مدیریت تولید و عملیات، دوره ۱۰، پیاپی ۱۹، شماره ۲، پاییز زمستان ۱۳۹۸
دریافت: ۱۳۹۶/۱۰/۲۵ پذیرش: ۱۳۹۸/۰۹/۰۶
صص: ۱۸۱- ۲۰۱

به‌کارگیری الگوریتم شاخه و حد با حدودِ پایین قوی برای حل مسئلهٔ حداقل‌کردن زمان انجام کل کارها روی ماشین پردازندهٔ انباشته

سیده ناهید هاشمی[۱]، علی حسین‌زاده کاشان[۲*]

۱- کارشناس ارشد دانشکدهٔ مهندسی صنایع و سیستم‌ها، دانشگاه تربیت مدرس، تهران، ایران، hasheminahidsadat@yahoo.com
۲- استادیار دانشکدهٔ مهندسی صنایع و سیستم‌ها، دانشگاه تربیت مدرس، تهران، ایران، a.kashan@modares.ac.ir

چکیده: در این مقاله مسئلهٔ زمان‌بندی ماشین پردازندهٔ انباشته با فرض وجود کارهایی با اندازهٔ غیریکسان و با هدف حداقل‌کردن زمان انجام کل کارها (C_{max}) بررسی شده است. هدف این مقاله، حل مسئلهٔ مدنظر با بهره‌گیری از حدود پایین قوی و با استفاده از الگوریتم شاخه و کران حد، یکی از روش‌های حل دقیق، است. در این الگوریتم از دو روش جدید به‌نام‌های LB_2 و LB_3 برای تولید حد پایین استفاده و نتایج با حد پایین موجود در ادبیات به‌نام LB_1 مقایسه شده است. برای ارزیابی عملکردِ روش ارائه‌شده، دسته‌ای از نمونه مسائل به‌صورت تصادفی تولید و روش شاخه و حد با حدود پایینِ متفاوت روی این مسائل آزمایش شده است. نتایج محاسبات نشان می‌دهد در الگوریتم شاخه و کران وقتی اندازهٔ کارها نسبت به ظرفیت ماشین بزرگ باشد، حد پایین LB_2 بهترین عملکرد را دارد و زمانی‌که اندازهٔ کارها نسبت به ظرفیت ماشین کوچک باشد (حداکثر به‌اندازهٔ G نصف ظرفیت ماشین)، الگوریتم با حد پایین LB_1 عملکرد بهتری دارد. همچنین زمانی‌که اندازهٔ کارها متوسط باشد، LB_3 بهترین عملکرد را دارد.

واژه‌های کلیدی: زمان‌بندی، ماشین‌های پردازندهٔ انباشته، روش شاخه و کران، حد پایین

*نویسنده مسؤول

مقدمه

امروزه مسائل زمان‌بندی در اکثر محیط‌های اقتصادی به‌صورت مزیت رقابتی مطرح و شناخته شده است. طی سه‌دهۀ اخیر به مسائل زمان‌بندی که در آنها انباشته‌سازی کارها مطرح است توجه زیادی شده است، به‌نحوی‌که امروزه زمان‌بندی انباشته‌ها در سیستم تولید یکی از سیاست‌های معمول در بسیاری از صنایع است. علت این انگیزش در افزایش سرعت تولید و کاهش هزینه‌های مرتبط با آن ازطریق سیاست تولید انباشته‌ای به‌جای تولید منفرد کارها است. ماشین‌های پردازش انباشته[۱] در سیستم‌های تولید انباشته‌ای قابلیت پردازش دسته‌ای از کارها را درقالب یک انباشته به‌طور هم‌زمان دارند. منظور از یک انباشته، گروهی از کارهاست که به‌طور هم‌زمان زیر عملیات ماشین قرار می‌گیرند؛ بنابراین ظرفیت ماشین تعیین‌کنندۀ اندازۀ انباشته است. خانوادۀ مسائل زمان‌بندی ماشین‌های پردازندۀ انباشته با کارهای با اندازۀ غیریکسان در دسته مسائل NP-hard قرار می‌گیرد (اوزوی[۲]، ۱۹۹۴).

در ادبیات زمان‌بندی تولید انباشته‌ای، مثال‌های متعددی از ایستگاه‌های تولیدی ذکر شده‌اند که در آن از ماشین‌های پردازش انباشته استفاده می‌شود؛ برای نمونه عملیات فر درون‌سوز در تولید بوردهای مدار چاپی، داخل فرهایی انجام می‌شود که می‌توانند چندین گروه از بوردها را در خود و زیر عملیات حرارتی قرار دهند (لی و همکاران، ۱۹۹۹). در اینجا فرها نقش ماشین‌های پردازش انباشته را بازی می‌کنند. مثالی دیگر، فرآیند حکاکی هدایت الکتریکی[۳] در بوردهای مدار چاپی است. طی این فرآیند و طی عملیات فوتولیتوگرافی[۴] از تانک‌هایی (ماشین‌های پردازش انباشته) استفاده می‌شود که حاوی مواد شیمیایی برای از بین بردن لایۀ فلزی روی بورد هستند که با مادۀ محافظت‌کننده پوشانده نشده است (احمدی و همکاران، ۱۹۹۲). همچنین می‌توان به روترهای کنترل عددی[۵] استفاده‌شده در برش ورق‌های فلزی و یا مدارهای چاپی اشاره کرد. طی فرآیند برش ورق‌ها، با چرخش صفحه‌گردنده یک انباشته از قطعات فلزی هم‌خانواده تولید می‌شود. در اینجا روتر برنده با یک ماشین پردازش انباشته مدل می‌شود (احمدی و همکاران، ۱۹۹۲). همچنین می‌توان به کاربرد مدل‌های زمان‌بندی پردازش انباشته در سایر صنایع ازجمله صنایع ریخته‌گری (ماتیراجان و سیواکومار[۶]، ۲۰۰۶) ، صنایع تولید مبلمان و اثاثیۀ خانه (یعقوبیان و همکاران[۷]، ۱۹۹۹) ، صنایع تولید آهن و فولاد (اولامارا[۸]، ۲۰۰۷) و غیره اشاره کرد.

اگرچه کاربردهای مدل‌های زمان‌بندی ماشین‌های پردازش انباشته در طیف وسیعی از صنایع تولیدی یافت می‌شوند، حجم عظیمی از پژوهش‌های انجام‌شده فقط به کاربرد این مدل‌ها در صنایع تولید نیمه‌هادی‌ها مربوط می‌شوند. تعداد زیادی از ماشین‌آلات استفاده‌شده در صنایع تولید نیمه‌هادی‌ها در زمرۀ ماشین‌های پردازش انباشته هستند که قابلیت انجام عملیات هم‌زمان را بر کارهای درون‌انباشته دارند. بعضی از این ماشین‌آلات عبارتند از cleaning، diffusion، oxidation و burn-in operation. عملیات انجام‌شده با فرهای درون‌سوز (burn-in operation) قابل مدل‌سازی درقالب ماشین(های) پردازش انباشته است. فقط ممکن است ظرفیت فر، بزرگ‌تر از اندازۀ یک کار باشد؛ بنابراین تعداد بوردهایی که می‌توانند داخل فر قرار بگیرند تعیین‌کنندۀ ظرفیت فر است. اندازۀ یک کار نیز با تعداد بوردهای لازم تعریف می‌شود. ازآنجاکه چیپ‌های IC برای مدتی طولانی بیش از آنچه که برای آنها در نظر گرفته شده است در داخل فر زیر عملیات حرارتی قرار می‌گیرند، قرارگیری چیپ‌های مربوط به سفارشات مختلف داخل فر امکان‌پذیر است. این در حالی است که خارج‌کردن آنها از فر پیش از مدت زمان لازم برای پخت امکان‌پذیر

نیست. گروهی از بوردهایی که هم‌زمان داخل فر قرار می‌گیرند یک انباشته را تشکیل می‌دهند. زمان لازم برای انجام عملیات حرارتی روی انباشته برابر زمان لازم برای انجام عملیات روی کاری است که در بین سایر کارهای درونِ انباشته زمان بیشتری را برای انجام عملیات حرارتی می‌طلبد. با شروع عملیات روی یک انباشته، هیچ کاری نمی‌تواند به انباشته اضافه و یا از آن خارج شود. همچنین ازآنجاکه دمای فر همواره ثابت است، نیاز به هیچ‌گونه آماده‌سازی حرارتی نیست (لی و همکاران، ۱۹۹۹). اغلب در فرآیند تولید نیمه‌هادی‌ها، ایستگاه‌هایی که این ماشین‌ها در آنها فعالیت می‌کنند ایستگاه گلوگاه تولید هستند. این امر بدین خاطر است که این تجهیزات بسیار گران است؛ بنابراین میزان بهره‌گیری از آنها زیاد است. به‌علاوه عملیات تولید در این ماشین‌ها طولانی است و بدون انقطاع انجام می‌شود و این موجب انتظار طولانی سایر قطعات می‌شود؛ بنابراین هرگونه بهبود در زمان‌بندی و کاهش زمان سیکل نتایج مثبتی را از دیدگاه اقتصاد تولید دارد (ماتیراجان و سیواکومار[۹]، ۲۰۰۶). باتوجه‌به اینکه تلاش‌های انجام‌شده در این زمینه به ارائهٔ روش‌های حل محدود است و در رابطه با ارزیابی عملکرد واقعی این روش‌ها تلاش زیادی انجام نشده است، در این مقاله ضعف موجود درزمینهٔ ارزیابی عملکرد الگوریتم‌های ارائه‌شده برای مسئلهٔ زمان‌بندی ماشین‌های پردازش انباشته با وجود کارهای با اندازهٔ غیریکسان جبران می‌شود.

ساختار ادامهٔ مقاله بدین شرح است که درادامهٔ پژوهش حاضر، مروری بر پیشینهٔ پژوهش ارائه می‌شود. سپس تعریف مسئله و مفروضات بیان و بعد از آن حدود بالا و پایین تشریح می‌شود. بخش بعدی به رویکرد حل مسئله اختصاص داده می‌شود. درپایان نیز بحث و نتایج حاصل از محاسبات تشریح و نتیجه‌گیری ارائه می‌شود.

مروری بر پیشینهٔ پژوهش

نخستین تلاش‌های انجام‌شده در زمان‌بندی ماشین‌های پردازندهٔ انباشته به ایکورا و جیمپل[۱۰] (۱۹۸۶) نسبت داده می‌شود. همچنین نخستین افرادی که زمان‌بندی ماشین‌های پردازش‌گر انباشته را با فرض وجود کارهای با اندازه نامساوی[۱۱] بررسی کردند، دابسون و نامبیادوم[۱۲](۲۰۰۱) بودند. اوزوی (۱۹۹۴) مسئلهٔ حداقل‌کردن C_{max} و SC_i روی یک ماشین پردازش‌گر انباشته را بررسی و محل کاربرد آن را در صنایع تولید نیمه‌هادی معرفی کرد. وی برای مسئلهٔ حداقل‌کردن C_{max} نشان داده است که این مسئله با فرض یکسان‌بودن کلیهٔ زمان‌های پردازش معادل مسئلهٔ شناخته‌شدهٔ بسته‌بندی اشیاء در ظروف و مسئله‌ای NP-hard است. سپس برمبنای الگوریتم first-fit که برای مسئلهٔ بسته‌بندی اشیاء در ظروف ابداع شده است، ۴ الگوریتم برای مسئلهٔ حداقل‌کردن C_{max} ارائه کرده است. مبنای عمل کلیهٔ این الگوریتم‌ها ابتدا مرتب‌سازی کارها براساس ترتیب‌های مختلف و سپس استفاده از الگوریتم first-fit برای انباشته‌سازی کارها است. کارایی این الگوریتم‌ها ازطریق محاسبات کامپیوتری بررسی و معلوم شده است. یکی از این الگوریتم‌ها که برمبنای مرتب‌سازی کارها براساس LPT[۱۳] عمل می‌کند، دارای عملکرد بهتری نسبت به سایرین است. دربارهٔ مسئلهٔ حداقل‌کردن SC_i نیز نشان داده شده است که این مسئله NP-hard است و الگوریتم شاخه و حدی برای به دست آوردن جواب بهینه در مسائل کوچک ارائه شده است. وی همچنین ۲ الگوریتم ابتکاری برای این مسئله توسعه داده است.

