یک نسخه دودویی از الگوریتم پیچک برای حل مسائل کوله پشتی صفر-یک

سعیده محمدزاده[1]، دانشجوی کارشناسی ارشد، عین‌الله پیرا[2*]، دانشیار، علیرضا روحی[3]، دانشیار، سجاد اسفندیاری[4]، استادیار

[1] دانشکده فناوری اطلاعات و مهندسی کامپیوتر- دانشگاه شهید مدنی آذربایجان - تبریز-ایران-mohammadzadehs37@gmail.com

[2و3] دانشکده فناوری اطلاعات و مهندسی کامپیوتر- دانشگاه شهید مدنی آذربایجان - تبریز-ایران-{pira,rouhi}@azaruniv.ac.ir

[4] گروه مهندسی کامپیوتر، دانشگاه ملایر، ملایر، ایران. رایانامه: esfandyari@malayeru.ac.ir

چکیده: در این پژوهش، نسخه‌ای باینری از الگوریتم الهام‌گرفته از رشد پیچک (IVY) با هدف حل مسأله کوله‌پشتی صفر و یک طراحی و ارزیابی شده است. مسأله کوله‌پشتی یکی از مسائل کلاسیک بهینه‌سازی ترکیبیاتی است که در حوزه‌های مختلفی از جمله تخصیص منابع، زمان‌بندی و برنامه‌ریزی پروژه کاربرد دارد. حل این مسأله با توجه به پیچیدگی نمایی فضای جستجو، نیازمند استفاده از الگوریتم‌های مؤثر و کارآمد است. نسخه BiIVY با ترکیب مکانیزم رشد جهت‌دار، تابع جریمه و الگوریتم ترمیم ویژه، قابلیت یافتن پاسخ‌های بهینه با تعداد تکرار کمتر نسبت به الگوریتم‌های مرسوم را فراهم می‌کند. پارامترهای الگوریتم با توجه به منابع پیشین و آزمون‌های آزمایشی تعیین شده‌اند. کارایی BiIVY روی 25 مجموعه داده استاندارد L1–L25 ارزیابی شده و با الگوریتم‌های دودویی گرده‌افشانی گل (BFPA) و الگوریتم سینوسی-کسینوسی دودویی (BSCA) مقایسه گردیده است. نتایج آزمون میانگین رتبه فریدمن نشان می‌دهد که BiIVY در اکثر موارد از لحاظ کل سود و تعداد تکرار عملکرد بهتری دارد و توانسته پاسخ‌های بهینه یا نزدیک به بهینه را با تعداد تکرار کمتر ارائه دهد، که اثربخشی و توانایی الگوریتم پیشنهادی را در حل مسائل ترکیبیاتی پیچیده نشان می‌دهد.

واژه‌های کلیدی: الگوریتم پیچک، کوله‌پشتی صفر-یک، بهینه‌سازی، دودویی، الگوریتم‌های فراابتکاری

* نویسنده مسئول، pira@azaruniv.ac.ir

A binary version of the Ivy algorithm for solving zero-one knapsack problems

Saeedeh Mohammadzadeh[1], M.Sc. Student, Einollah Pira[2*], Associate Professor, Alireza Rouhi[3], Associate Professor, Sajad Esfandyari[4], Assistant Professor

[1]Faculty of Information Technology and Computer Engineering, Azarbaijan Shahid Madani University, Tabriz, Iran, mohammadzadehs37@gmail.com

[2,3] Faculty of Information Technology and Computer Engineering, Azarbaijan Shahid Madani University, Tabriz, Iran, {pira, rouhi}@azaruniv.ac.ir

[4]Department of Computer Engineering, Malayer University, Malayer, Iran. Email: esfandyari@malayeru.ac.ir

Abstract: In this study, a binary version of the Ivy-inspired algorithm (IVY) was designed and evaluated to solve the 0–1 Knapsack Problem. The Knapsack Problem is a classical combinatorial optimization problem with applications in resource allocation, scheduling, and project planning. Due to the exponential complexity of its search space, efficient and effective algorithms are required to solve it. The BiIVY version combines a directed growth mechanism, a penalty function, and a specialized repair algorithm, enabling it to find higher-quality solutions with fewer iterations compared to conventional algorithms. Algorithm parameters were determined based on previous studies and experimental trials. The performance of BiIVY was evaluated on 25 standard datasets (L1–L25) and compared with Binary Flower Pollination Algorithm (BFPA) and Binary Sine-Cosine Algorithm (BSCA). Friedman mean-rank test results indicate that BiIVY generally outperforms the other algorithms in terms of total profit and iteration count, providing optimal or near-optimal solutions with fewer iterations, which demonstrates the effectiveness and capability of the proposed algorithm in solving complex combinatorial problems.

Keywords: *Ivy algorithm, knapsack 0-1, optimization, binary, metaheuristic algorithms.*

*** corresponding author, pira@azaruniv.ac.ir**

1. مقدمه

بهینه‌سازی، فرایندی است که در آن بهترین جواب (با توجه به مجموعه‌ای از معیارها) از میان مجموعه‌ای از جواب‌های ممکن برای یک مسأله‌ی خاص انتخاب می‌شود [1]. بنابراین، بهینه سازی کار ساده‌ای نیست و از نظر محاسباتی غالباً نیازمند کاوش در فضای جستجوی گسترده‌ای است تا بتوان بهترین راه‌حل را یافت که تمامی محدودیت‌ها و مشخصات مورد نظر را برآورده کند [2]. طراحی یک الگوریتم اختصاصی برای حل هر مسأله بهینه‌سازی نیز چالشی جدی به شمار می‌آید. به همین دلیل، دانشمندان و مهندسان از الگوریتم‌های فرابتکاری استفاده می‌کنند که قادرند در مدت زمان معقولی، راه‌حلی بهینه ارائه دهند. این راه‌حل ممکن است بهترین گزینه در کل فضای جستجو نباشد، اما معمولاً به اندازه‌ای مناسب است که بتواند نیازها را بدون صرف بیش از حد منابع محاسباتی یا زمان، برآورده سازد.

با پیشرفت علوم و فناوری، مسائل بهینه سازی متنوع‌تری پدیدار شده‌اند که اغلب دارای ویژگی هایی همچون غیرخطی بودن، چندوجهی بودن و ناپیوستگی در تابع هدف یا قیود هستند. این ویژگی‌ها موجب پیچیدگی بیشتر در فرایند حل این مسائل شده و ضرورت استفاده از روش‌های هوشمند و کارآمد را دوچندان می‌سازد [3].

در سال‌های اخیر، الگوریتم‌های فرابتکاری[1] به‌عنوان رویکردهایی کارآمد جهت حل مسائل پیچیده، غیرخطی و با ابعاد بالا، مورد توجه بسیاری از پژوهشگران قرار گرفته‌اند. با توجه به پیچیدگی ذاتی بسیاری از مسائل دنیای واقعی و پویایی محیط‌های مسأله، نیاز به روش‌هایی که توانایی مقابله با این پیچیدگی را داشته باشند، بیش از پیش احساس می‌شود. در نتیجه، حوزه طراحی و توسعه الگوریتم‌های فرابتکاری به‌طور مداوم در حال تحول و گسترش بوده و به‌طور مستمر شاهد معرفی الگوریتم های جدید در این زمینه هستیم. بخش قابل توجهی از این مسائل، در دسته مسائل NP-hard قرار می‌گیرند؛ مسائلی که حل دقیق آن‌ها در زمان چندجمله‌ای امکان‌پذیر نیست. به بیان ساده، این مسائل از فضای جستجوی بسیار بزرگی برخوردارند و به همین دلیل، روش‌های کلاسیک بهینه‌سازی نظیر برنامه‌ریزی خطی، الگوریتم گرادیان کاهشی، روش های شبه‌نیوتن و سایر رویکردهای قطعی، در دستیابی به پاسخ‌های بهینه برای این مسائل ناتوانا ند. در مقابل، الگوریتم های فرابتکاری و بهینه‌سازی تصادفی با تکیه بر جستجوی هوشمندانه در فضای مسأله، تلاش می‌کنند تا به راه‌حل‌هایی نزدیک به مقدار بهینه دست یابند [4]. اگرچه این الگوریتم‌ها تضمینی برای یافتن پاسخ دقیق ارائه نمی‌دهند، اما در عمل، توانسته‌اند در حل بسیاری از مسائل پیچیده عملکرد قابل قبولی از خود نشان دهند و به عنوان جایگزینی مناسب برای روش‌های سنتی مطرح شوند.

می‌توانیم الگوریتم‌های فرابتکاری را به چهار دسته تقسیم کنیم: الگوریتم‌های هوش جمعی (SI) [5]، الگوریتم‌های تکاملی (EA) [6]، الگوریتم‌های مبتنی بر فیزیک (PhAs) [7] و الگوریتم‌های مبتنی براندسان [8]. الگوریتم‌های فرابتکاری عموماً در ساختار خود از دو مؤلفه‌ی اصلی بهره می‌برند: اکتشاف[2] برای جستجو در نواحی جدید فضای مسئله و بهره‌برداری[3] برای بهبود پاسخ‌های به‌دست‌آمده. برقراری تعادل میان این دو فرآیند، نقشی کلیدی در عملکرد موفق الگوریتم ایفا می‌کند. در این مطالعه، به‌منظور تضمین تنوع در فضای جستجو و همزمان تقویت بهره‌برداری، از عملگرهای ترکیب[4] و جهش[5] استفاده

[1] Meta-heuristic Algorithms
[2] Exploration
[3] Exploitation
[4] Crossover
[5] Mutation

گردید. این عملگرها پیش‌تر در تحقیقات متعددی برای تولید راه‌حل‌های کاندید و ارتقای کیفیت جستجو به‌کار رفته‌اند [9]. الگوریتم پیچک IVYA نوعی الگوریتم الهام‌گرفته از زیست است که برگرفته از الگوهای رشد گیاهان پیچک می‌باشد [10]. رشد هماهنگ و منظم جمعیت و گسترش و تکامل گیاهان پیچک را شبیه‌سازی می‌کند. الگوریتم IVYA یک روش بهینه‌سازی مبتنی بر جمعیت است که از طریق یک فرآیند تکراری عمل می‌کند و از توانایی جستجوی جمعی اعضای جمعیت بهره می‌برد. جمعیت در الگوریتم IVYA از افرادی تشکیل شده است که شبیه به جامعه گیاهان پیچک عمل می‌کنند. موقعیت هر عضو در فضای جستجو، مقادیر بالقوه‌ای برای متغیرهای تصمیم مسأله ارائه می‌دهد. به‌طور ریاضی، هر عضو جمعیت (که یک گیاه پیچک است) به‌عنوان یک بردار نمایش داده می‌شود و به‌عنوان یک راه‌حل بالقوه برای مسأله عمل می‌کند. نرخ رشد گیاهان پیچک با استفاده از یک معادله دیفرانسیل و یک فرآیند داده‌محور مدل‌سازی شده و جهت رشد با توجه به نزدیک‌ترین و قوی‌ترین همسایه تعیین می‌شود. ویژگی‌های منحصر به فرد الگوریتم IVYA مانند حفظ تنوع جمعیت، سادگی و انعطاف‌پذیری آن، امکان تغییر و گسترش آسان را فراهم می‌کند [10].

روش‌های گوناگونی برای حل مسأله کوله‌پشتی به کار گرفته می شوند که به دو دسته الگوریتم‌های دقیق و فراابتکاری تقسیم می‌شوند. الگوریتم‌های دقیق، مانند برنامه‌نویسی پویا[6] و شاخه و کران[7] (B&B)، قادر به ارائه راه‌حل‌های دقیق هستند و در بدترین حالت، با افزایش نمایی زمان محاسبه، اندازه مسأله نیز افزایش می‌یابد. از سوی دیگر، الگوریتم‌های فراابتکاری می‌توانند در زمان‌های معقول، راه‌حل‌های تقریبی ارائه دهند [11]. مسائل تصمیم‌گیری کوله‌پشتی 0-1 در بسیاری از حوزه‌ها کاربرد دارند، از جمله تصمیم‌گیری‌های سرمایه‌گذاری، بودجه‌بندی اصلی، بهینه‌سازی نگهداری املاک، مشکلات بارگیری محموله، تخصیص منابع، حافظه کامپیوتر و مسأله مسکن [12].

