مهندسی صنایع و مدیریت شریف
دوره‌ی ۱ - ۲۶، شماره‌ی ۲، ص. ۳۵-۴۵

معرفی یک الگوریتم جست‌وجوی ممنوع برای حل مسئله‌ی تک‌سطری چیدمان

حامد سمرقندی (دانشجوی کارشناس ارشد)
کورش عشقی (استاد)
دانشکده‌ی مهندسی صنایع، دانشگاه صنعتی شریف

طراحی چیدمان عبارت است از تعیین یک چینش مناسب برای تعدادی تجهیزات به‌نحوی که کل هزینه‌های مرتبط با جریان میان قسمت‌ها را کمینه کند. یکی از مسائلی که در طراحی چیدمان کاربرد عملی زیادی دارد، مسئله‌ی چیدمان تک‌سطری یا یک‌ردیفه‌ی امکانات (SRFLP)[۱] است. این مسئله، مسئله‌یی از رده‌ی NP-Complete است و تلاش‌های فراوانی برای به دست آوردن جواب‌های نزدیک به بهینه یا مدل‌سازی مجدد آن صورت گرفته است. در نوشتار حاضر ابتدا به بررسی حالتی خاص در SRFLP می‌پردازیم و قضیه‌ی سودمندی را در رابطه با جواب بهینه‌ی این حالت اثبات می‌کنیم. سپس یک الگوریتم جست‌وجوی ممنوع (TS)[۲] را به‌کمک جواب بهینه‌ی حالت خاص مذکور برای حل SRFLP توسعه داده و نحوه‌ی عملکرد آن را بررسی می‌کنیم. نتایج محاسباتی نشان‌گر کارآیی و قدرت محاسباتی چشم‌گیر الگوریتم پیشنهادی در مقایسه با سایر الگوریتم‌های مشابه برای حل مسئله است، به‌نحوی که جواب نزدیک به بهینه برای مسائل SRFLP که حتی تا ۲۰۰ قسمت دارند در زمان بسیار اندکی به دست می‌آید.

واژگان کلیدی: مسئله‌ی چیدمان، مسئله‌ی مرتب‌سازی خطی، الگوریتم جست‌وجوی ممنوع.

Hamed.Samarghandy@Gmail.com
Eshghi@Sharif.Edu

مقدمه

به‌لحاظ عملی ثابت شده است که حدود ۲۰ تا ۵۰ درصد از قیمت تمام‌شده‌ی یک محصول ناشی از هزینه‌های حمل‌ونقل آن است و با استفاده از یک چیدمان مناسب می‌توان این هزینه‌ها را به ۲۰ تا ۳۰ درصد قیمت تمام‌شده کاهش داد.[۱] بعضی از مؤلفان دیگر این درصد را بین ۳۰ تا ۷۰ نیز تخمین زده‌اند.[۲] به‌همین جهت، تاکنون محققان بسیاری مسائل طراحی چیدمان را بررسی کرده‌اند. در این میان می‌توان مسائل طراحی چیدمان را به مسائلی چون مرتب‌سازی[۳]، چیدمان تک‌سطری (یک‌ردیفه) و چیدمان چندسطری (چندردیفه) تقسیم کرد.[۳] مسائل دسته اول شامل مسائلی هستند که در آن‌ها قصد داریم قسمت‌هایی را بدون در نظر گرفتن طول و عرض آن‌ها در نقاط از پیش‌تعیین‌شده قرار دهیم. دسته‌ی بعدی مسائل تک‌سطری (یک‌ردیفه) هستند که فرضیات آن‌ها در این نوشتار به‌طور کامل مطرح خواهد شد. این مسائل از لحاظ پیچیدگی محاسباتی در سطحی بالاتر از مسائل مرتب‌سازی قرار می‌گیرند.[۳] در نهایت دسته‌ی سوم مسائل چندسطری (چندردیفه) را شامل می‌شود که با هدف طراحی چیدمان در یک صفحه طراحی می‌شوند و از بالاترین سطح پیچیدگی محاسباتی برخوردارند.[۳]

یکی از مسائل طراحی چیدمان که در عمل کاربرد فراوانی دارد، چیدمان تک‌سطری (یک‌ردیفه) است که به‌اختصار SRFLP نامیده می‌شود. مسئله‌ی چیدمان یک‌ردیفه را می‌توان چنین توصیف کرد: قصد داریم n قسمت مستطیل‌شکل را روی یک خط مستقیم با جهت مشخص قرار دهیم. در این مسئله، l_i طول قسمت معادل i فرض شده است. همچنین یک ماتریس $n \times n$ به‌صورت $C = [c_{ij}]$ داده شده است که در آن c_{ij} نشان‌گر میزان جریان میان قسمت‌های i و j است. فاصله‌ی بین دو قسمت نیز از مرکزشان اندازه‌گیری می‌شود. هدف مسئله چیدن این قسمت‌ها است به‌نحوی که مجموع وزنی فواصل میان تمامی قسمت‌ها کمینه شود.

همان‌طورکه اشاره شد، مسئله‌ی SRFLP کاربردهای عملی فراوانی دارد، مانند: چیدمان قسمت‌های یک سوپرمارکت، بیمارستان یا ساختمان دفتری در یک سمت یک راهرو،[۴] چیدمان کتاب‌ها در یک طبقه از کتابخانه، نحوه‌ی تخصیص گیت‌های یک فرودگاه به هواپیماها،[۳] نحوه‌ی تخصیص فایل‌های یک رایانه به سیلندرهای دیسک سخت آن،[۵] چیدمان ماشین‌آلات در مسیری به‌شکل خط مستقیم که توسط یک راهبر خودکار (AGV)[۴] پیموده می‌شود.[۶] مورد آخر به‌خصوص در سیستم‌های ساخت و تولید انعطاف‌پذیر (FMS)[۵] کاربرد فراوانی دارد. بیشتر چیدمان‌های موجود در سیستم‌های FMS را می‌توان با مسئله‌ی یک‌ردیفه‌ی مطرح‌شده در این تحقیق متناظر کرد زیرا این مدل را می‌توان برای حل چیدمانی که از یک مسیر منحنی‌شکل استفاده می‌کند نیز به کار برد. البته درمورد این چیدمان لازم است که تجهیزات یا همان ماشین‌آلات در یک طرف AGV قرار گیرند. همچنین، یادآور می‌شود که مسئله‌ی SRFLP یک مسئله‌ی NP-hard است[۷] و بنابراین علاوه بر کاربردهای

تاریخ: دریافت ۱۳۸۷/۶/۱۸، داوری ۱۳۸۷/۱۰/۲۲، پذیرش ۱۳۸۹/۴/۱.

عملی، برای محققان نیز جذابیت‌های تحقیقاتی فراوانی دارد. همین امر موجب به وجود آمدن ادبیاتی غنی و در حال گسترش در مورد این مسئله شده است که در ادامه به آن خواهیم پرداخت.

در نوشتار حاضر ابتدا به مرور ادبیات علمی مرتبط با SRFLP خواهیم پرداخت. سپس با اضافه‌کردن یک محدودیت، به بررسی حالت خاصی از SRFLP می‌پردازیم و جواب‌های بهینه‌ی این حالت را می‌یابیم. سپس از این جواب‌بهینه برای راه‌اندازی یک الگوریتم جست‌وجوی ممنوع (TS) در حل مسئله‌ی SRFLP استفاده می‌شود؛ الگوریتمی که تلاش دارد برای حالت کلی مسئله جواب‌های مناسبی را بیابد. نتایج محاسباتی که در انتها خواهند آمد، مؤید کارآیی بسیار مناسب الگوریتم پیشنهادی در مقایسه با سایر روش‌هاست.

مرور ادبیات

به‌دلیل NP-hard بودن مسئله،[۷] الگوریتم‌های ابتکاری و فراابتکاری فراوانی به‌موازات روش‌های دقیق برای حل SRFLP ارائه شده‌اند. در یک رویه‌ی پیشنهادی برای حل SRFLP، در هر گام یک قسمت به انتهای جواب کنونی اضافه می‌شود تا بدین ترتیب جواب نهایی حاصل شود.[۸]

برای حل این مسئله روشی بر پایه‌ی استفاده از مقادیر ویژه‌ی ماتریس میزان جریان ارائه شد.[۹] سپس مطابق روشی که بعدها مطرح شد، در ابتدا دو قسمتی که بیشترین ارتباط را با یکدیگر دارند در مرکز قرار می‌گیرند و سپس در هر مرحله یک قسمت به سمت چپ یا راست این دو قسمت اضافه می‌شود.[۶]

همین افراد چند سال بعد یک مدل برنامه‌ریزی خطی درهم‌آمیخته (LMIP)[۶] برای مسئله‌ی SRFLP معرفی و سپس این مدل را از طریق روش جریمه[۷] حل کردند.[۶] اولین روش فراابتکاری درمورد این مسئله در سال ۱۹۹۲ به‌صورت یک الگوریتم عملیات حرارتی تبرید (SA)[۸] طراحی شد.[۱۰] چندسال بعد با استفاده از یک الگوریتم ترکیبی[۹] برمبنای الگوریتم‌های ژنتیک (GA و SA) نسبت به کمینه‌سازی میزان برگشت به عقب در ترتیب‌های مختلف قسمت‌ها اقدام شد.[۱۱] در این الگوریتم نیز طول کلیه‌ی قسمت‌ها با هم برابر فرض شده‌اند و محل‌های قرارگیری قسمت‌ها نیز از پیش تعیین شده است. پس از آن یک الگوریتم با رهیافت آزمندی[۱۰] برای حل SRFLP توسعه یافت. این الگوریتم سعی دارد قسمت‌هایی که میان آن‌ها رابطه‌ی زیاد وجود دارد را به مکان‌هایی در کنار یکدیگر و بر روی یک خط مستقیم تخصیص دهد. اما الگوریتم مورد بحث طول تمامی قسمت‌ها را برابر واحد در نظر می‌گیرد که بسیار محدودکننده است.[۱۲]

محققین برای حل مسئله‌ی SRFLP از الگوریتم ژنتیک بهره گرفته‌اند.[۱۳] در الگوریتم ژنتیکی که آنان پیشنهاد کردند، فرض بر این است که هر جواب موجه در واقع تشکیل‌دهنده‌ی یک ژن است. سپس از ژن‌هایی که در طول فرایند حل به دست آمده‌اند به‌عنوان والد استفاده می‌شود.

