
بسم الله
الرحمن الرحیم

دانشکده علوم ریاضی

رشته ریاضی کاربردی، گرایش تحقیق در عملیات

پایان‌نامه کارشناسی ارشد

الگوریتم‌هایی برای حل مسأله کوله‌پشتی چند هدفه

نگارنده: مرضیه عباسی

استاد راهنما

دکتر جعفر فتحعلی

استاد مشاور

دکتر مهرداد غزنوی

بهمن ۱۳۹۷

باسمه‌تعالی

مدیریت تحصیلات تکمیلی

فرم شماره (۳) صورتجلسه نهایی دفاع از پایان نامه دوره کارشناسی ارشد

با نام و یاد خداوند متعال، ارزیابی جلسه دفاع از پایان نامه کارشناسی ارشد خانم **مرضیه عباسی** با شماره دانشجوی **۹۵۳٤۷۵٤** رشته ریاضیات و کاربردها گرایش تحقیق در عملیات تحت عنوان **الگوریتم‌هایی برای حل مسأله کوله‌پشتی چند هدفه** که در تاریخ **۱۳۹۷/۱۱/۱** با حضور هیأت محترم داوران در دانشگاه صنعتی شاهرود برگزار گردید به شرح ذیل اعلام می گردد:

قبول (با درجه: خیلی خوب) ☑ **مردود** ☐

نوع تحقیق: **نظری** ☑ **عملی** ☐

عضو هیأت داوران	نام ونام خانوادگی	مرتبهٔ علمی	امضاء
۱ـ استادراهنمای اول	دکتر جعفر فتحعلی	دانشیار	
۲- استادراهنمای دوم			
۳- استاد مشاور	دکتر مهرداد غزنوی	استادیار	
۴- نماینده تحصیلات تکمیلی	دکتر عبدالله آل‌هوز	استادیار	
۵- استاد ممتحن اول	دکتر محمد هادی نوری اسکندری	استادیار	
۶ـ استاد ممتحن دوم	دکتر علیرضا ناظمی	دانشیار	

نام و نام خانوادگی رئیس دانشکده: دکتر ابراهیم هاشمی

تاریخ و امضاء و مهر دانشکده:

دانشکده ریاضی

تبصره: در صورتی که کسی مردود شود حداکثر یکبار دیگر (در مدت مجاز تحصیل) می تواند از پایان نامه خود دفاع نماید (دفاع مجدد نباید زودتر از ۴ ماه برگزار شود).

تقدیم به:

روح پاک پدرم که عالمانه به من آموخت تا چگونه در عرصه زندگی، ایستادگی را تجربه نمایم

و به مادرم، دریای بی‌کران فداکاری و عشق که وجودم برایش همه رنج بود و وجودش برایم همه مهر

و به همسرم، اسطوره زندگیم، پناه خستگیم و امید بودنم.

سپاس‌گزاری

صمیمانه‌ترین و خالصانه‌ترین سپاس‌ها را با اشک شوق به پیشگاه کبریایی پروردگار مهربانم تقدیم میدارم، او که همواره مبهوت حکمتش بوده و، هستم، خاضعانه در برابر الطاف بیکرانش پیشانی خضوع و بندگی بر خاک می‌نهم.

از استاد راهنمای گرانقدرم جناب آقای دکتر جعفر فتحعلی که با راهنمایی‌های ارزنده‌شان همیشه مشوق من بوده‌اند صمیمانه سپاسگزارم.

ازخداوند متعال آرزوی طول عمر با عزت و توفیق روزافزون برای ایشان مسئلت می‌نمایم.

نهایت سپاس خود را از الطاف استاد مشاور گرانقدرم جناب آقای دکتر مهرداد غزنوی دارم، استاد ارجمندی که همواره با خلق نیکو و صبر و حوصله‌ی بی‌نظیرشان پاسخگوی تمامی سوالاتم بودند و با بزرگواری خود، نقایص این حقیر را مورد اغماض قرار می‌دادند.

از اساتید محترم جناب آقای دکتر علیرضا ناظمی و جناب آقای دکتر محمد هادی نوری اسکندری که زحمت مطالعه و داوری پایان‌نامه را به عهده گرفتند بسیار سپاسگزارم.

از استاد محترم جناب آقای دکتر عبدالله آل‌هوز نماینده تحصیلات تکمیلی که در جلسه دفاع اینجناب شرکت می‌کنند بسیار سپاسگزارم.

برای تمامی این عزیزان آرزوی سلامتی و موفقیت روزافزون از خداوند متعال مسئلت می‌نمایم.

مرضیه عباسی
بهمن ۱۳۹۷

تعهد نامه

اینجانب **مرضیه عباسی** دانشجوی کارشناسی ارشد رشته **ریاضی کاربردی علوم ریاضی** دانشگاه شاهرود، نویسنده پایان‌نامه با عنوان **الگوریتم‌هایی برای حل مسأله کوله‌پشتی چند هدفه** ، تحت راهنمایی **جعفر فتحعلی** متعهد می‌شوم:

- تحقیقات در این پایان‌نامه توسط اینجانب انجام شده است و از صحت و اصالت برخوردار است.
- در استفاده از نتایج پژوهش‌های دیگر پژوهش‌گران، به مرجع مورد استفاده استناد شده است.
- مطالب این پایان‌نامه، تا کنون توسط خود، یا فرد دیگری برای دریافت هیچ نوع مدرک یا امتیازی در هیچ‌جا ارایه نشده است.
- حقوق معنوی این اثر، به دانشگاه صنعتی شاهرود تعلق دارد، و مقالات مستخرج با نام ‘‘ دانشگاه صنعتی شاهرود ’’ یا “ Shahrood University of Technology ” به چاپ خواهد رسید.
- حقوق معنوی تمام افرادی که در به‌دست آوردن نتایج اصلی پایان‌نامه تاثیرگذار بوده‌اند، در مقالات مستخرج از پایان‌نامه رعایت می‌گردد.
- در تمام مراحل انجام این پایان‌نامه، در مواردی که از موجود زنده (یا بافت‌های آنها) استفاده شده است، ضوابط و اصول اخلاقی رعایت شده است.
- در تمام مراحل انجام این پایان‌نامه، در مواردی که به حوزه اطلاعات شخصی افراد دسترسی یافته (یا استفاده شده است)، اصل رازداری و اصول اخلاق انسانی رعایت شده است.

مرضیه عباسی
بهمن ۱۳۹۷

مالکیت نتایج و حق نشر

- تمام حقوق معنوی این اثر و محصولات آن (مقالات مستخرج، کتاب، برنامه‌های رایانه‌ای، نرم‌افزارها و تجهیزات ساخته شده) متعلق به دانشگاه صنعتی شاهرود می‌باشد. این مطلب باید به نحو مقتضی، در تولیدات علمی مربوطه ذکر شود.
- استفاده از اطلاعات و نتایج موجود در این پایان‌نامه بدون ذکر منبع مجاز نمی‌باشد.

چکیده

در این پایان‌نامه، یک روش جدید مبتنی بر فرمول‌بندی برنامه‌ریزی خطی ریاضی برای یافتن همه جواب‌های کارا مسأله کوله پشتی دوهدفه صفر و یک پیشنهاد می‌شود و تعداد محدودی از مسائل تک‌هدفه را برای پیدا کردن جواب‌های کارای مسأله کوله پشتی دو هدفه صفر و یک حل می‌شود. در پایان الگوریتم‌هایی برای حل مسأله کوله‌پشتی چندهدفه آورده شده است.

کلمات کلیدی: مسأله کوله‌پشتی دوهدفه صفر و یک، مسأله کوله‌پشتی چندهدفه، بهینه‌سازی کلنی مورچه، جستوجوی پراکنده

پیش گفتار

در این پایان‌نامه، به بیان الگوریتم‌هایی برای حل مسأله کوله‌پشتی تک هدفه، دو هدفه و چندهدفه با استفاده از روش‌های فراابتکاری می‌پردازیم.

در ابتدا با مسائل کوله‌پشتی تک هدفه، دوهدفه و چندهدفه، نقاط کارا و کارای ضعیف آشنا می‌شویم. مسأله کوله‌پشتی چندهدفه یک توسعه از مسأله کوله‌پشتی معمولی است که بیش از یک معیار را در نظر می‌گیرد.

برای حل مسأله کوله‌پشتی تک هدفه صفر ویک الگوریتم حریصانه و الگوریتم شاخه و کران استفاده شده است.

فرض کنید مجموعه‌ای از اشیا که هر کدام دارای وزن و ارزش خاصی هستند در اختیار دارید. می‌خواهیم تعدادی از آن ها را به گونه‌ای انتخاب کنیم که وزن اشیا انتخاب شده کوچکتر یا مساوی از یک مقدار از پیش تعیین شده باشد و ارزش آن‌ها بیشینه شود. علت نامگذاری این مسأله، جهانگردی است که کوله پشتی‌ای با اندازه محدود دارد و باید آن‌را با مفیدترین صورت ممکن از اشیا پر کند. از نظر ریاضی فرض کنید یک کوهنورد می‌خواهد از بین n شئ تعدادی را به همراه خود ببرد. شئ jام دارای وزن a_j و ارزش c_j است. حداکثر وزنی که می‌تواند همراه ببرد b است. او چه کالاهایی را همراه ببرد که مجموع وزن آن‌ها از b بیشتر نشود و دارای بیشترین ارزش باشند؟
برای مدل کردن مسأله تعریف می‌کنیم:

$$x_j = \begin{cases} ۱, & \text{اگر شئ } j \text{ را به همراه ببرد} \\ ۰, & \text{در غیر این صورت} \end{cases}$$

مدل مسأله کوله پشتی را می‌توان به صورت زیر نوشت:

$$Max \sum_{j=۱}^{n} c_j x_j$$

$$s.t.$$

$$\sum_{j=۱}^{n} a_j x_j \leq b$$

$$x_j \in \{۰, ۱\} \quad j \in \{۱, \cdots, n\}$$

در ادامه الگوریتمی برای حل مسائل کوله‌پشتی دو هدفه آورده شده است. مدل مسأله بصورت

$$\begin{aligned} & Max(f(x), g(x)) \\ & s.t. \\ & \sum_{i=1}^{n} a_{ij} x_{ij} \leq W_j \\ & x_{ij} \in \{0, 1\}, j = 1, \ldots, m \end{aligned} \tag{1}$$

است.

در فصل سه روشی برای حل مسائل کوله پشتی چندهدفه با استفاده از تکنیک فراابتکاری ارائه شده است. ابتدا روش اجتماع مورچگان و الگوریتمش را توصیف کرده‌ایم، سپس روش جستجوی پراکنده و الگوریتم آن‌را برای حل مسأله کوله‌پشتی آورده‌ایم.

حالت کلی مسأله کوله‌پشتی به صورت زیر است:

$$\begin{aligned} & Max(f_1(x), f_2(x), \ldots, f_s(x)) \\ & s.t. \\ & A_i x \leq b_i \qquad i = 1, 2, \cdots, m \\ & x_j \in \{0, 1\} \qquad j = 1, 2, \cdots, n \end{aligned} \tag{2}$$

که در آن برای هر $(r = 1, \ldots, s)$ $f(x) = (f_1(x), f_2(x), \ldots, f_r(x))$ و برای هر $(i = 1, \cdots, m)$ $A_i = (a_{i1}, a_{i2}, \ldots, a_{in})$ و $X = (x_1, x_2, \ldots, x_n)^T$.

در پایان با ارائه مثالی به ارزیابی روش‌ها پرداخته شده است.

فهرست مطالب

فهرست تصاویر

فهرست جداول

فصل ۱

تعاریف و مفاهیم اولیه

۱.۱ تاریخچه

در این فصل تعاریف و مفاهیم اولیه‌ای در مورد مسأله کوله پشتی تک هدفه و چند هدفه آورده شده است. مسأله کوله پشتی چند هدفه یک توسعه از مسأله کوله پشتی معمولی است که بیشتر از یک معیار را در نظر می‌گیرد. مسائل تک هدفه NP ـ کامل هستند، بنابراین قابل پیش‌بینی است که از روش‌های شمارش ضمنی مانند برنامه ریزی پویا، شاخه و کران یا روش‌های ابتکاری به خصوص روش‌های فراابتکاری برای تقریب جواب‌های کارا استفاده شود. روش‌های دیگری برای بدست آوردن مجموعه‌ای از تمام جواب‌های غیر مغلوب در تاریخچه وجود دارد. رویکرد شاخه و کران توسط وایز [۱] [1] و اخیرا توسط کپتیوو [۲] [2] و همکاران بررسی شده است.

برای حل مسأله چندهدفه صفر و یک روش‌هایی به وسیله بیتران [۳] [3] و دکرو [۴] [4] و همکاران [۴] ارائه شده است. بیتران از تکنیک آزادسازی برای تولید جواب‌های کارا استفاده کرد. او یک مسأله آزاد سازی تعریف کرد و ثابت کرد که جواب‌های کارای مسأله آزادسازی که برای مسأله اصلی شدنی هستند، می‌توانند کارا برای مسأله اصلی باشند. دکرو و همکاران نتایج

[1] Visee
[2] Captivo
[3] Bitran
[4] Deckro

محاسباتی خود را با بیتران مقایسه کردند. لیو [5] و همکاران [۵] یک روش دیگر پیشنهاد دادند و از تکنیک تحلیل پوششی داده‌ها برای تولید جواب‌های کارا استفاده کردند. آن‌ها یک واحد تصمیم گیری [6] (DMU) مربوط به هر جواب شدنی از مسأله تعریف کردند و یک الگوریتم دو مرحله‌ای برای تولید و ارزیابی DMU ها ایجاد کردند.

۲.۱ تعاریف

فرض کنید مجموعه‌ای از اشیا که هر کدام دارای وزن و ارزش خاصی هستند در اختیار دارید. می‌خواهیم تعدادی از آن ها را به گونه‌ای انتخاب کنیم که وزن اشیا انتخاب شده کوچکتر یا مساوی از یک مقدار از پیش تعیین شده باشد و ارزش آن‌ها بیشینه شود. علت نامگذاری این مسأله، جهانگردی است که کوله پشتی‌ای با اندازه محدود دارد و باید آن‌را با مفیدترین صورت ممکن از اشیا پر کند. از نظر ریاضی فرض کنید یک کوهنورد می‌خواهد از بین n شئ تعدادی را به همراه خود ببرد. شئ jام دارای وزن a_j و ارزش c_j است. حداکثر وزنی که می‌تواند همراه ببرد b است. او چه کالاهایی را همراه ببرد که مجموع وزن آن‌ها از b بیشتر نشود و دارای بیشترین ارزش باشند؟
برای مدل کردن مسأله تعریف می‌کنیم:

$$x_j = \begin{cases} ۱, & \text{اگر شئ } j \text{ را به همراه ببرد} \\ ۰, & \text{در غیر این صورت} \end{cases} \qquad (۱.۱)$$

مدل مسأله کوله پشتی را می‌توان به صورت زیر نوشت:

$$Max \sum_{j=۱}^{n} c_j x_j$$

$$s.t.$$

$$\sum_{j=۱}^{n} a_j x_j \leq b$$

$$x_j \in \{۰, ۱\} \quad j \in \{۱, \cdots, n\}$$

حال فرض کنید در مسأله کوله پشتی دو هدف مورد نظر باشد. مثلا کوهنورد بخواهد کالاهایی را با خود همراه داشته باشد که دارای بیشترین ارزش و بیشترین راحتی برای او باشد. به این گونه مسائل، مسائل کوله پشتی دو هدفه [7] می‌گویند. مدل مسأله کوله پشتی دو هدفه به

[5] Liu
[6] Decision Making Unit
[7] Bi-objective Knapsack Problem

صورت زیر می‌باشد:

$$\begin{aligned} & Max(f(x), g(x)) \\ & s.t. \\ & \sum_{j=1}^{n} a_j x_j \leq b \\ & x_j \in \{0, 1\} \quad j \in \{1, \ldots, n\} \end{aligned} \tag{۲.۱}$$

که در آن $f(x)$ و $g(x)$ توابعی هستند که باید ماکزیمم شود. تعمیمی دیگر از مسأله کوله پشتی مسأله کوله پشتی چند هدفه [۸]است که در آن به جای یک تابع هدف، چندین تابع هدف مورد بررسی قرار می‌گیرد. مسأله کوله پشتی چند هدفه به صورت زیر تعریف می‌شود:

$$\begin{aligned} & Max(f_1(x), f_2(x), \ldots, f_s(x)) \\ & s.t. \\ & A_i x \leq b_i \qquad i = 1, 2, \ldots, m \\ & x_j \in \{0, 1\} \qquad j = 1, 2, \ldots, n \end{aligned} \tag{۳.۱}$$

که در آن برای هر $(r = 1, \ldots, s)$ $f(x) = (f_1(x), f_2(x), \ldots, f_r(x))$ و برای هر $(i = 1, \cdots, m)$ $A_i = (a_{i1}, a_{i2}, \ldots, a_{in})$ و $X = (x_1, x_2, \ldots, x_n)^T$.