دوپونت و جولای[۱۴] (۱۹۹۸) برای مسئلهٔ حداقل‌کردن C_{max} دو الگوریتم ابتکاری ارائه کرده‌اند که یکی از آنها برمبنای اصلاح الگوریتم‌های ارائه‌شده به‌وسیلهٔ اوزوی(۱۹۹۴) در استفاده از الگوریتم best-fit به‌جای first-fit است و دیگری برمبنای تعیین اقلام یک انباشته با استفاده از حل مسئلهٔ کوله‌پشتی است. نتایج محاسباتی نشان داده‌اند هر دو الگوریتم عملکرد بهتری نسبت به الگوریتم‌های قبلی دارند. آنها همچنین برای مسئلهٔ حداقل‌کردن SC_i تعدادی الگوریتم ابتکاری ارائه کرده‌اند و ازطریق محاسبات کامپیوتری نشان داده‌اند عملکرد این الگوریتم‌ها در مقایسه با الگوریتم‌های ارائه‌شده به‌وسیلهٔ اوزوی (۱۹۹۴) بهتر است. ژانگ و همکاران[۱۵] (۲۰۰۱) عملکرد الگوریتم‌های ارائه‌شدهٔ اوزوی (۱۹۹۴) را برای مسئلهٔ حداقل‌کردن C_{max} در بدترین حالت تحلیل و یک الگوریتم تقریب جدید ارائه کرده‌اند. حسین‌زاده کاشان و همکاران (۲۰۰۹) فرضیات مسئلهٔ معرفی‌شده در مرجع ژانگ و همکاران (۲۰۰۱) را تعمیم و یک الگوریتم ابتکاری با نسبت عملکرد معلوم ارائه داده‌اند. آنها همچنین یک الگوریتم تقریب برای حالتی ارائه داده‌اند که اندازه و زمان پردازش کارها موافق یکدیگر هستند.

دوپونت و فیلپو[۱۶](۲۰۰۲) مسئلهٔ زمان‌بندی یک ماشین پردازندهٔ انباشته با کارهایی با اندازهٔ غیریکسان را مطالعه کرده‌اند و برای این مسئله یک الگوریتم شاخه و حد ارائه داده‌اند. همچنین رفیعی پارسا و همکارانش[۱۷](۲۰۱۰) برای این مسئله حد پایین مناسبی ارائه و با استفاده از روش تولید ستون و روش شاخه و کران، الگوریتم شاخه و قیمت را برای رسیدن به جواب بهینه پیشنهاد کرده‌اند. این الگوریتم قابلیت رقابت با روش شاخه و کرانِ دوپونت و فیلپو(۲۰۰۲) را دارد. مالاپرت و همکارانش[۱۸] (۲۰۱۲) با استفاده از برنامه‌ریزی محدودیت‌ها همین مسئله را با تابع هدف L_{max} بررسی کرده‌اند. کاشان و کریمی (۲۰۱۰) نیز در مقالهٔ خود این مسئله را بررسی کرده‌اند و دو روش جدید تولید حد پایین (LB[۱۹]) روی مقدار بهینهٔ تابع هدف به‌نام‌های LB_2 و LB_3 ارائه داده‌اند. آنها ثابت کرده‌اند این روش نسبت به حد پایین LB_1 که به‌وسیلهٔ اوزوی(۱۹۹۴) ارائه شده است عملکرد بهتری دارد. همچنین در این مرجع ثابت شده است عملکرد LB_3 حداقل به‌خوبی عملکرد LB_2 است. لی و ژانگ (۲۰۱۸) مسئلهٔ حداقل‌کردن زمان انجام کارها را در یک ماشین پردازش دسته‌ای مطالعه کرده‌اند. برخلاف مسائل زمان‌بندی ماشینِ پردازشِ دسته‌ای کلاسیک که در آن ظرفیت یک ماشین به‌صورت مسئلهٔ کوله‌پشتی یک‌بعدی مدل‌سازی می‌شود، در این مطالعه ظرفیت ماشین و اندازهٔ کارها با مستطیل دوبعدی نشان داده شده است. یک انباشته از کارها زمانی شدنی است که کارها بتوانند بدون هم‌پوشانی با یکدیگر در ماشین قرار بگیرند. کارها دارای اندازهٔ متفاوت هستند و با طول، عرض و زمان پردازش نشان داده می‌شوند. هدف، تعیین تعداد انباشته‌ها برای تخصیص کارها به انباشته‌ها به‌نحوی است که زمان انجام کل کارها حداقل شود. مسئلهٔ مطالعه‌شده در این مقاله، ترکیبی از مسئلهٔ بسته‌بندی اقلام در ظروف به‌صورت دوبعدی (2D-BPP) و مسئلهٔ زمان‌بندی ماشین پردازندهٔ انباشته (BPM) است. آنها برای حل این مسئله مدل برنامه‌ریزی عدد صحیح مختلط و چند روش ابتکاری ارائه داده‌اند. مسئلهٔ زمان‌بندی یک ماشین پردازندهٔ انباشته با دو تابع هدف هم‌زمان "حداقل‌کردن زمان تکمیل تمام کارها" و "حداقل‌کردن هزینهٔ انرژی کل" به‌وسیلهٔ وانگ و همکاران[۲۰] (۲۰۱۶) بررسی و این مسئله به‌صورت یک مسئلهٔ برنامه‌ریزی عدد صحیح مدل شده است. سپس روش دقیق ε محدودیت[۲۱] برای به دست آوردن نقاط پارتوی دقیق و دو روش ابتکاری برمبنای ایدهٔ تجزیه برای به دست آوردن نقاط پارتوی تقریبی در مسائل با اندازه‌های بزرگ توسعه یافته است.

حل مسائل زمان‌بندی ماشین‌های پردازندهٔ انباشته با فرض وجود کارهای با اندازهٔ نامساوی با استفاده از الگوریتم‌های فراابتکاری توجه پژوهشگران زیادی را به‌خود جلب کرده است. ملوک و همکارانش[۲۲](۲۰۰۴) برای مسئلهٔ حداقل‌کردن C_{max} روی یک ماشین پردازندهٔ انباشته با فرض اینکه که زمان پردازش و اندازهٔ کارهای مختلف متفاوت است، الگوریتم شبیه‌سازی تبرید را استفاده کرده‌اند. نونگ و همکارانش[۲۳](۲۰۱۲) همین مسئله را در دو حالتِ مختلف برای اندازهٔ کارها بررسی کرده‌اند. در حالت نخست اندازهٔ کارها یکسان در نظر گرفته و یک الگوریتم تقریب با پیچیدگی زمانی چندجمله‌ای ارائه شده است. در حالت دوم نیز اندازهٔ کارها متفاوت فرض و یک الگوریتم تقریب پیشنهاد شده است. حسین‌زاده کاشان و همکاران[۲۴] (۲۰۰۶) نیز این مسئله را بررسی کرده‌اند و برای آن یک الگوریتم ژنتیک برمبنای نمایش کروموزومی براساس کلیدهای تصادفی و یک الگوریتم ژنتیک ترکیبی براساس نمایش کروموزومی ابداعی پیشنهاد کرده‌اند. نتایج مقایسات آنها نشان می‌دهد الگوریتم ژنتیک ترکیبی عملکرد بهتری را نسبت به شبیه‌سازی تبریدِ ارائه‌شده به‌وسیلهٔ ملوک و همکارانش(۲۰۰۴) دارد. چانگ و سون (۲۰۱۸) مسئلهٔ یک ماشین پردازندهٔ انباشته با فرض وجود کارها با زمان آزادسازی و اندازه‌های متفاوت را برای حداقل‌کردن زمان انجام کل کارها در نظر گرفته‌اند و یک الگوریتم سیستم ایمنی مصنوعی مبتنی‌بر ایمونوگلوبولین (IAIS) برای حل این مسئله ارائه داده‌اند. الگوریتم مدنظر دو مشخصهٔ اصلی دارد؛ نخست اینکه فضای جستجو به فرایند نوسازی جسمی محدود شده است تا زمان محاسبات کاهش یابد و دیگر اینکه ترکیبی از چندین روش جستجوی محلی استفاده شده است تا در هر بار اجرا همسایگی تغییر کند و از بهینگی محلی جلوگیری شود. نتایج محاسبات آنها نشان داده است روش ارائه‌شده عملکرد خوبی دارد.

مسئلهٔ حداقل‌کردن هم‌زمان C_{max} و T_{max} به‌وسیلهٔ کاشان و همکاران[۲۵] (۲۰۱۰) مطالعه شده است. برای این مسئله مؤلفین یک الگوریتم ژنتیک دوهدفه ترکیبی ارائه کرده‌اند. نتایج محاسبات آنها نشان می‌دهد این الگوریتم قابلیت یافتن جواب‌های نزدیک به جواب‌های بهینهٔ پارتو را دارد. کاشان و کریمی (۲۰۰۸) نیز چارچوبی برمبنای بهینه‌سازی کلونی مورچگان در دو نسخهٔ متفاوت ارائه داده‌اند. زو و همکاران[۲۶] (۲۰۱۲) مسئلهٔ حداقل‌کردن C_{max} را با فرض وجود زمان‌های ورود پویا برای کارها با اندازهٔ غیریکسان روی یک ماشین پردازندهٔ انباشته در نظر گرفته‌اند و یک مدل برنامه‌ریزی عدد صحیح مختلط و یک حد پایین معتبر برای این مسئله پیشنهاد داده‌اند. چون این مسئله $NP-hard$است، آنها برای حل یک الگوریتم ابتکاری و یک الگوریتم بهینه‌سازی کلونی مورچگان برمبنای فرض‌های ارائه‌شده پیشنهاد دادند. نتایج کار نشان می‌دهد الگوریتم کلونی مورچگان پیشنهادشده به‌وسیلهٔ آنها جواب‌های بهتری را در زمان قابل قبولی در مقایسه با دو روش ابتکاری ژنتیک و سیمپلکس تولید می‌کند؛ مخصوصاً زمانی‌که اندازهٔ کارها بزرگ است. لی و همکاران[۲۷] (۲۰۱۳) الگوریتمی برای مسئله‌ای با یک ماشین پردازش انباشته با کارهایی ارائه کرده‌اند که دارای موعد تحویلِ مشخص، هستند. در این مسئله، محدودیت‌های فازی برای برقراری روابط پیش‌نیازی و پس‌نیازی اعمال شده است. تابع هدف نیز به‌صورت حداقل‌کردن زمان تکمیل تمام کارها تعریف شده است. چاندرو و همکاران[۲۸] (۱۹۹۳) در مقالهٔ خود یک روش شاخه و حد برای حداقل‌کردن زمان انجام کل کارها روی یک ماشین ارائه کردند. آنها همچنین برای مسئلهٔ ماشین‌های موازی چند روش ابتکاری معرفی کردند. لی و همکارانش[۲۹] (۲۰۱۳) چند روش ابتکاری برای مسئلهٔ حداقل‌کردن C_{max} روی تعدادی ماشین متفاوت به‌صورت موازی درحالت متفاوت‌بودن اندازهٔ کارها ارائه کرده‌اند.

کوه و همکارانش[۳۰] (۲۰۰۵) برای مسئلهٔ پردازش دسته‌ای با خانواده‌های ناسازگار کارها و اندازهٔ متفاوت کارها و سه تابع هدفِ "حداقل‌کردن زمان تکمیل تمام کارها"، "حداقل‌کردن مجموع زمان تکمیل کارها" و "حداقل‌کردن مجموع وزنی زمان تکمیل کارها" مدلی ریاضی ارائه کرده‌اند. همچنین برای حل مسائل با اندازهٔ واقعی تعدادی روش ابتکاری و یک الگوریتم ژنتیک ترکیبی پیشنهاد کرده‌اند. لی[۳۱] (۲۰۱۲) نیز مسئلهٔ حداقل‌کردن C_{max} را روی چند ماشین موازی درحالت متفاوت‌بودن اندازهٔ کارها در نظر گرفته و برای آن یک الگوریتم تقریب ارائه کرده - است. در مقالهٔ جیا و همکاران[۳۲] (۲۰۱۷) مسئلهٔ زمان‌بندی کارهای با اندازهٔ غیریکسان و زمان رسیدن پویا روی مجموعه‌ای از ماشین‌های موازی با ظرفیت متفاوت با هدف حداقل‌کردن C_{max} در نظر گرفته شده است. در این مقاله ابتدا مدلی ریاضی برای مسئله ارائه و حد پایینی برای آن معرفی شده است، سپس دو روش فراابتکاری برمبنای الگوریتم کلونی مورچگان برای حل مسئله ارائه شده است. داموداران و همکاران[۳۳] (۲۰۱۲) نیز این مسئله را مطالعه کرده‌اند و یک الگوریتم بهینه‌سازی ازدحام ذرات (PSO) را برای حل آن ارائه داده‌اند. در مقالهٔ جیا و همکاران[۳۴] (۲۰۱۵) نیز همین مسئله، مطالعه و برای حل آن یک روش ابتکاری برمبنای الگوریتم First-Fit-Decreasing (FFD) و یک روش ابتکاری برمبنای الگوریتم Max-Min Ant System (MMAS) ارائه شده است. نتایج این مقاله نشان می‌دهد این دو روش عملکرد بهتری را نسبت به روش ابتکاری معرفی‌شده به‌وسیلهٔ داموداران و همکاران (۲۰۱۲) برای این مسئله دارند. از طرف دیگر MMAS عملکرد بهتری در مقایسه با FFD دارد. ژو و همکاران(۲۰۱۸) مسئلهٔ حداقل‌کردن C_{max} را روی ماشین‌های موازی نامرتبط با فرض وجود کارهای با اندازهٔ غیریکسان و زمان‌های آزادسازی دلخواه بررسی کرده‌اند و برای حل این مسئله دو حد پایین و یک الگوریتم ژنتیک برمبنای کدگزاری کلید تصافی ارائه داده‌اند. عملکرد الگوریتم پیشنهادی با یک حل‌کنندهٔ تجاری (ILOG CPLEX)و دو روش فراابتکاری موجود در ادبیات (الگوریتم حریصانه و الگوریتم بهینه‌سازی ازدحام ذرات) مقایسه شده و آزمایش‌ها محاسباتی نشان داده است الگوریتم پیشنهادی در مقایسه با روش‌های دیگر راه‌حل‌های بهتری را تولید می‌کند. آنها همچنین کیفیت حدود پایین پیشنهادی را ارزیابی کرده‌اند و نشان داده‌اند که از حدود پایین موجود، عملکرد بهتری دارد. اروبو و همکاران[۳۵] (۲۰۱۷) مسئلهٔ زمان‌بندی مجموعه‌ای از کارهای با اندازهٔ متفاوت و زمان آماده‌سازی غیرصفر را روی مجموعه‌ای از ماشین‌های موازی غیرمرتبط برای حداقل‌کردن C_{max} در نظر گرفته‌اند و برمبنای الگوریتم‌های FF[۳۶] و BF[۳۷]، چند روش ابتکاری برای مسئله توسعه داده‌اند. در این مرجع یک مدل برنامه‌ریزی عدد صحیح مختلط و یک حد پایین برای ارزیابی کیفیت ابتکاری‌ها ارائه شده است.

