم نسبت به جستجوی عادی آن افزایش نشان می‌دهد.

جدول ۳ نتایج حاصل از اجرای موازی الگوریتم‌های ITS، SA-N و SA-F بر اساس مدل MPI به تفکیک طبقه‌های نمونه‌ای

طبقه	SA-F- MPI			SA-N- MPI			ITS- MPI		
	Gap %	σ	*Time (sec.)*	*Gap %*	σ	*Time (sec.)*	*Gap %*	σ	*Time (sec.)*
C_1	۰/۵۲	۱۳/۳۷	۶/۳۲	۰/۴۳	۳۰/۴۱	۶/۰۶	۱/۰۶	۴۸/۶۱	۵/۰۳
C_2	۰/۷۹	۳۸/۳۸	۱۱/۳۳	۵/۴۸	۶۶/۳۲	۱۷/۵۲	۷/۱۰	۹۸/۶۴	۱۳/۵۲
C_3	۱/۲۸	۴۷/۴۶	۱۷/۷۶	۱۰/۴۹	۱۳۰/۳۰	۳۷/۸۱	۱۲/۶۶	۹۲/۸۱	۲۶/۷۰
C_4	۱/۸۶	۹۲/۴۳	۲۱/۳۲	۵/۳۹	۸۴/۲۸	۵۸/۱۷	۶/۵۳	۱۸۰/۱۶	۳۷/۳۸
C_5	۳/۱۹	۲۰۸/۲۰	۳۵/۱۴	۶/۰۷	۲۷/۶۸	۱۰۱/۰۱	۶/۷۸	۱۰۰/۸۴	۴۶/۲۴
C_6	۲/۹۷	۷۳/۸۶	۴۳/۱۷	۴/۵۸	۱۵/۱۰	۱۲۷/۴۱	۶/۵۰	۴۳۳/۷۹	۴۰/۷۸
C_7	۵/۵۴	۵۸۹/۸۹	۵۱/۲۹	۷/۶۱	۶۸/۱۵	۱۴۴/۲۰	۸/۲۴	۲۲۴/۱۸	۴۵/۴۷
C_8	۰/۵۳	۵۱/۷۹	۱۱/۳۷	۲/۶۶	۵۷/۰۸	۵۰/۷۷	۳/۷۶	۸۳/۶۶	۲۴/۰۶
C_9	۴/۴۴	۳۶۴/۳۵	۴۹/۷۲	۱۶/۲۹	۴۳۰/۵۴	۳۱۶/۴۶	۲۷/۳۸	۹۲۵/۵۹	۶۶/۷۹
C_{10}	۱۶/۵۳	۳۴۸/۴۴	۱۹/۹۳	۱۵/۰۰	۸۰/۸۷	۱۲۱/۲۴	۲۰/۰۲	۱۵۸/۰۰	۳۱/۰۹
C_{11}	۱۵/۶۶	۷۰۵/۵۳	۶۳/۸۲	۲۰/۱۳	۶۲۱/۸۳	۲۸۲/۱۴	۲۱/۸۵	۶۷۱/۲۶	۱۰۷/۳۳
میانگین	۵/۵۷	۲۵۱/۴۴	۳۰/۳۷	۹/۴۳	۱۷۹/۱۹	۱۲۹/۶۳	۱۲/۴۱	۳۰۹/۰۵	۴۳/۶۰

جدول ۴ نتایج اجرای متمرکز الگوریتم‌های ITS، SA-N و SA-F به تفکیک طبقه‌های نمونه‌ای