مجموعه X را که یک مجموعه جواب‌های شدنی مسأله می‌نامند که به صورت زیر تعریف می‌شود:

$$X = \{x | A_i x \leq b_i, x_j \in \{0, 1\}, i = 1, \ldots, m \quad , j = 1, \ldots, n\}$$

تعریف ۱.۲.۱. مسأله برنامه ریزی چندهدفه زیر را در نظر بگیرید:

$$\begin{aligned} & Max(f_1(x), f_2(x), \ldots, f_k(x)) \\ & Min(g_1(x), g_2(x), \ldots, g_t(x)) \\ & s.t. \quad x \in X \end{aligned} \tag{۴.۱}$$

که $f_1, f_2, \ldots, f_k$ و $g_1, g_2, \ldots, g_t$ توابع هدف و X ناحیه شدنی است.

$\bar{x} \in X$ یک جواب کارای مسأله فوق است اگر و فقط اگر یک نقطه $x^0 \in X$ وجود نداشته باشد به‌طوریکه:

$$(f_1(x^0), \ldots, f_k(x^0), -g_1(x^0), \ldots, -g_t(x^0)) \geq (f_1(\bar{x}), \ldots, f_k(\bar{x}), -g_1(\bar{x}), \ldots, -g_t(\bar{x})) \tag{۵.۱}$$

تعریف ۲.۲.۱. متناظر با هر $x \in X$ که X مجموعه جواب‌های شدنی مسأله (۳.۱) است، بردار y را به صورت زیر در نظر بگیرید:

[8] Multi-objective Knapsack Problem

$$y = (y_۱, y_۲, \ldots, y_s)^T = (c_۱x, c_۲x, \ldots, c_sx)^T$$

گفته می‌شود که بردار $y = (y_۱, y_۲, \ldots, y_s)$ بر بردار $y^\circ = (y_۱^\circ, y_۲^\circ, \ldots, y_s^\circ)$ غالب است اگر برای هر $r(r = ۱, ۲, \ldots, s)$ ، $y_r \geq y_r^\circ$ و حداقل یک l وجود داشته باشد به‌طوریکه $y_l > y_l^\circ$.

تعریف ۳.۲.۱. مجموعه F که به صورت زیر تعریف می‌شود:

$$F = \{Y | Y = (c_۱x, c_۲x, \ldots, c_sx)^T, A_ix \leq b_i \quad , i = ۱, ۲, \ldots, m \quad , x_j \in \{\circ, ۱\} \quad , j = ۱, \ldots, n\}$$

فضای مقادیر توابع هدف در مسأله (۳.۱) نامیده می‌شود.

تعریف ۴.۲.۱. بردار g که به صورت زیر تعریف می‌شود:

$$g = (g_۱, g_۲, \ldots, g_s)^T = (c_۱x_۱^*, c_۲x_۲^*, \ldots, c_sx_s^*)^T$$

یک بردار ایده‌آل نامیده می‌شود، که x_i^* مقدار بهینه

$$\begin{aligned} &\max \quad c_ix \\ &x \in X \end{aligned}$$

است.

تعریف ۵.۲.۱. نقطه $\bar{x} \in X$ را یک نقطه کارای ضعیف برای مسأله $Max(f_۱(x), \ldots, f_s(x))$ گویند هرگاه هیچ $x \in X$ دیگری وجود نداشته باشد که $f(x) > f(\bar{x})$.

تعریف ۶.۲.۱. در اینجا، تعریف رسمی دو نوع از مسائل کوله‌پشتی چندهدفه را ارائه می‌دهیم. مسأله کوله‌پشتی چند هدفه صفر و یک با یک محدودیت (MOKP-۱) [۹] ، که به صورت زیر فرمول‌بندی می‌شود:

$$\begin{aligned} &Max \quad Z(x) = (Z_۱(x), Z_۲(x), \ldots, Z_i(x), \ldots, Z_m(x)) \\ &s.t. \\ &\sum_{j=۱}^{n} w_jx_j \leq W \\ &x_j \in \{\circ, ۱\}, j \in \{۱, ۲, \ldots, n\} \end{aligned} \tag{۶.۱}$$

مدل دوم مسأله کوله پشتی چندهدفه، مسأله کوله پشتی چندهدفه صفر و یک با چندین قید است (MOKP-۲) [۱۰] که همانند فرمول ریاضی مسأله (MOKP-۱) است، تنها در تعداد

[9] Multi Objective Knapsack Problem-1

[10] Multi Objective Knapsack Problem-2

محدودیت‌ها متفاوت است. بنابراین به صورت زیر تعریف می‌شود:

$$
\begin{aligned}
&Max \quad Z(x) = (Z_۱(x), Z_۲(x), \ldots, Z_i(x), \ldots, Z_m(x)) \\
&s.t. \\
&\sum_{j=۱}^{n} w_j^i x_j \leq W_i, \quad i \in \{۱, ۲, \ldots, m\} \\
&x_j \in \{۰, ۱\}, j \in \{۱, ۲, \ldots, n\}
\end{aligned}
\tag{۷.۱}
$$

که در آن $Z_i(x) = \sum_{j=۱}^{n} c_j^i x_j$ ، i ـامین تابع هدف را نشان می‌دهد. n تعداد متغیرها است و m تعداد توابع هدف است.

۳.۱ الگوریتم حریصانه

روش حریصانه [۱۱] یکی از روش‌های مشهور و پرکاربرد طراحی الگوریتم‌ها است که با ساختاری ساده در حل بسیاری از مسائل استفاده می‌شود. این روش اغلب در حل مسائل بهینه‌سازی استفاده شده و در پاره‌ای مواقع جایگزین مناسبی برای روش‌هایی مانند برنامه‌ریزی پویا است. در حالت کلی این روش سرعت و مرتبه‌ی اجرایی بهتری نسبت به روش‌های مشابه خود دارد؛ اما متناسب با مسئله ممکن است به یک جواب بهینه‌ی سراسری ختم نشود.

در روش حریصانه رسیدن به هدف در هر گام مستقل از گام قبلی و بعدی است. یعنی در هر مرحله برای رسیدن به هدف نهایی، مستقل از این که در مراحل قبلی چه انتخاب‌هایی صورت گرفته و انتخاب فعلی ممکن است چه انتخاب‌هایی در پی داشته باشد، انتخابی که در ظاهر بهترین انتخاب ممکن است صورت می‌پذیرد. به همین دلیل است که به این روش، روش حریصانه گفته می‌شود. زمانی که یک دزد عجول و حریص وارد خانه‌ای می‌شود، در مسیر حرکت خود هر وسیله و کالای با ارزشی را داخل کیسه می‌اندازد. وی در این حالت چندان توجهی نمی‌کند که چه اشیائی را قبلا برداشته و ممکن است در آینده چه اشیاء گرانبهاتری به دست آورد. او در هر گام تنها از بین اشیاء دم دست خود با ارزش‌ترین آن را انتخاب کرده و به وسایل قبلی اضافه می‌کند. این روش کاربردهای عمومی دیگری نیز دارد. زمانی که در مقابل خرید از یک فروشگاه یک اسکناس تحویل فروشنده داده می‌شود، وی با یک حساب سرانگشتی سعی می‌کند با حداقل اسکناس‌ها و سکه‌های ممکن باقیمانده‌ی پول را تولید کرده و به خریدار تحویل دهد. این حساب سرانگشتی ممکن است روش حریصانه باشد.

ایده اصلی روش حریصانه این است که از یک مجموعه اولیه (مانند تهی) برای ساختن جواب مسأله شروع می‌کند و در هر تکرار قسمتی از جواب که نسبت به بقیه بهتر است را انتخاب می‌کند. تضمینی نیست که روش حریصانه جواب بهینه را به دست آورد ولی می‌تواند یک جواب اولیه مناسب برای روش‌هایی باشد که نیاز به یک جواب اولیه برای شروع دارند.

[11] Greedy

در بعضی الگوریتم‌ها روش حریصانه جواب بهینه را پیدا می‌کند مانند روش دایکسترا [۱۲] برای پیدا کردن کوتاهترین مسیر در شبکه که روش حریصانه است. ولی اگر وزن رئوس منفی باشد، روش حریصانه دایکسترا جواب بهینه را نمی‌تواند پیدا کند.

۱.۳.۱ ساختار روش حریصانه

کلیت روش حریصانه در هر مرحله، انتخاب یک عنصر از عناصر موجود است. این عنصر قسمتی از جواب مسئله است که به مجموعه عناصر نهایی اضافه می‌شود. در طی این مسیر گام‌های زیر اتفاق می‌افتد:

۱. روال انتخاب حریصانه: در این گام یک عنصر برای اضافه شدن به مجموعه جواب انتخاب می‌شود. معیار یا روال انتخاب عنصر برای اضافه شدن، ارزش آن عنصر است. بسته به نوع مسئله هر عنصر ارزشی دارد که با ارزشترین آنها انتخاب می‌شود.

۲. امکان‌سنجی و افزودن: پس از انتخاب یک عنصر به صورت حریصانه، باید بررسی شود که آیا امکان اضافه کردن آن به مجموعه جواب‌های قبلی وجود دارد یا نه. گاهی اضافه شدن عنصر یکی از شرایط اولیه‌ی مسئله را نقض می‌کند که باید به آن توجه نمود. اگر اضافه کردن این عنصر هیچ شرطی را نقض نکند، عنصر اضافه خواهد شد؛ وگرنه کنار گذاشته شده و بر اساس گام اول عنصر دیگری برای اضافه شدن انتخاب می‌شود. اگر گزینه‌ی دیگری برای انتخاب وجود نداشته باشد، اجرای الگوریتم به اتمام می‌رسد.

۳. در هر مرحله پس از اتمام گام ۲ و اضافه شدن یک عنصر جدید به مجموعه جواب، باید بررسی کنیم که آیا به یک جواب مطلوب رسیده‌ایم یا نه؟ اگر نرسیده باشیم به گام اول رفته و چرخه را در مراحل بعدی ادامه می‌دهیم.

به زبان ساده، در روش حریصانه طی هر مرحله یک عنصر به روش حریصانه به مجموعه جواب اضافه شده، شرط محدودیت‌ها بررسی شده و در صورت نبود مشکل، عنصر و عناصر بعدی به همین ترتیب به مجموعه جواب اضافه می‌شوند. در طی این گام‌ها اگر به یک شرط نهایی خاص برسیم، یا امکان انتخاب عنصر دیگری برای اضافه کردن به مجموعه جواب وجود نداشته باشد، الگوریتم پایان یافته و مجموعه جواب به دست آمده به عنوان جواب بهینه ارائه می‌شود. توجه داشته باشید که ممکن است بر اساس نوع مسئله، ترتیب اضافه شدن عناصر به مجموعه جواب اهمیت داشته باشد.

[12] Dijestra

۴.۱ الگوریتم حریصانه برای حل مسأله کوله‌پشتی تک‌هدفه

الگوریتم حریصانه برای مسأله کوله‌پشتی به این ترتیب است که اشیا را به ترتیب نزولی بر حسب ارزش به واحد وزن مرتب می‌کند، سپس از شئ شماره ۱ شروع می‌کند. یعنی بیشترین تعداد ممکن از آن‌را در کوله‌پشتی قرار می‌دهد، تا زمانی که دیگر جای خالی برای آن نوع باقی نماند. سپس سراغ شئ بعدی می‌رود (دقت کنید که در اینجا، محدودیتی برای تعداد اشیا نداریم) اگر تعداد مجاز از هر شئ محدود باشد، ممکن است خروجی این الگوریتم از پاسخ بهینه بسیار دور باشد.

۱.۴.۱ الگوریتم حریصانه برای مسائل کوله‌پشتی پیوسته (آزاد سازی شده)

در این مسئله هدف پر کردن یک کوله‌پشتی از وسایل پر ارزشی است که وزن‌های مختلفی دارند. این کوله‌پشتی باید به نحوی پر شود که وزن بار آن از حد مجاز بیشتر نشده و ارزش وسایل داخل آن بیشینه باشد. در مسئله‌ی کوله‌پشتی پیوسته بر خلاف کوله‌پشتی صفر و یک این امکان وجود دارد که بتوان کسری از یک وسیله ــ مثل پارچه ــ را جدا کرده و به وسایل داخل کوله‌پشتی اضافه کرد.

مثال ۱.۴.۱. مسأله زیر را در نظر بگیرید:

$$\begin{aligned} Max \quad & Z = ۲۵x_۱ + ۲۴x_۲ + ۱۵x_۳ + ۱۸x_۴ \\ & ۱۸x_۱ + ۱۵x_۲ + ۱۰x_۳ + ۱۴x_۴ \leq ۲۶ \\ & x_i \in \{۰, ۱\}, ۱ \leq i \leq n \end{aligned}$$

استراتژی‌های مختلف برای حل این مسئله را در ادامه بررسی می‌کنیم:

استراتژی اول:

ابتدا شئ‌ای را که دارای بیشترین ارزش است را انتخاب کنید و متغیر متناسب با آن‌را برابر با یک در نظر بگیرید. سپس با استفاده از این شئ بعدی که دارای ارزش بیشتری نسبت به بقیه است را انتخاب کنید و همین روند را ادامه دهید. برای نمونه در مثال فوق داریم:

$$x_۱ = ۱, x_۲ = x_۳ = x_۴ = ۰ \Longrightarrow Z^* = ۲۵$$

اگر شرط $x_i \in \{۰, ۱\}$ را با شرط $۰ \leq x_i \leq ۱$ عوض کنیم، مدل آزاد سازی شده داریم. بنابراین:

$$x_۱ = ۱, x_۲ = \frac{۸}{۱۵}, x_۳ = x_۴ = ۰ \Longrightarrow Z^* = ۳۷/۸$$

استراتژی دوم:

در این مرحله سبکترین شئ را انتخاب کنید که در اینجا مربوط به متغیر $x_۳$ و بعد از آن $x_۲$ است. متغیر متناسب با $x_۳$ را برابر با یک در نظر می‌گیریم و مسأله را حل می‌کنیم.

$$x_۳ = ۱, x_۴ = ۱, x_۳ = ۰, x_۲ = ۰ \Longrightarrow Z^* = ۳۳$$

در مدل آزاد سازی شده داریم:

$$x_۳ = ۱, x_۴ = ۱, x_۲ = \frac{۲}{۱۵}, x_۱ = ۰ \Longrightarrow Z^* = ۳۶/۲$$

استراتژی سوم:

بیشترین مقدار $\frac{p_i}{w_i}$ را به دست آورده و متغیر متناسب با آنرا برابر با یک در نظر می‌گیریم و مسأله را حل می‌کنیم.

$$\frac{p_i}{w_i} = (\frac{۲۵}{۱۸}, \frac{۲۴}{۱۵}, \frac{۱۵}{۱۰}, \frac{۱۸}{۱۴}) = (۱/۳۹, ۱/۶۰, ۱/۵, ۱/۲۳)$$
$$x_۲ = ۱, x_۳ = ۱, x_۴ = ۰, x_۱ = ۰ \Longrightarrow Z^* = ۳۹$$

بنابراین در حالت آزاد سازی شده:

$$\frac{p_i}{w_i} = (\frac{۲۵}{۱۸}, \frac{۲۴}{۱۵}, \frac{۱۵}{۱۰}, , \frac{۱۸}{۱۴}) = (۱/۳۹, ۱/۶۰, ۱/۵, ۱/۲۳)$$
$$x_۲ = ۱, x_۳ = ۱, x_۱ = \frac{۱}{۱۸}, x_۴ = ۰ \Longrightarrow Z^* = ۴۰/۳۸$$

قضیه ۱.۴.۱. اگر در مسأله کوله‌پشتی متغیرها مرتب شده باشند بطوریکه:

$$v_i/w_i \geq v_{i+۱}/w_{i+۱} \quad (i = ۱, ۲, \ldots, N-۱)$$

آنگاه جواب بهینه

$$\max \sum_{i=۱}^{N} v_i x_i$$
$$s.t.$$
$$\sum_{i=۱}^{N} w_i x_i \leq W$$
$$x_i \geqq ۰, i = ۱, \ldots, N$$

اگر $v_i > ۰$ برابر با $x_i = ۰, i = ۱, \ldots, N$ است و $x_۱ = \frac{W}{w_۱}, x_۲ = x_۳ = \ldots, x_N = ۰$ اگر $v_i \leq ۰$. اگر متغیرها کراندار باشند و $v_۱, v_۲, \ldots, v_{t+۱} > ۰$ یعنی $x_i \leq u_i (i = ۱ \ldots, N)$ و

$w_۱u_۱ \leq W$ جواب بهینه با شروع $i = ۱$ ، و جایگزاری $x_i(i = ۱, \ldots, t \leq N)$ برابر با u_i تا زمانی‌که $\sum_{i=۱}^{t+۱} w_iu_i > W$ و سپس با جایگزاری $x_{t+۱} = (w - \sum_{i=۱}^{t} w_iu_i)/w_{t+۱}$ با متغیرهای باقی‌مانده به صفر برسد. اگر $v_k \leq ۰(۱ \leq k \leq t + ۱)$ آنگاه جواب بهینه برنامه‌ریزی خطی $x_i = u_i$ برای $i = ۱, \ldots, k - ۱$ است و $x_i = ۰$ برای $i = k, k + ۱, \ldots, N$ است.

