گزارش پروژه مسئله بسته‌بندی دوبعدی

کیارش صدقی قادیکلائی
دانشکده مهدسی کامپیوتر
دانشگاه اصفهان
kiarash.sedghi99@gmail.com

چکیده

مسائل بسته‌بندی و برش اشیاء ازجمله مسائلی می‌باشند که در بسیاری از صنایع که اهداف متفاوت و نیازمندی‌های متفاوتی را دنبال می‌کنند، موردتوجه قراردارند. به‌عنوان‌مثال، صنایع شیشه، چوب و کاغذسازی عمدتاً با مسئله برش کارآمد اشیایی که اشکال منظم و مشخص دارند درگیر هستند و یا صنایع مربوط به کشتی‌رانی با مسئله بارگیری و قرار دادن کالاها با اشکال هندسی متفاوت و گاهی نامنظم در کنار هم در یک مکان مشخص به‌طوری‌که ظرفیت کشتی تا حد امکان مورداستفاده قرار گیرد و فضایی بدون استفاده نماند، سروکار دارند. همان‌طور که مشخص است، این دسته از مسائل بسیار شایع و رایج هستند و این موضوع سبب می‌شود تا پیدا کردن راه‌حلی مناسب برای این دسته از مسائل ضروری شود. استفاده از علوم کامپیوتر و مدل‌سازی کامپیوتری این مسائل و درنهایت طراحی الگوریتم‌هایی کارآمد برای حل این دسته از مسائل می‌تواند راه‌حل این مسائل باشد. در این گزارش، سعی شده است راه‌حلی مبتنی بر الگوریتم‌های تکاملی جهت حل دسته‌ای خاص از این مسائل تحت عنوان بسته‌بندی دوبعدی اشیاء، ارائه شود و جزییات پیاده‌سازی این الگوریتم در قالب یک برنامه تحت عنوان kPack مورد بررسی قرار گیرد.

۱ مقدمه

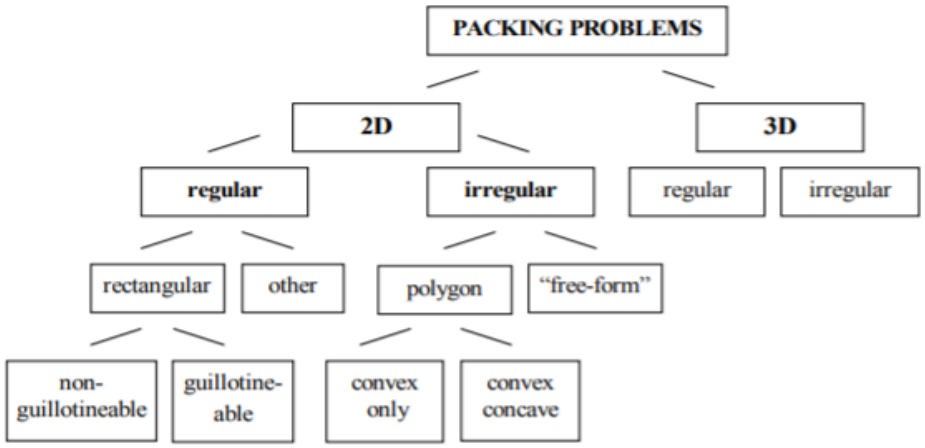
در این گزارش سعی شده است که انواع مسائل بسته‌بندی به‌اختصار موردبررسی قرار گیرند و راه‌حل‌های مبتنی بر رایانش تکاملی ارائه‌شده برای آن‌ها معرفی گردد. همچنین برای دسته‌ای مشخص از این مسائل تحت عنوان بسته‌بندی دوبعدی، راه‌حل مبتنی بر رایانش تکاملی پیشنهادشده برای آن به‌طور کامل موردبررسی قرار خواهد گرفت و برنامه‌ی kPack که پیاده‌سازی این راه‌حل هست به‌طور کامل شرح داده خواهد شد. ساختار گزارش به این صورت است که در بخش دوم، کلیت مسائل بسته‌بندی و تقسیم‌بندی آن‌ها و تحقیقات انجام‌شده در هر مورد، موردبررسی قرار خواهد گرفت. در بخش سوم به بررسی ساختار اجرایی برنامه kPack و الگوریتم تکاملی پیاده‌سازی شده در آن پرداخته خواهد شد. در بخش چهارم، برنامه kPack توسط چند مثال ارزیابی شده، و در بخش پنجم نیز نتایج به‌دست‌آمده از ارزیابی برنامه ارائه خواهد شد. بخش ششم نیز به بیان نحوه دسترسی به منبع کد پیاده‌سازی شده برای برنامه kPack خواهد پرداخت.

۲ مسائل بسته‌بندی

مسائل بسته‌بندی ازجمله مسائلی می‌باشند که هدف مسئله در آن‌ها پیدا کردن بهترین ترکیب قرار دادن اشیاء در یک یا چند ظرف به‌طوری‌که این ترکیب یا ترکیب‌ها از یک سری از قوانین تحت عنوان قیود مسئله، پیروی کنند. به‌عنوان‌مثال، از قیود مسئله می‌توان به حجم محدود ظرف یا بیشترین وزن قابل‌تحمل ظروف اشاره کرد. این قیود سبب می‌شوند تا هر ترکیب دلخواه از اشیاء موجود، قابلیت انتخاب به‌عنوان راه‌حل مسئله را نداشته باشند. در برخی از مواقع همچنین، ارزش اشیاء موجود می‌توانند با یکدیگر متفاوت باشند. در چنین مسائلی، علاوه بر وجود قیود وزن و حجم، قید انتخاب پرارزش‌ترین ترکیب یا ترکیبات از اشیاء نیز مطرح شود. همان‌طورکه مشخص است، این قیود می‌توانند مسائل ساده را به مسائل پیچیده‌ی بهینه‌سازی تبدیل کنند و پیدا کردن راه‌حلی کارآمد برای مسائل را با پیچیدگی‌های متفاوتی همراه کنند.

به‌منظور حل این دسته از مسائل، ابتدا بهتر است نگاهی به تقسیم‌بندی دسته‌های مختلف این مسائل بیندازیم. به‌طورکلی، مسائل بسته‌بندی را می‌توان به دو دسته بسته‌بندی ۲ بعدی و ۳ بعدی تقسیم نمود. در هرکدام از این دسته مسائل، ما می‌توانیم بسته‌بندی اشیایی که شکل منظم و مشخص هندسی دارند، مانند دایره، مثلث، مربع و غیره، را طرح کنیم و یا اشیایی که شکل نامنظم دارند مانند چندضلعی‌ها و اشکال محدب و مقعر.

دسته‌بندی این مسائل در شکل (۱) مشخص‌شده است:



شکل ۱: تقسیم‌بندی مسائل بسته‌بندی [۱]

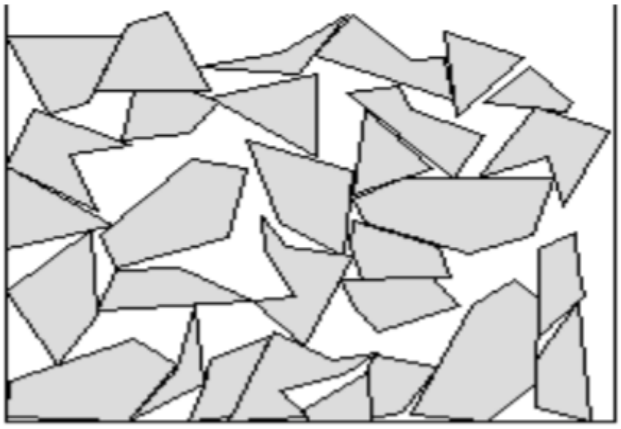
هرکدام از این دسته مسائل، نیازمندی‌ها و راه‌حل‌های پیشنهادشده مشخص و مخصوص به خود را دارند که به راه‌حل‌های ارائه‌شده برای هر دسته و پژوهش‌های انجام‌شده، در ادامه به اختصار پرداخته خواهد شد.

۱.۲ بسته‌بندی دوبعدی

اکثر مسائل بسته‌بندی اشیاء، بسته‌بندی دوبعدی اشیاء می‌باشد. این مسائل برحسب اشیایی که در آن‌ها قرار می‌گیرند، می‌توانند به دودسته تقسیم شوند. دسته‌ای از این مسائل در رابطه با بسته‌بندی اشیاء هندسی هستند که شکل منظم دارند به‌طوری‌که با چند پارامتر ساده قابل‌بیان هستند. به‌عنوان‌مثال، شیء دایره می‌تواند به‌سادگی با مختصات مرکز و شعاع بیان شود، شیء مربع به‌سادگی با مختصات مرکز و طول ضلع بیان می‌شود. در مقابل این اشیاء، اشیایی هستند که به‌سادگی و با چند پارامتر مشخص قابل‌بیان نیستند. به‌عنوان‌مثال، شکل (۲)، نمونه‌ای از کنار هم قراردادن این اشیا را نشان می‌دهد. در ادامه، به بررسی بسته بندی دوبعدی اشکال منظم و نامنظم پرداخته خواهد شد.

۱.۱.۲ بسته‌بندی دوبعدی اشکال منظم

یکی از مسائل بسته‌بندی اشکال منظم، بسته‌بندی اشکال مستطیل شکل می‌باشد. این دسته از مسائل خود، بسته‌بندی اشکال مستطیل شکلی که گیوتین شکل برش خورده‌اند و نخورده‌اند را شامل می‌شود. مقالات [۳]، [۴]، [۵] و [۶] به بررسی مسائل



شکل ۲: بسته‌بندی اشیاء نامنظم [۲]

فوق می‌پردازند. در این مسائل نیز هدف، قرار دادن مستطیل‌ها در کنار یکدیگر می‌باشد به‌طوری‌که فضای کمتری از ظرف مستطیل شکل هدر رود.

