حل مساله کوله پشتی چندبعدی با استفاده از اتوماتاهای یادگیر

سمیرا نوفرستی[۱] محمدرضا میبدی[۲]

[۱] دانشکده مهندسی کامپیوتر، دانشگاه مهندسی شهید نیکبخت، زاهدان، ایران

[۲] دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی امیرکبیر، تهران، ایران

چکیده: در این مقاله یک الگوریتم تکرارشونده مبتنی بر اتوماتاهای یادگیر برای حل مساله کوله‌پشتی چندبعدی پیشنهاد می‌شود. در این الگوریتم، مساله کوله‌پشتی با یک گراف کامل مدل می‌شود که هر گره از گراف متناظر با یکی از کالاهاست. هر گره از گراف به یک اتوماتای یادگیر مجهز است که انتخاب یا عدم انتخاب کالای متناظر با گره برای قرار گرفتن در کوله‌پشتی را مشخص می‌کند. نتایج شبیه‌سازی‌های نشان داده است که الگوریتم پیشنهادی را در مقایسه با الگوریتمهای موجود از کارایی بالاتری بر خوردار است. نتایج شبیه سازیها همچنین نشان داده است که الگوریتم پیشنهادی برای مسائل با اندازهای بزرگ دارای سرعت همگرایی بالایی میباشد.

کلید واژه: مساله کوله‌پشتی چندبعدی، اتوماتاهای یادگیر، مسایل مشکل

Solving Multidimensional Knapsack Problem using Learning Automata

[1]S. Noferesti [2]M. R. Meybodi

Computer Engineering Department, Shahid Nikbakht Engineering University, Zahedan, Iran[1]
Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran[2]

Abstract: In this paper an iterative algorithm based on learning automata for solving Multidimensional Knapsack Problem (MKP) is proposed. In the proposed algorithm the knapsack problem is modeled by a complete graph in which each node is associated to an object of MKP. Each node of the graph is equipped with a learning automaton with two actions which determines the presence or the absence of the corresponding object for locating in knapsacks. Simulation results have show that the proposed algorithm obtains better results than the existing algorithms. Simulation results have also shown that the proposed algorithm find the solution for large size problems in a reasonable running time.

Keywords: Multidimensional Knapsack Problem, Learning Automata, Hard Problems

۱- مقدمه

مساله کوله‌پشتی چندبعدی[1] یکی از مسایل بهینه‌سازی ترکیبی است که کاربردهای فراوانی از قبیل تخصیص منابع، برنامه‌ریزی بودجه، تخصیص سهام و بارگیری محموله‌ها دارد. مساله کوله‌پشتی چندبعدی شامل m کوله‌پشتی با ظرفیتهای $b_1,b_2,...,b_m$ و n کالا است. کالای iام دارای ارزش p_i می‌باشد و وزن w_{ji} از کوله‌پشتی m_j را اشغال می‌کند. هدف پر کردن کوله‌پشتی‌ها با زیرمجموعه‌ای از کالاها است به نحوی که بیشترین سود حاصل شود و مجموع وزن کالاهای یک کوله‌پشتی از ظرفیت آن تجاوز نکند. یک کالا یا در همه کوله‌پشتی‌ها قرار می‌گیرد یا برای هیچیک از کوله‌پشتی‌ها انتخاب نمی‌شود. به طور دقیقتر می‌توان مساله کوله‌پشتی را به صورت زیر تعریف کرد:

$$\begin{aligned} &\max \ imize && \sum_{j=1}^{n} p_j \, x_j \\ &subject && to \quad C_i : \sum_{i=1}^{n} w_{ij} \, x_j \le b_i && , \forall i \in 1..m \\ &x_j \in \{0,1\} && && , \forall j \in 1..n \end{aligned} \qquad (۱)$$

که x_j متغیر تصمیم‌گیری متناظر با کالای jام است. در صورتی که کالای jام انتخاب شود، x_j مقدار ۱ و در غیر ایـن صـورت

[1] Multiple knapsack problem

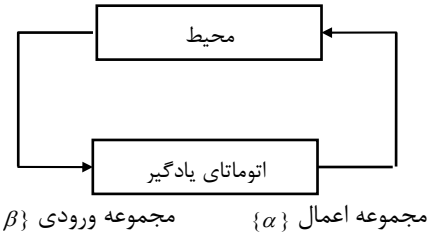
مقدار صفر را اختیار میکند. متغیرهای p_j، w_{ij} و b_i نمی‌توانند مقادیر منفی اختیار کنند.

