روشی کارا برای کاهش فاصله ثانویه در حل نوع خاصی از مسئله کوله پشتی

کورش عشقی* و حسن جوانشیر**

دانشکده مهندسی صنایع، دانشگاه صنعتی شریف

واحد علوم و تحقیقات، دانشگاه آزاد اسلامی

(دریافت مقاله: ۸۲/۳/۲۸ – دریافت نسخه نهایی: ۸۳/۳/۲۴)

چکیده – **یکی از انواع مسئله کوله پشتی مسئله کوله پشتی جدایی پذیر غیر خطی نام دارد. این مسئله به دلیل کاربردهای فراوان مورد توجه محققان قرار گرفته است. یکی از روشهای اصلی حل این مسئله برنامه ریزی پویا است اما به دلیل آنکه فضای متغیر حالت به سرعت رشد می‌کند مشکل ابعادی را بوجود می‌آورد. در این مقاله روشی کارا ارائه می‌شود تا ضرایب جانشین را در هر مرحله از برنامه‌ریزی پویا بیابد و با این کار مسئله اصلی را به مسئله‌ایی با یک محدودیت موسوم به مسئله جانشین تبدیل کند. بر طبق نتایج محاسباتی حاصله حدود بالایی و پایینی ناشی از حل مسئله جانشین می‌تواند متغیرهای حالت بسیاری را در برنامه ریزی پویا حذف کرده و فاصله ثانویه را به نحو چشمگیری کاهش دهد.**

واژگان کلیدی : مسئله کوله پشتی جدایی پذیر، محدودیتهای جانشین، برنامه ریزی پویا

An Efficient Algorithm for Reducing the Duality Gap in a Special Class of the Knapsack Problem

K. Eshghi and H. Djavanshir
Industrial Engineering Department, Sharif University of Technology
Science and Research Branch, Islamic Azad University

Abstract: ***A special class of the knapsack problem is called the separable nonlinear knapsack problem. This problem has received considerable attention recently because of its numerous applications. Dynamic programming is one of the basic approaches for solving this problem. Unfortunately, the size of state-pace will dramatically increase and cause the dimensionality problem. In this paper, an efficient algorithm is developed to find surrogate multipliers in each stage of dynamic***

---

* – دانشیار ** – دانشجوی دکترا

[DOR: 20.1001.1.2251600.1384.24.1.4.7]

programming in order to transform the original problem to a single constraint problem called surrogate problem. The upper and lower bounds obtained by solving the surrogate problem can eliminate a large number of state variables in dynamic programming and extremely reduce the duality gap according to our computational results.

Keywords: ***Separable knapsack problem, Surrogate constraints, Dynamic programming***

۱- مقدمه

۱-۱- تعریف مدل

مسئله کوله پشتی جدایی پذیر غیرخطی[1] که در این مقاله به اختصارSNKP نامیده می شود با فرض غیرمنفی بودن تمام ضرایب و متغیرهای تصمیم‌گیری به صورت زیر تعریف می‌شود:

$$\textbf{SNKP}\quad \begin{cases} R = \text{Max} \quad \sum_{j=1}^{n} r_j(x_j) \\ \sum_{j=1}^{n} a_{ij}(x_j) \le b_i \quad i = 1,2,..,m \\ x_j \in S_j = \{1,2,\ldots..,k_j\} \quad j = 1,2,..,n \end{cases}$$

در این مدل فرض برآن است که تصمیم گیرنده می‌خواهد در n موضوع تصمیم‌گیری کند و هر موضوع شامل تعدادی آلترناتیو است. متغیر تصمیم‌گیری x_j می‌تواند یکی از مقادیر ۱و۲ و.... و k_j را اختیار کند که انتخاب q از بین آلترناتیوهای۱و۲و ... و k_j نشانگر آن است که آلترناتیو q برای موضوع j ام انتخاب و یا به عبارت دیگر $x_j=q$ شده است. مقدار برگشت انتخاب این آلترناتیو نیز با تابع غیرپیوسته $r_j(q)$ مشخص شده به نحوی که انتخاب این گزینه مقدار $a_{ij}(q)$ از منبع i ام را مصرف می‌کند. حداکثر مقدار منبع i ام نیز با b_i مشخص می‌شود. معمولاً تعداد آلترناتیوها برای تمام متغیرهای یک مسئله مساوی و برابرL فرض می‌شود. همواره یک آلترناتیو به نام "انجام هیچ کار" وجود دارد که در آن در حقیقت هم مصرف منابع و هم مقدار برگشتی صفر است. فرض می‌شود که در حالت عادی نتوان هیچ یک از آلترناتیوهای یک متغیر را با توجه به مصرف منابع و برگشتی، بر سایر آلترناتیوهای همان متغیر برتری داد. به عبارت دیگر آلترناتیوی وجود ندارد که با مصرف کمتر، سود بیشتری را نصیب ما سازد. بنابراین فرض می‌شود حذف آلترناتیوی به نفع آلترناتیو دیگر در ابتدا ممکن نیست. به همین دلیل معمولاً مقادیر برگشتی و مصارف منابع آلترناتیوهای هر متغیر به صورت صعودی مرتب می‌شوند تا بتوان به راحتی دریافت که هر آلترناتیو ضمن تولید برگشتی بیشتر، منبع بیشتری را نیز به کار گرفته است.

این مدل را می‌توان به صورت یک مسئله برنامه‌ریزی عدد صحیح با متغیرهای صفر و یک نیز مدلسازی کرد. در این صورت اگر r_{jk} را میزان برگشت آلترناتیو k ام از متغیر j ام، a_{ijk} را میزان مصرف آلترناتیو k ام از متغیر j ام و از منبع i ام فرض کنیم، می‌توان گفت مدل برنامه‌ریزی 1-0 مسئله SNKP به شکل زیر است که آن را به اختصار ZOKP می‌نامیم:

$$\textbf{ZOKP}\quad \begin{cases} R = \text{Max} \sum_{j=1}^{n}\sum_{k=2}^{k_j} r_{jk}\, x_{jk} \\ \sum_{j=1}^{n}\sum_{k=2}^{k_j} a_{ijk}\, x_{jk} \le b_i \quad i = 1,2,\ldots,m \qquad (۱) \\ \sum_{k=2}^{k_j} x_{jk} \le 1 \quad j = 1,2,\ldots,n \qquad (۲) \\ x_{jk} = 0,1 \quad k = 1,2,\ldots,k_j,\ j = 1,2,\ldots,n \qquad (۳) \end{cases}$$