راه‌حل تکاملی و به‌ویژه الگوریتم ژنتیک اولین بار توسط smith در [۷] به‌منظور بسته‌بندی اشکال مستطیل شکل درون یک شکل مستطیل شکل با ابعاد ثابت مورداستفاده قرارگرفت. الگوریتم استفاده‌شده در این مقاله از ترکیبی از یک تابع اکتشافی و الگوریتم ژنتیک جهت قرار دادن اشیاء استفاده می‌کند. ارزیابی یک بسته‌بندی در این روش با تقسیم مساحت اشغال‌شده به مساحت اشغال نشده به دست می‌آید که بدیهی است، هرچقدر این نسبت بیشتر باشد، بسته‌بندی بهتری انجام‌شده است. این الگوریتم از دو الگوریتم دیگر تحت عنوان Sliding و Skyline استفاده می‌کند. الگوریتم Sliding به این صورت عمل می‌کند که یک شکل مستطیل را در یک گوشه از ظرف قرار می‌دهد و با حرکات زیگزاگی در امتداد قطر حرکت می‌کند تا بتواند نقطه‌ای را برای قرار دادن آن شیء پیدا نماید. الگوریتم Skyline تمامی حالات ممکن ازنظر زاویه و قرارگیری را در ظرف برای آن شیء بررسی می‌کند و در صورت تائید، آن شیء در یک محل مشخص و با زاویه‌ای مشخص قرار می‌گیرد.

یک راه‌حل تکاملی دیگر توسط Jakobs در [۸] به‌منظور کاهش پراکندگی مساحت اشغال‌شده توسط اشیاء مستطیل شکل در درون یک ظرف مستطیل شکل ارائه شد. این راه‌حل بعدها تعمیم داده شد تا چندضلعی‌ها و اشکال محدب و مقعر را نیز شامل شود. الگوریتم جایگذاری در این روش، الگوریتم BL-algorithm یا Bottom-Left-algorithm می‌باشد. به این معنی که اشیاء تا حد امکان در پایین ظرف قرار می‌گیرند و در صورت عدم امکان، تا حدامکان در سمت چپ ظرف قرار می‌گیرند. این روش به دلیل بار محاسباتی بالا، دیگر مورد توجه قرار نگرفت.

۲.۱.۲ بسته‌بندی دوبعدی اشکال نامنظم

بسته‌بندی اشکال نامنظم شامل بسته‌بندی چندضلعی‌ها و اشکال محدب و مقعر می‌باشد. راه‌حل‌های مختلفی برای بسته‌بندی چندضلعی‌ها پیشنهادشده است که هرکدام مزایا و معایب خود را دارند. در این قسمت به‌اختصار، به بررسی راه‌حل‌های ارائه‌شده برای بسته‌بندی این اشکال پرداخته خواهد شد.

یک راه‌حل ترکیبی توسط fujita در [۹] معرفی شد که این راه‌حل ترکیبی از الگوریتم ژنتیک و الگوریتم کمینه ساز محلی برای حل بسته‌بندی اشکال محدب بود. در این روش، الگوریتم کمینه ساز محلی جهت تبدیل ترکیب اشیاء به‌دست‌آمده به ترکیب فیزیکی آن‌ها استفاده شد. این روش از یک متد تحت عنوان Quasi-Newton جهت شناسایی فضای بین اشیاء و توصیف آن‌ها توسط مجموعه‌ای از متغیرها استفاده می‌کرد اما ازآنجایی‌که هزینه ارزیابی ترکیب اشیاء و هزینه بررسی همپوشانی اشیاء در این روش بسیار زیاد بود، عملاً هیچ‌گاه فرصت مقایسه با دیگر روش‌ها را پیدا نکرد.

همچنین تلاش‌های Hon and Ismail در [۱۰] و [۱۱] و Jakobs در [۸]، در تبدیل اشیاء نامنظم به اشیاء منظم مانند اشیاء مستطیل شکل نیز ازجمله دیگر راه‌حل‌هایی می‌باشد که برای حل این دسته از مسائل ارائه‌شده است. به این ترتیب که اشیاء و ظروف تا حد امکان به نزدیکترین شکل مستطیل شکل تبدیل می‌شوند.

۲.۲ بسته‌بندی سه‌بعدی

مسائل بسته‌بندی سه‌بعدی عمدتاً برای اشیاء منظم و به‌ویژه برای اشکال مکعبی تعریف می‌شوند. پیچیدگی این مسائل هنگامی‌که اشیاء نامنظم نیز به مسئله اضافه می‌شوند افزایش پیدا می‌کند.

دو الگوریتم مبتنی بر الگوریتم ژنتیک توسط Prosser در [۱۲] به‌منظور حل مسئله بسته‌بندی دسته‌های ظروف در بسته‌های مشخص به‌طوری‌که قیود هندسی و وزنی را رعایت کند، مطرح شد. مشکل اصلی اولین الگوریتم، در نظر گرفتن تعداد بسته‌های بسته‌بندی‌شده به‌عنوان تابع برازندگی بود. دلیل آن‌هم تولید تعداد کمی بسته با تراکم ظرف بالابود. به‌منظور حل این مشکل، یک الگوریتم ژنتیک ترکیبی معرفی شد. در این روش، ابتدا هر بسته تا آنجا که قیود وزن و تعداد ظروف در هر بسته اجازه می‌دهد، توسط ظروف پر می‌شود. سپس با اعمال الگوریتم ژنتیک، بسته‌های ظروف به‌صورت تصادفی با یکدیگر جابه‌جا می‌شوند.

مسئله بسته‌بندی سه‌بعدی اشیاء نامنظم در یک ظرف استوانه‌ای توسط Ikonen در [۱۳] مطرح شد. او برای نشان دادن قرارگیری هر شیء و زاویه آن، از نمایش چند–کروموزوم استفاده کرد. یک کروموزوم به‌منظور نشان دادن ترتیب قرارگیری اشیاء

مورداستفاده قرار گرفت. یک کروموزوم دیگر نیز به‌منظور نشان دادن زاویه قرارگیری اشیاء مورداستفاده قرار گرفت. در این روش، راه‌حل‌های انتخاب‌شده، توسط Linear order crossover ترکیب می‌شوند و درنهایت توسط یک الگوریتم جابه‌جایی (swapping) در مرحله جهش، جهش می‌یابند و مکان قرارگیری دو شیء و زاویه قرارگیری آن‌ها در ظرف تغییر می‌کند.

۳ برنامه kPack

برنامه kPack برنامه‌ای است که با زبان برنامه‌نویسی پایتون توسعه داده‌شده است و یک راه‌حل تکاملی برای حل مسائل بسته‌بندی دوبعدی ارائه کرده است. در ادامه ابتدا به بررسی ساختار اجرایی این برنامه پرداخته خواهد شد و نشان داده خواهد شد که برنامه از چه قسمت‌هایی تشکیل‌شده و چهارچوب کلی استفاده از آن به چه صورت است، و سپس ساختار الگوریتم تکاملی پیاده‌سازی شده در آن به‌صورت مفصل شرح داده خواهد شد.

۱.۳ ساختار اجرایی برنامه kPack

برنامه kPack برای مشخص کردن اشیاء موجود در مسئله بسته‌بندی، از یک فایل با ساختار مشخص استفاده می‌کند. توسط این فایل می‌توان ظرف یا ظرف‌هایی که قرار است اشیاء را در آن‌ها قرار دهیم به‌علاوه‌ی خود اشیاء را تعریف کرد. در اینجا منظور از تعریف اشیاء، مشخص نمودن تعداد یک شیء، پارامترهای هندسی مربوط به آن شیء ، وزن مربوط به آن شیء و ارزش مربوط به آن شیء می‌باشد. شکل (۳) نمونه‌ای از این فایل را نشان می‌دهد.

همان‌طور که در شکل (۳) مشخص است، خطوط کامنت با علامت # مشخص‌شده‌اند و کاربر می‌تواند در هرجای این فایل کامنت خود را بنویسید. طبیعی است که این خطوط توسط برنامه پردازش نمی‌شود. برای مشخص نمودن ظروف مسئله از نشانگر خاص[1] container استفاده می‌شود. از نشانگر خاص objects نیز برای تعریف اشیاء استفاده می‌شود. بعد از مشخص نمودن نشانگرهای خاص، می‌توان ظروف و اشیاء را با ساختاری مشخص و طبق زیر تعریف کرد:

$$(objectCount,)?\ objectType : parameter\ \ (parameter\ ,\ )* \qquad (۱)$$

[1] Directive

```
# This object file defines the set of objects that are used
# to define the container(s) and the item(s) that are going
# to be added to the container(s).


# Define container(s)

container:

    # One container which is a square
    circle: r= 3.3 , v=12 ,w=120


# Define Item(s)


objects:


    square: a=4.5 ,v=50 , w=40

    7,ellipse: rx= 3, ry=2 , v=8 ,w=0

    6,rti: a= 3.9 , v=4 , w=4
```