به دلیل اهمیت مساله کوله‌پشتی تاکنون الگوریتمهای متعددی برای حل آن گزارش شده است. این الگوریتمها را می‌توان به دو گروه کلی تقسیم کرد: الگوریتمهای دقیق و الگوریتمهای تقریبی. با توجه به اینکه مساله کوله‌پشتی یک مساله NP_Complete می‌باشد الگوریتمهای دقیق که معمولاً از روشهای برش‌وانشعاب[۲] یا روشهای ترکیبی با برنامه‌نویسی پویا[۳] استفاده می‌کنند[۱،۲،۳،۴] در بدترین حالت دارای پیچیدگی نمایی هستند و برای استفاده در کاربردهای عملی مناسب نمی‌باشند. به همین دلیل الگوریتمهای تقریبی متعددی برای حل مساله کوله‌پشتی گزارش شده است. بسیاری از تحقیقات انجام شده به حل مساله کوله‌پشتی تک‌بعدی[۴] (m=1) پرداخته‌اند[۵،۶]. تلاشهای زیادی نیز در جهت حل تقریبی مساله کوله‌پشتی چندبعدی انجام گرفته است. از جمله الگوریتمهای تقریبی برای حل مساله کوله‌پشتی چندبعدی، الگوریتمهای تکرارشونده[۵] هستند. در این روشها رسیدن به یک پاسخ بهینه تضمین نمی‌شود اما در اغلب موارد جوابهای تقریبی قابل‌قبولی تولید می‌کنند. از جمله الگوریتمهای تکرارشونده می‌توان به الگوریتم ژنتیکی [۷،۸،۹] و الگوریتم کلونی مورچه‌ها [۱۰،۱۱] اشاره کرد.

در این مقاله یک الگوریتم تکرارشونده مبتنی بر اتوماتاهای یادگیر[۶] برای حل مساله کوله‌پشتی چندبعدی پیشنهاد می‌شود. در این الگوریتم مساله کوله‌پشتی با یک گراف کامل مدل می‌شود که هر گره از گراف متناظر با یکی از کالاهاست. هر گره از گراف به یک اتوماتای یادگیر با دو عمل مجهز است که انتخاب یا عدم انتخاب کالای متناظر برای قرار گرفتن در کوله‌پشتی را مشخص می‌کند. در هر تکرار از الگوریتم تعدادی عامل بر روی گره‌های گراف قرار داده می‌شوند که وظیفه فعالسازی اتوماتاهای یادگیر را برعهده دارند. هر عامل منجر به یک راه‌حل می‌شود. با توجه به ارزش راه‌حل بدست آمده از هر تکرار، بردار احتمالات اتوماتاهای یادگیر بروز می‌شود. این روند به دفعات تکرار می‌گردد و در خاتمه الگوریتم، بهترین راه‌حل بدست آمده به عنوان جواب نهایی انتخاب می‌شود. نتایج شبیه‌سازی‌های انجام گرفته کارایی الگوریتم پیشنهادی را در مقایسه با دیگر الگوریتمهای گزارش شده نشان می‌دهد. به علاوه برای مسائل با اندازه بزرگ الگوریتم پیشنهادی از سرعت همگرایی بالایی برخوردار است. ادامه مقاله بدین صورت سازماندهی شده است. در ابتدا در بخش ۲ اتوماتاهای یادگیر به صورت اجمالی معرفی می‌گردد. در بخش ۳ الگوریتم پیشنهادی برای حل مساله کوله‌پشتی ارائه می‌شود. در بخش ۴ نتایج شبیه‌سازی‌های انجام گرفته ارائه می‌شود و بخش پایانی مقاله نتیجه‌گیری می‌باشد.

۲- اتوماتاهای یادگیر

یک اتوماتای یادگیر [۱۲] یک مدل انتزاعی است که می‌تواند تعداد محدودی عمل را انجام دهد. هر عمل انتخاب شده توسط محیطی تصادفی ارزیابی می‌گردد و پاسخی به اتوماتای یادگیر داده می‌شود. اتوماتا از این پاسخ استفاده نموده و عمل خود را برای مرحله بعد انتخاب می‌کند. محیط تصادفی را می‌توان با سه‌تایی $E \equiv \{\alpha, \beta, c\}$ تعریف نمود که $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعه ورودی‌ها، $\beta \equiv \{\beta_1, \beta_2, ..., \beta_m\}$ مجموعه خروجی‌ها و $c \equiv \{c_1, c_2, ... c_r\}$ مجموعه احتمالهای جریمه شدن می‌باشند. هرگاه β_i دو مقداری باشد $\beta_i = 1$ به عنوان جریمه و $\beta_i = 0$ به عنوان پاداش در نظر گرفته می‌شود. c_i احتمال اینکه عمل α_i نتیجه نامطلوب داشته باشد، می‌باشد. در محیط پایدار مقادیر c_i بدون تغییر باقی می‌مانند، حال آنکه در محیط ناپایدار این مقادیر در طی زمان تغییر می‌کنند. شکل ۱ ارتباط بین اتوماتای یادگیر و محیط را نشان می‌دهد.



