نشریه پژوهش‌های مهندسی صنایع در سیستم‌های تولید
IERPS
ISSN: 2345-2269

نشریه پژوهش‌های مهندسی صنایع در سیستم‌های تولید

سال پنجم، شماره یازدهم، پاییز و زمستان ۱۳۹۶، صفحه ۱۴۹-۱۷۱
www.ier.basu.ac.ir

DOI: 10.22084/ier.2018.11182.1512

ارائه یک الگوریتم جدید با ترکیب و بهبود روش‌های حریصانه و وزن دهی هزینه‌ها برای حل مسائل متنوع تخصیص افزونه

مظاهر ضیایی[۱،*]، فضل اله حجتی[۲]

۱. هیئت‌علمی، پژوهشگاه فضایی ایران، پژوهشکده مواد و انرژی اصفهان، ایران.
۲. کارشناسی ارشد مهندسی صنایع، پژوهشگاه فضایی ایران، پژوهشکده مواد و انرژی اصفهان، ایران.

اطلاعات مقاله

تاریخچه مقاله:
دریافت ۱۳۹۵/۰۵/۲۰
پذیرش ۱۳۹۶/۱۰/۲۰

کلمات کلیدی:
بهینه‌سازی قابلیت اطمینان
تخصیص افزونه
سیستم پل
سیستم سری-موازی

خلاصه

به دلیل اهمیت موضوع قابلیت اطمینان و نقش کلیدی آن در عملکرد سیستم‌ها و میزان و ابعاد هزینه‌های آن، بهینه‌سازی قابلیت اطمینان بخصوص مسائل تخصیص افزونه (RAP) در کانون توجه طراحان قرار گرفته است. ازآنجاکه مسائل RAP بسیار متنوع و از نوع NP-hard هستند، بنابراین برای حل هر دسته از آن‌ها روش‌های مختلفی به کار گرفته شده است که اغلب طیف محدودی از این مسائل را پوشش می‌دهند. در این مقاله یک الگوریتم جدید ارائه می‌شود که با انجام برخی بهبودهای ابتکاری بر روی روش‌های حریصانه و وزن دهی هزینه‌ها، آن‌ها را به‌گونه‌ای تلفیق می‌کند که برای حل انواع مسائل RAP با پیکربندی‌های مختلف، تنوع در قطعات و انواع استراتژی افزونگی قابل به‌کارگیری باشد. به‌کارگیری یک روش نو در وزن دهی به هزینه‌ها، استفاده از یک شاخص حریصانه ابتکاری و مکانیزم خاص جستجو در این الگوریتم باعث افزایش سرعت، دقت و انعطاف آن در حل انواع مسائل RAP می‌شود. قابلیت‌های الگوریتم از طریق حل چند مسئله، با پیکربندی متنوع و شرایط مختلف که بعضاً فضاهای جواب بسیار بزرگی دارند نشان داده می‌شود. نتایج مبین این است که الگوریتم، قادر به حل مسائل متنوع می‌باشد و جواب‌های برابر یا بسیار نزدیک به جواب بهینه یا بهترین جواب‌های موجود را در زمان کوتاه تولید می‌کند. اجرای الگوریتم به دلیل منطق عملی و ملموسی که دارد می‌تواند بینش عملیاتی طراحان را توسعه دهد.

۱- مقدمه

قابلیت اطمینان در صنایع حساس همچون صنایع نظامی و فضایی، به دلایل گزاف بودن هزینه‌های این بخش، تبعات سنگین شکست مأموریت‌ها، زمان‌بر بودن پروژه‌ها و محدود بودن تولیدات، دارای اهمیت زیادی است. مضاف بر این وابستگی زندگی روزمره بشر به عملکرد صحیح بسیاری از سیستم‌های بزرگ و پیچیده مانند سیستم‌های تولید و توزیع برق، کنترل ترافیک، سیستم‌های ماهواره‌ای، اینترنت و...، تئوری قابلیت اطمینان را به یکی از مهم‌ترین شاخص‌ها، در طراحی و مهندسی تبدیل کرده است؛ اما مسائل بهینه‌سازی قابلیت اطمینان بسیار پیچیده می‌باشند. زیرا اولاً تنوع روش‌های افزایش قابلیت اطمینان بسیار زیاد است و اغلب هرکدام راه‌حل خاص خود را می‌طلبد. ثانیاً پیکربندی و الزامات فنی سیستم‌ها نیز بسیار متفاوت است. ثالثاً ماهیت احتمالی و نوع روابط بین متغیرها به‌گونه‌ای است که شامل توابع پیچیده و بزرگی است. طبیعتاً اهداف و محدودیت‌های یک مسئله‌ی قابلیت اطمینان نیز در شرایط مختلف متفاوت است.

قابلیت اطمینان هر سیستم به قابلیت اطمینان اجزاء آن از یک‌سو و نحوه ارتباط اجزاء با یکدیگر (پیکربندی سیستم) از سوی دیگر وابسته است. از دیدگاه قابلیت اطمینان، انواع پیکربندی‌های

* نویسنده مسئول: مظاهر ضیایی
تلفن: ۳۳۲۲۲۴۲۹-۰۳۱؛ پست الکترونیکی: m.ziaei@isrc.ac.ir

سیستم‌ها به‌صورت، موازی، سری، سری-موازی، پل و سیستم‌های مختلط تقسیم می‌شوند [۱]. بـرای افزایش قابلیت اطمینان سیستم‌ها، دو رویـکرد متداول وجود دارد: رویکرد اول، تخصیص افزونه است که به‌عنوان مسائل RAP مشهورند و رویکـــــرد دوم، بالا بردن قابلیت اطمینان اجزا سیستم (مسائل انتخاب قطعه) است. این مقاله در مورد رویکرد RAP می‌باشد.

مسائل RAP جزو مسائل Np-hard است [۲]. لذا برای حل آن‌ها روش‌های بسیاری بکار گرفته‌شده است. روش‌های برنامه‌ریزی ریاضی مانند برنامه‌ریزی خطی [۳, ۴] برنامه‌ریزی عدد صحیح [۵-۷] و برنامه‌ریزی پویا [۸-۱۰]، روش‌های ابتکاری [۱۱-۱۴] و روش‌های فراابتکاری از قبیل الگوریتم ژنتیک [۱۵, ۱۶]، الگوریتم‌های مبتنی بر کلونی مورچه [۱۷, ۱۸]، جستجوی ممنوعه [۱۹]، روش‌های ترکیبی [۲۰] و... برای حل این مسائل استفاده شده است. در یک بررسی دقیق‌تر می‌توان به پژوهش‌های زیر اشاره کرد.

هسیه [۴] ضمن خطی سازی روابط مربوط به قابلیت اطمینان راهکارهایی را برای استفاده از روش‌های دقیق ریاضی برای حل مسائل RAP با پیکربندی سری-موازی ارائه کرد؛ اما با توجه به NP-hard بودن مسائل تخصیص مازاد در عمل حل مسائل با ابعاد بزرگ توسط این روش‌ها با چالش روبرو بود. کاچرال و اسمیت و کویت [۱۹] برای حل مسائل RAP با پیکربندی سری-موازی روشی مبتنی بر جستجوی ممنوعه طراحی کردند. رامیرز و کویت [۲۱] یک الگوریتم ابتکاری دومرحله‌ای برای حل مسائل تخصیص افزونه در سیستم‌های سری-موازی را باهدف حداقل کردن هزینه ارائه کردند. این الگوریتم کار را با تولید یک جواب اولیه امکان‌پذیر شروع و سپس اقدام به بهبود جواب اولیه با متد تعریف‌شده می‌نمود و درنهایت از بین بهترین راه‌حل‌های کشف‌شده تعداد مشخصی راه‌حل جدید که هزینه و قابلیت اطمینان برتری داشتند انتخاب می‌کرد. سپس این راه‌حل‌ها برای کشف مناطق امکان‌پذیری که به راه‌حل بهتر منجر می‌شوند استفاده‌شده و به‌عنوان جواب اولیه الگوریتم در نظر گرفته می‌شدند. محققان این پژوهش، روش ابتکاری خود را با الگوریتم ژنتیک مقایسه کرده و سادگی و سهولت روش ارائه‌شده را نقطه قوت آن نسبت به الگوریتم ژنتیک دانستند. کو و ون [۲۲] در سال ۲۰۰۷، مقاله‌ای تحت عنوان «پیشرفت‌های اخیر در تخصیص بهینه قابلیت اطمینان» منتشر کردند که در آن جمع‌بندی از مطالعات محققان درزمینه‌ی تخصیص بهینه قابلیت اطمینان صورت گرفت. این گزارش ضمن دسته‌بندی روش‌های حل این‌گونه مسائل به روش‌های دقیق و روش‌های غیردقیق (ابتکاری و فراابتکاری) به بررسی ضعف و قوت هرکدام پرداخته است. همچنین بیان شد، برای حل مسائل با ابعاد بزرگ و فضای جواب گسترده، روش‌های حل ابتکاری و فراابتکاری (اکتشافی و فرااکتشافی) کارآمدتر است. در این پژوهش پیشنهاد شد برای کاهش فضای جستجو، ممکن است بتوان از ترکیب دو الگوریتم فراابتکاری استفاده کرد. کومار، چاترودی و پهوجا [۱۴] نتایج تحقیق خود را با عنوان «روشی ابتکاری برای تخصیص افزونگی در سیستم‌های پیچیده» منتشر کردند. آن‌ها موضوع تخصیص افزونگی را در سیستم‌های پیچیده که شبکه‌های پل را شامل می‌شد بررسی کردند. این الگوریتم قادر به حل مسائل تخصیص افزونگی شبکه‌های پل با سرعت قابل قبولی بود. کائو، مورات و چاینام [۲۳]، در حل مسئله‌ی تخصیص افزونگی با پیکربندی سری-موازی آن را به‌عنوان یک مسئله‌ی چندهدفه در نظر گرفتند. ازاین‌رو مسئله‌ی اصلی را به چندین زیر مسئله‌ی چندهدفه تجزیه کرده و زیر مسائل را حل کردند و سپس به‌صورت سیستماتیک راه‌حل‌ها را ترکیب نمودند. روش مبتنی بر تجزیه می‌تواند تمامی راه‌حل‌های پارتو بهینه را برای مسئله‌ی تخصیص افزونگی ایجاد کند. نتایج آزمایشی نیز نشان داده است که این متد پیشنهادی، نسبت به متدهای ابتـکاری و فراابتکاری دارای عملکرد بهتری ازنظر تولید جواب بهینه بوده است اما از مدت‌زمان حل مسئله صحبتی به میان نیامده است. از سویی پیچیدگی روش ارائه شده نیـز قابل‌تأمل است. گارگ، رانی و شارما [۲۴] یک رویکرد دوفازی برای حل مسائل تخصیص افزونگی با پیکربندی‌های سری، سری-موازی و پل با محدودیت منابع غیرخطی ارائه کردند که در مرحله اول با استفاده از یک الگوریتم بر اساس کلونی زنبور مصنوعی اقدام به حل مسئله تخصیص می‌کرد و در مرحله دوم به بهبود جواب حاصل از این الگوریتم می‌پرداخت. گارگ [۲۵] در سال ۲۰۱۵، در تحقیقی دیگر به کمک الگوریتمی مبتنی بر جستجوی فاخته به بهینه یابی قابلیت اطمینان در مسائل متنوع تخصیص افزونگی با محدودیت منابع غیرخطی پرداخت. ابویی اردکان و زینال همدانی [۲۶] برای حل مسائل تخصیص افزونه با پیکربندی سری-موازی از الگوریتم ژنتیک استفاده کردند. لیو و کین [۲۷] و هانگ [۲۸] از روش ازدحام ذرات برای حل مسائل RAP استفاده کردند و برای نشان دادن قدرت الگوریتم ارائه‌شده مسائلی با پیکربندی سری، سری-موازی و پل را حل نمودند. کونگ، گائو، اویانگ و لای [۲۹]، نیز از روشی بر پایه ازدحام ذرات برای حل مسائل RAP با پیکربندی سری-موازی استفاده کردند. زنگ و چن [۳۰] مسئله‌ی تخصیص افزونگی را به‌عنوان یک مسئله بهینه‌سازی چندهدفه در نظر گرفتند که در آن قابلیت اطمینان سیستم و هزینه به‌طور هم‌زمان منظور گردیده و فرموله می‌شد. توابع هدف شامل دو تابع به‌صورت حداکثر قابلیت اطمینان و حداقل هزینه بودند. آن‌ها در این پژوهش الگوریتمی با چرخه تکرار ۵۰ بار، بر مبنای بهینه‌سازی ازدحام ذرات ارائه کردند که قادر به حل مسائل بهینه‌سازی چندهدفه مذکور بود. فیض‌آبادی و جهرمی [۳۱] در بحث‌های مربوط به بهینه‌سازی قابلیت اطمینان با استفاده از تخصیص افزونگی، ساختار سری-موازی را مورد بررسی قرار دادند. آن‌ها بیان کردند که در بسیاری از مدل‌های پیشین برای بهینه‌سازی قابلیت اطمینان سیستم‌های سری-موازی با استفاده از تخصیص مازاد، اجزاء مازاد مورداستفاده یکسان فرض شده‌اند و این

موضوع به‌عنوان یک محدودیت، طراحان سیستم را در انتخاب قطعات و دستیابی به سطوح بالاتری از قابلیت اطمینان با مشکل مواجه می‌کند. در پژوهش مذکور یک مدل جدید بر پایه الگوریتم ژنتیک برای بهینه‌سازی قابلیت اطمینان سیستم‌های سری-موازی با اجزاء غیر همسان ارائه شد.

همان‌گونه که ملاحظه شد روش‌های ارائه‌شده برای حل انواع مسائل RAP هم شامل روش‌های دقیق برنامه‌ریزی ریاضی و هم شامل روش‌های ابتکاری و فراابتکاری است. در استفاده از روش‌های برنامه‌ریزی ریاضی معمولاً با مشکل زمان‌بر بودن حل مسئله و درنتیجه محدودیت در ابعاد مسئله مواجه هستیم [۲۲, ۲۷]. بعلاوه اغلب با تغییر جزئی در مفروضات و شرایط مسئله نیاز به فرمولاسیون جدید وجود دارد [۱۹]. بدین ترتیب می‌توان نتیجه‌گیری کرد، روش‌های دقیق ریاضی ارائه شده توسط محققین مختلف در حل مسائل موضوع این پژوهش هرچند از جنبه تولید جواب بهینه دقیق هستند، اما عملاً برای حل مسائل با ابعاد بزرگ کاربرد ندارند و به سبب NP-hard بودن این مسائل، حل آن‌ها با روش‌های فراابتکاری رواج بیشتری پیدا کرده است. از طرفی پیچیدگی روش‌های حل فراابتکاری متداول، نیاز به تکرار در حل، نیاز به در نظر گرفتن پارامترهای گوناگون برای حل هر مسئله و نیاز به تغییر پارامترها متناسب با هر مسئله، نقاط ضعف روش‌های فراابتکاری محسوب می‌شود. مضاف بر این روش‌های فراابتکاری ارائه‌شده عموماً بسته به نوع پیکربندی مسئله‌ی RAP طراحی‌شده‌اند و درواقع حل مسائل با پیکربندی متنوع، عموماً طراحی الگوریتم فراابتکاری متناسب با پیکربندی خود را می‌طلبد [۲۲]. بدین ترتیب ارائه متدی که این نقاط ضعف را پوشش دهد، همچنین قادر به حل انواع مسائل RAP با پیکربندی‌های متنوع باشد، ضروری است.