شکل ۳: نمونه فایل تعریف اشیاء در برنامه kPack

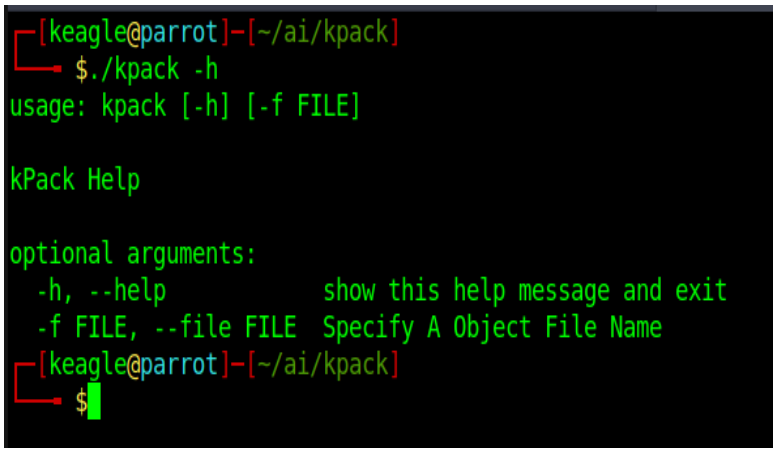
طبق این ساختار، تعداد شیء توسط objectCount مشخص می‌شود که همان‌طور مشخص است، مشخص نمودن آن اختیاری است. درصورتی‌که مشخص نشود، تعداد یک به‌صورت پیش‌فرض انتخاب می‌شود. وجود این پارامتر می‌تواند از تکرار مجدد در تعریف یک شیء در فایل جلوگیری نماید. پارامتر objectType نوع شیء را مشخص می‌کند. نوع شیء می‌تواند دایره، مربع، مثلث قائم‌الزاویه و بیضی در مورد این پروژه باشد که با اسامی circle، square، rti و ellipse به ترتیب، مشخص می‌شوند. در صورت لزوم، می‌توان اشیاء دیگر را نیز به پروژه در آینده اضافه نمود. عبارت parameter می‌تواند تمامی پارامترهایی که تاکنون صحبت شده است را مانند مشخصات هندسی شامل طول شعاع دایره، طول ضلع مربع و مثلث قائم‌الزاویه و طول شعاع‌های بیضی، ارزش و وزن آن‌ها را مشخص نماید.

بعد از ساختن فایل اشیاء، نوبت به اجرای برنامه‌ی kPack می‌رسد. این برنامه را می‌توان از طریق خط فرمان و طبق شکل (۴) اجرا کرد. این برنامه از تعدادی گزینه خط فرمان پشتیبانی می‌کند. به‌منظور دسترسی به کمک برنامه، از h– یا help– استفاده می‌شود که در شکل (۵) نشان‌داده شده است. برای مشخص نمودن فایل اشیاء، از f– یا file– استفاده می‌شود.

```
┌─[keagle@parrot]─[~/ai/kpack]
└──╼ $./kpack -f test/objects
```

شکل ۴: اجرای برنامه kpack

همان‌طورکه در شکل (۴) مشخص است، از f– برای مشخص نمودن فایل مربوط به اشیاء استفاده‌شده است.



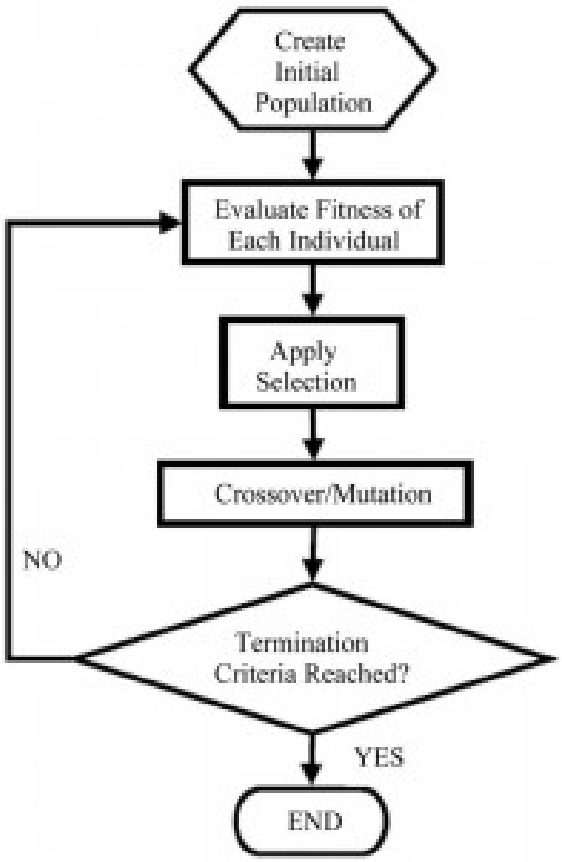
شکل ۵: گزینه h– برنامه kpack

۲.۳ ساختار الگوریتم برنامه kPack

همان‌طورکه گفته‌شد برنامه‌ی kPack یک راه‌حل تکاملی برای حل مسئله‌ی بسته‌بندی دوبعدی ارائه می‌دهد. مراحل یک راه‌حل تکاملی در شکل (۶) نشان داده‌شده است. در ادامه به بررسی هر مرحله از این الگوریتم و نحوه پیاده‌سازی آن برای مسئله بسته‌بندی دوبعدی پرداخته خواهد شد.

۱.۲.۳ ایجاد جمعیت اولیه

در یک الگوریتم تکاملی، اولین مرحله، ایجاد جمعیت اولیه از راه‌حل‌ها می‌باشد. هر یک از اعضای جمعیت اولیه، کاندیدهای اولیه برای پاسخ مسئله می‌باشد. هر یک از راه‌حل‌ها، یک ترکیب تصادفی از قرارگیری اشیاء همراه با نقطه قرارگیری و زاویه قرارگیری در درون ظرف می‌باشد. برای ساخت یک راه‌حل، یک شیء به‌صورت تصادفی از مجموعه اشیایی که موجود است انتخاب می‌شود و به‌صورت تصادفی در یک نقطه از ظرف با یک زاویه تصادفی قرار می‌گیرد. ازآنجایی‌که از قیود مسئله، عدم همپوشانی اشیاء با ظرف و اشیاء موجود دیگر در ظرف است و ازآنجایی‌که انتخاب نقطه قرارگیری و زاویه قرارگیری به‌صورت



شکل ۶: مراحل الگوریتم تکاملی [۱۴]

تصادفی انتخاب می‌شود، این احتمال وجود دارد که الگوریتم موقع ساخت یک راه‌حل، به مدت بسیار طولانی درگیر شود و عملاً ساخت جمعیت اولیه بسیار طولانی شود. برای حل این موضوع، الگوریتم تکاملی برنامه kPack از دو پارامتر کنترلی با مقادیر دلخواه به شرح زیر استفاده می‌کند:

$$initSolSetupTries = 100 \quad (۲)$$

$$objectAdditionTries = 50 \quad (۳)$$

پارامتر initSolSetupTries مشخص می‌کند که الگوریتم تکاملی چند بار جهت ساخت یک راه‌حل مسئله تلاش کند. هر بار تلاش الگوریتم در این پارامتر به معنای تلاش برای انتخاب تصادفی اشیاء و تلاش برای جایگذاری آن‌ها در ظرف می‌باشد. پارامتر objectAddtionTries نیز کنترل می‌کند که الگوریتم، بعد از انتخاب تصادفی یک شیء، چند بار تلاش کند تا آن شیء را در ظرف قرار دهد. همان‌طور که گفته شد، ازآنجایی‌که قرار دادن شیء به‌صورت تصادفی صورت می‌گیرد، لازم است تا آستانه‌ای برای آن قرارداد تا اگر الگوریتم در جایگذاری شیء ناموفق بود، از آن شیء عبور کرده و شیء دیگری برای قرار دادن انتخاب کند.

پارامتر دیگری که جمعیت اولیه را تحت تأثیر قرار می‌دهد، پارامتر populationSize می‌باشد که تعداد راه‌حل‌های اولیه را نشان می‌دهد.

$$populationSize = 100 \tag{۴}$$

۲.۲.۳ محاسبه برازندگی

مرحله بعد در الگوریتم تکاملی، محاسبه برازندگی جمعیت هدف می‌باشد. نکته‌ی مهم در مورد مسئله بسته‌بندی دوبعدی آن است که هدف نهایی این مسئله این است که مجموع ارزش اشیاء موجود در ظرف بیشینه شود به‌طوری‌که مجموع وزن اشیاء موجود در ظرف از مجموع وزن قابل‌تحمل توسط ظرف بیشتر نباشد و همچنین اشیاء نه با یکدیگر و نه با ظرف همپوشانی نداشته باشند. این موارد به‌صورت ریاضی به شرح زیر بیان می‌شوند:

$$\Sigma_{i=1}^{n} w_i x_i \leq W \tag{۵}$$

$$\forall i \in n, x_i = 1 : intersect(s_i, p_i, r_i, S) = 1 \tag{۶}$$

$$\forall i, j \in n, i \neq j, x_i = 1, x_j = 1 : intersect(s_i, p_i, r_i, s_j, p_j, r_j) = 1 \quad (۷)$$

در عبارت‌های بالا متغیرهای s، p، w و v به ترتیب مشخص‌کننده‌ی شکل هندسی، زاویه قرارگیری، وزن و ارزش شیء درون ظرف هستند. متغیرهای S و W نیز به ترتیب نشان‌دهنده‌ی شکل هندسی ظرف و وزن قابل‌تحمل ظرف می‌باشند. متغیر دودویی x نیز برای یک شیء نشان می‌دهد که آیا آن شیء درون ظرف قرار دارد یا خیر.

