یک الگوریتم جستجوی پراکنده برای مسئله زمانبندی جایگشت جریان فروشگاه توزیع شده

چکیده

مسئله جایگشت توزیع شده جریان فروشگاه به تازگی به عنوان یک تعمیم از تنظیمات جریان منظم فروشگاه پیشنهاد شده است که در آن بیش از یک کارخانه برای پردازش مشاغل دردسترس هستند. تولید توزیع شده برای شرکت های بزرگ که در بازار جهانی رقابت می نمایند یک وضعیت رایج است. این مسئله دو بعد دارد: تخصیص مشاغل به کارخانه ها و زمان بندی مشاغل اختصاص یافته به هر کارخانه. با وجود اینکه این مورد به تازگی معرفی شده است، این مسئله زمانبندی جالب، توجهات را جلب کرده است و چندین روش اکتشافی و فراابتکاری در نوشته ها مطرح شده است. در این مقاله ما یک روش جستجوی پراکنده (SS) را برای این مسئله برای بهینه سازی تفاوت زمان بین شروع و اغاز یک شغل ارائه می دهیم. SS به ندرت برای تنظیمات جریان فروشگاه بررسی شده است. در این الگوریتم پیشنهادی، ما از برخی از تکنیک های پیشرفته مانند یک مجموعه مرجع ساخته شده از راه حل های کامل و جزئی همراه با ویژگی های دیگر مانند ر جستجوی محلی و شروع دوباره استفاده نمودیم. یک کمپین محاسباتی جامع شامل 10 الگوریتم موجود، همراه با تجزیه و تحلیل های آماری، نشان می دهد که الگوریتم جستجوی پراکنده پیشنهادی، با اختلاف قابل توجهی نتایج بهتری را نسبت به الگوریتم های موجود تولید می کند. علاوه بر این تمام 720 بهترین شده راه حل شناخته برای این مسئله، بهبود می یابند.

کلمات کلیدی: زمانبندی توزیع شده، جایگشت جریان فروشگاه، جستجوی پراکنده

1. مقدمه

زمانبندی، با تخصیص منابع، به طور معمول ماشین آلات، به وظایف (معمولا به عنوان مشاغل نامیده می شوند) در طول زمان با هدف بهینه سازی یک هدف معین سرو کار دارد (Pinedo، 2012). زمانبندی یک مسئله مهم است

که به طور عمده در صنایع تولیدی به نظر می رسد. به خوبی مشخص شده است که زمانبندی های خوب تا حد زیادی به عملکرد کلی یک شرکت (McKay، Pinedo، و Webster، 2002) کمک می کنند. طرح ماشین آلات در طبقه تولید، همراه با جریان مشاغل در ماشین آلات، همراه با هزاران محدودیت و تنظیمات زندگی واقعی نوع حل مسئله زمانبندی را تعیین می کند. مسئله زمانبندی جریان فروشگاه (FSP) در عمل مسلما رایج ترین طرح پردازش است، همانطور که برای کارخانه های تولیدی، تولید یک خانواده معین از محصولات که باید از ماشین آلات در نظمی مشخص بازدید نمایند، معمول است. به عنوان مثال، در تولید خودرو، نقاشی بدنه خودرو باید پس از جوشکاری بدنه و قبل از هر گونه عملیات مونتاژ و از این رو یک ساختار جریان فروشگاه انجام شود. Reisman، Kumar، و Motwani (1997) موارد عملی را بررسی نمودند و نتیجه گرفتند که مسئله جریان فروشگاه دارای بسیاری از کاربردها در زندگی واقعی است. این قابلیت کاربرد جریان فروشگاه نیز در بسیاری از بازنگری های خروجی نوشته ها مانند Framinan، Gupta، و Leisten (2004)، Ruiz و Maroto (2005)، Hejazi و Saghafian (2005) و Gupta و Stafford (2006) برجسته شده است. در واقع، زمانی که این مسئله به جریان های فروشگاه ترکیبی یا مسائل جریان خط انعطاف پذیر تعمیم داده شود، بسیاری از مسائل تولید را می توان پس از یک جریان فروشگاه مدلسازی نمود. (Linn و Zhang، 1999؛ Vignier، Billaut، و Proust، 1999؛ Wang، 2005؛ Quadt و Kuhn، 2007، Ruiz و Vázquez-Rodríguez، 2010؛ Ribas، Leisten، و Framinan، 2010). FSP را می توان به طور رسمی به شرح زیر توصیف نمود: مجموعه N از n شغل مختلف و مستقل باید زمانبندی شوند. مشاغل معمولاً، سفارشات مشتری و یا دسته های محصولاتی که باید پس از انجام یک فرایند زمانبندی تولید مدلسازی می نمایند (Pochet و Wolsey، 2006). هر شغل، j $j \in N$ باید به منظور تمام m ماشین آلات در مجموعه ای از M ماشین آلات بازدید شود. بدون از دست دادن کلیت، هر شغل، اولین دستگاه 1را بازدید می کند ، پس از آن دستگاه 2 و به همین ترتیب تا m دستگاه. تا زمانی که در دستگاه فعلی به پایان برسد، یک شغل نمی تواند به دستگاه بعدی برود و یک دستگاه نمی تواند بیش از یک شغل را در همان زمان پردازش کند. به عنوان یک نتیجه از رفتن کار از یک دستگاه به بعدی هر شغل به m وظیفه برای هر ماشین تقسیم بندی می شود. هر وظیفه از

یک شغل j، $j \in N$ نیاز به یک زمان پردازش معین در هر دستگاه i. دارد. این زمان پردازش به عنوان p_{ij} نشان داده می شود و قطعی، شناخته شده از قبل و معمولا نامنفی است که توسط یک مقدار عدد صحیح نشان داده می شود.

هدف در FSP، پیدا کردن یک توالی برنامه یا پردازش تمام مشاغل در ماشین آلات است به طوری که یک معیار بهینه سازی معین بهینه سازی شود. با توجه به بررسی مقالات قبلا اشاره شده، هدف اغلب مورد مطالعه قرار گرفته، حداقل نمودن حداکثر زمان اتمام و یا تفاوت زمان بین شروع و اغاز یک شغل است که به صورت $C_{\max}$ نشان داده می شود. با توجه به زمان اتمام شغل در آخرین m دستگاه ، که به صورت C_j نشان داده می شود، تفاوت زمان بین شروع و اغاز یک شغل، به حداقل رساندن حداکثر C_j ، $j = 1, \ldots, n$ است. از آنجا که بسیاری از زمانبندی های ممکن به عنوان دنباله های شغلی در هر دستگاه وجود دارد، تعداد کل راه حل ها $(n!)^m$است، یعنی، تمام جایگشت های شغلی ممکن است در هر دستگاه، با توجه به اینکه این جایگشت ها می تواند از دستگاهی به دستگاه دیگر تغییر نماید. با توجه به این فضای جستجوی بزرگ، FSP معمولا به آنچه که مسئله زمانبندی جایگشت جریان فروشگاه و یا PFSP با ممنوع نمودن عبور شغل نامیده می شود، ساده سازی می شود، یعنی، زمانی که توالی تولید برای یک دستگاه، تثبیت شود، تمام ماشین آلات از یک دنباله تولید پیروی می نمایند. این تعداد کل راه حل ها را به n! پایین می آورد. با استفاده از نماد میدان سه شناخته شده برای مسائل زمانبندی (Graham، Lawler، Lenstra، و Rinnooy Kan، 1979؛ Pinedo، 2012)، PFSP با معیار تفاوت زمان بین شروع و اغاز یک شغل به عنوان $F/prmu/C_{\max}$ نشان داده می شود.

این مسئله در ابتدا تقریبا 60 سال پیش توسط Johnson (1954) مورد مطالعه قرار گرفت که در آن الگوریتم به خوبی شناخته شده Johnson برای حل نسخه دو دستگاهی پیشنهاد شد. برای سه یا بیشتر ماشین آلات، این مسئله به عنوان $\mathcal{NP}$ کامل در معنای قوی Garey، Johnson، و Sethi (1976) شناخته شده است. امروزه، نوشته ها در مورد PFSP بسیار زیاد است و مسئله و بسیاری از انواع به طور کامل مورد بررسی قرار گرفته اند. بنابراین این موضوع به طور گسترده ای مورد مطالعه قرار گرفته است که حتی برخی از جزوه و مقالات اختصاصی

مانند Chakraborty (2009) و Emmons و Vairaktarakis (2012)، و یا حتی برای برخی از مدل ها، مانند جریان بهر در Sarin و Jaiprakash (2007) وجود دارد. با این حال، یک گسترش وجود دارد که تنها به تازگی ارائه شده است. در Naderi و Ruiz (2010) و Naderi و Ruiz (2010) یک نوع را مورد مطالعه قرار دادند که به عنوان مسئله زمانبندی توزیع شده جابجایی جریان فروشگاه و یا DPFSP اشاره شد. در اصل، PFSP منظم، یک کارخانه تنها را در نظر می گیرد که در آن محصولات تولید می شوند. با این حال، شرکت های چند کارخانه ای در اقتصاد جهانی بسیار رقابتی هستند. نوشته ها در مورد سیستم های تولید با مثال هایی که در آن نشان داده شده است که تولید توزیع شده، کلیدی برای کیفیت محصول بالا، هزینه های تولید کم و کاهش خطرات مدیریت، در میان بسیاری از مزایای دیگر است، فراوان هستند (Wang، 1997؛ Moon، Kim، و Hur، 2002؛ Kahn، 2004، در میان بسیاری دیگر). تولید توزیع شده در حال حاضر یک موضوع مورد علاقه است، همانطور که سرمقاله اخیر در یک موضوع خاص از یک مجله تولید معتبر نشان می دهد (Chan و Chung، 2013). در آن سرمقاله و در بسیاری از مقالات موضوع خاص بیان شده، اهمیت و مزایای تولید پراکنده ستایش و برجسته شده اند. در DPFSP، یک پیچیدگی مهم اضافه شده با توجه به PFSP وجود دارد: مشاغل باید به کارخانه ها اختصاص داده شوند و پس از آن یک زمانبندی باید برای هر کارخانه ساخته شود. به طور رسمی تر، DPFSP، جایگشت جریان فروشگاه را به طور منظم به صورت زیر گسترش می دهد: مجموعه N از n شغل باید توسط یک مجموعه G از F کارخانه یکسان پردازش شوند. هر کارخانه دارای همان مجموعه M از m ماشین آلات است. دفعات پردازش تمام وظایف یک شغل معین از یک کارخانه به کارخانه دیگر تغییر نمی کند. هنگامی که یک شغل به یک کارخانه اختصاص داده شود، یک شغل باید در آن کارخانه تکمیل شود. هدف، به حداقل رساندن حداکثر تفاوت زمان بین شروع و اغاز یک شغل در میان تمام کارخانه ها است. Naderi و Ruiz (2010) به این مسئله به عنوان $DF/prmu/C_{\max}$ اشاره نموده اند. همان محققین نشان دادند که هیچ کارخانه بدون هیچ مشاغل اختصاص داده شده نباید باقی بماند (n>F معین) همانطور که این تفاوت زمان بین شروع و اغاز یک شغل را بهبود نمی دهد. آنها همچنین به این نتیجه رسیدند که

تعداد کل راه حل ها در DPFSP، $\binom{n-1}{F-1}n!$ است. علاوه بر این، از آنجا که DPFSP به PFSP منظم کاهش می یابد، اگر F = 1، این نتیجه گیری آسان است که DPFSP نیز یک مسئله $\mathcal{NP}$-سخت است.