محققین دیگری برای حل مسئله‌ی مورد بحث از الگوریتم بهینه‌سازی کولونی مورچگان (ACO)[۱۱] استفاده کرده‌اند.[۱۴] در این تحقیق یک مدل تخصیص درجه‌ی دو یا (QAP)[۱۲] برای مسئله معرفی شده است. در سال ۲۰۰۵ نیز از تکنیک جدیدی به‌نام برنامه‌ریزی نیمه‌معین (SDP)[۱۳] برای حل SRFLP استفاده شد[۱۵] به‌نحوی که از تقارن طبیعی مسئله برای توسعه‌ی یک حد بالا برای SRFLP استفاده شده است. سپس همین محققان در سال ۲۰۰۸ و به‌کمک همین رهیافت روشی را برای یافتن جواب‌بهینه‌ی این مسئله طراحی کردند.[۱۶] نهایتاً در سال ۲۰۰۹ روشی برای یافتن حد پایین این مسئله برمبنای ارائه‌ی یک مدل برنامه‌ریزی خطی و معادلات محدودکننده‌ی فضای آزاد توسعه یافت.[۱۷]

با بررسی روش‌های ابتکاری که تاکنون برای حل مسئله‌ی SRFLP پیشنهاد شده است، متوجه می‌شویم که این الگوریتم‌ها حداقل دارای یکی از ضعف‌های زیرند:[۱۰،۱۱،۱۳ و۱۴]

۱. اندازه قسمت‌ها در نظر گرفته نشده‌اند، یا یکسان فرض شده‌اند؛

۲. محل قرارگیری بعضی از ماشین‌آلات از ابتدا مشخص فرض شده است؛

۳. اندازه‌ی ماشین‌آلات فقط هنگام قراردادن فیزیکی ماشین‌آلات در نظر گرفته شده‌اند و در محاسبات دخیل نیستند.

این نوشتار ابتدا حالت خاصی از SRFLP را بررسی کرده و جواب بهینه‌ی آن را به دست می‌آورد. سپس از این جواب برای راه‌اندازی یک الگوریتم TS استفاده می‌شود که تلاش دارد مشکلات فوق را بی‌اثر سازد. در الگوریتم پیشنهادی اندازه‌ی قسمت‌ها با یکدیگر متفاوت فرض شده و محل قرارگیری آن‌ها از پیش تعیین نمی‌شود. در بخش بعدی به بررسی حالت خاصی از SRFLP خواهیم پرداخت که نقش مهمی در توسعه‌ی الگوریتم پیشنهادی دارد.

بررسی حالت خاص در SRFLP

مدل ریاضی مسئله‌ی چیدمان یک‌ردیفه، معروف به ABSMODEL، در سال ۱۹۹۱ معرفی شده است.[۶] شرح این مدل عبارت است از:

$$\min z = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} d_{ij} \tag{1}$$

$s.t:$

$$d_{ij} \geq \frac{1}{2}(l_i + l_j) + s_{ij}; \qquad i = 1, 2, ..., n; \;\; j = i+1, ..., n \tag{2}$$

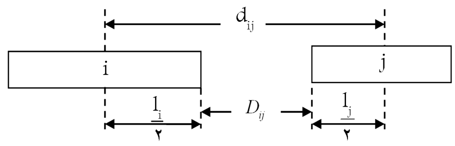
$$d_{ij} \geq 0; \qquad i = 1, 2, ..., n; \;\; j = i+1, ..., n \tag{3}$$

در این مدل، d_{ij} فاصله‌ی مرکز به مرکز دو قسمت i، j و l_i طول قسمت iام است. همچنین در این مدل s_{ij} فضای خالی مورد نیاز میان قسمت‌های i و j است که به‌عنوان یک پارامتر داده می‌شود.

بررسی حالات خاص به‌هنگام طراحی الگوریتم فراابتکاری بسیار مهم است. امروزه میزان محاسبات مورد نیاز برای رسیدن به جواب نزدیک به بهینه در الگوریتم‌های فراابتکاری بسیار مورد توجه است. در واقع تلاش محاسباتی مورد نیاز برای حل یک مسئله نشانه‌یی از کارآیی یک الگوریتم فراابتکاری است.[۱۸] وجود جواب‌های بهینه یا جواب‌های خوب برای حالات خاص می‌تواند باعث آغاز جست‌وجو در الگوریتم فراابتکاری از یک جواب مناسب شود که این امر نقش مهمی در کاهش تلاش محاسباتی خواهد داشت.

در اینجا قصد داریم حالت خاصی را بررسی کنیم که در آن $\forall i, j : c_{ij} = c$. در این‌صورت تابع هدف مسئله را می‌توان چنین نوشت:

$$\min z = c \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} \tag{4}$$



شکل ۱. نحوه‌ی محاسبه‌ی فاصله‌ی مرکز به مرکز قسمت‌ها.

در حالت کلی، فواصل بین قسمت‌های i و j مطابق رابطه‌ی ۵ قابل محاسبه است:[۶]

$$d_{ij} = \frac{l_i + l_j}{۲} + D_{ij} \qquad (۵)$$

که در آن D_{ij} فضای میان قسمت‌های i و j است. برای درک بهتر این مفهوم به شکل ۱ توجه کنید. توجه داشته باشید که مقدار D_{ij} لزوماً برابر s_{ij} نیست، زیرا ممکن است $s_{ij} = ۰$، ولی مابین قسمت‌های i و j قسمت دیگری مانند قسمت k وجود داشته باشد که در این‌صورت داریم $D_{ij} \neq ۰$.

قضیه‌ی ۱: اگر $\forall i, j : c_{ij} = c; s_{ij} = ۰$، آنگاه ابتدا قسمت‌ها را برحسب طول‌شان به‌طور صعودی مرتب و شماره‌گذاری می‌کنیم. یعنی کوتاه‌ترین قسمت با اندیس ۱ و بلندترین قسمت با اندیس n شناسایی می‌شود. در این حالت اگر i فرد باشد طرح چیدمان بهینه عبارت است از:

... i ... ۳ ۱ ۲ ... i-۱ ...

و اگر i زوج باشد، طرح چیدمان بهینه عبارت است از:

... i ... ۲ ۱ ۳ ... i-۱ ...

اثبات: اگر مقدار تابع هدف مسئله‌ی SRFLP برای این طرح چیدمان z فرض شده باشد، از برهان خلف برای اثبات آن استفاده می‌کنیم. فرض می‌کنیم تغییر محل قسمت‌های i و j باعث تولید طرح چیدمانی با تابع هدف z' می‌شود، به‌نحوی که $z > z'$. بدون واردشدن خللی به کلیت مسئله، فرض می‌کنیم قسمت i در طرح چیدمان اولیه سمت چپ قسمت j قرار دارد و تعداد قسمت‌ها یک عدد فرد است. از طرف دیگر، تغییر درتابع هدف z به‌دلیل جابه‌جایی قسمت‌های i و j با سایر قسمت‌ها است. پس تابع هدف z' را می‌توان مطابق رابطه‌ی ۶ از طریق تابع هدف z تولید کرد:

$$z' = z + \sum_{k=۱}^{n} (d'_{ik} + d'_{jk} - d_{ik} - d_{jk}) \qquad (۶)$$

حال برای این که $z > z'$ باشد، در واقع باید داشته باشیم:

$$\sum_{k=۱}^{n} (d_{ik} + d_{jk}) > \sum_{k=۱}^{n} (d'_{ik} + d'_{jk}) \qquad (۷)$$

می‌دانیم که:

$$\begin{aligned} d_{ik} &= \frac{l_i + l_k}{۲} + D_{ik} \\ d_{jk} &= \frac{l_j + l_k}{۲} + D_{jk} \\ d'_{ik} &= \frac{l_i + l_k}{۲} + D'_{ik} \\ d'_{jk} &= \frac{l_j + l_k}{۲} + D'_{jk} \end{aligned} \qquad (۸)$$

با جایگذاری مقادیر داده شده در رابطه‌های ۸ در نامعادله‌ی ۷ و ساده‌سازی خواهیم داشت:

$$\sum_{k=۱}^{n} (D_{ik} + D_{jk}) > \sum_{k=۱}^{n} (D'_{ik} + D'_{jk}) \qquad (۹)$$

برای محاسبه‌ی مقادیر D، ابتدا قسمت‌ها را به سه مجموعه تقسیم می‌کنیم:

۱. قسمت‌هایی که در طرح چیدمان اولیه سمت چپ قسمت i قرار داشته‌اند. این قسمت‌ها را قسمت‌های M می‌نامیم و فرض می‌کنیم که تعدادشان m باشد.