با توجه به محدودیت‌های الگوریتم‌های دقیق در مواجهه با مسائل با ابعاد بزرگ، استفاده از الگوریتم‌های فراابتکاری رویکردی مؤثر برای دستیابی به پاسخ‌های مناسب در زمان معقول است [13]. پژوهش حاضر بر آن است تا با بهره‌گیری از الگوریتم پیچک، نسخه‌ای باینری از آن را طراحی و پیاده‌سازی نماید، که قادر به حل مؤثر مسأله کوله‌پشتی 0-1 باشد. اهداف تحقیق شامل طراحی و توسعه نسخه باینری الگوریتم پیچک، بررسی مفاهیم نظری الگوریتم و تحلیل عملکرد آن در مسائل پیوسته، و توسعه روش باینری‌سازی الگوریتم با استفاده از توابع نگاشت مناسب برای انطباق با فضای گسسته است [10]. الگوریتم پیچک به دلیل توانایی بالای آن در اکتشاف و بهره‌برداری در مسائل بهینه‌سازی ترکیبیاتی و انعطاف‌پذیری برای تبدیل به نسخه باینری، به‌عنوان الگوریتم مناسب برای حل مسأله کوله‌پشتی 0–1 انتخاب شد. نسخه BiIVY با بهره‌گیری از تابع جریمه و مکانیزم رشد جهت‌دار، قادر است پاسخ‌های با کیفیت بالاتر و تعداد تکرار کمتر نسبت به الگوریتم‌های فراابتکاری موجود ارائه دهد. نوآوری اصلی این پژوهش در طراحی نسخه باینری الگوریتم پیچک و بهبود مکانیزم رشد جهت‌دار و تابع جریمه برای مسائل گسسته است، که این نسخه توانایی ارائه پاسخ‌های بهینه‌تر با تعداد تکرار کمتر را دارا می‌باشد. از آنجا که بسیاری از روش‌های فراابتکاری برای حل مسأله کوله‌پشتی با ضعف‌هایی مانند همگرایی زودرس و عدم تعادل مناسب میان اکتشاف و بهره‌برداری مواجه‌اند، توسعه الگوریتم‌های کارآمدتر ضرورت دارد. الگوریتم پیچک با وجود عملکرد مناسب در نسخه پیوسته، فاقد نسخه باینری برای مسائل گسسته بوده است.

[6] Dynamic Programming

[7] Branch-And-Bound

نوآوری این پژوهش در ارائه نسخه باینری الگوریتم پیچک و بهبود مکانیزم رشد جهت‌دار و تابع جریمه برای ارتقای کیفیت پاسخ‌ها و سرعت همگرایی است.

2. ادبیات پژوهش

در این بخش، ابتدا مسایل بهینه‌سازی و کوله‌پشتی صفر-یک مورد بررسی واقع می‌شوند، سپس مرور کوتاهی بر الگوریتم پیچک خواهیم داشت. انتهای این بخش نیز به پیشینه پژوهش تعلق دارد.

1,2. مسائل بهینه‌سازی

مسائل بهینه‌سازی در علوم رایانه، مهندسی و ریاضیات به مسائلی اطلاق می‌شود که در آن‌ها هدف یافتن بهترین جواب از بین مجموعه‌ای از جواب‌های ممکن است. این بهترین جواب بر اساس بیشینه یا کمینه کردن یک تابع هدف[8] تعیین می شود. اگر متغیرهای تصمیم گسسته یا دودویی باشند، مسأله بهینه‌سازی به شکل گسسته[9] خواهد بود [۱٤].

ساختار کلی مسأله بهینه‌سازی به شکل زیر است:

$f(x)$	تابع بهینه‌سازی
$g_i(x) \le 0$	محدود به : $i = 1,2,\dots,m$
$h_i(x) = 0$	$j = 1,2,\dots,p$
$x \in D$	

در این مقاله، هدف حل یک مسأله بهینه‌سازی گسسته تک‌هدفه با محدودیت است، یعنی انتخاب زیرمجموعه‌ای از آیتم‌ها از میان مجموعه موجود، به‌گونه‌ای که مجموع سود بیشینه و مجموع وزن از ظرفیت مجاز کمتر یا مساوی باشد. به این نوع مسائل، بهینه‌سازی ترکیبی[10] گفته می‌شود.

2,2. مساله کوله‌پشتی صفر-یک

این مساله یکی از مسائل کلاسیک و بنیادی در زمینه بهینه‌سازی ترکیبیاتی محسوب می‌شود که کاربردهای فراوانی در علوم رایانه، اقتصاد، مهندسی صنایع و سیستم‌های تصمیم‌یار دارد. این مسأله در رده مسائل NP-Complete قرار می‌گیرد و یافتن راه‌حل دقیق آن در مقیاس‌های بزرگ به‌طور معمول نیازمند زمان محاسباتی بالا و منابع زیاد است. همین امر باعث شده تا این مسأله به عنوان یک معیار استاندارد برای ارزیابی و آزمون الگوریتم‌های فراابتکاری مختلف مورد استفاده قرار گیرد [15].

در این مسأله، فرض می‌شود مجموعه‌ای از اقلام وجود دارد که هرکدام دارای دو ویژگی هستند: وزن و سود (یا ارزش). هدف، انتخاب ترکیبی از این اقلام است به‌گونه‌ای که جمع وزن‌های انتخاب‌شده از یک حد مشخص (ظرفیت کوله‌پشتی) بیشتر نباشد و در عین حال مجموع ارزش‌های آن‌ها حداکثر شود. در نسخه صفر-یک این مسأله، هر قلم کالا یا کامل انتخاب می‌شود (مقدار 1) یا اصلاً انتخاب نمی‌شود (مقدار 0) و نمی‌توان بخشی از یک آیتم را انتخاب کرد.

این مساله بصورت ریاضی بصورت زیر تعریف می‌شود: این مساله مجموعه ای از n آیتم است و هر آیتم دارای وزن w_iو سود p_i و حداکثر ظرفیت وزن W است که در آن x_iیک متغیر باینری برای انتخاب این مورد $(x_i = 1)$ یا عدم انتخاب $(x_i = 0)$ است. هدف، به حداکثر رساندن سود اقلام موجود در کوله پشتی است به طوری که وزن کلی کمتر یا برابر با ظرفیت کوله پشتی باشد. در حالت کلی، خواهیم داشت:

- $Maximize \quad \sum_{i=1}^{n} p_i x_i$
- $Where: \sum_{i=1}^{n} w_i x_i \le W \;\; x_i \in \{0,1\}$

3,2. الگوریتم پیچک

[8] Objective Function
[9] Discrete Optimization
[10] Combinatorial Optimization

الگوریتم پیچک (IVYA) یک روش فراابتکاری مبتنی بر جمعیت است که از الگوهای رشد گیاه پیچک الهام گرفته شده است. در این الگوریتم، هر عضو جمعیت به‌صورت یک بردار در فضای جستجو نمایش داده می‌شود و نمایانگر یک راه‌حل بالقوه است. این الگوریتم مراحل مختلف زندگی گیاه پیچک مانند رشد، بالا رفتن و گسترش را در قالب یک دسته از گیاهان پیچک (جمعیت) شبیه‌سازی می‌کند [10]. ساختار کلی IVYA به صورت زیر است:

- ایجاد جمعیت اولیه‌ای از پیچک‌ها به عنوان راه‌حل‌های اولیه مسأله؛
- مراحل جستجو و رشد جمعیت در الگوریتم پیشنهادی IVYA شامل:
  - رشد هماهنگ و مرتب جمعیت؛
  - رشد به سمت منبع نور خورشید برای بهینه‌سازی موقعیت؛
  - گسترش و تکامل گیاهان پیچک؛
- انتخاب بازماندگان برای ادامه نسل بعدی.

در ابتدا، جمعیت اولیه به‌صورت تصادفی ایجاد می‌شود. موقعیت اعضا با رابطه زیر تعیین می‌گردد:

$$\boldsymbol{I_i = I_{min} + rand(1,D) \odot (I_{max} - I_{min})} \quad (1)$$

که در آن، I_{min} و I_{max}به‌ترتیب حد پایین و بالای فضای جستجو و $\odot$ عملگر ضرب هادامارد است. $\boldsymbol{rand(1,D)}$ یک بردار تصادفی با طول $\boldsymbol{D}$ است که هر مؤلفه آن عددی واقعی و یکنواخت از بازه [0,1] است.

فاز 1: رشد هماهنگ و منظمی

این مرحله مشابه رشد ابتدایی پیچک روی زمین است. نرخ رشد هر عضو با یک معادله دیفرانسیل توصیف می‌شود:

$$\boldsymbol{\frac{dGv(t)}{dt} = \psi . Gv(t) . \varphi(Gv(t))} \quad (2)$$

در این جا $Gv(t)$ نرخ رشد پیچک در لحظه t، ψ ضریب رشد که سرعت رشد را کنترل می‌کند، $\varphi(Gv(t))$ تابع اصلاحی که میزان انحراف از رشد طبیعی را کنترل می‌کند.

و به‌صورت گسسته با رابطه زیر به‌روزرسانی می‌شود:

$$\boldsymbol{\Delta Gv_i(t)\ (t+1) = rand^2 \odot (N(1,D) \odot \Delta Gv_i(t)\ (t))} \quad (3)$$

که در آن بردارهای $\Delta Gv_i(t)$و $\Delta Gv_i(t+1)$ نرخ رشد را در یک سیستم زمان گسسته (در لحظه های زمانی t و t + 1) نشان می دهند، $rand$ یک عدد واقعی و یکنواخت از بازه [0,1] است. $\boldsymbol{N(1,D)}$توزیع نرمال با میانگین 1 و واریانس D و $\odot$ ضرب مؤلفه‌ای بین بردارها است.

فاز 2: رشد جهت‌دار به سمت نور

هر پیچک از نزدیک‌ترین همسایه قوی‌تر پیروی می‌کند. موقعیت جدید بر اساس رابطه زیر محاسبه می‌شود:

$$\boldsymbol{I_i^{new} = I_i + |N(1,D)| \odot (I_{ii} - I_i) + N(1,D) \odot \Delta Gv_i} \quad (4)$$

که در آن $\boldsymbol{I_i^{new}}$ موقعیت جدید عضو iدر فضای جستجو را نشان می‌دهد. $\boldsymbol{I_i}$ موقعیت فعلی عضو i است و $\boldsymbol{I_{ii}}$موقعیت نزدیک‌ترین همسایه قوی‌تر آن عضو را مشخص می‌کند .بخش $\boldsymbol{|N(1,D)| \odot (I_{ii} - I_i)}$ مسئول جهت‌گیری رشد پیچک به سمت نور (همسایه قوی‌تر) است و $\boldsymbol{N(1,D) \odot \Delta Gv_i}$بخش تصادفی و متغیر برای حفظ تنوع و اکتشاف در فضای جستجو ایجاد می‌کند. توضیح این نمادها شفافیت مدل و قابلیت تکرار الگوریتم را تضمین می‌کند.

فاز 3: گسترش و تکامل

پس از یافتن بهترین عضو، سایر اعضا در اطراف آن جستجو می‌کنند:

$$\boldsymbol{I_i^{new} = I_{Best} \odot (rand(1,D) + N(1,D) \odot \Delta Gv_i)} \quad (5)$$

در این رابطه، موقعیت جدید عضو i بر اساس تأثیر بهترین عضو جمعیت و تغییرات رشد پیچک محاسبه می‌شود. ابتدا $\boldsymbol{I_{Best}}$که نشان‌دهنده موقعیت بهترین عضو در نسل فعلی است، به‌عنوان الگوی اصلی رشد در نظر گرفته می‌شود. حاصل‌ضرب سطری $\boldsymbol{N(1,D) \odot \Delta Gv_i}$نشان‌دهنده میزان اثرگذاری تغییرات

رشد بر هر بعد است. در نهایت، جمع این دو مؤلفه تصادفی در I_{Best}ضرب می‌شود تا موقعیت جدید I_i^{new}شکل گیرد. این فرمول باعث می‌شود حرکت عضو i هم جهت‌دار باشد (به سمت بهترین عضو) و هم شامل مقدار مناسبی از تنوع، تا الگوریتم بتواند اکتشاف و بهره‌برداری را همزمان حفظ کند.