بنابراین، راه‌حلی به‌عنوان بهترین راه‌حل انتخاب می‌شود که ازیک‌طرف، قیود وزن و همپوشانی را رعایت کرده باشد و از طرف دیگر، بیشترین مجموع ارزش اشیاء در ظرف را داشته باشد. اما سؤالی که مطرح است این است که مقدار برازندگی یک راه‌حل به چه صورت تعریف می‌شود؟ به‌راحتی مقدار برازندگی را می‌توان متناسب با مجموع ارزش اشیاء موجود در ظرف تعریف کرد. هرچقدر مجموع ارزش اشیاء موجود در ظرف بیشتر باشد، برازندگی آن راه‌حل بیشتر است. اما این امکان وجود دارد که دو راه‌حل، مجموع ارزش اشیاء یکسانی داشته باشند. در آن صورت راهکار به‌منظور انتخاب بین این دو راه‌حل چیست؟ به‌سادگی می‌توان مساحت باقی‌مانده در ظرف را به‌عنوان داور قرارداد. به‌این‌ترتیب که هرچقدر میزان مساحت باقی‌مانده در ظرف کمتر باشد، آن راه‌حل برازنده‌تر است، چراکه نشان می‌دهد اشیاء تا حد امکان از فضاهای موجود در ظرف استفاده کرده‌اند. اما همان‌طور که مشخص است رابطه مشخصی برای برازندگی برای آنکه هردو موضوع ارزش اشیاء و مساحت باقی‌مانده گفته‌شده را دربر داشته باشد وجود ندارد. رابطه‌ای که بتواند تأثیر مثبت مجموع ارزش اشیاء و تأثیر مثبت کم بودن مساحت باقی‌مانده را نشان دهد.

برای حل این مشکل، یک رابطه به‌عنوان ایده‌ای جدید در حل این مسئله به صورت زیر پیشنهاد می‌شود:

$$Fitness\,value\ =\ \Sigma v_i\ +\ 1/(Remaining\,area) \quad (۸)$$

رابطه فوق، به‌خوبی تأثیر مجموع ارزش اشیاء و مساحت باقی‌مانده در ظرف را نشان می‌دهد. اما نکته‌ای که وجود دارد این است که این احتمال وجود دارد، اعمال این رابطه بر روی راه‌حل‌ها سبب شود که دو راه‌حل که ارزش کاملاً متفاوتی دارند، درنهایت دارای یک برازندگی شوند. دلیل آن‌هم این است که مقادیر بایستی در یک فاصله مطمئن از یکدیگر قرار گیرند. به همین دلیل لازم است قبل از محاسبه برازندگی، مقادیر مجموع ارزش اشیاء در راه‌حل‌ها و مساحت باقی‌مانده در هر راه‌حل در یک فاصله مطمئن قرار گیرند. برای این کار، مقادیر مجموع ارزش اشیاء در هر راه‌حل، با ۱ جمع می‌شوند و سپس در یک مقدار

ثابت تحت عنوان valueNormConst که فاصله اطمینان را ایجاد می‌کند ضرب می‌شوند. مقدار مساحت باقی‌مانده نیز تنها با ۱ جمع می‌شود تا زمانی که در رابطه در مخرج کسر قرار می‌گیرد، مقدار آن درصورتی‌که مقدار مساحت باقی‌مانده مقداری بین ۰ و ۱ است، به بی‌نهایت میل نکند.

$$valueNormConst = 100 \tag{۹}$$

۳.۲.۳ انتخاب راه‌حل

مرحله بعد، انتخاب والدین می‌باشد. انتخاب والدین جهت ایجاد نسل بعدی انجام می‌شود. طبیعی است، والدینی مدنظر هستند که مقدار برازندگی بیشتری داشته باشند. مقدار برازندگی هر راه‌حل در بخش قبلی توضیح داده شد و در این قسمت متناسب با آن مقادیر یک جمعیت جدید از راه‌حل‌ها تولید خواهد شد.

۴.۲.۳ ترکیب راه‌حل

در مرحله ترکیب، والدین به‌صورت دوبه‌دو ترکیب‌شده و بر اساس یک احتمال مشخص که توسط mateProb تعیین می‌شود، راه‌حل‌های جدیدی از ترکیب تولید می‌شود.

$$mateProb = 0.7 \tag{۱۰}$$

برای ترکیب راه‌حل‌ها از روش crossover استفاده می‌شود. در این روش ظرف موردنظر به‌صورت کاملاً تصادفی به دو قسمت تقسیم می‌شود. روشی که kPack برای این کار استفاده می‌کند به این صورت است که ابتدا یک نقطه به‌صورت تصادفی در ظرف انتخاب‌شده و سپس یک زاویه شیب‌خط نیز به‌صورت تصادفی بین ۰ تا ۳۶۰ انتخاب می‌شود. خطی که از این نقطه تصادفی عبور می‌کند و دارای زاویه شیب انتخاب‌شده است، ظرف موردنظر را به دو قسمت تقسیم می‌کند. بعد از تقسیم ظرف به دو قسمت، تمامی اشیایی که در هر راه‌حل در هر قسمت ظرف قرار دارند شناسایی می‌شوند. تعدادی از اشیاء نیز در هر راه‌حل بر روی خط تقسیم‌کننده‌ی ظرف قرار می‌گیرند که آن‌ها نیز بایستی در عملیات crossover شرکت کنند. سپس عملیات

crossover به این صورت انجام می‌شود که اشیاء راه‌حل اول در قسمت اول و اشیاء راه‌حل دوم در قسمت دوم ظرف باهم و اشیاء راه‌حل اول در قسمت دوم و اشیاء راه‌حل دوم در قسمت اول ظرف نیز با یکدیگر ترکیب می‌شوند. نتیجه این ترکیب، ایجاد دو فرزند جدید به‌عنوان دو راه‌حل جدید می‌باشد. بعد از ایجاد راه‌حل‌های جدید، این امکان وجود دارد که براثر ترکیب، اشیاء تکراری در راه‌حل‌های جدید وجود داشته باشد. اشیاء تکراری در هر فرزند بایستی شناسایی و حذف گردند.

درآخر نوبت به اشیایی می‌رسد که بر روی خط تقسیم‌کننده‌ی ظرف در هر راه‌حل قرار داشتند. برای این اشیاء به این صورت عمل می‌شود که تمامی اشیایی که در ظرف اول و بر روی خط تقسیم‌کننده قرار داشتند را در یک فرآیند مشابه فرآیند ساخت راه‌حل برای جمعیت اولیه، برای قرار دادن در ظرف دوم شرکت می‌دهیم. همین کار را نیز برای اشیایی که در ظرف دوم و بر روی خط تقسیم‌کننده قرار داشتند انجام می‌دهیم. در این قسمت، می‌توان کمی آسان‌تر از ساخت جمعیت اولیه عمل کرد. این بدین معنی است که متغیرهایی که به‌عنوان آستانه برای تلاش برای قرار دادن یک شیء در ظرف مورداستفاده قرار گرفتند، می‌توانند بر اساس یک متغیر مقیاس، تعدیل شوند.

متغیر مقیاس معرفی شده به‌صورت زیر است:

$$mateItemBoundaryScale = 2 \tag{۱۱}$$

متغیر mateItemBoundaryScale متغیر objectAdditionTries را کوچکتر یا بزرگتر می‌کند که سبب می‌شود که تلاش برای اضافه کردن یک شی به ظرف بیشتر و یا کمتر شود. به‌عنوان مثال، مقدار ۲ در اینجا، تعداد تلاش‌ها برای اضافه کردن شیء به ظرف را نصف می‌کند.

۵.۲.۳ جهش راه‌حل‌ها

مرحله آخر در این فرآیند نیز مربوط به جهش فرزندان تولیدشده در قسمت قبل می‌باشد. در این مرحله نیز، فرزندان بر اساس یک احتمال مشخص که توسط mutProb تعیین می‌شود دچار جهش می‌شوند.

$$mutProb = 0.9 \tag{۱۲}$$

در مسئله بسته‌بندی، جهش یک راه‌حل می‌تواند اضافه شدن یک شیء جدید به ظرف باشد، حذف شدن یک شیء از ظرف باشد و یا تغییر در یکی از اشیاء ظرف باشد. بنابراین، جهش می‌تواند یکی از سه اتفاق اضافه شدن به ظرف، حذف شدن از ظرف و تغییر در ظرف باشد. هر یک از این سه اتفاق الزاماً با یک احتمال رخ نمی‌دهند. به همین جهت سه پارامتر mutAddProb، mutRemovProb و mutModProb مقدار احتمال این سه اتفاق را نشان می‌دهند:

$$mutAddProb = 0.7 \tag{۱۳}$$

$$mutRemovProb = 0.1 \tag{۱۴}$$

$$mutModProb = 0.2 \tag{۱۵}$$

همان‌طور که مشخص است، دو اتفاق اضافه کردن شیء و تغییر در یک شیء در ظرف، نیازمند تلاش برای قراردادن شیء جدید حاصل از دو اتفاق در ظرف می‌باشند. این تلاش مانند موضوع تلاشی که در قسمت ترکیب والدین نیز مطرح شد، می‌تواند توسط پارامترهای مقیاسی مشخصی تعدیل شود. پارامترهای مقیاسی این بخش نیز عبارت‌اند از mutAddItemScale و mutModItemScale که به شکل زیر تعریف شده‌اند:

$$mutAddItemScale = 2 \tag{۱۶}$$

$$mutModItemScale = 2 \tag{۱۷}$$

۶.۲.۳ شرط خاتمه

در برنامه‌ی kPack الگوریتم تکاملی طراحی‌شده، به تعداد ثابت ۳۰۰۰ نسل اجرا می‌شود. درنتیجه، تنها شرط پایان برنامه، پیشرفت کامل نسل‌های الگوریتم می‌باشد. اما بااین‌وجود، برنامه kPack یک شرط دیگری را نیز برای خاتمه در نظر گرفته است. این شرط بیان می‌کند که اگر مقدار بهترین برازندگی در یک جمعیت، بعد از ایجاد تعداد مشخصی از نسل در الگوریتم، ثابت ماند و تغییر نکرد که اصطلاحاً به آن همگرایی جمعیت گفته می‌شود، درآنصورت، الگوریتم خاتمه می‌یابد. پارامتر convIter تعداد چرخه‌هایی که الگوریتم بدون خاتمه درصورت ثابت ماندن بهترین برازندگی جمعیت می‌تواند طی کند را مشخص می‌کند.

$$convIter = 12 \tag{۱۸}$$

۴ ارزیابی برنامه

در این بخش به ارزیابی برنامه با چند مثال و نمایش خروجی تولیدشده توسط برنامه پرداخته خواهد شد. در هر مثال، اشیاء موردنظر و پارامترهای هندسی، پارامتر وزن و ارزش هریک در جدول موجود در همان مثال آورده شده است.