۲. قسمت‌هایی که در طرح چیدمان اولیه سمت راست قسمت i و سمت چپ قسمت j قرار داشته‌اند. این قسمت‌ها را قسمت‌های B می‌نامیم و فرض می‌کنیم تعدادشان b باشد.

۳. قسمت‌هایی که در طرح چیدمان اولیه سمت راست قسمت j قرار داشته‌اند. این قسمت‌ها را قسمت‌های R می‌نامیم و فرض می‌کنیم تعدادشان r باشد.

حال معادله‌ی ۹ را با توجه به دسته‌بندی‌های انجام شده می‌توان بازنویسی کرد:

$$\begin{aligned} &\sum_{k \in M} (D_{ik} + D_{jk}) + \sum_{k \in B} (D_{ik} + D_{jk}) + \sum_{k \in R} (D_{ik} + D_{jk}) > \\ &\sum_{k \in M} (D'_{ik} + D'_{jk}) + \sum_{k \in B} (D'_{ik} + D'_{jk}) + \sum_{k \in R} (D'_{ik} + D'_{jk}) \end{aligned} \qquad (۱۰)$$

با توجه به طرح چیدمان اولیه و طرح چیدمان پس از انجام جابه‌جایی، درمورد قسمت‌های دسته‌ی M، مانند قسمت k داریم:

k ... i ... ۳ ۱ ۲ ... j ...

k ... j ... ۳ ۱ ۲ ... i ...

$$\begin{aligned} D'_{ik} &= D_{jk} + l_j - l_i \\ D'_{jk} &= D_{ik} \end{aligned} \qquad (۱۱)$$

و در مورد هر قسمت دسته‌ی B، مانند قسمت k، داریم:

i ... k ... ۳ ۱ ۲ ... j ...

j ... k ... ۳ ۱ ۲ ... i ...

$$\begin{aligned} D'_{ik} &= D_{jk} \\ D'_{jk} &= D_{ik} \end{aligned} \qquad (۱۲)$$

همچنین درمورد هر قسمت دسته‌ی R، مانند قسمت k، داریم:

i ... ۳ ۱ ۲ ... j ... k

j ... ۳ ۱ ۲ ... i ... k

$$\begin{aligned} D'_{ik} &= D_{jk} \\ D'_{jk} &= D_{ik} - l_j + l_i \end{aligned} \qquad (۱۳)$$

پس از جایگذاری مقادیر معادلات ۱۱ تا ۱۳ در معادله‌ی ۱۰ و ساده‌سازی آن، معادله‌ی ۱۴ حاصل می‌شود:

$$\sum_{k\in M}(l_j - l_i) + \sum_{k\in R}(l_i - l_j) < 0 \qquad (۱۴)$$

با توجه به این که تعداد اعضای M برابر m و تعداد اعضای R برابر r است، می‌توان معادله‌ی ۱۴ چنین ساده کرد:

$$T < 0 \qquad (۱۵)$$

که در آن:

$$T = (m-r)\times l_j - (m-r)\times l_i \qquad (۱۶)$$

در این‌صورت تمامی حالات ممکن را می‌توان در سه حالت خلاصه کرد:

حالت اول

$$m = r \Rightarrow T = 0 \qquad (۱۷)$$

در این حالت $T \nless 0$ و بنابراین جابه‌جایی قسمت‌های i و j باعث کاهش مقدار تابع هدف نمی‌شود.

حالت دوم

$$m > r; \quad l_j \geq l_i \qquad (۱۸)$$

در این حالت نیز $T \nless 0$ و بنابراین جابه‌جایی قسمت‌های i و j باعث کاهش مقدار تابع هدف نمی‌شود.

حالت سوم

$$m > r; \quad l_j \leq l_i \qquad (۱۹)$$

توجه داشته باشید که طبق فرضیات قضیه، اگر $l_j < l_i$، این حالت نشدنی است. چراکه طبق معادله‌ی ۱۹ در طرح اولیه تعداد قسمت‌های سمت چپ قسمت i از تعداد قسمت‌های سمت راست قسمت j بیشتر است و در این‌صورت طبق فرضیات باید $l_j > l_i$ که تناقض دارد. اما اگر $l_j = l_i$، باز هم $T \nless 0$ و بنابراین جابه‌جایی قسمت‌های i و j باعث کاهش مقدار تابع هدف نمی‌شود.

حالت چهارم

$$m < r; \quad l_j \geq l_i \qquad (۲۰)$$

توجه داشته باشید که طبق فرضیات قضیه، اگر $l_j > l_i$، این حالت نیز نشدنی است. چراکه طبق معادله‌ی ۲۰ در طرح اولیه تعداد قسمت‌های سمت چپ قسمت i از تعداد قسمت‌های سمت راست قسمت j بیشتر است و در این‌صورت طبق فرضیات باید $l_j < l_i$ که تناقض است. اما اگر $l_j = l_i$، باز هم $T \nless 0$ و بنابراین جابه‌جایی قسمت‌های i و j باعث کاهش مقدار تابع هدف نمی‌شود.

حالت پنجم

$$m < r; \quad l_j \leq l_i \qquad (۲۱)$$

در این حالت نیز $T \nless 0$ و بنابراین جابه‌جایی قسمت‌های i و j باعث کاهش مقدار تابع هدف نمی‌شود.■

از معادلات ۱۷، ۱۹ و ۲۰ می‌توان برای یافتن سایر جواب‌های بهینه‌ی مسئله‌ی SRFLP با شرایط و فرضیات این بخش استفاده کرد. این معادلات در واقع نشان می‌دهند که اگر قسمت‌های i و j را به‌نحوی انتخاب کنیم که $T = 0$، جابه‌جایی جفتی این دو قسمت با یکدیگر بر مقدار تابع هدف بی‌تأثیر است. بدین ترتیب می‌توان با انتخاب مناسب قسمت‌های i و j، جواب‌های بهینه‌ی چندگانه‌ی این مسئله را به دست آورد.

الگوریتم TS پیشنهادی برای حل SRFLP

روش جست‌وجوی ممنوع در سال ۱۹۸۶ توسعه داده شد[۱۹] و یکی از موفق‌ترین روش‌های حل نزدیک به بهینه‌ی مسائل بهینه‌سازی ترکیبی به‌شمار می‌رود.[۲۰] به‌همین دلیل، با وجود قدمت تاریخی روش TS، استفاده از آن برای حل نزدیک به بهینه‌ی مسائل بهینه‌سازی ترکیبی به‌شکل روزافزون در حال افزایش است که عمده دلایل آن را می‌توان مؤثر بودن TS برای حل مسائل و سادگی اجرا و پیاده‌سازی آن عنوان کرد. در هر حال، برای انتخاب یک الگوریتم فراابتکاری با توجه به نوع مسئله‌ی مورد بررسی، روشی نظام‌مند موجود نیست و سلیقه‌ی محقق در این انتخاب بسیار مؤثر بوده و مناسب‌بودن الگوریتم انتخاب‌شده، عمدتاً برمبنای نتایج محاسباتی سنجیده می‌شود. در TS، مانند بسیاری از الگوریتم‌های فراابتکاری دیگر، از جواب کنونی x_t در تکرار t الگوریتم، به بهترین جواب موجود در همسایگی $N(x_t)$ گذر می‌کنیم. این جواب جدید را x_{t+1} می‌نامیم. از آنجا که x_{t+1} لزوماً از x_t بهتر نیست، یک سازوکار ممنوع‌سازی برای جلوگیری از دورافتادگی در نظر می‌گیریم.

یکی از روش‌های جلوگیری از دورافتادگی، ممنوع‌کردن بازگشت به جواب‌هایی است که قبلاً به دست آمده‌اند ولی استفاده از چنین روشی نیازمند نگه‌داری تعداد زیادی جواب در حافظه‌ی حل مسئله است. روش دیگر، در نظرگرفتن بعضی از خصوصیات جواب‌های به‌دست آمده و ممنوع‌کردن این خصوصیات در جواب‌های جدید برای تعداد دلخواهی تکرار (مثلاً θ تکرار) در آینده است. معمولاً این سازوکار را حافظه‌ی کوتاه‌مدت (STM)[۱۴] می‌نامیم.[۲۰]

در TS معمولاً از سازوکارهایی مانند متنوع‌سازی[15] و قدرت‌مندسازی[16] نیز استفاده می‌شود. هدف از متنوع‌سازی حصول اطمینان از جست‌وجوی مناطق وسیعی از فضای شدنی است. قدرت‌مندسازی جست‌وجو پیرامون بهترین جواب به دست آمده در فضای شدنی را شامل می‌شود.[۲۰] به‌علاوه معمول است که در هنگام طراحی ساختار همسایگی از یک معیار رضایت‌مندی[17] استفاده شود. معیار رضایت‌مندی بر STM تقدم دارد. برای مثال، اگر مشخصات یک جواب خاص در STM وجود داشته باشد ولی پذیرش این جواب باعث دورافتادگی نشود و همچنین این جواب از کلیه‌ی جواب‌های قدیمی بهتر باشد، معیار رضایت‌مندی پذیرش این جواب را ممکن می‌سازد.[۲۰] اطلاعات دقیق و جامعی درمورد روش جست‌وجوی ممنوع وجود دارد که می‌توان به آنها مراجعه کرد.[۱۹] پس از انجام این بررسی اجمالی، به معرفی الگوریتم پیشنهادی خواهیم پرداخت.