شکل ۱: ارتباط بین اتوماتای یادگیر و محیط

اتوماتاهای یادگیر به دو گروه اتوماتای یادگیر با ساختار ثابت و اتوماتای یادگیر با ساختار متغیر تقسیم می‌شوند. در ادامه به شرح

---

[2] Branch and cut
[3] Dynamic programming
[4] Uni-dimensional
[5] Iterative
[6] Learning Automata

مختصری درباره اتوماتاهای یادگیر با ساختار متغیر که در این مقاله استفاده شده است می‌پردازیم.

اتوماتای یادگیر با ساختار متغیر: اتوماتای یادگیر با ساختار متغیر توسط ۵تایی $LA \equiv \{\alpha, \beta, p, T, c\}$ نشان داده می‌شود که $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعه عملهای اتوماتای یادگیر، $\beta \equiv \{\beta_1, \beta_2, ..., \beta_m\}$ مجموعه ورودی‌های اتوماتای یادگیر، $p \equiv \{p_1, p_2, ..., p_r\}$ بردار احتمال انتخاب عملها، $T \equiv p(n+1) = T[\alpha(n), \beta(n), p(n)]$ الگوریتم یادگیری و $c \equiv \{c_1, c_2, ..., c_r\}$ احتمال جریمه شدن هر عمل می‌باشند. اگر در اتوماتای یادگیر عمل α_i در مرحله nام انتخاب شود و پاسخ مطلوب از محیط دریافت نماید، احتمال $p_i(n)$ افزایش یافته و سایر احتمال‌ها کاهش می‌یابند و برای پاسخ نامطلوب احتمال $p_i(n)$ کاهش یافته و سایر احتمال‌ها افزایش می‌یابند. تغییرات به گونه‌ای صورت می‌گیرد تا حاصلجمع $p_i(n)$ ها همواره ثابت و مساوی یک باقی بماند.

الف- پاسخ مطلوب

$$\begin{aligned} p_i(n+1) &= p_i(n) + \alpha[1 - p_i(n)] \\ p_j(n+1) &= (1-b)p_j(n) \qquad \forall j \quad j \neq i \end{aligned} \qquad (۲)$$

ب- پاسخ نامطلوب

$$\begin{aligned} p_i(n+1) &= (1-b)p_i(n) \\ p_j(n+1) &= \frac{b}{r-1} + (1-b)p_j(n) \qquad \forall j \quad j \neq i \end{aligned} \qquad (۳)$$

در روابط فوق، a پارامتر پاداش و b پارامتر جریمه می‌باشد. با توجه به مقادیر a و b سه حالت مختلف را می‌توان در نظر گرفت. زمانیکه a و b با هم برابر باشند، الگوریتم را L_{RP} [۷] می‌نامند. زمانیکه b از a خیلی کوچکتر باشد، الگوریتم را $L_{R\varepsilon P}$ [۸] و زمانیکه b مساوی صفر باشد، الگوریتم را L_{RI} [۹] می‌نامند. برای مطالعه بیشتر در رابطه با اتوماتاهای یادگیر با ساختار ثابت و متغیر می‌توان به [۱۲،۱۳] مراجعه نمود.

۳- حل مساله کوله‌پشتی توسط اتوماتاهای یادگیر

در الگوریتم پیشنهادی که آنرا MKPLA می‌نامیم، مساله با یک گراف کامل مدل می‌شود که هر گره از گراف متناظر با یک کالا در مساله کوله‌پشتی است. هر گره از گراف به یک اتوماتای یادگیر با دو عمل انتخاب کالا برای قرار گرفتن در کوله‌پشتی‌ها و عدم انتخاب کالا مجهز است.