یک مدل ابتکاری ساده بر پایه روش حریصانه توسط ضیایی و حجتی ارائه شد که پیچیدگی‌های موجود در الگوریتم‌های فراابتکاری متداول را نداشت و در زمان کوتاهی جواب مسئله را تولید می‌کرد؛ اما این روش تنها قادر به حل مسائل بهینه‌سازی قابلیت اطمینان سیستم‌های سری تحت محدودیت بودجه بود. این روش برای حل چند مسئله‌ی «انتخاب قطعه» با پیکربندی سری که شامل انتخاب یک قطعه از بین چند قطعه تحت محدودیت هزینه بود بکار گرفته شد [۳۲]. نتایج نشان‌دهنده، تولید جواب‌هایی مساوی و یا بسیار نزدیک به بهترین جواب‌های دیگران [۳۳] در زمان‌های بسیار کوتاه‌تر بود.

در این مقاله با تلفیق روش حریصانه و روش وزن دهی هزینه‌ها یک الگوریتم ابتکاری با منطقی ساده ارائه می‌شود که انعطاف بسیار زیادی را برای حل مسائل با پیکربندی متنوع دارد. این الگوریتم به سه دلیل عمده، دقت و سرعت بسیار بالایی در همگرایی به سمت جواب بهینه داشته که عبارت‌اند از:

۱- استفاده از یک متد نو در وزن دهی به هزینه‌ها که باهدف جستجوی دقیق‌تر جواب بهینه طراحی‌شده است.

۲- به‌کارگیری یک شاخص حریصانه ابتکاری که سرعت همگرایی به سمت جواب بهینه را به‌شدت زیاد می‌کند.

۳- مکانیزم خاص الگوریتم که برخلاف الگوریتم‌های متداول ارائه‌شده قبلی چرخه‌های تکرار طولانی را نداشته، بلکه چرخه الگوریتم تا زمانی که محدودیت منابع مسئله اجازه دهند ادامه داشته و در هر چرخه جواب قطعاً بهبود می‌یابد.

در حل مسائل با استفاده از این الگوریتم نیازی به تعریف و تغییر پارامترهای متنوع نیست و فقط با یک‌بار اجرای الگوریتم جواب تولید می‌شود. گرچه ماهیت الگوریتم به‌گونه‌ای است که جواب تولیدشده لزوماً جواب بهینه نیست اما در زمان‌های بسیار کوتاه جواب‌های به‌دست‌آمده برابر یا بسیار نزدیک به جواب بهینه هستند و در مواردی بهتر از بهترین جواب‌های موجود غیر بهینه می‌باشند.

مقاله با تشریح مسئله RAP و انواع آن در بخش ۲ شروع و با تشریح الگوریتم GW در بخش ۳ ادامه می‌یابد. سپس در بخش ۴ چندین مسئله‌ی آزمایشی با پیکربندی و شرایط مختلف توسط الگوریتم حل‌شده و با جواب الگوریتم‌های قبلی مقایسه می‌شوند. در مواردی که زمان اجرای الگوریتم‌ها گزارش شده، زمان‌ها نیز مقایسه شده‌اند. در پایان نتیجه‌گیری در بخش ۵ صورت گرفته است.

۲- تشریح مسئله RAP

در حالت کلی وقتی‌که برای بهبود قابلیت اطمینان یک سیستم، متشکل از چندین زیرسیستم، از افزونه استفاده شود، مسئله تخصیص افزونه مطرح می‌شود. برحسب پیکربندی زیرسیستم‌ها، تعداد گزینه‌ها برای هر زیرسیستم، نوع افزونه و هدف مسئله، نوع RAP متفاوت خواهد بود. بنابراین هرکدام فرمولاسیون متفاوتی دارند که ممکن است الگوریتم‌های متفاوتی برای حل آن‌ها به خدمت گرفته شود.

۲-۱- انواع RAP

مفروضاتی که RAP و نوع آن را مشخص می‌کنند، عبارت‌اند از:

- در بالاترین سطح سیستم، S زیرسیستم با پیکربندی مشخصی وجود دارد که افزونه‌ها به‌صورت موازی به زیرسیستم‌ها اضافه می‌شود. از متداول‌ترین پیکربندی‌ها، سری، موازی، سری-موازی و پل است.
- برای هر زیرسیستم i، تعداد m_i گزینه‌ی مختلف برای انتخاب به‌عنوان افزونه وجود دارد. اگر $\forall i m_i=1$ باشد به مسئله «تک گزینه‌ای» و وقتی برای بعضی از زیرسیستم‌ها $m_i>1$ باشد به آن «چندگزینه‌ای» گویند.
- افزونه معمولاً به دو نوع سرد[1] و گرم[2] (فعال) تقسیم می‌شود.

1. Could
2. Warm/Active

نوع افزونه ممکن است به الزامات فنی و قطعات مربوط باشد و در مواردی ممکن است انتخاب نوع آن جزء مسئله باشد. به‌کارگیری افزونه به هر نوعی که باشد، هزینه و قابلیت اطمینان سوئیچینگ را مطرح می‌کند.

- برای عملکرد موفق زیرسیستم i لازم است حداقل تعداد K_i قطعه در حال کار باشند. اگر برای بعضی از زیرسیستم‌ها $K_i>1$ باشد به آن مسئله «k از n» گویند.
- ممکن است برای زیرسیستم‌ها انتخاب قطعات متفاوت، مجاز یا غیرمجاز باشد. حالتی که قطعات هر زیرسیستم باید یکسان باشند به آن «افزونه یکسان» و حالتی که اختلاط قطعات مجاز باشد را «افزونه غیر یکسان» گویند.
- در RAP اغلب فرض می‌شود که تعداد قطعات در دسترس نامحدودند، خرابی یک قطعه به سیستم صدمه نمی‌زند و قطعات قابل تعمیر نیستند.

۲-۲- بیان ریاضی مسئله RAP

نمادهای به کار گرفته‌شده برای بیان ریاضی مسائل RAP در زیر معرفی شده‌اند.

- i: اندیس زیرسیستم $i=\{1,2,\ldots,S\}$
- S: تعداد زیرسیستم‌های سیستم
- j: اندیس گزینه‌های موجود $j=\{1,2,\ldots,m_i\}$
- m_i: تعداد گزینه‌های موجود برای زیرسیستم iام
- t: زمان کارکرد سیستم
- x_{ij}: تعداد انتخاب شده از جزء jام برای زیرسیستم i
- w_{ij}: وزن گزینه jام مورد استفاده برای زیرسیستم i
- $c_{sw_{ij}}$: هزینه سوییچ مورد استفاده در زیرسیستم i برای سوییچ گزینه j
- $w_{sw_{ij}}$: وزن سوییچ مورد استفاده در زیرسیستم i برای سوییچ گزینه j
- C: حداکثر هزینه مجاز سیستم
- W: حداکثر وزن مجاز سیستم
- $Tr\ (x_{ij})$: اگر از سوییچ استفاده بشود برابر ۱ و در غیر این صورت برابر ۰ است.
- K_i: حداقل تعداد جزء سالم برای کارکرد موفق زیرسیستم iام
- R: قابلیت اطمینان
- n_{max_i}: حداکثر تعداد مجاز جزء قابل انتخاب برای زیرسیستم iام

وقتی هدف حداکثر کردن قابلیت اطمینان تحت دو محدودیت وزن و هزینه‌ی سیستم باشد و این دو محدودیت برحسب تعداد قطعات خطی باشند، بیان ریاضی مسائل RAP بر اساس [۳۴] به‌صورت زیر است:

$$\text{Max } R(t,x) \qquad (۱)$$

Sub. To:

$$\sum_{i=1}^{s}\sum_{j=1}^{m_i} c_{ij}x_{ij} + c_{sw_{ij}}Tr\ (x_{ij}) \le C \qquad (۲)$$

$$\sum_{i=1}^{s}\sum_{j=1}^{mi} w_{ij}x_{ij} + w_{sw_{ij}}Tr(x_{ij}) \le W \qquad (۳)$$

$$K_i \le \sum_{j=1}^{m_i} x_{ij} \le n_{max_i} \quad i=1,\ldots,S \qquad (۴)$$

$$x_{ij} \in \{0,1,\ldots,n_{max_i}\} \quad i=1,2,\ldots,S,\ j=1,\ldots,m_i \qquad (۵)$$

در رابطه (۱) R(t,x) بیانگر مقدار قابلیت اطمینان کل سیستم بوده و هدف حداکثر سازی آن است. این مقدار تحت تأثیر پیکربندی، سیاست افزونگی، تعداد افزونه‌ها و تابع توزیع شکست قطعات و زمان است. در این رابطه t زمان کارکرد سیستم و x بردار x_{ij} است که متغیرهای تصمیم و عدد صحیح می‌باشد. روابط (۲) و (۳) محدودیت بر روی هزینه و وزن را نشان می‌دهند. رابطه (۴) حداقل و حداکثر تعداد مجاز جزء برای هر زیرسیستم را بیان می‌کند.

وقتی‌که افزونه سرد مطرح نباشد و یا هزینه و وزن آن قابل صرف‌نظر کردن باشد، دو رابطه‌ی (۲) و (۳) ساده‌تر شده به‌صورت روابط (۶) و (۷) درمی‌آیند.

$$\sum_{i=1}^{s}\sum_{j=1}^{m_i} c_{ij}x_{ij} \le C \qquad (۶)$$

$$\sum_{i=1}^{s}\sum_{j=1}^{mi} w_{ij}x_{ij} \le W \qquad (۷)$$

در روابط (۶) و (۷) اگر تنها یک گزینه برای افزونه وجود داشته باشد، یعنی RAP از نوع تک گزینه‌ای باشد، روابط بازهم ساده‌تر شده و به‌صورت روابط (۸) و (۹) خواهند بود.

$$\sum_{i=1}^{s} n_i\, c_i \le C \qquad (۸)$$

$$\sum_{i=1}^{s} n_i\, w_i \le W \qquad (۹)$$

که در آن n_i تعداد جزء انتخاب‌شده برای زیرسیستم i است.

در ادامه برای هر مسئله آزمایشی برحسب شرایط آن، تابع هدف به‌صورت صریح تعیین و کل فرمولاسیون مشخص و الگوریتم پیشنهادی این پژوهش بر روی آن اجرا می‌شود.

۳- الگوریتم GW

الگوریتم پیشنهادی GW ترکیبی از دو روش حریصانه و روش وزن دهی هزینه‌ها است. روش حریصانه که عموماً در مسائل بهینه‌سازی کاربرد دارد، بر این منطق استوار است که در هر نقطه‌ی تصمیم، تصمیمی که بر مبنای معیاری معین بهترین یا بهینه‌ترین به نظر می‌رسد را بدون توجه به انتخاب‌هایی که قبلاً انجام‌شده یا در آینده انجام خواهد شد برمی‌گزینند، با این امید که با تکیه‌بر انتخاب‌های بهینه در هر مرحله، یک حل بهینه سرتاسری یافت شود[۳۵].

روش وزن دهی هزینه‌ها نیز از تکنیک‌های پرکاربرد در مسائل بهینه‌سازی است که معمولاً در شرایطی که مسئله‌ی بهینه‌سازی با بیش از یک محدودیت روبرو است، به کار گرفته می‌شود. مثلاً وقتی‌که مسئله‌ی بهینه‌سازی قابلیت اطمینان شامل دو محدودیت بر روی هزینه و وزن باشد، یک‌راه حل ساده استفاده از تکنیک وزن دهی به هزینه‌ها برای حل مسئله است [۳۶].

در الگوریتم‌های مبتنی بر روش حریصانه یک موضوع مهم تعداد و نوع گزینه‌هایی است که در هر مرحله تصمیم‌گیری بررسی می‌شوند. الگوریتم GW در هنگام تصمیم‌گیری در هر مرحله به شکلی عمل می‌کند که تصمیم جدید هم شامل اضافه کردن یک قطعه جدید و هم شامل جابجایی برخی از قطعات فعلی با قطعات دیگر است. در این صورت تعداد قطعات سیستم تغییری نمی‌کند. در حالتی که در مسئله اختلاط قطعات مجاز نباشد، یعنی مسئله از نوع افزونه یکسان باشد، جابجایی کل قطعات هر زیرسیستم با قطعات جدید بررسی می‌شود و در حالت نایکسان بودن افزونه‌ها جهت حفظ سرعت الگوریتم تنها اختلاط دو نوع قطعه بررسی می‌شود. ضمناً تحدب تابع هدف در مسائل RAP، نگرانی از افتادن سریع الگوریتم حریصانه در بهینه‌های محلی را مرتفع می‌نماید.

۳-۱- نقطه شروع الگوریتم GW

برای به‌کارگیری الگوریتم GW نیاز به یک نقطه شروع داریم. بدین منظور R_i را قابلیت اطمینان زیرسیستم i تعریف می‌کنیم که از رابطه (۱۰) به دست می‌آید.

$$R_i = 1 - \prod_{i=1}^{n_i} (1 - r_{ij})^{x_{ij}} \qquad (۱۰)$$

با در نظر گرفتن تعاریف قبلی، در رابطه (۱۰)، r_{ij} قابلیت اطمینان گزینه‌ی jام از زیرسیستم iام و n_i تعداد قطعات انتخاب‌شده برای زیرسیستم iام است. یک نقطه شروع از طریق تخصیص حداقل تعداد قطعه موردنیاز (K_i)، برای هر زیرسیستم به وجود می‌آید. بنابراین می‌توان در تقسیم‌بندی مسائل به تک گزینه‌ای و چندگزینه‌ای به شکل زیر عمل کرد.

- اگر مسئله تک گزینه‌ای باشد یعنی اگر تنها یک نوع قطعه برای افزونه وجود داشته باشد (m_i=۱)، نقطه شروع تنها یک حالت دارد و آن منظور کردن یک قطعه برای هر زیرسیستم است.
- اگر مسئله چندگزینه‌ای باشد، یعنی برای تمام یا بعضی از زیرسیستم‌ها m_i>۱ باشد، برای تعیین نقطه شروع، ابتدا فرض می‌شود که $\forall i\ m_i = 1$، و قابلیت اطمینان هر زیرسیستم i هم برابر یک است (R_i=۱)، که با این فرض قابلیت اطمینان کل سیستم نیز برابر یک خواهد بود (R_s=۱). در مرحله بعد برای به دست آوردن نقطه شروع الگوریتم GW با لحاظ کردن m_i>۱ با هر تخصیص جدید برای هر زیرسیستم، R_i و R_s را به‌روز کرده و در این تخصیص جدید حداکثر کردن R_s با کمترین مصرف منابع را هدف‌گذاری می‌کنیم.

با الهام از معیار تعریف‌شده توسط ناکاگاوا و ناکاشیما [۱۱]، هر تخصیصی که معیار رابطه (۱۱) را حداکثر کند به‌عنوان تخصیص جدید انتخاب می‌شود. این کار تا انجام تخصیص برای کلیه زیرسیستم‌ها ادامه می‌یابد. در این رابطه α و β ضرایب وزنی هزینه‌ها هستند.

$$S_i = \gamma R_s + \frac{(1-\gamma)}{K_i(\alpha c_{ij} + \beta w_{ij})} \qquad (۱۱)$$

با مقادیر مختلف γ چند نقطه شروع تولید می‌شود و الگوریتم برای هر نقطه اجرا می‌شود. مثلاً اگر $\gamma \in \{۰، ۰/۰۱، ۱/۲، ...، ۱\}$ باشد، ۱۱ نقطه شروع خواهیم داشت.

۳-۲- تشریح الگوریتم GW

فلوچارت الگوریتم GW در شکل (۱) نشان داده‌شده است. این الگوریتم برای حل مسائل قابلیت اطمینان از شاخص «نسبت مقدار افزایش قابلیت اطمینان سیستم به مقدار افزایش در هزینه‌ها» به‌عنوان معیار روش حریصانه استفاده می‌کند که این شاخص را GI می‌نامیم. بنابراین در هر نقطه تصمیم از بین کلیه تصمیمات تعریف‌شده توسط الگوریتم، آن تصمیمی که حداکثر مقدار شاخص GI را دارد به‌عنوان تصمیم جدید اتخاذ می‌شود. GI هم برای اضافه شدن یک قطعه به سیستم و هم جابجایی قطعات فعلی با قطعات جدید به‌گونه‌ای که ΔR_S را افزایش دهد محاسبه و تصمیمی که بالاترین مقدار شاخص را دارد به‌عنوان یک تصمیم جدید اتخاذ می‌نماید. این کار تا رسیدن به محدودیت‌های منابع ادامه می‌یابد.