۱. جواب‌های اولیه

الگوریتم TS نیز مانند بسیاری از الگوریتم‌های جست‌وجوی محلی دیگر، برای راه‌اندازی نیاز به تعدادی جواب اولیه دارد. خوشبختانه یافتن جواب‌های اولیه برای SRFLP بسیار ساده است. درواقع هر ترتیب قرارگرفتن n قسمت بر روی یک خط مستقیم یک جواب مسئله‌ی SRFLP را تولید می‌کند.

جواب‌های اولیه‌یی که برای راه‌اندازی الگوریتم پیشنهادی مورد استفاده قرار

می‌گیرند، توسط قضیه‌ی ۱ تولید می‌شوند. نقش کلیدی استفاده از این جواب‌ها در بخش نتایج محاسباتی مورد بررسی قرار می‌گیرد.

چنان که پیش‌تر اشاره شد، می‌توان از معادلات ۱۷، ۱۹ و ۲۰ برای یافتن سایر جواب‌های بهینه‌ی SRFLP با شرایط و فرضیه‌های قضیه‌ی ۱ استفاده کرد. این معادلات در واقع نشان می‌دهد که اگر قسمت‌های i و j را به‌نحوی انتخاب کنیم که تعداد قسمت‌های مجموعه‌ی L و تعداد قسمت‌های مجموعه برابر R باشند، جابه‌جایی جفتی این دو قسمت با یکدیگر بر مقدار تابع هدف بی‌تأثیر است. بدین‌ترتیب می‌توان با انتخاب مناسب قسمت‌های i و j، جواب‌های بهینه‌ی چندگانه‌ی مسئله‌ی قضیه‌ی ۱ را به دست آورد.

تعداد جواب‌های اولیه که می‌توان با کمک این واقعیت برای مسئله‌ی SRFLP با n قسمت تولید کرد، حداقل برابر $۲^{\left[\frac{n}{۲}\right]}$ است که همگی آن‌ها نیز جواب بهینه‌ی مسائلِ دارای شرایط قضیه‌ی ۱ هستند. از بخشی از این جواب‌ها برای راه‌اندازی الگوریتم TS استفاده می‌کنیم. حال اگر الگوریتم برای شروع به کار به تعداد بیشتری جواب اولیه احتیاج داشته باشد، این جواب‌های اولیه‌ی اضافی را به‌طور تصادفی تولید خواهیم کرد. در نهایت، مقدار تابع هدف کلیه‌ی جواب‌های تولیدشده محاسبه، و همگی آن‌ها را به‌همراه مقدار تابع هدف‌شان به حافظه‌ی تطابقی یا AM [۱۸] که ماهیت آن در بخش‌های بعد توضیح داده خواهد شد، منتقل می‌کنیم.

۲. ساختار همسایگی

تعریف ساختار همسایگی یکی از مهم‌ترین بخش‌ها در توسعه‌ی الگوریتم TS یا هر الگوریتم فراابتکاری دیگر است. در تعریف الگوریتم پیشنهادی از یک ساختار همسایگی برمبنای جابه‌جایی‌های دوتایی استفاده می‌شود. بدین معنا که اگر یک جواب x_t را داشته باشیم، انتخاب هر دو قسمت از این طرح چیدمان و جابه‌جایی آنها با یکدیگر باعث به‌وجود آمدن جوابی دیگر در همسایگی جواب کنونی می‌شود. حال اگر جابه‌جایی این دو قسمت باعث کاهش مقدار تابع هدف ABSMODEL شود، این جابه‌جایی پذیرفته خواهد شد؛ در غیر این‌صورت جابه‌جایی پذیرفته نشده و دو قسمت دیگر برای جابه‌جایی (به‌طور تصادفی) انتخاب می‌شوند.

پس از این که یک جابه‌جایی مورد پذیرش واقع شد، قسمت‌های جابه‌جا شده به‌صورت یک دوتاییِ غیرمرتب وارد لیست ممنوع [۱۹] می‌شوند. برای مثال، اگر جابه‌جایی قسمت‌های i و j مورد پذیرش واقع شود، دوتایی (i, j) وارد لیست ممنوع خواهد شد. این بدان معناست که تا θ تکرار بعدی الگوریتم، جابه‌جایی قسمت‌های i و j ممنوع است مگر این که هیچ جابه‌جایی دیگری مورد پذیرش قرار نگیرد، یا جابه‌جایی این دو قسمت باعث بهترشدن تابع هدف شود به‌نحوی که مقدار تابع هدف حاصل از این جابه‌جایی از تمامی مقادیر توابع هدف‌به دست آمده‌ی کنونی بهتر باشد.

نکته‌ی قابل توجه این که ممکن است یافتن یک جواب بهتر در همسایگی جواب کنونی عملی وقت‌گیر یا حتی غیرممکن باشد. بنابراین فرض می‌کنیم اگر بعد از a جابه‌جایی جواب بهتری در همسایگی جواب کنونی یافت نشد، به جست‌وجو در همسایگی این جواب خاتمه دهیم و برای جست‌وجو در همسایگی آن جوابی دیگر را از AM انتخاب کنیم.

۳. حافظه‌ی بلندمدت، قدرت‌مندسازی و شرط توقف

قدرت‌مندسازی عبارت است از استفاده از جواب‌های خوبی که برای تولید جواب‌های بهتر و نزدیک‌تر به جواب بهینه، تاکنون تولید شده‌اند. در این الگوریتم برای قدرت‌مندسازی از تکنیکی به‌نام حافظه‌ی تطابقی(AM) استفاده می‌شود. مفهوم AM احتمالاً توان‌مندترین ابزار قدرت‌مندسازیِ TS است.[۲۰] AM مجموعه‌یی از جواب‌های خوب است که در حین فرایند جست‌وجو تولید شده‌اند.[۲۱] از این ابزار در الگوریتم پیشنهادی نیز استفاده می‌کنیم.

به‌طور خلاصه، AM لیستی به‌طول L است که در آن طرح‌های چیدمان تولیدشده توسط الگوریتم نگه‌داری می‌شوند. هر طرح چیدمان در AM برچسبی دارد که مقدار آن برابر مقدار تابع هدف همان طرح چیدمان است. طرح‌های چیدمان تولیدشده که در AM قرار می‌گیرند، برحسب مقدار برچسب خود به‌طور صعودی مرتب می‌شوند. معنای این امر آن است که طرح چیدمانی که در ابتدای لیست قرار می‌گیرد کم‌ترین مقدار تابع هدف را دارد و طرح چیدمانی که در انتهای لیست قرار می‌گیرد دارای بیشترین مقدار تابع هدف است. در ابتدای آغاز به کار الگوریتم، با روشی که در بخش جواب‌های اولیه توضیح داده شد، L جواب اولیه به دست می‌آید که آن‌ها را در لیست AM قرار می‌دهیم و برحسب تابع هدف‌شان آنها را مرتب می‌کنیم. در تکرارهای بعدی الگوریتم و با تولید جواب‌های جدید، این جواب‌ها را نیز در AM قرار می‌دهیم و سپس عملیات مرتب‌سازی برحسب توابع هدف تکرار می‌شود.

الف) نحوه‌ی انتخاب یک طرح چیدمان از حافظه‌ی تطابقی برای تولید جواب‌های جدید

نحوه‌ی انتخاب یک طرح چیدمان از AM برای ساختن جوابی در همسایگی آن، در واقع هر دو عمل قدرت‌مندسازی و متنوع‌سازی را انجام می‌دهد. در این الگوریتم برای انتخاب یک طرح چیدمان از AM از یک رویکرد احتمالی بهره می‌گیریم. در این رویکرد احتمالی، جواب‌های بهتر با احتمال بیشتری انتخاب می‌شوند.

با این رویکرد الگوریتم در واقع اهمیت بیشتری برای جواب‌های خوب قائل شده و در اطراف این جواب‌ها به‌دنبال جواب‌های بهتر می‌گردد. از طرف دیگر، در این رویکرد عملیات متنوع‌سازی نیز انجام می‌شود چرا که انتخاب جواب‌هایی که قابل رقابت با جواب‌های بسیار خوب نیستند نیز محتمل است. بدین‌ترتیب الگوریتم حول این جواب‌ها نیز به جست‌وجوی جواب‌های مناسب می‌پردازد.

با گذشت زمان و تداوم فرایند حل، AM به‌تدریج با جواب‌های بهتر پر شده و جواب‌های نامناسب از آن حذف می‌شوند. بدین‌ترتیب، با ادامه‌ی فرایند حل از اهمیت متنوع‌سازی کاسته شده (چرا که جواب‌های نامناسب از AM حذف می‌شوند) و به عملیات قدرت‌مندسازی اهمیت بیشتری داده می‌شود (چرا که AM با جواب‌های بهتر انباشته می‌شود). با توجه به این توضیحات، توسعه‌ی تابع احتمال برای انتخاب مناسب جواب‌ها از AM اهمیت زیادی دارد.

در این الگوریتم از یک تابع احتمالِ معرفی‌شده استفاده می‌کنیم[۲۲] که برای iامین طرح چیدمان نامناسب موجود در AM احتمال انتخابی برابر $\frac{۲i}{|L|\times|L+۱|}$ را در نظر می‌گیرد. بدین‌ترتیب، طرح چیدمانی که در انتهای لیست AM قرار دارد احتمال انتخابی برابر $\frac{۲}{|L|\times|L+۱|}$ خواهد داشت و طرحی که در ابتدای لیست AM قرار دارد احتمال انتخابی برابر $\frac{۲\times|L|}{|L|\times|L+۱|}$ را دارد.

همچنین در الگوریتم پیشنهادی، شرط توقف عبارت است از توقف بعد از k بار تکرار الگوریتم.