جدول مقادیری که برای متغیرهای کنترلی الگوریتم تکاملی kPack در نظر گرفته شده است، برای تمامی مثال‌ها یکسان و به شرح زیر است:

جدول ۱: پارامترهای کنترلی استفاده شده در ارزیابی برنامه

populationSize	objectAdditionTries	initSolSetupTries	mateProb	mutProb
۵۰	۱۰۰۰	۱۰	۰.۷	۰.۹

جدول ۲: پارامترهای کنترلی استفاده شده در ارزیابی برنامه (ادامه)

mutAddProb	mutRemovProb	mutModProb	mutModItemScale	mutAddItemScale
۰.۸	۰.۰۵	۰.۱۵	۲	۲

جدول ۳: پارامترهای کنترلی استفاده شده در ارزیابی برنامه (ادامه)

valueNormConst	mateItemBoundaryScale
۱۰۰۰	۲

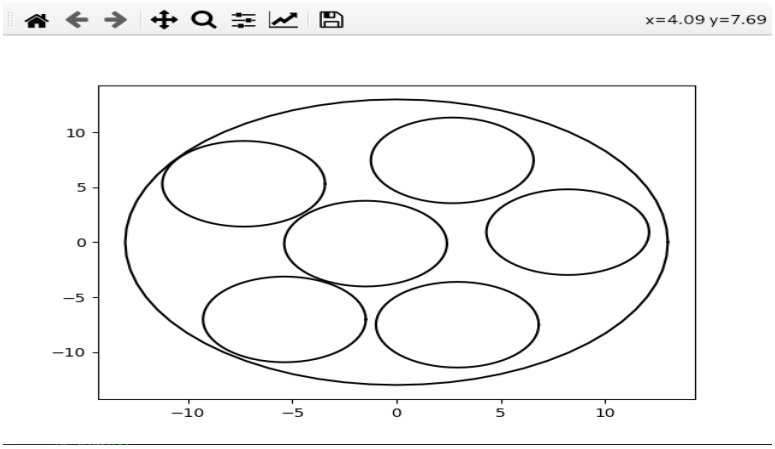
۱.۴ مثال ۱

اطلاعات مربوط به ظرف موردنظر و اشیاء داده‌شده طبق جدول زیر می‌باشد:

جدول ۴: اشیاء مثال ۱

نقش شیء	نوع شیء	تعداد	ابعاد شیء	وزن شیء	ارزش شیء
ظرف	دایره	۱	شعاع ۱۳	۱۰۰۰۰۰	۱
شیء	دایره	۷	شعاع ۳.۹	۰	۱

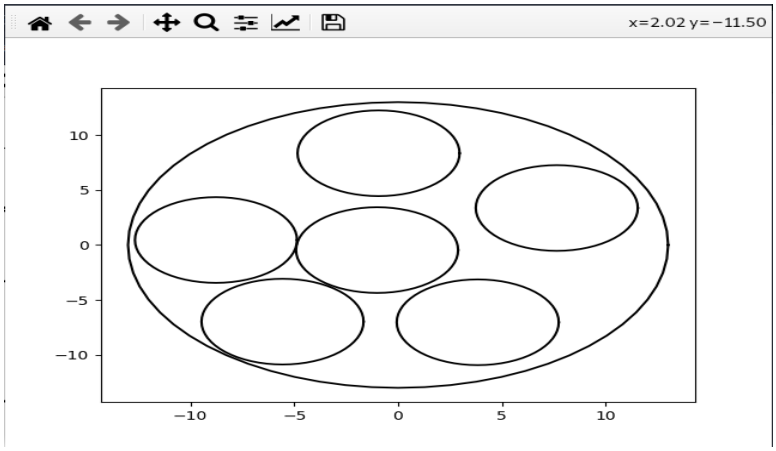
هنگامی‌که برنامه را اجرا می‌کنیم، یکی از راه‌حل‌های اولیه در شکل (۷) نمایش داده‌شده است. همان‌طور که مشخص است، تنها ۶ دایره از اشیاء موجود در راه‌حل قرار دارند و انتظاری که ما از الگوریتم داریم، به دست آوردن راه‌حلی است که ۷ دایره در ظرف به‌عنوان جواب به ما برگردانده شود. هنگامی‌که الگوریتم را اجرا می‌کنیم، مشاهده می‌کنیم که الگوریتم پس از ۲۵۵ نسل به همگرایی برخورد می‌کند و متوقف می‌شود. همگرایی فوق در شکل (۸) نشان داده‌شده است. همان‌طور که در شکل (۸) مشخص است، ۲ جواب بعد از پایان الگوریتم وجود دارد و یکی از جواب‌ها در شکل (۹) نشان داده‌شده است.



شکل ۷: مثال ۱- راه‌حل اولیه

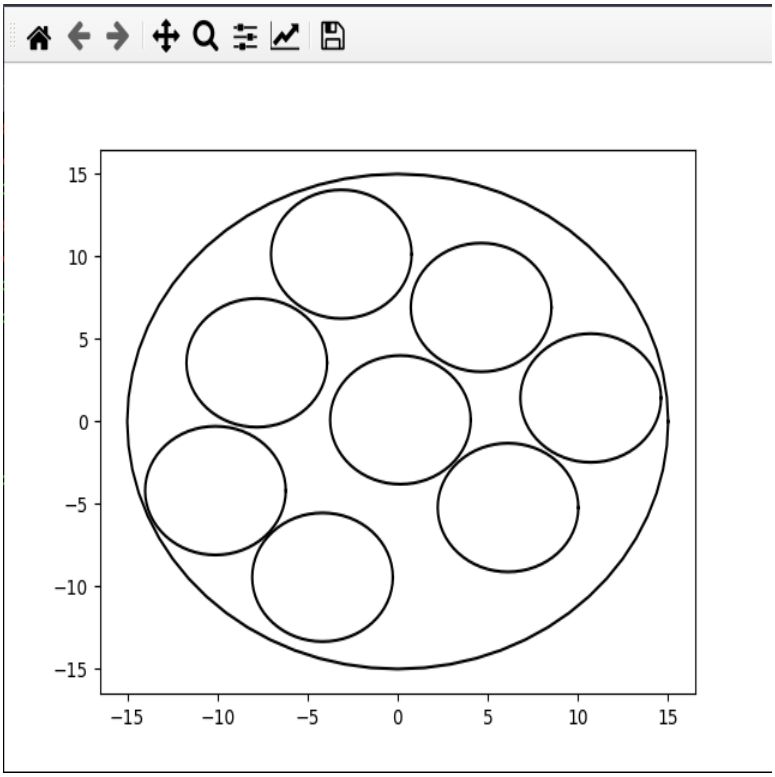
```
242
243
244
245
246
247
248
249
250
251
252
253
254
255
[INF] Convergence Detected
[INF] Multiple Solutions Have Been Found (2 Solutions)
```

شکل ۸: مثال ۱- همگرایی الگوریتم



شکل ۹: مثال ۱- اجرای کامل الگوریتم و توقف آن

اجرای چندین‌باره‌ی برنامه، همچنان ما را به جواب ۶ شیء در دایره می‌رساند. اگر شعاع ظرف را بیشتر کرده و از ۱۳ به ۱۵ برسانیم، درنهایت الگوریتم راه‌حلی که ۷ دایره در ظرف قرارگرفته‌اند را تحویل خواهد داد. شکل (۱۰) این موضوع را نشان می‌دهد.



شکل ۱۰: مثال ۱- راه‌حل برازنده‌تر پس از افزایش شعاع ظرف

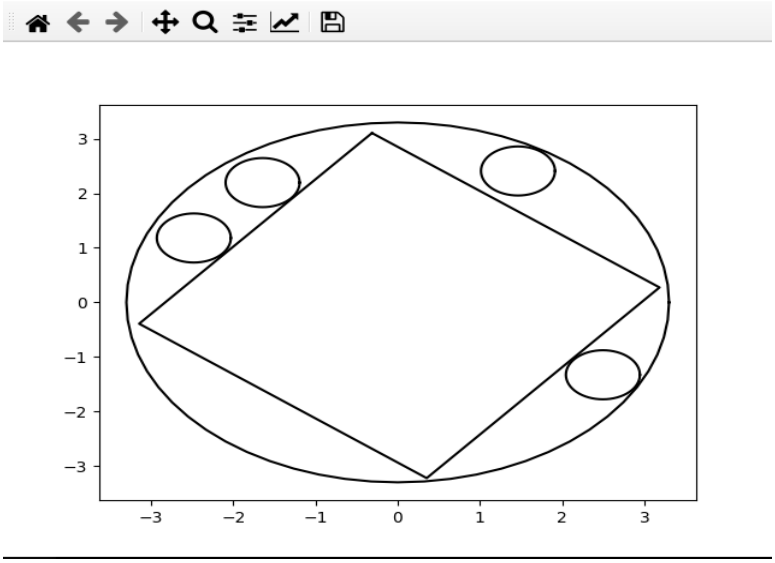
۲.۴ مثال ۲

اطلاعات مربوط به ظرف موردنظر و اشیاء داده‌شده طبق جدول زیر می‌باشد:

جدول ۵: اشیاء مثال ۲

نقش شیء	نوع شیء	تعداد	ابعاد شیء	وزن شیء	ارزش شیء
ظرف	دایره	۱	شعاع ۳.۳	۱۲۰	۱
شیء	دایره	۱	شعاع ۰.۴۵	۲۰	۵
شیء	دایره	۱	شعاع ۰.۴۵	۲۰	۱۰
شیء	دایره	۱	شعاع ۰.۴۵	۲۰	۱۵
شیء	دایره	۱	شعاع ۰.۴۵	۲۰	۲۰
شیء	مربع	۱	ضلع ۴.۵	۴۰	۵۰

نتیجه اجرای الگوریتم بر اشیاء در شکل (۱۱) نشان داده‌شده است.