نرخ رشد نیز مطابق رابطه زیر به‌روزرسانی می‌شود:

$$\Delta G v_i^{new} = I_i^{new} \oslash (I_{max} - I_{min}) \quad (6)$$

در الگوریتم IVYA، فرآیند انتخاب بازماندگان برای حفظ بهترین اعضای جمعیت انجام می‌شود. هر عضو I_i بر اساس مقدار تابع هدف $f(I_i)$ با بهترین عضو جمعیت $f(I_{Best})$مقایسه می‌گردد. یک ضریب دینامیک تصادفی به صورت زیر تعریف می‌شود:

$$\beta = \frac{2 + rand}{2} \quad (7)$$

اگر $f(I_{Best}).\beta > f(I_i)$، عضو وارد مرحله گسترش عرضی شده و در غیر این صورت **به** مرحله رشد صعودی هدایت می‌شود. سپس جمعیت جدید با قبلی ادغام و مرتب شده و در نهایت بهترین N عضو به نسل بعدی منتقل می‌گردند:

$$\vec{I} = \{I_1^{Merged}, I_2^{Merged}, \dots, I_N^{Merged}\} \quad (8)$$

هر مؤلفه I_1^{Merged}نمایانگر موقعیت اصلاح‌شده‌ی یک پیچک در فضای جستجو است و ترکیبی از موقعیت‌های قبلی و تغییرات ناشی از مکانیزم‌های الگوریتم محسوب می‌شود. این بردار در نهایت برای ارزیابی کیفیت راه‌حل‌ها و انتخاب بهترین عضو جمعیت استفاده می‌شود.

4,2. پیشینه پژوهش

در سال‌های اخیر، بسیاری از الگوریتم‌های فراابتکاری (به‌ویژه الگوریتم‌های مبتنی بر ازدحام) برای حل مسائل بهینه‌سازی ترکیبی مورد استفاده قرار گرفته‌اند. مسائل بهینه‌سازی ترکیبی نقش بسیار مهمی در حوزه‌هایی مانند مهندسی نرم‌افزار، یادگیری ماشین و هوش مصنوعی ایفا می‌کنند. تلاش برای توسعه الگوریتم‌های فراابتکاری برای این مسائل همچنان ادامه دارد و پایان‌ناپذیر است.

الگوریتم ژنتیک GA نخستین بار توسط هالند توسعه یافت و شبیه سازی انتخاب طبیعی موجودات زنده را انجام می‌دهد. این الگوریتم از دو عملگر اصلی شامل ترکیب و جهش برای حل مسائل بهینه‌سازی استفاده می‌کند. با این حال، الگوریتم ژنتیک با مشکل همگرایی زودهنگام مواجه است، که باعث می‌شود جمعیت به راه‌حل‌های زیر بهینه محدود شود. همچنین، افزایش اندازه جمعیت برای حفظ تنوع می‌تواند زمان محاسبات را افزایش دهد گوپتا [16] یک نسخه سریع‌تر از الگوریتم ژنتیک پیشنهاد داد که یک تابع جدید اضافه می‌کند. این تابع پس از هر فرآیند جهش، جمعیت را بررسی کرده و تمامی راه‌حل های غیرممکن را با نمونه‌های تصادفی جدید جایگزین می‌کند.

الگوریتم بهینه‌سازی ازدحام ذرات[11] PSO [17] برای حل مسائل بهینه‌سازی پیوسته طراحی شده است. نسخه‌ای دودویی از این الگوریتم به نام BPSO [18] با استفاده از روش سیگموئید، فضای جستجو را به صورت گسسته در می‌آورد. با این حال، در صورت افزایش مقادیر سرعت، تغییر در احتمال‌ها کاهش می‌یابد که باعث محدود شدن جستجو می شود. برای رفع این مشکل، نسخه اصلاح‌شده‌ای به نام MBPSO [19] ارائه شده است که از یک تابع احتمال جدید استفاده می‌کند تا میزان جستجوی فضای جستجو را افزایش دهد. نتایج تجربی نشان می‌دهد که MBPSO عملکرد بهتری نسبت به BPSO دارد. نگوین و همکاران [20] نسخه‌ای دودویی از PSO را معرفی کردند که شامل دو عملگر جدید برای مدیریت محدودیت‌ها بود: عامل جریمه[12] و تابع

[11] Particle Swarm Optimization

[12] Penalty Factor

تعمیر[13]. این عملگرها بر اساس استراتژی حریصانه و انتخاب تصادفی عناصر مسئله طراحی شده‌اند و کارایی الگوریتم را در حل مسائل بهبود می‌بخشند.

الگوریتم شب‌تاب[14] FA [21] توسط یانگ برای حل مسائل بهینه سازی چندوجهی طراحی شده است و در مقایسه با PSO نتایج بهتری ارائه می‌دهد .الگوریتم ترکیبی شب‌تاب الهام‌گرفته از محاسبات کوانتومی PSO [22] با ترکیب FA و PSO مزایای هر دو الگوریتم را به کار می‌گیرد. که از مفهوم محاسبات کوانتومی برای جلوگیری از همگرایی زودهنگام در FA استفاده می‌کند و همچنین یک عملگر تعمیر برای رفع نواقص احتمالی دارد. فِنگ و وانگ [23] نسخه بهبودیافته‌ای از FA به نام IFA ارائه کردند که از یک استراتژی حریصانه برای حل مسائل کوله‌پشتی تصادفی با شرایط متغیر در زمان استفاده می‌کند. IFA عملکرد بهتری نسبت به الگوریتم ژنتیک GA نشان می‌دهد، اما موفقیت IFA تحت تاثیر مقادیر پارامترهایش است و به همین دلیل، ارزیابی‌های زیادی برای تعیین مقادیر بهینه پارامترها انجام می‌شود.

الگوریتم میمون[15] MA [24] با شبیه‌سازی رفتار میمون‌ها در کوهنوردی برای حل مسائل بهینه‌سازی پیوسته جهانی به کار می‌رود. این الگوریتم به دلیل پیاده سازی آسان، پارامترهای کم و نرخ همگرایی سریع در مواجهه با مسائل پیچیده (غیرخطی بودن، ابعاد بالا و غیرقابل مشتق بودن) از کارایی بالایی برخوردار است. نسخه‌ای دودویی از این الگوریتم که فرایند همکاری و استراتژی حریصانه را ترکیب می‌کند CGMA [25]، با ادغام دو فرایند چرخش و همکاری، به ترتیب، نرخ جستجو و همگرایی جمعیت را افزایش می‌دهد. این الگوریتم با بازتنظیم جمعیت، از افتادن در بهینه های محلی جلوگیری می‌کند. علاوه بر این، از الگوریتم حریصانه برای تبدیل راه‌حل‌های غیرقابل قبول به راه‌حل‌های قابل قبول استفاده می‌شود که سرعت الگوریتم را در دستیابی به بهترین راه‌حل‌ها بالا می‌برد. مقایسه‌ها نشان می‌دهد که CGMA در مقایسه با دیگر الگوریتم‌های فراابتکاری نتایج موثری ارائه می‌دهد.

الگوریتم بهینه‌سازی پروانه سلطنتی[16] MBO [26] یک الگوریتم فراابتکاری الهام‌گرفته از طبیعت است که مهاجرت پروانه‌های سلطنتی را شبیه‌سازی می‌کند و برای مسائل پیوسته به کار می‌رود. الگوریتم MBO با استراتژی حریصانه و ترکیب خودتطبیقیGCMBO [27] به منظور رفع کاستی‌های MBO با اضافه کردن ترکیب و استراتژی حریصانه طراحی شد. نتایج اولیه نشان‌دهنده عملکرد بهتر GCMBO نسبت به MBO است .فِنگ و همکاران [28] با ادغام MBO و فرایند جهش گوسی برای حل مشکل همگرایی و استفاده از نگاشت‌های آشوبی برای مقابله با مسائل بهینه‌سازی ترکیبی COPs توانستند مسائل کوله‌پشتی 0–1 را بهبود دهند. در پژوهش دیگری، فنگ و همکاران [29] یک نسخه دودویی جدید از MBO به نام BMBO را برای حل مسئله کوله‌پشتی 0–1 معرفی کردند که از روش حریصانه و یک عملگر تعمیر بهره می‌برد.

الگوریتم گرده‌افشانی گل‌ها[17] [11] برای حل مسئله کوله‌پشتی 0-1 به نسخه‌ای دودویی BFPA تبدیل شده است. در این نسخه، از یک تابع جریمه برای حذف راه‌حل‌های غیرمجاز که خارج از ناحیه راه‌حل‌های مجاز قرار دارند، استفاده می‌شود. این تابع به راه‌حل‌های غیرمجاز یک مقدار منفی اختصاص می‌دهد تا از آنها صرف نظر شود. علاوه بر این، برای مقابله با مسئله کوله‌پشتی چندبعدی، لای و همکاران [30] یک الگوریتم

13 Repair Function
14 Firefly Algorithm
15 Monkey Algorithm
16 monarch butterfly optimization
17 Flower Pollination Algorithm

بهینه‌سازی ازدحام ذرات کوانتومی[18] QPSO جدید ارائه کردند که در آن از استراتژی حفظ تنوع مبتنی بر فاصله و یک روش بهینه‌سازی محلی مبتنی بر تغییر متغیرهای همسایگی استفاده شده است.

بدون شک، نویسندگان متعددی الگوریتم‌هایی را پیشنهاد کرده‌اند که در حل مسائل کوله‌پشتی 0-1 با ابعاد کم و زیاد عملکرد چشمگیری دارند. به عنوان مثال، در [12] نسخه دودویی یک الگوریتم بهینه سازی جدید به نام الگوریتم شکارچیان دریایی[19] MPA ارائه شده است. این الگوریتم با بررسی مجموعه‌ای از توابع انتقال برای تبدیل فضای جستجوی پیوسته به دودویی عمل می‌کند. در [31] یک نسخه بهبود یافته از الگوریتم بهینه‌سازی کلونی مورچه‌ها[20] ACO برای حل مسائل کوله‌پشتی 0-1 در یک محیط فازی ارائه شده است. در این روش پیشنهادی، برای هر یک از N شی N گروه کاندید ایجاد می‌شود و هر مورچه از هر گروه یک مقدار کاندید را انتخاب می‌کند.

یک الگوریتم جدید بهینه‌سازی فراابتکاری مبتنی بر الهام طبیعی به نام Aquila Optimizer (AO) معرفی می‌شود. الگوریتم AO پیشنهادی رفتار Aquila را در طول شکار شبیه سازی می‌کند و اقدامات هر مرحله از شکار را نشان می‌دهد [32]. یک الگوریتم جدید برای مسائل بهینه سازی باینری BAO پیشنهاد شده است [13]. الگوریتم کسینوس سینوسی SCA یک الگوریتم فراابتکاری اخیر است. این الگوریتم مانند سایر الگوریتم های فراابتکاری با مجموعه ای از راه حل های تصادفی شروع می شود. به طور مکرر، این راه حل ها به سمت بهترین راه حل در جمعیت با استفاده از عملگرهای سینوس و کسینوس هدایت می شوند. از الگوریتم سینوس کسینوس SCA برای حل مسائل کوله پشتی 0-1 اتخاذ شده است. به دلیل ماهیت باینری مشکل کوله پشتی 0-1، SCA با استفاده از یک تابع نگاشت دستکاری می شود [33].

3. الگوریتم ارائه شده

نسخه استاندارد الگوریتم پیچک (IVY) برای حل مسائل پیوسته طراحی شده است در حالیکه، مسأله کوله‌پشتی یک مسأله گسسته محسوب می‌شود، زیرا تصمیم‌گیری در این مسأله بر اساس انتخاب یا عدم انتخاب یک آیتم برای قرارگیری در کوله‌پشتی انجام می‌گیرد. نسخه پیشنهادی BiIVY شامل چند تفاوت اساسی با نسخه اولیه IVYA است تا بتواند در فضای باینری مسأله کوله‌پشتی عملکرد مؤثرتری داشته باشد. نخست، یک تابع انتقال برای تبدیل موقعیت‌های پیوسته IVYA به مقادیر باینری تعریف شد. دوم، سازوکار رشد جهت‌دار به‌گونه‌ای اصلاح گردید که انتخاب همسایه قوی‌تر در فضای گسسته معنا پیدا کند. سوم، یک تابع جریمه پویا برای جلوگیری از تولید راه‌حل‌های نامعتبر (وزن بیش از ظرفیت) به الگوریتم افزوده شد. این تغییرات موجب شده نسخه BiIVY در مسائل گسسته دقت بالاتر و همگرایی پایدارتر نسبت به نسخه اصلی داشته باشد.