۴. قدرت‌مندسازی نهایی

برای حصول اطمینان از انجام یک جست‌وجوی کامل‌تر در همسایگی بهترین جوابِ به دست آمده، پس از توقف اجرای الگوریتم TS و انتخاب بهترین جواب موجود

از AM، از این جواب قسمت‌های مجاور را انتخاب کرده و محل آن‌ها را در طرح چیدمان با یکدیگر عوض می‌کنیم.

اگر یک جابه‌جایی باعث کاهش مقدار تابع هدف شود، این جابه‌جایی پذیرفته خواهد شد و کلیه‌ی مراحل قدرت‌مندسازی نهایی از ابتدا تکرار می‌شوند. پس از انجام این جابه‌جایی‌ها، طرح چیدمان نهایی به‌عنوان جواب مسئله پذیرفته می‌شود.

۵. فضای خالی لازم بین قسمت‌ها

در تمامی مسائلی که از ادبیات مسئله‌ی SRFLP استخراج شده‌اند، میزان فضای خالی لازم بین تمامی قسمت‌ها برابر صفر بوده است. بنابراین در الگوریتم پیشنهادی نیز این فضا برابر صفر فرض شده است.

خلاصه‌ی گام‌های الگوریتم

گام ۱: مقادیر پارامترهای θ (طول لیست ممنوع)، a (میزان تلاش برای یافتن جواب بهتر در همسایگیِ جواب موجود)، L (طول لیست AM) و k (تعداد دفعات تکرار الگوریتم) را تعیین کنید.

گام ۲: با کمک مطالب بخش ۱، به تعداد مورد نیاز برای مسئله جواب اولیه یافته (یعنی L جواب) و در AM قرار دهید. سپس حافظه‌ی تطابقی را برحسب برچسب هر جواب، که در واقع مقدار تابع هدف ABSMODEL برای آن جواب است، به‌طور صعودی مرتب کنید.

گام ۳: احتمال انتخاب iامین جواب نامناسب در AM که قصد داریم از آن در گام‌های آتی استفاده کنیم، برابر $\frac{2i}{|L|\times|L+1|}$ است. با توجه به این مقادیر احتمال، یک جواب را از لیست AM انتخاب کنید.

گام ۴: دو قسمت از جواب فوق را به‌طور تصادفی انتخاب، و جای آن‌ها را در طرح چیدمان با یکدیگر عوض کنید. اگر مقدار تابع هدف مسئله با این جابه‌جایی بهبود یابد، این تغییر مورد پذیرش واقع می‌شود. فرض کنید بین قسمت‌های i و j جابه‌جایی صورت گرفته است. زوج غیر مرتب (i, j) را به لیست ممنوع اضافه کنید. این امر بدان‌معناست که جابه‌جایی قسمت‌های i و j در θ تکرار بعدی ممنوع است مگر این که این جابه‌جایی جوابی را تولید کند که مقدار تابع هدف آن از کلیه‌ی جواب‌های کنونی بهتر باشد یا جابه‌جایی هیچ دو قسمت دیگری باعث بهبود جواب موجود نشود.

در صورتی که با این جابه‌جایی مقدار تابع هدف بهبود نیابد، دو قسمت دیگر را به‌طور تصادفی انتخاب، و گام ۴ را تکرار کنید. این فرایند حداکثر a بار تکرار می‌شود. در صورتی که با این تعداد تکرار جواب مورد پذیرش جدیدی تولید شود به گام ۵ بروید، و در غیر این‌صورت به گام ۳ بازگردید.

گام ۵: جواب تولیدشده‌ی جدید را به‌همراه برچسب آن به لیست AM اضافه کنید. این لیست را به‌طور صعودی مرتب کرده و بدترین جواب موجود را از آن حذف کنید. این عمل باعث می‌شود طول این لیست از پارامتر L بیشتر نشود. سپس به گام ۶ بروید. دقت شود که وجود جواب‌های تکراری در لیست AM پذیرفتنی است و عملاً موجب‌گذار از فرایند متنوع‌سازی به فرایند قدرت‌مندسازی می‌شود.

گام ۶: شرایط توقف را بررسی کنید. درصورتی که شرایط توقف ارضاء شده باشند بهترین جواب موجود در AM از لحاظ تابع هدف را انتخاب کرده و به گام ۷ بروید. در غیر این‌صورت به گام ۳ بازگردید.

گام ۷: محل قسمت‌های مجاور در جواب به دست آمده از گام ۶ را با یکدیگر جابه‌جا کنید. هر جابه‌جایی که تابع هدف را بهبود دهد پذیرفته می‌شود و در غیر این‌صورت مورد پذیرش قرار نمی‌گیرد. اگر یک جابه‌جایی مورد پذیرش قرارگیرد، این گام از ابتدا و مجددا اجرا می‌شود. پس از انجام تمامی این جابه‌جایی‌ها، طرح چیدمان به دست آمده را به‌عنوان جواب نهایی مسئله تعیین کنید.

نتایج محاسباتی

الگوریتم پیشنهادی دارای ۴ پارامتر است که پیش از اجرای الگوریتم باید مقادیر آن‌ها مشخص شود. این پارامترها عبارت‌اند از:

- طول AM که آن را با L نمایش می‌دهیم؛
- طول لیست ممنوع که آن را با θ نمایش می‌دهیم؛
- تعداد دفعات تلاش برای بهبود یک جواب خاص انتخاب‌شده از AM که آن را با a نمایش می‌دهیم؛
- تعداد دفعات تکرار الگوریتم که آن را با k نمایش می‌دهیم.

در الگوریتم پیشنهادی، برای یافتن مقادیر مناسب پارامترهای ذکرشده، آزمایش‌های فراوانی بر روی مسائل مختلف و با مقادیر متفاوت پارامترها طراحی و انجام شده است. در نهایت با توجه به نتایج به دست آمده، بهترین مقادیر ممکن برای پارامترها که کیفیت جواب‌های تولیدشده توسط این مقادیر از کیفیت سایر جواب‌ها بهتر است، عبارت‌اند از: $L = 2\frac{n}{3}; \theta = \frac{n}{3}; a = \frac{n}{2}; k = 50n$ برای پیاده‌سازی الگوریتم از زبان برنامه‌نویسی Borland C++ استفاده شده و برای اجرای الگوریتم نیز از یک دستگاه رایانه‌ی قابل حمل با سرعت پردازنده‌ی ۲ گیگاهرتز و مقدار حافظه (RAM) ۱ گیگابایت بهره‌گیری شده است.[۲۳]

مسائل موجود در مقالات را می‌توان به دو دسته‌ی عمده تقسیم کرد: مسائلی که جواب بهینه برای آن‌ها پیدا شده، و مسائلی که هنوز جواب بهینه‌ی آن‌ها پیدا نشده است. درمورد مسائل دسته‌ی اول، در جدول ۱ موارد مورد بررسی به‌ترتیب عبارت‌اند از: منبعی که مسئله در آن معرفی شده، تعداد قسمت‌های مسئله، بهترین جواب غیردقیق که تاکنون برای این مسئله در مقالات مختلف پیدا شده، زمان صرف‌شده برای به دست آمدن این جواب برحسب ثانیه و منبع آن، جوابی که توسط الگوریتم پیشنهادی برای این مسئله یافت شده و زمان صرف‌شده برحسب ثانیه برای رسیدن به این جواب، مقدار تابع هدف جواب بهینه و منبع معرفی این جواب. در ستون آخر جدول ۱ نیز درصد تفاوت تابع هدف به دست آمده از الگوریتم پیشنهادی و تابع هدف جواب بهینه آورده شده است. از آنجا که الگوریتم پیشنهادی الگوریتمی با ساختارهای تصادفی است، هر مسئله پنج‌بار توسط آن حل شده و بهترین جواب حاصل از این پنج اجرا و زمان حل مربوط به آن به‌عنوان جواب نهایی در جداول آورده شده است. همچنین گروهی از مسائل در جدول ۱ وجود دارند که تاکنون در ادبیات علمی مسئله توسط هیچ الگوریتم غیردقیقی مورد بررسی قرار نگرفته‌اند و طبیعتاً در این جدول در ردیف‌های مربوطه برای آن‌ها اطلاعاتی ذکر نشده است.

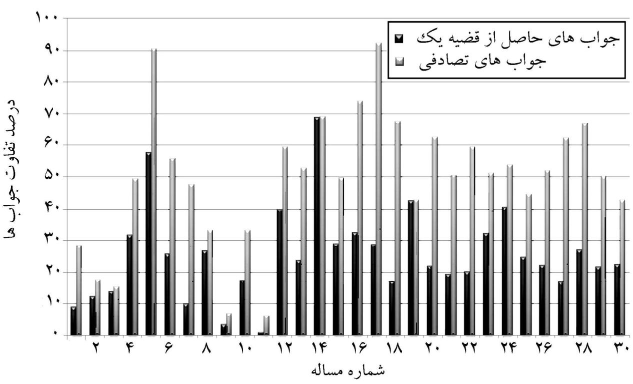
در جدول ۱، بیشترین میزان تفاوت جواب‌های به دست آمده از الگوریتم پیشنهادی و جواب بهینه ۲٫۵ درصد است و الگوریتم پیشنهادی توانسته تابع هدف به دست آمده از تمامی الگوریتم‌های غیردقیق قبلی را در مدت زمانی بسیار کم‌تر تکرار کند. در نمودار ۱ نقش قضیه‌ی ۱ در تولید جواب‌های اولیه نشان داده شده است. برای کشیدن این نمودار مقدار تابع هدف اولین جوابی که از طرح چیدمان

جدول ۱. نتایج محاسباتی درمورد مسائلی که جواب بهینه‌ی آن‌ها موجود است.