اگر در مسئله‌ی بهینه‌سازی قابلیت اطمینان تنها یک محدودیت، مثلاً هزینه وجود داشته باشد، می‌توان شاخص حریصانه GI را به شکل رابطه (۱۲) نوشت، که در آن ΔR_S مقدار افزایش در قابلیت اطمینان سیستم و ΔC_S مقدار افزایش در هزینه است.

$$GI = \frac{\Delta R_s}{\Delta C_s} \qquad (۱۲)$$

زمانی که مسئله‌ی بهینه‌سازی قابلیت اطمینان شامل دو محدودیت، مثلاً وزن و هزینه، باشد شاخص GI با استفاده از روش وزن‌دهی هزینه‌ها به دست می‌آید. با تعریف α و β به‌عنوان ضرایب وزنی محدودیت‌ها، محاسبه شاخص GI به شکل رابطه (۱۳) خواهد بود.

$$GI = \frac{\Delta R_s}{\alpha \Delta C_s + \beta \Delta W_s} \qquad (۱۳)$$

که در آن مقادیر مختلف شاخص روش حریصانه (GI) با ضرایب وزنی (αوβ) متفاوت بین مقادیر مصرف این دو محدودیت محاسبه‌شده و بهترین آن‌ها به‌عنوان تصمیم جدید انتخاب می‌شود.

اما برای اجرای الگوریتم GW لازم است سازوکار معینی برای تعیین مقدار α و β ایجاد شود. یک روش ساده و متداول این است که ۰≤α≤۱ و β=۱-α تعریف شوند و مسئله را یک‌بار با مقدار α=۰ و β=۱ حل کرد. سپس مقدار کوچکی مثلاً g_a=۰/۱ به α اضافه کرده و با مقادیر جدید مثلاً α =۰/۱ و β=۰/۹، مجدد مسئله را حل

نمود. این کار را تا رسیدن مقدار α به ۱ تکرار کرده و بهترین جواب به‌عنوان جواب نهایی انتخاب شود. این روش گرچه مزایای خود را دارد اما وقتی شدت محدودیت‌ها بسیار متفاوت باشد ممکن است یک بهینه محلی را پیگیری کند و اگر بخواهیم دقت را زیاد کنیم تعداد تکرارها زیاد می‌شود.

بنابراین در الگوریتم پیشنهادی سازوکار دیگری استفاده شده است که گرچه برای α و β جداگانه اجرا می‌شود اما برای هر دو یکسان است. به همین دلیل فقط مکانیزم تغییر α به شرح زیر توضیح داده می‌شود.

اگر α^n ضریب وزنی محدودیت اول (هزینه) در حلقه تکرار nام، متغیرهای L^n_α و U^n_α به ترتیب حد پایین و حد بالای α^n، همچنین g^n_α را مقداری که در حلقه nام به α^n اضافه می‌شود، تعریف کنیم و α^n_{max} را ضریب وزنی بنامیم که در بازه‌ای که α^n تغییر می‌کند، بهترین جواب را تولید کرده است، آنگاه در اولین حلقه تکرار، حد پایین α^1، را $L^1_{\alpha=0}$ و حد بالای آن $U^1_{\alpha=1}$ در نظر گرفته و مقدار کوچکی برای g^1_α (مثلاً $g^1_{\alpha=0/2}$) در نظر می‌گیریم. ابتدا مسئله با $\alpha=0$ حل کرده و سپس در تکرارهای بعدی هر بار مقدار g^1_α به α اضافه می‌شود و حل مسئله تکرار می‌شود تا اینکه α^1 به حد بالای خود (U^1_α) برسد. در این حلقه‌ی تکرار بهترین جواب به‌عنوان S^* و مقدار α^1 متناظر آن به‌عنوان α^1_{max} حفظ می‌شود. در حلقه‌های تکرار بعدی (مثلاً n+1ام) مقادیر L^{n+1}_α و U^{n+1}_α و g^{n+1}_α و α به‌صورت دسته روابط (۱۴) بهنگام می‌شوند.

$$\begin{aligned} L^{n+1}_\alpha &= \alpha^n_{max} - g^n_\alpha \\ U^{n+1}_\alpha &= \alpha^n_{max} + g^n_\alpha \\ g^{n+1}_\alpha &= 0.2 * g^n_\alpha \\ \alpha^{n+1} &= L^{n+1}_\alpha \end{aligned} \qquad (۱۴)$$

با این مقادیر جدید، حلقه تکرار n+1ام برای حل مسئله اجرا می‌شود. نقطه توقف حلقه‌های تکرار وقتی است که g^n_α به‌دقت موردنظر برسد، که تحت عنوان $g_{\alpha min}$ بیان می‌شود.

مثال: فرض کنیم $g_{\alpha min}=0/01$ است. در اولین حلقه تکرار مقادیر $g^1_{\alpha=0/2}$، $U^1_{\alpha=1}$، $L^1_{\alpha=0}$، در نظر گرفته می‌شود. مسئله ۶ بار با مقادیر $\alpha=0، 0/2، 0/4، ...، 1$ حل‌شده و بهترین جواب با مقدار $\alpha^1=0/6$ به‌دست‌آمده است. بنابراین $\alpha^1_{max}=0/6$ می‌شود. اکنون با استفاده از دسته روابط (۱۴) برای حلقه تکرار دوم پارامترها به‌صورت زیر بهنگام می‌شوند:

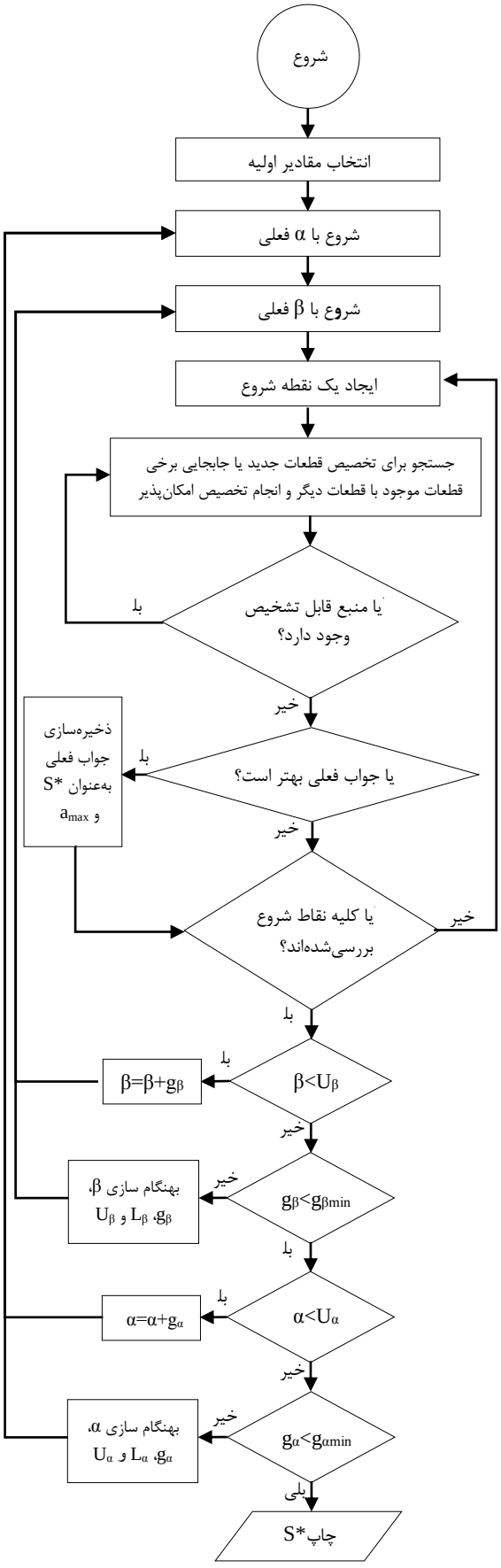
$L^2_\alpha = 0/6-0/2=0/4$

$U^2_\alpha = 0/6+0/2=0/8$

$g^2_\alpha = 0/2*0/2=0/04$

حلقه تکرار جدیدی برای مقادیر $\alpha \in \{0/4، 0/44، 0/48، ...، 0/8\}$ محاسبه و مجدداً بهترین جواب و مقادیر متناظر α^2 به‌عنوان α^2_{max} انتخاب و حلقه‌ی تکرار جدیدی اجرا می‌شوند. این کار تا وقتی g^n_α کمتر از ۰/۰۱ شود ادامه می‌یابد.

سازوکار مشابهی برای β نیز اجرا می‌شود که مستقل از α است. درواقع حلقه‌های تکرار β در دل حلقه‌های تکرار α قرار می‌گیرد.



شکل(۱): فلوچارت الگوریتم GW

در اجرای برنامه برای مسائل مختلف جواب‌های نهایی با روش

پیشنهادی معمولاً بهتر از روش اول بوده‌اند.

۴- اجرای الگوریتم GW بر روی مسائل آزمایشی

الگوریتم GW برای حل بسیاری از مسائل با تنوع در پیکربندی قابل به‌کارگیری است. بنابراین برای کارایی سنجی و ارزیابی آن، الگوریتم بر روی مسائل مختلف با تنوع زیاد که قبلاً به‌وسیله‌ی الگوریتم‌های دیگری حل‌شده، اجرا و نتایج آن با جواب‌های بهینه و یا بهترین جواب‌های موجود مقایسه شده است. در مواردی که گزارش زمان اجرای الگوریتم‌های دیگر موجود بوده‌اند زمان‌ها نیز مقایسه شده‌اند.

مسائل انتخاب‌شده شامل پیکربندی پل، یک پیکربندی خاص و چند پیکربندی سری-موازی با شرایط متفاوت می‌باشند و بنابراین تنوع زیادی دارند. ازلحاظ رابطه بین هزینه، وزن و قابلیت اطمینان چندین حالت مختلف در گزینه‌ها وجود دارد. مسائل هم از نوع یک گزینه‌ای و هم چندگزینه‌ای هستند و بعلاوه برخی مسائل از نوع «افزونه یکسان» و برخی «افزونه نایکسان» می‌باشند. همچنین در یک مسئله، نوع افزونه سرد و گرم لحاظ شده است.

الگوریتم GW با زبان برنامه‌نویسی C# کد شده و بر روی یک رایانه با پردازشگر intel i5 CPU 3.2 GHz با حافظه 2GB به اجرا درآمده است. درحالی‌که برای هر مسئله، الگوریتم‌هایی که GW با آن‌ها مقایسه شده‌اند اغلب به‌طور خاص برای آن مسئله طراحی شده و پارامترهای آن بهینه‌شده است اما GW با مقادیر ثابت برای پارامترها، به‌طور یکسان برای همه مسائل به اجرا درآمده است. بنابراین ممکن است با مقادیر دیگری برای پارامترها جواب‌های بهتری در زمان کوتاه‌تر نیز حاصل شود. مقادیر استفاده‌شده برای پارامترها به شرح ذیل می‌باشد.

$L^{1}_{\alpha}= L^{1}_{\beta}=1 \qquad U^{1}_{\alpha}= U^{1}_{\beta}=0$

$g^{1}_{\alpha}= g^{1}_{\beta}=0.2$

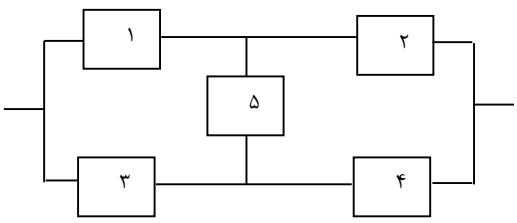
$g_{\alpha min} = g_{\beta min}= 0.01$

$g^{n+1}_{\alpha}= 0.2* g^{n}_{\alpha},\ g^{n+1}_{\beta}= 0.2 *g^{n}_{\beta}$

$\gamma \in \{0, 0.1, \ldots, 1\}$

۴-۱- مسئله آزمایشی P1

پیکربندی مسئله آزمایشی P1، یک سیستم پل مطابق شکل (۲) است که از پیکربندی‌های معروف در سیستم‌های پیچیده می‌باشد.



شکل (۲): یک چیدمان پل

اگر R_i قابلیت اطمینان زیرسیستم i و Q_i عدم اطمینان زیرسیستم i تعریف شود ($Q_i=1- R_i$)، در حالت کلی این پیکربندی را می‌توان به شکل دسته روابط (۱۵) تا (۱۹) فرموله کرد.

$$\text{Max } R_s =(1-Q_1Q_3)(1-Q_2Q_4)+Q_5(1-(1-R_1R_2)(1-R_3R_4)) \quad (15)$$

Sub. To:

$$\sum_{i=1}^{s}\sum_{j=1}^{m_i} c_{ij}x_{ij} \le C \quad (16)$$

$$\sum_{i=1}^{s}\sum_{j=1}^{mi} w_{ij}x_{ij} \le W \quad (17)$$

$$K_i \le \sum_{j=1}^{m_i} x_{ij} \le n_{max_i} \qquad i = 1 \ldots . S \quad (18)$$

$$x_{ij} \in \{0,1,2,\ldots, n_{max_i}\},\ i= 1,2,\ldots,S\ ,\ j=1,\ldots,m_i \quad (19)$$

جدول (۱) اطلاعات مسئله که تنها شامل محدودیت بر روی هزینه است و قطعات نیز تنها یک گزینه دارند (m_i=۱)، را نشان می‌دهد. بنابراین می‌توان از روابط (۸) و (۹) به‌جای روابط (۶) و (۷) استفاده کرد.

جدول (۱): اطلاعات مسئله p1

i	۱	۲	۳	۴	۵
R_{ij}	۰/۸۱۰۶	۰/۶۹۴۰	۰/۶۹۷۴	۰/۸۰۶۸	۰/۶۳۳۱
C_{ij}	۴۵	۱	۸	۵۶	۲۵
n_{maxi}	۴	۱۴۶	۱۹	۳	۵
C=۲۹۰					

برای اجرای الگوریتم به دلیل وجود تنها یک محدودیت، لزومی به استفاده از ضرایب مختلف نیست. ضمناً به دلیل نداشتن گزینه‌های مختلف برای هر زیرسیستم نقطه‌ی شروع به‌راحتی با تخصیص اولین قطعه ایجاد می‌شود.

جواب تولیدشده توسط الگوریتم GW برای P1 برابر R=۰/۹۹۹۵۱۳۵، در زمان T=۰/۰۰۰۱۵ ثانیه است. مقایسه نتایج به‌دست‌آمده توسط GW با الگوریتم‌های ارائه‌شده در پژوهش‌های قبلی در جدول (۲) منعکس شده است.

جدول (۲): مقایسه نتایج الگوریتم GW **و دیگران برای** P1

روش	R	T sec
GW	۰/۹۹۹۵۱۴	۰/۰۰۰۱۵*
[۱۴]	۰/۹۹۹۵۴۶	۰/۰۴۹۹۲**
[۱۴]	۰/۹۹۹۲۹۳	۰/۰۰۳۲۶**
[۱۲]	۰/۹۹۷۴۳۲	۰/۲۴۲۰۵**
[۱۱]	۰/۹۹۹۵۱۴	۰/۵۰۰
[۱۳]	۰/۹۹۹۵۴۶	۰/۲۶۷۸۴**

*پردازشگر PC ,intel i5, 3.2 GHz

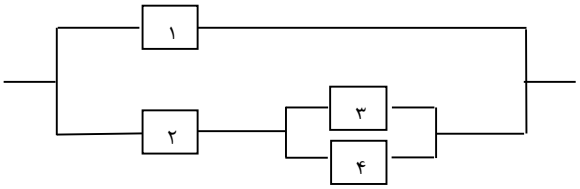
** پردازشگر PC, Pentium R.D 3.4 GHz

مشاهده می‌شود که جواب GW بسیار نزدیک به بهترین جواب است و برابر جواب [۱۱] و بهتر از جواب‌های دو روش دیگر است. در مورد زمان حل مسئله با توجه به تفاوت رایانه‌های

مورداستفاده امکان مقایسه وجود ندارد. بدین ترتیب با لحاظ نوع پردازشگر مورداستفاده برای سایر الگوریتم‌ها، زمان اجرای GW نیز ذکر شده که البته بسیار کمتر از سایر الگوریتم‌هاست.