۵.۱ روش شاخه و کران

تعریف ۱.۵.۱. روش شاخه و کران یک الگوی طراحی الگوریتم برای مسائل بهینه سازی است. این روش در یک جستجوی فضای حالات جواب‌های احتمالی مسئله را پیمایش می‌کند. در اینجا مجموعه‌ی جواب‌های احتمالی به صورت یک درخت در نظر گرفته می‌شود که ریشه‌ی آن متناظر با همه‌ی جواب‌ها و انشعابات آن زیر مجموعه‌هایی از جواب های احتمالی هستنند، قبل از پیمایش مجموعه جواب‌های یک زیر شاخه، الگوریتم مجموعه جواب‌های شاخه را با کران پایین و بالای جواب مسئله‌ی بهینه سازی به طور کلی چک می‌کند و درصورتی که زیر شاخه توانایی تولید جواب بهتری برای مسئله را نداشته باشد، از پیمایش کل زیر شاخه صرف نظر می‌شود.

۱.۵.۱ ویژگی‌های روش

برای حل مسائل بهینه سازی‌ای که الگوریتمی با زمان چند جمله‌ای برای آن‌ها پیدا نشده است، از الگوریتم‌هایی با پیچیدگی نمایی که زمان اجرای پایینی دارند استفاده می‌شود. روش شاخه و کران در این سبک مسائل گزینه‌ی مناسبی است.

۲.۵.۱ ساختار کلی روش

اگر بخواهیم الگوریتمی ارائه دهیم که تابع f را کمینه کند و تابع g کران پایینی برای مقدار f در راس‌های یک زیر درخت از فضای حالات باشد، ساختار کلی این روش به شکل زیر خواهد بود:

۱. ابتدا یک جواب دلخواه x پیدا می‌کنیم و مقدار B را برابر با $f(x)$ قرار می‌دهیم ، از این پس مقدار B نشان دهنده‌ی بهترین جواب بدست آمده تا این نقطه جستجو خواهد بود.

۲. یک صف از راس‌های فضای حالات در نظر گرفته و ریشه‌ی درخت فضای حالات را به آن اضافه می‌کنیم.

۳. مراحل بعد را تا وقتی که صف خالی شود تکرار می‌کنیم.

a. یک راس از داخل صف بیرون کشیده

b. در صورتی که این راس نشان دهنده‌ی جوابی خاص از مسئله مثلا x باشد و $f(x) < b$ ، این جواب بهترین جواب یافته شده تا کنون است در نتیجه مقدار $f(x)$ را درون B قرار می‌دهیم.

c. در غیر این صورت به ازای همه‌ی انشعابات این راس مثلا N_i

d. اگر $g(N_i) < B$:

i: این انشعاب ممکن است به جواب بهتری برسد. پس N_i را به لیست اضافه می‌کنیم.

ii: در غیر این صورت این انشعاب ارزشی ندارد چون کران پایین جواب‌های آن از کران بالای جواب مسأله بزرگ‌تر است.

e. به دستور ۳ بازگرد.

۶.۱ الگوریتم شاخه و کران برای مسأله کوله‌پشتی

در الگوریتم شاخه و کران برای مسأله کوله‌پشتی ابتدا مسأله را به حالت آزاد سازی شده تبدیل می‌کنیم سپس با استفاده از الگوریتم حریصانه جواب‌ها را به دست آورده و روش شاخه و کران را پیاده سازی می‌کنیم. مثال زیر را در نظر بگیرید:

مثال ۱.۶.۱. مسأله کوله‌پشتی زیر را با روش شاخه و کران حل کنید:

$$\begin{aligned} Max \quad & R = ۲۳x_۱ + ۱۹x_۲ + ۲۸x_۳ + ۱۴x_۴ + ۴۴x_۵ \\ & ۸x_۱ + ۷x_۲ + ۱۱x_۳ + ۶x_۴ + ۱۹x_۵ \leq ۲۵ \\ & x_۱ = x_۲ = x_۳ = x_۴ = x_۵ = ۰ \text{یا} ۱ \end{aligned}$$

آزاد سازی این مسأله بصورت زیر است:

$$\begin{aligned} Max \quad & R = ۲۳x_۱ + ۱۹x_۲ + ۲۸x_۳ + ۱۴x_۴ + ۴۴x_۵ \\ & ۸x_۱ + ۷x_۲ + ۱۱x_۳ + ۶x_۴ + ۱۹x_۵ \leq ۲۵ \\ & ۰ \leq x_۱, x_۲, x_۳, x_۴, x_۵ \leq ۱ \end{aligned}$$

بیشترین مقدار $\frac{p_i}{w_i}$ را به دست آورده و متغیر متناسب با آنرا برابر با یک در نظر می‌گیریم و مسأله را حل می‌کنیم.

$$\frac{p_i}{w_i} = (\frac{۲۳}{۸}, \frac{۱۹}{۷}, \frac{۲۸}{۱۱}, \frac{۱۴}{۶}, \frac{۴۴}{۱۹}) = (۲/۸۷, ۲/۷۱, ۲/۵۴, ۲/۳۳, ۲/۳۱)$$

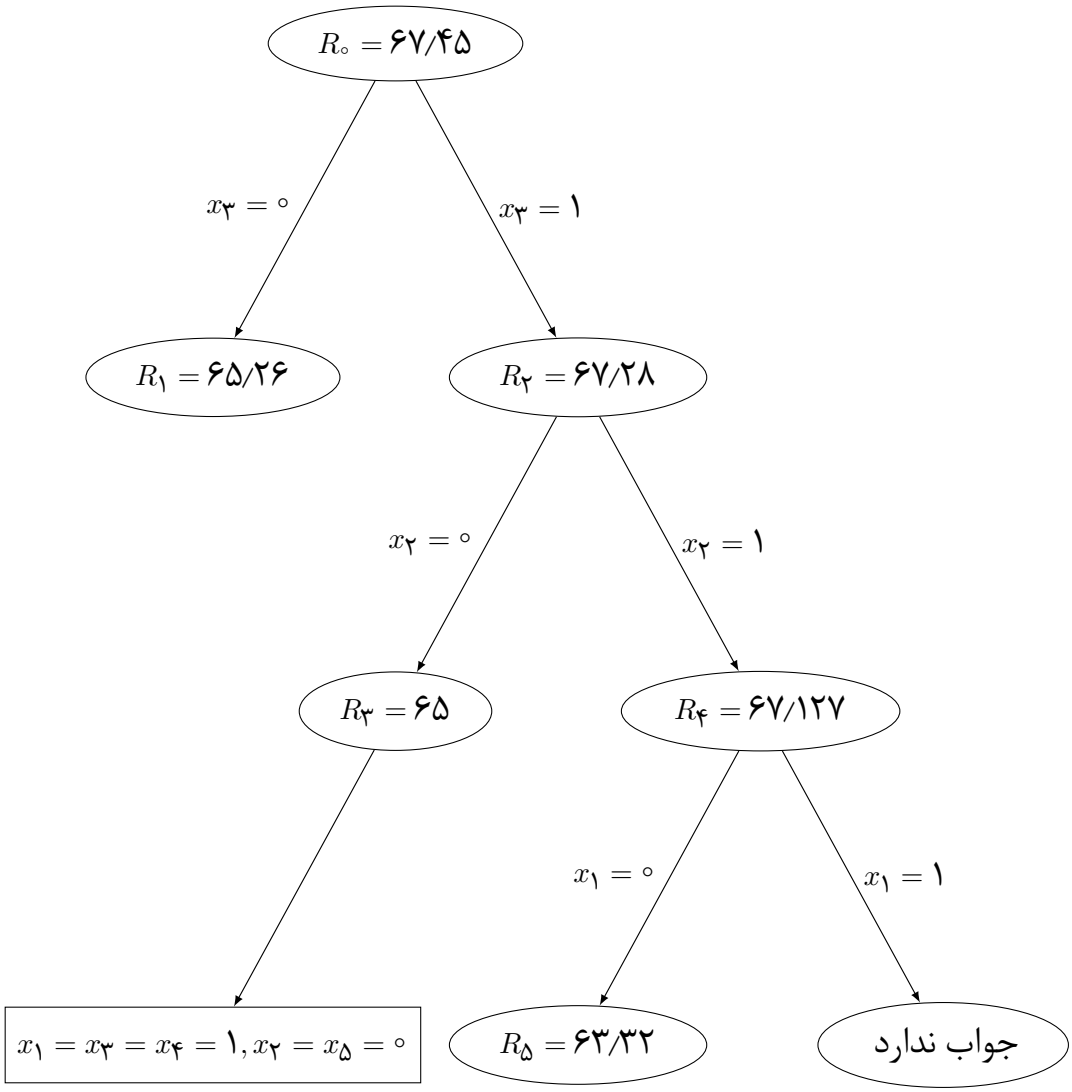
$$x_۱ = ۱, x_۲ = ۱, x_۳ = \frac{۱۰}{۱۱}, x_۴ = x_۵ = ۰ \Longrightarrow R_۰ = ۶۷/۴۵$$

بنابراین شاخه را روی $[\frac{۱۰}{۱۱}] + ۱ > x_۳ > [\frac{۱۰}{۱۱}]$ می‌زنیم. اگر $x_۳ = ۰$ باشد، آنگاه $R_۱ = ۶۵/۲۶$ ، با ادامه این شاخه متغیر $x_۴$ کسر بزرگتر از واحد می‌شود و شاخه قطع می‌گردد. اگر $x_۳ = ۱$ آنگاه $R_۲ = ۶۷/۲۸$ ، دوباره شاخه می‌زنیم، در این مرحله $x_۱ = x_۳ = ۱, x_۲ = \frac{۶}{۷}, x_۴ = x_۵ = ۰$ پس یعنی شاخه را روی متغیر $x_۲$ می‌زنیم.

اگر $x_۲ = ۰$ آنگاه چون $x_۵ = ۰$ است مقدار $R_۳ = ۶۵$ و تمامی متغیر ها یا مقدار صفر دارند و یا مقدار یک می‌گیرند و اینجا تمام می‌شود.

ولی اگر $x_۲ = ۱$ مقدار $R_۴ = ۶۷/۱۲۷$ و مسأله ادامه می‌یابد.

این روند تا زمانی ادامه می‌یابد که تمامی متغیرها یا یک یا صفر به دست آیند و یا مسأله دیگر جوابی نداشته باشد. در شکل زیر روش شاخه و کران را برای این مثال ملاحظه می‌کنید:



فصل ۲

الگوریتمی برای حل مسأله کوله پشتی دوهدفه

۱.۲ مقدمه

در این فصل روش ارائه شده در [۶] را برای حل مسأله کوله پشتی دوهدفه بیان می‌کنیم. روش پیشنهاد شده همه‌ی جواب‌های کارای مسأله کوله پشتی دوهدفه صفر و یک را با محدودیت‌ها و متغیرهای کمتری نسبت به روش‌های ارائه شده در مراجع دیگر پیدا می‌کند.

۲.۲ مسأله کوله‌پشتی دو هدفه

مسأله کوله پشتی دوهدفه زیر را درنظر بگیرید:

$$
\begin{aligned}
& P_۱ : Max(f(x), g(x)) \\
& s.t. \\
& \sum_{i=۱}^{n} a_{ij} x_{ij} \leq W_j \quad j = ۱, \ldots, m \\
& x_{ij} \in \{۰, ۱\}
\end{aligned}
\tag{۱.۲}
$$

هدف این است که هر دو تابع هدف را در یک زمان ماکزیمم کنیم. برای حل این مسأله ما اول دو مسأله کوله پشتی تک هدفه $P_۲$ و $P_۳$ را در نظر می‌گیریم:

$$\begin{aligned} & P_۲ : Max \quad f(x) \\ & s.t. \\ & \sum_{i=۱}^{n} a_{ij}x_{ij} \leq W_j \quad j = ۱, \cdots, m \\ & x_{ij} \in \{۰, ۱\} \end{aligned} \tag{۲.۲}$$

$$\begin{aligned} & P_۳ : Max \quad g(x) \\ & s.t. \\ & \sum_{i=۱}^{n} a_{ij}x_{ij} \leq W_j \quad j = ۱, \cdots, m \\ & x_{ij} \in \{۰, ۱\} \end{aligned} \tag{۳.۲}$$

فرض کنید که $x_۱^*$ جواب بهینه مسأله $P_۲$ و $x_۲^*$ جواب بهینه مسأله $P_۳$ باشد. در گام دوم ما مسائل تک هدفه زیر را حل می‌کنیم.
در تکرار اول مسأله به صورت زیر تعریف می‌شود:
مسأله کوله پشتی مرحله یک:(KP۱)

$$\begin{aligned} & KP۱(P_۴) : Max(f(x) + g(x)) \\ & s.t. \\ & f(x) > f(x_۱^*) - Mt_{۱۱} \\ & f(x) > f(x_۲^*) - Mt_{۱۲} \\ & g(x) > g(x_۱^*) - Mt_{۲۱} \\ & g(x) > g(x_۲^*) - Mt_{۲۲} \\ & \sum_{i=۱}^{n} a_{ij}x_{ij} \leq W_j \\ & t_{۱۱} + t_{۱۲} = ۱ \\ & t_{۲۱} + t_{۲۲} = ۱ \\ & x_{ij} \in \{۰, ۱\}, t_{iq} \in \{۰, ۱\}, q = ۲, \quad j = ۱, \cdots, m \end{aligned} \tag{۴.۲}$$

که M یک عدد مثبت بزرگ است. در k ـ امین تکرار مسأله به صورت زیر تعریف می‌شود:

k ـامین مرحله مسأله کوله‌پشتی (KPk)

$$
\begin{aligned}
& KPk(P_۵) : Max(f(x)+g(x)) \\
& s.t. \\
& \sum_{i=۱}^{n} a_{ij}x_{ij} \le W_j \\
& f(x) \ge f(x_q^*) - Mt_{۱q}, \quad q = ۱,\cdots,k-۱ \\
& g(x) \ge g(x_q^*) - Mt_{۲q}, \quad q = ۱,\cdots,k-۱ \\
& t_{۱q} + t_{۲q} = ۱, \quad q = ۱,\cdots,k-۱ \\
& x_{ij} \in \{۰,۱\} \quad \forall i,j \\
& t_{۱q}, t_{۲q} \in \{۰,۱\}, \quad q = ۱,\cdots,k-۱, \quad j = ۱,\cdots,m
\end{aligned}
\tag{۵.۲}
$$

مسأله در k ـامین تکرار دارای $m+۳\times(k-۱)$ محدودیت است که m تعداد محدودیت $P_۱$ است. همچنین دارای $n+۲\times(k-۱)$ متغیر است که n تعداد متغیرهای $P_۱$ است. این روند ادامه دارد تا زمانی‌که مسأله نشدنی شود یا جواب بهینه آن مشابه یکی از بهترین نتایج به دست آمده قبلی باشد. اگر مسأله دارای جواب‌های بهینه دگرین باشد ما باید همه آن‌ها را تعیین کنیم. در هر تکرار الگوریتم پیشنهاد شده، حداقل یک جواب کارا پیدا می‌شود. چون تعداد جواب‌های شدنی متناهی است الگوریتم همگراست. از آنجا که در هر تکرار الگوریتم، برخی از محدودیت‌ها به مسأله تکرار قبلی اضافه می‌شود حل مسائل نیازمند محاسبات زیاد است. در بخش بعدی تعدادی لم را برای رفع این مشکل آورده‌ایم.