شماره مساله	مرجع	تعداد دپارتمان	بهترین جواب غیر دقیق تا حال حاضر			روش پیشنهادی		جواب بهینه		فاصله (٪)
			OFV	زمان	مرجع	OFV	زمان	OFV	مرجع	
۱	[۲۴]	۴	۷۸٫۰۰	۰٫۰۰	[۱۴]	۷۸٫۰۰	۰٫۰۰	۷۸٫۰۰	[۲۴]	۰
۲	[۲۱]	۴	-	-	-	۶۳۸٫۰۰	۰٫۰۰	۶۳۸٫۰۰	[۲۱]	۰
۳	[۲۵]	۵	-	-	-	۱۵۱٫۰۰	۰٫۰۰	۱۵۱٫۰۰	[۲۱]	۰
۴	[۲۶]	۵	۱٫۱۰۰	۰٫۰۰	[۱۴]	۱٫۱۰۰	۰٫۰۰	۱٫۱۰۰	[۱۸]	۰
۵	[۲۶]	۶	۱٫۹۹۰	۰٫۰۰	[۱۴]	۱٫۹۹۰	۰٫۰۰	۱٫۹۹۰	[۱۸]	۰
۶	[۲۶]	۷	۴٫۷۳	۰٫۰۰	[۱۴]	۴٫۷۳	۰٫۰۰	۴٫۷۳	[۱۸]	۰
۷	[۲۶]	۸	۶٫۲۹۵	۰٫۰۰	[۱۴]	۶٫۲۹۵	۰٫۰۰	۶٫۲۹۵	[۱۸]	۰
۸	[۴]	۸	-	-	-	۸۰۱٫۰۰	۰٫۰۰	۸۰۱٫۰۰	[۱۸]	۰
۹	[۴]	۸	۲,۳۲۴٫۵۰	۰٫۰۰	[۱۴]	۲,۳۲۴٫۵۰	۰٫۰۰	۲,۳۲۴٫۵۰	[۱۸]	۰
۱۰	[۴]	۹	-	-	-	۲,۴۶۹٫۵۰	۰٫۰۰	۲,۴۶۹٫۵۰	[۱۸]	۰
۱۱	[۴]	۹	-	-	-	۴,۶۹۵٫۵۰	۰٫۰۰	۴,۶۹۵٫۵۰	[۱۸]	۰
۱۲	[۴]	۱۰	۲,۷۸۱٫۵۰	۰٫۰۱	[۱۴]	۲,۷۸۱٫۵۰	۰٫۰۰	۲,۷۸۱٫۵۰	[۱۸]	۰
۱۳	[۴]	۱۱	۶,۹۳۳٫۵۰	۰٫۰۳	[۱۴]	۶,۹۳۳٫۵۰	۰٫۰۱۶	۶,۹۳۳٫۵۰	[۱۸]	۰
۱۴	[۶]	۱۲	-	-	-	۱۸,۱۹۰٫۰	۰٫۰۱۷	۱۸,۱۴۰٫۲۳	[۱۸]	۰٫۲۷
۱۵	[۶]	۱۵	۴۴,۵۹۹٫۹	۰٫۱۸	[۱۴]	۴۴,۶۰۰٫۰	۰٫۰۱۶	۴۴,۶۰۰٫۰۰	[۱۸]	۰
۱۶	[۲۱]	۱۵	-	-	-	۶,۳۰۵٫۰۰	۰٫۰۳۱	۶,۳۰۵٫۰۰	[۱۸]	۰
۱۷	[۶]	۲۰	۱۱۹٫۷۱۰	۱٫۸	[۱۴]	۱۱۹٫۷۱۰	۰٫۰۳۴	۱۱۹٫۷۱۰	[۱۸]	۰
۱۸	[۶]	۲۰	۱۵,۵۴۹٫۰	۲٫۳	[۱۴]	۱۵,۵۴۹٫۰	۰٫۰۲۰	۱۵,۵۴۹٫۰۰	[۱۸]	۰
۱۹	[۱۸]	۲۵	-	-	-	۴,۶۳۱٫۰۰	۰٫۰۳۸	۴,۶۱۸٫۰۰	[۱۸]	۰٫۲۸
۲۰	[۱۸]	۲۵	-	-	-	۳۷,۱۱۶٫۵	۰٫۰۳۹	۳۷,۱۱۶٫۵۰	[۱۸]	۰
۲۱	[۱۸]	۲۵	-	-	-	۲۴,۵۶۰٫۰	۰٫۰۴۷	۲۴,۳۰۱٫۰۰	[۱۸]	۱٫۰۶
۲۲	[۱۸]	۲۵	-	-	-	۴۸,۲۹۱٫۵	۰٫۰۴۰	۴۸,۲۹۱٫۵۰	[۱۸]	۰
۲۳	[۱۸]	۲۵	-	-	-	۱۵,۶۲۳٫۰	۰٫۰۴۱	۱۵,۶۲۳٫۰۰	[۱۸]	۰
۲۴	[۱۸]	۳۰	-	-	-	۸,۲۴۷٫۰۰	۰٫۰۵۲	۸,۲۴۷٫۰۰	[۱۸]	۰
۲۵	[۱۸]	۳۰	-	-	-	۲۱,۵۸۲٫۵	۰٫۰۵۱	۲۱,۵۸۲٫۵۰	[۱۸]	۰
۲۶	[۱۸]	۳۰	-	-	-	۴۶,۲۱۲٫۰	۰٫۰۶۲	۴۵,۴۴۹٫۰۰	[۱۸]	۱٫۶۸
۲۷	[۱۸]	۳۰	-	-	-	۵۸,۲۹۷٫۵	۰٫۰۵۹	۵۶,۸۷۳٫۵۰	[۱۸]	۲٫۵
۲۸	[۱۸]	۳۰	-	-	-	۱۱۵,۸۲۶٫۰	۰٫۰۶۸	۱۱۵,۲۶۸٫۰	[۱۸]	۰٫۴۸
۲۹	[۶]	۳۰	۳۳۴٫۸۹۶۸	۳۷٫۳	[۱۴]	۳۳۴٫۸۷۰	۰٫۰۷۷	۳۳۴٫۸۷۰	[۱۸]	۰
۳۰	[۶]	۳۰	۴۴,۴۶۶٫۵	۹۱٫۸	[۱۲]	۴۵,۱۲۶٫۰	۰٫۰۶۶	۴۴,۹۶۵٫۰۰	[۱۸]	۰٫۳۶

قضیه‌ی ۱ به دست می‌آید، میزان تفاوت آن با بهترین جواب موجود، میانگین تابع هدف حاصل از ۳ جواب تصادفی و میزان تفاوت آن با بهترین جواب موجود محاسبه شده و سپس این درصد تفاوت در نمودار آورده شده است. نمودار ۱ در واقع نقش استفاده از قضیه‌ی ۱ برای تولید جواب‌های اولیه‌ی مناسب را به‌خوبی نشان می‌دهد.

با این که قضیه‌ی ۱ جواب بهینه‌ی مسئله را فقط در یک حالت خاص تولید کرده و به‌کارگیری آن تضمینی برای مناسب‌بودن جواب‌ها در حالت کلی نیست، ملاحظه می‌شود که در نمودار ۱ حتی یک مورد وجود ندارد که در آن میانگین سه تابع هدفِ تصادفی از تابع هدفِ طرح چیدمان قضیه‌ی ۱ بهتر باشد. در تمامی موارد طرح چیدمان حاصل از قضیه‌ی ۱ بسیار بهتر از روش‌های تصادفی است به‌طوری که میانگین تفاوت جواب تولیدشده توسط قضیه‌ی ۱ و جواب بهینه در مسائل جدول ۱ برابر ۲۵٫۴۰ درصد است در حالی که میانگین فاصله‌ی جواب‌های تصادفی و جواب بهینه برابر ۴۹٫۶۷ درصد است.



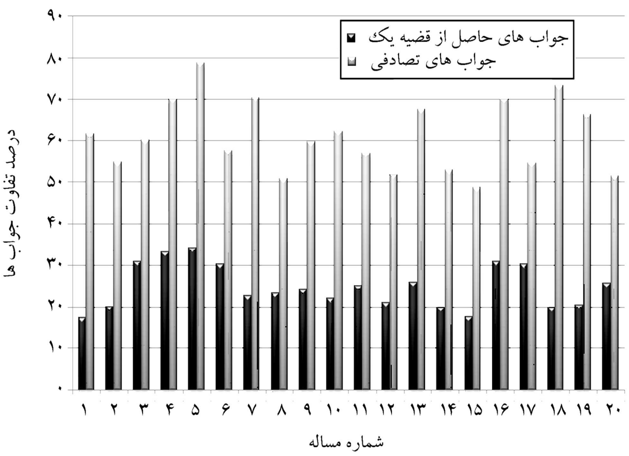
نمودار ۱. مقایسه‌ی جواب‌های حاصل از قضیه‌ی ۱ و جواب‌های تصادفی در مورد مسائل جدول ۱.

جدول ۲. نتایج محاسباتی برای مسائلی که جواب بهینه برای آن‌ها موجود نیست.