محدودیتهای اصلیSNKP تشکیل محدودیتهای ردیف (۱) را درZOKP می‌دهند. همچنین برای جلوگیری از انتخاب بیش از یک آلترناتیو از هر متغیر، محدودیتهای ردیف (۲) اضافه شده‌اند که به نام محدودیتهای گزینش چندگانه[2] نامیده می‌شوند. باید یادآور شد که با تبدیل SNKP به شکل بالا، آلترناتیو "انجام هیچ کار" از هر متغیر، به دلیل آنکه ضرایب متغیر مربوطه چه در تابع هدف و چه در محدودیتها صفر است، به طور خودکار از مدل ZOKP حذف می‌شود و عدم انتخاب هیچ یک از آلترناتیوهای

یک متغیر، به منزلهٔ آلترناتیو "انجام هیچ کار" است. (توجه کنید که محدودیت (۲) به صورت نامساوی است.) آنچه که قابل توجه است، با این تبدیل ابعاد مسئله به شدت بزرگ می‌شود چرا که اگر مدل SNKP را شامل m محدودیت و n متغیر تصمیم‌گیری فرض کنیم که هر متغیر تصمیم گیری دارای L آلترناتیو (با احتساب آلترناتیو "انجام هیچ کار") باشد آن گاه مدل ZOKP معادل آن به دلیل آنکه به ازای هر متغیر یک محدودیت گزینش چندگانه را خواهد داشت دارای m+n محدودیت و n(L-1) متغیر خواهد بود.

۲- تاریخچهٔ علمی مسئله

مسئله SNKP به دلیل قابلیت تبدیل آن به شکلهای مختلف کاربرد گسترده‌ای در مسائل تخصیص منابع، بودجه بندی سرمایه، ضایعات برش و همچنین نظریه‌های برنامه‌ریزی عدد صحیح و پویا دارد[۳]. به همین خاطر ارائه یک روش کارا برای حل آن از دیرباز مورد نظر بسیاری از محققان بوده است. بیشتر این روشها متکی به استفاده از روش برنامه ریزی پویا برای حل SNKP هستند اما به دلیل رشد ابعاد متغیر حالت[۳] و نیاز فوق‌العاده به حافظهٔ رایانه در مورد مسائل با متغیرهای گسسته، اغلب آنها با محدودیت روبرو شده و تنها برای مسائل با ابعاد کوچک قابل اجرا هستند. ابتدا در مرجع[۵] الگوریتمی برای حل مسائل برنامه‌ریزی پویای چند بعدی در یک فضای کاهش یافتهٔ متغیر حالت به نام Imbedded – State Space ارائه شد. سپس در مرجع[۳] در طی الگوریتمی به نام (M & MDP) قابلیت رویکرد Imbedded – State را در یافتن جواب بهینهٔ مسائل SNKP نشان داده شد. در همان سال با توسعهٔ الگوریتمی موسوم به Hybrid DP / B&B استفاده از روش B & B برای کاهش حافظهٔ مورد نیاز در حل مدل برنامه‌ریزی پویا پیشنهاد شد[۴] که بر پایهٔ آن با داشتن حدود بالایی و پایینی جواب بهینهٔ مسئله می توان نقاط غیرکارای بیشتری را از فضای متغیر حالت حذف کرد که نرم افزار رایانه‌ای MMDP با استفاده از این ایده طراحی شد و در مرجع[۲] توسعه داده شد.

متاسفانه در برنامهٔ رایانه‌ای MMDP علی‌رغم کارایی آن تخمین حدود بالایی و پایینی اولیه بسیار ضعیف انجام می‌شد به گونه‌ایی که الگوریتم تنها برای مسائل کوچک و متوسط قابل استفاده بود. یک روش برای یافتن حد بالایی، انتخاب تصادفی تنها یکی از محدودیتها و صرف نظر کردن از سایر محدودیتها و حل مسئله تک محدودیتی حاصل از آن برای تعیین حد بالایی جواب بهینه است. این روش در حالات خاصی از مسائل SNKP از جمله مسئله کوله پشتی ۰-۱ در مرجع[۷] مورد بررسی قرار گرفته اما تنها قادر به تأثیر اندکی در حد بالایی است. به همین دلیل لزوم بررسی تکنیکهای دیگری در این رابطه احساس می‌شد که از جمله استفاده از محدودیتهای جانشین[۴] است که مورد توجه محققان واقع شد. در محدودیتهای جانشین تمام محدودیتها توسط یک محدودیت که از ترکیب خطی محدودیتهای اولیه به دست می‌آید جایگزین می‌شوند، که با این کار ما قادریم با حل مسئله تک محدودیتی حد بالایی (و نیز پایینی) مسئله اصلی SNKP را محاسبه کنیم.

در مرجع [۸] با ترکیب روش جستجوی ممنوع و برنامه‌ریزی خطی روشی برای حل مسئله ZOKP در حالت خاصی که محدودیتهای گزینش چندگانه وجود ندارد ارائه شد. در مرجع[۶] نیز روشی برای حل مسئله SNKP در حالت خاصی که درآن تنها یک محدودیت داریم طراحی شده است. در مرجع [۱] با بهره‌گیری از برنامه‌ریزی پویا و روش به روز در آوردن حد بالایی و پایینی الگوریتمی تحت نام "الگوریتم هایبرید اصلاح شده" توسعه داده شد. در الگوریتم مذکور ابتدا حدود بالایی و پایینی تعیین می‌شدند و ضمن حل برنامه ریزی پویا مسئله اصلی، این حدود تنگتر شده و رشد متغیرهای حالت را کند می‌ساختند. اگر چه با این روش حدود رشد متغیرهای حالت کندتر می‌شود، لیکن هنوز نمی‌تواند کاهش بسیار زیادی را به وجود آورد. از این رو در این مقاله با بهره‌گیری از

[DOR: 20.1001.1.2251600.1384.24.1.4.7]

محـدودیتهای جانشـین بـه یـافتن حـدود تنگتـر و قـویتری می‌پردازیم.