باتوجه‌به اینکه روش‌های استفاده‌شده در مقالات موجود در ادبیات برای حل این مسئله، بیشتر روش‌ها برمبنای روش‌های تقریبی است و به روش‌های حل بهینه کمتر توجه شده است. هم‌چنین باتوجه‌به اهمیت این مسائل در حوزهٔ صنعت و صرفه‌جویی هزینه‌ای که درنتیجهٔ بهینه‌سازی زمان‌بندی عملیات تولید دست‌یافتنی است، نتیجه می‌شود توسعهٔ روش‌های حل دقیق مانند روش شاخه و حد برای این‌گونه مسائل اهمیت زیادی دارد؛ ازاین‌رو هدف این پژوهش ارائهٔ روش شاخه و حد کارا با استفاده از حدود پایین و حدود بالا روی مقدار بهینهٔ تابع هدف در مسئلهٔ زمان‌بندی ماشین‌های پردازش انباشته با وجود کارهای با اندازهٔ غیریکسان است.

تعریف مسئله و مفروضات

مسئله‌ای که در این پژوهش درحال بررسی است، مسئلهٔ زمان‌بندی یک ماشین پردازندهٔ انباشته با فرض وجود کارهای با اندازهٔ غیریکسان است. مفروضات مسئله به‌شرح زیر است:

۱- تعداد n کار برای زمان‌بندی روی یک ماشین پردازش انباشته وجود دارد که همگی در لحظهٔ صفر در سیستم در دسترس هستند. کلیهٔ کارها با یکدیگر سازگار وتنها متعلق به یک خانواده هستند (زمان‌بندی بدون خانواده‌های کارها).

۲- با شروع عملیات ماشین روی یک انباشته، توقف عملیات امکان‌پذیر نیست و پیش از اتمام عملیات ماشین، هیچ‌کاری نمی‌تواند به انباشته اضافه و یا از آن خارج شود.

۳- مقادیر زمان‌های پردازش و اندازهٔ کارها از قبل مشخص و قطعی هستند.

۴- زمان شروع و زمان ختم عملیات تمامی کارهایی که متعلق به یک انباشته هستند با یکدیگر یکسان است. زمان پردازش انباشته نیز برابر زمان پردازش کاری است که بین سایر کارهای متعلق به انباشته زمان پردازش لازم برای آن طولانی‌تر است.

۵- حداکثر ظرفیت ماشین برای انجام عملیات هم‌زمان روی کارها برابر B است؛ به عبارت دیگر مجموع ابعاد کارهای متعلق به یک انباشته نباید از مقدار B متجاوز شود. همچنین فرض می‌شود کلیهٔ کارها دارای اندازه‌ای کوچک‌تر یا مساوی با ظرفیت ماشین است.

۶- خرابی ماشین(ها) مجاز نیست.

ماشین‌های پردازش انباشته در مقایسه با سایر عملیات تولید و تست دارای زمان انجام عملیات طولانی‌تری هستند و ایستگاه گلوگاه تولید به‌شمار می‌روند؛ بنابراین میزان بهره‌برداری زیاد از ماشین(ها) در این عملیات باعث افزایش توان عملیاتی کل سیستم می‌شود. ازآنجاکه میزان بهره‌برداری بیشتر به‌معنی کاهش زمان انجام عملیات کلیهٔ کارها (یا به‌طور معادل مقدار C_{max}) با ماشین‌ها است، معیار عملکرد، حداقل‌کردن C_{max} است.

اندیس‌ها و نمادها

j : نشانگر کار jام، j=1,2,…,n (n تعداد کل کارها برای زمان‌بندی).

s_j: زمان پردازش کار jام.

P_j: زمان پردازش کار jام.

B: حداکثر ظرفیت ماشین.

مجموعه‌ها.

S: مجموعهٔ اندازهٔ کارها.

P: مجموعهٔ زمان پردازش کارها.

$S(u,v)$: مجموعه کارهایی که اندازهٔ آنها بزرگ‌تر از u و کوچک‌تر یا مساوی با v است.

$\bar{S}(u,v)$: مجموعه کارهایی که اندازهٔ آنها بزرگ‌تر یا مساوی با u و کوچک‌تر یا مساوی با v است.

حدود بالا و پایین

در این بخش روش‌های تولید حد پایین و بالای موجود برای مسئلهٔ زمان‌بندی یک ماشین پردازندهٔ انباشته با فرض وجود کارهایی با اندازهٔ غیریکسان بررسی می‌شود و در بخش بعد در بدنهٔ روش شاخه و حد به کار گرفته می‌شود.

حدود بالا

در این پژوهش از دو الگوریتم FFLPT[۳۸] (اوزوی، ۱۹۹۴) و BFLPT[۳۹] (دوپونت و جولای، ۱۹۹۸) برای تولید حدود بالا استفاده شده است و کمترین مقدار از بین این دو در مراحل حل مسئله برای حد بالا لحاظ شده است.

حدود پایین

ازجمله مهم‌ترین کاربردهای حدود پایین، استفاده از آنها در بدنهٔ الگوریتم‌های دقیق مانند روش شاخه و حد است. در این مقاله تلاش شده است از حدود پایین LB_1، (اوزوی، ۱۹۹۴)، LB_2 و LB_3، (کاشان و کریمی، ۲۰۱۲) برای تولید حد پایین در الگوریتم شاخه و کران استفاده و نتایج مقایسه شود. درادامه الگوریتم‌های مذکور تشریح می‌شود.

الگوریتم LB_1

روش کار در این الگوریتم به‌صورت زیر خلاصه شده است:

گام۱. ابتدا کلیهٔ کارها را به‌ترتیب نزولیِ زمان پردازش‌شان مرتب کنید.

گام۲. با شروع از ابتدای لیست، نخستین کار را در تنها انباشته‌ای قرار دهید که به‌طور کامل پر نشده است (ظرفیت استفاده‌نشدهٔ آن از B یعنی ظرفیت انباشته کمتر است). اگر همهٔ انباشته‌ها به حد بالای ظرفیت خود رسیده باشند یک انباشتهٔ جدید تشکیل دهید و اگر فضای لازم برای قراردادن کار در انباشتهٔ مذکور وجود ندارد، قسمتی از کار را که ظرفیت خالی انباشته را به‌طور کامل اشغال می‌سازد درون انباشته قرار دهید. سپس یک انباشتهٔ جدید تشکیل و باقیماندهٔ کار خردشده را در آن قرار دهید.

گام۳. این فرآیند را تا زمانی تکرار کنید که هیچ‌کاری در لیست باقی نمانده باشد. بدین ترتیب تعداد انباشته‌های حاصل برابر $\lceil \sum_{j=1}^{n} s_j/B \rceil$ است.

مبنای الگوریتم اشاره‌شده در بالا براساس آزادسازی است. بدین ترتیب که در آن فرض مجازنبودن شکست کارها به اجزا، آزاد شده است. هنگامی‌که اندازهٔ کارها نسبت به ظرفیت ماشین کوچک است، عملکرد متوسط LB_1 خوب تلقی می‌شود؛ درواقع ثابت شده است هرچقدر اندازهٔ کارها نسبت به ظرفیت ماشین کوچک‌تر شود عملکرد LB_1 در بدترین حالت بهتر می‌شود. براساس تجربه نیز ثابت شده است برای بسیاری از مسائلی که در آنها اندازهٔ کارها نسبت به ظرفیت ماشین کوچک است، مقدار حد پایین تولیدشده به‌وسیلهٔ LB_1 با مقدار C^* برابر است؛ اما عملکرد LB_1 زمانی تقلیل می‌یابد که اندازهٔ کارها به نسبت بزرگ باشد. کاشان و کریمی (۲۰۱۲) از این نقطه ضعف برای تولید حدود پایین جدید استفاده و حد پایین LB_2 و LB_3 را معرفی کرده‌اند.

الگوریتم *NLB*

برای دست‌یابی به یک حد پایین روی مقدار C^* به‌ازاءِ مقادیر مختلف ε ($\varepsilon \leq B/2$) تنها اجازه داده می‌شود کارهایی شکست‌پذیر باشد که اندازهٔ آنها در دامنهٔ [ε, B- ε] باشد. فرض کنید $S(u,v) = \{g \mid u < s_g \leq v, g \in J\}$ دلالت بر مجموعه کارهایی داشته باشد که اندازهٔ آنها بزرگ‌تر از u و کوچک‌تر یا مساوی با v است. به‌طور مشابه تعریف می‌شود $\bar{S}(u,v) = \{g \mid u \leq s_g \leq v, g \in J\}$. روش زیر با عنوان *NLB* یک حد پایین معتبر را روی مقدار C^* به دست می‌دهد.

گام۱. به‌ازاءِ $\varepsilon \in [0, B/2]$، یک حد پایین را روی مقدار C^* به‌صورت زیر محاسبه کنید.

$$C^{NLB^{\varepsilon}} = \sum_{j \epsilon s(B-\varepsilon,B)} p_j + C^{LB_1}_{\bar{S}(\varepsilon,B-\varepsilon)} \qquad \text{رابطهٔ ۱}$$

که در آن $C^{LB_1}_{\bar{S}(\varepsilon,B-\varepsilon)}$ مقدار حد پایین به‌دست‌آمده به‌وسیلهٔ LB_1 براساس مجموعه کارهایی است که اندیس آنها متعلق به مجموعهٔ $\bar{S}(\varepsilon, B-\varepsilon)$ است.

گام۲. با تکرار گام ۱ به‌ازاءِ تمامی مقادیر ε، بزرگ‌ترین مقدار به‌دست‌آمده برای $C^{NLB^{\varepsilon}}$ یک حد پایین روی مقدار C^* است؛ یعنی:

$$C^{NLB} = max_{\varepsilon \epsilon [0,B/2]}\{C^{NLB^{\varepsilon}}\} \qquad \text{رابطهٔ ۲}$$

این الگوریتم یک حد پایین معتبر روی مقدار C^* به دست می‌دهد. همچنین ثابت شده است مقدار حد پایینِ حاصل از الگوریتم *NLB* بیشتر از حد پایین تولیدشده به‌وسیلهٔ الگوریتم LB_1 است؛ اما درادامه حد LB_2 معرفی می‌شود که عملکرد آن از هر دو الگوریتم *NLB* و LB_1 بهتر است؛ بنابراین پس از معرفی این حد از آن برای حد پایین در روش شاخه و کران استفاده می‌شود.