۴-۲- مسئله آزمایشی P2 و P3

این دو مسئله دارای پیکربندی خاص مطابق شکل (۳) هستند.



شکل (۳): پیکربندی P2 و P4

این نوع پیکربندی را می‌توان به شکل دسته روابط (۲۰) تا (۲۴) فرموله کرد.

$$\text{Max } R_s=R_1+Q_1R_2R_4+Q_1R_2R_3Q_4 \quad (۲۰)$$

Sub. To:

$$\sum_{i=1}^{s}\sum_{j=1}^{m_i} c_{ij}x_{ij} \le C \quad (۲۱)$$

$$\sum_{i=1}^{s}\sum_{j=1}^{m_i} w_{ij}x_{ij} \le W \quad (۲۲)$$

$$K_i \le \sum_{j=1}^{m_i} x_{ij} \le n_{max_i} \qquad i = 1 \ldots . S \quad (۲۳)$$

$$x_{ij} \in \{0,1,2,\ldots, n_{max_i}\},\ i=1,2,\ldots S,\ j=1,2,\ldots, m_i \quad (۲۴)$$

جدول (۳) اطلاعات مسئله‌ی P2 و P3 را که از نوع تک گزینه‌ای بوده و دارای دو محدودیت می‌باشند را نشان می‌دهد. تفاوت دو مسئله فوق در نسبت هزینه و وزن گزینه‌های آن‌ها است. در P2 هزینه و وزن گزینه‌ها نزدیک یکدیگر هستند؛ اما در مسئله P3 در نسبت هزینه و وزن گزینه‌ها تفاوت زیادی وجود دارد.

جدول (۳): اطلاعات مسائل P2 و P3

	I	۱	۲	۳	۴
P2	r_i	۰/۶۹۸۴	۰/۶۲۵	۰/۸۴۶۴	۰/۷۵۳۶
	c_i	۲	۶۴	۳	۴
	w_i	۴۸	۷۴	۲۳	۷۴
	n_{maxi}	۶	۱	۱۳	۴
		C=۱۳۲	W=۳۴۱		
P3	r_i	۰/۸۰	۰/۷۵	۰/۷۰	۰/۶۵
	c_i	۶	۴	۳	۲
	w_i	۹	۴	۴	۳
		C=۳۰	W=۴۰		

جواب تولیدشده توسط GW برای P2 برابر R=۰/۹۸۹۶۱۱۲ و زمان تولید جواب ۰/۰۰۰۳۵ ثانیه است. این جواب برابر با جواب بهینه [۳۷] است. در پژوهش‌های دیگر این جواب توسط ۵ الگوریتم دیگر نیز تولید شده که اطلاعات آن در جدول (۴) نشان داده شده است. کمترین و بیشترین زمان تولید جواب به ترتیب ۰/۰۰۳۴۲ ثانیه و ۰/۹۰۴۸۶ ثانیه گزارش شده که زمان تولید جواب با استفاده از الگوریتم پیشنهادی این پژوهش با لحاظ نوع رایانه‌های مورد استفاده، بسیار کوتاه‌تر بوده است.

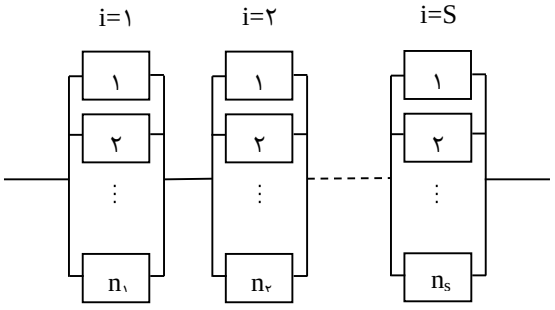
با اجرای GW برای حل مسئله P3، در مدت‌زمان ۰/۰۰۰۳۴ ثانیه جواب به‌دست‌آمده که جواب تولیدشده برابر R=۰/۹۹۷۴ است. این جواب برابر جواب بهینه [۳۸] است. همچنین بهتر از جواب [۱۲] بوده که مقداری برابر R=۰/۹۹۷۰ را به دست آورده است. در رابطه با مسئله P3 گزارشی از زمان اجرای الگوریتم‌های فوق دیده نشد.

جدول (۴): مقایسه نتایج الگوریتم GW و دیگران برای P2 و P3

	روش	R	T sec
P2	GW	۰/۹۸۹۶۱۱۲	۰/۰۰۰۳۵
	[۳۷]	۰/۹۸۹۶۱۱۲	۰/۰۰۴۱۳
	[۳۷]	۰/۹۸۹۶۱۱۲	۰/۰۰۳۴۲
	[۱۲]	۰/۹۸۹۶۱۱۲	۰/۰۰۲۶۴
	[۱۱]	۰/۹۸۹۶۱۱۲	۰/۰۱۹۶۳
	[۱۳]	۰/۹۸۹۶۱۱۲	۰/۹۰۴۸۶
P3	GW	۰/۹۹۷۴	۰/۰۰۰۳۴
	[۳۷]	۰/۹۹۷۴	-
	[۱۲]	۰/۹۹۷۰	-

۴-۳- مسئله آزمایشی P4

مسئله آزمایشی P4 که از معروف‌ترین پیکربندی‌ها می‌باشد شامل S زیرسیستم سری مطابق شکل (۴) است، که افزونه‌ها ازنظر مشخصات عملکردی یکسان هستند (m_i=1). بنابراین نوع مسئله RAP از نوع افزونه یکسان می‌باشد.



شکل (۴): پیکربندی سری-موازی

مسائلی با این نوع پیکربندی را می‌توان به شکل دسته روابط (۲۵) تا (۲۹) فرموله کرد.

$$Max\ R_s = \prod_{i=1}^{s}(1-(1-r_i)^{n_i}) \quad (۲۵)$$

Sub. To:

$$\sum_{i=1}^{s} n_i c_i \le C \qquad (۲۶)$$

$$\sum_{i=1}^{s} n_i w_i \le W \qquad (۲۷)$$

$$K_i \le \sum_{j=1}^{m_i} x_{ij} \le n_{max_i} \quad i = 1 \ldots . S \qquad (۲۸)$$

$$x_{ij} \in \{0,1,2,\ldots, n_{max_i}\},\ i= 1,2,\ldots,S,\ j=1,\ldots,m_i \qquad (۲۹)$$

اطلاعات مسئله در جدول (۵) ارائه شده است.

الگوریتم GW جوابی برابر R=۰/۹۴۵ در مدت‌زمان ۰/۰۰۰۱۰۱ ثانیه تولید کرد که دقیقاً برابر جوابی است که در [۳۹] با استفاده از دو روش متفاوت بر پایه الگوریتم ژنتیک در مدت‌زمان ۵/۰۴ و ۳/۷۵ ثانیه با اجرا بر روی یک رایانه شخصی پنتیوم با پردازشگر 800MHz ارائه شده است. مشاهده می‌شود با لحاظ نوع رایانه مورد استفاده، GW جواب برابر در زمان هزاران بار کوتاه‌تر تولید کرده است. در این مثال تفاوت فاحش سرعت دو الگوریتم، کارایی شاخص حریصانه ابتکاری به کار گرفته شده در الگوریتم GW را با وضوح بیشتری نشان می‌دهد، هرچند که در این رابطه تفاوت مکانیزم نقطه توقف دو الگوریتم نیز به‌شدت تأثیرگذار است.

۴-۴- مسئله آزمایشی P5

مسئله آزمایشی P5 همانند P4 دارای S زیرسیستم سری است. با این تفاوت که امکان انتخاب‌های چندگانه برای هر زیرسیستم وجود دارد ($m_i \ge 1$) و افزونه‌ها غیر یکسان هستند. بنابراین می‌توان مسئله با این پیکربندی را به شکل دسته روابط (۳۰) تا (۳۴) فرموله کرد.

$$\text{Max } R_s = \prod_{i=1}^{s}\left(1 - \prod_{i=1}^{m_i}(1 - r_{ij})^{x_{ij}}\right) \qquad (۳۰)$$

Sub. To:

$$\sum_{i=1}^{s}\sum_{i=1}^{m_i} c_{ij}x_{ij} \le C \qquad (۳۱)$$

$$\sum_{i=1}^{s}\sum_{i=1}^{m_i} w_{ij}x_{ij} \le W \qquad (۳۲)$$

$$K_i \le \sum_{j=1}^{m_i} x_{ij} \le n_{max_i} \quad i = 1 \ldots . S \qquad (۳۳)$$

$$x_{ij} \in \{0,1,2,\ldots, n_{max_i}\},\ i= 1,2,\ldots,S,\ j= 1,\ldots,m_i \qquad (۳۴)$$

معروف‌ترین مسئله سری- موازی مسئله‌ای است که فیف و همکاران [۹]، در سال ۱۹۶۸ ایجاد کردند و بعداً ناکاگاوا و میازاکی [۱۰]، آن را توسعه دادند. این مسئله شامل ۱۴ زیرسیستم با پیکربندی سری است که هر زیرسیستم دارای ۳ یا ۴ گزینه است و برای هر زیرسیستم i، تعداد افزونه مجاز برابر ۸ و حداقل تعداد قطعه سالم لازم برای کارکرد موفق زیرسیستم iام برابر یک ($\forall i\ K_{i=1}$ و $n_{maxi=8}$) می‌باشد.

جدول (۵): اطلاعات مسئله آزمایشی P4

J	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴	۱۵
R_j	۰/۹۰	۰/۷۵	۰/۶۵	۰/۸۰	۰/۸۵	۰/۹۳	۰/۷۸	۰/۶۶	۰/۷۸	۰/۹۱	۰/۷۹	۰/۷۷	۰/۶۷	۰/۷۹	۰/۶۷
C_j	۵	۴	۹	۷	۷	۵	۶	۹	۴	۵	۶	۷	۹	۸	۶
W_j	۸	۹	۶	۷	۸	۸	۹	۶	۷	۸	۹	۷	۶	۵	۷

C=۴۰۰ W=۴۱۴

جدول (۶): اطلاعات مسئله آزمایشی P5

Subsystem	Component 1			Component 2			Component 3			Component 4		
i	r	c	w	r	c	w	r	c	w	r	c	w
۱	۰/۹۰	۱	۳	۰/۹۳	۱	۴	۰/۹۱	۲	۲	۰/۹۵	۲	۵
۲	۰/۹۵	۲	۸	۰/۹۴	۱	۱۰	۰/۹۳	۱	۹			
۳	۰/۸۵	۲	۷	۰/۹۰	۳	۵	۰/۸۷	۱	۶	۰/۹۲	۴	۴
۴	۰/۸۳	۳	۵	۰/۸۷	۴	۶	۰/۸۵	۵	۴			
۵	۰/۹۴	۲	۴	۰/۹۳	۲	۳	۰/۹۵	۳	۵			
۶	۰/۹۹	۳	۵	۰/۹۸	۳	۴	۰/۹۷	۲	۵	۰/۹۶	۲	۴
۷	۰/۹۱	۴	۷	۰/۹۲	۴	۸	۰/۹۴	۵	۹			
۸	۰/۸۱	۳	۴	۰/۹۰	۵	۷	۰/۹۱	۶	۶			
۹	۰/۹۷	۲	۸	۰/۹۹	۳	۹	۰/۹۶	۴	۷	۰/۹۱	۳	۸
۱۰	۰/۸۳	۴	۶	۰/۸۵	۴	۵	۰/۹۰	۵	۶			
۱۱	۰/۹۴	۳	۵	۰/۹۵	۴	۶	۰/۹۶	۵	۶			
۱۲	۰/۷۹	۲	۴	۰/۸۲	۳	۵	۰/۸۵	۴	۶	۰/۹۰	۵	۷
۱۳	۰/۹۸	۲	۵	۰/۹۹	۳	۵	۰/۹۷	۲	۶			
۱۴	۰/۹۰	۴	۶	۰/۹۲	۲	۷	۰/۹۵	۵	۶	۰/۹۹	۶	۹