شماره مساله	تعداد دپارتمان	الگوریتم پیشنهادی OFV	الگوریتم پیشنهادی زمان(ثانیه)	Anjos et al. OFV [15]	Anjos et al. زمان CPU [15]
۱	۶۰	**۱,۴۷۷,۸۴۰٫۰۰**	۰٫۸۲۰	۱,۴۹۳,۷۰۴٫۰۰	۵ ساعت
۲	۶۰	**۸۴۲,۸۴۲٫۰۰**	۰٫۹۸۰	۸۴۳,۶۴۴٫۰۰	۵ ساعت
۳	۶۰	**۶۴۹,۹۶۶٫۵**	۰٫۹۰۰	۶۵۶,۲۷۲٫۵۰	۵ ساعت
۴	۶۰	**۴۰۰,۷۳۲٫۰**	۰٫۹۱۳	۴۰۵,۴۳۳٫۰۰	۵ ساعت
۵	۶۰	۳۱۹,۸۹۵٫۰۰	۰٫۷۶۲	**۳۱۹,۵۰۱٫۰۰**	۵ ساعت
۶	۷۰	۱,۵۵۱,۱۳۱٫۰۰	۱٫۴۹۹	**۱,۵۴۳,۰۹۸٫۰۰**	۵ ساعت
۷	۷۰	**۱,۴۴۲,۳۲۱٫۰**	۱٫۹۴۰	۱,۴۹۴,۱۸۲٫۰۰	۵ ساعت
۸	۷۰	۱,۵۲۵,۱۳۲٫۵۰	۱٫۷۶۱	**۱,۵۲۴,۱۷۱٫۵۰**	۵ ساعت
۹	۷۰	**۹۷۱,۵۷۷٫۰۶**	۱٫۲۳۳	۹۷۴,۸۵۶٫۰۰	۷ ساعت
۱۰	۷۰	**۴,۲۲۰,۴۰۴٫۵**	۱٫۵۷۰	۴,۲۳۰,۹۱۲٫۵۰	۷ ساعت
۱۱	۷۵	۲,۴۰۸,۲۷۰٫۵۰	۲٫۰۱۰	**۲,۳۹۹,۵۸۳٫۵۰**	۱۰ ساعت
۱۲	۷۵	**۴,۳۲۷,۰۲۷٫۰**	۲٫۱۹۸	۴,۳۴۸,۵۴۴٫۰۰	۱۰ ساعت
۱۳	۷۵	**۱,۲۵۱,۹۶۲٫۰**	۲٫۹۱۲	۱,۲۹۵,۰۸۵٫۰۰	۱۰ ساعت
۱۴	۷۵	**۳,۹۴۷,۴۸۴٫۵۰**	۲٫۵۱۶	۳,۹۴۹,۲۷۶٫۵۰	۱۰ ساعت
۱۵	۷۵	**۱,۷۹۱,۴۰۸٫۰۰**	۲٫۰۹۸	۱,۸۱۶,۴۵۵٫۰۰	۱۰ ساعت
۱۶	۸۰	**۲,۰۷۱,۳۳۶٫۵**	۳٫۹۷۵	۲,۱۳۸,۰۸۳٫۵۰	۱۰ ساعت
۱۷	۸۰	**۱,۹۲۶,۷۴۸٫۰۰**	۵٫۶۴۱	۱,۹۳۹,۹۳۸٫۰۰	۱۰ ساعت
۱۸	۸۰	**۳,۲۵۹,۷۲۰٫۰**	۴٫۷۹۷	۳,۳۳۲,۴۲۱٫۰۰	۱۰ ساعت
۱۹	۸۰	**۳,۷۶۹,۵۵۰٫۰۰**	۳٫۴۵۳	۳,۷۷۳,۴۲۹٫۰۰	۱۰ ساعت
۲۰	۸۰	**۱,۵۹۴,۶۶۴٫۰**	۳٫۷۶۹	۱,۶۱۱,۴۹۵٫۰۰	۱۰ ساعت
میانگین	-	-	**۲٫۲۹**	-	**۸ ساعت**

در جدول ۲ به بررسی مسائل بزرگ‌تری می‌پردازیم که جواب بهینه‌یی برای آن‌ها موجود نیست.[15] برای این مسائل جواب‌هایی یافت شده است که در ادامه خواهند آمد. در این جدول اعداد پررنگ نشان‌دهنده‌ی بهترین جواب یافت‌شده برای این مسائل‌اند. چنان‌که از جدول ۲ پیداست، الگوریتم پیشنهادی موجب بهبود جواب‌های ۱۶ مسئله از ۲۰ مسئله‌ی مورد بررسی است و درمورد بقیه‌ی مسائل نیز جواب یافت‌شده توسط آن به‌میزان ناچیزی بیشتر است. نکته این که بیشتر جواب‌های بهبود داده شده مربوط به مسائل با نمودهای بزرگ هستند که خود نشان‌دهنده‌ی قدرت الگوریتم پیشنهادی در مواجهه با مسائل بزرگ است. به‌علاوه زمان صرف‌شده برای حل مسائل توسط الگوریتم پیشنهادی (در مورد تمامی ۲۰ مسئله‌ی مورد بررسی) به‌هیچ عنوان قابل مقایسه با الگوریتم معرفی شده[15] نیست. به‌عنوان مثال، بیشترین زمان صرف‌شده توسط الگوریتم پیشنهادی برای حل مسئله‌ی ۱۷، و برابر ۵٫۶۴۱ ثانیه بوده است. این در حالی است که الگوریتم معرفی‌شده[15] دست کم به ۵ ساعت زمان برای حل مسائل کوچک‌تر نیاز دارد.

در نمودار ۲ نقش قضیه‌ی ۱ در تولید جواب‌های اولیه نشان داده شده است.



نمودار ۲. مقایسه‌ی جواب‌های قضیه‌ی ۱ و جواب‌های تصادفی در مورد مسائل جدول ۲.

جدول ۳. مسائل جدید با نمودهای بسیار بزرگ.

شماره مساله	تعداد تجهیزات	OFV	زمان CPU (ثانیه)	تعداد تجهیزات با طول یکتا	درصد عناصر صفر در ماتریس جریان
۱	۱۰۰	۲۱۱,۷۸۹,۷۱۲/۰۰	۶/۶۱۲	۶۱	۱/۸۵
۲	۱۰۰	۱۷۷,۶۵۹,۹۹۱/۵۰	۶/۲۰۶	۵۱	۱/۷۷
۳	۱۰۰	۲۱۲,۹۷۴,۰۳۳/۰۰	۵/۷۷۵	۵۸	۲
۴	۱۰۰	۲۰۰,۴۴۷,۲۹۲/۰۰	۶/۳۴۸	۵۶	۱/۷۹
۵	۱۰۰	۲۰۶,۰۴۲,۳۹۰/۰۰	۵/۸۹۵	۶۱	۱/۸۶
۶	۱۲۰	۳۳۲,۳۲۹,۷۴۴/۵۰	۷/۹۷۶	۶۰	۱/۸۸
۷	۱۲۰	۳۵۸,۹۵۵,۵۰۵/۰۰	۷/۹۷۴	۶۳	۱/۸۶
۸	۱۲۰	۳۴۰,۳۶۰,۹۰۸/۵۰	۷/۱۴۹	۶۳	۲
۹	۱۲۰	۳۲۲,۰۱۵,۱۷۲/۵۰	۶/۳۳۵	۶۲	۱/۹۵
۱۰	۱۲۰	۳۵۰,۸۸۸,۶۳۳/۰۰	۶/۸۴۳	۵۳	۲/۱۶
۱۱	۱۵۰	۶۸۱,۸۸۶,۴۰۶/۵۰	۲۵/۹۹۹	۵۹	۱/۹۶
۱۲	۱۵۰	۶۵۱,۳۹۸,۶۰۰/۵۰	۲۴/۰۸۲	۵۹	۲/۰۱
۱۳	۱۵۰	۶۵۲,۴۸۴,۵۶۳/۰۰	۲۶/۶۶	۶۲	۱/۸۸
۱۴	۱۵۰	۷۱۰,۸۹۲,۴۵۱/۵۰	۲۵/۶۵۲	۵۷	۲/۱۷
۱۵	۱۵۰	۶۹۱,۶۹۹,۷۲۱/۵۰	۲۷/۲۵	۶۰	۲/۰۱
۱۶	۲۰۰	۱,۶۶۱,۱۵۶,۸۷۲/۰۰	۴۹/۲۸۸	۶۳	۱/۷۷
۱۷	۲۰۰	۱,۵۶۶,۱۳۸,۵۴۳/۵۰	۵۰/۲۹۹	۶۱	۲/۲۵
۱۸	۲۰۰	۱,۶۵۳,۰۶۷,۷۱۶/۵۰	۴۸/۰۵۲	۶۰	۲/۱۰
۱۹	۲۰۰	۱,۴۸۶,۶۷۰,۱۲۴/۰۰	۴۷/۳۰۵	۵۸	۱/۹۸
۲۰	۲۰۰	۱,۷۵۷,۰۷۶,۹۲۶/۵۰	۴۸/۸۲۶	۶۱	۲/۰۲
میانگین			**۲۲/۰۳**	**۵۹/۴**	**۱/۹۶**

برای کشیدن این نمودار مقدار تابع هدف اولین جوابی که از طرح چیدمان قضیه‌ی ۱ به دست می‌آید؛ میزان تفاوت آن با بهترین جواب موجود، میانگین تابع هدف حاصل از سه جواب تصادفی و میزان تفاوت آن با بهترین جواب موجود محاسبه‌شده، و سپس این درصد تفاوت در نمودار آورده شده است. نمودار ۲ در واقع نقش استفاده از قضیه‌ی ۱ برای تولید جواب‌های اولیه‌ی مناسب را به‌خوبی نشان می‌دهد.