شکل ۱۱: مثال ۲- راه‌حل برازنده‌تر پس از اجرای برنامه

۳.۴ مثال ۳

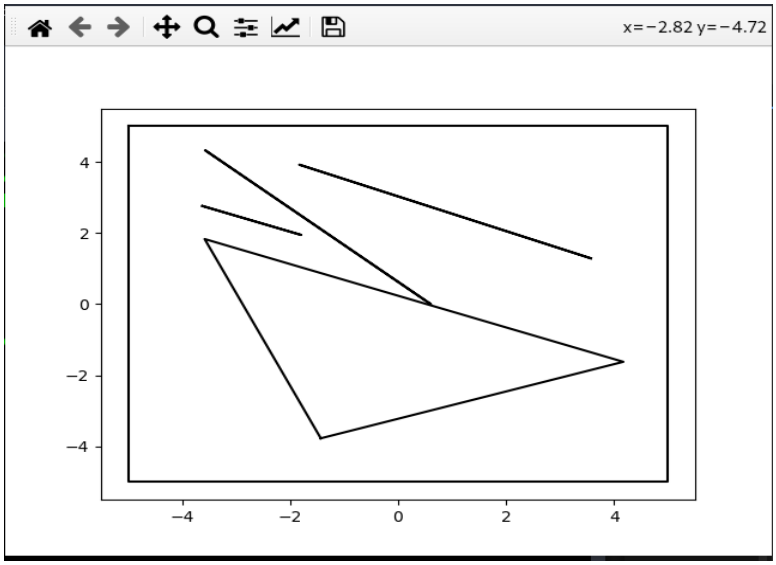
اطلاعات مربوط به ظرف موردنظر و اشیاء داده‌شده طبق جدول زیر می‌باشد:

جدول ۶: اشیاء مثال ۳

نقش شیء	نوع شیء	تعداد	ابعاد شیء	وزن شیء	ارزش شیء
ظرف	مربع	۱	ضلع ۱۰	۳۲	۱۲
شیء	مثلث قایم‌الزاویه	۱	ضلع ۶	۱۰	۱۰
شیء	مثلث قایم‌الزاویه	۱	ضلع ۶	۱۰	۲۰
شیء	بیضی	۱	قطر بزرگ ۶ قطر کوچک ۰.۶	۵	۵
شیء	بیضی	۱	قطر بزرگ ۳ قطر کوچک ۰.۶	۵	۵
شیء	بیضی	۱	قطر بزرگ ۶ قطر کوچک ۰.۶	۱۰	۵
شیء	بیضی	۱	قطر بزرگ ۳ قطر کوچک ۰.۶	۱۰	۵

هنگامی‌که برنامه را اجرا می‌کنیم، یکی از راه‌حل‌های اولیه در شکل (۱۲) نمایش داده‌شده است. همان‌طور که مشخص است، تنها ۳ بیضی و ۱ مثلث از اشیاء موجود در راه‌حل قرار دارند.

هنگامی‌که الگوریتم را اجرا می‌کنیم، مشاهده می‌کنیم که الگوریتم پس از ۲۵۵ نسل به همگرایی برخورد می‌کند و متوقف می‌شود. همگرایی فوق در شکل (۱۳) نشان داده‌شده است. همان‌طور که در شکل (۱۳) مشخص است، ۲ جواب بعد از پایان الگوریتم وجود دارد و یکی از جواب‌ها در شکل (۱۴) نشان داده‌شده است.

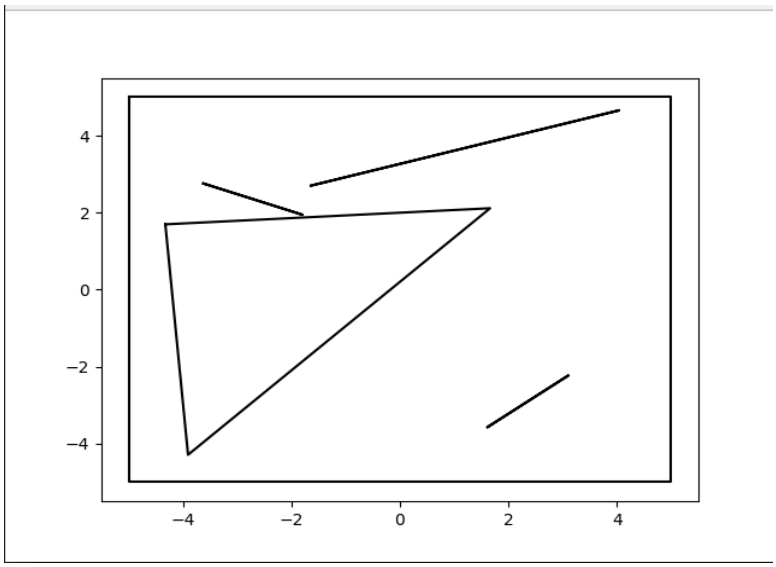


شکل ۱۲: مثال ۳- راه‌حل اولیه

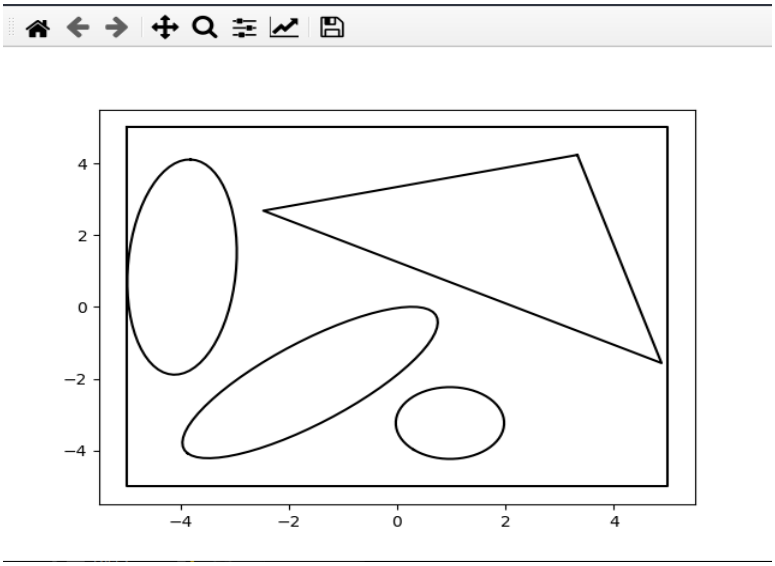
با توجه به اندازه کوچک قطر کوچک بیضی و نسبت بقیه اضلاع به یکدیگر، بیضی به‌صورت یک خط دیده می‌شود. این موضوع که آیا واقعاً بیضی رسم شده است یا خیر، به‌راحتی قابل‌اثبات است. برای این کار اگر قطر کوچک بیضی را بزرگ‌تر کرده و به ۲ برسانیم، آنگاه یکی از راه‌حل‌ها به‌صورت شکل (۱۵) دیده خواهد شد.