از مقاله Naderi و Ruiz (2010)، چند نویسنده دیگر آن نتایج را گرفتند و چندین روش برای حل این مسئله جدید پیشنهاد شده اند. Naderi و Ruiz (2010) برخی از مدل های ریاضی، فن آوری هوشمند ساده و روش جستجوی محلی را پیشنهاد نمودند. بنابراین، روش های پیچیده تر ممکن است راه حل های جالب جدید را برای این مسئله سخت ترکیبی نشان دهند. علاوه بر این، با توجه به روش های اخیر موجود پیشنهادی، مقایسه اثر بخشی و کارایی روش های موجود برای تعیین آخرین وضعیت این روش ها نیز ارزشمند است. اینها برخی از اهداف این مقاله هستند. هنگام تصمیم گیری در مورد اینکه کدام تکنیک های پیشرفته می توانند برای DPFSP اعمال شوند، مشاهده کردیم که روش مبتنی بر جستجوی محلی ساده موفق به فرار از نقاط بهینه قوی محلی نشد و بنابراین ما یک روش قدرتمند را انتخاب نمودیم: جستجوی پراکندگی (Glover، Laguna، و Marti، 2000؛ Laguna و Marti، 2003؛ Marti، Laguna، و Glover، 2006، در میان دیگران). بر خلاف بسیاری از چارچوب های فراابتکاری موجود، که چندین بار برای مسائل جریان فروشگاه استفاده شده است، جستجوی پراکنده (SS) به ندرت برای این تنظیمات زمانبندی استفاده می شود. منابع با کاربردهای جستجوی پراکنده برای جریان های فروشگاه منظم کمیاب هستند. Nowicki و Smutnicki (2006) برخی از روش ها، از جمله ایده هایی از پیوند دوباره مسیر و جستجوی پراکنده را برای PFSP منظم با معیار تفاوت زمان بین شروع و اغاز یک شغل ارائه دادند، اما موفق نشدند تا به طور قابل توجهی پیشرفتی در این کار صورت دهند. در یک مقاله کوتاه، Saravanan، Noorul Haq، Vivekraj، و Prasad (2008) یکی دیگر از روش های جستجوی پراکنده را برای این مسئله مشابه پیشنهاد دادند و متوسط درصد انحرافات نسبت به راه حل های شناخته شده برای معیار Taillard (1993) را کمی بیش از 1٪ گزارش نمودند. این روش به وضوح بهتر از انحرافات زیر 0.5٪ با توجه به روش ساده تر Iterated Greedy (IG) از Ruiz و Stützle (2007) و یا انحرافات 0.22٪ معین در Vallada و Ruiz (2009) نیست. در مورد PFSP، به نظر می رسد که هیچ کاربرد جستجوی پراکندگی قابل توجهی وجود ندارد. بنابراین، این تفکر را قابل قبول است که

روش های جستجوی پراکنده برای مسائل جریان فروشگاه هنوز هم برخی از سقفها را برای بهبود قائل شده است و بنابراین ما آنها را برای این مقاله انتخاب نموده ایم. علاوه بر این، تنوع کنترل شده در جستجوی پراکنده نشان دهنده قدرت بزرگ در DPFSP است، همانطور که ما به لحاظ تجربی نشان می دهیم.

ادامه این مقاله به شرح زیر سازماندهی شده است: بخش 2 بررسی جامع در مورد نوشته ها در مورد DPFSP را فراهم می کند. بخش 3 روش جستجوی پراکنده پیشنهادی در جزئیات را ارائه می دهد. این روش در بخش 4 تنظیم می شود. در همین بخش، تقریبا تمام الگوریتم های مربوطه از نوشته ها در مورد DPFSP دوباره پیاده سازی می شود و با دقت ارزیابی می شود. از طریق تجزیه و تحلیل جامع محاسباتی و آماری، ما نشان می دهیم که الگوریتم جستجوی پراکنده ارائه شده می تواند به عنوان یک روش مدرن برای DPFSP و به حداقل رساندن تفاوت زمان بین شروع و اغاز یک شغل در نظر گرفته شود.در نهایت، بخش 5 این مقاله نتیجه گیری و برخی از راه ها برای تحقیقات آینده را پیشنهاد می کند.

2. بررسی نوشته ها

در Naderi و Ruiz (2010)، نویسندگان شش مدل برنامه نویسی مختلف ترکیبی عدد صحیح خطی را برای DPFSP همراه با 12 روش اکتشافی ارائه نمودند که از اعمال دو شغل مختلف برای قوانین انتساب کارخانه به شش روش اکتشافی معروف برای مسئله منظم جریان فروشگاه حاصل می شد. این دو قانون به شرح زیر است:

- اختصاص دادن یک شغل j معین به کارخانه با کمترین C_{max} فعلی، که شامل شغل j نمی شود.
- اختصاص دادن شغل j به کارخانه که در نخستین بار آن را کامل می کند، به عنوان مثال، این کارخانه که منجر به کمترین C_{max} پس از اختصاص دادن شغل j می شود.

این قواعد هر زمانی که یک شغل زمانبندی می شوند اعمال می شوند. از شش روش اکتشافی آزمایش شده، روش NEH Nawaz، Enscore، و Ham (1983) با شغل دوم برای قاعده انتساب کارخانه (که NEH2 نامیده می شود) منجر به بهترین عملکرد اکتشافی شد. به غیر از روش اکتشافی، Naderi و Ruiz (2010) یک فرود

همسایگی متغیر ساده (VND، Mladenovic' و Hansen (1997)) را با شروع از راه حل NEH2 و با دو همسایگی معرفی نمودند. یکی جستجوی محلی ورودی برای تمام کارخانه (تا زمانی که نقطه بهینه محلی در هر کارخانه) و جستجوی ثانویه محلی، کارخانه تولیدکننده مقدار تفاوت زمان بین شروع و اغاز یک شغل را در نظر بگیرید و همه مشاغل آن را استخراج کند و آنها را در تمام کارخانه های دیگر آزمایش نماید. دو معیار پذیرش مختلف استفاده می شوند: (الف) پذیرش راه حل جدید اگر تفاوت بحرانی زمان بین شروع و اغاز یک شغل (بزرگترین تفاوت زمان بین شروع و اغاز یک شغل در میان تمام کارخانه ها) کاهش یابد و (ب) پذیرش راه حل اگر سود خالصی در مقادیر تفاوت زمان بین شروع و اغاز یک شغل بین کارخانه های مرتبط در جستجوی محلی وجود داشته باشد. جزئیات بیشتر در Naderi و Ruiz (2010) ارائه شده است. روش های VND حاصل با هر دو معیار پذیرش به عنوان VND(a) و VND(b) به ترتیب ارجاع شدند. نتایج تجربی نشان داد که VND(a) یک درصد انحراف متوسط را نسبت به بهترین راه حل های شناخته شده برای مسائل بزرگ تا 500 مشاغل، 20 ماشین آلات و 7 کارخانه تنها به اندازه 0.10٪ تولید می کند. توجه داشته باشید که تمام روش های ارائه شده توسط Naderi و Ruiz (2010) سریع می باشند،، همانطور که کندترین روش - VND(a) – کمتر از به طور متوسط 0.15 ثانیه روی یک کامپیوتر Intel Core 2 Duo در 2. 4 گیگاهرتز با 2 گیگابایت حافظه RAM نیاز داشت.

همانطور که Naderi و Ruiz (2010) اشاره کردند، قبل از 2010، تقریبا هیچ نوشته ای در مورد زمانبندی جریان فروشگاه توزیع شده غیر از برخی از مقالات آزادانه مرتبط وجود نداشت. با این حال، پس از انتشار آن مقاله، چندین نویسنده مطالعات را پیگیری نمودند. اولین کار، کار Liu و Gao (2010) بود. این نویسندگان یک کار فراابتکاری الکترومغناطیسی پیچیده (که به عنوان EM در این مقاله اشاره شده است) را ارائه نمودند. آنها جستجوی محلی VND از کار Naderi و Ruiz (2010) را بهبود دادند و آن را به یک جستجوی همسایگی متغیر قوی تر (VNS، Mladenovic' و Hansen (1997)) با همسایگی های مختلف از جمله درج در کارخانه حیاتی (که تولیدکننده تفاوت زمان بین شروع و اغاز یک شغل است)، مبادله در کارخانه حیاتی و درج عمومی و مبادله گسترش دادند. در ارزیابی محاسباتی خود، Liu و Gao (2010) به طور مستقیم در برابر VND(a) مقایسه ننمودند، بلکه به بهبود

151 راه حل بهتر شناخته شده از 720 نمونه بزرگ ارائه شده در Naderi و Ruiz (2010) اشاره نمودند. همانطور که ما بعداً برجسته خواهیم نمود، این مقایسه ها می تواند گمراه کننده باشد. علاوه بر این، زمان های CPU از روش EM به میزان قابل توجهی بزرگتر از VND(a) است. بنابراین، باید دید که آیا EM با VND(a) رقابتی است یا خیر. پس از آن، Gao و Chen (2011a)، یک الگوریتم ترکیبی ژنتیک را با جستجوی محلی (GA_LS) معرفی نمودند که ما به سادگی به عنوان HGA می نامیم. این روش ژنتیکی از GA برای جایگشت جریان فروشگاه به طور منظم از Ruiz، Maroto ، و Alcaraz (2006) الهام گرفته است. این الگوریتم NEH2 و VND(a) را به عنوان مقدار دهی اولیه استفاده می کند. فاز جستجوی محلی شبیه به VND(a) است اما یک همسایگی سوم گنجانده شده است که در آن تبادل مشاغل از کارخانه حیاتی و همه مشاغل دیگر در تمام کارخانه های دیگر مورد آزمایش قرار می گیرند. در آزمایشات خود، HGA، راه حل های بهتری را از VND(a) گزارش نمود اما دوباره این کار با CPU بسیار بزرگتر انجام شد. با توجه به نتایج حاصل از Gao و Chen (2011a)، روش HGA آنها از تقریبا 246 برابر بیشتر زمان CPU نسبت به VND(a) استفاده می کند. در این مقاله، همان نویسندگان، HGA پیشنهادی خود را با همان زمان CPU به اندازه VND(a) آزمایش نمودند و نتایج به دست آمده کاملا برعکس با HGA بود که نشاندهنده عملکرد ظاهرا بدتر از VND(a) است. بنابراین، آزمایش جالب دیگر، آزمون HGA در مقابل VND(a) در یک سناریوی کاملا قابل مقایسه است.