برای فعالسازی اتوماتاهای یادگیر از عاملها استفاده می‌شود. در هر تکرار از الگوریتم تعدادی عامل وجود دارد که هر عامل یک راه‌حل را می‌سازد. در ابتدا یک عامل به صورت تصادفی بر روی یکی از گره‌های گراف قرار می‌گیرد و باعث فعالسازی اتوماتای یادگیر آن گره می‌شود. هرگاه یک اتوماتای یادگیر فعال می‌شود طبق بردار احتمالات انتخاب اعمال خود یکی از دو عمل خود را انتخاب می‌کند. در ابتدای الگوریتم احتمال انتخاب عمل حضور کالا در کوله‌پشتی‌ها برابر ۱ در نظر گرفته شده است تا کلیه کالاها شانس انتخاب را پیدا کنند. پس از فعال شدن یک اتوماتای یادگیر اگر عمل انتخابی آن اتوماتا، انتخاب کالا برای قرار گرفتن در کوله‌پشتی‌ها باشد، آن کالا را در لیست کالاهای انتخابی قرار می‌دهیم. سپس عامل از بین گره‌هایی که تاکنون پیمایش نشده‌اند یک گره را انتخاب می‌کند. عامل k گره بعدی را بر اساس رابطه ۴ انتخاب می کند.

$$p(j) = \frac{\eta_j}{\sum_{i \in validset} \eta_i} \quad , \quad \eta_j = \frac{p_j}{\sum_{i=1}^{m} \frac{w_{ij}}{s_i}} \qquad (۴)$$

که $validset$ مجموعه گره‌هایی است که تاکنون توسط عامل k پیمایش نشده‌اند و انتخاب آنها باعث انحراف از شرط (۱) نمی‌شود و s_i ظرفیت باقی‌مانده از کوله‌پشتی i تا این مرحله می‌باشد. در این رابطه عامل گرهی را انتخاب می‌کند که نسبت سود کالای متناظر با آن به مجموع وزن اشغال کننده از ظرفیت باقی‌مانده کوله‌پشتی‌ها ماکزیمم باشد. در رابطه فوق عامل k با احتمال q گره دارای بیشترین احتمال را انتخاب می کند و با احتمال $1\text{-}q$ گره بعدی را با احتمال متناظر با آن گره به صورت تصادفی انتخاب می‌کند. هر گره تنها یکبار توسط هر عامل پیمایش می‌شود. هرگاه مجموعه $vaildset$ یک عامل تهی شد کار آن عامل به پایان می‌رسد و راه‌حل ایجاد شده توسط آن عامل

---

[۷] Linear Reward Penalty

[۸] Linear Reward Epsilon Penalty

[۹] Linear Reward Inaction

طبق الگوریتم شکل ۲ بهبود می‌یابد. برای بهبود راه‌حل طبق [۱۴] مقادیر LP برای کالاها محاسبه می‌شود. سپس کالاها به ترتیب نزولی مقادیر LP بررسی می‌شوند. اگر بتوان کالایی را بدون انحراف از شرط (۱) برگزید، آن کالا به لیست کالاهای انتخابی کوله‌پشتی‌ها اضافه می‌شود.

```
Procedure improve ( x⃗ )
    sort x⃗ as x^LP[j] ≥ x^LP[j+1]
    for j = 1 to n do
       if x[j]=0 then x[j]=1;
    if any C_i is violated then x[j]=0;
```

شکل ۲: الگوریتم بهبود راه‌حل ایجاد شده توسط یک عامل

در هر تکرار از الگوریتم، از ارزش بهترین راه‌حل ایجاد شده توسط عاملها، برای ارزیابی عمل انتخابی اتوماتاهای یادگیر استفاده می‌شود. اگر ارزش کالاهای انتخابی بهترین عامل از ارزش بهترین لیست کالاهای بدست آمده تا این مرحله بیشتر باشد، اعمال انتخاب شده توسط اتوماتاهای یادگیر بهترین عامل تکرار فعلی از طریق افزایش احتمال انتخاب آنها پاداش داده می‌شود. فرمول یادگیری برای پاداش عمل انتخابی بصورت زیر می‌باشد:

$$p(t+1) = p(t) + \theta.a.(1 - p(t)) \qquad (۵)$$

که $\theta = |\frac{f(c) - c_{\max}}{f(c)}|$ و $f(c)$ مجموع ارزش کالاهای انتخابی است که در تکرار tام تولید شده و C_{max} ارزش بهترین راه‌حل بدست آمده تا این مرحله است. a پارامتر پاداش نامیده می‌شود.