الگوریتم LB_2

ازآنجاکه دو کاری که اندازهٔ آنها در بازهٔ $[B/2, B-\mathcal{E}]$ قرار می‌گیرد نباید در یک انباشته باشند، هنگامی‌که $C^{LB_1}_{\bar{S}(\varepsilon,B-\varepsilon)} < \sum_{j \in S(B/2,B-\varepsilon)} p_j$، ممکن است مقدار حد C^{NLB} قوی‌تر شود؛ ازاین‌رو حد C^{LB_2} به‌صورت زیر تعریف می‌شود.

$$C^{LB_2} = \max_{\varepsilon \epsilon [0,B/2]} \left\{ \sum_{j \epsilon s(B-\varepsilon,B)} p_j + max \left\{ \sum_{j \in S(B/2,B-\varepsilon)} p_j , C^{LB_1}_{\bar{S}(\varepsilon,B-\varepsilon)} \right\} \right\} =$$
$$\max_{\varepsilon \epsilon [0,B/2]} \left\{ \sum_{j \epsilon s(B/2,B)} p_j + max \left\{ 0, C^{LB_1}_{\bar{S}(\varepsilon,B-\varepsilon)} - \sum_{j \in S(B/2,B-\varepsilon)} p_j \right\} \right\} = \qquad \text{رابطهٔ ۳}$$
$$max\{C^*_{S(B/2,B),}, \max_{\varepsilon \epsilon [0,B/2]}\{C^{NLB^{\varepsilon}}\}\}$$

که در آن $C^*_{S(B/2,B),} = \sum_{j \epsilon s(B/2,B)} p_j$

الگوریتم LB_3

با استفاده از ایدهٔ تفکیک کارها درقالب سه مجموعه، حد پایین *LB3* به‌صورت زیر ارائه شده است.

$$C^{LB_3} = max \left\{ C^*_{S(B/3,B),} \max_{\varepsilon \epsilon [0,B/3]} \{C^{NLB^{\varepsilon}}\} \right\} \qquad \text{رابطهٔ ۴}$$

که در آن، $C^*_{S(B/3,B),}$ مقدار بهینهٔ C_{max} در زیرمسئله‌ای از مسئلهٔ اصلی است که فقط کارهای با اندازهٔ بزرگ‌تر از $B/3$ در نظر گرفته شده است.

طبق تعریف، مجموعه کارهایی که اندیس آنها در مجموعهٔ $S(B/3,B)$ قرار دارد درقالب سه زیرمجموعه تفکیک می‌شود؛ نخستین زیرمجموعه به‌صورت $S(2B/3,B)$، دومین زیرمجموعه به‌صورت $S(B/2,2B/3)$ و سومین زیرمجموعه به‌صورت $S(B/3,B/2)$ است. همچنین مجموعهٔ x به‌این‌صورت تعریف می‌شود؛ $X = S(B/3,B/2) \cup H$ که در آن H مجموعه‌ای از زیرمجموعهٔ دوم است و می‌تواند با یکی از کارهای مجموعهٔ سوم گروه‌بندی شوند. بدین ترتیب باتوجه‌به ترکیب‌بندی کارها به‌صورت بالا می‌توان $C^*_{S(B/3,B)}$ را به‌صورت رابطهٔ زیر ارائه کرد.

رابطهٔ ۵ $$C^*_{S(B/3,B)} = \sum_{j\in s(B/2,B)} P_j + C^*_X - \sum_{j\in H} P_j$$

بنابراین با محاسبهٔ مقدار C^*_X می‌توان به مقدار C^{LB_3} رسید. درنهایت C^{LB_3} به‌صورت رابطهٔ زیر نمایش داده می‌شود.

رابطهٔ ۶
$$C^{LB_3} = max_{\varepsilon\in[0,B/3]}\left\{\sum_{j\in S(B-\varepsilon,B)} p_j + max\left\{\sum_{j\in s(B/2,B-\varepsilon)} P_j + C^*_X - \sum_{j\in H} P_j, C^{LB_1}_{S(\varepsilon,B-\varepsilon)}\right\}\right\} = max_{\varepsilon\in[0,B/3]}\left\{\sum_{j\in s(B/2,B)} P_j + C^*_X - \sum_{j\in H} P_j + max\left\{0, \left(C^{LB_1}_{S(\varepsilon,B-\varepsilon)} - \sum_{j\in S\left(\frac{B}{2},B-\varepsilon\right)} P_j - C^*_X + \sum_{j\in H} P_j\right)\right\}\right\}$$

که مقدار C^*_X ازطریق تبدیل مسئلهٔ زمان‌بندی مربوطه به یکی از مسائل شناخته‌شده در نظریهٔ گراف (مسئلهٔ حداکثر تطابق وزنی) به دست می‌آید. لازم به ذکر است در محاسبهٔ الگوریتم LB_3 باید یک مسئلهٔ حداکثر تطابق وزنی حل شود که در پژوهش قبل از جعبهٔ بهینه‌سازی موجود در نرم‌افزار متلب استفاده شده است و مبتنی‌بر روش شاخه و حد عمل می‌کند؛ اما در این پژوهش از الگوریتم ادموند (گابو[۴۰]، ۱۹۷۶) برای حل این مسئله استفاده شده است. این الگوریتم قادر است در زمان چندجمله‌ای مسئلهٔ مدنظر را حل کند؛ بنابراین در زمان حل الگوریتم بهبود حاصل شده است.

روش شاخه و حد

در این بخش برای محاسبهٔ حداقل C_{max} در مسئلهٔ زمان‌بندی یک ماشین پردازندهٔ انباشته با اندازهٔ کارهای غیریکسان، الگوریتم شاخه و کرانی به کار گرفته شده است که برمبنای روش دوپونت و فیلیپو (۲۰۰۲) است. برای شاخه‌زدن در الگوریتم شاخه و کران، در هر مرحله باید دو تصمیم گرفته شود؛ نخست کار انتخابی باید به انباشتهٔ جاری اضافه شود یا در انباشتهٔ جدید قرار بگیرد و دوم اینکه مشخص شود کدام کار باید به انباشته اضافه شود. همچنین دو روش برای اضافه‌کردن یک کار جدید به جواب جزئی (p) وجود دارد که عبارتند از:

۱- نوع یک: در این نوع، کار جدید در یک انباشتهٔ جدید قرار می‌گیرد.

۲- نوع دو: در این نوع، کار جدید به انباشتهٔ موجود در P اضافه می‌شود، به‌طوری‌که انباشته ظرفیت پذیرش کار جدید را داشته باشد.

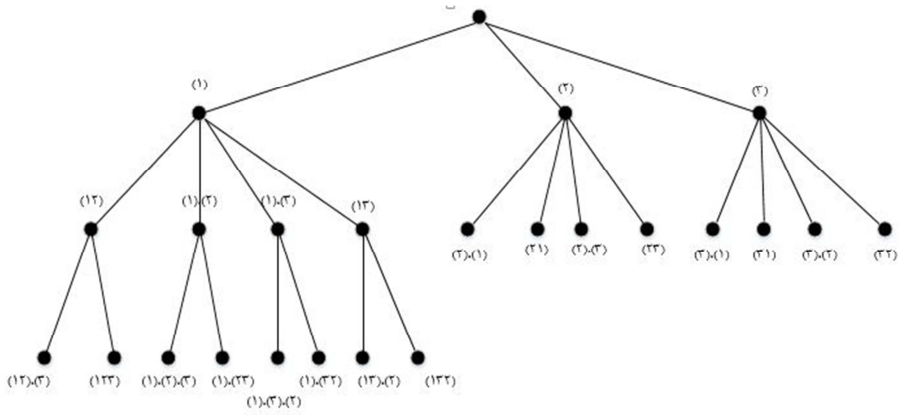
در سطح نخست از درخت شاخه و کران، در هر جواب جزئی P، دقیقاً یک کار در یک انباشته قرار می‌گیرد. زمانی‌که یک گره برای شاخه‌زدن انتخاب می‌شود، فهرستی از کارهای موجود که هنوز زمان‌بندی نشده‌اند برای

اضافه‌شدن به P در نظر گرفته می‌شود. یک مثال از نحوهٔ شاخه‌زدن یک مسئله با تعداد n=۳ کار با اندازهٔ یک و B=۳ در شکل ۱ آورده شده است. سطح نخست از درخت تنها شامل گره‌های نوع یک است که درواقع هر کار در انباشتهٔ مربوط به خود قرار می‌گیرد. در سطح بعدی درخت، گره‌های نوع یک و نوع دو تولید هستند. کارهایی که با هم در یک پرانتز قرار دارند نشان‌دهندهٔ کارهایی است که در یک انباشته با هم پردازش می‌شوند. باتوجه‌به اینکه ترتیب قرارگرفتن کارها در انباشته‌ها اهمیتی ندارد و کارهای موجود در یک انباشته به‌صورت هم‌زمان پردازش می‌شوند و زمان شروع و پایان یکسانی دارند، تعداد زیادی از جواب های جزئی ایجادشده تکراری هستند؛ برای مثال دو جواب جزئی ((۲۳)(۱)) و ((۳۲)(۱)) یکی هستند. برای افزایش کارایی الگوریتم از سه نکتهٔ اشاره‌شده درادامه استفاده شود تا از این تکرارها در الگوریتم شاخه و کران اجتناب کرد (دوپونت و فیلیپو، ۲۰۰۲):

الف) ابتدا کارها را براساس زمان پردازششان از بیشتر به کمتر(به‌صورت نزولی) مرتب کنید.

ب) برای شاخه‌زدن، درصورتی‌که گره فرزند از نوع دوم باشد باید زمان پردازش کار جدیدی که قرار است به انباشته اضافه شود از زمان پردازش مابقی کارهایِ موجود در انباشته کمتر یا مساوی باشد.

ج) در نظر نگرفتن جواب‌های جزئی که انباشته‌هایی با کارهای یکسان دارند؛ برای مثال دو جواب جزئی ((۱)(۳۲)) و ((۳۲)(۱)) یکی هستند و می‌توان یکی از آنها را در نظر نگرفت.



شکل ۱- نحوهٔ شاخه‌زدن در درخت شاخه و کران

در روش شاخه و کران برای رسیدن به جواب بهینه به حدود بالا و پایین احتیاج است. هرچه مقدار این حدود بالا و پایین به یکدیگر نزدیک‌تر باشد، این روش سریع‌تر به جواب می‌رسد و تعداد گره‌های کمتری بررسی می‌شود. در اینجا برای محاسبهٔ مقدار اولیه از هر دو روش ابتکاری ***FFLPT*** و ***BFLPT*** استفاده شده است و کمترین مقدار از بین آنها برای یک حد بالا در مسئله استفاده می‌شود. برای توصیف این الگوریتم، اگر مجموعه کارهای زمان‌بندی‌نشده مجموعهٔ **j** نامیده شود، برای به دست آوردن یک حد پایین روی مجموعهٔ ***j*** می‌توان از الگوریتم تولید حدود پایین ذکرشده در بخش قبل استفاده کرد. مقدار حد پایین روی مجموعهٔ **j**، ***BInf(j)*** نامیده می‌شود (اگر ***j*** مجموعهٔ کل کارها باشد، این حد پایین با ***Cmin*** نشان داده می‌شود). فرض کنید ***Valbest*** مقدار بهترین جواب شناخته‌شده و ***Valsol*** جواب جزئی مسئلهٔ زمان‌بندی روی انباشتهٔ نخست تا انباشتهٔ فعلی باشد، دراین‌صورت اگر

$valsol + BInf(j) \geq valbest$، می‌توان این گره را قطع کرد؛ چون جواب بهتری تولید نمی‌کند؛ درغیراین‌صورت با استفاده از یک الگوریتم حریصانه جوابی شدنی روی مجموعهٔ j تولید کنید و آن را *valgreedy* نام‌گذاری کنید (در اینجا از *FFlPT* استفاده شده است). درصورتی‌که $valsol + valgreedy \leq valbest$ باشد، درواقع یک بهترین جواب جدید به دست آمده است. همچنین اگر $Cmin = valsol + valgreedy$ جواب بهینه روی j به دست آمده باشد، می‌توان عملیات را در این گره به پایان رساند. در حالت کلی در دو حالت الگوریتم به پایان می‌رسد؛ نخست اینکه جواب بهینه برای مسئله پیدا شود و یا زمان در نظر گرفته شده برای حل مسئله به پایان برسد. در روش حاضر برای تولید حد پایین از الگوریتم‌های LB_1 و LB_2 و LB_3 استفاده و نتایج آنها با هم مقایسه شده است. تجربه ثابت کرده است این الگوریتم‌ها بسیار سریع هستند و جواب‌های نزدیک به بهینه را تولید می‌کنند. درادامه برای بررسی کیفیت جواب‌های حاصل‌شده، درصورت استفاده از حدود پایین LB_1 و LB_2 و LB_3 و انجام مقایسات، یک برنامهٔ کامپیوتری در نرم‌افزار متلب نوشته و آزمایش‌ها روی یک رایانه با RAM ۴ Gb و CPU ۲/۴۰ GHz انجام شده است.