۳- الگوریتم Imbedded – State

الگـوریتم Imbedded – State یـک روش حـل بـرای مسـئله SNKP می باشد که به منظور آشنایی کامل با آن می توانیـد بـه مراجـع [۳ و ۵] مراجعـه کنیـد. چـون ایـن روش اسـاس کلیـه روشهایی است که تا کنون برای بهبود در حل مسئله SNKP بـا کمک برنامه‌ریزی پویا توسعه داده شده اسـت، بـه اختصـار بـه توضیح آن می‌پردازیم. در ایـن روش در ابتـدا مجموعـه G_j بـه صورت زیر تعریف می‌شود:

$$G_j= \{g_j \mid g_j = [a_{1j}(x_j) , a_{2j}(x_j) ,\ldots,a_{mj}(x_j)] , \forall x_j \in S_j \}$$

در این‌صورت فضای متغیر حالت در رویکردImbedded – State برای j امین مرحله به ترتیب زیر تعریف می‌شود:

$$F_1 = G_1$$
$$F_j = G_j \, O \, G_{j-1} \qquad j = 2 , 3 , \ldots , n$$

که در آن منظور از اپراتور "O " مجموع هر عنصر از G_j بـا هـر عنصر از G_{j-1} ، برای تمـام عناصـر موجـود در G_j و G_{j-1} اسـت. بنابراین F_j تمامی فضای ممکـن متغیـر حالـت را بـرای مراحـل ۱و۲و ... و n دربر می‌گیرد که باید با انجام آزمونهای قابل قبـول بودن[۵] و تفوق داشتن[۶] بهبود یابد. در نتیجـه آزمـون قابـل قبـول بودن، عناصر غیرقابل قبول مجموعـه F_j (عناصـری کـه مقـدار مصرف منابع آنها از میزان موجودی منابع بیشـتر اسـت) حـذف شده و مجموعه ای از نقاط قابل قبول در هر مرحلـه بـه دسـت می‌آید. اگر $F^f_j \subseteq F_j$ را به عنوان مجموعه نقاط قابـل قبـول از فضای ممکن متغیر حالت در jامین مرحله تعریف کنیم، در ایـن صورت:

$$F^f_j = \{ g_j' \mid g_j' \in (G_j \, O \, G_{j-1}) , g_j' \le (b_1,b_2,\ldots,b_m)\}$$

کـه در آن بـردار ($b_1 , b_2 ,\ldots , b_m$) مقـادیر موجـودی منـابع $m , \ldots , 2 , 1$(بردار سمت راست) را نشان می‌دهد. بـا اجـرای آزمون تفوق نیز نقاطی از فضای قابـل قبـول متغیـر حالـت کـه دارای مقدار برگشت کمتر با مصرف منابع بیشـتر (یـا مسـاوی) نسبت به سایر نقاط هستند، حـذف مـی‌شـوند. بنـابراین F^e_j بـه عنوان مجموعه‌ای از نقاط قابل قبول و کـارا بـه دسـت خواهـد آمـد. بـه عبـارت دیگـر ایـن مجموعـه در j – امـین مرحلـه $F^e_j \subseteq F^f_j \subseteq F_j$ است.

بنابراین در هر مرحله مانند j ابتـدا بایـد کلیـه نقـاط فضـای ممکن متغیر حالت F_j را یافت و سـپس بـا انجـام آزمـون قابـل قبول بودن، نقاط غیرقابل قبول را حذف و زیر مجموعـه F^f_j را از آن به دست آورد و سرانجام بـا انجـام آزمـون تفـوق، نقـاط غیرکارا نیز حذف و زیر مجموعه F^e_j را در j امین مرحله از حل مسئله برنامه‌ریزی پویا به دست آورد. این فرآینـد تسلسـلی بـه ازای $j = 1 , 2 , \ldots , n$ ادامه یافته تا زمانی که در n امـین مرحلـه F^e_n به دست آید. این مجموعه در واقـع مجموعـه کلیـه نقـاط قابل قبول و کارا برای مراحل $j =1,2,\ldots,n$ را در بر می‌گیـرد و لذا در ایـن مجموعـه نقطـه (و یـا نقـاطی) بـا بیشـترین مقـدار برگشت تابع هدف، جواب بهینه مسئله را نشان می‌دهد.

۴- مسئله جانشین

چون رویکرد اصلی در الگـوریتم Imbedded – State شـمارش کلیه جوابهای ممکن فضای متغیر حالت مسئله است، با افـزایش ابعاد متغیرهای تصـمیم‌گیـری، ابعـاد متغیرهـای حالـت نیـز بـه صـورت سرسـام آوری بـزرگ مـی‌شـود و اگـر چـه از طریـق آزمونهای قابل قبـول بـودن و تفـوق داشـتن فضـای متغیرهـای حالت تا حدی کاهش می‌یابد، اما در مسایل بزرگ همین مقـدار نیز در حجم حافظه‌های در دسترس رایانه‌های امـروزی در حـد معقولی نیست.

الگوریتم‌هایی به منظـور بهبـود Imbedded – State توسـعه داده شده است که از جمله می توان به الگوریتمهای ً هایبرید ً و ً هایبرید اصلاح شده ً اشاره کـرد کـه در مراجـع[۱، ۳ و ۵] بـه تفضـیل در مـورد آن سـخن رفتـه اسـت. ایـده اصـلی در ایـن الگوریتم تلفیق روش انشـعاب و تحدیـد (B&B) بـا رویکـرد Imbedded – State است، بدین معنی کـه بـا بـه دسـت آوردن

[DOR: 20.1001.1.2251600.1384.24.1.4.7]

حدود بالایی و پایینی برای مسئله اصلی، حجم زیادی از جوابهای غیرکارا را در همان مراحل اولیه حل مسئله حذف کرده و سپس با به هنگام‌سازی حدود مذکور، حذف جوابهای غیرکارا برای مراحل بعدی نیز ادامه می‌یابد.

از آنجایی که در تعیین حدود اولیه در الگوریتمهای فوق، از روشهای ساده‌ای بهره گرفته می‌شود، علی‌رغم بهبود نسبی در فضای ممکن متغیرهای حالت باز هم در مسائل با ابعاد بزرگ کاهش نسبتاً کوچکی در این فضا ایجاد شده و این الگوریتمها غیرکارا هستند.

آنچه که مسلم است، در مسئله برنامه‌ریزی کوله‌پشتی غیرخطی جدایی پذیر(SNKP) ، باید تا حد امکان فضای متغیر حالت کاهش یابد و این مهم به شدت وابسته به آن است که حدود اولیه تا آنجا که ممکن است، به جواب بهینه نزدیکتر باشند تا بتوان تعداد بیشتری از عناصر هر متغیر حالت را که غیرکارا هستند، حذف کرد.