Gao و Chen (2011b)، یک بهبود از کار اکتشافی NEH، Nawaz و همکاران (1983) و NEH2 از Naderi و Ruiz (2010) را معرفی نمودند. این بهبود، متشکل از قرار دادن F شغل در یک زمان (یکی برای هر کارخانه) به جای یک شغل در یک زمان است همانطور که در NEH اکتشافی معمول است. این چند ورودی از طریق یک شعبه نامشخص و روش محدود انجام می شود و نویسندگان نیز شغل دوم قبلا اظهار نظر شده را برای قاعده انتساب کارخانه و همچنین دیگر بهبودهای منتشر شده از NEH استفاده نمودند. بهترین روش ترکیبی پیشنهادی به عنوان NEHdf نامیده می شود. در آزمایش های محاسباتی، نشان داده شده است که NEHdf کمی بهتر از NEH2 عمل

می کند (با این حال، در یک آزمایش آماری ارائه شده، نشان داده شده است که NEHdf از نظر آماری بهتر از NEH2 عمل می کند). باز هم، این عملکرد بهتر به CPU اضافی منجر می شود.

اخیرا، Gao، Chen، و Liu (2012b)، یک الگوریتم ژنتیک را ارائه نمودند که کمی بهتر از الگوریتم HGA از Gao و Chen (2011a) عمل می نماید. در مقایسه های خود، متوسط درصد انحراف نسبی الگوریتم جدید، که به عنوان GA_KB، نامیده می شود، به اندازه 3.0٪ کاهش می یابد که یک بهبود نسبتاً حاشیه ای است. زمان های CPU نیز اندکی کاهش می یابد، اما بیش از 200 برابر بزرگتر از VND(a) باقی می ماند.

در همان سال، نویسندگان مرتبط (Gao، Chen، Deng، و Liu، 2012a) یک روش VNS تجدید نظر شده را ارائه کرده اند. در واقع، این نویسندگان، VND(a) از Naderi و Ruiz (2010) با روش NEHdf بهبودیافته ارائه شده خود در Gao و Chen (2011b) ترکیب نمودند. الگوریتم به دست آمده به عنوان VNS(B & B) نامیده می شود. تجزیه و تحلیل های محاسباتی نشان می دهد که VNS(B & B) نسبت به VND(a) عالی تر است اما بدیهی است که این عالی بودن با زمان CPU اضافی همراه است.

اخیرا، میزان مقالات در حوزه DPFSP در حال افزایش است. Gao، Chen، و Deng (2013) یک روش جستجوی ممنوعه را ارائه کرده اند. الگوریتم پیشنهادی بر اساس طرح های جستجوی محلی ارائه شده در Gao و Chen (2011a) ایجاد شده است و شامل برخی از فرآیندهای جستجوی محلی گسترده تر می شود. در بخش تجربی، نشان داده شده است که این روش جدید TS با اختلاف خوب، بهتر از HGA از Gao و Chen (2011a) کار می کند که کارایی محاسباتی را نیز بهبود می بخشد. با این حال، از جداول داده شده در Gao و همکاران، TS پیشنهادی تقریباً 117 برابر کندتر از VND(a) است.

همچنین به تازگی، Lin، Ying، و Huang (2013) روش حریص تکراری را با الهام از کار Ruiz و Stützle (2007) پیشنهاد نمودند. چهار نوع IG ارائه می شوند و یکی از بهترین ها، که توسط IG_{VST} اشاره شده است، در برابر HGA از Gao و Chen (2011a) و TS از Gao و همکاران (2013) مقایسه شده است. این نتایج با اختلاف

گسترده به نفع روش IG_{VST} است و همچنین زمان های CPU تا حد زیادی کاهش می یابد، البته شرایط به طور کامل قابل مقایسه نیستند و زمان های CPU گزارش شده هنوز هم بسیار بزرگتر از VND(a) می باشد.

پس از اینکه همه آزمایش ها و تجزیه و تحلیل های این مقاله به پایان رسید، ما یک مقاله که اخیرا منتشر شده است آگاه شدیم (Wang، Wang، Liu، و Xu، 2013). این نویسندگان یک برآورد از الگوریتم توزیع (EDA) را ارائه نمودند. در حالی که روش ارائه شده بهتر از VND(a) عمل می نماید، راه حل های جدید به دست آمده به خوبی همانند موارد گزارش شده در دیگر مقالات اخیر نیستند. علاوه بر این، الگوریتم ارائه شده بسیار آهسته تر از VND(a)، بدون نیاز به کمتر از تقریبا 788 برابر زمان بیشتر CPU از VND(a) می باشد.

همانطور که ما می توانیم ببینیم، تعداد بی شماری از روش های فراابتکاری وجود دارد که اخیرا برای DPFSP معرفی شده اند. همانطور که بررسی مهم ما نشان می دهد، بسیاری از این روش ها در برابر یکدیگر مقایسه نشده اند. اکثر مقایسه ها در برابر VND(a) انجام شده است که اساسا یک روش اکتشافی بهبود یافته توسط برخی از مکانیسم های جستجوی محلی است. روش های پیشرفته جدیدتر و بیشتر ممکن است راه حل ها را بسیار بیشتر بهبود دهند.

3. روش جستجوی پراکنده

جستجوی پراکنده یک نوع از الگوریتم تکاملی است که به شدت بر اساس یک رویکرد اصولی برای تولید و ترکیب دوباره راه حل ها و هدایت به دور از اتفاقی بودن دیگر روش های تکاملی مانند الگوریتم های ژنتیک است. مشخصه اصلی SS، تنوع راه حل ها به عنوان وسیله ای برای بهینه سازی کیفیت بالا است. قدمت و ریشه آن به دهه 1970 با آثار Glover (1977) و یا Glover (1998)، که بعداً در Glover و همکاران. (2000)؛ Laguna و Marti (2003) و یا Marti و همکاران. (2006) برای نام نهادن چند روش رسمی شد، باز می گردد.

SS به کار گرفته شده در این مقاله به شرح زیر از یک قالب عمومی معین در Laguna و Marti (2003) و در Marti و همکاران. (2006) پیروی می کند بر اساس "پنج روش" شناخته شده است: (1) روش تولید تنوع. جمعیت اولیه آماری این روش با استفاده از یک راه حل ورودی ایجاد می شود. در اینجا یک حوزه P از راه حل های

متنوع PSize ایجاد شده است. (2) روش بهبود. یک مکانیسم، معمولا یک شکل از جستجوی محلی، به منظور بهبود راه حل از هر یک از مجموعه های کاری است. به طور معمول، این مورد برای مجموعه P در آغاز روش SS اعمال می شود. (3) روش به روز رسانی مجموعه مرجع. در SS، مجموعه مرجع و یا $RefSet$ در ابتدا معمولا شامل بهترین راه حل از P می شود. این مجموعه مورد نظر تا حد ممکن متنوع است، بنابراین نه تنها بهترین انتخاب می شود، بلکه راه حل های حداکثر متنوع ترجیح داده می شود. $RefSet$، ابتدائاً یک لیست مرتبه بندی شده با بهترین راه حل است. زمانی که روش SS تکرار می شود، راه حل های جدید با توجه به کیفیت و تنوع آنها وارد $RefSet$ می شوند (4) روش تولید زیر مجموعه. در اینجا، برخی از راه حل های $RefSet$ برای پردازش بعدی انتخاب می شوند. ساده ترین روش، تولید تمام جفت های ممکن است از راه حل ها از $RefSet$ به عنوان زیر مجموعه ها است. (5) روش ترکیبی راه حل. راه حل های انتخاب شده در روش تولید زیر مجموعه، برای ایجاد راه حل های جدید دوباره ترکیب می شوند. به طور معمول، راه حل های جدید با استفاده از روش بهبود ارتقا داده می شود و بعداً برای درج در $RefSet$ در روش به روز رسانی مجموعه مرجع در نظر گرفته می شود. کل فرآیند در حالی تکرار می شود که تغییرات در $RefSet$ وجود دارد، یعنی در حالی که راه حل های جدید مختلف در حال کشف هستند. اجازه بدهید ما نمونه ای از تمام این روش ها را در الگوریتم SS پیشنهادی خود بیاوریم.

3.1. ارائه راه حل و روش تولید تنوع

در نوشته های PFSP، رایج ترین ارائه راه حل جایگشت n شغل است. از آنجا که در DPFSP، این جایگشت در میان F کارخانه تقسیم می شود، ساده ترین آن، داشتن F فهرست، در هر کارخانه است. هر لیست شامل یک جایگشت جزئی با نظمی می شود در آن مشاغل باید در هر کارخانه پردازش شوند. این ارائه راه حل است که Naderi و Ruiz (2010) و نویسندگان بعدی به کار گرفته اند. برای مثال، اگر ما یک مسئله با 10 شغل (n = 10) و سه کارخانه (F = 3) داشته باشیم، یک راه حل ممکن اینست:

$$\begin{cases} 4,8,1 \\ 2,10,5 \\ 7,6,3,9 \end{cases}$$

در این راه حل، مشاغل 4، 8 و 1 به کارخانه 1 منسوب می شوند و مشاغل 2، 10 و 5 به کارخانه 2 و به همین ترتیب. دنباله در هر کارخانه توسط اسکن هر یک از لیست های شغلی از چپ به راست به دست می آید.

در روش SS پیشنهادی ما برای DPFSP، ما دو مجموعه ویژه ساخته شده در داخل مجموعه مرجع را داریم. اولی، H است که حاوی تعداد b از بهترین راه حل های یافته شده می باشد. مجموعه دوم، که S نا می گیرد، از l بردار انتساب کارخانه ساخته شده است. متحد شدن این دو مجموعه، مجموعه مرجع را می سازد، به عنوان مثال، $RefSet = H \cup S$ با اندازه $b + l$. این مجموعه به وضوح متفاوت است. مجموعه H شامل راه حل های کامل با توجه به ارائه راه حل فوق می شود. با این حال، مجموعه S تنها شامل انتسابات کارخانه برای مشاغل می شود، یعنی، با توجه به 10 مشاغل، 3 کارخانه DPFSP به عنوان مثال، یکی از اعضای مجموعه S می تواند موارد زیر باشد: $\{2,3,1,1,2,2,2,3,1,3\}$ که به این معنی است که شغل 1 به کارخانه 2، شغل 2 به کارخانه 3 و به همین ترتیب تا زمانی که شغل 10 به کارخانه 3 منسوب می شود. اینها راه حل های کامل نیست بلکه فقط انتسابات کارخانه هستند، زمانی که هیچ سفارش کاری در هر کارخانه داده نشده است. منطق پشت این دو مجموعه مجزا در داخل مجموعه مرجع، پس از روش ترکیبی راه حل روشن خواهد شد.