برای سازگار کردن IVY با مسأله کوله‌پشتی، از تابع سیگموئید استفاده می‌شود. این تابع مقادیر حقیقی تولید شده توسط نسخه استاندارد IVY را به مقادیر صفر و یک تبدیل می‌کند؛ به‌گونه‌ای که مقدار یک نشان‌دهنده‌ی انتخاب آیتم برای قرار گرفتن در کوله‌پشتی و مقدار صفر نشان‌دهنده‌ی عدم انتخاب آن آیتم است.

1,3. تابع جریمه ظرفیت کوله پشتی

تابع تناسب برای ارزیابی کیفیت هر راه‌حل با محاسبه مجموع سود اقلام انتخاب‌شده در کوله‌پشتی به‌کار می‌رود، اما ممکن است

18 Quantum Particle Swarm Optimization Algorithm
19 Marine Predators Algorithm
20 Ant Colony Optimization

برخی راه‌حل‌ها با وجود سود بالا محدودیت وزن $\sum_{i=1}^{n} w_i x_i <$ Wرا نقض کنند و به‌عنوان راه‌حل‌های نامعتبر شناخته شوند. برای جلوگیری از انتخاب این راه‌حل‌ها، الگوریتم از تابع جریمه استفاده می‌کند؛ به‌گونه‌ای که به راه‌حل‌های نامعتبر مقدار تناسب منفی اختصاص داده شده و عملاً از فرآیند انتخاب حذف می‌شوند، در حالی که راه‌حل‌های معتبر بدون تغییر باقی می‌مانند. این رویکرد موجب می‌شود الگوریتم بر راه‌حل‌های بهینه و معتبر متمرکز شده و کیفیت کلی نتایج بهبود یابد. روال عملکرد تابع جریمه را می‌توان به‌صورت شبه‌کد در شکل 1 نمایش داد.

Algorithm 1 Penalty function

```
1. foreach Ivy I_i in the population
2.     Calculate total_weight of I_i via ∑_{i=1}^{n} w_i x_i
3.     If (total_weight > knaspsack_capacity )
4.         PF = knaspsack_capacity – total_weight
5.     else
6.         PF = total_weight
7.     End if
8. End for
9. Return PF
```

شکل 1- شبه‌کد تابع جریمه

2,3. الگوریتم ترمیم

در فرآیند جستجوی الگوریتم‌های فراابتکاری مانند الگوریتم پیچک دودویی، ممکن است برخی راه‌حل‌ها محدودیت ظرفیت کوله‌پشتی را نقض کنند و به‌عنوان راه‌حل‌های نامعتبر شناخته شوند. برای بازگرداندن این راه‌حل‌ها به فضای معتبر، از الگوریتم ترمیم استفاده می‌شود. این الگوریتم با رویکردی حریصانه، اقلامی را که بیشترین سهم در نقض محدودیت دارند شناسایی و حذف می‌کند. معیار تصمیم‌گیری در این فرآیند نسبت سود به وزن هر آیتم است که به‌صورت $RO = \frac{p_i}{w_i}$محاسبه می‌شود؛ که در آن p_i سود آیتم i ام، w_i وزن آیتم i ام، آیتم‌هایی که مقدار کمتری دارند، از اولویت حذف بالاتری برخوردار هستند، به‌طوری‌که آیتم‌هایی با نسبت سود به وزن کمتر، اولویت بیشتری برای حذف دارند زیرا وزن بالاتری را به‌ازای سود کمتر ایجاد می‌کنند. در این روش ابتدا نسبت سود به وزن برای تمام آیتم‌های انتخاب‌شده محاسبه شده و سپس اقلام بر اساس این مقدار به‌صورت صعودی مرتب می‌شوند. در ادامه، اگر مجموع وزن کل راه‌حل از ظرفیت مجاز فراتر رود، حذف آیتم‌ها از کمترین مقدار RO آغاز می‌شود و این فرآیند تا زمانی ادامه می‌یابد که مجموع وزن به سطح مجاز برسد. در نهایت، راه‌حل ترمیم‌شده به الگوریتم اصلی بازگردانده می‌شود تا فرآیند جستجو ادامه یابد. شبه‌کد الگوریتم ترمیم شکل 2 نمایش داده شده است.

Algorithm 2 Repair function

```
1. Input: infeasible solution I_i
2. for i =1: n
3.     Calculate RO = p_i / w_i
4. end for
5. sort the items according to the order of RO
6. fitness = evaluate solution I_i using PF algorithm
7. while (fitness<0)
8.     Remove items with lowest RO
9.     fitness = evaluate solution I_i
10. end while
```

شکل 2- شبه‌کد الگوریتم ترمیم

3,3. الگوریتم بهبود

در این مرحله، پس از ترمیم اولیه و بازگشت راه‌حل به ناحیه معتبر، الگوریتم تلاش می‌کند تا کیفیت (سود) راه‌حل را با افزودن اقلام بهبود دهد. بدین منظور، اقلام با بیشترین نسبت که هنوز در کوله‌پشتی قرار ندارند، به ترتیب افزوده می‌شوند، مشروط بر اینکه با افزودن آن‌ها، محدودیت ظرفیت نقض نشود. در صورتی که پس از افزودن یک آیتم، ظرفیت کوله‌پشتی از حد مجاز فراتر رود، همان آیتم حذف شده و فرآیند متوقف می‌شود. این فرآیند در الگوریتم 3 نمایش داده شده است. مطابق با این الگوریتم، ابتدا برای هر یک از اقلام موجود در مسأله (i=1 تا n)، نسبت سود به وزن آنها که با نماد RO نمایش داده می‌شود، محاسبه می‌گردد تا نشان دهد هر قلم نسبت به وزنش چه مقدار

سودآوری دارد. سپس اقلام بر اساس این نسبت RO به ترتیب نزولی مرتب می‌شوند تا اقلام با بیشترین کارایی در اولویت انتخاب قرار گیرند. در ادامه، با استفاده از الگوریتم PF، میزان تناسب یا همان fitness راه‌حل اولیه I_i محاسبه می‌شود. تا زمانی که مقدار fitness مثبت باشد، اقلامی که بالاترین مقدار RO را دارند به راه‌حل اضافه می‌شوند و مجدداً fitness راه‌حل به‌روزرسانی می‌گردد. این روند ادامه می‌یابد تا زمانی که افزودن اقلام جدید باعث کاهش یا بی‌اثر شدن fitness شود و در نهایت، یک راه‌حل معتبر I_i (feasible solution) که محدودیت‌های مسأله را رعایت می‌کند، به‌عنوان خروجی بازگردانده می‌شود. این فرآیند باعث می‌شود راه‌حل نامعتبر اولیه به تدریج به یک پاسخ قابل‌قبول و بهینه نزدیک شود. شکل 3 شبه‌کد الگوریتم بهبود را نشان می‌دهد.

Algorithm 3 Improvement function

1. Input: infeasible solution I_i from the repair phase
2. **for** i =1: n
3. Calculate $RO = \frac{p_i}{w_i}$
4. **end for**
5. sort the items according to the order of RO
6. fitness = evaluate solution I_i using PF algorithm
7. **while** (fitness > 0)
8. Add items with highest RO
9. fitness = evaluate solution I_i
10. **end while**
11. **return** feasible solution I_i

شکل 3- شبه‌کد الگوریتم بهبود

4,3. تابع سیگموید

در پیاده‌سازی الگوریتم BiIVY برای مسأله کوله‌پشتی، از تابع انتقال سیگموید برای باینری‌سازی موقعیت‌های پیوسته استفاده شده است. این تابع مقادیر پیوسته‌ی تولیدشده توسط الگوریتم را به احتمال انتخاب (بین 0 تا 1) نگاشت کرده و با مقایسه با عدد تصادفی، تصمیم می‌گیرد که مقدار نهایی صفر یا یک باشد. این روش از معروف‌ترین روش‌های باینری‌سازی در الگوریتم‌های فراابتکاری است. تابع سیگموید یک روش پرکاربرد در باینری‌سازی است، چون مقادیر نزدیک به صفر، احتمال تبدیل‌شدن به 1 را کم دارند. مقادیر مثبت بزرگ‌تر، احتمال زیادی دارند که به 1 تبدیل شوند. این باعث می‌شود رفتار پیوسته الگوریتم حفظ شود، در حالی که در نهایت مقادیر باینری خروجی می‌گیریم. تابع سیگموید که در زیر آورده شده است.

$$X_{S_i} = \frac{1}{1 + e^{-I_i}} \qquad (9)$$

برای هر مقدار به دست آمده از تابع با استفاده از فرمول زیر تبدیل به صفر یا یک خواهد شد.

$$X = \begin{cases} 1 & if\ rand < X_{S_i} \\ 0 & otherwise \end{cases} \qquad (10)$$

راه حل با مقدار حقیقی با استفاده از تابع سیگموئید نگاشت شده و سپس با استفاده از الگوریتم PF ارزیابی می‌شود. اگر PF مقدار منفی را برگرداند، الگوریتم FR را روی این راه حل غیرممکن اعمال می‌کنیم.

5,3. جزئیات الگوریتم ارائه شده

در این بخش، شبه‌کد الگوریتم BiIVY را ارائه می کنیم. همانطوریکه در الگوریتم 4 قابل مشاهده است در ابتدا، الگوریتم نیاز به تعیین چند ورودی کلیدی دارد: ابتدا اندازه‌ی جمعیت اولیه که با نماد N نمایش داده می‌شود مشخص می‌گردد. سپس، حداکثر تعداد تکرارها یا همان *MaxIter* تعیین می‌شود که مشخص می‌کند الگوریتم تا چه زمانی به جستجو در فضای پاسخ ادامه دهد. همچنین ظرفیت کوله‌پشتی، وزن و سود هر یک از اقلام و در نهایت تابع هدف یا همان *fobj* که برای ارزیابی کیفیت پاسخ‌ها به کار می‌رود، به عنوان ورودی دریافت می‌شوند.

Algorithm 4 BiIVY function

```
1. Input: Population size N
2. Maximum iterations MaxIter
3. Knapsack capacity, item weights, item profits
4. Objective function fobj
5. Initialize:
6. Random binary population pos of size N × num_items
7. Compute corresponding GV values
8. Evaluate fitness of initial population
9. Set BestFitness, BestPosition
10. Main Loop (t = 1 to MaxIter):
11.     For each individual i in population:
12.         Evaluate fitness fit(i)
13.         If pos(i) better than BestFitness
14.             Update BestFitness, BestPosition
15.     Generate new population via Ivy propagation:
16.     For each individual i
17.         β₁ = 1+rand()*0.5
18.         If fitness(i) < β₁ × BestFitness then
              // Normal growth
19.           Update position using neighbor difference and
              growth velocity
20.         Else
              // Climbing behavior
21.           Move towards best solution using growth velocity
22.         EndIf
23.         Update GV(i)
24.         Bound newsol to[۰،۱]
25.         Convert newsol to binary via sigmoid transfer
26.         Repair and improve if solution is infeasible
27.         Evaluate fitness
28.         Update best solution if improved
29.     Merge new population with current Sort
        population by fitness
30.     (descending) Keep top N individuals (competitive
        exclusion)
31.     Update convergence curve
32. End Loop
33.Return BestFitness, BestPosition, ConvergenceCurve,
ElapsedTime
```

شکل4- شبه‌کد الگوریتم BiIVY

پس از آماده‌سازی ورودی‌ها، الگوریتم مرحله‌ی مقداردهی اولیه را آغاز می‌کند. در این مرحله، یک جمعیت اولیه به صورت تصادفی تولید می‌شود؛ این جمعیت یک ماتریس دودویی با ابعاد N در تعداد اقلام موجود در مسأله است که هر سطر نمایانگر یک راه‌حل یا فرد در جمعیت می‌باشد. در ادامه، مقادیر *GV* متناظر با هر فرد محاسبه می‌شود. این مقادیر برای فرآیند انتشار پیچک *(Ivy Propagation)* در مراحل بعدی مورد استفاده قرار خواهند گرفت. سپس، تناسب یا همان *Fitness* اولیه‌ی تمام افراد موجود در جمعیت محاسبه می‌شود و بهترین مقدار تناسب *(BestFitness)* و بهترین موقعیت *(BestPosition)* ثبت می‌گردند.