هدف اصلی این مقاله، ارائه روشی برای تعیین حدود بالایی و پایینی مناسب برای به کارگیری در الگوریتم " هایبرید اصلاح شده" است. ساختار کلی این روش تبدیل مسئله اصلی به یک مسئله تک محدودیتی و استفاده از خواص موجود در مسئله تک محدودیتی برای تعیین حدود مسئله اصلی است.

برای توضیح بیشتر فرض کنید که مسئله تک محدودیتی حاصل از مسئله SNKP را که مسئله جانشین می نمایم به صورت زیر تعریف کنیم و با SP نشان دهیم:

$$SP \begin{cases} R_{SP} = \text{Max} \sum_{j=1}^{n} r_j(x_j) \\ \text{s.t :} \\ \sum_{i=1}^{m} \sum_{j=1}^{n} u_i a_{ij}(x_j) \le \sum_{i=1}^{m} u_i b_i \\ \sum_{i=1}^{m} u_i = 1 \\ u_i \ge 0 \ , \ i = 1,\ldots,m \ , \quad x_j \in S_j = \{1,2,\ldots,L\} \end{cases}$$

بردار $U = (u_1, u_2, \ldots, u_m)$ اصطلاحاً به بردار ضرایب جانشین[۷] معروف است و در این مقاله ما نحوهٔ یافتن آن را شرح خواهیم داد. اما قبل از آن بهتر است که به بررسی پاره‌ایی از خواص مسئله SP بپردازیم که به سادگی از روی تعریف به دست می‌آیند و در مرجع[۹] ذکر شده است:

خاصیت۱ : اگر X یک جواب قابل قبول مسئله اصلی باشد، این جواب حتماً در مسئله SP نیز قابل قبول خواهد بود.

خاصیت۲ : اگر X^* جواب بهینه مسئله SP باشد و در محدودیتهای مسئله اصلی نیز صدق کند آن گاه X^* جواب بهینه مسئله اصلی نیز خواهد بود. لازم به ذکر است که مسئله جانشین و خواص مربوط به آن در برنامه‌ریزی عدد صحیح کاربردهای فراوانی داشته که برای بررسی می‌توانید به مرجع[۹] رجوع کنید.

البته باید توجه داشت که هر جواب مسئله SP لزوماً در مسئله اصلی قابل قبول نیست و در واقع مسئلهSP مسئله کاهش یافته[۸] مسئله اصلی است. لذا می‌توان حدود پایینی و بالایی اولیه مناسبی را از طریق حل مسئله تک محدودیتیSP و به کارگیری آن در روش برنامه‌ریزی پویا به دست آورد. بدیهی است که اگر جواب بهینه مسئله SP محدودیتهای مسئله اصلی را نیز راضی کند، طبق خاصیت۲، جواب بهینهٔ مسئله اصلی نیز خواهد بود و اگر جواب بهینه مسئله SP در محدودیتهای مسئله اصلی صدق نکند، طبق خاصیت زیریک حد بالایی اولیه[۹] مناسب را برای جواب بهینه مسئله اصلی تولید می‌کند.

خاصیت۳ : اگر X^* جواب بهینه مسئله SP با مقدار برگشت $R_{sp}(X^*)$ باشد مقدار تابع هدف مسئله اصلی برای هر جواب قابل قبول X همواره کوچکتر یا مساوی$R_{SP}(X^*)$ است یعنی:

$$R(X) \le R_{SP}(X^*)$$

آشکار است که خاصیت۳ با توجه به خواص۱ و ۲ به دست آمده است. از طرف دیگر می‌توان با جستجو در جواب نهایی مسئله تک محدودیتیSP، از میان جوابهای متغیر حالت مرحله آخر(n ام) آن یک جواب قابل قبول برای مسئله اصلی جستجو کرد. این جواب نیز یک حد پایینی اولیه[۱۰] مناسب را برای جواب بهینه مسئله اصلی تولید می‌کند.

اما آنچه که اهمیت دارد آن است که بردار ضرایب جانشین

[DOR: 20.1001.1.2251600.1384.24.1.4.7]

را به نحوی بیابیم که حتی الامکان فاصلهٔ مابین حد بالا و جواب بهینه مسئله اصلی که اصطلاحاً به آن فاصله ثانویه[11] می‌گویند، صفر و یا بسیار کم شود تا در این صورت بتوان در رویکرد شاخه و کران مربوط به الگوریتم‌ هایبرید اصلاح شده، جوابهای جزیی[12] هر مرحله را با این حدود مقایسه کرده و در صورتی که خارج از آن باشد از ادامهٔ شاخه‌زنی خودداری ورزیده و گرهٔ مربوط به آن را بست و بدین طریق نقاطی که به جواب مناسبی رهنمون نمی‌شوند حذف می‌شوند. به عبارت دیگر هر چقدر محدودیت جانشین ما قویتر انتخاب شود فاصله ثانویه کمتر شده و به حل سریعتر مسئله اصلی منجر خواهد شد.

به دلیل آنکه مسئله SNKP دارای شکل جدایی پذیر بوده و غیر خطی است برای تعیین بردار جانشین آن از شکل تبدیل یافته آن یعنی ZOKP استفاده می‌کنیم. به علاوه برای ساده‌تر شدن و سرعت بخشیدن به روش یافتن بردار ضرایب جانشین شکل آزاد شده مسئله ZOKP که با حذف دسته محدودیتهای (۲) حاصل می‌شود را بررسی می‌کنیم:

$$LK\begin{cases} R_{LK} = \text{Max}\ \ r\,x \\ \text{s.t.} \quad\ \ Ax \le b \\ \qquad\ \ x = (0,1) \end{cases}$$

حال مسئله جانشین را برای مسئله فوق با کمک بردار U به صورت زیر به دست می‌آوریم:

$$LK^{(U)}\begin{cases} R^{(u)} = \text{Max}\ \ r\,x \\ \text{s.t.} \quad\ \ U\,(Ax-b) \le 0 \\ \qquad\ \ x = (0,1)\ ,\ U \ge 0 \end{cases}$$

مجموعه جوابهای شدنی مسئله $LK^{(U)}$ را نیز S می‌نامیم. چون به ازای هر $U \ge 0$ می‌توان یک $LK^{(U)}$ تشکیل داد، هدف ما آن است که بهترین U را انتخاب کنیم که این امر معادل با این است که از بین مقادیر حاصله برای تابع هدف مسئله $LK^{(U)}$ به ازای Uهای مختلف، بهترین بردار U انتخاب شود که تابع هدف را حتی الامکان کوچک کند .به عبارت دیگر بهترین بردار U از مسئله زیر حاصل می‌شود:

$$D_S\begin{cases} Z_{DS} = \text{Min}\ \ R^{(U)} \\ U \ge 0 \end{cases}$$

مسئله D_S به نام ثانویه جانشین خوانده می‌شود. به سادگی مشاهده می‌شود که $R^{(U)}$ یک تابع شبه محدب به ازای مقادیر مختلف U است.