جدول (۷): نتایج اجرای الگوریتم GW برای P5

	W	R	R_{GW}	$R_{Max-Min}$	R_{mvo2}	T_{GW}	$T_{Max-Min}$	T_{MVO2}
۱	۱۹۱	۰/۹۸۶۸	۰/۹۸۶۴	۰/۹۸۴۸	۰/۹۸۴۳	۰/۵۰۱۳	۴۰۹/۵	۲
۲	۱۹۰	۰/۹۸۶۴	۰/۹۸۵۵	۰/۹۸۴۳	۰/۹۸۴۶	۰/۵۰۳۴	۳۱۵/۵	۲/۶
۳	۱۸۹	۰/۹۸۵۹	۰/۹۸۴۵	۰/۹۸۴۰	۰/۹۸۴۵	۰/۴۷۱۲	۲۹۹/۵	۲/۲
۴	۱۸۸	۰/۹۸۵۴	۰/۹۸۴۴	۰/۹۸۲۹	۰/۹۸۴۳	۰/۴۹۸۲	۶۳۲/۱	۲/۱
۵	۱۸۷	۰/۹۸۴۷	۰/۹۸۳۸	۰/۹۸۳۹	۰/۹۸۳۷	۰/۴۹۱۶	۱۲	۲
۶	۱۸۶	۰/۹۸۴۲	۰/۹۸۳۵	۰/۹۸۲۴	۰/۹۸۳۷	۰/۴۵۶۲	۴۲۶/۳	۱/۸
۷	۱۸۵	۰/۹۸۳۵	۰/۹۸۲۵	۰/۹۸۲۴	۰/۹۸۳۵	۰/۳۳۰۰	۱۵۱	۱/۸
۸	۱۸۴	۰/۹۸۳۰	۰/۹۸۱۹	۰/۹۷۹۹	۰/۹۸۲۷	۰/۴۸۳۹	۴۹	۱/۹
۹	۱۸۳	۰/۹۸۲۳	۰/۹۸۱۲	۰/۹۸۱۱	۰/۹۸۲۲	۰/۴۷۷۷	۲۱۲/۵	۱/۷
۱۰	۱۸۲	۰/۹۸۱۵	۰/۹۸۰۹	۰/۹۷۹۵	۰/۹۸۰۱	۰/۳۳۰۲	۱۵۲/۵	۱/۸
۱۱	۱۸۱	۰/۹۸۱۰	۰/۹۸۰۴	۰/۹۸۰۰	۰/۹۸۰۰	۰/۳۱۷۰	۱۳۸/۸	۱/۸
۱۲	۱۸۰	۰/۹۸۰۳	۰/۹۷۹۷	۰/۹۷۷۶	۰/۹۷۸۲	۰/۳۲۱۵	۱۵۷/۴	۱/۶
۱۳	۱۷۹	۰/۹۷۹۵	۰/۹۷۸۵	۰/۹۷۸۰	۰/۹۷۸	۰/۴۶۹۲	۲۴۸/۷	۱/۵
۱۴	۱۷۸	۰/۹۷۸۴	۰/۹۷۷۸	۰/۹۷۷۲	۰/۹۷۷۱	۰/۳۰۸۸	۱۲۶/۱	۱/۳
۱۵	۱۷۷	۰/۹۷۷۶	۰/۹۷۷۲	۰/۹۷۷۳	۰/۹۷۶۷	۰/۳۰۹۹	۷۹	۱/۳
۱۶	۱۷۶	۰/۹۷۶۷	۰/۹۷۶۴	۰/۹۷۶۵	۰/۹۷۶۵	۰/۳۱۱۸	۳۸	۱/۲
۱۷	۱۷۵	۰/۹۷۵۷	۰/۹۷۴۱	۰/۹۷۳۴	۰/۹۷۵۳	۰/۳۰۵۷	۵۵/۵	۱/۲
۱۸	۱۷۴	۰/۹۷۴۹	۰/۹۷۳۵	۰/۹۷۴۵	۰/۹۷۳۲	۰/۳۰۳۳	۱۳/۵	۱/۱
۱۹	۱۷۳	۰/۹۷۳۸	۰/۹۷۲۹	۰/۹۷۲۸	۰/۹۷۲۷	۰/۳۰۵۷	۵/۷	۱
۲۰	۱۷۲	۰/۹۷۳۰	۰/۹۷۱۷	۰/۹۶۹۶	۰/۹۷۲۵	۰/۴۴۸۷	۱/۷	۱
۲۱	۱۷۱	۰/۹۷۱۹	۰/۹۷۱۱	۰/۹۶۹۴	۰/۹۷۱۴	۰/۲۹۵۷	۱/۱	۱
۲۲	۱۷۰	۰/۹۷۰۸	۰/۹۷۰۳	۰/۹۶۸۸	۰/۹۶۹	۰/۲۹۳۷	۳/۳	۰/۹
۲۳	۱۶۹	۰/۹۶۹۳	۰/۹۶۹۲	۰/۹۶۶۴	۰/۹۶۷۶	۰/۲۹۷۶	۱۴/۳	۰/۷
۲۴	۱۶۸	۰/۹۶۸۱	۰/۹۶۸۱	۰/۹۶۵۲	۰/۹۶۶۹	۰/۴۳۵۱	۲/۳	۰/۷
۲۵	۱۶۷	۰/۹۶۶۳	۰/۹۶۵۳	۰/۹۶۴۵	۰/۹۶۶۳	۰/۲۸۶۲	۱/۳	۰/۶
۲۶	۱۶۶	۰/۹۶۵۰	۰/۹۶۴۶	۰/۹۶۴۰	۰/۹۶۴	۰/۲۸۴۸	۱/۵	۰/۶
۲۷	۱۶۵	۰/۹۶۳۷	۰/۹۶۳۵	۰/۹۶۰۷	۰/۹۶۱۳	۰/۲۸۹۱	۱/۹	۰/۵
۲۸	۱۶۴	۰/۹۶۲۴	۰/۹۶۱۸	۰/۹۵۹۵	۰/۹۵۸۹	۰/۲۸۱۴	۱/۱	۰/۵
۲۹	۱۶۳	۰/۹۶۰۶	۰/۹۵۹۴	۰/۹۵۸۸	۰/۹۶۰۶	۰/۲۸۱۴	۰/۹	۰/۵
۳۰	۱۶۲	۰/۹۵۹۲	۰/۹۵۹۲	۰/۹۵۸۳	۰/۹۵۸۳	۰/۲۷۹۲	۰/۷	۰/۴
۳۱	۱۶۱	۰/۹۵۸۰	۰/۹۵۷۱	۰/۹۵۶۲	۰/۹۵۶۲	۰/۲۷۷۳	۰/۹	۰/۴
۳۲	۱۶۰	۰/۹۵۵۷	۰/۹۵۵۴	۰/۹۵۴۸	۰/۹۵۲۹	۰/۲۶۹۳	۰/۷	۰/۴
۳۳	۱۵۹	۰/۹۵۴۶	۰/۹۵۴۶	۰/۹۵۲۷	۰/۹۵۲۷	۰/۲۶۷۲	۰/۷	۰/۷
					AVE	۰/۳۵۷۰۶۹۷	۱۰۷/۷۲۷۲۷	۱/۲۲۴۲۴۲۴

اطلاعات P5 در جدول (۶) آورده شده است. با ثابت نگه‌داشتن حد مجاز هزینه برابر ۱۳۰ واحد (C=130) و تغییر حد مجاز وزن (W) از ۱۹۱ واحد تا ۱۵۹ مجموعاً ۳۳ مسئله تولید می‌شود.

نتایج اجرای الگوریتم GW برای P5 و مقایسه آن با سایر الگوریتم‌ها در جدول (۷) نشان داده‌شده است. بااینکه مسئله تاکنون توسط تعداد زیادی از محققین حل‌شده، اما در تعداد معدودی از آن‌ها زمان اجرا برنامه گزارش‌شده است. لذا انجام مقایسه نتایج GW با نتایج همین موارد صورت گرفته است. چنانکه مشاهده می‌شود GW جواب‌های بسیار نزدیک به جواب بهینه تولید کرده است. بهترین جواب را الگوریتم ILP [۴۰]، تولید کرده و کلیه جواب‌های آن با جواب بهینه برابر است. جواب‌های GW بهتر از الگوریتم MWO2 [۳] و بسیار بهتر از الگوریتم Max-Min [۴۱] است. از ازنظر زمان اجرا، الگوریتم GW با لحاظ نوع پردازشگر کمترین و الگوریتم Max-Min بیشترین زمان اجرا را دارد.

۴-۵- مسئله آزمایشی P8,P7,P6

مسئله آزمایشی P5 توجه زیادی را جلب و بسیاری از مطالعات بر روی آن انجام‌شده است؛ اما دارای برخی مشکلات است. به همین دلیل کویت و کوناک سه دسته مسئله‌ی سری- موازی دیگر مطابق جدول (۸) طراحی کردند [۳] که اولاً در آن S=20 و mi=4،

همچنین nmaxi=6 می‌باشد. ثانیاً در هر مسئله نسبت هزینه، وزن و قابلیت اطمینان قطعات متفاوت است، در P6 ترتیب هزینه، وزن و قابلیت اطمینان گزینه‌ها صعودی است. در P7 ترتیب هزینه‌ی گزینه‌ها صعودی، ترتیب وزن آن‌ها نزولی و قابلیت اطمینان تقریباً برابر است. در P8، گزینه ۱ و ۲ دارای هزینه، وزن و قابلیت اطمینان کم هستند، درحالی‌که گزینه‌های ۳ و ۴ دارای هزینه، وزن و قابلیت بالا هستند. ثالثاً، دامنه محدودیت‌ها بر روی منابع هزینه و وزن (W&C) متفاوت است و هرکدام از ۱۰۰ تا ۲۵۰ واحد تغییر می‌کنند، به‌طوری‌که هر مسئله دارای ۳۶ حالت مختلف است. بنابراین این سه مسئله جمعاً ۱۰۸ حالت را شامل می‌شوند. بیلیونت [۴۰] و کویت و کوناک [۳] مسائل را حل نموده‌اند که مقایسه بین جواب‌های آنان با الگوریتم GW در جدول (۹) آورده شده است.

جدول (۸): اطلاعات مسئله آزمایشی P6، P7، P8

	P6			P7			P8		
	ril,…,ri4	wil,…,wi4	cil,…,ci4	ril,…,ri4	wil,…,wi4	cil,…,ci4	ril,…,ri4	wil,…,wi4	cil,…,ci4
۱	۰/۷۱ ،۰/۸۲ ،۰/۹ ،۰/۹۹	۳،۵،۸،۱۰	۳،۵،۶،۱۰	۰/۹۲ ،۰/۹ ،۰/۹۱ ،۰/۹۲	۹،۷،۴،۱	۳،۴،۷،۱۰	۰/۸۱ ،۰/۸ ،۰/۹۶ ،۰/۹۸	۴،۳،۱۰،۷	۳،۴،۷،۱۰
۲	۰/۷ ،۰/۸۲ ،۰/۹۲ ،۰/۹۸	۴،۶،۸،۱۰	۲،۵،۷،۱۰	۰/۹۲ ،۰/۹ ،۰/۹۲ ،۰/۹۱	۱۰،۶،۵،۳	۳،۵،۷،۸	۰/۸۲ ،۰/۸۲ ،۰/۹۷ ،۰/۹۸	۴،۲،۱۰،۷	۳،۵،۷،۸
۳	۰/۷ ،۰/۸۳ ،۰/۹۱ ،۰/۹۷	۲،۶،۸،۱۰	۳،۵،۷،۱۰	۰/۹۱ ،۰/۹۳ ،۰/۹۳ ،۰/۹۲	۹،۶،۴،۱	۱،۵،۷،۹	۰/۸۲ ،۰/۸ ،۰/۹۶ ،۰/۹۷	۴،۱،۱۰،۷	۱،۵،۶،۹
۴	۰/۷۲ ،۰/۸۲ ،۰/۹۳ ،۰/۹۷	۲،۶،۷،۹	۴،۶،۸،۹	۰/۹۳ ،۰/۹۳ ،۰/۹۲ ،۰/۹	۹،۷،۵،۳	۱،۵،۷،۹	۰/۸۳ ،۰/۸ ،۰/۹۶ ،۰/۹۶	۵،۲،۸،۷	۲،۵،۶،۹
۵	۰/۷۳ ،۰/۸۱ ،۰/۹۲ ،۰/۹۷	۲،۵،۷،۱۰	۲،۵،۷،۹	۰/۹۳ ،۰/۹۱ ،۰/۹ ،۰/۹۱	۱۰،۷،۴،۲	۱،۵،۶،۹	۰/۸۲ ،۰/۸۱ ،۰/۹۶ ،۰/۹۸	۵،۲،۱۰،۷	۳،۵،۷،۸

ادامه‌ی جدول (۸): اطلاعات مسئله آزمایشی P6، P7، P8

P8			P7			P6			
ril,…,ri4	wil,…,wi4	cil,…,ci4	ril,…,ri4	wil,…,wi4	cil,…,ci4	ril,…,ri4	wil,…,wi4	cil,…,ci4	
۰/۸ ،۰/۸۲ ،۰/۹۸ ،۰/۹۸	۵،۳،۱۰،۶	۱،۴،۷،۱۰	۰/۹۱ ،۰/۹ ،۰/۹ ،۰/۹۲	۸،۷،۵،۱	۲،۴،۷،۹	۰/۷۱ ،۰/۸۱ ،۰/۹۲ ،۰/۹۹	۳،۵،۸،۱۰	۳،۵،۷،۹	۶
۰/۸ ،۰/۸۳ ،۰/۹۸ ،۰/۹۸	۵،۲،۹،۷	۱،۴،۶،۸	۰/۹۳ ،۰/۹ ،۰/۹ ،۰/۹	۱۰،۷،۴،۳	۱،۵،۶،۹	۰/۷۳ ،۰/۸۳ ،۰/۹۱ ،۰/۹۸	۴،۵،۷،۱۰	۲،۶،۸،۹	۷
۰/۸۳ ،۰/۸۳ ،۰/۹۸ ،۰/۹۸	۴،۱،۹،۷	۳،۴،۶،۹	۰/۹۲ ،۰/۹۲ ،۰/۹۱ ،۰/۹۲	۸،۷،۵،۳	۲،۵،۷،۹	۰/۷۳ ،۰/۸ ،۰/۹۱ ،۰/۹۸	۴،۵،۸،۱۰	۴،۵،۷،۱۰	۸
۰/۸۳ ،۰/۸۲ ،۰/۹۷ ،۰/۹۶	۵،۳،۹،۶	۲،۴،۶،۱۰	۰/۹ ،۰/۹۱ ،۰/۹۳ ،۰/۹	۸،۷،۵،۲	۱،۴،۶،۸	۰/۷۲ ،۰/۸۲ ،۰/۹۱ ،۰/۹۹	۴،۶،۸،۱۰	۴،۶،۷،۹	۹
۰/۸۱ ،۰/۸۱ ،۰/۹۶ ،۰/۹۸	۴،۳،۹،۷	۲،۴،۶،۹	۰/۹۱ ،۰/۹۲ ،۰/۹۱ ،۰/۹۳	۱۰،۷،۵،۱	۱،۵،۷،۹	۰/۷۲ ،۰/۸۳ ،۰/۹۲ ،۰/۹۹	۲،۶،۷،۹	۳،۶،۸،۱۰	۱۰
۰/۸۲ ،۰/۸۱ ،۰/۹۷ ،۰/۹۸	۵،۱،۱۰،۶	۱،۵،۶،۸	۰/۹۱ ،۰/۹ ،۰/۹۳ ،۰/۹۳	۱۰،۶،۴،۲	۱،۴،۶،۹	۰/۷۱ ،۰/۸۳ ،۰/۹۳ ،۰/۹۷	۲،۵،۷،۱۰	۴،۵،۷،۹	۱۱

ادامه‌ی جدول (۸): اطلاعات مسئله آزمایشی P6، P7، P8

P8			P7			P6			
ril,…,ri4	wil,…,wi4	cil,…,ci4	ril,…,ri4	wil,…,wi4	cil,…,ci4	ril,…,ri4	wil,…,wi4	cil,…,ci4	
۰/۸۳ ,۰/۸۳ ,۰/۹۸ ,۰/۹۷	۵،۳۰۹،۶	۲،۵۰۶،۸	۰/۹ ,۰/۹۳ ,۰/۹ ,۰/۹	۹،۶۰۴،۱	۱،۵۰۷،۸	۰/۷۱ ,۰/۸۱ ,۰/۹۱ ,۰/۹۷	۳،۶۰۸،۹	۲،۵۰۸،۹	۱۲
۰/۸۱ ,۰/۸۱ ,۰/۹۶ ,۰/۹۸	۴،۳۰۸،۶	۱،۵۰۶،۸	۰/۹۳ ,۰/۹ ,۰/۹۱ ,۰/۹۲	۹،۶۰۴،۱	۱،۴۰۶،۱۰	۰/۷۲ ,۰/۸۳ ,۰/۹۱ ,۰/۹۸	۲،۵۰۷،۱۰	۲،۶۰۸،۹	۱۳
۰/۸۱ ,۰/۸ ,۰/۹۷ ,۰/۹۶	۵،۳۰۸،۶	۳،۵۰۶،۹	۰/۹۳ ,۰/۹۲ ,۰/۹۱ ,۰/۹۱	۱۰،۶۰۵،۱	۲،۵۰۶،۹	۰/۷۳ ,۰/۸۳ ,۰/۹ ,۰/۹۸	۴،۶۰۸،۱۰	۲،۶۰۷،۱۰	۱۴
۰/۸ ,۰/۸۲ ,۰/۹۸ ,۰/۹۸	۵،۱۰۸،۷	۲،۴۰۶،۱۰	۰/۹۳ ,۰/۹۳ ,۰/۹۱ ,۰/۹	۸،۷۰۵،۳	۲،۴۰۶،۹	۰/۷۳ ,۰/۸۳ ,۰/۹۲ ,۰/۹۸	۲،۵۰۸،۱۰	۳،۶۰۸،۹	۱۵
۰/۸۲ ,۰/۸۲ ,۰/۹۸ ,۰/۹۸	۴،۲۰۹،۷	۱،۵۰۶،۸	۰/۹۱ ,۰/۹ ,۰/۹۳ ,۰/۹۱	۹،۶۰۴،۳	۲،۴۰۶،۱۰	۰/۷۱ ,۰/۸۳ ,۰/۹۲ ,۰/۹۸	۴،۵۰۸،۱۰	۴،۶۰۷،۱۰	۱۶
۰/۸ ,۰/۸۳ ,۰/۹۷ ,۰/۹۶	۵،۲۰۱۰،۷	۱،۴۰۷،۸	۰/۹۲ ,۰/۹۲ ,۰/۹ ,۰/۹۲	۱۰،۶۰۴،۳	۲،۵۰۷،۹	۰/۷ ,۰/۸ ,۰/۹۲ ,۰/۹۷	۲،۶۰۷،۹	۳،۵۰۷،۱۰	۱۷