بحث و نتایج محاسباتی

برای ارزیابی عملکرد حدود پایین ارائه‌شده و به دست آوردن جواب بهینه برای مسئلهٔ مدنظر، دسته‌ای از نمونه مسائل به‌صورت تصادفی در نرم‌افزار متلب تولید شده است و روش‌های ارائه‌شده در بخش قبل روی این داده‌ها آزمایش شده‌اند. ابتدا داده‌های استفاده‌شده بررسی و سپس نتایج محاسبات درقالب جداول و شکل‌ها در بخش بعد ارائه می‌شوند.

برای انجام آزمایش‌های محاسباتی، ۶ مجموعه مسئله در نظر گرفته شده است. برای هر نمونه مسئله دو سطح از زمان پردازش و ۵ سطح از اندازهٔ کارها بررسی می‌شود. همچنین برای هر مجموعه، ۱۰ نمونه مسئله به‌صورت تصادفی تولید شده است؛ درنتیجه تعداد مسائلی که در هر بخش آزمایش می‌شود برابر ۶۰۰ است. این مسائل در جدول ۱ به‌تفصیل آورده شده است.

جدول ۱- دسته‌بندی مسائل

شمارهٔ مسئله	اندازهٔ کارها	ظرفیت ماشین	اندازهٔ کارها
۱	[۱-۱۰]	۱۰	n=۲۰و۴۰و۶۰و۸۰و۱۰۰
۲	[۴-۸]	۱۰	n=۲۰و۴۰و۶۰و۸۰و۱۰۰
۳	[۱-۵]	۱۰	n=۲۰و۴۰و۶۰و۸۰و۱۰۰
۴	[۲-۴]	۱۰	n=۲۰و۴۰و۶۰و۸۰و۱۰۰
۵	[۱-۵]	۵	n=۲۰و۴۰و۶۰و۸۰و۱۰۰
۶	[۲-۴]	۵	n=۲۰و۴۰و۶۰و۸۰و۱۰۰

برای انجام آزمایش‌های محاسباتی، مجموعه مسائل مطرح‌شده در جدول ۱) در نظر گرفته و مقادیر جواب بهینه برای آنها حساب شده است. درادامه نتایج محاسبات درقالب جدول ۲ تا ۵ آورده می‌شوند.

جدول ۲- شاخه و کران B=10 ,pj∈[1,10]

		B&B(LB_1)						B&B(LB_2)					B&B (LB_3)				
s_j	n	#Opt	Avg. nodes	Avg .times	Gap (%)	Lb_1	Ub	#Opt	Avg. nodes	Avg .times	Gap (%)	Lb_2	#Opt	Avg .node	Avg .times	Gap (%)	Lb3
[۱-۱۰]	۲۰	۱۰	۱۷۳	۰/۲۱	۰	۶۲/۵۰	۶۷/۵۰	۱۰	۴۴	۰/۰۷	۰	۶۵/۹۰	۱۰	۴۴	۰/۱۳	۰	۶۶
	۴۰	۱۰	۲۸۵۷۲	۶۷/۰۹	۰	۱۳۰/۴۰	۱۳۸/۸۱	۱۰	۱۸۵	۵/۱۰	۰	۱۳۴/۱۰	۱۰	۱۳۰۴	۱۰/۵۹	۰	۱۳۴/۷۰
	۶۰	۵	۴۹۰۱۹۰	۱۱۰۳/۸۰	۰	۱۷۹/۹۰	۱۹۱/۰۰	۱۰	۲۱۶۵۷	۶۳/۴۸	۰	۱۸۲/۰۰	۱۰	۱۲۹۹۹	۱۲۱/۲۲	۰	۱۸۲/۹۰
	۸۰	۰	۵۱۸۵۸۰	۱۷۹۳/۳۰	۶/۶۳	۲۴۸/۰۰	۲۶۵/۵۰	۸	۱۱۶۶۰۰	۴۷۱/۸۲	۱/۰۸	۲۵۹/۸۰	۷	۶۹۹۸۴	۷۳۶/۲۰	۱/۰۵	۲۵۹/۹۰
	۱۰۰	۰	۴۵۲۴۶۲	۱۷۹۵/۱۰	۵/۴۸	۳۲۲/۷۰	۹/۰۰	۸	۱۱۷۶۰۰	۶۵۷/۶۱	۰/۹۷	۳۳۳/۷۰	۷	۳۴۵۰۵	۸۷۶/۶۰	۰/۷۲	۳۳۴/۷۰
[۴-۸]	۲۰	۱۰	۴۶	۰/۰۶	۰	۷۲/۴۰	۸۶/۶۲	۱۰	۵	۰/۰۲	۰	۸۵/۱۰	۱۰	۳	۰/۰۴	۰	۸۶
	۴۰	۱۰	۶۵۷	۱/۴۳	۰	۱۳۹/۸۰	۱۶۸/۳۰	۱۰	۸۵	۰/۲۹	۰	۱۶۴/۷۰	۱۰	۲۰	۰/۷۵	۰	۱۶۶/۲۰
	۶۰	۱۰	۳۳۰۰۲	۱۰۰/۴۲	۰	۲۰۲/۲۰	۲۴۶/۷۱	۱۰	۹۷	۰/۷۷	۰	۲۴۱/۱۰	۱۰	۲۳	۳/۹۴	۰	۲۴۳/۳۰
	۸۰	۹	۹۱۸۹۲	۳۵۶/۶۰	۱۸/۰۱	۲۶۹/۶۰	۳۲۱/۰۰	۱۰	۲۲۵۲	۱۵/۳۰	۰	۳۱۴/۹۰	۱۰	۳۵	۷/۱۸	۰	۳۱۸/۳
	۱۰۰	۸	۱۹۰۱۹۰	۶۸۰/۱۴	۱۹/۹۲	۳۴۶/۴۰	۴۱۹/۲۲	۱۰	۲۲۶۵۱	۱۵۳/۱۶	۰	۴۱۱/۸۰	۱۰	۵۰	۲۰/۱۲	۰	۴۱۵/۵۰
[۱-۵]	۲۰	۱۰	۲۹	۰/۰۵	۰	۳۶/۶۰	۳۷/۱۰	۱۰	۲۹	۰/۰۷	۰	۳۶/۶۰	۱۰	۲۹	۰/۱۷	۰	۳۶/۶۰
	۴۰	۱۰	۱۴۱	۰/۳۳	۰	۶۹/۶۰	۷۰/۵۱	۱۰	۱۴۱	۰/۴۳	۰	۶۹/۶۰	۱۰	۱۴۱	۳/۴۷	۰	۶۹/۶۰
	۶۰	۱۰	۲۷۱	۱/۲۳	۰	۱۰۷/۰۰	۱۰۸/۴۳	۱۰	۲۷۱	۱/۷۱	۰	۱۰۷/۰۰	۱۰	۲۷۱	۱۳/۸۸	۰	۱۰۷
	۸۰	۱۰	۱۶۲۰	۶/۶۷	۰	۱۳۵/۷۰	۱۳۶/۹۰	۱۰	۱۶۲۰	۹/۸۵	۰	۱۳۵/۷۰	۱۰	۱۶۲۰	۱۳۷/۲۲	۰	۱۳۵/۷۰
	۱۰۰	۱۰	۵۳۴	۳/۸۵	۰	۱۶۷/۶۰	۱۶۸/۴۵	۱۰	۵۳۴	۵/۶۹	۰	۱۶۷/۶۰	۱۰	۵۳۴	۶۶/۶۹	۰	۱۶۷/۶۰
[۲-۴]	۲۰	۱۰	۲۷	۰/۰۷	۰	۳۶/۱۰	۳۶/۹۲	۱۰	۲۷/۳۱	۰/۰۹	۰	۳۶/۱۰	۱۰	۲۷	۰/۱۵	۰	۳۶/۱۰
	۴۰	۱۰	۱۵۶۸	۳/۵۵	۰	۷۰/۹۰	۷۳/۸۱	۱۰	۱۵۶۸	۴/۵۵	۰	۷۰/۹۰	۱۰	۱۵۶۵	۸/۷۷	۰	۷۰/۹۰
	۶۰	۱۰	۴۶۳۳	۱۹/۶۸	۰	۱۰۳/۵۰	۱۰۷/۲۰	۱۰	۴۶۳۳	۲۳/۹۷	۰	۱۰۳/۵۰	۱۰	۴۶۳۳	۱۴۸/۶۹	۰	۱۰۳/۵۰
	۸۰	۷	۲۰۸۱۶۰	۵۶۸/۹۴	۰/۵۰	۱۳۶/۳۰	۱۳۹/۹۱	۷	۱۵۹۸۹۰	۵۷۹/۱۷	۰/۵۰	۱۳۶/۳۰	۷	۸۹۳۳۲	۶۲۴/۱۸	۲/۰۳	۱۳۶/۳۰
	۱۰۰	۷	۱۰۹۲۰۰	۶۸۸/۵۰	۰/۴۵	۱۷۶/۱۰	۱۷۴/۹۰	۷	۸۶۲۱۵	۷۵۴/۶۴	۰/۵۰	۱۷۶/۱۰	۶	۲۳۲۹۱	۹۱۵/۴۰	۰/۶۲	۱۷۶/۱۰

جدول ۳- روش شاخه و کران B=5 ,pj∈[1,10]

		B&B(LB_1)						B&B(LB_2)					B&B (LB_3)				
s_j	n	#Opt	Avg. nodes	Avg .times	Gap (%)	Lb1	Ub	#Opt	Avg .nodes	Avg .times	Gap (%)	Lb2	#Opt	Avg.node	Avg .times	Gap (%)	Lb3
[۱- ۵]	۲۰	۱۰	۴۵	۰/۰۹	۰	۶۷/۰۰	۷۰/۲۰	۱۰	۲۴	۰/۰۶	۰	۶۸/۴۰	۱۰	۲۲	۰/۰۹	۰	۶۸/۸۰
	۴۰	۱۰	۲۱۳	۰/۶۱	۰	۱۳۵/۴۰	۱۴۵/۲۰	۱۰	۴۷	۰/۲۱	۰	۱۴۲/۹۰	۱۰	۳۷	۰/۵۵	۰	۱۴۶/۶۰
	۶۰	۱۰	۲۲۳۷۴	۶۸/۷۲	۰	۲۰۱/۶۰	۲۱۱/۱۰	۱۰	۵۸۸	۳/۴۳	۰	۲۰۳/۹۰	۱۰	۴۰۶	۱۰/۵۴	۰	۲۰۵
	۸۰	۹	۱۰۶۶۷۰	۳۹۴/۴۰	۱/۰۹	۲۶۰/۰۰	۲۶۹/۰۰	۱۰	۹۹۱۷	۶۴/۳۴	۰	۲۶۲/۱۰	۱۰	۴۱۱۲	۱۰۹/۲۹	۰	۲۶۳/۶۰
	۱۰۰	۵	۲۴۴۸۴۰	۱۰۴۴	۲/۶۹	۳۲۱/۸۰	۳۳۴/۰۰	۹	۴۸۹۶۵	۳۳۰/۲۹	۱/۰۴	۳۲۵/۴۰	۹	۶۵۲۸	۴۶۶/۴۸	۱/۰۴	۳۲۷/۶۰
[۲-۴]	۲۰	۱۰	۲۷	۰/۰۷	۰	۷۱/۲۰	۸۲/۲۰	۱۰	۶	۰/۰۲	۰	۸۰/۳۰	۱۰	۳	۰/۱۲	۰	۸۲/۲۰
	۴۰	۱۰	۲۲۴	۰/۶۴	۰	۱۳۹/۱۰	۱۶۳/۱۰	۱۰	۳۱	۰/۱۷	۰	۱۵۸/۰۰	۱۰	۹	۰/۷۹	۰	۱۵۹/۹۰
	۶۰	۱۰	۲۹۷۴	۸/۲۳	۰	۱۹۶/۷۰	۲۲۶/۱۰	۱۰	۷۶۵	۲/۷۵	۰	۲۱۹/۱۰	۱۰	۳۸	۵/۸۰	۰	۲۲۲/۸۰
	۸۰	۱۰	۱۳۵۷۹	۴۵/۹۱	۰	۲۷۰/۱۰	۳۱۴/۳۰	۱۰	۱۴۰۵	۹/۵۲	۰	۳۰۴/۱۰	۱۰	۴۷	۱۵/۴۳	۰	۳۰۸/۸۰
	۱۰۰	۱۰	۲۳۹۹۸	۶۷/۶۶	۰	۳۴۴/۰۰	۳۹۸/۸۰	۱۰	۳۳۴۱	۱۵/۸۴	۰	۳۹۰/۸۰	۱۰	۵۰	۲۴/۶۹	۰	۳۹۴/۳۰

جدول ۴- روش شاخه و کران B=5, pj∈[1,5]