۳.۲ نتیجه اصلی و الگوریتم

لم ۱.۳.۲. مسأله KP۱ را به صورت زیر می‌توان کاهش داد:

$$
\begin{aligned}
& Max(f(x)+g(x)) \\
& s.t. \\
& f(x) > f(x_۲^*) \\
& g(x) > g(x_۱^*) \\
& \sum_{i=۱}^{n} a_{ij}x_{ij} \le W_j \\
& x_{ij} \in \{۰,۱\}, \quad j = ۱,\cdots,m
\end{aligned}
\qquad (NKP۱) \tag{۶.۲}
$$

برهان. اگر $t_{۱۱} = ۰$ ، چون $f(x_۱^*)$ ماکزیمم مقدار $f(x)$ است، محدودیت اول مسأله $P_۴$ برقرار نیست و ما جواب شدنی نداریم. بنابراین $t_{۱۱} = ۱$. چون $t_{۱۱} + t_{۱۲} = ۱$ پس $t_{۱۲} = ۰$. به طور

مشابه برای $t_{۲۱}$ و $t_{۲۲}$ نیز همین‌طور است. بنابراین مدل مسأله به صورت (۶.۲) است. □

لم ۲.۳.۲. برای k ـامین تکرار ما داریم:

$$
\begin{aligned}
& Max(f(x)+g(x)) \\
& s.t. \\
& f(x) \geq f(x^*_{۲}) \\
& g(x) \geq g(x^*_{۱}) \\
& f(x) \geq f(x^*_{k-۱}) - Mt_{۱} \\
& g(x) \geq g(x^*_{k-۱}) - Mt_{۲} \\
& \sum_{i=۱}^{n} a_{ij}x_{ij} \leq W_j \\
& t_{۱}+t_{۲}=۱, \quad x_{ij} \in \{۰,۱\}, \quad t_{۱},t_{۲} \in \{۰,۱\}, \quad j=۱,\cdots,m
\end{aligned}
\tag{۷.۲}
$$

$(NKPk)$

که $x^*_{k-۱}$ جواب بهینه مسأله در $(k-۱)$ ـامین تکرار است.

برهان. می‌دانیم که $x^*_{۱}$ جواب بهینه مسأله $P_{۲}$ و $x^*_{۲}$ جواب بهینه مسأله $P_{۳}$ است. محدودیت $f(x) > f(x^*_{۲})$و $g(x) > g(x^*_{۱})$ تضمین می‌کند که جواب k ـامین تکرار بر جواب مسأله $P_{۲}$ و $P_{۳}$ تسلط دارد. همچنین جواب k ـامین تکرار باید غالب بر جواب تکرارهای قبلی باشد. بنابراین یکی از محدودیت‌های $f(x) > f(x^*_{k-۱}) - Mt_{۱}$ یا $g(x) > g(x^*_{k-۱}) - Mt_{۲}$ باید صادق باشد. به دلیل اینکه $f(x)$ یا $g(x)$ با توجه به جواب‌های قبلی باید بهبود یابد و ارزش کوچکتری نسبت به قبل داشته باشد. محدودیت $t_{۱}+t_{۲}=۱$ باعث می‌شود این شرط صادق باشد. اگر محدودیت $f(x) > f(x^*_{k-۱})$ صادق باشد، آنگاه $g(x_k) > g(x^*_{k-۱})$ باشد. چون در مرحله kـام قید به مسأله اضافه می‌شود بنابراین مقدار تابع هدف از مر حله $(k-۱)$ ـام بدتر می‌شود یعنی داریم: $f(x^*_k)+g(x^*_k) < f(x^*_{k-۱}+g(x^*_{k-۱}))$ چون $f(x^*_k) > f(x^*_{k-۱})$ باید $g(x^*_k) > f(x^*_{k-۱})$. □

۴.۲ الگوریتم

مرحله ۰: مقدار دهی اولیه

گام ۱ – ۰ : مسأله $P_{۴}$ را حل می‌کنیم و جواب بهینه را به مجموعه G اضافه می‌کنیم، که مجموعه G مجموعه‌ای از جواب بهینه است.

گام ۲ – ۰ : مسأله $P_{۳}$ را حل می‌کنیم و جواب بهینه را به مجموعه G اضافه می‌کنیم،

مرحله ۱: تکرار روش

گام ۱ – ۱ : $k = ۱$. مسأله $P_۵$ را با استفاده از جواب‌های بهینه $P_۲$ و $P_۴$ حل کنید، و جواب بهینه را به مجموعه G اضافه کنید.
گام ۲ – ۱ : $k = k + ۱$. مسأله (NKPk) را حل کنید و جواب بهینه را به مجموعه G اضافه کنید.
گام ۳ – ۱ : اگر جواب بهینه برابر با تکرار قبلی است، به مرحله ۲ بروید. در غیر این صورت به گام (۲ – ۱) بروید.

مرحله ۲: توقف

تمام جواب‌های بهینه مشخص شده‌اند. توقف کنید.

لم ۱.۴.۲. $Min(Max(f(x) + g(x))) = f(x^*_۲) + g(x^*_۱)$

برهان. طبق دو محدودیت‌های اول (NKPk) ، min مقدار و حد پایین $f(x)$ و $g(x)$ ، به ترتیب $f(x^*_۲)$ و $g(x^*_۱)$ هستند. بنابراین min مقدار تابع $Max(f(x) + g(x))$ ، $f(x^*_۲) + g(x^*_۱)$ خواهد شد. □

لم ۲.۴.۲. Max مقدار $f(x) + g(x)$ به وسیله حل مسأله (NKP۱) به دست می‌آید.

لم ۳.۴.۲. کران بالا برای تعداد تکرارهای الگوریتم پیشنهاد شده وقتی که $x_i \in \mathbb{Z}$به صورت زیر است:

$$\frac{Max(Max(f(x) + g(x))) - Min(Max(f(x) + g(x)))}{Min(f(x) + g(x) \text{ در ضریب } x_i)} \qquad (۸.۲)$$

برهان. لم‌های قبل، Min و Max مقدار $f(x)+g(x)$ را به ما می‌دهد. چون متغیرها صحیح‌اند، بنابراین الگوریتم زمانی که مقادیر $Max(Max(f(x) + g(x)))$ و $Min(Max(f(x) + g(x)))$ برابر شوند، پایان می‌یابد. بنابراین:

$$Max(Max(f(x) + g(x))) - Min(f(x) + g(x) \text{ در ضریب } x_i) * k = Min(Max(f(x) + g(x)))$$

بنابران k برابر با کسر فوق می‌باشد. همانطور که در بخش قبل مشاهده شد، در تکرار $K(K > ۱)$، یک مسأله با $m + ۳(k - ۱)$ محدودیت و $n + ۲(k - ۱)$ متغیر داریم. بنابراین با افزایش یک تکرار، تعداد متغیرها و محدودیت‌ها افزایش می‌یابد؛ در حالی‌که برای روش پیشنهادی تعداد محدودیت‌ها و متغیرها ثابت و برابر با $m + ۵$ محدودیت و $n + ۲$ متغیر برای $(k > ۱)$ است. □

۵.۲ مثال عددی

در این بخش، برای نشان دادن روش پیشنهاد شده، یک مثال عددی ارائه شده است.

مثال ۱.۵.۲. مسأله دوهدفه کوله پشتی صفر و یک زیر را در نظر بگیرید:

$$
\begin{aligned}
&Max \quad ۹X_۱ + ۶X_۲ + ۵X_۳ + ۳X_۴ + ۵X_۵ + ۲X_۶ + ۵X_۷ + ۱۰X_۸ + X_۹ + ۶X_{۱۰}\\
&Max \quad ۱۵X_۱ + ۱۲X_۲ + ۸X_۳ + ۱۱X_۴ + ۱۴X_۵ + ۱۰X_۶ + ۱۴X_۷ + ۹X_۸ + ۱۱X_۹ + ۱۴X_{۱۰}\\
&s.t.\\
&۲۰۰X_۱ + ۱۵۰X_۲ + ۱۶۰X_۳ + ۵۰۰X_۴ + ۲۶۰X_۵ + ۱۴۰X_۶ + ۲۰۰X_۷ + ۴۰۰X_۸ + ۲۵۰X_۹ + ۱۲۰X_{۱۰} \le ۱۲۵۰\\
&۱۴X_۱ + ۸۳X_۲ + ۹۴X_۳ + ۳X_۴ + ۲۶X_۵ + ۱۴X_۶ + ۶۵X_۷ + ۸X_۸ + ۱۱X_۹ + ۸۷X_{۱۰} \le ۲۵۰\\
&X_i \in \{۰,۱\}, i = ۱,\cdots,۱۰.
\end{aligned}
$$

مرحله ۰:مقدار دهی اولیه

گام۰ـ۱:

$$
\begin{aligned}
(p_۲) :&Max \quad ۹X_۱ + ۶X_۲ + ۵X_۳ + ۳X_۴ + ۵X_۵ + ۲X_۶ + ۵X_۷ + ۱۰X_۸ + X_۹ + ۶X_{۱۰}\\
&s.t.\\
&۲۰۰X_۱ + ۱۵۰X_۲ + ۱۶۰X_۳ + ۵۰۰X_۴ + ۲۶۰X_۵ + ۱۴۰X_۶ + ۲۰۰X_۷ + ۴۰۰X_۸ + ۲۵۰X_۹ + ۱۲۰X_{۱۰} \le ۱۲۵۰\\
&۱۴X_۱ + ۸۳X_۲ + ۹۴X_۳ + ۳X_۴ + ۲۶X_۵ + ۱۴X_۶ + ۶۵X_۷ + ۸X_۸ + ۱۱X_۹ + ۸۷X_{۱۰} \le ۲۵۰\\
&X_i \in \{۰,۱\}, i = ۱,\cdots,۱۰.
\end{aligned}
$$

گام ۰ـ۲:

$$
\begin{aligned}
&Max \quad ۱۵X_۱ + ۱۲X_۲ + ۸X_۳ + ۱۱X_۴ + ۱۴X_۵ + ۱۰X_۶ + ۱۴X_۷ + ۹X_۸ + ۱۱X_۹ + ۱۴X_{۱۰}\\
&s.t.\\
&۲۰۰X_۱ + ۱۵۰X_۲ + ۱۶۰X_۳ + ۵۰۰X_۴ + ۲۶۰X_۵ + ۱۴۰X_۶ + ۲۰۰X_۷ + ۴۰۰X_۸ + ۲۵۰X_۹ + ۱۲۰X_{۱۰} \le ۱۲۵۰\\
&۱۴X_۱ + ۸۳X_۲ + ۹۴X_۳ + ۳X_۴ + ۲۶X_۵ + ۱۴X_۶ + ۶۵X_۷ + ۸X_۸ + ۱۱X_۹ + ۸۷X_{۱۰} \le ۲۵۰\\
&X_i \in \{۰,۱\}, i = ۱,\ldots,۱۰.
\end{aligned}
$$

$X = \{۱,۱,۰,۰,۱,۰,۰,۱,۰,۱\}$ و $X' = \{۱,۰,۰,۰,۱,۱,۱,۰,۱,۱\}$ جواب‌های بهینه منحصر به فرد مسائل $P_۲$ و $P_۳$ با مقادیر تابع هدف به ترتیب ۲۸ و ۶۴ هستند. بنابراین:

$G = \{(۱,۱,۰,۰,۱,۰,۰,۱,۰,۱), (۱,۰,۰,۱,۱,۱,۰,۱,۱)\}$.

مرحله ۱:

گام ۱ – ۱ : $k = ۱$

$$
\begin{aligned}
&Max \quad ۲۴X_۱ + ۱۸X_۲ + ۱۳X_۳ + ۱۴X_۴ + ۱۹X_۵ + ۱۲X_۶ + ۱۹X_۷ + ۱۹X_۸ + ۱۲X_۹ + ۲۰X_{۱۰} \\
&s.t. \\
&۹X_۱ + ۶X_۲ + ۵X_۳ + ۳X_۴ + ۵X_۵ + ۲X_۶ + ۵X_۷ + ۱۰X_۸ + X_۹ + ۶X_{۱۰} > ۲۸ \\
&۱۵X_۱ + ۱۲X_۲ + ۸X_۳ + ۱۱X_۴ + ۱۴X_۵ + ۱۰X_۶ + ۱۴X_۷ + ۹X_۸ + ۱۱X_۹ + ۱۴X_{۱۰} > ۶۴ \\
&۲۰۰X_۱ + ۱۵۰X_۲ + ۱۶۰X_۳ + ۵۰۰X_۴ + ۲۶۰X_۵ + ۱۴۰X_۶ + ۲۰۰X_۷ + ۴۰۰X_۸ + ۲۵۰X_۹ + ۱۲۰X_{۱۰} \leq ۱۲۵۰ \\
&۱۴X_۱ + ۸۳X_۲ + ۹۴X_۳ + ۳X_۴ + ۲۶X_۵ + ۱۴X_۶ + ۶۵X_۷ + ۸X_۸ + ۱۱X_۹ + ۸۷X_{۱۰} \leq ۲۵۰ \\
&X_i \in \{۰, ۱\}, \quad i = ۱, \ldots, ۱۰.
\end{aligned}
$$

با حل مسأله فوق، $X = \{۱, ۱, ۰, ۰, ۱, ۱, ۰, ۰, ۱, ۱\}$ با مقدار تابع هدف ۲۹ و ۷۶ به دست می‌آوریم. بنابراین:

$$G = \{(۱, ۱, ۰, ۰, ۱, ۰, ۰, ۱, ۰, ۱), (۱, ۰, ۰, ۰, ۱, ۱, ۱, ۰, ۱, ۱,), (۱, ۱, ۰, ۰, ۱, ۱, ۰, ۰, ۱, ۱)\}.$$

گام ۲ – ۱ : $k = ۲$

$$
\begin{aligned}
&Max \quad ۲۴X_۱ + ۱۸X_۲ + ۱۳X_۳ + ۱۴X_۴ + ۱۹X_۵ + ۱۲X_۶ + ۱۹X_۷ + ۱۹X_۸ + ۱۲X_۹ + ۲۰X_{۱۰} \\
&s.t. \\
&۹X_۱ + ۶X_۲ + ۵X_۳ + ۳X_۴ + ۵X_۵ + ۲X_۶ + ۵X_۷ + ۱۰X_۸ + X_۹ + ۶X_{۱۰} > ۲۸, \\
&۱۵X_۱ + ۱۲X_۲ + ۸X_۳ + ۱۱X_۴ + ۱۴X_۵ + ۱۰X_۶ + ۱۴X_۷ + ۹X_۸ + ۱۱X_۹ + ۱۴X_{۱۰} > ۶۴, \\
&Mt۱ + ۹X_۱ + ۶X_۲ + ۵X_۳ + ۳X_۴ + ۵X_۵ + ۲X_۶ + ۵X_۷ + ۱۰X_۸ + X_۹ + ۶X_{۱۰} > ۲۹, \\
&Mt۲ + ۱۵X_۱ + ۱۲X_۲ + ۸X_۳ + ۱۱X_۴ + ۱۴X_۵ + ۱۰X_۶ + ۱۴X_۷ + ۹X_۸ + ۱۱X_۹ + ۱۴X_{۱۰} > ۷۶, \\
&t۱ + t۲ = ۱, \\
&۲۰۰X_۱ + ۱۵۰X_۲ + ۱۶۰X_۳ + ۵۰۰X_۴ + ۲۶۰X_۵ + ۱۴۰X_۶ + ۲۰۰X_۷ + ۴۰۰X_۸ + ۲۵۰X_۹ + ۱۲۰X_{۱۰} \leq ۱۲۵۰, \\
&۱۴X_۱ + ۸۳X_۲ + ۹۴X_۳ + ۳X_۴ + ۲۶X_۵ + ۱۴X_۶ + ۶۵X_۷ + ۸X_۸ + ۱۱X_۹ + ۸۷X_{۱۰} \leq ۲۵۰, \\
&X_i \in \{۰, ۱\}, \quad i = ۱, \ldots, ۱۰, \\
&t۱, t۲ \in \{۰, ۱\}.
\end{aligned}
$$

با حل مسأله فوق برای $M = ۱۰۰۰۰$ ، $X = \{۱, ۰, ۰, ۰, ۱, ۰, ۱, ۱, ۰, ۱\}$ با مقدار تابع هدف ۳۵ و ۶۶ به دست می‌آوریم. بنابراین:

$$G = \{(۱, ۱, ۰, ۰, , ۱, ۰, ۰, ۱, ۰, ۱), (۱, ۰, ۰, ۰, ۱, ۱, ۱, ۰, ۱, ۱), (۱, ۱, ۰, ۰, ۱, ۱, ۰, ۰, ۱, ۱), (۱, ۰, ۰, ۰, ۱, ۰, ۱, ۱, ۰, ۱)\}$$