ادامه‌ی جدول (۸): اطلاعات مسئله آزمایشی P6، P7، P8

P8			P7			P6		
ril,…,ri4	wil,…,wi4	cil,…,ci4	ril,…,ri4	wil,…,wi4	cil,…,ci4	ril,…,ri4	wil,…,wi4	cil,…,ci4	
۰/۸ ,۰/۸ ,۰/۹۷ ,۰/۹۶	۴،۲،۹،۷	۳،۵،۶،۱۰	۰/۹۱ ,۰/۹۲ ,۰/۹ ,۰/۹۱	۸،۷،۵،۲	۱،۵،۶،۹	۰/۷۲ ,۰/۸ ,۰/۹۳ ,۰/۹۹	۳،۵،۸،۹	۲،۶،۷،۱۰	۱۸
۰/۸۱ ,۰/۸۳ ,۰/۹۸ ,۰/۹۸	۴،۲،۸،۶	۲،۴،۷،۹	۰/۹۲ ,۰/۹ ,۰/۹ ,۰/۹۳	۸،۶،۴،۳	۳،۵،۷،۸	۰/۷۱ ,۰/۸ ,۰/۹۳ ,۰/۹۷	۴،۵،۸،۱۰	۲،۶،۸،۹	۱۹
۰/۸ ,۰/۸ ,۰/۹۸ ,۰/۹۸	۵،۳،۹،۶	۱،۴،۶،۹	۰/۹۱ ,۰/۹۱ ,۰/۹ ,۰/۹۳	۹،۷،۵،۲	۱،۵،۶،۸	۰/۷ ,۰/۸۳ ,۰/۹ ,۰/۹۹	۳،۵،۸،۹	۴،۶،۸،۱۰	۲۰

جدول (۹): مقایسه نتایج الگوریتم GW با روش‌های دیگران برای P6، P7 و P8

P8					P7					P6						
T_{ILP}	T_{GW}	R_{MWO}	R_{IL}	R_G	T_{ILP}	T_G	R_{MWO}	R_{IL}	R_G	T_{ILP}	T_G	R_{MWO}	R_{IL}	R_G	W	C
۳۰۰	۱/۰۲	۰/۱۴۷۲۲	۰/۱۵۰۸۵	۰/۱۷۸۳۲۸	۰/۱۲	۰/۰۳۳۶	۰/۱۷۲۶۵	۰/۱۷۲۶۵	۰/۱۷۴۵۶۷	۱۰۱/۸۹	۱/۰۲۶	۰/۰۵۲۶۸	۰/۰۶۲۶۹	۰/۰۵۹۵۳	۱۰۰	۱۰۰	۱
۳۰۰	۱/۳۲۲	۰/۳۲۸۴	۰/۳۲۶۰۷	۰/۳۴۹۶۶۸	۳/۸۵	۰/۳۹۳۴	۰/۳۵۰۳۵	۰/۳۵۲۸۷	۰/۲۵۲۸۶۹	۱۱۹/۳۳	۰/۹۴۱	۰/۰۷۷۰۲	۰/۰۸۵۹۲	۰/۰۸۲۸۱۷	۱۳۰	۱۰۰	۲
۱۱/۳۳	۱/۴۶۵	۰/۵۵۵۰۳	۰/۵۶۴۶	۰/۵۶۷۰۸	۳۰۰	۰/۴۴۷۷	۰/۳۱۶۵۹	۰/۳۱۸۴۸	۰/۳۱۸۴۷۸	۲۶/۸	۰/۹۴۲	۰/۰۷۷۰۲	۰/۰۸۵۹۲	۰/۰۸۲۸۱۷	۱۶۰	۱۰۰	۳
۱/۲۳	۱/۹۵۵	۰/۶۴۷۷۳	۰/۶۵۳۷۷	۰/۶۵۸۰۲۶	۳۰۰	۰/۷۶۲۵	۰/۳۹۴۲۹	۰/۳۹۸۳۳	۰/۳۹۴۰۵	۲۶/۹۲	۰/۹۴۱	۰/۰۷۷۰۲	۰/۰۸۵۹۲	۰/۰۸۲۸۱۷	۱۹۰	۱۰۰	۴
۱/۵۴	۱/۶۴۱	۰/۷۲۹۸۷	۰/۷۳۳۴۸	۰/۷۲۸۳۹۸	۳۰۰	۰/۶۳۸۵	۰/۴۸۲۰۹	۰/۴۸۱۵۶	۰/۴۸۳۰۶۸	۲۶/۸۳	۰/۹۴۲	۰/۰۷۷۰۲	۰/۰۸۵۹۲	۰/۰۸۲۸۱۷	۲۲۰	۱۰۰	۵

ادامه جدول (۹): مقایسه نتایج الگوریتم GW با روش‌های دیگران برای P6، P7 و P8

P8					P7					P6						
T_{ILP}	T_{ILP}	T_{ILP}	T_{ILP}	T_{ILP}	T_{ILP}	T_{GW}	R_{MWO2}	R_{ILP}	R_{GW}	T_{ILP}	T_{GW}	R_{MWO2}	R_{ILP}	R_{GW}	W	C	
۰/۴۹	۱/۴۸۴	۰/۷۶۴۲۶	۰/۷۸۴۱۴	۰/۷۸۶۶۹۱	۳۰۰	۰/۹۸۴۵	۰/۵۸۶۵۵	۰/۵۸۸۷۵	۰/۵۸۸۷۴۹	۲۶/۸۱	۰/۹۴۲	۰/۰۷۷۰۲	۰/۰۸۵۹۲	۰/۰۸۲۸۱۷	۲۵۰	۱۰۰	۶
۳۰۰	۱/۱۲۷	۰/۳۱۳۵۵	۰/۳۱۳۱۱	۰/۳۳۷۶۰۴	۸/۰۴	۰/۳۵۴۲	۰/۲۴۱۲۹	۰/۲۴۳۸۸	۰/۳۴۱۶۹۶	۱۵/۱۷	۰/۷۸۰	۰/۰۸۰۹۱	۰/۰۸۰۹۱	۰/۰۷۹۹۷۵	۱۰۰	۱۳۰	۷
۱۲/۲۰	۱/۶۲۱	۰/۵۶۰۷۸	۰/۵۶۶۲۷	۰/۵۶۲۵۶۷	۱۴/۳۳	۰/۶۲۷۶	۰/۳۱۵۹۳	۰/۳۲۲۴۲	۰/۳۱۴۵۷۸	۱۲/۷۷	۱/۹۸۳	۰/۲۶۸۸۴	۰/۲۷۲۸	۰/۲۶۱۴۹۷	۱۳۰	۱۳۰	۸
۰/۶۸	۲/۱۶۱	۰/۶۶۵۹۴	۰/۶۷۰۸۷	۰/۶۶۵۱۷۶	۱۷/۱۸	۰/۷۷	۰/۳۹۴۷۶	۰/۴۰۵۴۳	۰/۳۹۲۴۹۱	۲/۵۳	۱/۱۱۳	۰/۳۱۵۰۶	۰/۳۳۵۲۴	۰/۳۳۵۲۳۹	۱۶۰	۱۳۰	۹
۰/۳۰	۲/۷۲	۰/۷۴۹۶۱	۰/۷۵۹۹۲	۰/۷۵۱۶۱۱	۲۴/۶۴	۰/۶۷۴۳	۰/۴۸۲۷۷	۰/۵۰۳۳	۰/۴۸۵۳۳	۱/۸۴	۱/۱۱۰	۰/۳۱۵۰۶	۰/۳۳۵۲۴	۰/۳۳۵۲۳۹	۱۹۰	۱۳۰	۱۰
۸/۴۴	۲/۵۷۲	۰/۸۳۴۲۱	۰/۸۳۸۳۶	۰/۸۴۴۷۳۷	۳۳/۶۲	۱/۲۷۵	۰/۵۹۰۴۱	۰/۶۱۴۵۲	۰/۶۰۵۰۱۹	۱/۸۴	۱/۱۰۸	۰/۳۱۵۰۶	۰/۳۳۵۲۴	۰/۳۳۵۲۳۹	۲۲۰	۱۳۰	۱۱
۰/۴۳	۲/۶۸۵	۰/۸۹۱۲۳	۰/۸۹۲۷۴	۰/۸۹۴۱۹۰	۳۰۰	۱/۴۹۰۸	۰/۷۱۹۷۱	۰/۷۱۸۶۳	۰/۷۲۱۰۲۹	۱/۸۴	۱/۱۰۸	۰/۳۱۵۰۶	۰/۳۳۵۲۴	۰/۳۳۵۲۳۹	۲۵۰	۱۳۰	۱۲
۱/۸۸	۱/۲۰۳	۰/۵۳۰۳۳	۰/۵۳۴۹۸	۰/۵۲۹۹۴۷	۴۳/۲۴	۰/۴۰۳۵	۰/۳۰۳۸۹	۰/۳۰۹۵۴	۰/۳۰۶۲۰۹	۱۵/۲۱	۰/۷۷۵	۰/۰۸۰۹۱	۰/۰۸۰۹۱	۰/۰۷۹۹۷۵	۱۰۰	۱۶۰	۱۳
۴/۵۲	۱/۹۵۵	۰/۶۴۸۰۲	۰/۶۵۴۲۷	۰/۶۴۵۸۹۷	۲/۹۵	۰/۷۵۲	۰/۳۸۹۶۱	۰/۴۰۶۱۲	۰/۳۹۸۷۳	۶/۳۲	۱/۴۲۰	۰/۲۸۵۹۵	۰/۲۸۸۲۴	۰/۲۸۲۵۱۷	۱۳۰	۱۶۰	۱۴
۴/۴۱	۲/۳۱۴	۰/۷۴۸۸۴	۰/۷۵۷۲۵	۰/۷۴۹۱۴۲	۱۳/۶۴	۱/۰۴۲۹	۰/۴۸۲۹۷	۰/۵۰۶۳۲	۰/۵۰۷۲۹	۳۰۰	۲/۹۸۷	۰/۵۲۹۴۱	۰/۵۵۱۸۳	۰/۵۴۳۸۰۴	۱۶۰	۱۶۰	۱۵
۷۷/۲۲	۲/۴۹۲	۰/۸۲۹۲۲	۰/۸۴۰۰۱	۰/۸۳۸۰۴۷	۳۰۰	۱/۱۸۷	۰/۵۹۰۴۷	۰/۶۱۵۵۳	۰/۶۱۶۵۲	۰/۵۶	۱/۵۸۹	۰/۶۵۷۶۲	۰/۶۵۷۶۲	۰/۶۴۸۰۶۱	۱۹۰	۱۶۰	۱۶

ادامه جدول (۹): مقایسه نتایج الگوریتم GW با روش‌های دیگران برای P6، P7 و P8

P8					P7					P6						
T_{ILP}	T_{ILP}	T_{ILP}	T_{ILP}	T_{ILP}	T_{ILP}	T_{GW}	R_{MWO2}	R_{ILP}	R_{GW}	T_{ILP}	T_{GW}	R_{MWO2}	R_{ILP}	R_{GW}	W	C
۱/۲۴	۲/۹۱۸	۰/۸۹۷۵۱	۰/۸۹۷۶۹۰	۳۰۰	۱/۵۸۷۱	۰/۷۵۴۷۳	۰/۷۵۴۷۳	۰/۷۵۸۷۴	۰/۷۴۹۷۹۲	۰/۳۵	۱/۵۷۳	۰/۶۵۷۶۲	۰/۶۵۷۶۲	۰/۶۴۸۰۶۱	۲۲۰	۱۶۰	۱۷
۳/۰۰	۳/۱۳۳	۰/۹۲۴۱۶	۰/۹۲۸۲۰۵	۰/۸۶	۱/۸۳۵	۰/۸۸۱۷۷	۰/۸۸۱۷۷	۰/۸۸۳۰۲	۰/۸۸۲۴۹	۰/۳۶	۱/۵۷۶	۰/۶۵۷۶۲	۰/۶۵۷۶۲	۰/۶۴۸۰۶۱	۲۵۰	۱۶۰	۱۸
۲/۸۳	۱/۵۳۴	۰/۶۰۳۶۴	۰/۶۰۵۴۶۴	۱۳۰/۹۵	۰/۶۳۶۹	۰/۳۸۰۶۷	۰/۳۸۰۶۷	۰/۳۸۲۱۷	۰/۳۸۱۵۸۸	۱۵/۱۸	۰/۷۷۶	۰/۰۸۰۹۱	۰/۰۸۰۹۱	۰/۰۷۹۹۷۵	۱۰۰	۱۹۰	۱۹
۳/۴۱	۱/۳۰۹	۰/۷۲۴۲۵	۰/۷۲۵۱۹۲	۱/۹۴	۰/۹۹	۰/۴۸۰۷۴	۰/۴۸۰۷۴	۰/۵۰۱۶۸	۰/۴۹۸۹۱	۶/۳۳	۱/۴۱۹	۰/۲۸۵۹۵	۰/۲۸۸۲۴	۰/۲۸۲۵۱۷	۱۳۰	۱۹۰	۲۰
۱/۰۰	۲/۵۵۶	۰/۸۲۶۹۷	۰/۸۲۸۱۰۷	۱۰/۴۹	۱/۲۷۱۲	۰/۶۱۹۹۵	۰/۶۱۹۹۵	۰/۶۲۱۲۶	۰/۶۱۹۲۶	۱۴/۰۳	۲/۱۶۲	۰/۶۰۳۷۱	۰/۶۱۵۹۲	۰/۵۹۵۲۸۲	۱۶۰	۱۹۰	۲۱
۱/۰۶	۲/۹۱۳	۰/۸۹۷۱۸	۰/۸۹۸۳۶	۳۰۰	۱/۴۳۳۲	۰/۷۵۸۰۹	۰/۷۵۸۰۹	۰/۷۶۰۴۶	۰/۷۵۹۳۹	۱/۳۱	۴/۱۲۵	۰/۷۵۸۷۴	۰/۷۶۸۶۶	۰/۷۵۹۴۵	۱۹۰	۱۹۰	۲۲
۵/۰۰	۳/۴۴۷	۰/۹۲۶۷۵	۰/۹۲۳۹۱۰	۰/۸۴	۱/۷۱۶۹	۰/۸۸۲۹۳	۰/۸۸۲۹۳	۰/۸۸۴	۰/۸۸۲۵۸	۰/۳۱	۳/۱۴۷	۰/۷۹۸۴۷	۰/۸۲۰۴۹	۰/۸۱۷۲۶۴	۲۲۰	۱۹۰	۲۳
۱/۱۰	۳/۹۳۸	۰/۹۴۷۲۵	۰/۹۴۹۰۳۴	۰/۷۵	۲/۱۵۷۲	۰/۹۰۷۸۳	۰/۹۰۷۸۳	۰/۹۰۸۴۷	۰/۹۰۵۹۷۴	۰/۳۷	۳/۱۴۷	۰/۸۰۰۹۷	۰/۸۲۰۴۹	۰/۸۱۷۲۶۴	۲۵۰	۱۹۰	۲۴
۰/۳۱	۰/۹۸۴	۰/۶۳۹۰۹	۰/۶۵۹۲۳۱	۲۸/۸۵	۰/۷۴۶۳	۰/۴۶۵۷۹	۰/۴۶۵۷۹	۰/۴۶۹۸۹	۰/۴۶۷۹	۱۵/۱۳	۰/۷۸۵	۰/۸۰۸۰۹۱	۰/۸۰۸۰۹۱	۰/۰۷۹۹۷۵	۱۰۰	۲۲۰	۲۵
۳/۰۲	۲/۶۰۳	۰/۷۹۸۸۲	۰/۷۹۹۴۶۳	۳/۱۷	۱/۱۱۳۹	۰/۵۸۵۸۵	۰/۵۸۵۸۵	۰/۶۰۷۲۶	۰/۵۸۶۶۷۸	۶/۳۲	۱/۴۲۰	۰/۲۸۵۹۵	۰/۲۸۸۲۴	۰/۲۸۲۵۱۷	۱۳۰	۲۲۰	۲۶
۰/۴۶	۲/۹۸۷	۰/۸۸۷۸۸	۰/۸۸۸۴۷۵	۳۰۰	۱/۴۷۵۱	۰/۷۲۷۸۵	۰/۷۲۷۸۵	۰/۷۵۴۷۴	۰/۷۵۲۹۹	۱۹/۹۶	۲/۱۶۳	۰/۶۰۳۷۱	۰/۶۱۵۹۳	۰/۵۹۵۲۸۲	۱۶۰	۲۲۰	۲۷
۰/۳۱	۳/۳۲۴	۰/۹۲۲۳۹	۰/۹۲۲۱۷۵	۱/۰۶	۱/۶۵۹۷	۰/۸۸۲۱۴	۰/۸۸۲۱۴	۰/۸۸۲۱۴	۰/۸۸۰۷۲	۰/۵۸	۲/۹۹۰	۰/۷۹۸۲۴	۰/۸۰۸۰۷	۰/۷۹۳۸۶۷	۱۹۰	۲۲۰	۲۸
۲۱/۳۱	۳/۸۹۵	۰/۹۴۶۷۸	۰/۹۴۶۴۵۴	۱۰/۲۶	۲/۱۶۹۸	۰/۹۰۶۱۶	۰/۹۰۶۱۶	۰/۹۰۶۵۳	۰/۹۰۵۲۵	۱/۱۶	۴/۰۳۷	۰/۸۷۲۸۱	۰/۸۷۵۸۳	۰/۸۷۲۴۳۶	۲۲۰	۲۲۰	۲۹