اگر ارزش بهترین راه‌حل تکرار فعلی از ارزش بهترین راه‌حل بدست آمده تاکنون کمتر باشد اعمال انتخاب شده توسط اتوماتاهای یادگیر گره‌های متناظر با کالاهای شرکت کننده در بهترین راه‌حل پاداش و اعمال انتخاب شده توسط اتوماتاهای یادگیر دیگر گره‌ها جریمه می‌شوند. برای جریمه عمل انتخابی از فرمول زیر استفاده می‌شود.

$$p(t+1) = p(t) - \gamma.b.p(t) \qquad (۶)$$

که $\gamma = |\frac{1}{f(c) - c_{\max} + 1}|$ و b پارامتر جریمه نامیده می‌شود. کل این فرآیند به صورت تکراری و به دفعات انجام می‌شود تا زمانی که راه-حل بهینه حاصل شود یا شرط ماکزیمم تعداد تکرارهای الگوریتم برقرار شود. شبه کد الگوریتم MKPLA در شکل ۳ نشان داده شده است.

```
Algorithm MKPLA
 construct a complete graph that associates a node to each object;
 equip each node with an learning automaton with tow actions {1,2};
 set probability of selecting action 1 to 1;
 while not maximum number of cycles reached do
    for each agent k, construct a solution Sl_k as follows:
      Randomly choose a first object O_i and active LA_i in node p_i;
      action = the action chosen by LA_i in node p_i
      if action=1 then Sl_k = Sl_k∪{O_i};  //select object
      validset={O_i€{1,2,..,n}|O_i∉Sl_k & can be selected without violating resource
                                                                  constraints}
      while validset<>∅ do
           Rand= a random variable € [0,1]
           if rand<=q then choose an object€ validset with maximum p(i)
           else choose an object€validset with probability ps(i)
           action = the action chosen by LA_i in node p_i
           if action=1 then Sl_k = Sl_k∪{O_i};  //select object
           remove from validset every object that violates some resource
                                                              constraints.
      end while
      improve Sl_k
    end for
    update probability vectors of learning automata
 end while
```

شکل۳: الگوریتم MKPLA برای حل مساله کوله پشتی چندبعدی

۴- نتایج شبیه‌سازی‌های انجام شده

برای ارزیابی کارایی الگوریتم MKPLA از مجموعه تست پیشنهادی Chu و Beasley [۱۵،۱۶] که در OR-library [۱۷] موجود است، استفاده شده است. نتایج حاصل از الگوریتم پیشنهادی با چهار الگوریتم دیگر که در [۱۱] گزارش شده‌اند، مقایسه شده است. جزئیات این الگوریتمها در مراجع [۱۰،۱۱،۱۶و۱۸] ارائه شده است.

در آزمایشات انجام گرفته مقدار پارامتر تصمیم‌گیری q برابر ۰/۳، مقدار پارامتر پاداش a برابر ۱، مقدار پارامتر α برابر ۱و مقدار پارامتر β برابر ۳ در نظر گرفته شده است. به علاوه تعداد عاملها در هر تکرار برابر ۳۰ انتخاب شده است. مقدار پارامتر جریمه b در ابتدا ۱ در نظر گرفته شده است اما اگر بهترین نتیجه حاصل شده در ۵۰ تکرار مجددا حاصل شد مقدار b به میزان ۰/۳ کاهش می یابد. جدول ۱ نتایج حاصل برای ۲۰ نمونه تست با ۱۰۰ کالا و ۵ کوله‌پشتی را نشان می‌دهد. ستون اول جدول شماره مساله در مجموعه تست، ستون دوم جدول بهترین جواب شناخته شده برای مساله که توسط الگوریتم ژنتیک حاصل شده است[15] و ستونهای بعدی نتایج حاصل از اجرای الگوریتمهای مذکور را نشان می‌دهد. نتایج حاصل از الگوریتم MKPLA شامل میانگین ۲۰ بار اجرای الگوریتم با ۱۰۰۰ تکرار در هر اجرا و متوسط تعداد تکرارهای لازم برای رسیدن به بهترین جواب (C*) می‌باشد. همانطور که از جدول ۱ مشخص است الگوریتم پیشنهادی جوابهای بهتری نسبت به الگوریتمهای گزارش شده تولید می‌کند.