قضیه : **به ازای هر $U \ge 0$ داریم: $Z_{DS} \ge R_{LK}$**

اثبات: فرض کنید متغیرهای بهینه مسئله LK را با X_{LK}^* نشان دهیم. در این صورت $AX_{LK}^* \le b$ است. با ضرب طرفین این نامعادله در در $U \ge 0$ داریم:

$$UAX_{LK}^* \le Ub \Rightarrow U(AX_{LK}^* - b) \le 0$$

به عبارت دیگر X_{LK}^* در مسئله $LK^{(U)}$ شدنی است. از طرف دیگر مسئله $LK^{(U)}$ یک مسئله از نوع بیشینه‌سازی است و در نتیجه هر جواب بهینه مسئله $LK^{(U)}$ به ازای بردارهای مختلف U یک حد بالایی را برای R_{LK} به دست می‌دهد:

$$\forall U \ge 0 : R_{LK} \le R^{(U)}$$

لذا بهترین مقدار $R^{(U)}$ را که همانا $Z_{DS} = \text{Min}\ R^{(U)}$ است نیز بهترین حد بالا برای R_{LK} محسوب می‌شود.

سئوالی که اکنون با آن مواجه هستیم آن است که با توجه به یک نقطه شروع، چگونه می‌توان افزاینده‌های جانشین را تولید و سپس بهبود بخشید و یا به عبارتی جهت نزولی مسئله جانشین چگونه به دست می‌آید (توجه کنید که ما قصد داریم به $\min R^{(U)}$ برسیم). برای این منظور به قضیه زیر توجه کنید:

قضیه : اگر $\overline{U} > 0$ بردار جانشین فعلی باشد، هر بردار d که به ازای $\alpha \ge 0$ در شرط زیر صدق کند جهت صعودی را برای مسئله $LK^{(U)}$ دارد یعنی در آن $R^{\overline{U}+\alpha d} \ge R^{\overline{U}}$

$$\begin{cases} d\,(AX-b) < 0 \\ X \in \Omega\,(D_S) \end{cases}$$

{ **توجه :** منظور از $\Omega(\ )$ ، فضای جواب مسئله ( ) است. }

اثبات : اگر $\underline{X}$ عضوی از $\Omega(LK^{(\overline{U})})$ باشد، با توجه به شدنی بودن $LK^{(\overline{U})}$ داریم:

$$\overline{U} \quad (A\underline{X} - b) \leq 0 \qquad (1)$$

حال اگر $d\,(A\underline{X} - b) < 0$ باشد، در این صورت با توجه به فرض $\alpha \geq o$ می‌توان گفت: $\alpha d\,(A\underline{X} - b) < 0$

در نتیجه با توجه به (۱) داریم: $(\overline{U} + \alpha d)\,(A\underline{X} - b) < 0$

یعنی $\underline{X}$ در مسئله $LK^{(\overline{U}+\alpha d)}$ نیز قابل قبول است و چون مسئله به صورت Max است، پس

$$R^{(\overline{U}+\alpha d)} \geq r\,\underline{X} = R^{(\overline{U})}$$

که در نتیجه $\overline{U} + \alpha d$ جهت صعودی می‌شود.

با توجه به قضیه فوق اگر بخواهیم بردار d را به نحوی تعیین کنیم که $\overline{U} + \alpha d$ جهت غیر صعودی داشته باشد باید $d\,(AX - b) > 0$ شود. به عبارت دیگر این شرط یک شرط لازم برای اینکه جهت غیر صعودی داشته باشیم است.

بدین ترتیب اگر $\overline{U}$ بهترین افزاینده جانشین نباشد، به ازای $X \in \Omega(LK^{(\overline{U})})$ باید d را به گونه‌ایی یافت که رابطه $(\overline{U} + \alpha d)\,(AX - b) > 0$ برقرار باشد که به معنی آن است که $X \in \Omega(D_S)$ در مسئله $LK^{(\overline{U}+\alpha d)}$ غیر قابل قبول می‌شود.

نکته‌ایی را که باید در نظر داشت آن است که اگر در تکرار k ام بخواهیم U_{k+1} را برای تکرار بعد تعیین کنیم باید تمام متغیرهایی را که از تکرار ۱ تا تکرار k ام حذف کردیم همچنان حذف شده باقی بمانند. برای این منظور از روشی مشابه روش بندرز[۱۳] استفاده می‌کنیم. یعنی مسئله را به شکل زیر تشکیل می‌دهیم :

$$\text{Max} \quad y$$
$$d\,(AX_j - b) > y \qquad j = 1, 2, \ldots, k,\ d \geq 0$$

بدین ترتیب که در هر تکرار تنها یک محدودیت به محدودیتهای قبلی مسئله بالا اضافه می‌شود. این کار سبب می‌شود که اولاً تمام $X \in \Omega\,(LK^{(U)})$ در $\Omega(LK^{(U+\alpha d)})$ غیرقابل قبول شده و ثانیاً با توجه به y می‌توان ادامه و یا توقف جستجو را تعیین کرد. بدین شکل که اگر $y \geq 0$ بود، همچنان به جستجوی خود ادامه می‌دهیم. ولی اگر $y \leq 0$ شد، یعنی اینکه دیگر هیچ افزاینده‌ایی نظیر U_{k+1} وجود ندارد که بتواند تمام نقاط $x_1, \ldots, x_k$ که قبلاً تولید شده است را حذف کند و در نتیجه متوقف می‌شویم.

۵- الگوریتم یافتن بردار جانشین U

تعاریف:

$X_k = LK^{Uk}$ جواب حاصل از حل مسئله

در تکرار k ام

Ω = مجموعه جواب

k = شمارنده تعداد تکرار الگوریتم

N = حداکثر تعداد تکرار

X^* = بهترین مقدار D_S تا تکرار فعلی

U_k = k بردار جانشین حاصل از تکرار

$g_k = AX_k - b$

گام ۰ : $k = 1$ قرار دهید. با یک مقدار اولیه غیرصفر برای U_1 شروع کنید و آن را نرمال کنید.