برای ساخت اولیه مجموعه H، ما با Psize 25 جایگشت کار تصادفی شروع می کنیم. 24 تا از این جایگشت ها به عنوان سفارش اولیه استفاده می شوند که برای روش NEH2 از Naderi و Ruiz (2010) اتخاذ می شوند. به یاد بیاورید که این، گسترش روش Nawaz NEH و همکاران (1983) است. برای 25 جایگشت آخر، ما از سفارش اولیه NEH منظم به جای تصادفی استفاده می نماییم. در واقع، در NEH2، مشاغل یک به یک و با توجه به سفارش اولیه به تمام موقعیت ها از همه کارخانه ها درج می شوند. این شغل در نهایت در موقعیت قرار می گیرد که نتیجه آن حداقل تفاوت زمان بین شروع و اغاز یک شغل جزئی است. شغ دوم برای قاعده انتساب کارخانه (بخش 2 را ببینید) استفاده می شود. اجازه دهید ما یک مثال از مورد قبلی با 10 شغل و 3 کارخانه بزنیم. اجازه بدهید ما

سفارش اولیه شغلی را به عنوان F4 در نظر $\{4,2,7,6,1,3,10,5,9,8\}$ در نظر بگیریم. با شروع راه حل جزئی به شرح زیر:

$$\begin{cases}4,1\\2\\7,6\end{cases}$$

شغل بعدی برای وارد کردن، شغل 3 است، همانند 4، 2، 7، 6 و 1 (مشاغل قبلی در سفارش اولیه) در حال حاضر در راه حل هستند. بنابراین، کار 3 به 8 موقعیت مختلف در راه حل قبلی اضافه می شود، و در نتیجه، جایگزین ها به صورت زیر هستند:

$$\begin{cases}3,4,1\\2\\7,6\end{cases},\ \begin{cases}4,3,1\\2\\7,6\end{cases},\ \begin{cases}4,1,3\\2\\7,6\end{cases},\ \begin{cases}4,1\\3,2,\\7,6\end{cases}$$

$$\begin{cases}4,1\\2,3,\\7,6\end{cases}\ \begin{cases}4,1\\2\\3,7,6\end{cases},\ \begin{cases}4,1\\2\\7,3,6\end{cases},\ \begin{cases}4,1\\2\\7,6,3\end{cases}$$

جایگزین منجر به بهترین تفاوت زمان بین شروع و اغاز یک شغل جزئی انتخاب می شود. به منظور سرعت بخشیدن به روش درج، شتاب های به خوبی شناخته شده از Taillard (1990) استفاده می شوند. این روش برای تمام جایگشت های شغلی با 25 راه حل بهبود یافته NEH2 اعمال می شوند. سپس، b راه حل بهتر در این 25 در مجموعه H گنجانده می شوند. توجه کنید که این برای ساخت مجموعه H اولیه اعمال می شود. پس از آن، در هر تکرار از روش SS، مجموعه H شامل بهترین b راه حل بازدید شده می شود.

همانند مجموعه S، مورد استفاده برای تنوع، ما به سادگی آن را با شغل تصادفی به انتسابات کارخانه مقداردهی اولیه می نماییم. همانطور که خواهیم دید، در هر تکرار از الگوریتم SS، مجموعه H و S ترکیب می شوند. بنابراین، و به منظور حفظ تنوع، مجموعه S به طور تصادفی در هر تکرار از روش SS دوباره تولید می شود.

3. 2. تولید زیر مجموعه و روش های ترکیبی راه حل

در روش پیشنهادی SS، روش تولید زیر مجموعه با توجه به ماهیت دو مجموعه H و S در داخل $RefSet$ نیز متفاوت از اکثر کاربردهای جستجوی پراکنده است. این روش شامل انتخاب تمام ترکیبات ممکن از راه حل ها در مجموعه H با انتسابات کارخانه در مجموعه S می شود. بنابراین، در هر تکرار، جفت b.l در نظر گرفته می شوند. به عنوان مثال، فرض کنید ما b = 3 و L = 2، به عنوان مثال، $H = \{h_1, h_2, h_3\}$ و $S = \{s_1, s_2\}$. بنابراین ما باید شش ترکیبات: $(h_1, s_1), (h_1, s_2), (h_2, s_1), (h_2, s_2), (h_3, s_1)$ and (h_3, s_2) را داریم.

این روش ترکیبی، در رویه SS بسیار مهم است. تمام زوجهای انتخاب شده در روش تولید زیر مجموعه قبلی تحت ترکیب قرار می گیرند. ما به راه حل انتخاب شده از مجموعه H به عنوان p1 و به بردار انتساب کارخانه انتخاب شده از S به عنوان p2 اشاره می کنیم. راه حل ترکیبی جدید، که به عنوان pn اشاره شده است، در ابتدا یکسان با p1 می باشد. این روش ترکیبی دارای n تکرار است. در هر تکرار، یک شغل از pn به طور تصادفی بدون تکرار انتخاب می شود، ، به طوری که در پایان همه شغل ها انتخاب شده اند. ما به این شغل به صورت تصادفی انتخاب شده به عنوان h اشاره می کنیم. اگر یک عدد تصادفی یکنواخت بین 0 و 1 (rand) توزیع شده کمتر از یک مقدار p معین باشد، این روش ترکیبی کنترل می کند که آیا شغل h به کارخانجات مختلف در pn و در این شغل به بردار انتساب کارخانه p2 منسوب شده است یا خیر. اگر این مورد برقرار باشد، شغل h از کارخانه فعلی آن در pn استخراج می شود و در تمام موقعیت های ممکن از کارخانه نشان داده شده در p2 مورد آزمایش قرار می گیرد. مکان نهایی شغل h، موقعیتی است که منجر به کمترین تفاوت زمان بین شروع و اغاز یک شغل در کارخانه نشان داده شده در p2 می شود. اگر rand بزرگتر یا مساوی از p باشد آنگاه این شغل به کارخانه دیگری منسوب نمی شود و دست نخورده باقی می ماند. اجازه دهید ما این مکانیزم ترکیبی را با استفاده از آن در مثالی با 10 شغل و 10 کارخانه بیشتر نشان دهیم. فرض کنید زیر مجموعه ها عبارتند از:

$$h_i = \begin{cases} 2,5,6,1 \\ 10,3,7 \\ 4,9,8 \end{cases} \quad \text{and } s_j = \{3,1,2,2,1,1,3,2,1,2\}$$

شغل به صورت تصادفی انتخاب شده، شغل 4 است و مقدار تصادفی 0. 12 است (p = 0. 2). از این رو، ما شغل 4 را چک می کنیم. این شغل در hi به کارخانه 3 و در s_j به کارخانه 2 منسوب می شود. بنابراین ما این شغل را از کارخانه 3 برمی داریم و به کارخانه 2 اختصاص می دهیم. برای قرار دادن این شغل در دنباله ای از شغل ها در کارخانه 2 ، 4 موقعیت ممکن به شرح زیر وجود دارد:

$$\begin{cases} 2,5,6,1 \\ 4,10,3,7, \\ 9,8 \end{cases} \quad \begin{cases} 2,5,6,1 \\ 10,4,3,7, \\ 9,8 \end{cases} \quad \begin{cases} 2,5,6,1 \\ 10,3,4,7, \\ 9,8 \end{cases} \quad \begin{cases} 2,5,6,1 \\ 10,3,7,4 \\ 9,8 \end{cases}$$

تفاوت زمان بین شروع و اغاز یک شغل برای هر راه حل محاسبه می شود و راه حل منجر به بهترین تفاوت زمان بین شروع و اغاز یک شغل، نشانه موقعیت جدیدی برای شغل 4 است. فرض کنید که کار بعدی به صورت تصادفی انتخاب شده، شغل 8 و مقدار تصادفی 0.43 است. از آنجا که این مقدار بیشتر از p = 0. 2 است، ما از تغییر موقعیت این شغل پرش می نماییم و به شغل بعدی می رویم. این روش برای تمام مشاغل تکرار می شود. شکل 1 نشان دهنده یک لیست شبه الگوریتمی از روش ترکیبی پیشنهادی است.

توجه داشته باشید که پارامتر p، شدت تنوع را کنترل می کند. خیلی کمتر از مقدار ap و pn اساسا شبیه به p1 خواهد بود در حالی که اگر p بزرگ باشد، بیشتر شغل ها به کارخانجات مختلف منسوب خواهد شد. آزمایش های اولیه نشان داد که یک مقدار کم p برای حفظ تنوع بسنده است. در بخش 4.1 ما با استفاده از روش های آماری صدا، دیگر پارامترهای مهم تر از روش ارائه شده SS را مدرج می نماییم.

3. 3. روش بهبود

روش اجرایی بهبود در هر تکرار SS برای هر pn راه حل به دست آمده با استفاده از روش ترکیبی راه حل استفاده می شود. توجه داشته باشید که ما آن را پس از تولید اولیه $RefSet$ صدق نمی کند. روش پیشنهادی، یک ساده سازی از روش Naderi VND و Ruiz (2010) است. دو روش جستجوی محلی مکررا اعمال می شوند تا زمانی که راه حل بهبود یافته با توجه به هر دو همسایگی به یک بهینه محلی می رسد. بطور دقیق تر، در اولین جستجوی محلی، برای هر کارخانه، هر شغل منسوب شده به آن کارخانه استخراج می شود و وارد تمام موقعیت های ممکن از توالی در آن کارخانه می شود. موقعیت منجر به بهترین تفاوت زمان بین شروع و اغاز یک شغل برای آن کارخانه انتخاب می شود. اگر یک بهبود در مقدار تفاوت زمان بین شروع و اغاز یک شغل آن کارخانه وجود داشته باشد، این روش تکرار می شود. بنابراین، در پایان اولین طجستجوی محلی، هر کارخانه شامل راه حل های بهینه محلی با توجه به همسایگی ورودی می شود. توجه داشته باشید که شتاب های Taillard (1990) نیز در اینجا استفاده می شوند.

در جستجوی محلی دوم، هر شغل از کارخانه حیاتی (کارخانه با حداکثر مقدار تفاوت زمان بین شروع و اغاز یک شغل) استخراج می شود و داخل تمام موقعیت های ممکن از دنباله ای از تمام کارخانه های دیگر می شود. این روش ادامه می یابد، در حالی که هیچ بهبودی در تفاوت حداکثر زمان بین شروع و اغاز یک شغل یافت نمی شود. با این حال، پس از آن که تفاوت حداکثر زمان بین شروع و اغاز یک شغل بهبود می یابد، جستجوی ثانویه محلی خاتمه می یابد و ما به اولین طرح جستجوی محلی همانند یک روش فرود متغیر همسایگی (VND) باز می گردیم. در مقابل، این فرآیند در صورتی پایان می پذیرد (و VND نیز پایان می پذیرد) که همه شغل ها از کارخانه حیاتی به تمام موقعیت ها از همه کارخانه های ناموفق دیگر وارد می شود. باز هم شتاب های Taillard (1990) در جستجوی دوم محلی نیز استفاده می شوند. توجه به این مورد مهم است که پس از بهبود در تفاوت زمان بین شروع و اغاز یک شغل حداکثر، تنها دو کارخانه تحت تاثیر قرار می گیرند (یک کارخانه که از آن شغل استخراج شده است و یکی دیگر که شغل وارد آن شده است) بنابراین، در هنگام استفاده دوباره از روش اولین جستجوی محلی، تنها این دو کارخانه مورد بررسی قرار می گیرند.