جدول ۱: مقایسه نتایج الگوریتم MKPLA با دیگر الگوریتمهای گزارش شده برای ۱۰۰ کالا و ۵ کوله‌پشتی

شماره مساله	بهترین جواب شناخته شده	L. & M. (میانگین)	FIDANOVA (بهترین)	ALAYA میانگین	ALAYA (C*)	MKPLA (میانگین)	MKPLA (C*)
۰	۲۴۳۸۱	۲۴۳۳۱	۲۳۹۸۴	۲۴۳۴۲	۵۲۲	۲۴۳۶۴,۰۵	۴۵۳
۱	۲۴۲۷۴	۲۴۲۴۵	۲۴۱۴۵	۲۴۲۴۷	۴۶۹	۲۴۲۷۰	۳۲۲
۲	۲۳۵۵۱	۲۳۵۲۷	۲۳۵۲۳	۲۳۵۲۹	۴۸۳	۲۳۵۳۷,۲۵	۲۸۲
۳	۲۳۵۳۴	۲۳۴۶۳	۲۲۸۷۴	۲۳۴۶۲	۵۰۰	۲۳۴۹۵,۳	۳۳۵
۴	۲۳۹۹۱	۲۳۴۶۹	۲۳۷۵۱	۲۳۹۴۶	۵۸۹	۲۳۹۵۹,۹۵	۳۱۳
۵	۲۴۶۱۳	۲۴۵۶۳	۲۴۶۰۱	۲۴۵۸۷	۵۳۵	۲۴۶۰۲,۴,۹	۴۹۴
۶	۲۵۵۹۱	۲۵۵۰۴	۲۵۲۹۳	۲۵۵۱۲	۴۸۰	۲۵۵۴۹,۶۵	۴۰۸
۷	۲۳۴۱۰	۲۳۳۶۱	۲۳۲۰۴	۲۳۳۷۱	۵۰۹	۲۳۴۱۰	۱۴۵
۸	۲۴۲۱۶	۲۴۱۷۳	۲۳۷۶۲	۲۴۱۷۲	۵۷۱	۲۴۲۰۲,۴۵	۲۰۰
۹	۲۴۴۱۱	۲۴۳۲۶	۲۴۲۵۵	۲۴۳۵۶	۵۸۸	۲۴۴۰۸,۴۵	۹۶
۱۰	۴۲۷۵۷		۴۲۷۰۵	۴۲۷۰۴	۵۳۷	۴۲۷۰۵	۲۲۶
۱۱	۴۲۵۴۵		۴۲۴۴۵	۴۲۴۵۶	۵۷۷	۴۲۴۵۰,۹۵	۲۲۰
۱۲	۴۱۹۶۸		۴۱۵۸۱	۴۱۹۳۴	۶۳۵	۴۱۹۳۶,۸۵	۲۴۴
۱۳	۴۵۰۹۰		۴۴۹۱۱	۴۵۰۵۶	۶۲۷	۴۵۰۲۵,۴	۳۹۲
۱۴	۴۲۲۱۸		۴۲۰۲۵	۴۲۱۹۴	۵۱۲	۴۲۲۰۰,۴۵	۳۳۹
۱۵	۴۲۹۲۷		۴۲۶۷۱	۴۲۹۱۱	۴۸۴	۴۲۸۹۸,۱۵	۲۶۰
۱۶	۴۲۰۰۹		۴۱۷۷۶	۴۱۹۷۷	۴۵۸	۴۱۹۸۸,۷	۷۵
۱۷	۴۵۰۲۰		۴۴۶۷۱	۴۴۹۷۱	۴۹۰	۴۵۰۰۷,۹۳	۳۰۰
۱۸	۴۳۴۴۱		۴۳۱۲۲	۴۳۳۵۶	۵۱۴	۴۳۳۸۶,۲۷	۲۳۰
۱۹	۴۴۵۵۴		۴۴۴۷۱	۴۴۵۰۶	۵۱۷	۴۴۵۱۵,۷	۱۳۴

جدول ۲ نتایج حاصل برای ۲۰ نمونه تست با ۱۰۰ کالا و ۱۰ کوله‌پشتی را نشان می‌دهد. همانطور که از جدول مشخص است الگوریتم پیشنهادی در اکثر موارد به جوابهای بهتری دست یافته است. جدول ۳ نتایج حاصل برای ۵ نمونه تست با ۵۰۰ کالا و ۵ کوله‌پشتی را نشان می‌دهد. بهترین نتایج بدست آمده برای این مجموعه در [۲۰] گزارش شده است. در این مقاله برای حل مساله کوله‌پشتی یک الگوریتم ترکیبی که از جستجوی tabu و برنامه‌نویسی خطی استفاده می‌کند، گزارش شده است. نتایج حاصل از الگوریتم MKPLA شامل میانگین ۱۰ بار اجرای الگوریتم با ۱۰۰۰ تکرار در هر اجرا می‌باشد. همان‌طور که از جدول مشخص است الگوریتم پیشنهادی در همه موارد جوابهای بهتری نسبت به الگوریتم Alaya تولید کرده است. به علاوه الگوریتم پیشنهادی از سرعت همگرایی به مراتب بالاتری برخوردار است.