با تکمیل مقداردهی اولیه، الگوریتم وارد حلقه‌ی اصلی می‌شود که از شمارشگر *t=1* شروع شده و تا رسیدن به *MaxIter* ادامه می‌یابد. در هر تکرار از این حلقه‌ی اصلی، ابتدا برای هر فرد i در جمعیت، میزان تناسب یا همان *fit(i)* محاسبه می‌شود. اگر این مقدار تناسب از *BestFitness* فعلی بهتر باشد، مقادیر *BestFitness* و *BestPosition* با مقادیر جدید به‌روزرسانی می‌شوند تا بهترین پاسخ تاکنون ذخیره گردد. سپس، الگوریتم به مرحله‌ی کلیدی انتشار پیچک یا *Ivy Propagation* می‌رسد. در این مرحله، برای هر فرد در جمعیت بررسی می‌شود که آیا شطر مشخص شده برقرار است یا خیر. اگر این شرایط برقرار باشد، یک راه‌حل جدید یا *newsol* برای این فرد تولید می‌شود. مقدار *GV* مربوط به این فرد به‌روزرسانی می‌شود. سپس مقادیر *newsol* به بازه‌ی [۰،۱] محدود می‌شوند تا از خروج آنها از دامنه‌ی معتبر جلوگیری شود. پس از آن، این مقادیر پیوسته‌ی *newsol* با استفاده از تابع انتقال سیگموئید به مقادیر باینری (صفر و یک) تبدیل می‌شوند تا با فضای گسسته‌ی مسأله سازگار شوند.

در صورتی که راه‌حل تولیدشده *newsol* نامعتبر باشد (برای مثال، وزن اقلام انتخاب‌شده از ظرفیت کوله‌پشتی فراتر رود)، فرایند ترمیم و بهبود اجرا می‌شود تا راه‌حل به یک حالت معتبر و عملی بازگردانده شود. پس از این ترمیم، میزان تناسب راه‌حل جدید ارزیابی می‌شود. اگر این راه‌حل جدید عملکرد بهتری داشته باشد، بهترین پاسخ موجود به‌روزرسانی می‌شود.

در ادامه، جمعیت جدید و فعلی با یکدیگر ادغام می‌شوند و تمام افراد بر اساس میزان تناسب به ترتیب نزولی مرتب می‌گردند. از بین این افراد، تنها N فرد برتر انتخاب می‌شوند تا در تکرار بعدی باقی بمانند. این فرآیند با نام حذف رقابتی یا *Competitive Exclusion* شناخته می‌شود که موجب می‌گردد بهترین افراد در جمعیت حفظ شوند. در پایان هر تکرار، مقدار شاخص همگرایی یا *Convergence Curve* نیز به‌روزرسانی می‌شود تا روند پیشرفت الگوریتم در طول زمان ثبت گردد.

در نهایت، پس از تکمیل همه‌ی تکرارها، الگوریتم بهترین مقدار تناسب *(BestFitness)*، موقعیت بهترین پاسخ *(BestPosition)*، منحنی همگرایی *(ConvergenceCurve)* و زمان سپری‌شده *(ElapsedTime)* را به عنوان خروجی بازمی‌گرداند.

5,3. اثبات همگرایی الگوریتم BiIVY

فضای جستجوی مسأله کوله‌پشتی صفر–یک شامل مجموعه‌ای متناهی از بردارهای باینری به طول n است؛ بنابراین، تعداد کل راه‌حل‌های ممکن برابر با 2^n و محدود می‌باشد. الگوریتم BiIVY یک الگوریتم تصادفی مبتنی بر جمعیت است که در هر تکرار، مجموعه‌ای از راه‌حل‌های باینری را با استفاده از عملگرهای رشد جهت‌دار، نگاشت سیگموئید، جهش تصادفی و الگوریتم‌های ترمیم و بهبود تولید می‌کند.

فرآیند تولید جمعیت در BiIVY را می‌توان به‌صورت یک زنجیره مارکوف همگن در نظر گرفت که حالت‌های آن متناظر با تمام زیرمجموعه‌های ممکن فضای جستجوی باینری هستند. به دلیل وجود مؤلفه‌های تصادفی در فرآیند به‌روزرسانی موقعیت‌ها (توزیع نرمال، تابع سیگموئید و اعداد تصادفی یکنواخت)، برای هر راه‌حل باینری مجاز احتمال تولید آن در هر تکرار مقدار غیرصفری دارد. علاوه بر این، وجود مکانیزم انتخاب نخبه‌گرا (elitism) در الگوریتم تضمین می‌کند که بهترین راه‌حل یافت‌شده در هر تکرار حفظ شود و مقدار تابع هدف به‌صورت غیرکاهشی بهبود یابد. از آنجا که فضای حالت محدود بوده و زنجیره مارکوف حاصل، غیرقابل‌کاهش و نامتناوب است، طبق نظریه زنجیره‌های مارکوف، این فرآیند دارای توزیع مانای یکتا خواهد بود. در نتیجه، با افزایش تعداد تکرارها، احتمال بازدید از حالت متناظر با جواب بهینه سراسری افزایش یافته و در حد بی‌نهایت به 1 همگرا می‌شود. بنابراین، الگوریتم BiIVY دارای خاصیت همگرایی احتمالی به جواب بهینه سراسری بوده و با افزایش تعداد تکرارها، تضمین می‌شود که الگوریتم با احتمال یک به پاسخ بهینه سراسری دست یابد.

4. نتایج

در این بخش به بررسی و تحلیل نتایج حاصل از پیاده‌سازی نسخه باینری الگوریتم پیچک برای مسأله کوله‌پشتی صفر و یک پرداخته می‌شود. هدف از این بخش ارزیابی میزان کارایی و پایداری الگوریتم ارائه‌شده در حل این نوع مسائل بوده است. بدین منظور، مجموعه‌ای شامل 25 مسأله‌ی استاندارد تحت عنوان L1 تا L25 مورد استفاده قرار گرفت که ویژگی‌های هر مسأله شامل تعداد آیتم‌ها، وزن‌ها، سود هر آیتم و ظرفیت کوله‌پشتی به‌صورت مشخص در اختیار قرار داشته است (جدول 1). این مجموعه از مسائل به‌صورت گسترده در مقالات و پژوهش‌های پیشین برای بررسی کیفیت الگوریتم‌های فراابتکاری به کار گرفته شده‌اند و دارای سطوح مختلفی از سختی هستند [11]. نتایج الگوریتم BiIVY با الگوریتم گرده افشانی گل دودویی (BFPA) و الگوریتم الگوریتم سینوسی کسینوس (BSCA) مقایسه می‌شوند.

محیط اجرایی این الگوریتم ها عبارت است از: نرم افزار مطلب نسخه 2017 و ویندوز Ultimate10 نسخه 64 بیتی با CPU Intel® Core™ i5 و RAM 8GB. نتایج ارائه شده شامل معیارهای بهترین (Best)، بدترین (Worst)، میانگین (Mean) مقادیر بدست آمده است.

به‌منظور ارز یابی عملکرد الگوریتم پیچک دودویی برای حل مسأله کوله‌پشتی صفر-یک، این الگوریتم تحت شرایطی مشخص با تعداد جمعیت N برابر با 30 و حداکثر تکرارها برابر با 1000 اجرا شده است. برای هر یک از 25 نمونه مسأله استاندارد کوله‌پشتی(L1 تا L25)، الگوریتم 30 بار به صورت مستقل اجرا گردیده و نتایج ثبت شده است. مقادیر اولیه‌ی پارامترهای الگوریتم BiIVY از مقالات پیشین اقتباس شده‌اند و به‌گونه‌ای انتخاب شده‌اند که عملکرد الگوریتم در مسائل کوله‌پشتی 0–1 مطابق استانداردهای گزارش‌شده در منابع معتبر باشد. این انتخاب پارامترها باعث می شود نتایج ارائه شده قابل مقایسه با الگوریتم‌های مشابه با شد و تاثیر پارامترها بر کیفیت پا سخ‌ها به وضوح قابل مشاهده باشد. در جدول 2، سه معیار Best، Worst و Mean به‌همراه تعداد تکرارهای موردنیاز برای رسیدن به جواب بهینه و همچنین زمان صرف شده برای هر نمونه مسأله آورده شده‌اند، این نتایج، کارایی و پایداری الگوریتم را در یافتن پاسخ‌های بهینه برای مسأله کوله‌پشتی نشان می‌دهند.

از آنجا که نسخه‌ی پیاده‌سازی شده از الگوریتم پیچک در این پژوهش، به صورت نسخه‌ی دودویی تو سعه داده شده ا ست، نتایج نهایی این الگوریتم نیز در قالب بردارهای دودویی به دست آمده‌اند. در این بردارها، هر مقدار 1 نشان‌دهنده‌ی انتخاب شدن آن آیتم در کوله‌پشتی و هر مقدار 0 نشان‌دهنده‌ی عدم انتخاب آیتم مربوطه است. در جدول 3 نتایج این بردارهای باینری برای نمونه‌های مختلف آورده شده است. این جدول نشان‌دهنده‌ی بهترین بردار باینری حا صل شده از میان 30 اجرای مستقل برای هر نمونه‌ی تستی است که بهینه‌ترین ترکیب آیتم‌ها را در کوله‌پشتی مطابق با ظرفیت و سوددهی مسأله مشخص می‌کند. برای ارزیابی عملکرد نسخه باینری الگوریتم پیچک، دو الگوریتم فراابتکاری شناخته شده یعنی الگوریتم گرده‌افشانی گل دودویی BFPA و الگوریتم سینوسی-کسینوسی باینری BSCA به عنوان الگوریتم‌های مرجع انتخاب شدند. این انتخاب بر ا ساس سابقه موفقیت‌آمیز آن‌ها در حل مسائل کوله‌پشتی 0–1، تفاوت در سازوکار جستجو و بهره‌برداری نسبت به BiIVY و استفاده گسترده آن‌ها در مطالعات مرجع انجام شده ا ست. بهره‌گیری از این الگوریتم ها امکان مقایسه دقیق عملکرد BiIVY از نظر کیفیت پا سخ‌ها و تعداد تکرارهای لازم را فراهم می‌کند و نشان می‌دهد که نسخه پیشنهادی در چه زمینه‌هایی برتری دارد. معیارهای مدنظر جهت مقایسه، شامل کل سود کسب‌شده ، تعداد تکرارهای لازم جهت دستیابی به پا سخ و زمان اجرای الگوریتم می‌باشند. نتایج حاصل از این مقایسه در جدول4 ارائه شده است. بر ا ساس نتایج گزارش شده، الگوریتم پیشنهادی پیچک دودویی در اکثر موارد از منظر کل سود و تعداد تکرار نسبت به دو الگوریتم دیگر عملکرد مطلوب‌تری داشته و توانسته است پا سخ‌های بهینه‌تری را با تعداد تکرار کمتری به د ست آورد. این امر نشان‌دهنده‌ی توانمندی این الگوریتم در بهره‌برداری منا سب از فضای جست‌وجو و حرکت سریع‌تر به سوی پا سخ‌های بهینه است.

برای بررسی آماری عملکرد الگوریتم‌ها، آزمون فریدمن به منظور مقایسه میان چند الگوریتم مورد استفاده قرار گرفت و در صورت نیاز، آزمون ویلکاکسون برای مقایسه جفتی اعمال شد. نتایج نشان داد که الگوریتم BiIVY در بیشتر مسائل نسبت به BFPA و BSCA عملکرد بهتری دارد. این برتری نا شی از مکانیزم ر شد جهت‌دار بهبود یافته، تابع جریمه کارآمد و توازن مناسب بین اکتشاف و بهره‌برداری است که باعث می‌شود الگوریتم بتواند پاسخ‌های با کیفیت بالا را با تعداد تکرار کمتر ارائه دهد. مقدار p-value به دست آمده نشان داد که تفاوت عملکرد الگوریتم BiIVY با الگوریتم‌های BFPA و BSCA در بیشتر مسائل معنی‌دار ا ست، به‌گونه‌ای که این برتری از نظر آماری تأیید شده است [34] .