گام ۲ – ۲ : ۳ $= k$

$$
\begin{aligned}
&Max \quad ۲۴X_۱ + ۱۸X_۲ + ۱۳X_۳ + ۱۴X_۴ + ۱۹X_۵ + ۱۲X_۶ + ۱۹X_۷ + ۱۹X_۸ + ۱۲X_۹ + ۲۰X_{۱۰}\\
&s.t.\\
&۹X_۱ + ۶X_۲ + ۵X_۳ + ۳X_۴ + ۵X_۵ + ۲X_۶ + ۵X_۷ + ۱۰X_۸ + X_۹ + ۶X_{۱۰} > ۲۸,\\
&۱۵X_۱ + ۱۲X_۲ + ۸X_۳ + ۱۱X_۴ + ۱۴X_۵ + ۱۰X_۶ + ۱۴X_۷ + ۹X_۸ + ۱۱X_۹ + ۱۴X_{۱۰} > ۶۴,\\
&Mt۱ + ۹X_۱ + ۶X_۲ + ۵X_۳ + ۳X_۴ + ۵X_۵ + ۲X_۶ + ۵X_۷ + ۱۰X_۸ + X_۹ + ۶X_{۱۰} > ۳۵,\\
&Mt۲ + ۱۵X_۱ + ۱۲X_۲ + ۸X_۳ + ۱۱X_۴ + ۱۴X_۵ + ۱۰X_۶ + ۱۴X_۷ + ۹X_۸ + ۱۱X_۹ + ۱۴X_{۱۰} > ۶۶,\\
&t۱ + t۲ = ۱,\\
&۲۰۰X_۱ + ۱۵۰X_۲ + ۱۶۰X_۳ + ۵۰۰X_۴ + ۲۶۰X_۵ + ۱۴۰X_۶ + ۲۰۰X_۷ + ۴۰۰X_۸ + ۲۵۰X_۹ + ۱۲۰X_{۱۰} \le ۱۲۵۰,\\
&۱۴X_۱ + ۸۳X_۲ + ۹۴X_۳ + ۳X_۴ + ۲۶X_۵ + ۱۴X_۶ + ۶۵X_۷ + ۸X_۸ + ۱۱X_۹ + ۸۷X_{۱۰} \le ۲۵۰,\\
&X_i \in \{۰,۱\}, \quad i = ۱,\dots,۱۰,\\
&t۱, t۲ \in \{۰,۱\}.
\end{aligned}
$$

با حل مسأله فوق برای $M = ۱۰۰۰۰$ ، ما به دست می‌آوریم $X = \{۱,۱,۰,۰,۱,۱,۰,۰,۱,۱\}$ جواب‌های به دست آمده در مجموعه G که مجموعه جواب‌های کارا می‌باشد، است و چون جواب به دست آمده برابر با جواب بهینه در مراحل قبل است، الگوریتم خاتمه می‌یابد. جواب‌های نهایی

هستند. $G = \{(۱,۱,۰,۰,۱,۰,۰,۱,۰,۱), (۱,۰,۰,۰,۱,۱,۱,۰,۱,۱,), (۱,۱,۰,۰,۱,۱,۰,۰,۱,۱,), (۱,۰,۰,۰,۱,۰,۱,۱,۰,۱)\}$

فصل ۳

حل مسأله کوله پشتی چندهدفه با استفاده از تکنیک فراابتکاری

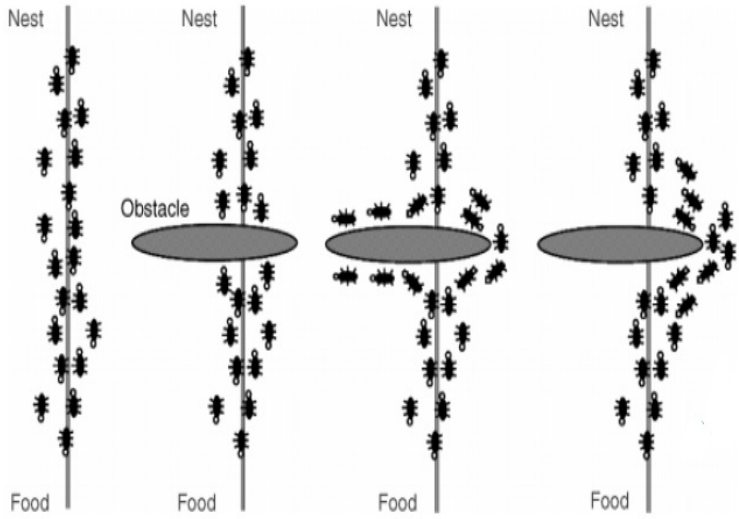
۱.۳ مقدمه

در این فصل، به بررسی یک روش ترکیبی برای حل مسأله کوله پشتی می‌پردازیم که در [۱۱] آمده است. روش پیشنهاد شده برای حل دو نوع مسأله چندهدفه استفاده می‌شود، مسأله کوله پشتی چندهدفه با یک قید و مسأله کوله پشتی چند هدفه با چندین قید. این مسأله در بسیاری از شرایط عملی مانند انتخاب پروژه‌های سرمایه گذاری و کنترل بودجه ظاهر می‌شود. معمولا ما از الگوریتم‌های دقیق یا روش‌های ابتکاری استفاده می‌کنیم که الگوریتم‌های دقیق جواب‌های دقیقی را فراهم می‌کنند ولی روش ابتکاری جواب دقیق را تضمین نمی‌کند، اما به طور کلی به ما این اجازه را می‌دهد که جواب‌های تقریبی خوبی را به دست آوریم.
روش پیشنهادی از دو تکنیک فراابتکاری تشکیل شده است: بهینه سازی اجتماع مورچه و روش جستجوی پراکنده. در این فصل در ابتدا در مورد بهینه سازی اجتماع مورچه بحث شده است؛ سپس به روش جستجوی پراکنده می‌پردازد.

۲.۳ رفتار در جستجوی غذای مورچگان

چگونه مورچه‌ها می‌توانند کوتاه‌ترین مسیر بین منبع غذا و لانه خود را بدون هیچ گونه مکانیسم هدایتی دیداری، مرکزی یا فعال پیدا کنند؟ مطالعات رفتارهای بعضی از گونه‌های واقعی مورچه‌ها نشان داده است که یک الگوی تصادفی و شانسی در حرکت مورچه‌ها برای یافتن غذا وجود دارد. به محض اینکه یک منبع غذا در محلی در حوالی لانه مورچه‌ها قرار داده می‌شود، الگوی تصادفی به مرور سازماندهی بیشتری یافته و در نهایت منجر به رسیدن به یک مسیر مشابه توسط مورچه‌ها، خواهد انجامید و به صورت جادویی دیده می‌شود که مورچه‌ها کوتاه‌ترین مسیر را پیدا کرده‌اند. این رفتار عجیب نتیجه تأثیر مورچه‌هایی که منبع غذا را پیدا کرده‌اند بر سایر مورچه‌ها است. این تأثیر ممکن است در بعضی از گونه‌های مورچه‌ها به صورت مستقیم و بعضی گونه‌های دیگر به صورت غیر مستقیم باشد. نشان داده شده است که در اغلب گونه‌های مورچه این تأثیر با به کار گیری مورچه‌های دیگر به صورت غیر مستقیم اتفاق می‌افتد. مورچه‌هایی که غذا را یافته‌اند با سایر مورچه‌ها از طریق یک مکانیسم ارتباطی به نام اثر فرومون ارتباط برقرار می‌نمایند. زمانی که یک مورچه یک منبع غذا را پیدا می‌کند و مقداری غذا به خانه می‌برد، در مسیر حرکت خود ماده شیمیایی به نام فرومون [1] منتشر می‌نمایند. بقیه مورچه‌ها که به دنبال غذا می‌گردند، در هنگام انتخاب مسیر به غلظت فرومون مسیر توجه می‌نمایند. مسیرهای با غلظت فرومون بیشتر، احتمال بیشتری برای انتخاب دارند. هر چقدر مورچه‌های بیشتری از یک مسیر عبور نمایند، غلظت فرومون آن مسیر افزایش یافته و در نتیجه، مورچه‌های بیشتری نیز نسبت به آن مسیر علاقه‌مند می‌شوند.

[1] Pheromone



شکل ۱.۳: رفتار مورچه در یافتن کوتاه‌ترین مسیر بین لانه و منبع غذا

۳.۳ شکل ساده‌ای از بهینه سازی اجتماع مورچگان

بهینه سازی اجتماع مورچگان [۲] ، یک روش متداول تصادفی از دیدگاه رفتار مورچه‌های واقعی در طبیعت است. مسأله عمومی، یافتن کوتاه‌ترین مسیر بین دو رأس در یک گراف $G = (V, E)$ که Vمجموعه رأس‌ها و E ماتریسی است که ارتباط بین رأس‌ها را بیان می‌کند را در نظر بگیرید. این گراف $n_G = |V|$ رأس دارد. طول، L^k ، که توسط مورچه k ام ساخته شده است، به صورت فاصله مبدأ تا مقصد تعریف می‌گردد. در این حالت، یک غلظت فرومون τ_{ij} ، به هر یال (i, j) از گراف، تخصیص داده می‌شود.

در بهینه سازی اجتماع مورچگان ساده (SACO) [۳]، در ابتدا به هر یال، یک مقدار تصادفی به عنوان فرومون اولیه، $\tau_{ij}(۰)$ ، تخصیص داده می‌شود. در اصل، در ابتدای الگوریتم به علت نبود غلظت فرومون یا مقدار خیلی کم آن، مورچه‌ها به طور تصادفی مسیرها را انتخاب می‌نمایند. تعدادی از مورچه‌ها، $k = ۱, \ldots, n_k$، در رأس مبدأ قرار داده می‌شود. در هر تکرار (SACO) ، هر مورچه به صورت پله پله یک مسیر را بین مبدأ و مقصد می‌سازد. در هر رأس، هر مورچه فرایند تصمیم گیری را برای تعیین یال بعدی مسیر خود به کار می‌برد. اگر مورچه k در رأس i قرار داشته باشد، رأس بعدی $j \in N_i^k$ را بر اساس یک تابع احتمال گذر به شکل زیر تعیین

[2] Ant Colony Optimization
[3] Simple Ant Colony Optimization

می‌نمایند:

$$p_{ij}^{k}(t)=\begin{cases}\frac{\tau_{ij}^{\alpha}(t)}{\sum_{j\in N_i^k}\tau_{ij}^{\alpha}(t)}, & j\in N_i^k \text{ اگر}\\ ۰, & \text{در غیر این صورت}\end{cases} \qquad (۱.۳)$$

که در آن N_i^k مجموعه یال‌های قابل انتخاب برای مورچه k در رأس i می‌باشد. اگر برای هر رأس i و مورچه k ، $N_i^k=\emptyset$ آنگاه رأس پیشین رأس i در نظر گرفته می‌شود (در صورتیکه مورچه به مسیر بسته برسد، برمی‌گردد). این مسأله منجر به ایجاد یک حلقه می‌گردد، ولی مشکلی را ایجاد نمی‌کند زیرا پس از رسیدن به مقصد، در هنگام معرفی مسیر بین مبدأ و مقصد، حلقه‌ها حذف می‌گردند. در معادله بالا، α یک مقدار ثابت مثبت برای شدت بخشیدن به اثر غلظت فرومون است. مقدار بزرگ α ، اهمیت غلظت فرومون را افزایش داده و تأثیر رفتارهای تصادفی را کمتر می‌کند و منجر به همگرایی سریع به سمت یک بهینه محلی می‌گردد.

بعد از آنکه تمامی مورچه‌ها، یک مسیر کامل بین مبدأ تا مقصد را ساختند و حلقه‌های احتمالی حذف شدند، هر مورچه ردپای خود را به سمت رأس مبدأ دنبال می‌نماید و مقداری فرومون، در هر یال (i,j) روی مسیر طی شده، مطابق با قاعده زیر منتشر می‌نماید:

$$\Delta\tau_{ij}^{k}\propto\frac{۱}{L^{k}(t)} \qquad (۲.۳)$$

که در آن $L^k(t)$ معرف طول مسیر ساخته شده توسط مورچه kام در مرحله زمانی t می‌باشد. در نتیجه، میزان غلظت فرومون هر مسیر مطابق با قاعده زیر به هنگام می‌گردد:

$$\tau_{ij}(t+۱)=\tau_{ij}(t)+\sum_{k=۱}^{n_k}\Delta\tau_{ij}^{k}(t) \qquad (۳.۳)$$

که در آن n_k تعداد مورچه‌ها می‌باشد. با استفاده از معادله (۲.۳)، مقداری چگالی جدید روی هر یال تولید می‌گردد و این مقدار بستگی به کیفیت مسیرهایی دارد که توسط مورچه‌ها با استفاده از آن یال ساخته شده است. به عبارت دیگر، هر مورچه مقداری فرومون در هر یال متناسب با کیفیت مسیرش به جا می‌گذارد. بعد از یک تکرار، میزان فرومون موجود در یک یال با توجه به مورچه‌های عبوری از یال تغییر می‌یابد. مقدار چگالی اضافه شده با مقدار قبلی چگالی فرومون آن مسیر طبق معادله (۳.۳) به هنگام می‌گردد.

در SACO ، مقدار کیفیت هر جواب با یک تابع ساده به صورت عکس طول مسیر طی شده بیان می‌گردد. البته معیارهای دیگری نیز می‌توان تعریف کرد. اگر $x^k(t)$ نشان دهنده یک جواب در مرحله زمانی t باشد، آنگاه $f(x^k(t))$ کیفیت آن جواب را بیان می‌کند. در عمده موارد از یک روش ارزیابی صریح، که در آن مقادیر فرومون متناسب با بعضی از شاخص‌های کیفی جواب‌های ساخته شده هستند، برای تعیین میزان انتشار فرومون استفاده می‌گردد.

اگر مقدار فرومون انتشاری رابطه عکس با طول مسیر داشته باشد در آن صورت، مقدار بیشتر $f(x^k(t))$ (که نشان دهنده بدتر بودن جواب ساخته شده است) موجب می‌گردد که مقدار کمتری فرومون $\frac{۱}{f(x^k(t))}$ در آن مسیر منتشر گردد. الگوریتم زیر ساختار الگوریتم را نشان می‌دهد. در این الگوریتم، قواعد توقف متعددی می‌توان استفاده کرد. برای مثال چند قاعده در زیر آمده‌اند:

۱. توقف زمانی که تعداد ماکزیممی از تکرارها انجام گرفت.

۲. توقف زمانی که یک جواب قابل قبول ساخته شد ($f(x^k(t)) \leq \epsilon$)

۳. توقف زمانی که تمامی مورچه‌ها (یا اغلب آن ها) از یک مسیر مشابه پیروی کردند.

بنابراین، یک مسیر طولانی موجب می‌گردد که تمامی اجزای آن دارای جذابیت کمتری شوند (فرومون کمتری در آنها منتشر گردد).
در آزمایشات اولیه، مسأله پل بایناری در مرجع [۱۲] نشان داده شد که مورچه‌ها به سرعت به یک جواب همگرا می‌شوند و پس از اینکه مقدار کمی زمان سپری گردید، مسیرهای جایگزین را انتخاب می‌کنند. برای اجبار مورچه‌ها در کاوش بیشتر و جلوگیری از همگرایی زودرس، اجازه داده شد که مقدار شدت فرومون در یک مسیر در هر تکرار از الگوریتم، قبل از ساخت مسیر، دچار تبخیر شود. در این حالت، مقداری از شدت فرومون هر مسیر کاهش (تبخیر) پیدا می‌کند. در این حالت نحوه به هنگام سازی مقدار شدت فرومون طبق معادله زیر انجام می‌گردد:

$$\tau_{ij} \leftarrow (۱ - \rho)\tau_{ij}(t) \qquad (۴.۳)$$

که در آن $\rho \in [۰, ۱]$ یک مقدار ثابت است که نرخ تبخیر فرومون در هر مرحله را نمایش می‌دهد. این تبخیر منجر به فراموش شدن درصدی از وقایع گذشته (تصمیمات گرفته شده در گذشته یا به عبارت دیگر مسیرهای انتخاب شده در گذشته) می‌گردد و به عبارت دیگر یک پارامتر کنترلی تاریخچه جستجو است. برای مقادیر بالای ρ ، نرخ بالای تبخیر، توجه به گذشته کم شده و جستجو بیشتر وضعیت تصادفی و شانسی به خود می‌گیرد. در صورتیکه $\rho = ۱$ باشد، جستجو کاملا تصادفی و شانسی خواهد بود.