3.4 روش به روز رسانی مجموعه مرجع و روش راه اندازی مجدد

پس از استفاده از روش بهبود برای pn، ما باید بررسی نماییم که آیا این راه حل جدید در مجموعه H از RefSet گنجانیده می شود یا خیر. با الهام از طرح های تولید Ruiz و همکاران. (2006) و Vallada و Ruiz (2010)، pn در H گنجانده می شود اگر و تنها اگر: (1) تفاوت زمان بین شروع و اغاز یک شغل pn، بهتر از تفاوت زمان بین شروع و اغاز یک شغل با بدترین راه حل در مجموعه H باشد و (2) این منحصر به فرد است، یعنی، هیچ راه حل یکسان دیگری در مجموعه H وجود ندارد.

اگر تمام شرایط برآورده شوند، pn، بدترین راه حل را در مجموعه H جایگزین می نماید، در غیر این صورت، pn به سادگی کنار گذاشته می شود. توجه داشته باشید که ما برخی از مکانیسم های تنوع دقیقاً تشریح شده دیگر، مانند اضافه کردن یک شرط سوم را آزمایش نمودیم که به واسطه آن، pn نباید تنوع مجموعه H را کاهش دهد، حتی اگر بهتر از بدترین و منحصر به فرد باشد. با این حال، چک کردن مداوم تنوع، هزینه بر است و بعد از آزمایش های مفصل تر و کالیبراسیون ها (در اینجا به دلیل محدودیت فضا نشان داده نشده است) نتایج بهتری ارائه نشد. در نتیجه، ما بررسی تنوع را از SS کاهش دادیم. این نیز الگوریتم نهایی را ساده سازی می نماید. با این حال، پس از آزمایش های اولیه، حذف چک کردن تنوع نیز به یک همگرایی سریع به راه حل بهینه موضعی منجر شد. ما باید در نظر بگیریم که مجموعه H شامل راه حل های کامل می شود و اینها هرگز بعد از روش تولید تنوع اولیه متغیر نمی شوند. تنها مجموعه S، که شامل شغل تصادفی برای انتسابات کارخانه می شود به طور تصادفی در هر تکرار دوباره تولید می شود. بنابراین، ما یک رویه را برای تنظیم مجدد مجموعه H پس از یک تعداد تکرارها بدون بهبودها در بهترین راه حل می گنجانیم. این رویه ساده است؛ پس از a تکرار بدون بهبود در بهترین راه حل، بدترین 50 درصد از راه حل ها در مجموعه H دور انداخته می شوند و روش تولید تنوع برای تولید راه حل های جدید به کار گرفته می شود. یک اظهار نظر مهم اینست که این روش راه اندازی مجدد تا زمانی اعمال می شود که بهترین راه حل بهبود یابد، یعنی، شمارنده تکرارها بدون بهبود پس از اعمال رویه راه اندازی مجدد، دوباره تنظیم می شود.

روش SS پیشنهادی کامل در شکل شبه الگوریتم در شکل 2 معین است.

```
procedure Solution_Combination_Method(p, p1, p2)
 pn = p1
 for j := 1 to n do
  Take a random job, without repetition, from pn, let this job be h
  if (rand < p) then
   if factory assigned to h in pn <> factory assigned to h in p2 then
    Extract h from its factory in pn and assign it to the factory indicated in p2
    Insert h into all positions of the factory indicated in p2
    Place h at the position with the best C_max at the factory indicated in p2
   endif
  endif
 endfor
 return pn
end
```

شکل. 1. شبه الگوریتم روش ترکیبی راه حل.

```
procedure SS(b, l, a)
 Set p := 0.1; counter := 0
 Generate 25 solutions with the NEH2 heuristic  %diversification gen method
 Initialize set H with the best b solutions among the 25
 while (termination criterion not satisfied) do
  Generate new set S with l vectors randomly    %diversification gen method
  for i := 1 to b do        %subset generation method
   p1 = i-th solution from set H
   for j := 1 to l do       %subset generation method
    p2 = j-th solution from set S
    pn=solution_combination_method(p,p1,p2)
    pn'=solution_improvement_method(pn)
    reference_set_update_method(pn')
   endfor
  endfor
  if best solution in H has improved then counter := 0 else counter + +
  if counter > a then apply restart_procedure to set H
 endwhile
end
```

شکل 2. روش جستجوی پیشنهادی پراکنده (SS). به پارامترهای B و L توجه داشته باشید.

4. کالیبراسیون، مقایسه محاسباتی و تجزیه و تحلیل آماری

در این بخش، ما ابتدا جستجوی پراکنده ارائه شده کالیبره می نماییم. سپس ما یک مقایسه محاسباتی دقیق و جامع از روش جستجوی پراکنده پیشنهادی را در برابر بهترین روش های موجود از نوشته ها انجام می دهیم. ما با دقت جنبه های مقایسه، نمونه های آزمایش شده و تمام شرایطی که تعمیم و تکرار نتایج به دست آمده را تسهیل می نماید توضیح می دهیم.

4.1 کالیبراسیون روش جستجوی پراکنده پیشنهادی

ما تنها کالیبره کردن پارامترهای معنی دار را انتخاب کرده ایم. اندازه b از $RefSet$ به طور معمول بیشتر از 20 است (Marti و همکاران، 2006). با توجه به اینها و دیگر نشانه های به خوبی شناخته شده، عوامل زیر در سطوح زیر آزمایش می شوند که منتجر به 48 ترکیب می شود: (1) اندازه b از مجموعه H در $RefSet$. آزمایش شده در چهار سطح: $\{2, 5, 10, 15\}$ اندازه l از مجموعه H در $RefSet$. آزمایش شده در سه سطح: $\{2, 5, 10\}$ و (3) تعداد تکرارها قبل از راه اندازی مجدد رخ می دهد. آزمایش شده در چهار سطح: $\{10, 20, 30, 40\}$

Naderi و Ruiz (2010) دو مجموعه از موارد را برای DPFSP معرفی نمودند. اولین مجموعه شامل 420 نمونه کوچک از حدود 16 شغل، 5 ماشین آلات و 4 کارخانه می شود. این موارد کوچک برای حل مدل های MILP پیشنهادی در آن مقاله مورد استفاده قرار گرفتند و تلقی می شود که برای کالیبراسیون و آزمایش نیز آسان هستند. در نتیجه آنها در باقی مانده این مقاله استفاده نمی شوند. Naderi و Ruiz (2010) نیز مجموعه ای از 720 نمونه بزرگ را بر اساس 120 نمونه از Taillard (1993) ارائه دادند که دارای 12 مجموعه با ترکیبات مختلف زیر از تعداد n شغل و تعداد ماشین آلات (n x m) m است: $\{(20, 50, 100) \times (5, 10, 20)\}, \{200 \times (10, 20)\}$ and 500×20. هر ترکیب دارای 10 تکرار و در نتیجه 120 نمونه در کل است. همه این 120 نمونه با تعداد مختلف کارخانه ها در نظر گرفته می شوند. ما $F = \{2, 3, 4, 5, 6, 7\}$ داریم که به ما 720 نمونه را در کل ارائه می دهد. همه موارد از http://soa. iti. es در دسترس هستند.

لازم به ذکر است که اندازه گیری و کالیبراسیون جستجوی پراکنده پیشنهادی با استفاده از 720 نمونه از Naderi و Ruiz (2010) به تناسب یا کالیبراسیون بیش از حد منجر می شود. روش های کالیبراسیون در موارد مشابهی که آنها بعداً آزمایش می شوند، کار نادرستی است و به طور بالقوه ناعادلانه است. در عوض، ما مجموعه ای از 50 نمونه تصادفی را ارائه می دهیم. در این مجموعه، n، m و F به صورت تصادفی از ترکیب قبلی انتخاب می شوند. پس از انتخاب، زمان های پردازش به صورت تصادفی از یک توزیع یکنواخت در محدوده [99،1] همانطور که در نوشته های

زمانبندی رایج است، نمونه برداری می شوند. بنابراین، 50 نمونه کالیبراسیون، متفاوت از 720 نمونه آزمون می باشند. این موارد کالیبراسیون به صورت آنلاین نیز در دسترس می باشند.

ما از روش طراحی آزمایشات (DOEs) (Montgomery، 2012) برای کالیبراسیون استفاده کرده ایم. پیکربندی تجربی، یک آزمایش فاکتوریل کامل همانند بسیاری از ترکیبات قبلی (48) است. l,b و a عوامل کنترل شده هستند. متغیر پاسخ، انحراف درصد نسبی روی بهترین راه حل محاسبه شده شناخته شده برای هر نمونه، به شرح زیر است: $\mathrm{RPD} = \frac{Some_{sol} - Best_{sol}}{Best_{sol}} \cdot 100$. راه حل به دست آمده توسط هر یک از 48 پیکربندی های SS در یک مثال معین و $Best_{sol}$ کمترین تفاوت زمان بین شروع و اغاز یک شغل شناخته شده برای نمونه است. به منظور افزایش قدرت آزمایش، ما از 5 تکرار استفاده نمودیم که تعداد کل کارها را به $48 \times 50 \times 5 = 12,000$ افزایش می دهد. با چنین تعداد زیادی از نتایج، قدرت آزمایش مورد انتظار بالا است.

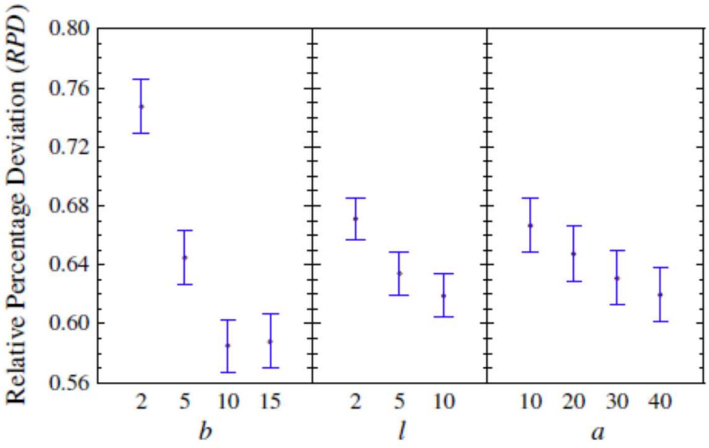
نتایج این آزمایش با استفاده از تجزیه و تحلیل روش واریانس (ANOVA) مورد تجزیه و تحلیل قرار گرفت. ANOVA یک ابزار پارامتری آماری است و سه فرضیه باید بررسی شوند. از اهمیت بیشتر به کمتر، استقلال از باقیمانده ها، هموواریانسی سطوح عامل و نرمال بودن باقیمانده (همچنین به عنوان همگنی واریانس شناخته می شود) این سه فرضیه می باشند. پس از چک کردن دقیق، ما انحراف قابل توجهی را از این فرضیه ها نیافتیم. توجه داشته باشید که غربالگری طراحی کامل فاکتوریل تجربی به هیچ وجه یک فرآیند کالیبراسیون شفاف و زیرتنظیم شده نیست. در واقع، یک طراحی کامل فاکتوریل تجزیه و تحلیل شده با استفاده از ANOVA را می توان به عنوان اولین قدم در کالیبراسیون الگوریتم در نظر گرفت. برای روش های جامع تر، خواننده به Bartz-Beielstein، Chiarandini، Paquete، و Preuss (2012) که در آن تکنیک های پیشرفته نشان داده شده است ارجاع داده می شود. دلیل انتخاب ما از کالیبراسیون ساده، چیزی غیر از جلوگیری از مقایسه ناعادلانه با روش موجود نیست. پس از همه اینها، اگر یک کالیبراسیون با تنظیم خوب کامل و گسترده روی روش های جستجوی پراکنده پیشنهادی انجام شود، ما قادر به اطمینان از ارزیابی محاسباتی نیستیم، اگر یک عملکرد بهتر به دلیل ساختار الگوریتم خوب و اپراتورها و یا فقط به خاطر فرآیند کالیبراسیون بهتر به دست آید.