۵- نتیجه گیری

در این مقاله یک الگوریتم تکرارشونده مبتنی بر اتوماتاهای یادگیر برای حل مساله کوله‌پشتی چندبعدی پیشنهاد شد. نتایج بدست آمده از آزمایشها نشان داد که الگوریتم پیشنهادی برای حل مساله کوله‌پشتی از کارایی بالایی برخوردار است. نتایج شبیه‌سازی‌های انجام گرفته کارایی الگوریتم پیشنهادی را هم از لحاظ کیفیت جوابهای تولید شده و هم از لحاظ سرعت همگرایی به جواب در مقایسه با تعدادی از الگوریتمهای گزارش شده نشان می‌دهد.

جدول ۲: مقایسه نتایج الگوریتم *MKPLA* با دیگر الگوریتمهای گزارش شده برای ۱۰۰ کالا و ۱۰ کوله‌پشتی

شماره مساله	بهترین جواب شناخته شده	L. & M. (میانگین)	ALAYA میانگین	ALAYA C*	MKPLA میانگین	MKPLA C*
۰	۲۳۰۶۴	۲۲۹۹۶	۲۳۰۱۶	۵۳۸	۲۳۰۵۳,۴	۳۵۰
۱	۲۲۸۰۱	۲۲۶۷۲	۲۲۷۱۴	۵۷۵	۲۲۷۳۸,۱۳	۳۱۳
۲	۲۲۱۳۱	۲۱۹۸۰	۲۲۰۳۴	۵۹۸	۲۲۰۸۶,۳۸	۴۸۹
۳	۲۲۷۷۲	۲۲۶۳۱	۲۲۶۳۴	۷۰۰	۲۲۶۱۹,۷۵	۱۶۷
۴	۲۲۷۵۱	۲۲۵۷۸	۲۲۵۴۷	۶۴۰	۲۲۵۹۶	۴۰۱
۵	۲۲۷۷۷	۲۲۵۶۵	۲۲۶۰۲	۶۴۵	۲۲۶۲۸,۲۳	۲۴۴
۶	۲۱۸۷۵	۲۱۷۵۸	۲۱۷۷۷	۵۵۲	۲۱۷۸۸,۸۷	۱۲۷
۷	۲۲۶۳۵	۲۲۵۱۹	۲۲۴۵۳	۵۸۶	۲۲۵۹۵,۸	۳۲۰
۸	۲۲۵۱۱	۲۲۲۹۲	۲۲۳۵۱	۵۳۴	۲۲۴۰۸,۴	۴۲۰
۹	۲۲۷۰۲	۲۲۵۸۸	۲۲۵۹۱	۵۸۸	۲۲۷۰۲	۸۶
۱۰	۴۱۳۹۵		۴۱۳۲۹	۵۰۱	۴۱۲۶۷,۶۵	۳۵۱
۱۱	۴۲۳۴۴		۴۲۲۱۴	۵۵۹	۴۲۱۶۴,۴۵	۵۱۶
۱۲	۴۲۴۰۱		۴۲۳۰۰	۵۸۴	۴۲۲۸۴,۲	۴۰۳
۱۳	۴۵۶۲۴		۴۵۶۲۱	۵۶۲	۴۵۴۱۰,۴	۴۵۶
۱۴	۴۱۸۸۴		۴۱۷۳۹	۵۳۶	۴۱۷۵۷,۶	۲۷۵
۱۵	۴۲۹۹۵		۴۲۹۰۹	۵۲۵	۴۲۹۳۸,۲	۳۶۷
۱۶	۴۳۵۵۹		۴۳۴۶۴	۵۹۷	۴۳۵۳۲,۷۳	۲۱۹
۱۷	۴۲۹۷۰		۴۲۹۰۳	۴۳۹	۴۲۸۷۲,۴	۴۰۰
۱۸	۴۲۲۱۲		۴۲۱۴۶	۵۹۸	۴۲۱۴۷,۹۵	۳۹۵
۱۹	۴۱۲۰۷		۴۱۰۶۷	۵۴۸	۴۱۱۰۳,۴۷	۱۵۲