طبقه	SA-F			SA-N			ITS		
	Gap %	σ	Time (sec.)	Gap %	σ	Time (sec.)	Gap %	σ	Time (sec.)
C_1	۰/۴۶	۸/۸۳	۲/۲۸	۰/۲۹	۱۰/۱۰	۳/۶۹	۰/۵۳	۲۰/۹۸	۱/۲۲
C_2	۱/۷۹	۸۴/۱۷	۴/۰۵	۱/۴۵	۳۰/۵۴	۱۰/۸۲	۱/۸۶	۶۱/۶۲	۳/۵۶
C_3	۲/۲۶	۹۸/۰۹	۶/۲۴	۳/۱۵	۹۴/۲۶	۲۳/۳۷	۴/۶۲	۱۴۴/۹۰	۷/۷۴
C_4	۲/۳۶	۸۳/۸۴	۷/۴۳	۴/۰۴	۱۸۱/۴۲	۳۵/۷۲	۴/۹۶	۱۷۳/۴۱	۱۱/۷۲
C_5	۴/۴۸	۲۵۰/۷۷	۱۱/۸۸	۳/۱۸	۱۲۷/۱۳	۷۵/۳۹	۳/۷۰	۱۵۰/۱۴	۲۴/۵۷
C_6	۳/۳۴	۱۳۲/۹۵	۱۴/۵۳	۴/۳۱	۵۶۷/۸۱	۱۳۰/۰۱	۴/۳۰	۴۷۹/۹۳	۴۲/۲۵
C_7	۶/۸۸	۳۳۸/۸۹	۱۸/۲۴	۳/۹۳	۳۲۸/۹۸	۱۹۸/۱۹	۴/۰۸	۳۸۲/۸۴	۶۴/۳۷
C_8	۰/۷۷	۵۵/۹۷	۶/۶۳	۱/۸۴	۸۶/۳۲	۱۴/۸۸	۳/۰۰	۱۰۴/۲۶	۵/۱۲
C_9	۷/۴۰	۵۲۲/۹۷	۳۳/۰۳	۳۳/۱۴	۱۴۲۷/۱۲	۵۱۳/۲۱	۳۵/۰۴	۱۴۲۸/۳۹	۱۶۲/۶۵
C_{10}	۱۸/۸۲	۳۸۵/۳۳	۱۳/۳۲	۱۵/۹۸	۴۲۵/۸۱	۷۶/۱۵	۱۹/۶۸	۵۵۲/۳۸	۲۴/۷۴
C_{11}	۱۶/۹۲	۵۶۴/۹۶	۴۴/۹۵	۲۳/۴۳	۸۰۳/۷۷	۱۲۸۷/۹۹	۲۳/۳۴	۸۸۹/۳۸	۵۲۳/۲۵
میانگین	۶/۷۷	۲۵۹/۵۷	۱۶/۶۵	۱۰/۵۹	۴۲۹/۲۷	۲۶۵/۷۶	۱۱/۷۰	۴۶۳/۴۸	۸۶/۸۸

نتایج حاصل از اجرای متمرکز الگوریتم‌های ITS، SA-N و SA-F به تفکیک طبقه‌های نمونه‌ای در جدول ۴ آمده است. در این آزمایش‌ها، برای اجرای الگوریتم ITS از مولفه‌های استاندارد به‌کار گرفته‌شده در مرجع [۱۲] و در اجرای الگوریتم SA-N، از مولفه‌های مشابه با جستجوی عادی به همراه حلقه‌هایی متمرکز به طول ۱۰۰ استفاده شده است. الگوریتم SA-F حلقه‌های عادی به طول ۲۰۰ را بدون تمرکز و تنها با به‌کارگیری طرح‌های به‌روزرسانی یادشده در بخش ۳-۲ به‌کار گرفته است. در این الگوریتم، برای مولفه‌های c_f، P_{min} و P_{max} به ترتیب مقادیر ۰/۹۹۹۸، ۰/۰۰۵ و ۰/۹۵ انتخاب شده‌اند.

۵- نتیجه‌گیری

در این پژوهش، روش تبرید شبیه‌سازی‌شده که از ایده‌های ابتکاری موفق در حل مسائل بهینه‌سازی ترکیباتی است، مبنای ساخت روشی برای حل مسئله‌ی مکان‌یابی تسهیلات تک‌منبع قرار گرفته است. سازوکارهای جستجو که در روش تبرید شبیه‌سازی‌شده تعبیه شده‌اند، اغلب برای یافتن جواب‌های مطلوب به زمان زیادی نیاز دارند. ازاین‌رو، دیدگاه‌های متفاوتی برای سرعت بخشیدن به سازوکار جستجوی آن موردمطالعه و ارزیابی قرار گرفته‌اند.