برای آزمایشات محاسباتی، ما یک خوشه با 30 تیغه محاسبه را در اختیار داریم، که هر یک دارای دو پردازنده Intel XEON E5420 در حال اجرا در 2. 5 گیگاهرتز با 4 هسته در هر یک است (که 8 هسته را در هر تیغه می سازد). هر تیغه دارای 16 گیگابایت حافظه RAM است. بنابراین، در مجموع ما 240 هسته و 480 گیگابایت داریم. این خوشه برای ما، استفاده از بسیاری از دستگاه های مختلف مجازی را برای آزمایش که هر یک در حال اجرا در سیستم عامل ویندوز XP با یک پردازنده مجازی تک و 2 گیگابایت حافظه RAM هستند را میسر می سازد. این ماشین های مجازی برای آزمایش محاسباتی برای توزیع بار محاسباتی مورد استفاده قرار گرفتند.

تعیین معیار توقف معنی دار برای هر پیکربندی جستجوی پراکنده مهم است. یک خطای رایج در نوشته ها، هنگام کالیبره نمودن الگوریتم ها، ارائه تعداد ثابتی از تکرارها برای هر ترکیبی از عوامل است. بدیهی است، یک $RefSet$ بزرگتر نیاز به زمان CPU به طور قابل ملاحظه ای بزرگتر دارد و در نهایت می توان نتیجه گرفت که یک پیکربندی با مجموعه های بزرگتر بهتر است در حالی که حقیقت واقعی این است که فقط به این خاطر بهتر است که زمان پردازنده بیشتر میسر است. بنابراین، ما از معیار ختم زمان پردازنده سپری شده استفاده می نماییم که یک تابع از تعداد n شغل، تعداد m ماشین آلات و تعداد کارخانه F است. این به منظور مشاهده اثر آماری از عوامل مورد آزمایش قرار گرفته مورد نیاز است. اگر یک زمان CPU ثابت مورد استفاده قرار می گرفت، نمونه های کوچکتر در نهایت با نتایج بسیار خوبی خاتمه می یافتند، زمانی که زمان CPU نسبتا بزرگ استفاده می شد. این اثر برعکس نمونه های بزرگ خواهد بود که در آن همان زمان CPU احتمالا کافی نمی باشد. این سناریو، زمانی که زمانی که یک متغیر کمین کننده، " زمان CPU " اثر عوامل را بپوشاند، فاجعه بار خواهد بود. در نتیجه، ما از عبارت زیر به عنوان یک زمان CPU معیار ختم برای هر اجرا از پیکربندی های جستجوی پراکنده پیشنهادی استفاده می نماییم: $n \times m \times F \times C$، که در آن C در 10 تنظیم می شود و عبارت کلی برحسب میلی ثانیه است. این یک زمان CPU نسبتا کوتاه است، زمانی که برای بزرگترین نمونه از 500 شغل، 20 ماشین آلات و 7 کارخانه، مجموع مدت زمان CPU سپری شده 700 ثانیه باشد و درست 2 ثانیه برای کوچکترین موارد $20 \times 5 \times 2$ باشد.

نتایج ANOVA به شرح زیر خلاصه شده است (جدول ANOVA در اینجا به دلیل محدودیت فضا دوباره تولید نشده است، بلکه بر حسب درخواست از نویسندگان در دسترس است). همه سه ضریب b. l و a با نسبت های F بالا و مقادیر p بسیار نزدیک به صفر از نظر آماری معنی دار می باشد. بنابراین، از نظر آماری اختلافات معناداری در متغیر پاسخ بین سطوح عوامل مورد مطالعه وجود دارد. در جزئیات بیشتر، مهمترین عامل، اندازه b از مجموعه H است. مهمترین عامل دوم، اندازه l از مجموعه H و سومی، تعداد تکرارها است که پس از آن راه اندازی مجدد اعمال می شود (a). نمودارهای میانگین این عوامل، همراه با 95٪ تفاوت معنادار (HSD) فاصله اطمینان صادقانه Tukey در شکل 3 نشان داده شده است. باید ذکر نمود که بازه های زمانی اعتماد دارای همپوشانی نشان می دهند که تفاوت ها در متغیر پاسخ (RPD) ابزار دارای همپوشانی از نظر آماری معنی دار نیستند.

2 تعامل سطح بین عوامل، معنی دار نیستند. از نمودارها، ما می بینیم که سطوح 10 و 15 برای ضریب b از نظر آماری برابر هستند. همین امر برای سطوح 5 و 10 برای ضریب l صدق می کند. ما مقادیر 10 و 10 را به ترتیب را انتخاب می نماییم، همانطور که آنها با هم 20 را می سازند، یک اندازه ایده آل برای $RefSet$ با توجه به Marti و همکاران. (2006). برای تعداد تکرارها قبل از راه اندازی مجدد، سطح 40 معادل 20 و 30 است، اما به یک متوسط کمتر منجر می شود. اگر چه در اینجا نشان داده نشده است، مقادیر بزرگتر از 10 برای l و بزرگتر از 40 برای a منجر به عملکرد بدتر در آزمایش های تایید می شود. در نتیجه ما b و l را در 10 و 40 تثبیت می نماییم.



شکل 3. نمودار میانگین ها برای اندازه b از مجموعه H، اندازه l از مجموعه S و تعداد تکرارها قبل از راه اندازی مجدد برای آزمایش کالیبراسیون ANOVA SS. تمام میانگین های دارای فواصل تفاوت معنادار صادقانه (HSD) Tukey در سطح اطمینان 95٪ هستند.

4.2 روش های مقایسه شده و تنظیم تجربی

ما در حال حاضر جزئیات تنظیمات تجربی را برای کمپین محاسباتی ارائه می دهیم. روش های زیر در مقایسه گنجانده می شوند:

- یک فراابتکار الکترومغناطیس گسسته از Liu و Gao (2010) که به عنوان EM نامیده می شود. این الگوریتم شامل چهار همسایگی در مرحله جستجوی محلی VNS می شود. مقدار دهی اولیه تصادفی همانند طراحی نویسندگان.
- الگوریتم ترکیبی ژنتیک با جستجوی محلی (GA_LS) از Gao و Chen (2011a)، که HGA نامیده می شود. این الگوریتم ، NEH2 و VND(a) را به عنوان مقدار دهی اولیه به کار می گیرد.
- NEH بهبودیافته Gao و Chen (2011b)، که NEHdf نامیده می شود. توجه داشته باشید که در مقاله اصلی، جزئیات شاخه به کار گرفته شده و روش محدود استفاده شده در داخل NEHdf داده نشده اند. ما برای کمک و منبع کدها با نویسندگان تماس گرفتیم. کدهای منبع شدند به ما داده نمی شوند. در عوض، نویسندگان اصلی به با یک نسخه کمی گسترده از مقاله (Gao و Chen، 2011c) ارائه دادند. با این حال، این مقاله فاقد توضیحات کافی است. در پایان، چون شاخه و کران، تمام انتسابات ممکن کارخانه را به شمار می آورد و حداکثر مقدار F برابر 7 در محک است، ما دریافتیم که این در واقع با استفاده از همه شتاب های امکان پذیر برای آزمایش تمام 7! راه حل ممکن در هر مرحله از NEHdf سریعتر است. با استفاده از این مورد، ما به زمان های قابل مقایسه، اگر نه سریع تر، برای CPU می رسیم که در Gao و Chen (2011b) گزارش شده است.

- VND بهبود یافته از Gao و همکاران. (2012a)، که به عنوان VNS(B & B) ارجاع می شود. توجه داشته باشید که این الگوریتم اساسا ترکیبی از VND(a) و NEHdf قبلی است و ما با همان مسائل پیاده سازی مجدد مواجه می شویم.
- روش جستجوی ممنوعه Gao و همکاران. (2013)، که به عنوان TS نامیده می شود.
- بهترین الگوریتم تکرارشده حریص از Lin و همکاران. (2013) که نویسندگان IG_{VST} می نامند و به سادگی به عنوان IG در اینجا نامیده می شود.
- این مقایسه همچنین شامل روش اصلی ارائه شده در Naderi و Ruiz (2010) می شود، یعنی NEH1، NEH2، VND(a) و VND(b). توجه داشته باشید که VND(a) و VND(b) کمی اصلاح می شوند تا در یک زمان CPU مشخص و معین متوقف شوند، و نه بعد از اینکه بهینگی محلی به دست می آید. تأکید می شود که این روش دارای هیچ مکانیزم تنوعی نیست، بنابراین آنها در نهایت به یک راه حل بهینه محلی می رسند که از آن نمی توانند فرار کنند. در هر صورت، این تغییر در معیار توقف می شوند به منظور کاهش مقایسه میان روش معرفی می شود.
- ما در نهایت، در مقایسه، روش جستجوی پراکنده SS پیشنهادی را می گنجانیم.

در کل ما 11 روش را مقایسه می نماییم. همانطور که ما می توانیم از لیست قبلی و از بررسی نوشته های بخش 2 ببینیم، تنها دو الگوریتم در مقایسه محاسباتی گنجانده نشده اند. ما GA_KB از Gao و همکاران. (2012b) را دوباره پیاده سازی و آزمایش نمی کنیم، همانطور که مطابق با کار این نویسندگان، عملکرد بسیار شبیه به عملکرد HGA از Gao و Chen (2011a) است. همچنین، این مقاله دارای جزئیات اندکی است و بعید است که پیاده سازی مجدد مستقل از GA_KB بدون دسترسی به کد منبع موفقیت آمیز باشد. همانطور که در بخش 2 اظهار نظر شد، مقاله Wang و همکاران. (2013) بعد از اینکه همه آزمایش ها در این مقاله به پایان رسید منتشر شد. در هر صورت، و همانطور که ذکر شد، روش EDA ارائه شده در آن مقاله رقابتی نیست، که تا حدودی بهتر از VND(a) است، اما نیاز به زمان CPU خیلی بیشتر دارد. روشن است که این روش بسیار بدتر از سایر روش های اخیر مانند IG

یا TS بالا است و در نتیجه ما پیاده سازی دوباره آن را انتخاب نکرده ایم. با این حال، ما بعدا در این بخش مقایسه های غیر مستقیم در برابر EDA را ارائه خواهیم نمود.