۴.۳ الگوریتم SACO

۱. مقدار $\tau_{ij}(۰)$ را برابر با یک مقدار تصادفی کوچک تنظیم کنید؛

۲. قرار دهید $t = ۰$ ؛

۳. n_k مورچه روی رأس‌های اصلی قرار دهید؛

۴. تکرار (تا زمانی که قاعده توقف برآورده شود)؛

۱.۴ برای هر مورچه $k = ۱, \ldots, n_k$ انجام دهید:

۲.۴ یک مسیر $x^k(t)$ را بسازید؛

۳.۴ $x^k(t) = \emptyset$؛

۴.۴ تکرار (تا زمانی که به مقصد برسید):

۱.۴.۴ رأس بعدی را بر پایه احتمال تعریف شده در معادله (۱.۳) انتخاب کنید؛

۲.۴.۴ یال (i, j) را به مسیر $x^k(t)$ اضافه کنید؛

۵.۴ کلیه حلقه‌ها را از $x^k(t)$ خارج کنید؛

۶.۴ طول مسیر $(f(x^k(t)))$ را محاسبه کنید؛

۵. پایان؛

۶. برای هر یال (i, j) از گراف انجام دهید:

۱.۶ تبخیر فرومون؛

۲.۶ کاهش فرومون، $\tau_{ij}(t)$ ، با استفاده از معادله (۴.۳) ؛

۷. پایان؛

۸. برای هر مورچه $k = ۱, \ldots, n_k$ انجام دهید:

۱.۸ برای هر یال (i, j) از $x^k(t)$ انجام دهید:

$$\Delta\tau^k = \frac{۱}{f(x^k(t))}$$

۲.۸ مقادیر $\tau_{ij}(t)$ را با استفاده از معادله (۳.۳) بهنگام کنید؛

۲.۸ پایان؛

۹. پایان؛

۱۰. $t = t + ۱$

۱۱. مسیر $x^k(t)$ با کمترین $f(x^k(t))$ را به عنوان بهترین جواب چاپ کنید.

۵.۳ بهینه سازی اجتماع مورچگان

طبق تحقیقات آلیا [۴] و همکارانش [۱۲]، یک مورچه یک هدف را انتخاب می‌کند، ($i \in \{۱,\dots,m\}$) و بر اساس این انتخاب، بهترین نقاط را در میان آن‌هایی که در دسترس هستند انتخاب می‌کند. هر مورچه دارای حافظه است که مسیرهای قبلا انتخاب شده را ذخیره می‌کند و از هر رأس که بیش از یک بار انتخاب شده باشد صرف نظر می‌کند. بعد از این مرحله، به روز رسانی مسیرهای فرومون هر رأس انجام می‌شود.
انتخاب یک عنصر رأس $j \in \{۱,\dots,n\}$ ، مطابق با قواعدی است که حاصل از در نظر گرفتن مقدار فرومون و اطلاعات اکتشافی مربوط به این رأس است، این قانون به صورت زیر تعریف شده است:

$$p_j = \frac{(\tau_j^i)^\alpha * (\eta_j^i)^\beta}{\sum_{\forall y \in cand}(\tau_y^i)^\alpha * (\eta_y^i)^\beta} \qquad (۵.۳)$$
$$\circ < p_i \leq ۱, \tau_j^i, \eta_j^i > \circ, \alpha, \beta \geq \circ$$

که τ_j^i و η_j^i به ترتیب نشان دهنده مقدار فرومون‌ها و اطلاعات ابتکاری و کارایی (که جذابیت حرکت را بیان می‌کند) مربوط به مورد j در عملکرد هدف i است. α و β به ترتیب دو پارامتر هستند که اهمیت τ_j^i و η_j^i را تعیین می‌کند، اگر $\alpha = \circ$ باشد، اطلاعات فرومونی مورد استفاده قرار نمی‌گیرد و سابقه گذشته جستجو فراموش می‌گردد و الگوریتم حالت جستجوی تصادفی را به خود می‌گیرد. اگر $\beta = \circ$ باشد، جذابیت حرکت‌ها صرف نظر می‌گردد و الگوریتم ساختار SACO را به خود می‌گیرد. cand لیستی است که شامل تمام اشیا کاندید می‌باشد و y یک عنصر عمومی از این لیست است. برای جلوگیری از ایجاد حلقه، ممکن است cand به صورت مجموعه تمامی رأس‌های همسایه‌ای که تاکنون توسط مورچه k ملاقات نشده‌اند تعریف گردد. برای این منظور یک لیست ممنوع برای هر مورچه تعریف می‌گردد. هر زمانی‌که یک مورچه یک رأس جدید را ملاقات می‌نماید، آن رأس به لیست ممنوع مورچه اضافه می‌گردد. برای اینکه یک رأس تنها یک‌بار ملاقات شود، رأس‌های قرار گرفته در لیست ممنوع، از مجموعه cand خارج می‌گردند. در مسأله کوله‌پشتی بسته به نوع مسأله کوله پشتی به ترتیب برای MOKP-۱ و MOKP-۲ ، اطلاعات را می‌توان از طریق زیر محاسبه کرد:

$$\eta_j^i = c_j^i / w_j, \quad \eta_j^i = c_j^i / w_j^i$$

به روز رسانی فرومون‌ها به صورت زیر انجام می‌شود:

$$\tau_j^i = (۱ - \rho) * \tau_j^i + \triangle\tau_j^i \qquad (۶.۳)$$

که ρ نشان دهنده نرخ تبخیر است ($\rho \in [\circ, ۱[$) و $\triangle\tau_j^i$ نشان دهنده نرخ به روز رسانی فرومون قرار گرفته در مورد $j \in \{۱,\dots,n\}$ است. این مقدار با توجه به کیفیت جواب ساخته شده،

[4] Alya

دقیق تر تعریف شده است: $\triangle\tau_j^i = ۱/(۱ + z_i' - z_i'')$ که z_i'' بالاترین سود در i ـ امین تابع هدف برای چرخه فعلی است و z_i' بالاترین سود حاصل از i ـ امین تابع هدف از آغاز اجرا است.
برای جلوگیری از همگرایی سریع نتایج، مقدار فرومون موجود در هر مورد بین دو مقدار محدود است، یعنی $\tau_{\min} \le \tau_j^i \le \tau_{\max}$.
شبه کد مربوطه از الگوریتم پیشنهادی ACO برای MOKP-۱ و MOKP-۲ در الگوریتم زیر ارائه شده است. این شبه کد با روش‌های زیر مشخص می‌شود:

۱. مقداردهی اولیه متغیرهای مسأله، به طور مثال، مقداردهی اولیه عوامل فرومون و لیستی که مجموعه‌ای از تمام جواب‌های کارا و غالب را ذخیره می‌کند.

۲. ساخت و ساز جواب، هر مورچه به طور تصادفی یک هدف را انتخاب می‌کند و بر اساس آن، طبق قاعده احتمالی که در بالا تعیین شد، بهترین موارد لیست را انتخاب می‌کند.

۳. به روز رسانی مسیرهای دنباله دار فرومون در رأس مربوط به بهترین جواب‌های دور و به روز رسانی لیستی که جواب‌های کارا را ذخیره می‌کند.

الگوریتم‌هاي مورچه را در حالت كلي مي‌توان به صورت زير بيان كرد:
گام اول: تعیین $\tau_{\max}$، فرومون اولیه و پارامترهای مربوط به انتخاب مسیر و تعداد مورچه‌ها و قرار دادن $t = ۰$.
گام دوم: ساختن یک جواب برای هر مورچه و جستجوی محلی برای هر جواب و بهنگام کردن فرومون در هر مسیر به دست آمده.
گام سوم: بهنگام کردن فرومون در بهترین جواب‌ها
گام چهارم: اگر $t + ۱ < \tau_{\max}$ آنگاه پارامترهای انتخاب مسیر را تغییر داده و به گام دوم می‌رویم.
گام پنجم: انتخاب بهترین جواب و پایان الگوریتم.

الگوریتم ACO برای مسأله MOKP-۱ و MOKP-۲

۱. ($\tau_j^i \leftarrow \tau_{\max}$) مقداردهی اولیه مسیرهای فرومون مربوط به هر بخش.

۲. $S \leftarrow \emptyset$ (مقداردهی اولیه لیستی که تمام جواب‌های کارا را ذخیره می‌کند.)

۳. A تعداد مورچه‌ها را مشخص می‌کند.

۴. تکرار (برای $(k = ۱)$ تا $|A|$ تا زمانی که به حداکثر تعداد دور یا بخش رسیده است، و تا زمانی که $\text{cand} \neq \emptyset$ مراحل زیر انجام شود:)

۴.۱ به طور تصادفی یک تابع هدف $\{z_۱, \ldots, z_i, \ldots, z_m\}$ انتخاب کنید، و آن‌را f_i قرار دهید.

۴.۲ $s^k \leftarrow \emptyset$ (لیستی که توسط مورچه k جواب ساخته شده را ذخیره می‌کند.)

۴.۳ لیستی از نامزدهای انتخابی s^k را انتخاب کنید $\{۱,\ldots,j,\ldots,n\}$ و آن‌را در cand قرار دهید.

۴.۴ یک شئ را با احتمال p_j(cand) انتخاب کنید و در j قرار دهید.

۴.۵ $s^k \leftarrow s^k \cup \{j\}$ (بدون نقض محدودیت)

۴.۶ به روز رسانی cand.

۵. برای $i = ۱$ تا m مراحل زیر انجام شود:

برای $j = ۱$ تا n مراحل زیر انجام شود:

۵.۱ $\tau_j^i \leftarrow (۱-\rho)\tau_j^i$ (تبخیر فرومون)

۵.۲ (اگر $j \in s^k$) آن‌گاه $\tau_j^i \leftarrow \tau_j^i + \Delta\tau_j^i$

۵.۳ اگر $(\tau_j^i < \tau_{\min})$ آن‌گاه $\tau_j^i \leftarrow \tau_{\min}$

۵.۴ اگر $(\tau_j^i > \tau_{\max})$ آن‌گاه $\tau_j^i \leftarrow \tau_{\max}$

۶. $S \leftarrow S \cup_{k=۱}^{|A|} s^k$

۷. پایان

۶.۳ روش جستجوی پراکنده

الگوریتم فرا ابتکاری جستجوی پراکندگی [۵] : همانند دیگر روش‌های تکاملی، الگوریتم جستجوی پراکندگی، به‌جای کار با یک جواب، با جمعیتی از جواب‌ها کار می‌کنند و از رویه‌هایی برای ترکیب جواب‌ها به‌منظور ساخت جمعیت جدید استفاده می‌کنند. یکی از ویژگی‌های مهم این رویکرد ارتباط تنگاتنگ آن با الگوریتم فرا ابتکاری جستجوی ممنوعه (TS) [۶] و همچنین استفاده از اصولی برای برخورداری از حافظه تطبیقی به همراه اصولی برای استفاده مؤثر از حافظه است. درواقع الگوریتم جستجوی پراکندگی و جستجوی ممنوعه ریشه مشترکی دارند و SS ابتدایی به‌عنوان یکی از جزء فرایندهای موجود در چهارچوب TS بررسی می‌شد. اگرچه خیلی از ادبیات TS و پیاده‌سازی‌های TS به این جزء توجهی نکرده‌اند. این امر باعث شده تا مزیت جستجوی پراکنده تا این اواخر که مطالعات اندکی آن را به‌صورت یک الگوریتم تکاملی مستقل در نظر گرفتند، ناشناخته باقی بماند. (کلمه ‘‘مستقل’’ به‌طور نسبی است زیرا تقریباً تمامی پیاده‌سازی‌های SS از برخی از طراحی‌های حافظه تطبیقی TS استفاده کرده‌اند. برای اینکه روش TS بهتر شناخته شود، ما در این مقاله بر روی اجزایی از SS تمرکز می‌کنیم که از فرایندهای مرتبط با TS مجزا هستند.) جستجوی تصادفی بر روی مجموعه‌ای از جواب‌ها کار می‌کند که به آن مجموعه مرجع گویند و این جواب‌ها را با یکدیگر ترکیب می‌کند تا جواب‌های جدید ساخته شوند. مکانیسم اصلی برای ترکیب جواب‌ها طوری هست که جواب جدید از ترکیب خطی دو جواب دیگر ساخته شود. برخلاف ’’جمعیت’’ در الگوریتم ژنتیک، اندازه مجموعه مرجع در الگوریتم جستجوی پراکندگی تقریباً کوچک است. الگوریتم جستجوی پراکندگی دو یا بیش از دو جزء از مجموعه مرجع را باهدف ساخت جواب‌های جدید، طی یک فرایند سیستماتیک انتخاب می‌کند. به این دلیل که تعداد زیرمجموعه‌های مجموعه مرجع (مثلاً دو عضوی تا پنج عضوی) خیلی زیاد می‌شود، حتی یک فرایند تولید ترکیبات جواب که غربالگری زیادی در انتخاب مثال‌های مطلوب این زیرمجموعه‌ها دارد، می‌تواند تعداد زیادی از ترکیبات را تولید نماید، پس بسیار ضروری است تا تعداد اعضای مجموعه مرجع کم باشد. جستوجوی پراکنده یک روش تکاملی است که بصورت موفقیت آمیز برای مسائل سخت در مورد بهینه سازی بکار می‌رود. اصول و مفاهیم اساسی این روش در دهه ۱۹۷۰ بر اساس فرمول‌هایی پیشنهاد شد که به تاریخ دهه ۱۹۶۰ برای ترکیب قواعد تصمیم گیری و محدودیت‌های مسئله باز می‌گشت. جستجوی پراکنده بر خلاف روش‌های تکاملی دیگر (مانند الگوریتم‌های ژنتیک)، بر اساس این قضیه پی‌ریزی شده است که روش ها و طرح‌های سیستماتیک برای ایجاد جواب‌های جدید، مزایای مهم‌تری در مقایسه با مزایای توسل به انتخاب تصادفی دارند. روش جستجوی پراکنده یک روش تکاملی مبتنی برجمعیت است. این

[5] Scatter Search
[6] Tabu Search

روش به وسیله گالور[۷] [۲۱] رسما معرفی شد و در حال انجام یک مطالعه اکتشافی برای مسائل برنامه ریزی خطی صحیح بود. برای ترکیب همه جواب‌ها، روش SS روش‌های پیش انتخاب شده را از منطق امکان پذیر می‌کند و بر روی مجموعه‌ای از جواب‌ها به نام مجموعه مرجع، که معمولا بهترین جواب‌های نمونه را تشکیل می‌دهد، عمل می‌کند. در مورد MOKP-۱ و MOKP-۲ مفهوم بهترین جواب فقط تنوع جواب‌ها را در نظر می‌گیرد. روش SS پیشنهادی در این فصل، نتیجه روش SS است که توسط گومز داسیلوا[۸] و همکاران [۸] ارائه شده است. روش پیشنهادی بر اساس ساختار معمولی روش SS که گالور توصیف کرده است سازماندهی شده است، که با الگوریتم ACO که در بخش قبل توضیح داده شده است ترکیب شده است.

۱.۶.۳ روش تنوع

این روش شامل ایجاد مجموعه‌ای از جواب‌های مختلف یعنی مجموعه اولیه جواب‌ها می‌شود. در این مورد ما از روش ACO توصیف شده در بخش قبلی برای به دست آوردن مجموعه اولیه از جواب کارا استفاده می‌کنیم.

۲.۶.۳ روش بهبود

این روش با هدف ارتقا جواب‌ها از روش تنوع است. در این مورد، از روش ACO برای به دست آوردن جواب‌های کیفی استفاده می‌کنیم. روش بهبود جواب‌ها به شدت به جواب‌های تولید شده از طریق روش ترکیبی اعمال خواهد شد.

۳.۶.۳ تولید مجموعه مرجع و به روز رسانی روش

به روزرسانی مجموعه مرجع، برای ایجاد و حفظ یک مجموعه مرجع تشکیل شده از b تعداد از بهترین جواب‌های یافته شده (که مقدار b معمولا کم است و برای مثال از ۲۰ بیشتر نمی شود)، که سازماندهی شده است تا دسترسی مؤثر توسط قسمت‌های دیگر این روش را فراهم سازد. جواب‌ها مطابق با کیفیت یا تنوع خود در مجموعه مرجع طبقه بندی می‌شوند. اساسا در این روش بهترین جواب‌ها از نمونه اصلی انتخاب می‌شوند تا مجموعه‌ای از جواب‌ها (مجموعه مرجع) را تشکیل دهند که در آخر برای به دست آوردن جواب‌های جدید ترکیب می‌شوند. در این مورد، مفهوم ”بهترین“ فقط تنوع جواب‌ها را از مجموعه اولیه در نظر می‌گیرد. به طور دقیق تر، هر جواب $s^u \in S(u = ۱,۲,\ldots,|S|)$ با توجه به مقادیر غیر قابل انطباق زیر، مرتب

[7] Golver
[8] Gomesdasilva

شده است:

$$d_u^{ref} = ||s^{ref} - s^u|| = \sum_{j=1}^{n} |x_j^{ref} - x_j^u|$$

$$u = 1, 2, \ldots, |S|$$

که s^{ref} جواب مرجع را نشان می‌دهد. یعنی جوابی که از آن مقادیر غیر مستقیم جواب‌های دیگر تعریف می‌شود. این جواب مربوط به $s^u \in S$ با مقدار وزن $\sum_{i=1}^{m} z_i(x^u)$ برای هر $u = 1, 2, \ldots, |S|$ است. پس از مرتب‌سازی همه جواب‌ها، مجموعه مرجع R از S ساخته شده است، با تقسیم آن به گروه‌ها با تعداد جواب‌های یکسان، و یک جواب از هر گروه به منظور تعریف مرجع انتخاب شده است. حداکثر ۲۰ عنصر را در مجموعه مرجع در نظر می‌گیریم.