ادامه جدول (۹): مقایسه نتایج الگوریتم GW با روش‌های دیگران برای P6، P7 و P8

P8					P7					P6						
T_{ILP}	T_{ILP}	T_{ILP}	T_{ILP}	T_{ILP}	R_{GW}	R_{ILP}	R_{MWO2}	T_{GW}	T_{ILP}	R_{GW}	R_{ILP}	R_{MWO2}	T_{GW}	T_{ILP}	W	C	
۲/۰۹	۴/۵۲۲	۰/۹۶۷۷۹	۰/۹۶۸۰۵	۰/۹۶۶۹۴۷	۰/۹۲۲۹۰۹	۰/۹۲۵۲۵	۰/۹۲۳۰۲	۲/۴۹۸۸	۳۰۰	۰/۸۹۶۶۱۷	۰/۹۰۰۰۶	۰/۸۹۴۵۴	۳/۱۵۲	۰/۳۲	۲۵۰	۲۲۰	۳۰
۰/۳۱	۰/۹۸۲	۰/۶۳۹۰۹	۰/۶۶۵۹۲	۰/۶۵۹۲۳۱	۰/۵۷۳۵۱	۰/۵۷۵۰۷	۰/۵۴۸۵۷	۰/۸۶۰۶	۱/۰۴	۰/۰۷۹۹۷۵	۰/۰۸۰۹۱	۰/۰۸۰۹۱	۰/۷۷۷	۱۵/۱۵	۱۰۰	۲۵۰	۳۱
۰/۹۶	۲/۹۴	۰/۸۴۷۴۴	۰/۸۵۷۹۳	۰/۸۵۳۰۳۲	۰/۷۱۱۱۲۲	۰/۷۱۱۲۹	۰/۷۱۱۷۲	۱/۲۱۷۸	۳۰۰	۰/۲۸۲۵۱۷	۰/۲۸۸۲۴	۰/۲۸۵۹۵	۱/۴۲۰	۶/۳۲	۱۳۰	۲۵۰	۳۲
۰/۹۶	۳/۳۴۶	۰/۹۱۳۴۵	۰/۹۱۴۸۱	۰/۹۱۲۴۵۷	۰/۸۷۳۱۲	۰/۸۷۶۲۹	۰/۸۷۴۴۴	۱/۵۸۴	۰/۳۴	۰/۵۹۵۲۸۲	۰/۶۱۵۹۲	۰/۶۰۳۷۱	۲/۱۶۱	۱۹/۸۹	۱۶۰	۲۵۰	۳۳
۰/۵۲	۳/۶۷۸	۰/۹۴۲۸۵	۰/۹۴۴۰۶	۰/۹۴۱۷۸۸	۰/۹۰۲۰۷۴	۰/۹۰۳۸۹	۰/۹۰۱۷۱	۱/۷۳۴۴	۴/۲۱	۰/۷۹۳۸۶۷	۰/۸۰۸۰۷	۰/۷۹۸۲۴	۲/۹۸۸	۰/۵۸	۱۹۰	۲۵۰	۳۴
۱/۹۰	۴/۴۱۲	۰/۹۶۴۰۱	۰/۹۶۵۸۵	۰/۹۶۵۹۶۰	۰/۹۲۲۰۳	۰/۹۲۳۶۴	۰/۹۲۳۸۸	۲/۷۷۷۹	۶۴/۳۱	۰/۸۸۷۲۴۷	۰/۸۹۰۶۱	۰/۸۸۵۸۱	۳/۱۳۱	۰/۹۸	۲۲۰	۲۵۰	۳۵
۳/۱۸	۴/۹۶۱	۰/۹۸۰۳۱	۰/۹۸۰۰۷	۰/۹۷۹۸۹۳	۰/۹۳۹۵۶	۰/۹۴۰۹۷	۰/۹۴۰۸	۲/۵۵۱۹	۲۷۷/۳۳	۰/۹۳۰۲	۰/۹۳۰۶۹	۰/۹۲۹۵۷	۵/۳۲۵	۰/۶	۲۵۰	۲۵۰	۳۶
۲۹/۹۵۰	۲/۵۳							۱/۲۳۸	۱۱۰/۶۹			Avr	۲/۱۲۳	۷۸/۹۹۶			

این مقایسه نشان می‌دهد که جواب‌های GW برای P6، P7 و P8 در ۱۶ مورد بهتر از بهترین جواب‌هایی است که توسط بیلیونت با الگوریتم ILP به‌دست‌آمده، در ۱۰ مورد مساوی آن‌ها و در سایر موارد نیز بسیار نزدیک به آن‌ها است. با در نظر گرفتن نوع رایانه‌های مورد استفاده متوسط زمان صرف شده برای اجرای GW بر روی ۱۰۸ حالت برابر ۱/۸۷ ثانیه و متوسط زمان اجرای الگوریتم ILP، ۷۳/۶۹ ثانیه بوده است. مقایسه نتایج جواب‌های GW با روش کویت و کوناک نشان می‌دهد که جواب‌های GW در ۷۲ مورد بهتر از روش MWO2 ارائه‌شده توسط آن‌ها بوده است و بنابراین کاملاً بر روش MWO2 برتری دارد. زمان اجرای این روش گزارش نشده است.

از بین این سه مسئله، مسئله‌ی P7 مشکل‌ترین مسئله برای حل است. بیلیونت برای نشان دادن این پیچیدگی، این مسئله را بدون محدودیت بر روی وزن حل و نتایج و زمان آن را گزارش کرده است. نتایج انجام کار مشابهی با استفاده از الگوریتم GW و مقایسه آن با نتایج بیلیونت [۴۰] در جدول (۱۰) آمده است.

مشاهده می‌شود که در همه حالت‌ها الگوریتم GW ازنظر کیفیت جواب تولیدشده بر الگوریتم ILP که توسط بیلیونت ارائه گردیده، برتری دارد. زمان نیز با لحاظ نوع رایانه‌های مورد استفاده، کاهش قابل توجهی داشته است.

جدول (۱۰): مقایسه نتایج GW با الگوریتم ILP برای مسئله P7

I	C	R_{LIP}	R_{GW}	T_{ILP}	T_{GW}
۹	۱۰۰	۰/۹۹۳۲۷۴	۰/۹۹۳۳۰۸۵۰	۱/۶۲	۰/۰۰۶
۲	۱۳۰	۰/۹۹۹۴۰۶	۰/۹۹۹۴۰۹۷۰	۱/۱۲	۰/۰۰۸
۳	۱۶۰	۰/۹۹۹۹۴۶	۰/۹۹۹۹۴۸۵۱	۱/۵۲	۰/۰۰۷
۴	۱۹۰	۰/۹۹۹۹۹۵	۰/۹۹۹۹۹۵۱۷	۱/۱۲	۰/۰۰۸
۵	۲۲۰	۰/۹۹۹۹۹۸	۰/۹۹۹۹۹۹۵۴	۱/۶۳	۰/۰۱۶
۶	۲۵۰	۰/۹۹۹۹۹۷	۰/۹۹۹۹۹۹۹۴	۱/۶۱	۰/۰۱۶

جدول (۱۱): اطاعات مسئله آزمایشی P9

Subsystem			Component choice 1			Component choice 2			Component choice 3			Component choice 4		
i	Ki	Type	λ_{ij}	C_{ij}	W_{ij}	λ_{ij}	C_i	W_{ij}	λ_{ij}	C_i	W_{ij}	λ_{ij}	C_{ij}	W_{ij}
۱	۱	A	۰/۰۰۱۰۵۴	۱	۳	۰/۰۰۰۷۲۶	۱	۴	۰/۰۰۰۹۴۳	۲	۲	۰/۰۰۰۵۱۳	۲	۵
۲	۲	A	۰/۰۰۰۵۱۳	۲	۸	۰/۰۰۰۶۱۹	۱	۱۰	۰/۰۰۰۷۲۶	۱	۹	–		
۳	۱	A	۰/۰۰۱۶۲۵	۲	۷	۰/۰۰۱۰۵۴	۳	۵	۰/۰۰۱۳۹۳	۱	۶	۰/۰۰۰۸۳۴	۴	۴
۴	۲	A	۰/۰۰۱۸۶۳	۳	۵	۰/۰۰۱۳۹۳	۴	۶	۰/۰۰۱۶۲۵	۵	۴	–		
۵	۱	A	۰/۰۰۰۶۱۹	۲	۴	۰/۰۰۰۷۲۶	۲	۳	۰/۰۰۰۵۱۳	۳	۵	–		
۶	۲	A	۰/۰۰۰۱۰۱	۳	۵	۰/۰۰۰۲۰۲	۳	۴	۰/۰۰۰۳۰۵	۲	۵	۰/۰۰۰۴۰۸	۲	۴
۷	۱	A	۰/۰۰۰۹۴۳	۴	۷	۰/۰۰۰۸۳۴	۴	۸	۰/۰۰۰۶۱۹	۵	۹	–		
۸	۲	S	۰/۰۰۲۱۰۷	۳	۴	۰/۰۰۱۰۵۴	۵	۷	۰/۰۰۰۹۴۳	۶	۶	–		
۹	۳	S	۰/۰۰۰۳۰۵	۲	۸	۰/۰۰۰۱۰۱	۳	۹	۰/۰۰۰۴۰۸	۴	۷	۰/۰۰۰۹۴۳	۳	۸
۱۰	۳	S	۰/۰۰۱۸۶۳	۴	۶	۰/۰۰۱۶۲۵	۴	۵	۰/۰۰۱۰۵۴	۵	۶	–		
۱۱	۳	S	۰/۰۰۶۱۹	۳	۵	۰/۰۰۰۵۱۳	۴	۶	۰/۰۰۰۴۰۸	۵	۶	–		
۱۲	۱	S	۰/۰۰۲۳۵۷	۲	۴	۰/۰۰۱۹۸۵	۳	۵	۰/۰۰۱۶۲۵	۴	۶	۰/۰۰۱۰۵۴	۵	۷
۱۳	۲	S	۰/۰۰۲۰۲	۲	۵	۰/۰۰۰۱۰۱	۳	۵	۰/۰۰۰۳۰۵	۲	۶	–		
۱۴	۳	S	۰/۰۰۱۰۵۴	۴	۶	۰/۰۰۰۸۳۴	۴	۷	۰/۰۰۰۵۱۳	۵	۶	۰/۰۰۰۱۰۱	۶	۹

۴-۶- مسئله آزمایشی P9

اطلاعات عددی P9 برای مسئله‌ای که ساختار کلی آن مشابه P5 است در جدول (۱۱) نشان داده شده است. در مسائل قبلی برای کلیه زیرسیستم‌ها استراتژی افزونگی فعال در نظر گرفته‌شده است. همچنین برای کلیه زیرسیستم‌ها $K_{i=1}$ بود. در مسئله P9 برای زیرسیستم‌های مختلف استراتژی‌های مختلف اما از پیش تعیین‌شده وجود دارد و بعلاوه K_i مقادیر متفاوتی دارد. بنابراین مسئله P9 به شرایط واقعی نزدیک‌تر است.

جدول (۱۲): مقایسه جواب الگوریتم GW و جواب بهینه W=۱۷۰ و C=۱۳۰

i	الگوریتم LP		الگوریتم GW	
	z_i	n_i	z_i	λ_i
۱	۳	۲	۳	۲
۲	۱	۲	۱	۲
۳	۴	۱	۴	۱
۴	۳	۳	۳	۳
۵	۲	۱	۲	۱
۶	۲	۲	۲	۲
۷	۲	۱	۲	۱
۸	۱	۳	۱	۳
۹	۳	۳	۳	۳
۱۰	۲	۴	۲	۴
۱۱	۱	۴	۱	۴
۱۲	۱	۲	۱	۲
۱۳	۲	۲	۲	۲
۱۴	۳	۴	۳	۴

وقتی‌که هزینه و وزن و خطای سوییچینگ ناچیز و مسئله از نوع افزونه یکسان باشد، با در نظر گرفتن A به‌عنوان مجموعه زیرسیستم‌ها با استراتژی سرد، S به عنوان مجموعه زیرسیستم‌های با استراتژی فعال، t زمان کارکرد سیستم و z_i به‌عنوان شماره گزینه انتخاب‌شده برای زیرسیستم i، می‌توان این مسئله را به شکل دسته روابط (۳۵) تا (۳۹) فرمول‌بندی کرد که البته تابع توزیع شکست، نمایی بوده و در آن λ_{ij} نرخ شکست در واحد زمان است [۴۲]. اگر t =۱۰۰ در نظر گرفته شود، قابلیت اطمینان هر قطعه مشابه P5 خواهد شد. کویت و لیو یک حالت خاص مسئله را (W=۱۷۰، C=۱۳۰) با برنامه‌ریزی خطی (LP) حل کرده‌اند و جواب بهینه‌ی R=۰/۴۴۶۶ را به‌دست آورده‌اند [۴۲]. آنها در پژوهش خود بیان کردند که روش ارائه‌شده عملاً برای مسائل بزرگ با تعداد متغیرها زیاد کارا نیست. ضمن اینکه گزارشی هم از زمان اجرای برنامه ارائه نکردند.

$$Max.R(t) = \prod_{i\in A}\sum_{l=k_i}^{n_i}\binom{n_i}{l}\left(\exp(-\lambda_{i,zi}t)\right)^l \times (1 - exp(-\lambda_{i,zi}\,t))^{n_i-l} \times \prod_{i\in S} exp(-\lambda_{i,zi}\,k_i t)\sum_{l=0}^{n_i-k_i}\frac{(-\lambda_{i,zi}\,k_i t)^l}{l!} \quad (۳۵)$$

Sub.To:

$$\sum_{i=1}^{s}\sum_{i=1}^{m_i} c_{ij}x_{ij} \le C \quad (۳۶)$$

$$\sum_{i=1}^{s}\sum_{i=1}^{m_i} w_{ij}x_{ij} \le W \quad (۳۷)$$

$$K_i \le \sum_{i=1}^{m_i} x_{ij} \le n_{max_i} \qquad i = 1 \ldots . S \quad (۳۸)$$

$$x_{ij} \in \{0,1,2,\ldots, n_{max_i}\},\ i= 1,2,\ldots,S\ ,\ j= 1,\ldots,m_i \quad (۳۹)$$

الگوریتم GW برای این مسئله نیز اجرا شده است. جواب GW برابر با جواب بهینه R_S=۰/۴۴۶۶ و C_S=۱۱۶ و W_S=۱۷۰ و در مدت‌زمان ۰/۰۱۲ ثانیه به‌دست آمده است. جدول (۱۲) نتایج دو الگوریتم را نشان می‌دهد. ترکیب همه زیرسیستم‌ها در هر دو الگوریتم دقیقاً برابر است.