توجه داشته باشید که تمام روش ها با دقت در C++ با توضیحات نویسنده اصلی در مقالات مربوطه کدنویسی شده اند. معیار توقف تمام روش ها اصلاح شده اند به طوری که تمام الگوریتم ها با استفاده از همان زمان CPU در تمام آزمایش ها خواهند بود. این زمان CPU از همان بیان کالیبراسیون روش جستجوی پراکنده پیشنهادی $(n \times m \times F \times C)$ را پیروی می نماید. با این حال، در این مورد، C در چندین مقدار، یعنی 20، 40، 60، 80 و 100 آزمایش شده است. این به این معنی است که زمان CPU به کار گرفته شده توسط همه روش ها در محدوده 4 ثانیه برای کوچکترین نمونه های $n = 20,\ m = 5,\ F = 2$ و کوتاه ترین زمان آزمایش شده C = 20-7000 ثانیه برای بزرگترین نمونه 500 x 20 x 7 و C = 100 هستند. توجه داشته باشید که ما هر روش را برای C = 100 آزمایش نمی کنیم و زمان ها در 20، 40 ، 60 و 80 ثبت می نماییم. در هر آزمون، هر الگوریتم از شروع دوباره آغاز می شود. این به اجتناب از خودهمبستگی در نتایج کمک می کند که برای آزمایش آماری بعدی مسئله ساز خواهد بود. آزمایش همه روش ها با 720 نمونه و با بسیاری از زمان های توقف آن که در محدوده از چند ثانیه تا تقریبا 2 ساعت هستند، طیف گسترده ای از نتایج و تجزیه و تحلیل آماری بی عیب را تضمین می کند. علاوه بر این، از آنجا که تمام الگوریتم ها در همان زبان کدنویسی شده اند و بر روی کامپیوتر با همان معیار زمان توقف CPU اجرا شده اند، ما یک کمپین محاسباتی کاملا قابل مقایسه را داریم. توجه داشته باشید که ارزیابی تفاوت زمان بین شروع و اغاز یک شغل، بیشتر اپراتورهای جستجوی محلی و روش مقدار دهی اولیه در میان روش ها مشترک هستند. اگر یک روش معین بهتر از دیگری کار کند، تنها می تواند به خود روش و نه به یک کامپیوتر سریع تر، برنامه نویسی بهتر است یا بار توقف مختلف نسبت داده شود.

در مجموع، ما 11 روش تست داریم. NEH1، NEH2 و NEHdf اکتشافی هستند و معیار توقف ندارند و تنها یک بار با هر نمونه آزمایش می شوند. همه 8 روش دیگر با 720 نمونه و با 5 بار توقف مختلف فوق مورد آزمون قرار گرفتند که بدان معنی است که ما 720 x 3 = 2160 نتایج برای الگوریتم اکتشافی و 720 x 5 x 8 = 28800

نتایج برای فراابتکاری داریم.با توجه به تعداد زیاد نتایج، ما از تکرارها استفاده نمی کنیم. زمان CPU کلی مورد نیاز برای نتایج فراابتکاری (بدون در نظر گرفتن کالیبراسیون جستجوی پراکنده یا الگوریتم اکتشافی) تقریبا 165 روز است. همان خوشه از کامپیوترها استفاده شده برای کالیبراسیون SS برای مقایسه استفاده می شوند.

4.3 نتایج اکتشافی برای نمونه های بزرگ

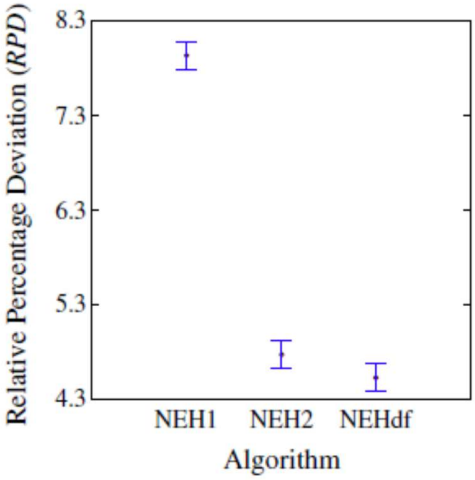
جدول 1 ، متوسط انحراف درصد نسبی را برای سه روش اکتشافی آزمایش شده، گروه بندی شده توسط تعداد کارخانه ها نشان می دهد. هر سلول شامل میانگین 120 نمونه در هر مقدار F می شود. زمان های CPU (بر حسب ثانیه) نیز فراهم می شوند.

NEH1 برای NEH2 تحتانی است که نتایج قبلی Naderi و Ruiz (2010) را تایید می نماید. در همان زمان، در نظر گرفته می شود که به طور متوسط، NEH1 تقریبا 4 برابر سریع تر است. NEHdf تنها کمی بهتر از NEH2 و همچنین حدود 50٪ کندتر است. هر سه روش اکتشافی در هر صورت فوق العاده سریع هستند که نیاز به این دارد که بدترین حالت کمتر از 0. 3 ثانیه (NEHdf برای نمونه اندازه 500 x 20 x 7) باشد. به یاد داشته باشید که Gao و Chen (2011b) نشان دادند که NEHdf از نظر آماری بهتر از NEH2 نیستند. اجازه دهید چک کنیم که آیا اگر این مورد برقرار است یا نه. ما یک ANOVA چند عاملی را انجام می دهیم که در آن n. m، F و روش های اکتشافی، عوامل کنترل هستند و میانگین انحراف درصد نسبی، متغیر پاسخ است. ما تنها نمودارهای میانگین الگوریتم عامل، که در شکل 4 داده می شود، را در نظر داریم.

	AVRPD			CPU time (seconds)		
F	NEH1	NEH2	NEHdf	NEH1	NEH2	NEHdf
2	6.35	4.58	4.29	0.007	0.015	0.029
3	7.17	4.82	4.45	0.006	0.018	0.030
4	8.21	5.00	4.51	0.005	0.022	0.032
5	8.51	5.03	5.01	0.005	0.026	0.035
6	9.32	5.40	5.27	0.005	0.028	0.038
7	10.28	6.04	5.91	0.004	0.032	0.049
Average	8.31	5.15	4.91	0.006	0.023	0.035

جدول 1: میانگین درصد نسبی انحراف (AVRPD) و زمان CPU مورد نیاز (در ثانیه) برای سه اکتشافی مورد آزمایش قرار گرفته گروه بندی شده توسط تعداد F کارخانه.

توجه داشته باشید که میانگین های ترسیم شده در واقع میانگین انحراف درصد نسبی برای همه 720 نمونه بزرگ هستند. همانطور که ما می توانیم ببینیم، ما نتایج قبلی Gao و Chen (2011b) را تایید می کنیم و نتیجه گیری می کنیم که در حالی که NEHdf نتایج کمی بهتر از NEH2 را به دست می آورد، این تفاوت ها بزرگ و / یا به اندازه کافی سازگار نیستند، بنابراین از نظر آماری معنی دار نیستند. در نتیجه، NEH2 روش ارجح معین نیز است که NEHdf، دارای پیاده سازی دشوار است و کندتر از NEH2 است.



شکل. 4. نمودار میانگین ها برای الگوریتم های اکتشافی. تمام میانگین دارای بازه های زمانی تفاوت معنادار صادقانه Tukey (HSD) در سطح اطمینان 95٪ هستند.

4.4 نتایج فراابتکاری برای نمونه بزرگ

جدول 2 نتایج از 8 الگوریتم فراابتکاری مورد آزمایش قرار گرفته را خلاصه نموده است. نتایج به دست آمده توسط هر سطح معیار زمان توقف CPU (C) و همچنین در تعداد F کارخانه گروه بندی می شوند. دوباره، هر سلول شامل متوسط 120 نتایج گروه بندی می شود. حتی اگر زمان توقف، یک معادله تثبیت شده برای هر نمونه و روش $(n \times m \times F \times C)$ باشد، آخرین ستون، میانگین زمان CPU (در ثانیه) را به عنوان یک هدایت ارائه می دهد.

نتایج ارزیابی محاسباتی شامل برخی از یافته های مهم می شود. اول از همه، ما عملکرد بهتر VND(a) را در مقابل VND(b) همانطور که در Naderi و Ruiz (2010) توضیح داده شد، تایید می نماییم. با این حال، بودن فقط روش های جستجوی محلی که در یک بهینه محلی و بدون هیچ گونه روش تنوع متوقف می شوند، VND(a) و VND(b) عملکرد خود را با زمان CPU اضافی بهبود نمی بخشند. هر دو روش قبل از اینکه کوتاه ترین زمان های CPU از C = 20 به دست آید، با هم فراهم می شوند. در واقع، و همانطور که Naderi و Ruiz (2010) نشان داده است، هر دو روش راه حل های خود را در 0.147 و 0.096 ثانیه به طور میانگین می یابند. روش های دیگر، راه حل های خود با بار اضافی بهبود نمی بخشند. یک مثال VNS(B & B) از Gao و (2012a) همکاران است. در هر صورت، انحراف میانگین به اندازه 2.68٪ به وضوح زیر VND (a) و VND(b) است که نتایج به دست آمده توسط نویسندگان اصلی را تایید می کند.

یک یافته مهم حاصل از این ارزیابی اینست که روش EM از Liu و Gao (2010)، جدا از این که به عنوان راه حل بهتری با زمان CPU اضافی یافت نمی شود، اینست که عملکرد آن در زیر تمام الگوریتم های فراابتکاری مورد آزمایش قرار گرفته دیگر در درصد نسبی 5.21٪ انحراف قرار دارد. در واقع، این انحراف بزرگتر از انحراف NEHdf و NEH2 از جدول 2 است. اگر ما میانگین زمان های CPU را در کوتاه ترین آزمایش از C = 20 اتخاذ نماییم، EM به 164.63 ثانیه به طور میانگین نیاز دارد، در حالی که NEHdf و NEH2 تنها به 0.035 و 0.023 ثانیه نیاز دارد. این به این معنی است که EM عملکرد مشابه را به دست می آورد اما در عین حال به مقدار گزافی از زمان CPU نیاز دارد که بین 4704 و 7157 بار طولانی تر است. توجه داشته باشید که در مقاله خود، Liu و Gao (2010) ادعا نمودند که 151 راه حل بهترین و شناخته شده از 720، از Naderi و Ruiz (2010) را بهبود داده اند. ما می خواهیم تاکید می نماییم که این مقایسه ها اغلب گمراه کننده هستند. ما ادعا نمی کنیم که نتایج آنها، بهترین راه حل های شناخته شده را بهبود نمی بخشد. در واقع، پیاده سازی مجدد ما از EM، نه 151 بلکه 161 راه حل بهترین شناخته شده را زمان مقایسه با بهترین راه حل اصلی داده شده در Naderi و Ruiz (2010) بهبود می بخشد، (پس از آن به نظر می رسد که پیاده سازی ما از EM در واقع کمی کارآمدتر است). واقعیت این است که

559 نمونه دیگری است که در آن EM بهترین راه حل های شناخته شده را بهبود نمی بخشد. نتیجه این است که حتی اگر Liu و Gao (2010)، 151 بهترین راه حل اصلی شناخته شده از Naderi و Ruiz (2010) را بهبود ببخشند، عملکرد متوسط خود، زمانی که با حالت سیب ها برای سیب ها مقایسه می شود، ضعیف است. در مجموع، بهبود کسری از بهترین راه حل های شناخته شده، نشان دهنده عملکرد خوب نیست. Liu و Gao (2010) در مقایسه EM خود در برابر VND(a) از Naderi و Ruiz (2010) نشان داد که VND(a) در عملکرد حدود 43٪ بهتر است و با توجه به نتایج این مقاله و در Naderi و Ruiz (2010)، در حدود 1120 برابر سریعتر می باشد.