جدول۳: مقایسه نتایج الگوریتم *MKPLA* با دیگر الگوریتمهای گزارش شده برای ۵۰۰ کالا و ۵ کوله‌پشتی

شماره مساله	بهترین جواب شناخته شده	ALAYA		MKPLA	
	بهترین	میانگین	C*	میانگین	C*
۰	۱۲۰۱۳۴	۱۱۹۶۵۸	۸۸۵	۱۱۹۹۹۲	۲۰۲
۱	۱۱۷۸۶۴	۱۱۷۴۲۳	۸۵۷	۱۱۷۶۹۰	۳
۲	۱۲۱۱۱۲	۱۲۰۶۲۲	۸۶۰	۱۲۰۹۷۶	۲۱۲
۳	۱۲۰۸۰۴	۱۲۰۲۷۹	۸۱۴	۱۲۰۶۵۰,۷	۱۲۲
۴	۱۲۲۳۱۹	۱۲۱۸۲۹	۸۲۶	۱۲۲۱۳۳	۲

مراجع

[1] Balas, E., and Zemel, E., "An algorithm for large zero-one knapsack problems", Operation Research, vol. 28, 1130–1154, 1980.

[2] Plateau, G., and Elkihel, M., "A hybrid algorithm for the 0-1 knapsack problem", Methods of Operation Research, vol. 49, pp. 277–293, 1985.

[3] Pisinger, D., "An exact algorithm for large multiple knapsack problem", European Journal of Operation Research, vol. 114, pp. 528–541, 1999.
[4] Pisinger, D., "Algorithms for knapsack problems", Ph.D. Thesis, DIKU, University of Copenhagen, Report 95/1 1995.
[5] Lee, J., Shragowitz, E., and Sahni, S., "A hypercube algorithm for the 0/1 knapsack problem", Journal of Parallel and Distributed Computing, vol. 5, pp. 438-456, 1988.
[6] Ibarra, O. H., and Kim, C. E., "Fast approximation algorithms for the knapsack problem", Journal of ACM, vol. 22, pp. 463-468, 1975.
[7] Riadl, G. R. Weight-Codings in a genetic algorithm for the multiconstraint knapsack problem", Proceedings of the 2002 IEEE Congress on Evolutionary Computation, pp. 1564-1569, 2002.
[8] Riadl, G. R.," An improved genetic algorithm for the multiconstrained 0–1 knapsack problem", Proceedings of the 5th IEEE Conference on Evolutionary Computation, Anchorage, Alaska, pp. 207-211, May 1998.
[9] Khuri, S., Back, T., and Heitkotter, J., "The Zero/One multiple knapsack problem and genetic algorithms", Proceedings of the 1994 ACM symposium of Applied Computation, pp. 188-193, 1994.
[10] Fidanova, S., "Evolutionary algorithm for multidimensional knapsack problem", Proceedings of PSNVII, 2002.
[11] Alaya, I., Solnon, Ch., and Ghedira, K., "Ant algorithm for the multidimensional knapsack problem", Dans Proceedings of Iinternational Conference on Bioinspired Methods and their Applications, Slovenia, 2004.
[12] Sutton, R. S., and Barto, A. G., Reinforcement learning: An introduction. MA: MIT Press, Cambridge, 1998.
[13] Thathachar, M. A. L., Sastry, P. S., "Varieties of learning automata: An overview", IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics, 32, 6, 2002.
[14] Pirkul, H., "A Heuristic solution procedure for the multiconstrained zero-one knapsack problem", Naval Research Logistics 34, pp. 161–172, 1987.
[15] Chu, P. C., "A genetic algorithm approach for combinatorial optimization problems", Ph.D. Thesis at the Management School, Imperial College of Science, London, 1997.
[16] Chu, P. C., and Beasley, J. E., "A genetic algorithm for the multidimensional knapsack problem, "Journal of Heuristics 4, pp. 63–86, 1998.
[17] Beasley, J. E., "OR-Library: Distributing test problems by electronic mail", Journal of Operational Research society 41, 1069-1072, 1990.
[18] Leguizamon, G., and Michalewicz, Z., "A new version of Ant System for subset problem", Proceedings of Congress on Evolutionary Computation, pp. 1459-1464, 1999.
[19] Vasquez, M., and Hao, J. K., "A hybrid approach for the 0-1 multidimensional knapsack problem", IJCAI-01, Washington, 2001.