جدول 1: جزئیات مسائل تست کوله پشتی صفر-یک

Instance	Dimension	Capacity	Optimal	Weights	Profits
L1	10	269	295	w = 95, 4, 60, 32, 23, 72, 80, 62, 65, 46	p = 55, 10, 47, 5, 4, 50, 8, 61, 85, 87
L2	20	878	1024	w = 92, 4, 43, 83, 84, 68, 92, 82, 6, 44, 32, 18, 56, 83, 25, 96, 70, 48, 14, 58	p = 44, 46, 90, 72, 91, 40, 75, 35, 8, 54, 78, 40, 77, 15, 61, 17, 75, 29, 75, 63
L3	4	20	35	w = 6, 5, 9, 7	p = 9, 11, 13, 15
L4	4	11	23	w = 2, 4, 6, 7	p = 6, 10, 12, 13
L5	15	375	481.0694	w = 56.358531, 80.874050, 47.987304, 89.596240, 74.660482, 85.894345, 51.353496, 1.498459, 36.445204, 16.589862, 44.569231, 0.466933, 37.788018, 57.118442, 60.716575	p = 0.125126, 19.330424, 58.500931, 35.029145, 82.284005,17.410810, 71.050142, 30.399487, 9.140294, 14.731285, 98.852504, 11.908322, 0.891140, 53.166295, 60.176397
L6	10	60	52	w = 30, 25, 20, 18, 17, 11, 5, 2, 1, 1	p = 20, 18, 17, 15, 15, 10, 5, 3, 1, 1
L7	7	50	107	w = 31, 10, 20, 19, 4, 3, 6	p = 70, 20, 39,37, 7, 5, 10
L8	23	10000	9767	w = 983, 982, 981, 980, 979, 978, 488, 976, 972, 486, 486, 972, 972, 485, 485, 969, 966, 483, 964, 963, 961, 958, 959	p = 981, 980, 979, 978, 977, 976, 487, 974, 970, 485, 485, 970, 970, 484, 484, 976, 974, 482, 962, 961, 959, 958, 857
L9	5	80	130	w = 15, 20, 17, 8, 31	p = 33, 24, 36, 37, 12
L10	20	879	1025	w = 84, 83, 43, 4, 44, 6, 82, 92, 25, 83, 56, 18, 58, 14, 48, 70, 96, 32, 68, 92	p = 91, 72, 90, 46, 55, 8, 35, 75, 61, 15, 77, 40, 63, 75, 29, 75, 17, 78, 40, 44
L11	30	577	1437	w = 46, 17, 35, 1, f 26, 17, 17, 48, 38, 17, 32, 21, 29, 48, 31, 8, 42, 37, 6, 9, 15, 22, 27, 14, 42, 40, 14, 31, 6, 34	p = 57, 64, 50, 6, 52, 6, 85, 60, 70, 65, 63, 96, 18, 48, 85, 50, 77, 18, 70, 92, 17, 43, 5, 23, 67, 88, 35, 3, 91, 48
L12	35	655	1689	w = 7, 4, 36, 47, 6, 33, 8, 35, 32, 3, 40, 50, 22, 18, 3, 12, 30, 31, 13, 33, 4, 48, 5, 17, 33, 26, 27, 19, 39, 15, 33, 47, 17, 41, 40	p = 35, 67, 30, 69, 40, 40, 21, 73, 82, 93, 52, 20, 61, 20, 42, 86, 43, 93, 38, 70, 59, 11, 42, 93, 6, 39, 25, 23, 36, 93, 51, 81, 36, 46, 96
L13	40	819	1821	w = 28, 23, 35, 38, 20, 29, 11, 48, 26, 14, 12, 48, 35, 36, 33, 39, 30, 26, 44, 20, 13, 15, 46, 36, 43, 19, 32, 2, 47, 24, 26, 39, 17, 32, 17, 16, 33, 22, 6, 12	p = 13, 16, 42, 69, 66, 68, 1, 13, 77, 85, 75, 95, 92, 23, 51, 79, 53, 62, 56, 74, 7, 50, 23, 34, 56, 75, 42, 51, 13, 22, 30, 45, 25, 27, 90, 59, 94, 62, 26, 11
L14	45	907	2033	w = 18, 12, 38, 12, 23, 13, 18, 46, 1, 7, 20, 43, 11, 47, 49, 19, 50, 7, 39, 29, 32, 25, 12, 8, 32, 41, 34, 24, 48, 30, 12, 35, 17, 38, 50, 14, 47, 35, 5, 13, 47, 24, 45, 39, 1	p = 98, 70, 66, 33, 2, 58, 4, 27, 20, 45, 77, 63, 32, 30, 8, 18, 73, 9, 92, 43, 8, 58, 84, 35, 78, 71, 60, 38, 40, 43, 43, 22, 50, 4, 57, 5, 88, 87, 34, 98, 96, 99, 16, 1, 25
L15	50	882	2240	w = 15, 40, 22, 28, 50, 35, 49, 5, 45, 3, 7, 32, 19, 16, 40, 16, 31, 24, 15, 42, 29, 4, 14, 9, 29, 11, 25, 37, 48, 39, 5, 47, 49, 31, 48, 17, 46, 1, 25, 8, 16, 9, 30, 33, 18, 3, 3, 3, 4, 1	p = 78, 69, 87, 59, 63, 12, 22, 4, 45, 33, 29, 50, 19, 94, 95, 60, 1, 91, 69, 8, 100, 32, 81, 47, 59, 48, 56, 18, 59, 16, 45, 54, 47, 84, 100, 98, 75, 20, 4, 19, 58, 63, 37, 64, 90, 26, 29, 13, 53, 83
L16	55	1050	2650	w = 27, 15, 46, 5, 40, 9, 36, 12, 11, 11, 49, 20, 32, 3, 12, 44, 24, 1, 24, 42, 44, 16, 12, 42, 22, 26, 10, 8, 46, 50, 20, 42, 48, 45, 43, 35, 9, 12, 22, 2, 14, 50, 16, 29, 31, 46, 20, 35, 11, 4, 32, 35, 15, 29, 16	p = 98, 74, 76, 4, 12, 27, 90, 98, 100, 35, 30, 19, 75, 72, 19, 44, 5, 66, 79, 87, 79, 44, 35, 6, 82, 11, 1, 28, 95, 68, 39, 86, 68, 61, 44, 97, 83, 2, 15, 49, 59, 30, 44, 40, 14, 96, 37, 84, 5, 43, 8, 32, 95, 86, 18
L17	60	1006	917	w = 7, 13, 47, 33, 38, 41, 3, 21, 37, 7, 32, 13, 42, 42, 23, 20, 49, 1, 20, 25, 31, 4, 8, 33, 11, 6, 3, 9, 26, 44, 39, 7, 4,34, 25, 25, 16, 17, 46, 23, 38, 10, 5, 11, 28, 34, 47, 3, 9, 22, 17, 5, 41, 20, 33, 29, 1, 33, 16, 14	p = 81, 37, 70, 64, 97, 21, 60, 9, 55, 85, 5, 33, 71, 87, 51, 100, 43, 27, 48, 17, 16, 27, 76, 61, 97, 78, 58, 46, 29, 76, 10, 11, 74, 36, 59, 30, 72, 37, 72, 100, 9, 47, 10, 73, 92, 9, 52, 56, 69, 30, 61, 20, 66, 70, 46, 16, 43, 60, 33, 84
L18	65	1319	2817	w = 47, 27, 24, 27, 17, 17, 50, 24, 38, 34, 40, 14, 15, 36, 10, 42, 9, 48, 37, 7, 43, 47, 29, 20, 23, 36, 14, 2, 48, 50, 39, 50, 25, 7, 24, 38, 34, 44, 38, 31, 14, 17, 42, 20, 5, 44, 22, 9, 1, 33, 19, 19, 23, 26, 16, 24, 1, 9, 16, 38, 30, 36, 41, 43, 6	p = 47, 63, 81, 57, 3, 80, 28, 83, 69, 61, 39, 7, 100, 67, 23, 10, 25, 91, 22, 48, 91, 20, 45, 62, 60, 67, 27, 43, 80, 94, 47, 31, 44, 31, 28, 14, 17, 50, 9, 93, 15, 17, 72, 68, 36, 10, 1, 38, 79, 45, 10, 81, 66, 46, 54, 53, 63, 65, 20, 81, 20, 42,24, 28, 1
L19	70	1426	3223	w = 4, 16, 16, 2, 9, 44, 33, 43, 14, 45, 11, 49, 21, 12, 41, 19, 26, 38, 42, 20, 5, 14, 40, 47, 29, 47, 30, 50, 39, 10, 26, 33, 44, 31, 50, 7, 15, 24, 7, 12, 10, 34, 17, 40, 28, 12, 35, 3, 29, 50, 19, 28, 47, 13, 42, 9, 44, 14, 43, 41, 10, 49, 13, 39, 41, 25, 46, 6, 7, 43	p = 66, 76, 71, 61, 4, 20, 34, 65, 22, 8, 99, 21, 99, 62, 25, 52, 72, 26, 12, 55, 22, 32, 98, 31, 95, 42, 2, 32, 16, 100, 46, 55, 27, 89, 11, 83, 43, 93, 53, 88, 36, 41, 60, 92, 14, 5, 41, 60, 92, 30, 55, 79, 33, 10, 45, 3, 68, 12, 20, 54, 63, 38, 61, 85, 71, 40, 58, 25, 73, 35
L20	75	1433	3614	w = 24, 45, 15, 40, 9, 37, 13, 5, 43, 35, 48, 50, 27, 46, 24, 45, 2, 7, 38, 20, 20, 31, 2, 20, 3, 35, 27, 4, 21, 22, 33, 11,5, 24, 37, 31, 46, 13, 12, 12, 41, 36, 44, 36, 34, 22, 29, 50, 48, 17, 8, 21, 28, 2, 44, 45, 25, 11, 37, 35, 24, 9, 40, 45, 8, 47, 1, 22, 1, 12, 36, 35, 14, 17, 5	p = 2, 73, 82, 12, 49, 35, 78, 29, 83, 18, 87, 93, 20, 6, 55, 1, 83, 91, 71, 25, 59, 94, 90, 61, 80, 84, 57, 1, 26, 44, 44, 88, 7, 34, 18, 25, 73, 29, 24, 14, 23, 82, 38, 67, 94, 43, 61, 97, 37, 67, 32, 89, 30, 30, 91, 50, 21, 3, 18, 31, 97, 79, 68, 85, 43, 71, 49, 83, 44, 86, 1, 100, 28, 4, 16
L21	8	1863633	3924400	w = 12,630, 284,975, 583,838, 575,342, 780,934, 164,152, 912,739, 412,657	p = 25,424, 604,597, 1,272,766, 1,174,735, 1,707,707, 313,906, 1,689,410, 860,062
L22	12	3256963	6473019	w = 419,614, 463,634, 305,284, 918,709, 743,181, 652,957, 256,487, 790,046, 310,107, 985,008, 43,471, 629,575	p = 800,734, 843,137, 551,965, 1,921,987, 1,429,742, 1,272,555, 552,649, 1,468,914, 645,615, 1,859,603, 89,001, 1,190,478
L23	12	2489815	5170626	w = 399,907, 241,106, 271,976, 105,019, 65,202, 864,369, 458,263, 27,528, 667,143, 681,262, 982,460, 215,395	p = 744,436, 446,887, 550,596, 191,341, 142,738, 1,571,133, 868,558, 54,288, 1,425,628, 1,318,834, 2,127,868, 422,621
L24	12	3453702	6941564	w = 992,884, 417,147, 996,822, 591,627, 482,278, 651,305, 491,683, 727,443, 135,904, 152,947, 590,330, 677,035	p = 2,157,066, 853,212, 1,845,571, 1,068,849, 962,615, 1,278,897, 1,026,191, 1,377,079, 264,669, 299,959, 1,080,762, 1,263,347
L25	12	2520392	5337472	w = 187,536, 919,812, 281,447, 290,967, 293,933, 146,982, 335,995, 76,949, 296,586, 732,368, 912,094, 566,115	p = 390,564, 1,896,554, 518,776, 534,038, 539,357, 281,514, 679,085, 164,965, 603,431, 1,601,666, 1,826,086, 1,235,821

جدول 2: جواب بهینه مسائل کوله پشتی صفر-یک که توسط BiIVY بدست آمده است.