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
[INF] Convergence Detected
[INF] Multiple Solutions Have Been Found (2 Solutions)
```

شکل ۱۳: مثال ۳- همگرایی الگوریتم



شکل ۱۴: مثال ۳- همگرایی الگوریتم و توقف آن



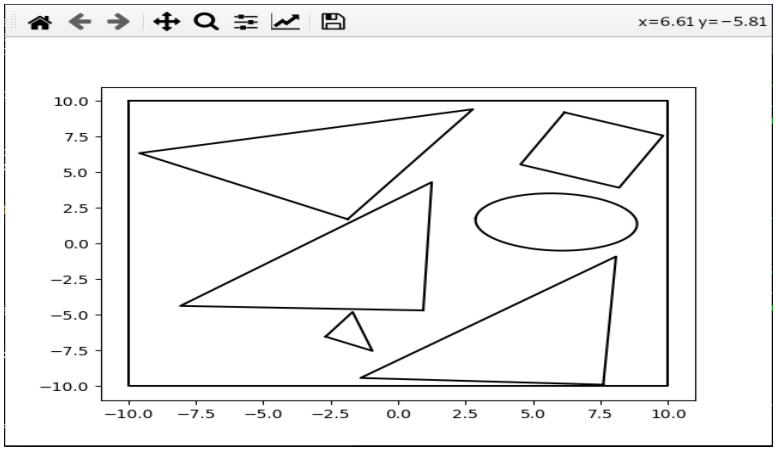
شکل ۱۵: مثال ۳- اثبات رسم بیضی

۴.۴ مثال ۴

اطلاعات مربوط به ظرف موردنظر و اشیاء داده‌شده طبق جدول زیر می‌باشد. نتیجه اجرای الگوریتم بر اشیاء در شکل (۱۶) نشان داده‌شده است.

جدول ۷: اشیاء مثال ۴

نقش شیء	نوع شیء	تعداد	ابعاد شیء	وزن شیء	ارزش شیء
ظرف	مربع	۱	ضلع ۲۰	۱۰۰	۱۲
شیء	مثلث قایم‌الزاویه	۱	ضلع ۲	۳۰	۲۰
شیء	مثلث قایم‌الزاویه	۳	ضلع ۹	۱۰	۱۰
شیء	بیضی	۱	قطر بزرگ ۶ قطر کوچک ۰.۶	۵	۵
شیء	بیضی	۱	قطر بزرگ ۶ قطر کوچک ۴	۵	۵
شیء	مربع	۱	ضلع ۴	۵	۱۵



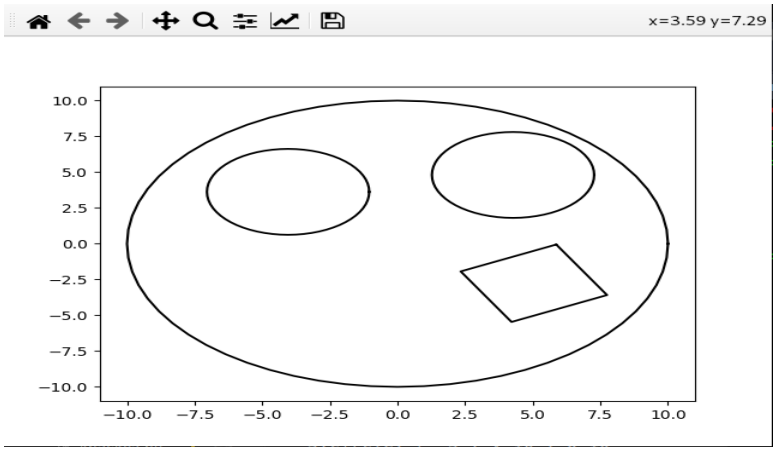
شکل ۱۶: مثال ۴- راه‌حل برازنده‌تر پس از اجرای برنامه

۵.۴ مثال ۵

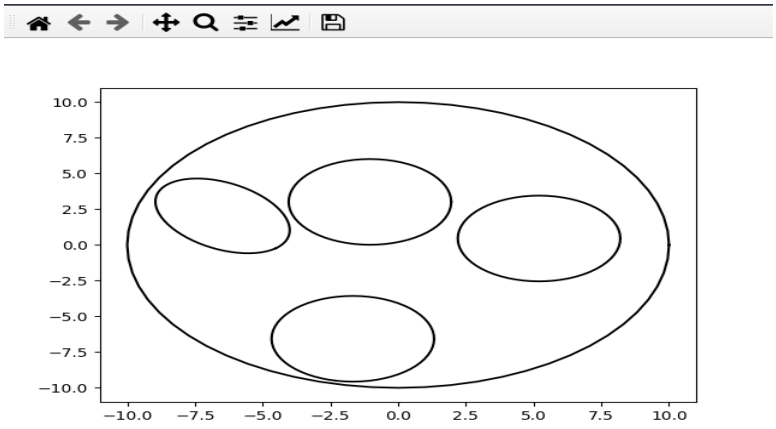
اطلاعات مربوط به ظرف موردنظر و اشیاء داده‌شده طبق جدول زیر می‌باشد. نتیجه اجرای الگوریتم بر اشیاء در شکل‌های (۱۷) و (۱۸) نشان داده‌شده است.

جدول ۸: اشیاء مثال ۵

نقش شیء	نوع شیء	تعداد	ابعاد شیء	وزن شیء	ارزش شیء
ظرف	مربع	۱	ضلع ۲۰	۱۰۰	۱۲
شیء	مثلث قایم‌الزاویه	۱	ضلع ۲	۳۰	۲۰
شیء	دایره	۳	شعاع ۳	۱۰	۱۰
شیء	بیضی	۱	قطر بزرگ ۶ قطر کوچک ۰.۶	۵	۵
شیء	بیضی	۱	قطر بزرگ ۶ قطر کوچک ۴	۵	۵
شیء	مربع	۱	ضلع ۴	۵	۱۵



شکل ۱۷: مثال ۵- راه‌حل برازنده اول پس از اجرای برنامه



شکل ۱۸: مثال ۵- راه‌حل برازنده دوم پس از اجرای برنامه

۵ تحلیل

بعد از اجرای چند مثال متفاوت و ارزیابی متفاوت پارامترهای کنترلی مربوط به الگوریتم تکاملی برنامه kPack می‌توان به نتایج زیر در مورد انتخاب پارامترهای الگوریتم تکاملی انتخابی رسید. نکته حائز اهمیت در بررسی این پارامترها این است که بررسی تنها یک پارامتر و ثابت نگه‌داشتن بقیه پارامترها نمی‌تواند به‌خوبی ما را دررسیدن به یک ترکیب مناسب کمک کند و تصمیم بر انتخاب یک ترکیب خاص، تنها با در نظرگرفتن تمامی پارامترها امکان‌پذیر است.

۱.۵ اندازه جمعیت (populationSize)

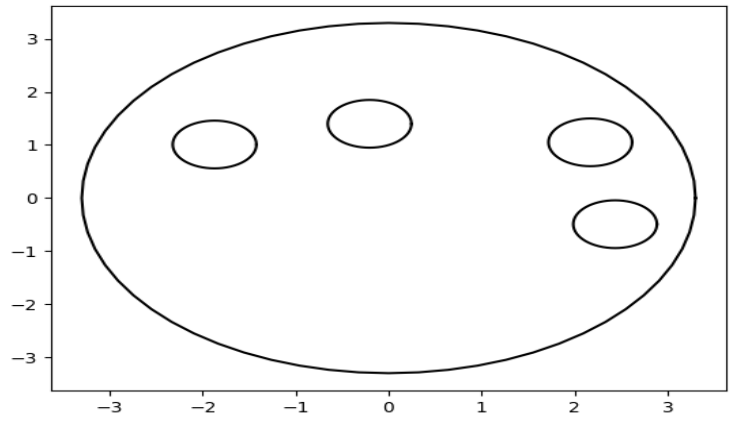
انتخاب اندازه جمعیت، می‌تواند دررسیدن به جواب مورد انتظار تأثیرگذار باشد. اگر تعداد راه‌حل‌های موجود در جمعیت زیاد باشد، می‌توان انتظار داشت که تنوع نیز در راه‌حل‌ها بیشتر خواهد بود. در روند الگوریتم، متنوع بودن راه‌حل‌های جمعیت می‌تواند در ساخت راه‌حل‌هایی با تنوع اشیاء بالا و درنتیجه مجموع برازندگی بیشتر کمک‌کننده باشد. البته اضافه کردن این نکته ضروری است که اندازه جمعیت بالا به این معنی نیست که تنوع اشیاء در یک راه‌حل در جمعیت اولیه بیشتر است و این موضوع توسط پارامترهای دیگر تعیین می‌شود.

۲.۵ تعداد تلاش برای ایجاد یک راه‌حل (initSolSetupTries)

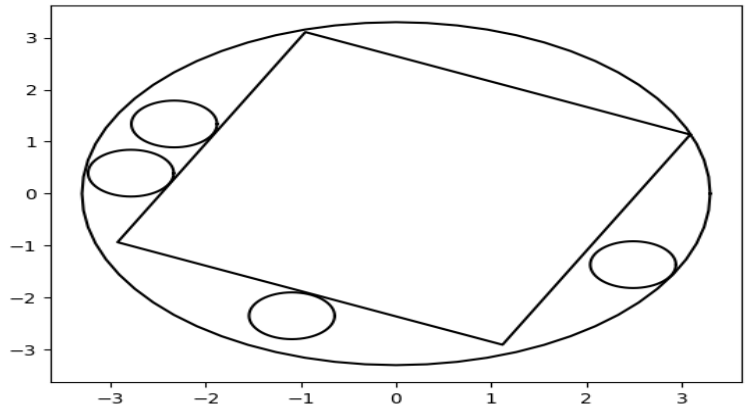
یکی دیگر از پارامترهای الگوریتم تکاملی برنامه kPack میزان تلاش برای ایجاد یک راه‌حل می‌باشد. هرچه تعداد تلاش برای ایجاد یک راه‌حل بیشتر باشد، احتمال ایجاد یک راه‌حل خوب بالاتر می‌رود. هرچه میزان و مقدار این پارامتر کمتر باشد، جمعیت اولیه با راه‌حل‌هایی ایجاد می‌شود که از برازندگی احتمالاً کمی برخوردار هستند چراکه پافشاری برای ساخت راه‌حل بهتر کم بوده‌است. این موضوع نیز طبیعی است که هر چه این پارامتر بیشتر باشد، زمانی بیشتری صرف ایجاد جمعیت اولیه به شرطی که اندازه جمعیت زیاد باشد، می‌شود.

۳.۵ تعداد تلاش برای اضافه نمودن شیء به یک راه‌حل (objectAdditionTries)

پارامتر تلاش برای اضافه نمودن شیء به یک راه‌حل، یکی دیگر از پارامترهای مهم در الگوریتم تکاملی می‌باشد. هرچه میزان این پارامتر بیشتر باشد، تلاش و پافشاری برای اضافه نمودن یک شیء به راه‌حل بیشتر خواهد بود. دلیل پافشاری نیز انتخاب تصادفی مرکز قرارگیری یک شیء و زاویه چرخش آن است. در مواقعی که اندازه اشیاء بسیار به یکدیگر نزدیک است، احتمال موفقیت در جایگذاری اشیاء بسیار کمتر می‌شود چراکه مساحت باقی‌مانده برای قرار دادن یک شیء و حق انتخاب برای موقعیت قرارگیری و زاویه چرخش آن کم شده است. در چنین شرایطی، افزایش میزان پافشاری می‌تواند احتمال موفقیت در اضافه کردن یک شیء که شاید بتواند میزان برازندگی آن راه‌حل را افزایش دهد بیشتر می‌شود.



شکل ۱۹: انتخاب ۱۰ و ۱۰۰ برای پارامترهای تلاش



شکل ۲۰: انتخاب ۵۰ و ۱۰۰۰ برای پارامترهای تلاش

به‌عنوان‌مثال، جهت نشان دادن تأثیر initSolSetupTries و objectAdditionTries دو شکل (۱۹) و (۲۰) که نشان‌دهنده یک راه‌حل اولیه در جمعیت هستند را در نظر بگیرید. در این دو شکل، هدف قرار دادن یک مربع با طول ضلع ۴.۵، وزن ۴۰ و ارزش ۵۰، چهار دایره با شعاع یکسان ۰.۴۵ و ارزش‌های به ترتیب ۱۵،۱۰،۵ و۲۰ و وزن‌های یکسان ۲۰ در یک ظرف دایره‌ای شکل با شعاع ۳.۳ و وزن ۱۲۰ می‌باشد.

در شکل (۱۸) دو پارامتر ذکرشده به ترتیب مقادیر ۱۰ و ۱۰۰ و در شکل (۱۹) مقادیر ۵۰ و ۱۰۰۰ را اختیار کرده‌اند. همان‌طور که مشخص است، هرچه میزان تلاش بیشتر شده است، شانس قرارگیری مربع در ظرف بالاتر رفته است. این نکته نیز حائز اهمیت است که اگر به ابعاد اشیاء موجود دقت شود، ابعاد بسیار به یکدیگر نزدیک است که این موضوع سبب می‌شود

یک شیء مانند مربع زمانی که ۴ دایره دیگر در ظرف قرار دارند، حق انتخاب کمتری برای نقطه قرارگیری و زاویه چرخش داشته باشد.

۴.۵ احتمال ترکیب (mateProb)

در الگوریتم تکاملی پیشنهادشده، ترکیب دو راه‌حل تغییر زیادی در جمعیت ایجاد نمی‌کند. به‌عبارت‌دیگر، بالا یا پایین بودن احتمال ترکیب دو راه‌حل به معنای تولید راه‌حل‌های مناسب یا نامناسب نمی‌باشد. دلیل آن‌هم این است که خود عملیات جهش در آینده، از اقدامات کافی مانند اضافه کردن شیء و یا تغییر در اشیاء موجود برخوردار است و می‌تواند به‌خوبی راه‌حل‌های مناسبی را ایجاد نماید. البته همان‌طور که گفته شد، بررسی هر پارامتر به‌تنهایی نمی‌تواند نتیجه‌گیری مناسبی را فراهم آورد. به‌عنوان‌مثال، اگر پارامترهای تعداد تلاش برای ایجاد راه‌حل و اضافه کردن شیء به ظرف کم باشند، درنتیجه تنوع اشیاء موجود در یک ظرف کم خواهد بود. در چنین حالتی، ترکیب دو راه‌حل می‌تواند سبب افزایش تنوع در راه‌حل‌ها شود و در چنین حالتی احتمال ترکیب بالا می‌تواند سودمند باشد.