گام ۱ : مسئله $LK^{(U)}$ را به ازای $U = U_k$ حل کرده و جواب آن را X_k فرض کنید. اگر $g_k \leq 0$ شد متوقف شوید. X_k جواب نهایی مسئله LK^{Uk} است و $U^* = U_k$ قرار دهید. در غیر این صورت به گام بعدی بروید.

گام ۲ : مسئله برنامه ریزی خطی زیر را حل کنید و جواب آن را y و d فرض کنید.

$$\textbf{Max} \quad y$$
$$\text{s.t :} \quad dg_j \geq y$$
$$1d = 1$$
$$d \geq 0$$

گام ۳ : اگر $y \leq 0$ شد، متوقف شوید. در این صورت $U^* = U_K$ می‌شود.

در غیر این صورت α را به گونه‌ای تعیین کنید که $U_{k+1} = U_k + \alpha d$ شود و آن گاه $(U_k + \alpha d)\,(AX_k - b) > 0$ می‌شود. سپس U_{k+1} را نرمال کنید. $k=k+1$ قرار دهید و به گام ۱ برگردید.

الگوریتم فوق به دلیل آنکه فرض می‌شود مجموعه جواب، گسسته و محدود است دارای همگرایی معینی خواهد بود. چرا

که هر گاه گام ۳ موفقیت آمیز باشد به یک U جدید منجر می‌شود. در حقیقت عضو دیگری را از مجموعه جواب شدنی $LK^{(U)}$ حذف می‌کنیم. اگر $y \leq 0$ شد متوقف می‌شویم زیرا که در واقع هیچ گونه افزاینده U_{k+1} وجود ندارد که بتواند تمام نقاط قبلی $x_1 , x_2 , \ldots . x_k$ قبلی را حذف کند و لذا $U^* = U_k$ قرار داده و متوقف می‌شویم.

در مورد مقدار اولیه U_1 که باید در گام ۰ مشخص شود، می‌توان تمام عناصر موجود در U_1 را برابر $\frac{1}{m}$ قرار داد که در آن، m تعداد محدودیتها است.

پس از یافتن بردار جانشین U^* از طریق فوق باید با توجه به آن، مسئله اصلی SNKP را به شکل مسئله SP تبدیل کرد و با حل مسئله جانشین حدود بالایی و پایینی را برای مسئله SNKP تعیین کرد.

الگوریتم زیر به طور خلاصه نحوه کاربرد محدودیت جانشین را در حل مسئله SNKP به روش "هایبرید اصلاح شده" نشان می‌دهد.

۶- الگوریتم کاربرد محدودیت جانشین در حل SNKP

گام ۰ : ورودی: داده‌های مسئله اصلی SNKP

گام ۱: تعیین بردار جانشین با استفاده از "الگوریتم یافتن بردار جانشین U " برای مسئله LK که شکل آزاد شده مسئله اصلی است.

گام ۲: اعمال بردار جانشین به دست آمده برای مسئله SNKP و تبدیل آن به شکل مسئله جانشین SP

گام ۳: حل مسئله جانشین SP با استفاده از نرم افزار "هایبرید اصلاح شده". در صورتی که جواب مسئله جانشین در مسئله اصلی صدق کند جواب بهینه مسئله اصلی به دست آمده است در غیر این صورت حدود بالایی و پایینی برای مسئله اصلی تعیین می‌شود.

گام ۴: حل مسئله اصلی SNKP با استفاده از نرم افزار "هایبرید اصلاح شده" و حدود به دست آمده از گام قبل.

مثال:

$$\text{Max } Z = \begin{vmatrix} 0 \\ 2 \\ 3 \\ 5 \\ 8 \end{vmatrix} x_1 + \begin{vmatrix} 0 \\ 3 \\ 4 \\ 5 \\ 6 \end{vmatrix} x_2 + \begin{vmatrix} 0 \\ 6 \\ 9 \\ 11 \\ 13 \end{vmatrix} x_3 + \begin{vmatrix} 0 \\ 4 \\ 7 \\ 10 \\ 11 \end{vmatrix} x_4$$

$$\text{s.t.} \quad \begin{vmatrix} 0 \\ 6 \\ 8 \\ 9 \\ 11 \end{vmatrix} x_1 + \begin{vmatrix} 0 \\ 7 \\ 10 \\ 12 \\ 14 \end{vmatrix} x_2 + \begin{vmatrix} 0 \\ 8 \\ 10 \\ 12 \\ 15 \end{vmatrix} x_3 + \begin{vmatrix} 0 \\ 5 \\ 6 \\ 9 \\ 10 \end{vmatrix} x_4 \leq 26$$

$$\begin{vmatrix} 0 \\ 3 \\ 4 \\ 5 \\ 7 \end{vmatrix} x_1 + \begin{vmatrix} 0 \\ 4 \\ 6 \\ 8 \\ 10 \end{vmatrix} x_2 + \begin{vmatrix} 0 \\ 6 \\ 8 \\ 9 \\ 12 \end{vmatrix} x_3 + \begin{vmatrix} 0 \\ 4 \\ 8 \\ 12 \\ 15 \end{vmatrix} x_4 \leq 30$$

$$x_j = (1 , 2 , 3 , 4 , 5)$$

ابتدا مسئله را به شکل مسئله LK تبدیل می‌کنیم:

$$\text{Max } Z = 2x_{12} + 3x_{13} + 5x_{14} + 8x_{15} + 3x_{22} + 4x_{23} + 5x_{24} + 6x_{25} + \ldots + 4x_{42} + 7x_{43} + 10x_{44} + 11x_{45}$$

s.t.

$$6x_{12} + 8x_{13} + 9x_{14} + 11x_{15} + \ldots + 5x_{42} + 6x_{43} + 9x_{44} + 10x_{45} \leq 26$$

$$3x_{12} + 4x_{13} + 5x_{14} + 7x_{15} + \ldots + 4x_{42} + 8x_{43} + 12x_{44} + 15x_{45} \leq 30$$

$$x_{ij} = (0,1) \quad , \quad i = 1,\ldots,4 \; , \; j = 2,\ldots,5$$

حال فرض کنید $U_1 = (\frac{1}{2} \; , \; \frac{1}{2})$ باشد در این صورت محدودیت جانشین به صورت زیر درخواهد آمد:

$$4.5x_{12} + 6x_{13} + 7x_{14} + 9x_{15} + \ldots . + 4.5x_{42} + 7x_{43} + 10.5x_{44} + 12.5x_{45} \leq 28$$

که با حل مسئله کوله پشتی خطی فوق خواهیم داشت:

$$x_{34} = x_{43} = x_{44} = 1$$

و مقادیر بقیه متغیرها صفر و $z = 28$ می‌شود.