با استفاده از الگوریتم GW مسائل در کل حالت‌ها حل شده‌اند که جواب‌های به‌دست‌آمده در جدول (۱۳) آمده است.

جدول (۱۳): نتایج الگوریتم GW برای مسئله P9

	W	R	T
۱	۱۹۱	۰/۶۴۱۶۵۰	۰/۰۲۴۵
۲	۱۹۰	۰/۶۲۸۸۱۹	۰/۰۲۳۷۵
۳	۱۸۹	۰/۶۲۸۸۱۹	۰/۰۲۱۵
۴	۱۸۸	۰/۶۱۴۲۹۴	۰/۰۲۱۲۵
۵	۱۸۷	۰/۶۰۳۹۲۵	۰/۰۲۰۷۵
۶	۱۸۶	۰/۵۹۱۸۴۸	۰/۰۲
۷	۱۸۵	۰/۵۹۱۸۴۸	۰/۰۱۹۲۵
۸	۱۸۴	۰/۵۷۹۲۵۹	۰/۰۱۹۵
۹	۱۸۳	۰/۵۷۲۹۸۰	۰/۰۱۸۵
۱۰	۱۸۲	۰/۵۶۰۲۴۵	۰/۰۱۸
۱۱	۱۸۱	۰/۵۴۳۸۵۴	۰/۰۱۷
۱۲	۱۸۰	۰/۵۳۸۰۶۶	۰/۰۱۶۷۵
۱۳	۱۷۹	۰/۵۲۷۳۰۶	۰/۰۱۶
۱۴	۱۷۸	۰/۵۲۷۳۰۶	۰/۰۱۵۷۵
۱۵	۱۷۷	۰/۵۱۶۰۹۰	۰/۰۱۵۵
۱۶	۱۷۶	۰/۵۱۰۴۹۵	۰/۰۱۴۵
۱۷	۱۷۵	۰/۴۸۸۲۳۹	۰/۰۱۴۲۵
۱۸	۱۷۴	۰/۴۸۸۲۳۹	۰/۰۱۴۵
۱۹	۱۷۳	۰/۴۷۷۸۵۴	۰/۰۱۳۲۵
۲۰	۱۷۲	۰/۴۷۲۶۷۳	۰/۰۱۳
۲۱	۱۷۱	۰/۴۵۶۲۸۶	۰/۰۱۲۵
۲۲	۱۷۰	۰/۴۴۶۵۸۱	۰/۰۱۲
۲۳	۱۶۹	۰/۴۴۱۷۳۹	۰/۰۱۱۵
۲۴	۱۶۸	۰/۴۲۳۱۲۰	۰/۰۱۱۷۵
۲۵	۱۶۷	۰/۴۱۴۱۲۰	۰/۰۱۰۷۵
۲۶	۱۶۶	۰/۴۰۹۶۳۱	۰/۰۱۰۲۵
۲۷	۱۶۵	۰/۳۹۵۴۲۹	۰/۰۰۹۷۵
۲۸	۱۶۴	۰/۳۸۷۰۱۸	۰/۰۰۹۷۵
۲۹	۱۶۳	۰/۳۸۲۸۲۳	۰/۰۰۸۵
۳۰	۱۶۲	۰/۳۴۹۲۶۲	۰/۰۰۸۵
۳۱	۱۶۱	۰/۳۴۵۴۷۶	۰/۰۰۷۷۵
۳۲	۱۶۰	۰/۳۳۳۴۹۹	۰/۰۰۷۲۵
۳۳	۱۵۹	۰/۳۲۶۴۰۵	۰/۰۰۶۷۵

۵- نتیجه‌گیری

در این مقاله یک الگوریتم جدید، تحت عنوان GW، برای حل مسائل RAP ارائه شد. در این الگوریتم تلفیق یک روش جدید برای وزن دهی به هزینه‌ها و یک شاخص حریصانه ابتکاری جدید برای انتخاب‌ها در هر دور چرخه تکرار الگوریتم، باعث می‌گردد جواب الگوریتم در هر دور چرخه الگوریتم دائماً بهتر گردد. محدودیت منابع مسئله هم، تعیین‌کننده‌ی نقطه توقف الگوریتم است. پیامد این سازوکارها سرعت و دقت بالای الگوریتم در حل مسائل متنوع است. ارزیابی الگوریتم ارائه‌شده از طریق اجرای آن بر روی ۹ مسئله آزمایشی با مشخصات و پیکربندی‌های بسیار متفاوت انجام گرفت. نتایج نشان می‌دهد این الگوریتم بدون اعمال هیچ‌گونه تغییر ساختاری یا پارامتری، قادر به حل انواع مسائل RAP با پیکربندی‌های گوناگون است، به‌نحوی‌که جواب‌های حاصل از آن برابر یا بسیار نزدیک و به جواب بهینه و یا بهترین جواب‌های موجود بوده و حتی در مواردی بهتر از بهترین جواب موجود می‌باشد. در مجموع، انعطاف‌پذیری، کارایی و منطق ساده و ملموس این الگوریتم، همچنین قدرت آن برای حل انواع مسائل RAP مزیت آن را به بسیاری از دیگر الگوریتم‌ها نشان می‌دهد.

مراجع

[1] Kuo, W., Prasad, V.R., (2000). "An annotated overview of system-reliability optimization", IEEE Transactions on Reliability, 49(2): 176-187.

[2] Chern, M.S., (1992). "On the computational complexity of reliability redundancy allocation in a series system", Operations Research Letters, 11(5): 309-315.

[3] Coit, D.W., Konak, A., (2006). "Multiple Weighted Objectives Heuristic for the Redundancy Allocation Problem", IEEE Transactions on Reliability, 55(3): 551-558.

[4] Hsieh, Y.C., (2003). "A linear approximation for redundant reliability problems with multiple component choices", Computers & Industrial Engineering, 44(1): 91-103.

[5] Bulfin, R.L. and C.Y. Liu. (1985). "Optimal Allocation of Redundant Components for Large Systems", IEEE Transactions on Reliability, 34(3): 241-247.

[6] Misra, K.B., Sharma, U., (1991). "An efficient algorithm to solve integer-programming problems arising in system-reliability design", IEEE Transactions on Reliability, 40(1): 81-91.

[7] Onishi, J., Kimura, S., *James*, R.J.W., Nakagawa, T., (2007). "Solving the Redundancy Allocation Problem with a Mix of Components Using the Improved Surrogate Constraint Method", IEEE Transactions on Reliability, 56(1): 94-101.

[8] Bellman, R., Dreyfus, S., (1958). "Dynamic Programming and the Reliability of Multicomponent Devices", Operations Research, 6(2): 200-206.

[9] Fyffe, D.E., Hines, W.W., Lee, N.K., (1968). "System Reliability Allocation and a Computational Algorithm", IEEE Transactions on Reliability, R-17(2): 64-69.
[10] Nakagawa, Y., Miyazaki, S., (1981). "Surrogate Constraints Algorithm for Reliability Optimization Problems with Two Constraints", IEEE Transactions on Reliability, R-30(2): 175-180.
[11] Nakagawa, Y., Nakashima, K., (1977). "A Heuristic Method for Determining Optimal Reliability Allocation", IEEE Transactions on Reliability, R-26(3): 156-161.
[12] Dinghua, S., (1987). "A New Heuristic Algorithm for Constrained Redundancy-Optimization in Complex Systems", IEEE Transactions on Reliability, R-36(5): 621-623.
[13] Agarwal, M., Gupta, R., (2005). Penalty function approach in heuristic algorithms for constrained redundancy reliability optimization, IEEE Transactions on Reliability, 54(3): 549-558.
[14] Kumar, P., Chaturvedi, D.K., Pahuja, G.L., (2010). "Heuristic methods for solving redundancy allocation in complex systems", International Journal of Reliability and Safety, 4(2-3): 285-298.
[15] Coit, D.W., Smith, A.E., (1996). "Reliability optimization of series-parallel systems using a genetic algorithm", IEEE Transactions on Reliability, 45(2): 254-266.
[16] Levitin, G., Lisnianski, A., Ben-Haim, H., Elmakis, D., (1998). "Redundancy optimization for series-parallel multi-state systems", IEEE Transactions on Reliability, 47(2): 165-172.
[17] Shelokar, P.S., Jayaraman, V.K., Kulkarni, B.D., (2002). "Ant algorithm for single and multiobjective reliability optimization problems", Quality and Reliability Engineering International, 18(6): 497-514.
[18] Zhao, J.H., Liu, Z., Dao, M.T., (2007). "Reliability optimization using multiobjective ant colony system approaches", Reliability Engineering & System Safety, 92(1): 109-120.
[19] Kulturel-Konak, S., Smith, A.E., Coit, D.W., (2003). "Efficiently Solving the Redundancy Allocation Problem Using Tabu Search", IIE Transactions, 35(6): 515-526.
[۲۰] جولای، فریبرز.، زارعی‌شوریجه، محمدعلی.، جویبار، سبحان. (۱۳۹۴). «توسعه یک روش برای حل مساله قابلیت اطمینان مبتنی بر ترکیب تکنیک شبیه‌سازی و الگوریتم بهینه‌سازی ازدحام ذرات در شرایط عدم قطعیت»، نشریه پژوهش‌های مهندسی صنایع در سیستم‌های تولید، ۳(۵): ۷۳-۸۹.
[21] Ramirez-Marquez, J.E., Coit, D.W., (2004). "A heuristic for solving the redundancy allocation problem for multi-state series-parallel systems", Reliability Engineering & System Safety, 83(3): 341-349.
[22] Kuo, W., Wan, R., (2007). "Recent Advances in Optimal Reliability Allocation", IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 37(2): 143-156.
[23] Cao, D., Murat, A., Chinnam, R.B., (2013). "Efficient xact optimization of multi-objective redundancy allocation problems in series-parallel systems", Reliability Engineering & System Safety, 111: 154-163.
[24] Garg, H., Sharma, S., (2013). "Multi-objective eliability-redundancy allocation problem using particle swarm optimization", Computers & Industrial Engineering, 64(1): 247-255.
[25] Garg, H., (2015). "An approach for solving constrained reliability-redundancy allocation problems using cuckoo search algorithm", Beni-Suef University Journal of Basic and Applied Sciences, 4(1): 14-25.
[26] Abouei Ardakan, M., Zeinal Hamadani, A., (2014). "Reliability optimization of series–parallel systems with mixed redundancy strategy in subsystems", Reliability Engineering & System Safety, 130: 132-139.
[27] Liu, Y., Qin, G., (2014). "A Modified Particle Swarm Optimization Algorithm for Reliability Redundancy Optimization Problem", Journal of Computers, Vol. 9: 2124-2131.
[28] Huang, C.L., (2015). "A particle-based simplified swarm optimization algorithm for reliability redundancy allocation problems", Reliability Engineering & System Safety, 142: 221-230.
[29] Kong, X., Gao, L., Ouyang, H., Li, S., (2015). "Solving the redundancy allocation problem with multiple strategy choices using a new simplified particle swarm optimization", Reliability Engineering & System Safety, 144: 147-158.
[30] Zhang, E., Chen, Q., (2016). "Multi-objective reliability redundancy allocation in an interval environment using particle swarm optimization", Reliability Engineering & System Safety, 145: 83-92.
[31] Feizabadi, M., Jahromi, A.E., (2017). "A new model for reliability optimization of series-parallel systems with non-homogeneous components", Reliability Engineering & System Safety, 157: 101-112.
[32] Ziaei, M., Hojati, F., (2014). "Using a huristic algorithm based on greedy method for reliability optimization of a series system with multiple choice and limited budget", in 3th international reliability engineering conference.
[33] Ahmadizar, F., Soltanpanah, H., (2011). "Reliability optimization of a series system with multiple-choice and budget constraints using an efficient ant colony approach", Expert Systems with Applications, 38(4): 3640-3646.
[34] Amari, S.V., Dill, G., (2010). "Redundancy optimization problem with warm-standby redundancy", in 2010 Proceedings - Annual Reliability and Maintainability Symposium (RAMS).
[35] Cormen, T.H., Charles, E. Leiserson. Ronald L. Rivest. (2009). Introduction to Algorithms, Third Edition. The MIT Press.
[36] Ushakov, I., (2013). "Optimal Resource

Allocation. 1nd ed. San Diego", USA: John Wiley & Sons Inc.

[37] Pardeep, K., Chaturvedi, D., Pahuja, G., (2010). "An efficient heuristic algorithm for determining optimal redundancy allocation of complex networks", Theory & applications, 3(18): 15-28.

[38] Jae-Hwan, K., Bong-Jin, Y., (1993). "A heuristic method for solving redundancy optimization problems in complex systems", IEEE Transactions on Reliability, 42(4): 572-578.

[39] Gao, J., Shan, J., Cui, H., Li, L., (2011). "A hybrid genetic algorithm with effective local search technique", Applied Mathematics, 39: 4814-4817.

[40] Billionnet, A., (2008). Redundancy Allocation for Series-Parallel Systems Using Integer Linear Programming, IEEE Transactions on Reliability, 57(3): 507-516.

[41] Lee, H., Kuo, W., Ha, C., (2003). "Comparison of max-min approach and NN method for reliability optimization of series-parallel system", Journal of Systems Science and Systems Engineering, 12(1): 39-48.

[42] Coit, D.W., Liu, J.C., (2000). "System Reliability Optimization With k-out-of-n Subsystems, International Journal of Reliability", Quality and Safety Engineering, 7(2): 129-142.

IERPS Journal of Industrial Engineering Research in Production Systems

ISSN: 2345-2269

Journal of Industrial Engineering Research in Production Systems

Autumn and Winter 2017, Volume 5, No. 11, pp.149-171
www.ier.basu.ac.ir

DOI: 10.22084/ier.2017.11693.1534

Providing a New Algorithm by Combining and Improving Greedy Method and Weighed Costs for Solving Different Redundancy Allocation Problems (RAP)

M. Ziaei[1,*], F. Hojati[1]

[1] Institute of Materials and Energy, Iranian Space Research Center, Isfahan, Iran.

ARTICLE INFO

***Article history*:**

Received 10 August 2016
Accepted ?

***Keywords*:**

Reliability optimization
redundancy allocation
Bridge system
Series-parallel systems

ABSTRACT

Due to the importance of reliability in systems performance and amount and variety of its costs, system designers have more attention to the reliability optimization, especially redundancy allocation problems (RAP). Reliability optimization problems are various and NP-hard. Thus, for any kind of them, different methods have developed which are usually suitable for limited kind of these problems. This paper provides a new algorithm (GW) that by doing some heuristic improvements on greedy method and weighed costs and combining them, it can be used for RAP with different configurations, variety of components and diversity in redundancy strategy. Using a new method of weighting costs, using an innovative greedy index and a specific search mechanism in this algorithm increas its speed, accuracy, and flexibility in solving the RAP types. The algorithm abilities are shown by solving several test problems with different conditions and variety of configurations while some of them are large. Results reveal that this algorithm is able to solve various problems and produces solutions in short time that are equal or very near to optimal solutions or the best existent solutions. GW has tangible and operational logic, therefore, it can develop the practical insight of system designers.

[*] Corresponding author. Mazaher Ziaei
Tel.: 031-33222429; *E-mail address: m.ziaei@isrc.ac.ir*