۴.۶.۳ روش تولید زیرمجموعه

در این مرحله مجموعه مرجع به زیرمجموعه‌ای از جواب‌ها برای ایجاد موارد جدید تقسیم می‌شود. به عنوان روش ترکیبی بر اساس اکتشاف تمایز بین جواب‌ها در هر زیرمجموعه، ما نباید از جواب‌های بسیار متفاوتی از R استفاده می‌کنیم. در این مورد زیرمجموعه‌ها به وسیله جفت‌های متوالی جواب از R تشکیل می‌شود. بنابراین $|R| - 1$ ترکیب وجود خواهد داشت. برای جلوگیری از ترکیب تکراری جواب‌ها، یک لیست تابو(ممنوع) ایجاد می‌شود. لیستی که همه‌ی جفت‌های متناظر قبلی ذخیره می‌شود.

۵.۶.۳ روش ترکیبی جواب

روش ترکیب جواب، تا یک زیرمجموعه معین از زیرمجموعه‌های ایجاد شده توسط روش ایجاد زیرمجموعه را به یک یا چند بردار جواب ترکیبی تبدیل کند. در این روش، جواب‌هایی از زیرمجموعه‌های ساخته شده برای به دست آوردن موارد جدید ترکیب می‌شوند، فرض کنید $s^0 = \{z^0, x^0\}$ و $s^1 = \{z^1, x^1\}$ دو جواب کارا از یک زیرمجموعه، که در آن z^0 و z^1 نشان دهنده بردارهای غالب و x^0 و x^1 جواب‌های کارای مربوط باشد. در نامگذاری گالور ، s^0 جواب آغازگر نامیده می‌شود و s^1 جواب هدایت (راهنما) نامیده می‌شود. جواب هدایت برای شناسایی متغیرهای تصمیم گیرنده استفاده می‌شود که مقادیر آن باید تغییر در جواب اولیه باشد تا آن‌را به جواب هدایت کند؛ یعنی با قرار دادن ویژگی‌های s^1 از s^0 جواب‌های جدید به دست می‌آید. به طوردقیق‌تر، مواردی که از s^0 حذف می‌شوند شناسایی می‌شوند (مواردی با $x_j^0 = 1$) و همچنین اقلامی که باید به s^0 وارد شوند (مواردی با $x_j^0 = 0$). مواردی با $x_j^0 = 1$ کارایی آن‌ها با توجه به مقادیر غیر قابل کاهش مرتب شده‌اند و مواردی با $x_j^0 = 0$ کارایی آن‌ها بر اساس مقادیر غیر قابل افزایش مرتب شده‌اند، راهی برای حذف موارد با کارایی کمتر و قرار دادن موارد با بالاترین کارایی است. کارایی مربوط به هر مورد با استفاده از معادلات $\eta_j^i = \max_{i=1}^{m}\{c_j^i/w_j\}$

و $\eta_j^i = \max_{i=1}^m \{c_j^i / w_j^i\}$ به ترتیب برای MOKP-۱ و MOKP-۲ محاسبه می‌شود. از آن‌جا که مسیر از s° به s^1 متفاوت از s^1 به s° است، جواب‌ها مبادله می‌شوند و یکبار دیگر روش ترکیبی بالا اعمال می‌شود.
الگوریتم SS پیشنهاد شده برای MOKP-۱ و MOKP-۲ در زیر ارائه شده است،که توسط روش‌های اصلی زیر مشخص می‌شود:

۱. مقدار دهی اولیه متغیرهای مسأله

۲. به دست آوردن مجموعه اولیه با استفاده از الگوریتم ACO

۳. ساخت و به روز رسانی مجموعه مرجع، تعریف زیرمجموعه‌ها و استفاده از روش ترکیبی برای به دست آوردن جواب‌های جدید

۴. به روز رسانی لیست شامل تمام جواب‌های غیرمغلوب یا کارا

الگوریتم SS برای MOKP-۱ و MOKP-۲

۱. $S \leftarrow \emptyset$ (مقداردهی اولیه‌ی لیستی که تمام جواب‌های کارا را ذخیره می‌کند.)

۲. $\hat{S} \leftarrow \emptyset$ (مقداردهی اولیه لیستی که تمام جواب‌های نتیجه را از روش ترکیبی ذخیره می‌کند.)

۳. $Tabu \leftarrow \emptyset$ مقداردهی اولیه لیست تابو. یعنی، لیستی که همه ترکیب‌های قبلی را ذخیره می‌کند.

۴. قرار دادن مجموعه کارای اولیه با بکارگیری الگوریتم ACO در S

۵. تکرار(تا زمانی‌که به ماکزیمم مقدار هر بخش برسید.)

۵.۱ $R \leftarrow \emptyset$ (مقداردهی اولیه مجموعه مرجع)

۵.۲ S^u با مقدار وزن $\sum_{i=1}^m z_i(x^u)(S)$ بیابید و در S^{ref} قرار دهید.

۶. برای $u = 1$ تا $|S|$، با استفاده از معادله $d_u^{ref} \leftarrow \sum_{j=1}^n |x_j^{ref} - x_j^u|$ مقادیر را محاسبه کنید.

۷. لیست S را با توجه به مقادیر غیر نزولی مرتب کنید.

۸. $R \leftarrow R \cup_{u=1}^{|s|} S^u$

۹. برای $(r = ۱)$ تا $|R| - ۱$، اگر $(\{R^r, R^{r+۱}\} \notin tabu)$، آن‌گاه مراحل زیر را انجام دهید:

$$\hat{S} \leftarrow \hat{S} \cup \{(R^r, R^{r+۱})\}$$
$$\hat{S} \leftarrow \hat{S} \cup \{(R^{r+۱}, R^r)\}$$
$$tabu \leftarrow tabu \cup \{R^r, R^{r+۱}\}$$
$$S \leftarrow S \cup \hat{S}$$

۱۰. پایان.

۷.۳ استراتژی برای ترکیب روش‌های ACO و SS

پیاده سازی‌های انجام شده در دو بخش اخیر متشکل از روش ACO و SS که در آن روش SS بعد از اجرای روش ACO اعمال می‌شود. علاوه بر این استراتژی خاص، دیگر راهبردهای ترکیب هردو روش ممکن است در نظر گرفته شود، مثلا ترکیبی که در آن الگوریتم SS قبل از اعمال الگوریتم ACO اجرا می‌شود. با این حال، این جایگزین نیاز به اجرای اضافی، یعنی یک روش جدید برای تولید مجموعه اولیه جواب‌ها دارد. با این وجود پیشنهاد می‌شود راه‌های ترکیبی زیر ارائه شود:

۱. ترکیب ACO-SS۱ : این دو روش در چرخه‌های جداگانه‌ای اجرا می‌شوند، یعنی الگوریتم به دست آمده به عنوان دو چرخه: در مرحله اول ACO و در مرحله دوم SS اجرا می‌شود.

۲. ترکیب ACO-SS۲ : هر دو روش در یک چرخه معمولی اجرا می‌شود، به طور دقیق‌تر،روش SS در چرخه مربوط به روش ACO گنجانده شده است. ایده این است که از هر دو روش برای بهبود متقابل نتایج تولید شده به صورت جداگانه استفاده شود. به عبارت دیگر، پس از ساختن جواب توسط هر مورچه، روش SS برای بهبود و یا یافتن جواب‌های جدید که به مورچه‌های جدید در زیر چرخه‌ها کمک می‌کند، استفاده می‌شود.

۳. ترکیب ACO-SS۳ : مانند الگوریتم ACO-SS۲ است. به طور دقیق‌تر، پس از ترکیب ACO-SS۲ روش SS دوباره اجرا می‌شود.

۸.۳ نتایج محاسباتی

نمونه‌های MOKP-۱ که به صورت تصادفی با استفاده از نمونه تولید شده است، توسط کلینگمن [۹] و همکاران در [۱۳] ارائه شده است و موارد مربوط به MOKP-۲ مواردی را

[9] Klingman

که زیتلر [10] در [۱۴] تعریف شده را استفاده می‌کنند. این موارد و نتایج الگوریتم پیشرفت در (وب سایت مؤسسه فناوری زوریخ) قابل دسترسی است. برای مقایسه عملکرد از Cـاندازه که در زیتلر و تائل [11] در [۱۵] معرفی شده‌است استفاده می‌کنیم:

$$C(S', S'') = \frac{|\{z'' \in S'' : \exists z' \in S'; z' \succcurlyeq z''\}|}{|S''|}$$

که $z' \succcurlyeq z''$ یعنی برای هر $i \in \{1, \ldots, m\}$، $z''_i \leq z'_i$ است، یعنی بردار z'' توسط z' کارای ضعیف است. وقتی که $C(S', S'') = 1$ ، یعنی این‌که همه نقاط در S'' با نقاط S' برابرند و یا غالب هستند؛ در حالی‌که اگر $C(S', S'') = 0$ ، یعنی این‌که هیچ کدام از نقاط در S'' توسط مجموعه S' پوشیده نمی‌شود. توجه کنید که $C(S', S'')$ و $C(S'', S')$ باید در نظر گرفته شود، زیرا: $C(S', S'') \neq 1 - C(S'', S')$. اجرای برنامه‌ها بر اساس مقادیر به دست آمده توسط Cـاندازه مربوطه به ۱۰ اجرا انجام می‌شود. پارامترهای الگوریتمی در جدول ۱.۳ نشان داده شده است.

جدول ۱.۳: تنظیمات پارامتر

variant	α	β	Aco	SS	Ants	ρ	$\tau_{\min}$	$\tau_{\max}$	$\lvert R\rvert$
ACO-SS1	1	4	4000	60	100	0.01	0.01	6	20
ACO-SS2	1	4	2500	8	100	0.01	0.01	6	20
ACO-SS3	1	4	2000	10	100	0.01	0.01	6	20

۹.۳ مقایسه انواع مختلف

در جدول ۲.۳، مقایسه‌ای بین سه الگوریتم ACO-SS در Cـاندازه در مورد موارد ۱-MOKP صورت گرفته است. از این جدول می‌توان دید که ACO-SS۳ در مقایسه با ACO-SS۲ و ACO-SS۱ در مثال n۱۰۰l۲ با ۲ هدف و ۱۰۰ مورد، نوعی نتایج بهتر تولید می‌کند. دقیقا حدود ۷۷/۴٪ از نتایج ACO-SS۲ توسط ACO-SS۳ پوشش داده می‌شود، در حالی‌که تنها ۷۴/۲٪ از نتایج ACO-SS۳ توسط ACO-SS۲ پوشش داده می‌شود. ۷۸/۶٪ از نتایج ACO-SS۱ توسط ACO-SS۳ تحت پوشش قرار می‌گیرند و تنها ۷۸/۳٪از نتایج ACO-SS۳ کارای ضعیف یا برابر با ACO-SS۱ هستند. به طور مشابه برای نمونه n۱۰۰l۳ مشاهده شده است که نتایج الگوریتم ACO-SS۳ همچنان بهترین گزینه است، حدود از نتایج ACO-SS۱ و از نتایج ACO-SS۲ کارای ضعیف هستند یا توسط ACO-SS۳ بدست آمده‌اند، در

[10] Zitzler
[11] Thiele

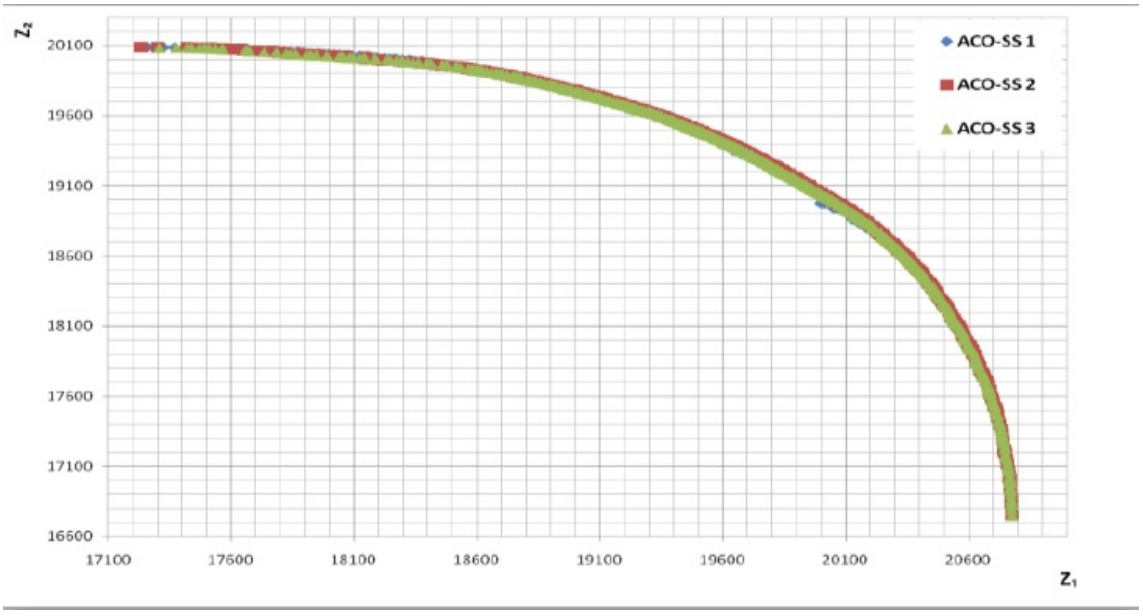
حالی‌که تنها ۳۹٫۸٪ و ۶۲٫۳٪ از نتایج ACO-SS۳ با آن‌هایی که از ACO-SS۱ و ACO-SS۲ به ترتیب پوشیده می‌شوند. با این حال، برای مورد $n۲۵۰l۲$ الگوریتم ACO-SS۱ به نظر می‌رسد نتایج بهتری نسبت به ACO-SS۱ و ACO-SS۳ دارد. در بقیه موارد، ACO-SS۲ به شدت بهتر از دو الگوریتم دیگر است. بنابراین یک تعادل مشخص بین الگوریتم‌های ACO-SS۲ و ACO-SS۳ وجود دارد، که هر دو در دو مورد به طور دقیق بهتر از دیگری است. مقایسه عملکرد سه الگوریتم اعمال شده به Mokp-۲، مشاهده می‌شود که هیچکدام از الگوریتم‌ها به طور کامل بهتر از دیگری در تمام موارد مورد آزمایش نیستند، یعنی، تعادلی بین الگوریتم‌های ACO-SS۱ و ACO-SS۲ وجود دارد، هر دو عملکرد خوبی را در همان تعداد مورد نشان می‌دهند (جدول ۳.۳ را مشاهده کنید) الگوریتم ACO-SS۱ در مثال‌های $ztn۲۵۰l۲, ztn۵۰۰l۲$ و $ztn۷۵۰l۲$؛ و در حالی‌که ACO-SS۲ عملکرد بهتری را در موارد $ztn۱۰۰l۲, ztn۱۰۰l۳$ و $ztn۲۵۰l۳$ نشان می‌دهد. همچنین دیده می‌شود که ACO-SS۳ به هیچ وجه بهتر از ACO-SS۱ و ACO-SS۲ در هر مورد آزمایش نیست. شکل ۲.۳ و ۳.۳ سطوح سازش توسط ACO-SS۱ و ACO-SS۲ و ACO-SS۳ را برای موارد دو هدفه با ۵۰۰ بخش برای MOKP-۱ و MOKP-۲ نشان می‌دهد. مجموعه‌های غیرمغلوب بیش از ده بار اجرا می‌شود. این ارقام نشان می‌دهد که برای MOKP-۱ یک توازن کوچک بین این سه نوع وجود دارد، در حالی‌که برای MOKP-۲ ما می‌دانیم که ACO-SS۱ به وضوح بهترین الگوریتم است.