۵.۵ احتمال جهش (mutProb)

همان‌طور که قبلاً نیز گفته شد، جهش شامل اقدامات مختلفی شامل اضافه کردن، حذف کردن و تغییر یک شیء می‌باشد. با توجه به اینکه احتمال رخ دادن اضافه کردن بیشتر از تغییر و تغییر بیشتر از حذف کردن است، می‌توان انتظار داشت که رخ دادن جهش می‌تواند سودمند باشد. در چنین شرایطی، بالا بودن احتمال جهش می‌تواند در تولید راه‌حل‌هایی بهتر سودمند باشد.

۶.۵ احتمال اضافه کردن شیء در جهش (mutAddProb)

اضافه کردن یک شیء به ظرف تنها اقدامی بین اقدامات ممکن در جهش است که می‌تواند میزان برازندگی را بیشتر نماید. بنابراین درمجموع، ما همواره دنبال اضافه کردن شیء به ظرف خواهیم بود و به همین دلیل این احتمال، نسبت به احتمال رخ دادن اقدامات دیگر بیشتر خواهد بود.

۷.۵ احتمال حذف کردن شیء در جهش (mutRemovProb)

حذف کردن یک شیء از ظرف میزان برازندگی آن ترکیب را کاهش می‌دهد. بنابراین درمجموع، ما همواره از حذف کردن اشیاء از ظرف دوری خواهیم کرد به همین دلیل این احتمال، نسبت به احتمال رخ دادن اقدامات دیگر کمترین خواهد بود. هرچند که می‌توان به‌جای استناد به انتخاب یکنواخت اشیاء درون ظرف برای حذف، آن شیء را از ظرف حذف کرد که بیشترین مساحت اشغالی، بیشترین وزن و کمترین ارزش را در بین اشیاء موجود در ظرف دارد. این عمل سبب می‌شود تا فضا برای اضافه کردن اشیاء دیگر که احتمالاً ارزش بالاتری دارند باز شود.

۸.۵ احتمال تغییر دادن یک شیء در جهش (mutModProb)

تغییر در یک شیء مانند تغییر در نقطه قرارگیری یا زاویه چرخش نیز می‌تواند فضا را برای اضافه کردن اشیاء دیگر در آینده بازنماید. احتمال رخ دادن این اقدام، مقداری بین احتمال رخ دادن دو اقدام قبل می‌باشد.

۹.۵ مقیاس تلاش در اضافه کردن شیء در جهش (mutAddItemScale)

این پارامتر می‌تواند در سرعت پیشرفت الگوریتم تأثیرگذار باشد. اما تأثیر آن در ایجاد یک راه‌حل احتمالاً مناسب عکس پارامتر تعداد تلاش برای اضافه نمودن شیء به یک راه‌حل می‌باشد. دلیل این موضوع نیز واضح است. هر چه این پارامتر کمتر باشد، تلاش ما برای اضافه کردن یک شیء به ظرف بیشتر خواهد بود. برعکس، هر چه این پارامتر بیشتر باشد، تلاش برای ایجاد جهش در یک شیء به‌منظور ساخت راه‌حلی احتمالاً مناسب کمتر خواهد بود.

۱۰.۵ مقیاس تلاش در تغییر دادن شیء در جهش (mutModItemScale)

این پارامتر می‌تواند در سرعت پیشرفت الگوریتم تأثیرگذار باشد. اما تأثیر آن ایجاد یک راه‌حل احتمالاً مناسب عکس پارامتر تعداد تلاش برای اضافه نمودن شیء به یک راه‌حل می‌باشد. دلیل این موضوع نیز واضح است. هر چه این پارامتر کمتر باشد، تلاش ما برای تغییر دادن یک شیء از پیش موجود برای اضافه کردن دوباره به ظرف بیشتر خواهد بود. برعکس، هر چه این پارامتر بیشتر باشد، تلاش برای ایجاد تغییر در یک شیء به‌منظور ساخت راه‌حلی احتمالاً مناسب کمتر خواهد بود.

۶ منبع برنامه kPack

برای دسترسی به منبع کد برنامه kPack به [۱۵] مراجعه شود.

منابع

[1] https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.20.5176&rep=rep1&type=pdf

[2] https://www.inf.utfsm.cl/ mcriff/EA/eva-space-planning/part1.pdf

[3] Dyckhoff H., 1990, Typology of cutting and packing problems, Eur. J. of OR, vol. 44, no. 2, pp. 145-159.

[4] Hässler R. W., Sweeney, P. E., 1991, Cutting stock problems and solution procedures, European Journal of Operational Research, vol. 54, part 2, pp.-150.

[5] Hinxman A. I., 1980, The trim loss and assortment problems, Eur. J. of OR, vol. 5, part 1, pp. 8-18.

[6] Kröger B., 1995, Guillontineable bin-packing: A genetic approach, Eur. J. of OR, vol. 84, pp. 645-661.

[7] Smith D., 1985, Bin-packing with adaptive search, in: Grefenstette (ed.), Proceedings of an International Conference on Genetic Algorithms and their Applications, Lawrence Erlbaum , pp. 202-206.

[8] Jakobs S, 1996, On genetic algorithms for the packing of polygons, Eur. J. of OR, vol. 88, pp. 165-181.

[9] Fujita K., Akagji, S., Kirokawa, N., 1993, Hybrid approach for optimal nesting using a genetic algorithm and a local minimisation algorithm, Proceedings of the 19th Annual ASME Design Automation Conference, Part 1 (of 2), Albuquerque, NM, USA, vol. 65, part 1, pp. 477-484.

[10]Ismail H. S., Hon K. K. B., 1992, New approaches for the nesting of two-dimensional shapes for press tool design, International Journal of Production Research, vol. 30, part 4, pp. 825-837.

[11] Ismail H. S., Hon K. K. B., 1995, Nesting of two-dimensional shapes using genetic algorithms, Proceedings of the Institution of Mechanical Engineers, Part B, vol. 209, pp. 115-124.

[12] Prosser P., 1988, A hybrid genetic algorithm for pallet loading, in: B. Radig (ed.), ECAI 88 Proceedings 8th European Conference on Artificial Intelligence, Pitman, London pp. 159-164.

[13] Ikonen I., Biles W. E., Kumar A., Ragade R. K., 1996, Concept for a genetic algorithm for packing 3D objects of complex shape, Proc. 1st Online Workshop on Soft Computing, Nagoya University, pp. 211-215.

[14] https://www3.beacon-center.org/blog/2016/02/26/an-evolutionary-computation-perspective-at-aaai-20

[15] https://github.com/keagleV/kPack