Instance	Optimal	Best	Mean	Worst	No. of iterations	Elapsed time(s)
L1	295	295	295	295	1	0.0243
L2	1024	1024	1024	1024	1	0.0123
L3	35	35	35	35	1	0.0370
L4	23	23	23	23	1	0.0100
L5	481.0694	481.0694	481.0694	481.0694	1	0.0351
L6	52	52	52	52	1	0.0622
L7	107	107	107	107	1	0.0259
L8	9767	9767	9767	9767	2	0.2744
L9	130	130	130	130	1	0.0166
L10	1025	1025	1025	1025	1	0.0139
L11	1437	1437	1437	1437	2	0.0572
L12	1689	1689	1689	1689	2	0.0429
L13	1821	1821	1821	1821	5	0.3035
L14	2033	2033	2033	2033	2	0.0390
L15	2240	2240	2240	2240	6	0.4886
L16	2650	2650	2650	2650	3	0.0948
L17	917	917	917	917	1	1.1714
L18	2817	2817	2817	2817	13	0.0164
L19	3223	3220	3219	3219.80	1000	77.5584
L20	3614	3614	3614	3614	15	0.0851
L21	3924400	3924400	3924400	3924400	1	0.0018
L22	6473019	6473019	6473019	6473019	1	0.0419
L23	5170626	5170626	5170626	5170626	1	0.0132
L24	6941564	6941564	6941564	6941564	1	0.0287
L25	5337472	5337472	5337472	5337472	1	0.0165

جدول 3: نمایش دودویی از جواب بهینه مسائل کوله پشتی صفر-یک که توسط BiIVY بدست آمده است.

Instance	Optimal	Optimal solution vector
L1	295	0111000111
L2	1024	11111111111110101011
L3	35	1101
L4	23	0101
L5	481.0694	001010110111011
L6	52	0010111111
L7	107	1001000
L8	9767	11111111001000011000000
L9	130	11110
L10	1025	11111111101111010111
L11	1437	11111011111001110110101111011
L12	1689	11011111111010111111101101110111111
L13	1821	00111100111101111110101111100111111111
L14	2033	11110100111111011111011111111111101011111001
L15	2240	11111001011111110110111111101011111111011111111111
L16	2650	11110111101111101111101001111111111001101101110101111
L17	917	11110101111011101111101111111111111001011111100111011111111101010
L18	2817	10111010100110110001000111111111111011001000101110100011010111
L19	3223	11110011101011011011111111101011111011111111111011101101011011111111111
L20	3614	01101111101100101111111111101111100111101111111011111111001111111111101101
L21	3924400	11101100
L22	6473019	100101101011
L23	5170626	011010011011
L24	6941564	110001110100
L25	5337472	010000001101

جدول4: مقایسه بین الگوریتم‌ها در نمونه‌های مورد بررسی از مسائل کوله پشتی صفر-یک

Instance	Optimal	BiIVY			BFPA			BSCA		
		Total profit	No. of iterations	Elapsed time(s)	Total profit	No. of iterations	Elapsed time(s)	Total profit	No. of iterations	Elapsed time(s)
L1	295	295	1	0.0243	295	1	0.0172	295	1	0.0026
L2	1024	1024	1	0.0123	1024	1	0.0082	1024	110	0.0817
L3	35	35	1	0.0370	35	2	0.0148	35	1	0.0007
L4	23	23	1	0.0100	23	1	0.0058	23	1	0.0009
L5	481.0694	481.0694	1	0.0351	481.0694	1	0.0221	481.0694	1000	0.6957
L6	52	52	1	0.0622	52	1	0.0230	52	1	0.0006
L7	107	107	1	0.0259	107	1	0.0041	107	1	0.0004
L8	9767	9767	2	0.2744	9767	4	0.1050	9756	1000	0.7288
L9	130	130	1	0.0166	130	1	0.0074	130	1	0.0005
L10	1025	1025	1	0.0139	1025	1	0.0064	1025	6	0.0057
L11	1437	1437	2	0.0572	1437	1	0.0129	1422	1000	0.9166
L12	1689	1689	2	0.0429	1689	2	0.0355	1674	1000	0.9296
L13	1821	1821	5	0.3035	1821	8	0.1970	1762	1000	1.0580
L14	2033	2033	2	0.0390	2033	2	0.0255	1952	1000	0.9114
L15	2240	2240	6	0.4886	2240	9	0.2157	2351	1000	0.9677
L16	2650	2650	3	0.0948	2650	7	0.1540	2509	1000	0.9923
L17	917	917	1	1.1714	917	1	0.0216	917	1000	1.0322

L18	2817	2817	13	0.0164	2817	12	0.4788	2610	1000	1.1850
L19	3223	3220	1000	77.5584	3223	31	0.8470	3059	1000	1.1141
L20	3614	3614	15	0.0851	3614	15	0.4734	3370	1000	1.1443
L21	3924400	3924400	1	0.0018	3924400	1	0.0007	3924400	3	0.0017
L22	6473019	6473019	1	0.0419	6473019	1	0.0169	6473019	21	0.0168
L23	5170626	5170626	1	0.0132	5170626	2	0.0426	5170626	3	0.0034
L24	6941564	6941564	1	0.0287	6941564	1	0.0153	6941564	32	0.0245
L25	5337472	5337472	1	0.0165	5337472	1	0.0195	5337472	40	0.0310

در آزمون فریدمن که برای مقایسه عملکرد سه الگوریتم BiIVY، BFPA و BSCA به‌کار گرفته شد، تعداد مشاهدات برابر با 25 (مربوط به 25 مجموعه داده استاندارد L1–L25) و درجه آزادی برابر با df=k−1 و چون k برابر است با 3 پس مقدار درجه آزادی برابر 2 است. نتایج آزمون میانگین رتبه‌ها نشان می‌دهد که تفاوت عملکرد الگوریتم‌ها از نظر کل سود و تعداد تکرار معنادار است.

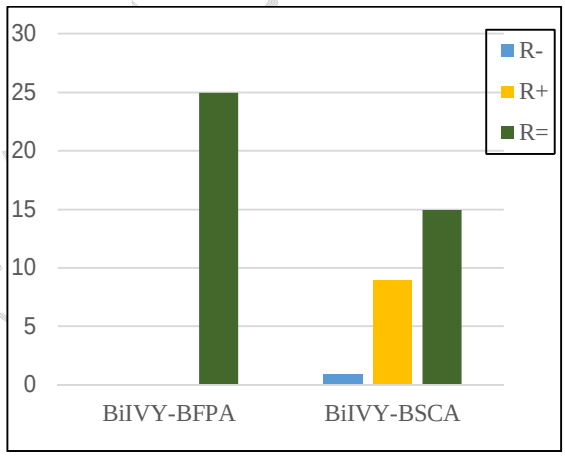
جدول 5 نتایج کلی این رتبه‌بندی را در زمینه سود نهایی حاصل از حل تمام مسائل ارائه می‌دهد. همانطور که مشاهده می شود، BiIVY به همراه BFPA در کسب سود کل حاصل‌شده، رتبه اول را به خود اختصاص داده است.

برای انجام مقایسه دقیق‌تر میان نتایج، از آزمون ناپارامتریک ویلکاکسون [35] با رتبه‌های علامت‌دار استفاده شده است. این آزمون به منظور بررسی تفاوت عملکرد میان دو نمونه وابسته به‌کار می‌رود و مشخص می‌کند که یک الگوریتم در چند مورد عملکرد بهتری، ضعیف‌تری یا مشابه با الگوریتم دیگر داشته است. خروجی این آزمون شامل سه آماره R^+، R^- و $R=$ است که به ترتیب نمایانگر تعداد مواردی هستند که الگوریتم اول بهتر، بدتر یا مشابه با الگوریتم دوم عمل کرده است. شکل 6 نتایج این آزمون را برای مقایسه زوجی الگوریتم BiIVY با دو الگوریتم BFPA و BSCA نمایش می‌دهد. همان‌طور که مشخص است، در مقایسه BiIVY با BFPA، مقادیر R^+ و R^- برابرند که نشان‌دهنده عملکرد مشابه این دو الگوریتم در زمینه کسب سود نهایی است. در مقابل، در مقایسه BiIVY با BSCA، مقدار R^+ بیشتر از R^- است که بیانگر برتری BiIVY در کسب سود کل نهایی نسبت به BSCA می‌باشد.

جدول 5: رتبه‌های کلی BiIVY و بقیه در سود کل حاصل شده در تمام موارد کوله پشتی صفر-یک

		BiIVY	BFPA	BSCA
Friedman Test	Mean Rank	2.16	2.16	1.68
	Rank	**1**	**1**	**2**

جدول 6 نتایج کلی رتبه‌بندی فریدمن را در زمینه تعداد تکرار لازم برای حل تمام مسائل ارائه می‌دهد. همانطور که مشاهده می‌شود، BiIVY دارای کمترین تعداد تکرار لازم است (رتبه 1 را کسب کرده است). همچنین شکل 7 نتایج آزمون ویلکاکسون را برای مقایسه زوجی الگوریتم BiIVY با دو الگوریتم BFPA و BSCA نمایش می‌دهد. همان‌طور که مشخص است، در همه مقایسه‌های زوجی، مقدار R^- بیشتر از R^+ است که بیانگر برتری BiIVY در کمترین تعداد تکرار لازم برای حل تمام مسائل نسبت به بقیه الگوریتم‌ها می‌باشد.

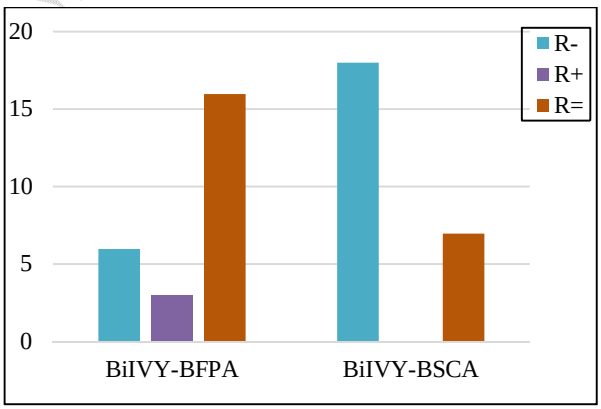


شکل6: نتایج آزمون رتبه علامت‌دار ویلکاکسون برای مقایسه BiIVY و سایر الگوریتم‌ها در سود کل به‌دست‌آمده در تمام موارد مسائل کوله‌پشتی صفر-یک

جدول 6: رتبه کلی BiIVY و بقیه در تعداد تکرارها در تمام نمونه‌های مسائل کوله پشتی صفر-یک

		BiIVY	BFPA	BSCA
Friedman Test	Mean Rank	1.58	1.70	2.72
	Rank	**1**	**2**	**3**

جدول 7 نتایج کلی رتبه‌بندی فریدمن را در زمینه مدت زمان صرف‌شده برای حل تمام مسائل ارائه می‌دهد. همانطور که مشاهده می‌شود، BiIVY به زمان زیادی برای حل مسائل نیاز دارد (رتبه 3 را کسب کرده است). همچنین شکل 8 نتایج آزمون ویلکاکسون را برای مقایسه زوجی الگوریتم BiIVY با دو الگوریتم BFPA و BSCA نمایش می‌دهد. همان‌طور که مشخص است، در همه مقایسه‌های زوجی، مقدار R^+ بیشتر از R^- است که بیانگر ضعف BiIVY در مدت زمان لازم برای حل تمام مسائل نسبت به بقیه الگوریتم‌ها می‌باشد.



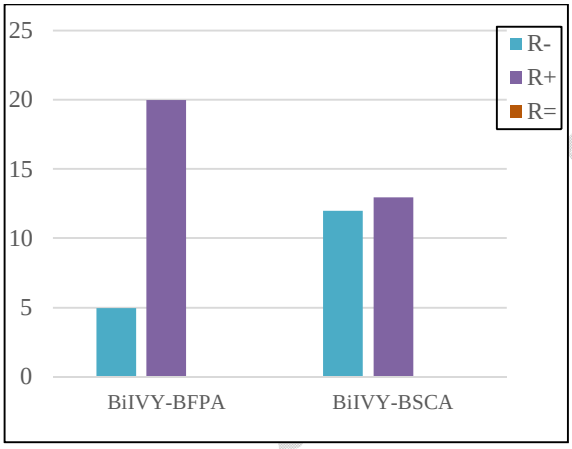
شکل 7: نتایج آزمون رتبه علامت‌دار ویلکاکسون برای مقایسه BiIVY و سایر الگوریتم‌ها از نظر تعداد تکرارها در تمام نمونه‌های مسائل کوله پشتی صفر-یک

جدول7: رتبه‌های کلی BiIVY و بقیه در زمان سپری شده در تمام موارد مسائل کوله پشتی صفر-یک

		BiIVY	BFPA	BSCA
Friedman Test	Mean Rank	2.32	1.56	2.12
	Rank	**3**	**1**	**2**

5. نتیجه‌گیری و کارهای آینده

در این مقاله، یک نسخه‌ی باینری و توسعه‌یافته از الگوریتم الهام‌گرفته از پیچک، که با نام BiIVY شناخته می‌شود، برای حل مسأله‌ی مشهور و چالش‌برانگیز کوله‌پشتی صفر و یک طراحی و پیاده‌سازی گردید. این مسأله به‌دلیل تعلق داشتن به دسته‌ی مسائل NP-Hard، همواره مورد توجه بسیاری از پژوهشگران و متخصصان حوزه‌ی بهینه‌سازی و هوش مصنوعی قرار گرفته است. هدف اصلی و محوری از ارائه و توسعه‌ی این الگوریتم، بهره‌گیری از قابلیت‌های جستجوی تصادفی، همگرایی سریع و توانایی کاوش دقیق الگوریتم پیچک در فضاهای جستجوی گسسته و باینری بود؛ تا از این طریق کیفیت و کارایی نتایج حاصل از حل چنین مسائل دشواری به‌طور چشمگیری بهبود یابد.