در این دسته مسائل نیز نقش بسیار مثبت استفاده از قضیه‌ی ۱ برای راه‌اندازی الگوریتم مشخص است. بیشترین مقدار تفاوت تابع هدف به دست آمده از به کارگیری قضیه‌ی ۱ با بهترین جواب موجود در کلیه‌ی نمودهای مسائل جدول ۲ برابر است با ۳۴/۱۸ درصد. حال آن‌که کم‌ترین تفاوت میانگین سه تابع هدف به دست آمده از طرح‌های چیدمان تصادفی ۴۸/۶۶ درصد است و بیشینه‌ی آن به ۷۸/۴۵ درصد می‌رسد. به‌علاوه در نمودار ۲ حتی یک مورد وجود ندارد که در آن میانگین سه تابع هدف تصادفی بهتر از تابع هدف طرح چیدمان حاصل از قضیه‌ی ۱ باشد، و در تمامی موارد طرح چیدمان حاصل از قضیه‌ی ۱ بسیار بهتر از روش‌های تصادفی است. میانگین تفاوت جواب حاصل از قضیه‌ی ۱ و جواب بهینه در مسائل جدول ۱ برابر ۲۴/۸۱ درصد است در حالی که میانگین فاصله‌ی جواب‌های تصادفی و جواب بهینه برابر ۶۰/۹۱ درصد است.

چنان که ملاحظه می‌شود، این میزان تفاوت نسبت به آنچه در نمودار ۱ وجود دارد به‌نحو چشم‌گیری بیشتر است. این مسئله نشان می‌دهد که با افزایش تعداد قسمت‌ها و بزرگ‌تر شدن نمود مسئله، طرح‌های به‌دست آمده از قضیه‌ی ۱ از طرح‌های تصادفی بیشتر فاصله می‌گیرد و کیفیت آن‌ها بهبود می‌یابد.

همچنین در این تحقیق، برای نشان‌دادن کارآیی محاسباتی الگوریتم TS پیشنهادی تعداد ۲۰ مسئله‌ی جدید با نمودهای بسیار بزرگ طراحی و حل شده‌اند. در طراحی این مسائل، طول تجهیزات یک عدد صحیح تصادفی و بین ۲۰ تا ۱۰۰ بوده، در حالی که جریان میان قسمت‌ها اعداد صحیح تصادفی بین ۰ تا ۵۰ فرض شده‌اند. جدول ۳ به بررسی این مسائل اختصاص دارد. در این جدول درمورد هر مسئله اطلاعاتی مبنی بر تعداد قسمت‌های دارای طول یکتا در میان تمامی اعداد مربوط به طول و درصدی از ماتریس جریان که برابر صفر بوده، آورده شده است.

نتیجه‌گیری

مسئله‌ی SRFLP یک مسئله‌ی NP-Complete است که با توجه به کاربردهای عملی فراوان، تلاش‌های زیادی برای حل آن توسط الگوریتم‌های دقیق و غیردقیق صورت گرفته است. با این‌که در سال‌های اخیر روش‌های دقیق و غیردقیقی برای حل SRFLP توسعه داده شده است، نیاز به الگوریتمی که بتواند جواب‌های نزدیک به بهینه‌ی نمودهای بزرگ مسئله را در زمان اندک تولید کند، به‌شدت احساس می‌شود. در این نوشتار الگوریتمی جدید برمبنای روش TS برای حل این مسئله پیشنهاد شده است. جواب‌های اولیه برای راه‌اندازی این الگوریتم پاسخ‌های بهینه‌ی مسئله‌ی SRFLP در یک حالت خاص هستند که بهینگی آن‌ها در قضیه‌ی ۱ اثبات شده است.

نتایج محاسباتی حاکی از کارآیی الگوریتم پیشنهادی در مقایسه با سایر الگوریتم‌ها، و نیز کارآیی جواب‌های تولیدشده به‌کمک قضیه‌ی ۱ در مقایسه با جواب‌های تصادفی‌اند. الگوریتم پیشنهادی قادر است جواب‌های نزدیک به بهینه‌ی مسائل SRFLP را که شامل ۲۰۰ قسمت هستند، در کم‌تر از ۱ دقیقه به دست آورد. این در حالی است که بزرگ‌ترین نمود مسئله‌ی SRFLP که با سایر روش‌ها حل شده است، شامل ۸۰ قسمت است و حل نزدیک به بهینه‌ی آن توسط روش‌های مذکور به ۱۰ ساعت زمان نیاز دارد.

پانوشت

1. single row facility layout problem (SRFLP)
2. tabu search
3. ordering
4. automated guided vehicle
5. flexible manufacturing system
6. linear mixed-integer programming
7. penalty
8. simulated annealing algorithm
9. hybrid
10. greedy algorithm
11. ant-colony optimization algorithm
12. quadratic assignment problem
13. semi-definite programming
14. short term memory
15. diversification
16. intensification
17. aspiration criterion
18. adaptive memory
19. tabu list

منابع

1. Tompkins, J.A.; White, J.A.; Bozer, Y.A.; Frazelle, E.H.; Tanchoco, J.M. and Trevino, J., *Facilities Planning,* New York, Wiley (1996).
2. Chiang, W.C. and Kouvelis, P. "Improved tabu search heuristics for solving facility layout problems", *International Journal of Production Research,* (9), pp. 2565-2585 (1996).
3. Heragu, S.S., *Facilities Design,* Boston, MA: PWS Publishing Company (1997).
4. Simmons, D.M. "One dimensional space allocation: An ordering algorithm", *Operation Research,* (17), pp. 812-826 (1969).
5. Picard, J.C. and Queyranne, M. "On the one-dimensional space allocation problem", *Operation Research,* (29), pp. 371-391 (1981).
6. Heragu, S.S. and Kusiak, A. "Efficient models for the facility layout problems", *European Journal of Operation Research,* (53), pp. 1-13 (1991).
7. Suresh, G. and Sahu, S. "Multiobjective facility layout using simulated annealing", *International Journal of Production Economics,* **32**, pp. 239-254 (1993).
8. Neghabat, F. "An efficient equipment layout algorithm", *Operation Research,* (22), pp. 622-628 (1974).
9. Drezner, Z. "A heuristic procedure for the layout of a large number of facilities", *Management Science,* **7**(33), pp. 907-915 (1987).
10. Heragu, S.S. and Alfa, A.S. "Experimental analysis of simulated annealing based algorithms for the facility layout problem", *European Journal of Operation Research,* (57), pp. 190-202 (1992).
11. Braglia, M. "Optimization of a simulated-annealing-based heuristic for single row machine layout problem by genetic algorithm", *International Transactions in Operational Research,* **1**(3), pp. 37-49 (1996).
12. Kumar, K.R.; Hadjinicola, G.C. and Lin, T.L. "A heuristic procedure for the single row facility layout problem", *European Journal of Operation Research,* (87), pp. 65-73 (1995).
13. Ficko, M.; Brezocnik, M. and Balic, J. "Designing the layout of single- and multiple-rows flexible manufacturing system by genetic algorithms", *Journal of Material Processing Technology,* pp. 150-58 (2004).
14. Solimanpur, M.; Vrat, P. and Shankar, R. "An ant algorithm for the single row layout problem in flexible manufacturing systems", *Computers & Operations Research,* **32**, pp. 583-598 (2005).
15. Anjos, M.F.; Kennings, A. and Vannelli, A. "A semidefinite optimization approach for the single-row layout problem with unequal dimensions", *Discrete Optimization,* **2**, pp. 113-122 (2005).
16. Anjos, M.F. and Vannelli, A. "Computing globally optimal solutions for single-row layout problems using semidefinite programming and cutting planes", *INFORMS Journal on Computing*, pp. 611-617 (2008).

17. Amaral, A.S. "A new lower bound for the single row facility layout problem", *Discrete Applied Mathematics,* **157**(1), pp. 183-190 (2009).

18. Jean-Francois, C. and Gilbert, L. *Tabu Search Heuristics for the Vehicle Routing Problem, in Metaheuristic Optimization via Memory and Evolution,* A.B.R. Cesar, Editor, Kluwer Academic Publishers: Boston/Dordrecht/London, pp. 145-164 (2005).

19. Glover, F. "Future paths for integer programming and links to artificial intelligence", *Computers & Operations Research,* **13**, pp. 533-549 (1986).

20. Glover, F. and Laguna, M. "Tabu search", *Kluwer Academic Publishers Boston/Dordrecht/London,* **1**, (1997).

21. Amaral, A.R.S. "On the exact solution of a facility layout problem", *European Journal of Operation Research,* (173), pp. 508-518 (2006).

22. Rochat, Y. and Taillard, E. "Probabilistic diversification and intensification in local search for vehicle routing", **1** pp. 147-167 (1995).

23. Anjos, M.F. and Kong, C., *FLP Database,* (2007), Available from: http://flplib.uwaterloo.ca/.

24. Beghin-Picavet, M. and Hansen, P. "Deux problems d'affectation non lineaires", *RAIRO Recherche Operationelle,* **16**(3), pp. 263-276 (1982).

25. Love, R.F. and Wong, J.Y. "On solving a one-dimensional allocation problem with integer programming", *Information Processes and Operation Research (INFOR),* (14), pp. 139-143 (1976).

26. Nugent, C.E.; Vollman, T.E. and Ruml, J. "An experimental comparison of techniques for the assignment of facilities to locations", *Operations Research,* **16**, pp. 150-173 (1968).