در این صورت بردار $G_1 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$ به دست می‌آید که در نتیجه آن باید مسئله برنامه‌ریزی خطی زیر را حل کرد:

$$\begin{aligned} & \text{Max } y \\ \text{s.t.} \quad & d_1 - d_2 \geq y \\ & d_1 + d_2 = 1 \\ & d_1, d_2 \geq 0 \end{aligned}$$

که با حل آن $d=(1,0)$ و $y=1>0$ می‌شود. حال باید مقدار α به گونه‌ایی تعیین شود که $(U_1+\alpha d)(Ax-b)>0$ شود به عبارت دیگر:

$$(U_1+\alpha d)(Ax-b)>0 \Rightarrow \left\{\left[\frac{1}{2},\frac{1}{2}\right]+\alpha[1,0]\right\}\begin{pmatrix}1\\-1\end{pmatrix}>0 \rightarrow \alpha>0$$

حال فرض کنید $\alpha=1$ قرار دهیم که در این صورت $U_2=(1.5,\ 0.5)$ شده که پس از نرمال کردن $U_2=(0.75,\ 0.25)$ می‌شود و محدودیت جانشین وابسته به آن به صورت زیر در می‌آید:

$$5.25x_{12}+7x_{13}+8x_{14}+10x_{15}+\ldots.+4.75x_{42}+6.5x_{43}+9.75x_{44}+11.75x_{45}\le 27$$

که با حل مسئله کوله پشتی خطی وابسته به این محدودیت خواهیم داشت:

$$x_{32}=x_{43}=x_{45}=1$$

و مقادیر بقیه متغیرها صفر و $z=24$ می‌شود. در این صورت چون بردار $G_2=\begin{pmatrix}-2\\-1\end{pmatrix}<0$ متوقف شده و بردار جانشین نهایی $U^*=(0.75,\ 0.25)$ است. حال با اعمال این بردار در مسئله SNKP آن را تبدیل به مسئله جانشین SP می‌کنیم:

$$\text{Max } Z=\begin{vmatrix}0\\2\\3\\5\\8\end{vmatrix}x_1+\begin{vmatrix}0\\3\\4\\5\\6\end{vmatrix}x_2+\begin{vmatrix}0\\6\\9\\11\\13\end{vmatrix}x_3+\begin{vmatrix}0\\4\\7\\10\\11\end{vmatrix}x_4$$

$$\text{s.t}\ \begin{vmatrix}0\\5.25\\7\\8\\10\end{vmatrix}x_1+\begin{vmatrix}0\\6.25\\9\\11\\13\end{vmatrix}x_2+\begin{vmatrix}0\\7.5\\9.5\\11.25\\14.25\end{vmatrix}x_3+\begin{vmatrix}0\\4.75\\6.5\\9.75\\11.25\end{vmatrix}x_4\le 26$$

$$x_j=(1,2,3,4,5)$$

که با حل این مسئله توسط الگوریتم "هایبرید اصلاح شده" جواب زیر به دست می‌آید:

$$x_{35}=x_{45}=1,\ z=24$$

چون این جواب در مسئله اصلی شدنی است پس جواب بهینه مسئله SNKP نیز هست.

۷- نتایج محاسباتی

برای بررسی و آزمایش الگوریتم ارائه شده در این مقاله نرم‌افزاری به زبان C تهیه شد که مطابق گامهای ذکر شده مسائل نمونه را حل می‌کند. این الگوریتم سپس برای مسایل نمونه و با استفاده از یک رایانه شخصی با پردازنده 450MHz و ۳۲ مگابایت حافظه اصلی اجرا شده است.

در جدول ۱ نتایج محاسباتی حاصل از میانگین حل در ۱۰ دسته مسالهٔ نمونه که دارای ابعاد متفاوتی‌اند ارائه شده است. در هر دسته مسئله نیز برای مسائل رده‌های TP1 الی TP5 ۱۰ نمونه مختلف و برای مسائل رده‌های TP6 الی TP10 ۴ نمونه مختلف مسالهٔ نمونه تولید شده و نتایج حاصل با توجه به میانگین نتایج در این نمونه‌ها به دست آمده است. بنابراین کلا ۷۰ مسئله نمونه تولید و بررسی شده‌اند. ابعاد مسائل نمونه به گونه‌ای در نظر گرفته شده است که قادر باشیم جواب بهینه اغلب آنها را با استفاده از نرم افزار"هایبرید اصلاح شده" برای حل مسئله SNKP نیز به دست آوریم.

کلیه ضرایب تابع هدف و محدودیتها در مسائل نمونه به صورت اعداد تصادفی یکنواخت در فاصله (۱۰و ۱۰۰) هستند. برای تعیین مقادیر سمت راست نیز یک عدد تصادفی در بین مجموع حداقل و حداکثر ضرایب هر محدودیت تولید شده است. در ستونهای جدول۱ به ترتیب از سمت چپ نام مسئله نمونه(TP)، تعداد موضوع تصمیم گیری (n)، تعداد محدودیتها(m) ، تعداد الترناتیوها(k)، میانگین درصد فاصله ثانویه در هر دسته از مسائل نمونه آن دسته (Gap %)، میانگین درصد اختلاف مابین حدود بالا و پایین حاصل از الگوریتم در هر دسته از مسائل نمونه آن دسته (Gap %)، بیشترین درصد فاصله ثانویه درمیان تمام مسائل نمونه آن دسته (WORST %D-Gap) میانگین زمان یافتن بردار جانشین برای هر مسئله برحسب ثانیه (T0) میانگین زمان یافتن جواب بهینه هر مسئله با استفاده از به کارگیری حدود بالایی و پایینی در نرم‌افزار "هایبرید اصلاح شده" برحسب ثانیه (T1) کل میانگین زمان یافتن جواب بهینه هر مسئله با استفاده از به کارگیری

[DOR: 20.1001.1.2251600.1384.24.1.4.7]