شکل 8: نتایج آزمون رتبه علامت‌دار ویلکاکسون برای مقایسه BiIVY و سایر موارد در زمان سپری شده در تمام موارد مسأله کوله پشتی صفر-یک

به‌منظور ارزیابی جامع کارایی و عملکرد الگوریتم BiIVY، یک مجموعه‌ی متنوع و استاندارد شامل 25 تابع مرجع مرتبط با مسائل کوله‌پشتی در مقیاس‌ها و سطوح مختلف دشواری انتخاب و استفاده گردید. نتایج حاصل از اجرای این الگوریتم با دو الگوریتم شناخته شده و مطرح دیگر در این حوزه، یعنی BFPA و BSCA، مقایسه شد تا بتوان توان نسبی الگوریتم پیشنهادی را در شرایط برابر و یکسان مورد تحلیل قرار داد. برای بررسی و

تحلیل کامل نتایج به‌دست‌آمده، سه معیار کلیدی شامل مجموع کل سود کسب‌شده، تعداد تکرار لازم برای رسیدن به پاسخ مناسب، و زمان کل اجرای الگوریتم‌ها در نظر گرفته شد؛ چرا که این شاخص‌ها می‌توانند تصویری واقعی از عملکرد الگوریتم در جنبه‌های مختلف ارائه دهند. افزون بر این، جهت بررسی اعتبار آماری نتایج و اطمینان از معنی‌داری تفاوت عملکردها، از آزمون ناپارامتریک فریدمن که یکی از روش‌های معتبر در مقایسه‌ی چند الگوریتم به‌طور هم‌زمان محسوب می‌شود، بهره گرفته شد.

با توجه به کارایی مناسب الگوریتم BiIVY در حل مسائل کوله‌پشتی در مقیاس کوچک و متوسط، پیشنهاد می‌شود نسخه‌های بهینه‌سازی‌شده و کارآمدتری از این الگوریتم برای حل مسائل بزرگ‌مقیاس طراحی و توسعه یابد. این بهبود می‌تواند در مواجهه با مسائل گسسته پیچیده‌تر همچون مسائل زمان‌بندی، تخصیص منابع و مسیریابی که از لحاظ ساختار به مسائل کوله‌پشتی شباهت دارند، نیز مورد ارزیابی قرار گیرد.

از آنجایی که مسأله کوله‌پشتی صفر و یک به‌عنوان یک مدل ریاضی در بسیاری از کاربردهای عملی و صنعتی از جمله بهینه‌سازی سبد سرمایه‌گذاری، انتخاب پروژه‌ها، طراحی شبکه‌های ارتباطی و مدیریت منابع به‌کار گرفته می‌شود، پیشنهاد می‌شود الگوریتم BiIVY در این حوزه های کاربردی مورد استفاده و ارزیابی قرار گیرد. انجام این تحلیل‌ها می‌تواند ارزش عملی الگوریتم را در حل مسائل واقعی با محدودیت‌ها و پیچیدگی‌های صنعتی اثبات نماید.

مراجع

[۱] A. Hamdipour and A. Basiri, "Memory-based ARO metaheuristic algorithm for feature selection problems," Soft Computing, pp. -, 2025, doi: 10.22052/scj.2026.257072.1309.[In Persian]

[۲] M. Ghaffar Alishahi, A. A. Pira, and A. Roohi, "Development of an Evolutionary Algorithm for City Councils to Solve Multi-Objective Optimization Problems," Soft Computing, vol. 13, no. 2, pp. 56-77, 2025, doi: 10.22052/scj.2024.253070.1154. [In Persian]

[۳] H. Shafiei, V. Rafe, and M. Amiri, "Optimization of vehicle routing based on a combination of ant colony and particle swarm algorithms with the heuristic function of the cosine of angles," Soft Computing, vol. 12, no. 2, pp. 146-164, 2024, doi: 10.22052/scj.2023.248702.1118. [In Persian]

[٤] Q. Tu, X. Chen, and X. Liu, "Multi-strategy ensemble grey wolf optimizer and its application to feature selection," *Applied Soft Computing*, vol. 76, pp. 16-30, 2019, doi: 10.1016/j.asoc.2018.11.047.

[٥] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE transactions on systems, man, and cybernetics, part b (cybernetics)*, vol. 26, no. 1, pp. 29-41, 1996, doi: 10.1109/3477.484436.

[٦] J. H. Holland, "Genetic algorithms," *Scientific american*, vol. 267, no. 1, pp. 66-73, 1992.

[٧] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: a gravitational search algorithm," *Information sciences*, vol. 179, no. 13, pp. 2232-2248, 2009, doi: 10.1016/j.ins.2009.03.004.

[٨] Y. Jiang, Q. Wu, S. Zhu, and L. Zhang, "Orca predation algorithm: A novel bio-inspired algorithm for global optimization problems," *Expert Systems with Applications*, vol. 188, p. 116026, 2022, doi: 10.1016/j.eswa.2021.116026.

[٩] S. M. Lim, A. B. M. Sultan, M. N. Sulaiman, A. Mustapha, and K. Y. Leong, "Crossover and mutation operators of genetic algorithms," *International journal of machine learning and computing*, vol. 7, no. 1, pp. 9-12, 2017.

[١٠] M. Ghasemi, M. Zare, P. Trojovský, R. V. Rao, E. Trojovská, and V. Kandasamy, "Optimization based on the smart behavior of plants with its engineering applications: Ivy algorithm," *Knowledge-Based Systems*, vol. 295, p. 111850, 2024, doi: /10.1016/j.knosys.2024.111850.

[١١] M. Abdel-Basset, D. El-Shahat, and I. El-Henawy, "Solving 0–1 knapsack problem by binary flower pollination algorithm," *Neural Computing and Applications*, vol. 31, no. 9, pp. 5477-5495, 2019, doi: 10.1007/s00521-018-3375-7.

[١٢] M. Abdel-Basset, R. Mohamed, R. K. Chakrabortty, M. Ryan, and S. Mirjalili, "New binary marine predators optimization algorithms for 0–1 knapsack problems," *Computers & Industrial Engineering*,2021, doi: 10.1016/j.cie.2020.106949.

[١٣] E. Baş, "Binary aquila optimizer for 0–1 knapsack problems," *Engineering Applications of Artificial Intelligence*, vol. 118, p. 105592, 2023, doi: 10.1016/j.engappai.2022.105592.

[۱٤] M. Gendreau and J.-Y. Potvin, *Handbook of metaheuristics*. Springer, 2010, 10.1007/978-1-4419-1665-5.

[۱٥] H. Kellerer ,U. Pferschy, D. Pisinger, H. Kellerer, U. Pferschy, and D. Pisinger, *Multidimensional knapsack problems*. Springer, 2004, doi: 10.1007/978-3-540-24777-7_9.

[۱٦] M. Gupta, "A fast and efficient genetic algorithm to solve 0–1 knapsack problem," *Int J Digit Appl Contemp Res*, vol. 1, no. 6 ,pp. 1-5, 2013.

[۱۷] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the sixth international symposium on micro machine and human science*, 1995: Ieee, pp. 39-43, doi: 10.1109/MHS.1995.494215.

[۱۸] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation*, 1997, vol. 5: ieee, pp. 4104-4108, doi: 10.1109/ICSMC.1997.637339.

[۱۹] J. C. Bansal and K. Deep, "A modified binary particle swarm optimization for knapsack problems," *Applied Mathematics and Computation*, vol. 218, no. 22, pp. 11042-11061, 2012, 10.1016/j.amc.2012.05.001.

[۲۰] P. H. Nguyen, D. Wang, and T. K. Truong, "A new hybrid particle swarm optimization and greedy for 0-1 knapsack problem ",*Indones J Electr Eng Comput Sci*, vol. 1, no. 3, pp. 411-418, 2016.

[۲۱] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *International symposium on stochastic algorithms*, 2009: Springer, pp. 169-178, doi: 10.1007/978-3-642-04944-6_14.

[۲۲] D. Zouache, F. Nouioua, and A .Moussaoui, "Quantum-inspired firefly algorithm with particle swarm optimization for discrete optimization problems," *Soft Computing*, vol. 20, no. 7, pp. 2781-2799, 2016, 10.1007/s00500-015-1681-x.

[۲۳] Y. Feng and G.-G. Wang, "An improved hybrid encoding firefly algorithm for randomized time-varying knapsack problems," in *2015 Second International Conference on Soft Computing and Machine Intelligence (ISCMI)*, 2015: IEEE, pp. 9-14, doi: 10.1109/ISCMI.2015.24.

[۲٤] R. Zhao and W. Tang, "Monkey algorithm for global numerical optimization," *Journal of Uncertain Systems*, vol. 2, no. 3, pp. 165-176, 2008.

[۲٥] Y. Zhou, X. Chen, and G. Zhou, "An improved monkey algorithm for a 0-1 knapsack problem," *Applied Soft Computing*, vol. 38, pp. 817-830, 2016, 10.1016/j.asoc.2015.10.043.

[۲٦] G.-G. Wang, S. Deb, and Z. Cui, "Monarch butterfly optimization," *Neural computing and applications*, vol. 31, no. 7, pp. 1995-2014, 2019, 10.1007/s00521-015-1923-y.

[۲۷] G.-G. Wang, S. Deb, X. Zhao, and Z. Cui, "A new monarch butterfly optimization with an improved crossover operator," *Operational Research*, vol. 18, no. 3, pp. 731-755, 2018, doi: 10.1007/s12351-016-0251-z.

[۲۸] Y. Feng, J. Yang, C. Wu, M. Lu, and X.-J. Zhao, "Solving 0–1 knapsack problems by chaotic monarch butterfly optimization algorithm with Gaussian mutation," *Memetic Computing*, vol. 10, no. 2, pp. 135-150, 2018, 10.1007/s12293-016-0211-4.

[۲۹] Y. Feng, G.-G. Wang, S. Deb, M .Lu, and X.-J. Zhao, "Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization," *Neural computing and applications*, vol. 28, no. 7, pp. 1619-1634, 2017, doi: 10.1007/s00521-015-2135-1.

[۳۰] X. Lai, J.-K. Hao, Z.-H. Fu, and D. Yue, "Diversity-preserving quantum particle swarm optimization for the multidimensional knapsack problem," *Expert Systems with Applications*, vol. 149, p. 113310, 2020, 10.1016/j.eswa.2020.113310.

[۳۱] C. Changdar, G. Mahapatra, and R. K. Pal, "An ant colony optimization approach for binary knapsack problem under fuzziness ",*Applied Mathematics and Computation*, vol. 223, pp. 243-253, 2013, doi: 10.1016/j.amc.2013.07.077.

[۳۲] L. Abualigah, D. Yousri, M. Abd Elaziz, A. A. Ewees, M. A. Al-Qaness, and A. H. Gandomi, "Aquila optimizer: a novel meta-heuristic optimization algorithm," *Computers & Industrial Engineering*, vol. 157, p. 107250, 2021, doi: 10.1016/j.cie.2021.107250.

[۳۳] K. Mahfouz, S. Ali, M. A. Al-Betar, and M. A. Awadallah, "Solving 0–1 knapsack problems using sine-cosine algorithm," in *2021 Palestinian international conference on information and communication technology (PICICT)*, 2021: IEEE, pp. 45-51, doi: 10.1109/PICICT53635.2021.00020 .

[۳٤] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *The annals of mathematical statistics*, vol. 11, no. 1, pp. 86-92, 1940.

[۳٥] F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in statistics*: Springer, 1992, pp. 196-202.