		B&B(LB_1)						B&B(LB_2)					B&B (LB_3)				
s_j	n	#Opt	Avg. nodes	Avg .times	Gap (%)	Lb_1	Ub	#Opt	Avg. nodes	Avg. times	Gap (%)	Lb_2	#Opt	Avg. node	Avg .times	Gap (%)	Lb_3
[۱-۵]	۲۰	۱۰	۴۹	۰/۱۱	۰	۳۸/۷۰	۴۱/۹۰	۱۰	۱۵	۰/۰۴	۰	۴۰/۹۰	۱۰	۱۵	۰/۰۹	۰	۴۰/۹۰
	۴۰	۱۰	۲۳۹	۰/۷۴	۰	۷۰/۶۰	۷۳/۳۰	۱۰	۴۸	۰/۲۶	۰	۷۱/۶۰	۱۰	۳۳	۰/۵۴	۰	۷۱/۹۰
	۶۰	۱۰	۲۴۵۰۱	۹۹/۶۰	۰	۱۱۰/۴۰	۱۱۸/۱۰	۱۰	۲۱۴	۱/۴۱	۰	۱۱۴/۸۰	۱۰	۱۴۳	۴/۶۲	۰	۱۱۵/۱۰
	۸۰	۸	۱۳۳۰۷۰	۵۳۹/۱۴	۵/۲۰	۱۴۵/۴۰	۱۵۴/۱۰	۱۰	۱۸۶۷	۱۴/۷۴	۰	۱۵۱/۴۰	۱۰	۴۵۸	۲۴/۲۳	۰	۱۵۲/۲۰
	۱۰۰	۷	۲۳۰۰۸۰	۹۸۴/۷۰	۴/۴۶	۱۸۱/۳۰	۱۹۲/۰۰	۱۰	۱۵۹۰	۱۱/۷۲	۰	۱۷۸/۴۰	۱۰	۲۳۷	۱۹/۶۵	۰	۱۸۸/۶۰
[۲-۴]	۲۰	۱۰	۲۷	۰/۰۷	۰	۳۷/۵۰	۴۳/۲۰	۱۰	۵	۰/۰۲	۰	۴۲/۳۰	۱۰	۲	۰/۰۷	۰	۴۲/۹۰
	۴۰	۱۰	۲۳۴	۰/۶۶	۰	۷۷/۲۰	۸۸/۹۰	۱۰	۳۳	۰/۱۵	۰	۸۷/۷۰	۱۰	۴	۰/۳۲	۰	۸۸/۲۰
	۶۰	۱۰	۱۱۳۵	۳/۳۸	۰	۱۰۵/۸۰	۱۲۲/۴۰	۱۰	۱۶۰	۰/۹۵	۰	۱۱۹/۹۰	۱۰	۲۱	۳/۰۴	۰	۱۲۱/۲۰
	۸۰	۱۰	۱۷۷۱۸	۳۶/۳۶	۰	۱۴۶/۵۰	۱۶۸/۲۰	۱۰	۸۱۸	۰/۶۰	۰	۱۶۵/۵۰	۱۰	۱۴	۳/۵۹	۰	۱۶۷/۰۰
	۱۰۰	۸	۱۰۳۴۷۰	۴۹۳/۱۷	۱۴/۴۰	۱۸۰/۵۰	۲۱۰/۴۰	۱۰	۵۷۷۵	۶۷/۲۰	۰	۲۰۴/۱۰	۱۰	۸۷	۲۸/۵۳	۰	۲۰۶/۸۰

جدول ۵- روش شاخه و کران pj∈[1,5],B=10

		B&B(LB_1)						B&B(LB_2)					B&B (LB_3)				
s_j	n	#Opt	Avg. nodes	Avg .times	Gap (%)	Lb_1	Ub	#Opt	Avg. nodes	Avg. times	Gap (%)	Lb_2	#Opt	Avg. node	Avg. times	Gap (%)	Lb_3
[۱-۱۰]	۲۰	۱۰	۱۹۸	۰/۲۸	۰	۳۹/۲۰	۴۲/۲۰	۱۰	۱۵	۰/۰۴	۰	۴۱/۲۰	۱۰	۱۴	۰/۰۶	۰	۴۱/۲۰
	۴۰	۱۰	۱۹۶۱	۵/۱۰	۰	۶۷/۶۰	۷۱/۷۰	۱۰	۱۳۹	۰/۵۰	۰	۶۹/۷۰	۱۰	۱۳۷	۱/۲۳	۰	۶۹/۹۰
	۶۰	۲	۵۱۶۷۳۰	۱۵۵۲/۵۰	۵/۲۰	۹۸/۸۰	۱۰۵/۵۰	۹	۹۲۳۱۴	۳۰۲/۸۰	۱/۴۰	۱۰۲/۲۰	۹	۴۴۲۸۵	۳۵۴/۰۸	۱/۲۰	۱۰۲/۵۰
	۸۰	۰	۵۱۰۳۹۹	۱۷۷۶/۸۰	۵/۳۴	۱۳۷/۶۰	۱۴۶/۳۰	۱۰	۸۴۰۷۱	۴۶۹/۶۰	۰	۱۴۰/۷۰	۸	۴۰۰۴۶	۶۱۵/۶۰	۱/۷۱	۱۴۱/۲۰
	۱۰۰	۰	۵۱۰۶۶۰	۱۷۹۹/۹۰	۶/۱۵	۱۶۶/۲۰	۱۷۸/۱۰	۹	۷۳۳۱۶	۴۷۵/۸۰	۱/۰۶	۱۷۲/۵۰	۸	۲۱۲۰۹	۵۳۰/۹۰	۰/۷۳	۱۷۳/۱۰
[۴-۸]	۲۰	۱۰	۸۸	۰/۱۷	۰	۳۶/۰۰	۴۴/۷۰	۱۰	۲۰	۰/۰۶	۰	۴۳/۵۰	۱۰	۱۰	۰/۱۸	۰	۴۳/۷۰
	۴۰	۱۰	۳۸۶۴	۷/۴۹	۰	۷۳/۴۰	۸۸/۴۰	۱۰	۷۹	۰/۳۶	۰	۸۶/۷۰	۱۰	۱۷	۰/۷۹	۰	۸۷/۴۰
	۶۰	۱۰	۱۱۲۰	۳/۰۵	۰	۱۱۳/۱۰	۱۳۶	۱۰	۲۰	۰/۱۴	۰	۱۳۵/۳۰	۱۰	۱۶	۱/۵۵	۰	۱۳۵/۶۰
	۸۰	۸	۱۵۳۱۶۰	۴۰۳/۱۲	۱۷/۱۶	۱۴۵/۵۰	۱۷۱/۵۰	۱۰	۱۱۹۸۳	۹۰/۱۳	۰	۱۶۹/۲۰	۱۰	۲۶	۶/۵۷	۰	۱۷۰/۴۰
	۱۰۰	۶	۲۰۰۲۸۰	۷۶۵/۰۴	۱۸/۵۷	۱۸۳/۶۰	۲۲۰/۵۰	۱۰	۳۲۸۱	۲۷/۱۴	۰	۲۱۶/۰۰	۱۰	۷۷	۲۵/۶۹	۰	۲۱۷/۶۰
[۱-۵]	۲۰	۱۰	۱۲	۰/۰۲	۰	۲۰/۹۰	۲۱/۰۰	۱۰	۱۲	۰/۰۴	۰	۲۰/۹۰	۱۰	۱۲	۰/۱۵	۰	۲۰/۹۰
	۴۰	۱۰	۱۴۵	۰/۳۹	۰	۴۰/۵۰	۴۱/۱۰	۱۰	۱۴۵	۰/۵۴	۰	۴۰/۵۰	۱۰	۱۴۲	۳/۶۹	۰	۴۰/۵۰
	۶۰	۱۰	۲۲۷	۱/۱۲	۰	۵۶/۴۰	۵۷/۰۰	۱۰	۲۲۷	۱/۵۶	۰	۵۶/۴۰	۱۰	۲۲۷	۹/۴۸	۰	۵۶/۴۰
	۸۰	۱۰	۲۸۱	۱/۷۸	۰	۷۲/۱۰	۷۲/۷۰	۱۰	۲۸۱	۲/۶۳	۰	۷۲/۱۰	۱۰	۲۸۱	۲۸/۸۰	۰	۷۲/۱۰
	۱۰۰	۱۰	۵۰	۰/۶۰	۰	۸۹/۱۰	۸۹/۳۰	۱۰	۵۰	۰/۹۵	۰	۸۹/۱۰	۱۰	۵۰	۱۲/۶۵	۰	۸۹/۱۰
[۲-۴]	۲۰	۱۰	۶	۰/۰۰	۰	۱۹/۷۰	۱۹/۹۰	۱۰	۶	۰/۰۱	۰	۱۹/۷۰	۱۰	۶	۰/۰۳	۰	۱۹/۷۰
	۴۰	۱۰	۸۳	۰/۳۷	۰	۳۷/۱۰	۳۸/۲۰	۱۰	۸۳	۰/۴۸	۰	۳۷/۱۰	۱۰	۸۳	۱/۱۲	۰	۳۷/۱۰
	۶۰	۱۰	۶۷۴۱۶	۱۱۳/۸۸	۰	۵۵/۷۰	۵۷/۶۰	۱۰	۶۷۴۱۶	۱۵۵/۴۰	۰	۵۵/۷۰	۱۰	۴۴۹۹۳	۱۶۷/۷۰	۰	۵۵/۷۰
	۸۰	۹	۲۷۵۵۸	۱۸۹/۱۴	۰/۲۵	۷۳/۸۰	۷۵/۸۰	۹	۱۹۷۹۱	۱۹۳/۶۵	۰/۲۵	۷۳/۸۰	۹	۴۹۵۲	۲۸۶/۳۰	۰/۲۵	۷۳/۸۰
	۱۰۰	۸	۱۹۹۹۸۰	۳۶۰/۵۰	۰/۶۱	۹۵/۶۰	۹۸/۱۰	۸	۱۵۷۸۷۰	۳۶۳/۲۰	۰/۶۱	۹۵/۶۰	۸	۵۸۸۲۳	۳۹۳/۲۰	۰/۶۱	۹۵/۶۰

در این جداول s_j نشان‌دهندهٔ اندازهٔ کارها و *#opt* نمایانگر تعداد مسائلی است که جواب بهینهٔ آنها در زمان کمتر از ۱۸۰۰ ثانیه به دست آمده است. همچنین *Avg. Nodes* میانگین تعداد گره‌های بررسی‌شده برای رسیدن به جواب بهینه و *Avg times* میانگین زمان اجرای برنامه برای ده نمونه مسئله را نشان می‌دهد. *Avg (LB1)* و *Avg (LB_2)* و *Avg.(LB_3)* به‌ترتیب نشانگر میانگین حدود پایین LB_1، LB_2، LB_3 و *Avg (UB)* میانگین حد بالا برای ده نمونه مسئله هستند. لازم به ذکر است، Gap که میانگین شکاف بهینگی نامیده می‌شود، برای مسائلی محاسبه می‌شود که جواب بهینه در زمان ۱۸۰۰ ثانیه به دست نمی‌آید. میانگین شکاف بهینگی از رابطهٔ زیر حاصل می‌شود:

$$\text{Gap} = \text{AVG}((C^{opt} - C^{LB})/C^{LB}) * 100 \qquad \text{رابطهٔ ۷}$$

در جدول ۲) ظرفیت انباشته برابر ۱۰ و زمان پردازش کارها به‌طور تصادفی در بازهٔ [۱۰-۱] در نظر گرفته شده است. همان‌طورکه مشاهده می‌شود زمانی‌که اندازهٔ کارها بزرگ باشد، زمان اجرای الگوریتم شاخه و حد نیز افزایش می‌یابد. هنگامی‌که از حد پایین LB_1 در الگوریتم استفاده شود، افزایش زمان اجرا بیشتر است؛ زیرا ازآنجایی‌که حد پایین LB_1 نسبت به LB_2 و LB_3 عمکرد ضعیف‌تری دارد، زمان اجرای الگوریتم نسبت به حالتی‌که از حد پایین LB_2 و LB_3 استفاده شود به‌طور چشم‌گیری افزایش پیدا می‌کند. بیشترین زمان در نظر گرفته شده برای اجرای این الگوریتم ۱۸۰۰ ثانیه است که در این زمان یا الگوریتم جواب بهینه را پیدا می‌کند یا با بهترین جواب پیداشده در این زمان متوقف می‌شود. باتوجه‌به نتایج جدول نتیجه می‌شود وقتی اندازهٔ کارها در بازهٔ [۱۰-۱] باشد و تعداد کارها ۶۰ انتخاب شود، عملکرد LB_1 کاهش شدیدی می‌یابد و الگوریتم شاخه و کران زمان اجرای بیشتری می‌طلبد؛ درنتیجه در زمان تعیین‌شده این الگوریتم فقط قادر به حل بهینهٔ نیمی از مسائل است. درصورتی‌که اگر از حدود پایین LB_2 و LB_3 در بدنهٔ روش شاخه و کران استفاده شود، برای هر ده نمونه مسائل، جواب بهینه به دست می‌آید. هنگامی‌که تعداد کارها

افزایش یابد و برابر ۸۰ تا ۱۰۰ انتخاب شود الگوریتم شاخه و کران با حد پایین LB_1 قادر به یافتن جواب بهینهٔ هیچ‌کدام از مسائل در زمان تعیین‌شده نیست. درحالی‌که اگر از حد پایین LB_2 و LB_3 استفاده شود بیش از نیمی از مسائل در زمان مدنظر به جواب بهینه می‌رسند. دراین‌حالت از نظر زمان اجرا عملکرد LB_2 از LB_3 بهتر است. البته برای مسائلی که در زمان تعیین‌شده به جواب بهینه نمی‌رسند مقدار Gap از رابطهٔ (۷) محاسبه می‌شود که این مقدار درصورت استفاده از LB_3 کمتر می‌شود؛ یعنی درصورتی‌که از حد پایین LB_3 در روش شاخه و کران استفاده شود و الگوریتم در زمان تعیین‌شده به جواب بهینه نرسد جواب به‌دست‌آمده در زمان مذکور نزدیک‌تر از حالتی است که از حد پایین LB_2 استفاده شود؛ زیرا الگوریتم تولید حد پایین LB_3نسبت به دو الگوریتم دیگر حد پایین قوی‌تری تولید می‌کند.