جدول ۲.۳: مقایسه سه نوع ACO-SS با استفاده از Cـ اندازه‌گیری (MOKP-۱)

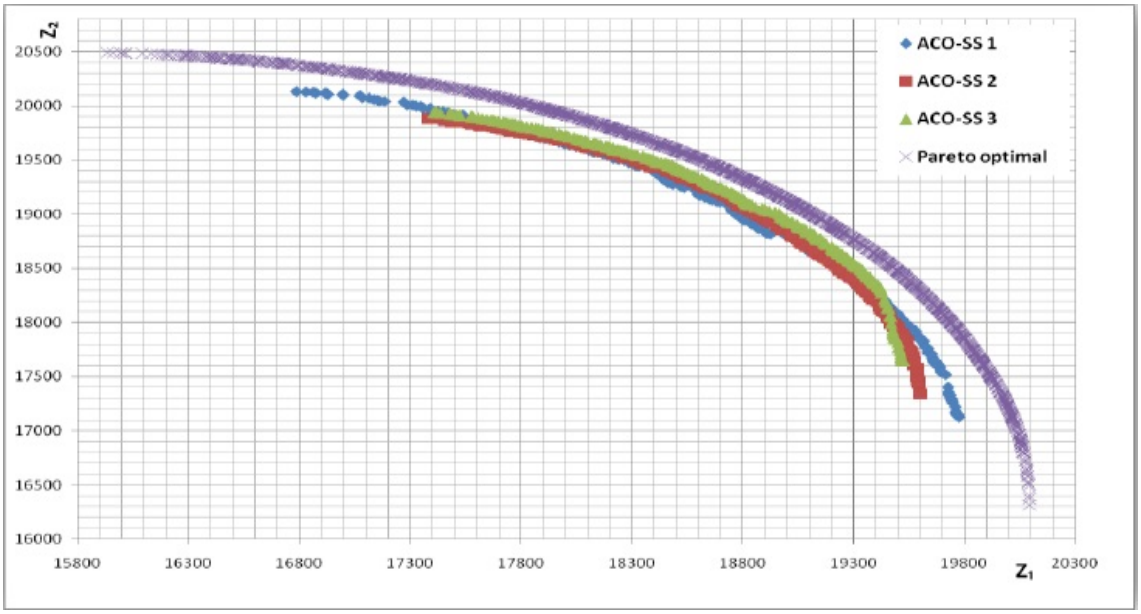
	Instance	C(1,2)	C(2,1)	C(1,3)	C(3,1)	C(2,3)	C(3,2)
Average	n100l2	0.776	0.757	0.783	0.786	0.742	0.744
Max		0.846	0.831	0.883	0.929	0.914	0.903
Min		0.724	0.613	0.667	0.636	0.571	0.651
Average	n100l3	0.425	0.713	0.398	0.718	0.623	0.660
Max		0.498	0.779	0.494	0.758	0.680	0.722
Min		0.378	0.628	0.356	0.647	0.535	0.555
Average	n250l2	0.661	0.437	0.649	0.465	0.501	0.610
Max		0.888	0.750	0.876	0.629	0.856	0.884
Min		0.357	0.169	0.497	0.222	0.218	0.281
Average	n500l2	0.396	0.413	0.433	0.364	0.508	0.461
Max		0.618	0.646	0.628	0.452	0.863	0.801
Min		0.201	0.185	0.302	0.202	0.200	0.145
Average	n750l2	0.286	0.346	0.322	0.296	0.550	0.374
Max		0.457	0.504	0.427	0.411	0.913	0.646
Min		0.151	0.213	0.163	0.057	0.294	0.054

جدول ۳.۳: مقایسه سه نوع ACO-SS با استفاده از Cـ اندازه‌گیری (MOKP-۲)

	Instance	C(1,2)	C(2,1)	C(1,3)	C(3,1)	C(2,3)	C(3,2)
Average	ztn100l2	0.497	0.552	0.550	0.519	0.594	0.494
Max		0.831	0.877	0.924	0.939	0.841	0.758
Min		0.231	0.284	0.191	0.194	0.414	0.186
Average	ztn100l3	0.337	0.431	0.374	0.388	0.462	0.383
Max		0.764	0.635	0.588	0.723	0.870	0.735
Min		0.104	0.096	0.068	0.167	0.141	0.025
Average	ztn250l2	0.648	0.287	0.682	0.234	0.433	0.503
Max		1.000	0.932	1.000	0.846	0.990	1.000
Min		0.007	0.000	0.016	0.000	0.000	0.000
Average	ztn250l3	0.030	0.553	0.038	0.448	0.437	0.382
Max		0.092	0.732	0.100	0.595	0.831	0.895
Min		0.005	0.469	0.008	0.219	0.037	0.020
Average	ztn500l2	0.583	0.254	0.475	0.325	0.374	0.551
Max		1.000	0.628	1.000	0.772	1.000	1.000
Min		0.139	0.000	0.004	0.000	0.000	0.000
Average	ztn750l2	0.433	0.139	0.461	0.168	0.367	0.560
Max		0.718	0.353	0.788	0.693	1.000	1.000
Min		0.161	0.000	0.013	0.000	0.000	0.000



شکل ۲.۳: سطوح سازش برای مثال با ۵۰۰ شئ (MOKP-۱)



شکل ۳.۳: سطوح سازش برای مثال با ۵۰۰ شئ (MOKP-۲)

۱۰.۳ مقایسه ACO-SS با الگوریتم‌های تکاملی

در اینجا برای هر نمونه از MOKP-۲ که دارای عملکرد بهتر در بخش قبلی بوده، با الگوریتم‌های ژنتیک FFGA در [۱۶] و HLGA در [۱۷] و NPGA در [۱۸] و NSGA در [۱۹] و الگوریتم تکاملی SPEA در [۱۵] و VEGA در [۲۰] آمده است، مقایسه‌ای صورت گرفته است.
هر ستون جدول ۴.۳ مقادیر Cـ اندازه‌گیری را مطابق با مقایسه بین الگوریتم ACO-SS (ستون ۱) و مجموعه‌ای از الگوریتم‌ها نشان می‌دهد:
ستون دوم مربوط به FFGA و ستون سوم مربوط به HLGA و چهارم مربوط به NPGA و پنجم NSGA و ششم SPEA و هفتم VEGA است. از این جدول می‌توان به وضوح نتیجه گرفت که نتایج حاصل از روش‌های ACO-SS به وضوح بهتر از نتایج الگوریتم‌های دیگر است. هیچ سناریویی وجود ندارد که نتایج حاصل از الگوریتم FFGA و HLGA نتایجی را که از الگوریتم ACO-SS حاصل شده است را پوشش دهد؛ در حالی‌که نتایج ACO-SS ، ۱۰۰٪ از این الگوریتم‌ها را پوشش می‌دهد. در الگوریتم‌های دیگر، تنها بخش کوچکی از نتایج ACO-SS کارای ضعیف هستند که بدیهی است که آن‌ها را بهتر از ACO-SS نمی‌کند. تنها استثنا در مثال $ztn۲۵۰l۲$ یافت می‌شود که نتایج زیتلر و تائل بهتر از ACO-SS است. در حدود ۱۸/۷٪ از نتایج روش SPEA کارای ضعیف هستند یا برابر با الگوریتم ACO-SS هستند، در حالی‌که حدود ۵۹٪ از نتایج ACO-SS با استفاده از روش SPEA پوشش داده می‌شود.

۱۱.۳ نتیجه‌گیری

در این فصل، الگوریتم ترکیبی مبتنی بر روش ACO و SS طراحی شده است که برای حل دو نوع مسأله کوله‌پشتی چندهدفه MOKP-۱ و MOKP-۲ ارائه شده است. در روش‌های پیشرفته، نتایج ابتدا توسط الگوریتم ACO تولید می‌شود و سپس توسط روش SS تدوین شده و تشدید می‌شود، که به سه روش منجر می‌شود:

۱. ACO-SS۱
۲. ACO-SS۲
۳. ACO-SS۳

اولین مورد شامل اجرای هر دو روش در دو چرخه مختلف می‌باشد، جایی که اول مربوط به الگوریتم ACO و دوم SS می‌باشد. در مورد دوم، دو روش برای بهبود متقابل نتایج تولید شده به صورت جداگانه استفاده می‌شود. روش SS در هر چرخه روش ACO نامیده می‌شود. در نهایت، مورد سوم شامل گسترش ACO-SS۲ جایگزین است، یعنی SS در پایان دوباره اجرا می‌شود.

در مقایسه با روش‌های دیگر روش ACO-SS بهترین الگوریتم تقریبا در مورد آزمایشات نتایج حاصل از آزمایش‌های محاسباتی نشان می‌دهد که ترکیبی از روش‌های ACO و SS قادر به ارائه نتایج جالبی است.

جدول ۴.۳: مقایسه بهترین الگوریتم‌های تکاملی با استفاده از Cـاندازه‌گیری (MOKP-۲)

	Instance	(1)-(2)	(2)-(1)	(1)-(3)	(3)-(1)	(1)-(4)	(4)-(1)	(1)-(5)	(5)-(1)	(1)-(6)	(6)-(1)	(1)-(7)	(7)-(1)
Average	ztn100l2	1.000	0.000	1.000	0.000	0.991	0.005	0.882	0.058	0.630	0.248	0.904	0.039
Max		1.000	0.000	1.000	0.000	1.000	0.051	1.000	0.182	0.947	0.446	0.966	0.082
Min		1.000	0.000	1.000	0.000	0.909	0.000	0.714	0.000	0.456	0.000	0.765	0.000
Average	ztn100l3	1.000	0.000	0.999	0.000	0.996	0.001	0.991	0.001	0.928	0.019	0.966	0.006
Max		1.000	0.000	1.000	0.000	1.000	0.005	1.000	0.007	0.994	0.060	1.000	0.022
Min		1.000	0.000	0.989	0.000	0.984	0.000	0.973	0.000	0.767	0.000	0.911	0.000
Average	ztn250l2	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000	0.187	0.590	1.000	0.000
Max		1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000	0.596	0.833	1.000	0.000
Min		1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000	0.000	0.285	1.000	0.000
Average	ztn250l3	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
Max		1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
Min		1.000	0.000	1.008	0.000	1.000	0.000	1.000	0.997	0.000	1.000	0.000	0.005
Average	ztn500l2	-	-	1.000	0.000	1.000	0.000	1.000	0.000	0.927	0.014	1.000	0.000
Max		-	-	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.081	1.000	0.000
Min		-	-	1.000	0.000	1.000	0.000	1.000	0.000	0.667	0.000	1.000	0.000
Average	ztn750l2	-	-	1.000	0.000	1.000	0.000	1.000	0.000	0.776	0.000	1.000	0.000
Max		-	-	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
Min		-	-	1.000	0.000	1.000	0.000	1.000	0.000	0.556	0.000	1.000	0.000

مراجع

[1] M. Visee, J. Teghem, E.L. Ulungu, ”Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem” **Journal of Global Optimization** , 12 (1998) 139–155.

[2] M. Captivo, J. Clımaco, J. Figueira, E. Martins, J.L. Santos, **Solving multiple criteria 0-1 knapsack problems using a labeling algorithm, Computers , Operations Research** 30 (2003) 1865–1886.

[3] G.R. Bitran, ”Theory of algorithms for linear multiple objective programs with zero-one variables”**Mathematical Programming** 17 (1979) 362–390.

[4] R.F. Deckro, E.P. Winkofsky, ”Solving zero-one multiple objective programs through implicit enumeration” **European Journal of Operational Research** 12 (1983) 362–374.

[5] F.H.F. Liu, C.C. Huang, Y.L. Yen,” Using DEA to obtain efficient solutions for multi-objective 0-1 linear programs”**European Journal of Operational Research** 126 (2000) 51–68.

[6] G.R. Jahanshahloo, F. Hosseinzadeh, N. Shoja, G. Tohidi, ”Solving the bi-objective zero-one knapsack linear programming problem by generating efficient solutions”**Applied Mathematical Computations** 169 (2005) 874–886.

[7] Alaya, I., Solnon, C., Ghedira, K. (2007). ”Ant Colony Optimization for Multi-objective Optimization Problems”. **19th IEEE International Conference on Tools with Artificial Intelligence** , pp. 450-457.

[8] Gomes da Silva, C., Climaco, J., Figueira, J. (2006). ”A Scatter Search Method for Bi-criteria 0,1-Knapsack Problems”. **European Journal of Operational Research** , 169, pp. 373-391.

[9] Visée, M., Teghem, J., Ulungu, E.L. (1998). ”Two-Phases Method and Branch and Bound Procedures to solve the Bi-objective Knapsack Problem”. **Journal of Global Optimization** , 12, pp. 139-155.

[10] Ghosh, A. (2004).” Evolutionary Algorithms for Multi-Criterion Optimization: A Survey”. **International Journal of Computing , Information Sciences**, 2 (1), pp. 38-57.

[11] Solving the Multi-objective Knapsack Problems using the Metaheuristic techniques Models, Implementations and Results. Quebo Kenge Clemente April 2010.

[12] Dorigo M, Di Caro G. (1999). ”The Ant Colony Optimization Meta Heuristic”. In D. Corne, M. Dorigo, and F. Golver, editors, **New Ideas in Optimization**, pages 11-32. McGraw-Hill.

[13] Klingman, D., Napier, A., Stutz, A. (1974). NETGEN: ”A Program for Generating Large Scale Capacitated Assignment, Transportation, and Minimum Cost Flow Network Problems”. **Management Science** , 20(5), pp. 814-821.

[14] Zitzler, E. (1999). ”Evolutionary Algorithms for Multi-objective Optimization: Methods and Applications”. **PhD thesis, Swiss Federal Institute of Technology**, Zurich.

[15] Zitzler, E., Thiele, L. (1999). ”Multi-objective Evolutionary Algorithms: A Comparative Case Study, the Strength Pareto Approach”. **IEEE Transactions on Evolutionary Computation**, 3, pp. 257-271.

[16] Fonseca, C.M., Fleming, P.J. (1993). ”Genetic Algorithms for Multi-objective Optimization: Formulation, Discussion, Generalization”. **The Fifth International Conference on Genetic Algorithms** , pp. 416-423.

[17] Hajela, P., Lin, C. Y. (1992). ”Genetic Search Strategies in Multi-criterion Optimal Design”. **Structural Optimization** , 4, pp. 99-107.

[18] Horn, J., Nafpliotis, N., Goldberg, D.E. (1994). ”A Niched Pareto Genetic Algorithm for Multi-objective Optimization”. First IEEE Conference.

[19] Srinivas, N., Deb, K. (1994).” Multi-objective Optimization using Non Dominated Sorting in Genetic Algorithms”. **Evolutionary Computation**, 2, pp. 221-248.

[20] Schaffer, J. (1985). ”Multiple Objective Optimization with Vector Evaluated Genetic Algorithms”. **International Conference on Genetic Algorithms, Lecture Notes in Computer Science**, pp. 93-100.

[21] Glover, F. (1977). Heuristics for Integer Programming Using Surrogate Constraints. **Decision Sciences**, 8 (1), pp. 156-166.

واژه‌نامه فارسی به انگلیسی

بهینه سازی ..... Optimization
بهینه سازی ترکیبی ..... Combinatorial Optimization
جستوجوی پراکنده ..... Scatter Search
چند هدفه ..... Multi objective
دو هدفه ..... Bi-objective
فراابتکاری ..... Metaheuristics
کارا ..... Efficient
کارای ضعیف ..... Weakly Efficient
کلنی(اجتماع) مورچه ..... Ant Colony
مسأله کوله‌پشتی ..... Knapsack problem

واژه‌نامه انگلیسی به فارسی

کلنی(اجنماع) مورچه ..... Ant Colony
دو هدفه ..... Bi-objective
بهینه سازی ترکیبی ..... Combinatorial Optimization
کارا ..... Efficient
مسأله کوله پشتی ..... Knapsack problem
فراابتکاری ..... Metaheuristics
چند هدفه ..... Multi objective
بهینه سازی ..... Optimization
جستوجوی پراکنده ..... Scatter Search
کارای ضعیف ..... Weakly Efficient

Aabstract

In this thesis, we propose a new procedure based on linear mathematical programming formulation to find all efficient solutions of bi-objective 0-1 knapsack problem. We solve a limited number of single objective problems to find efficient solutions of bi-objective 0-1 knapsack problem. In the end, algorithms have been developed to solve the multi-objective knapsack problem.

Keywords: Bi-objective 0-1 knapsack problem, Multi-objectove knapsack problem, Ant Colony Optimization, Scatter Search

Shahrood University of Technology

Faculty Of Mathematical Sciences

MSc Thesis in: Operation Research

Some Algorithms For Solving Multi Objective Knapsack Problem

By: Marzieh Abbasi

Supervisor

Jafar Fathali

Advisor

Mehrdad Ghaznavi

January 2019