جدول ۱- نتایج محاسباتی حاصل از اجرای الگوریتم بروی ۱۰ دسته مسئله نمونه

TP	n	m	k	% D-Gap	% Gap	Worst % D-Gap	T0	T1	T2	T3	T2/ T3
TP1	10	10	5	3.02	7.04	9.061	33	109	142	201	0.63
TP2	10	10	10	6.43	14.97	14.05	74	143	217	463	0.58
TP3	15	10	5	7.19	17.04	22.05	88	159	247	652	0.38
TP4	20	10	5	9.94	23.25	26.11	111	320	431	810	0.53
TP5	30	10	5	12.45	26.28	34.06	460	753	1213	1943	0.62
TP6	50	5	7	16.01	34.34	39.88	512	872	1384	2297	0.60
TP7	50	8	10	20.42	39.22	42.71	890	1511	2401	****	-
TP8	40	10	15	17.63	33.04	55.04	698	1069	1767	2658	0.66
TP9	40	15	10	26.53	41.04	58.2	639	1260	1899	***	-
TP10	60	15	15	37.90	54.54	67.38	1012	1420	2432	***	-

در مواردی که در جدول علامت *** درج شده است به این معنی است که در بیش از ۴۰ ٪ از مسائل نمونه در آن رده تعیین جواب بهینه به دلیل رشد چشمگیر متغیرهای حالت مستقیما با استفاده از الگوریتم هایبرید اصلاح شده امکانپذیر نبوده است.

حدود بالایی و پایینی در نرم افزار"هایبرید اصلاح شده" برحسب ثانیه (T2) (که در واقع مجموع عناصر موجود در دو ستون قبلی است) و میانگین زمان یافتن جواب بهینه هر مسئله بدون استفاده از به کارگیری حدود بالایی و پایینی در نرم افزار "هایبرید اصلاح شده" برحسب ثانیه (T3) و میانگین درصد صرفه جویی در زمان حل مسئله که در واقع نسبت مقادیر موجود در ستون T2/ T3 است.

برای تعیین فاصله ثانویه یا D-Gap % از رابطه زیر استفاده شده است:

% D-Gap = (UB-OPT)/OPT

که در آن منظور از OPT و UB به ترتیب جواب بهینه مسئله اصلی و حد بالای حاصل از الگوریتم محدودیت جانشین است. چون علاوه بر فاصله ثانویه، تعیین میزان اختلاف مابین حدود بالایی و پایینی حاصل از الگوریتم نیز مفید است برای این منظور از مفهوم Gap % به صورت رابطه زیر استفاده شده است:

% Gap = (UB-LB)/OPT

که در آن منظور ازLB حد پایین حاصل از الگوریتم محدودیت جانشین است.

از کل ۷۰ مسئله نمونه در ۲۳ مسئله جواب حاصل از الگوریتم جانشین در مسئله اصلی صدق کرده و جواب بهینه مسئله بوده است.

هنگامی که با استفاده از حدود اعمال شده از الگوریتم جانشین حل نهایی مدل انجام می شود در تمامی نمونه‌های حل شده آزمون تفوق زودتر از سایر آزمونها در الگوریتم هایبرید اصلاح شده شروع به کار کرده و نقاط غیر کارا را حذف می‌کند. آزمون قابل قبول بودن نیز در هیچ موردی زودتر از ۴۰٪ مراحل حل فعال نشده است.

همان گونه که در جدول مشخص است با رشد ابعاد مسئله حل آن توسط نرم افزار "هایبرید اصلاح شده" دچار مشکل می‌شود به نحوی که برای اغلب مسائل موجود در رده‌های TP7و TP9و TP10 حل مسائل نمونه به دلیل رشد بسیار زیاد متغیرهای حالت مستقیما به وسیله الگوریتم "هایبرید اصلاح شده" امکانپذیر نیست اما با به کارگیری حدود حاصل از الگوریتم جانشین نقاط غیر کارا بسیاری در همان مراحل اول بررسی حذف شده به نحوی که الگوریتم هایبرید اصلاح شده قادر به حل تمامی نمونه‌های موجود در این رده‌ها شده است که با توجه به آن کارایی الگوریتم ارائه شده مشخص می‌شود. نکته جالب آنکه حتی در مواقعی که حل مسایل نمونه مستقیما

[DOR: 20.1001.1.2251600.1384.24.1.4.7]

توسط الگوریتم "هایبرید اصلاح شده" امکانپذیر است بازهم بـه کارگیری حدود به دست آمده از الگوریتم جانشین بـر سـرعت حل به نحو چشمگیری می‌افزاید به نحوی که بین ۲۵٪ تـا ۶۵٪ صرفه‌جویی در زمان حل مسایل نمونه خواهیم داشت.

واژه‌نامه

1. separable nonlinear knapsack problem
2. multiple choice constraints
3. state variable
4. surrogate constraints
5. feasibility test
6. dominancy test
7. surrogate multiplier vector
8. relaxation problem
9. initial upper bound
10. initial lower bound
11. duality gap
12. partial solutions
13. Benders' method

مراجع

1. Ghassemi Tari, F., "A Hybrid Algorithm for Discrete Nonlinear Optimization," *Amirkabir Journal of Science & Technology,* Vol. 5, No. 20, 63-68, 1992.
2. Marsten, R.E., and Morin,T.L., "A Hybrid Approach to Discrete Mathematical Programming," *Math. Programming*, Vol. 14,pp . 21-40,1978.
3. Morin,T.L., and Marsten, R.E., "An Algorithm for Nonlinear Knapsack Problem," *Management Science*, Vol. 22, No.10, pp. 1147-1158, 1976.
4. Morin,T.L., and Marsten, R.E., "Branch and Bound Strategies for Dynamic Programming," *Operations Research*, Vol. 24, No.4, 1976
5. Morin,T.L., and Esogbue, A. M. O., "The Imbedded State Space Approach to Reducing Dimensionality in Dynamic Programs of Higher Dimensions," *Journal of Mathematical Analysis and Applications*, Vol. 48, pp. 801-810, 1974.
6. Nakagawa,Y., and Iwasaki, A. , "Modular Approach for Solving Nonlinear Knapsack Problems," *IEICE Trans.*, Vol. E82, No. 9, pp. 1860-1864,1999.
7. Sarin, S., Karwan, M.H., and Rardin, R.L., "Surrogate Duality in a B&B Procedure for IP," *European Journal of Operational Research* , Vol. 33, pp. 326-333, 1988.
8. Vasquez, M., and Hao, J.K., "A hybrid Approach for 0-1 Multidimensional Knapsack Problem," *Proc. of IJCAI*, pp.328-333, 2001.
9. Wolsey, L.A., *Integer Programming*, John Wiley& Sons, 1998.

[DOR: 20.1001.1.2251600.1384.24.1.4.7]