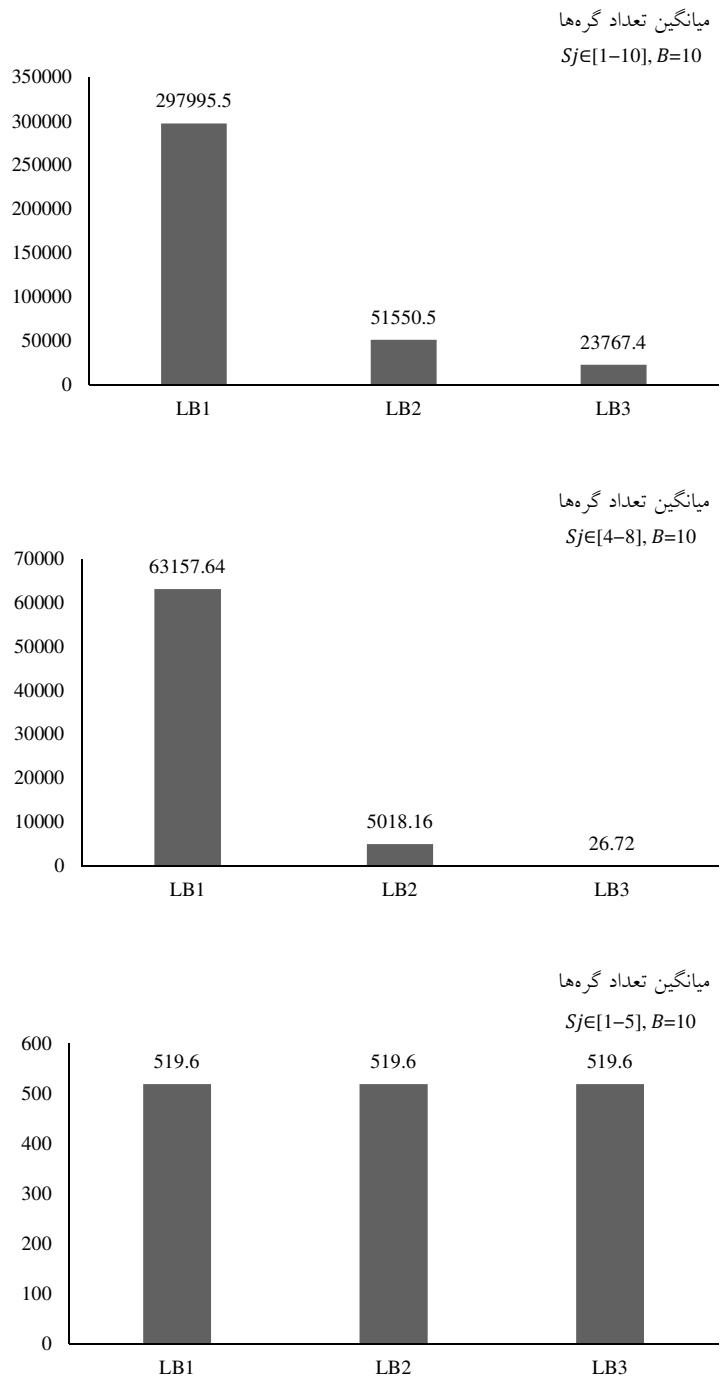
اگر اندازهٔ کارها در بازهٔ [۸-۴] انتخاب شود، درحالتی‌که تعداد کارها کمتر یا مساوی ۶۰ باشد، هر سه الگوریتم جواب بهینه را در زمان تعیین‌شده می‌یابند؛ با این تفاوت که روش شاخه و حدی که از حد پایین LB_2 استفاده می‌کند بهترین و LB_1 بدترین عملکرد را از نظر زمان اجرا دارند؛ اما هنگامی‌که تعداد کارها زیاد شود (بیشتر از ۶۰ کار) LB_3 بهترین عملکرد را دارد. زمانی‌که اندازهٔ کارها کوچک باشد (حداکثر به‌اندازهٔ نصف ظرفیت ماشین باشد) الگوریتم شاخه و کران با حد پایین LB_1 بهترین عملکرد و LB_3 بدترین عملکرد را از نظر زمان اجرا دارد. درواقع زمانی‌که ظرفیت انباشته زیاد و اندازهٔ کارها کوچک باشد LB_1 عملکرد بهتری دارد و قادر به یافتن جواب بهینه در زمان کمتری نسبت به LB_2 و LB_3 است.

در جدول (۳) **Error! Reference source not found.**ظرفیت انباشته کاهش یافته و برابر ۵ در نظر گرفته شده است. زمان پردازش کارها نیز به‌طور تصادفی در بازهٔ [۱۰-۱] در نظر گرفته شده است. وقتی اندازهٔ کارها در بازهٔ [۵-۱] است، LB_2 بهترین عملکرد را دارد و در زمان کمتری به جواب بهینه می‌رسد؛ اما هنگامی‌که اندازهٔ کارها در بازهٔ [۴-۲] قرار می‌گیرد روش شاخه و حد با هر سه حد معرفی‌شده، جواب بهینهٔ مسائل را در زمان تعیین‌شده به دست می‌آورد؛ چون اندازهٔ کارها کاهش می‌یابد. اگرچه دراین‌حالت نیز ابتدا LB_2 و سپس LB_3 بهترین عملکرد را دارد. در جداول (۴) و (۵) زمان پردازش کارها نسبت به جداول (۲) و (۳) کاهش یافته است و نتایج محاسبات نشان می‌دهد اگر زمان پردازش کارها کاهش یابد سرعت اجرای الگوریتم شاخه و حد افزایش می‌یابد.

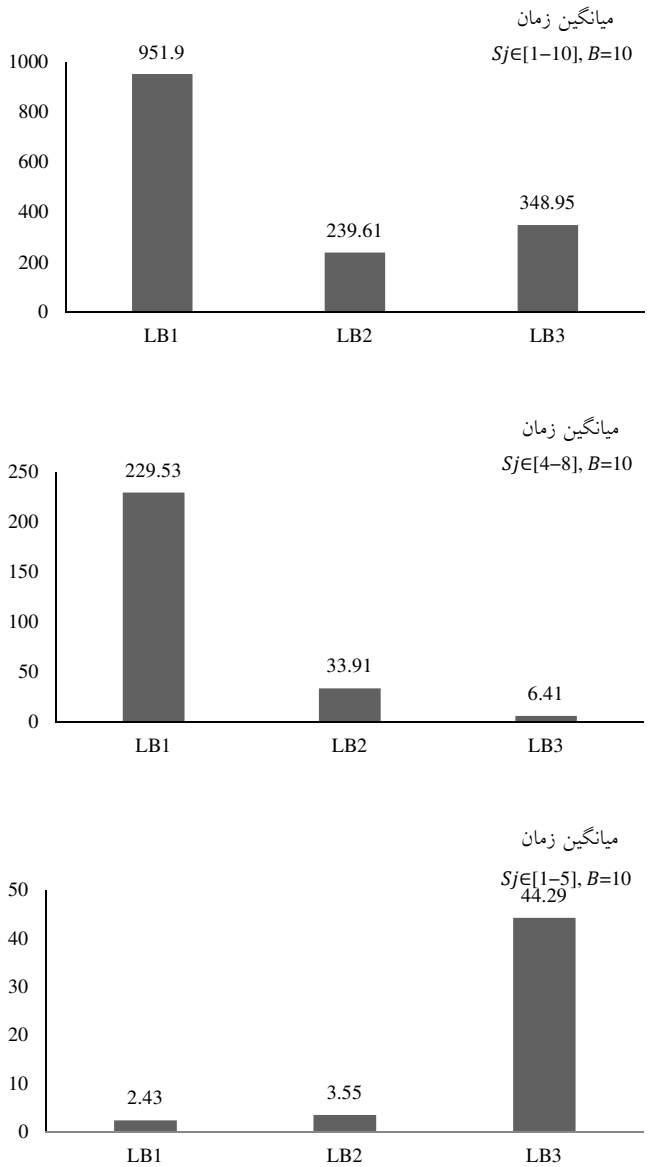
برای نمایش بهتر، نتایج روش شاخه و حد درقالب شکل ۱ تا شکل ۳ ارائه شده است. این شکل‌ها به‌ترتیب تعداد گره‌های بررسی‌شده، زمان و تعداد مسائل حل‌شدهٔ بهینه را درصورت استفاده از حدود پایین LB_1 و LB_2 و LB_3 نشان می‌دهد.

شکل (۱) میانگین تعداد گره‌های بررسی‌شده با الگوریتم شاخه و کران با حدود پایین مختلف را نشان می‌دهد. همان‌طورکه دیده می‌شود وقتی اندازهٔ کارها کوچک‌تر یا مساوی نصف ظرفیت ماشین باشد میانگین تعداد گره‌های بررسی‌شده به‌وسیلهٔ هر سه حد پایین برابر است و همهٔ مسائل در زمان مشخص‌شده حل شده‌اند؛ اما زمانی‌که اندازهٔ کارها نسبت به ظرفیت ماشین بزرگ می‌شود پیچیدگی مسئله افزایش می‌یابد و در درخت شاخه و حد میانگین تعداد گره‌های بررسی‌شده برای رسیدن به جواب بهینه زیاد می‌شود. دراین‌حالت حد پایین LB_1 عملکرد ضعیفی دارد و بهتر است از حدود پایین دیگر در بدنهٔ شاخه و حد استفاده شود. دربارهٔ استفاده از حد پایین LB_3 باید گفت که اگرچه کیفیت حدود پایینی که تولید می‌کند از دو الگوریتم دیگر بهتر است، زمان اجرای الگوریتم بیشتر است؛ چون در یکی از گام‌های الگوریتم باید مسئلهٔ حداکثر تطابق وزنی حل شود. هرچند در این پژوهش از الگوریتم ادموند (که باعث

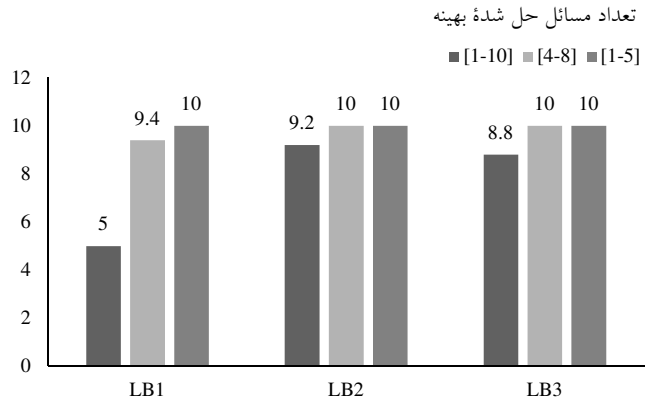
بهبود زمان اجرای الگوریتم می‌شود) برای حل مسئله استفاده شده است، به نظر می‌رسد اگر از زبان‌های برنامه‌نویسی با سرعت بیشتر مثل C استفاده شود سرعت اجرای الگوریتم افزایش می‌یابد و در مسائل با اندازهٔ کارهای بزرگ کارایی بهتری نسبت به حدود پایین دیگر دارد. شکل ۳ تعداد مسائل حل‌شدهٔ بهینه را با استفاده از روش شاخه و حد در زمان تعیین‌شده نشان می‌دهد. در این شکل نیز عملکرد بهتر حدود LB_2 و LB_3 نسبت به حد LB_1 در نمونه مسائل با کارهای با اندازهٔ بزرگ به‌وضوح مشخص است.