نتایج حاصل از آزمایش‌های گسترده در این پژوهش، برتری الگوریتم‌های پیشنهادی را در مقایسه با بهترین روش موجود در مبانی نظری مسئله، چه ازنظر میانگین زمان اجرا و چه ازنظر میانگین درصد فاصله از بهینگی، نشان می‌دهد. به بیان دقیق‌تر، در اجرای عادی (چنانکه در جدول ۱ گزارش شده)، روش‌های مبتنی بر *ITS* به شکافی حدود ۱۱ درصد (به‌طور میانگین) در زمان ۴۴۰ ثانیه (به‌طور میانگین) دست یافته‌اند. در مقابل، روش‌های مبتنی بر *SA* با شکافی حدود ۶ درصد در زمان ۹۰ ثانیه، به نتایج بهتری دست یافته‌اند. نتایج گزارش‌شده در جدول‌های ۲ و ۳ نیز به‌نوبه خود از برتری روش‌های مبتنی بر *SA* در اجرا با اعمال روش‌های موازی‌سازی حکایت دارد. به‌طور ویژه، روش‌های مبتنی بر *ITS* در این آزمایش‌ها به شکافی حدود ۱۲ درصد در زمان ۴۴ ثانیه دست یافته‌اند. در مقابل، روش‌های مبتنی بر *SA* با شکافی حدود ۶ درصد در زمان ۳۰ ثانیه، نتایج بهتری یافته‌اند. برتری روش‌های مبتنی بر *SA* در اجرای متمرکز نیز قابل‌مشاهده است. چنانکه در جدول ۴ گزارش شده، روش‌های مبتنی بر *ITS* در اجرای متمرکز به شکافی حدود ۱۲ درصد در زمان ۸۷ ثانیه دست یافته‌اند. در مقابل، روش‌های مبتنی بر *SA* با شکافی حدود ۷ درصد در زمان ۱۷ ثانیه، به نتایج بسیار بهتری دست یافته‌اند.

در برخی از کاربردهای عملی مسئله، هزینه اتصال مشتری به تسهیلات، متأثر از میزان تقاضای مشتری، تابعی غیرخطی (محدب) در نظر گرفته می‌شود تا مفهوم مهم اقتصاد مقیاس را در برگیرد. این مهم مطالعه الگوریتم‌های مطرح‌شده در این پژوهش را در حضور مدل‌های واقع‌گرایانه‌تر از هزینه طلب می‌کند که خود می‌تواند هدف مطالعات آتی قرار گیرد.

۶- منابع

[1] H. A. Eiselt and V. Marianov, (2011). *Foundations of location analysis*, Vol. 155. Springer Science & Business Media.

[2] A., Klose, and A. Drexl, Facility location models for distribution system design. *European journal of operational research*, 162 (1), 2005, 4-29.

[3] M. Akbari, A Model for Production and Inventory Control in Crisis Condition, *Management Research in Iran*, 19 (4), 2016, 45-70. (In Persian)

[4] A. Jafarnejhad, M. Esmaelian, and M. Rzvani, An Inventory-Location Model Formulation and Computational Results, *Management Research in Iran*, 12 (1), 2008, 105-125. (In Persian)

[5] M. Asadi, M. Ali Shafia and S .Yaghoubi, A Disaster Facilities Location-Allocation Model Considering Reliability under Uncertainty and Dynamic Demand (Case Study: Earthquake Disaster in Tehran). *Modern Researches in Decision Making*, 3 (1), 2018, 1-28. (In Persian)

[6] S. Basu, M. Sharma and P. S. Ghosh, Metaheuristic applications on discrete facility location problems: a survey. *Opsearch*, 52(3), 2015, 530-561.

[7] J. A. Diaz, and E. Fernández, A branch-and-price algorithm for the single source capacitated plant location problem. *Journal of the Operational Research Society*, 53 (7), 2002, 728-740.

[8] Z. Yang, F. Chu and H. Chen, A cut-and-solve based algorithm for the single-source capacitated facility location problem. *European Journal of Operational Research*, 221 (3), 2012, 521-532.

[9] R. D. Galvao, and V. Marianov, Lagrangean relaxation-based techniques for solving facility location problems. In Foundations of location analysis. *Springer US*., 2011, 391-420.

[10] R. K. Ahuja, J. B. Orlin, S. Pallottino, M. P. Scaparra and M. G. Scutella, A multi-exchange heuristic for the single-source capacitated facility location problem. *Management Science*, 50 (6), 2004, 749-760.

[11] I. A. Contreras, and J. A. Diaz, Scatter search for the single source capacitated facility location problem. *Annals of Operations Research*, 157 (1), 2008, 73-89.

[12] S. C. Ho, An iterated tabu search heuristic for the single source capacitated facility location problem, *Applied Soft Computing*, 27, 2015, 169-178.

[13] E. Aarts, J. Korst and W. Michiels, Simulated annealing, *Search methodologies*, Springer, Boston, MA, 2014, 265-285.

[14] S. Ghafoori, and M. Taghizadeh yazdi, Proposing a Multi-Objective Mathematical Model for RCPSP and Solving It with Firefly and Simulated Annealing algorithms. *Modern Researches in Decision Making*, 1(4), 2016, 117-142.

[15] B. Chapman, G. Jost and R. Van Der Pas, (2008). Using OpenMP: portable shared *memory parallel programming.* Vol. 10. MIT press.

[16] W. Gropp, E. Lusk and A. Skjellum, (2014). *Using MPI: portable parallel programming with the message-passing interface*. Vol. 1. MIT press.