شکل ۱- نتایج تعداد گره‌های بررسی‌شده با استفاده از روش شاخه و حد



شکل ۲- نتایج زمان حل مسائل با استفاده از روش شاخه و حد



شکل ۳- نتایج تعداد مسائل حل‌شدهٔ بهینه با استفاده از روش شاخه و حد

نتیجه‌گیری

در این مقاله روش شاخه و حد برای حل مسئلهٔ زمان‌بندی یک ماشین پردازندهٔ انباشته با فرض وجود کارهای با اندازهٔ غیریکسان ارائه شد. در این روش برای تولید حد بالا از دو روش ابتکاری *FFLPT* و *BFLPT* استفاده و کمترین مقدار از بین آنها برای یک حد بالا در مسئله به کار گرفته شد.هم‌چنین برای تولید حد پایین از سه الگوریتم LB_1 و LB_2 و LB_3 استفاده و نتایج آنها با هم مقایسه شد. در محاسبهٔ الگوریتم LB_3 باید یک مسئلهٔ حداکثر تطابق وزنی حل شود که در پژوهش قبل از جعبه بهینه‌سازی موجود در نرم‌افزار متلب استفاده شده است و مبتنی‌بر روش شاخه و حد عمل می‌کند؛ اما در این پژوهش از الگوریتم ادموند برای حل این مسئله استفاده شد و قادر است در زمان چندجمله‌ای مسئلهٔ مدنظر را حل کند؛ بنابراین در زمان حل الگوریتم، بهبود حاصل شده است. نتایج محاسبات نشان داد که در الگوریتم شاخه و کرانِ استفاده‌شده، وقتی اندازهٔ کارها نسبت به ظرفیت ماشین بزرگ باشد حد پایین LB_2 بهترین عملکرد را دارد؛ زیرا کیفیت حدود پایینی که تولید می‌کند از LB_1 و زمان اجرای آن نیز از الگوریتم LB_3 بهتر است؛ بنابراین تعداد مسائلی که جواب بهینهٔ آنها در زمان تعیین‌شده به‌دست آمده است بیشتر از حالتی است که از دو حد پایین دیگر استفاده شود؛ اما وقتی اندازهٔ کارها نسبت به ظرفیت ماشین کوچک باشد (حداکثر به‌اندازهٔ نصف ظرفیت ماشین) الگوریتم شاخه و کران با حد پایین LB_1 بهترین عملکرد را دارد. همچنین وقتی اندازهٔ کارها متوسط باشد LB_3 بهترین عملکرد را دارد؛ بنابراین طبق نتایج به‌دست‌آمده از این پژوهش، باتوجه‌به اندازهٔ کارها از حدود پایین مختلف استفاده می‌شود و نتایج روش شاخه و حد بهبود می‌یابد.

برای پژوهش‌های آینده، پیشنهاد می‌شود سایر روش‌های حل دقیق مانند روش تولید ستون برای حل این مسئله به کار برده شود و نتایج آن با نتایج الگوریتم شاخه و کران درصورت استفاده از حدود پایین LB_2 و LB_3 مقایسه شود. همچنین ممکن است استفادهٔ هم‌زمان از دو حد پایین به‌صورت هوشمندانه نتایج بهتری به دست آورد.

References

Ahmadi, J. H., Ahmadi, R. H., Dasu, S., & Tang, C. S. (1992). "Batching and scheduling jobs on batch and discrete processors". *Operations research*, 40(4), 750-763.

Arroyo, J.E.C., Leung, J.Y.T. (2017). "Scheduling unrelated parallel batch processing machines with non-identical job sizes and unequal ready times". *Computers & Operations Research*, 78, 117-128.

Chandru, V., Lee, C. Y., & Uzsoy, R. (1993). "Minimizing total completion time on batch processing machines". *The International Journal Of Production Research*, 31(9), 2097-2121.

Chung, T. P., & Sun, H. (2018). "Scheduling batch processing machine problem with non-identical job sizes via artificial immune system". *Journal of Industrial and Production Engineering*, 35(3), 129-134.

Damodaran, P., Diyadawagamage, D. A., Ghrayeb, O., & Vélez-Gallego, M. C. (2012). "A particle swarm optimization algorithm for minimizing makespan of nonidentical parallel batch processing machines". *The International Journal of Advanced Manufacturing Technology*, 58(9-12), 1131-1140.

Dobson, G., Nambimadom, R. S. (2001). "The batch loading and scheduling problem". *Operations research*, 49(1), 52-65.

Dupont, L., & Ghazvini, F. J. (1998). "Minimizing makespan on a single batch processing machine with non-identical job sizes." *Journal européen des systèmes automatisés*, 32(4), pp. 431-440.

Dupont, L., Dhaenens-Flipo, C. (2002). "Minimizing the makespan on a batch machine with non-identical job sizes: an exact procedure". *Computers & operations research*, 29(7), 807-819.

Gabow, H. N. (1976). "An efficient implementation of Edmonds' algorithm for maximum matching on graphs". *Journal of the ACM (JACM),* 23(2), 221-234.

Ikura, Y., Gimple, M. (1986). "Efficient scheduling algorithms for a single batch processing machine". *Operations Research Letters*, 5(2), pp.61-65.

Jia, Z., Li, X., & Leung, J.Y.T. (2017). "Minimizing makespan for arbitrary size jobs with release times on P-batch machines with arbitrary capacities". *Future Generation Computer Systems*, 67, 22-34.

Jia, Z.H., Li, K., & Leung, J.Y.T. (2015). "Effective heuristic for makespan minimization in parallel batch machines with non-identical capacities". *International Journal of Production Economics*, 169, 1-10.

Hosein Zade Kashan, A., Karimi, B. (2012) "New Lower Bounds for the Optimal Makespan on a Single Batch Processing Machine" . *Amirkabir Journal of Mechanical Engineering*. 43(2), pp. 75-84.

Husseinzadeh Kashan, A., & Karimi, B. (2008). "Scheduling a single batch-processing machine with arbitrary job sizes and incompatible job families: an ant colony framework". *Journal of the Operational Research Society*, 59(9), 1269-1280.

Husseinzadeh Kashan, A., Karimi, B, & Ghomi, S. F. (2009). "A note on minimizing makespan on a single batch processing machine with nonidentical job sizes". *Theoretical Computer Science*, *410*(27-29), 2754-2758.

Husseinzadeh Kashan, A., Karimi, B., & Jolai, F. (2006). "Effective hybrid genetic algorithm for minimizing makespan on a single-batch-processing machine with non-identical job sizes." *International Journal of Production Research*, 44(12), 2337-2360.

Husseinzadeh Kashan, A., Karimi, B., & Jolai, F. (2010). "An effective hybrid multi-objective genetic algorithm for bi-criteria scheduling on a single batch processing machine with non-identical job sizes". *Engineering Applications of Artificial Intelligence,* 23(6), 911-922.

Koh S.G., Koo P.H., Kim D.C., Hur W.S. (2005). "Scheduling a Single Batch Processing Machine with Arbitrary Job Sizes and Incompatible Job Families". *International Journal of Production Economic*, 98(1): 81–96.

Lee, C. Y. (1999). "Minimizing makespan on a single batch processing machine with dynamic job arrivals". *International Journal of Production Research*, 37(1), 219-236.

Li S, (2012), "Makespan Minimization on Parallel Batch Processing Machines with Release Times and Job Sizes". *Journal of Software*, 7(6): 1203-1210.

Li X., Huang Y., Tan Q., Chen H. (2013). "Scheduling Unrelated Parallel Batch Processing Machines with Non-Identical Job Sizes". *Computers & Operations Research*, 40(12): 2983–2990.

Li, X., & Zhang, K. (2018). "Single batch processing machine scheduling with two-dimensional bin packing constraints". *International Journal of Production Economics*, *196*, 113-121.

Li, X., Huang, Y., Tan, Q., & Chen, H. (2013). "Scheduling unrelated parallel batch processing machines with non-identical job sizes." *Computers & Operations Research*, 40(12), pp.2983-2990.

Malapert A., Guéret C., Rousseau L.M. (2012). "A Constraint Programming Approach for a Batch Processing Problem with Non-Identical Job Sizes". *European Journal of Operational Research*, 221(3): 533–545.

Mathirajan, M., Sivakumar, A. I. (2006). "A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor". *The International Journal of Advanced Manufacturing Technology*, 29(9-10), 990-1001.

Melouk, S., Damodaran, P., & Chang, P.Y. (2004). "Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing." *International journal of production economics*, 87(2), 141-147.

Nong Q.Q., Ng C.T., Cheng T.C.E. (2012). "The Bounded Single-Machine Parallel-Batching Scheduling Problem with Family Jobs and Release Dates to Minimize Makespan". *Operations Research Letters,* 36 (1): 61-66.

Oulamara, A. (2007). "Makespan minimization in a no-wait flow shop problem with two batching machines." *Computers & operations research*, 34(4), pp. 1033-1050.

Parsa, N. R., Karimi, B., & Husseinzadeh Kashan, A. (2010). "A branch and price algorithm to minimize makespan on a single batch processing machine with non-identical job sizes". *Computers & Operations Research*, 37(10), 1720-1730.

Uzsoy, R., (1994), "Scheduling a single batch processing machine with non-identical job sizes". *The International Journal Of Production Research*, 32(7), 1615-1635.

Wang, S., Liu, M., Chu, F., & Chu, C. (2016). "Bi-objective optimization of a single machine batch scheduling problem with energy cost consideration". *Journal of Cleaner Production,* 137, 1205-1215.

Xu, R., Chen, H., Li, X. (2012). "Makespan minimization on single batch-processing machine via ant colony optimization". *Computers & Operations Research*, 39(3), pp. 582-593.

Yaghubian, A.R., Hodgson, T.J., Joines, J.A., Culbreth, C.T., & Huang, J.C. (1999). "Dry kiln scheduling in furniture production." *IIE transactions*, 31(8), pp.733-738.

Zhang, G., Cai, X., Lee, C. Y., & Wong, C. K. (2001). "Minimizing makespan on a single batch processing machine with nonidentical job sizes". *Naval Research Logistics (NRL),* 48(3), 226-240.

Zhou, S., Xie, J., Du, N., & Pang, Y. (2018). "A random-keys genetic algorithm for scheduling unrelated parallel batch processing machines with different capacities and arbitrary job sizes". *Applied Mathematics and Computation*, *334*, 254-268.

[1]- Batch processing machines
[2]- Uzsoy, R.
[3]- Etching process
[4]- Photolithography
[5]- Numerically controlled routers
[6]- Mathirajan, M. & Sivakumar, A. I.
[7]- Yaghubian, A.R., Hodgson, T.J., Joines, J.A., Culbreth, C.T., & Huang, J.C.
[8]- Oulamara
[9]- Mathirajan, M. & Sivakumar, A. I.
[10]- Ikura, Y. & Gimple, M
[11]- Non-identical jobes
[12]- Dobson, G. & Nambimadom, R. S.
[13]- Longest Processing Time
[14]- Dupont, L. & Ghazvini, F. J
[15]- Zhang, G., Cai, X., Lee, C. Y., & Wong, C. K.
[16]- Dupont, L. & Dhaenens-Flipo, C.
[17]- Parsa, N. R., Karimi, B. & Husseinzadeh Kashan, A.
[18]- Malapert A., Guéret C. & Rousseau L.M
[19]- Lower Bound
[20]- Wang, S., Liu, M., Chu, F., & Chu, C.

[21]- ε constraint
[22]- Melouk, S., Damodaran, P. & Chang, P.Y
[23]- Nong Q.Q., Ng C.T. & Cheng T.C.E.
[24]- Husseinzadeh Kashan, A., Karimi, B. & Jolai, F.
[25]- Husseinzadeh Kashan, A., Karimi, B., & Jolai, F.
[26]- Xu, R., Chen, H. & Li, X
[27]- Li, X., Huang, Y., Tan, Q. & Chen, H.
[28]- Chandru, V., Lee, C. Y. & Uzsoy, R
[29]- Li X., Huang Y., Tan Q. & Chen H.
[30]- Koh S.G., Koo P.H., Kim D.C. & Hur W.S.
[31]- Li
[32]- Jia, Z., Li, X., & Leung, J.Y.T.
[33]- Damodaran, P., Diyadawagamage, D. A., Ghrayeb, O., & Vélez-Gallego, M. C.
[34]- Jia, Z.H., Li, K., & Leung, J.Y.T.
[35]- Arroyo, J.E.C., Leung, J.Y.T.
[36]- First fit
[37]- Best fit
[38]- First-Fit Longest Processing Time
[39]- Best-Fit Longest Processing Time
[40]- Gabow