جدول 2: میانگین درصد انحراف نسبی (AVRPD) و زمان CPU استفاده شده (برحسب ثانیه) برای الگوریتم های فراابتکاری مورد آزمایش قرار گرفته دسته بندی شده بر اساس زمان CPU محدود C و تعداد F کارخانجات. مقادیر پررنگ، نشان دهنده بهترین نتایج است.

C	F	EM	HGA	IG	SS	TS	VND(a)	VND(b)	VNS(B&B)	CPU time
20	2	4.33	2.72	2.51	**0.98**	1.66	2.77	3.02	2.56	73.17
	3	4.92	3.09	2.69	**1.02**	2.08	3.02	3.43	2.44	109.75
	4	4.90	3.34	2.74	**1.18**	2.82	3.27	3.27	2.44	146.33
	5	5.15	3.62	2.66	**1.50**	3.48	3.72	3.72	2.47	182.92
	6	5.42	3.95	2.50	**1.87**	4.05	4.08	4.08	2.77	219.50
	7	6.18	4.76	2.49	**2.44**	4.98	4.92	4.77	3.41	256.08
Average		5.15	3.58	2.60	**1.50**	3.18	3.63	3.78	2.68	164.63
40	2	4.68	2.63	2.37	**0.88**	1.54	2.77	3.02	2.57	146.33
	3	4.86	3.00	2.57	**0.96**	1.99	3.02	3.43	2.56	219.50
	4	5.00	3.32	2.45	**1.06**	2.76	3.27	3.56	2.45	292.67
	5	5.19	3.56	2.40	**1.29**	3.46	3.72	3.79	2.40	365.83
	6	5.46	3.87	2.30	**1.75**	3.96	4.08	4.08	2.73	439.00
	7	6.16	4.74	2.25	**2.25**	4.97	4.92	4.77	3.32	512.17
Average		5.23	3.52	2.39	**1.36**	3.11	3.63	3.78	2.67	329.25
60	2	4.73	2.56	2.27	**0.80**	1.46	2.77	3.02	2.62	219.50
	3	4.87	2.99	2.41	**0.81**	1.93	3.02	3.43	2.49	329.25
	4	5.00	3.26	2.46	**0.95**	2.70	3.27	3.56	2.50	439.00
	5	5.40	3.53	2.34	**1.25**	3.42	3.72	3.79	2.43	548.75
	6	5.54	3.85	2.25	**1.66**	3.84	4.08	4.08	2.66	658.50
	7	6.03	4.72	**2.07**	2.16	4.88	4.92	4.77	3.38	768.25
Average		5.26	3.49	2.30	**1.27**	3.04	3.63	3.78	2.68	493.88
80	2	4.52	2.56	2.21	**0.76**	1.43	2.77	3.02	2.63	292.67
	3	4.98	2.99	2.42	**0.75**	1.92	3.02	3.43	2.52	439.00
	4	4.93	3.22	2.43	**0.93**	2.66	3.27	3.56	2.41	585.33
	5	5.18	3.52	2.32	**1.16**	3.44	3.72	3.79	2.45	731.67
	6	5.58	3.83	2.11	**1.64**	3.94	4.08	4.08	2.78	878.00
	7	5.81	4.67	**1.96**	2.20	4.93	4.92	4.77	3.30	1024.33
Average		5.17	3.47	2.24	**1.24**	3.05	3.63	3.78	2.68	658.50
100	2	4.65	2.49	2.10	**0.70**	1.42	2.77	3.02	2.65	365.83
	3	4.84	2.90	2.34	**0.69**	1.96	3.02	3.43	2.50	548.75
	4	5.11	3.18	2.34	**0.90**	2.62	3.27	3.56	2.46	731.67
	5	5.32	3.52	2.22	**1.14**	3.28	3.72	3.79	2.41	914.58
	6	5.48	3.80	1.94	**1.57**	3.89	4.08	4.08	2.69	1097.50
	7	6.09	4.67	**1.99**	2.14	4.92	4.92	4.77	3.31	1280.42
Average		5.25	3.43	2.16	**1.19**	3.02	3.63	3.78	2.67	823.13
Tot. average		5.21	3.50	2.34	**1.31**	3.08	3.63	3.78	2.68	493.88

روش HGA از Gao و Chen (2011a)، همانطور که در بخش 2 در مورد آن اظهار نظر شد، بهتر از VND(a) عمل می کند اما در یک مزیت زمان CPU ناعادلانه. در آزمایشات، در این مقاله، همان زمان های CPU به کار گرفته

می شوند و ما تایید می کنیم که در واقع HGA منجر به یک درصد متوسط انحراف نسبی کمی بهتر از VND(a) (3. 50٪در مقابل 3.63٪) با این حال، باید دید که آیا این تفاوت کوچک در عملکرد در واقع از نظر آماری قابل توجه است یا خیر. در پایان این بخش، ما تجزیه و تحلیل های آماری اضافی را انجام خواهیم داد که این سوال را تایید می کند. TS از Gao و همکاران. (2013) عملکرد بهتر VND(a) و HGA را تایید می کند که نتایج از مقاله اصلی را تصویب می کند. با این حال، بسیار جالب است که پیاده سازی ما از VNS(B & B) از Gao و همکاران. (2012a) بسیار بهتر از هر دوی HGA و TS به نظر می رسد، البته VNS(B & B) در مقایسه این مقاله آخر Gao و همکاران. (2013) ذکر نشده و مورد استفاده قرار نگرفته است.

ما همچنین می توانیم در مورد روش IG اخیر Lin و همکاران. (2013) اظهار نظر نماییم. ما می توانیم ببینیم که IG، تمام الگوریتم های فراابتکاری موجود دیگر را به وضوح پشت سر می گذارد و انحراف درصد را به اندازه 2.34٪ پایین می آورد. IG، روش های ساده هستند و به همین دلیل ما با خیال راحت می توانیم روی EM، HGA، TS و IG VNS(B & B) توصیه نماییم. VND(a) و VND(b) در واقع بسیار سریعتر هستند و در نتیجه باید به طور جداگانه در نظر گرفته شوند. علاوه بر این، و به همین ترتیب برای HGA و TS، نتایج IG به طور پیوسته، زمانی که زمان CPU بیشتری ارائه می شود، بهبود می یابد. به عنوان مثال، IG با C = 20 دارای انحراف میانگین 1.60٪ مقایسه با انحراف در مقایس با 2.60٪ برای C = 100 است.

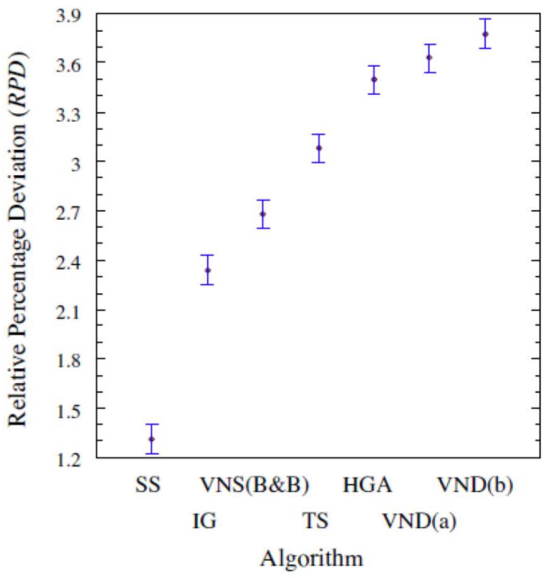
ما در نهایت در مورد نتایج حاصل از روش جستجوی پیشنهادی پراکنده SS اظهار نظر می نماییم. ما می توانیم ببینیم که انحراف کلی درصد نسبی فقط 1.31٪ است که تقریبا 79٪ بهتر از نزدیک ترین IG رقیب است. به جز در برخی از موارد جدا شده با F = 7 و زمان های CPU بزرگ که در آن IG، راه حل های کمی بهتر را مدیریت می نماید، SS، به مراتب، پایین ترین انحراف را در همه موارد به دست می آورد. از نتایج گزارش شده، همه نشانه ها اینست که SS پیشنهادی، یک عملگر بهتر پس از آزمایش در طیف گسترده ای از زمان های CPU و نمونه ها است. علاوه بر این، لازم به ذکر است که راه حل های ارائه شده توسط روش SS پیشنهادی هنگامی که با کوتاه ترین زمان CPU از C = 20 آزمایش شود، 719 از 720 راه حل شناخته شده گزارش شده در Naderi و Ruiz (2010) را

بهبود می دهد. برای مثال تک که در آن راه حل بهبود نیافته است، تفاوت زمان بین شروع و اغاز یک شغل گزارش شده فقط یک واحد بزرگتر است. برای C = 40، تمام 720 بهترین راه حل شناخته شده قبلاً بهبود می یابند. به طور مقایسه ای، الگوریتم اخیرا پیشنهادی IG از Lynn و همکاران. (2013) گزارش شده در آن مقاله، " تقریبا نیمی " از نمونه های Naderi و Ruiz (2010) را بهبود بخشید. یک آزمون غیرمستقیم مشابه را می توان با الگوریتم آزمایش نشده EDA از Wang و همکاران (2013) انجام داد. به یاد داشته باشید که این الگوریتم دوباره پیاده سازی نشد چرا که به تازگی پیشنهاد شده است. با این حال، نویسندگان بهترین جواب های یافته شده خود را گزارش نمودند، بنابراین مقایسه غیرمستقیم امکان پذیر است. در مقاله خود، نویسندگان 589 بهترین راه حل های جدید را ادعا نمودند. در ضمیمه، آنها فقط 585 بهترین راه حل را گزارش نمودند. با مقایسه همه این مقادیر در برابر بهترین و شناخته شده ترین راه حل های به دست آمده جدید ما در این مقاله، نتیجه می گیریم که روش EDA، انحراف متوسط درصد نسبی 2.28٪ را در این 585 نمونه تولید می کند که آنها بهبود می دهند. لازم به ذکر است که این بهترین سناریو است، از آنجایی که ما تنها 585 نمونه را از نویسندگان در نظر می گیریم که برای بهترین مقادیر شناخته شده از Naderi و Ruiz (2010) بهبود یافت. به طور مقایسه ای، SS پیشنهادی در این مقاله، برای کوتاه ترین زمان آزمایش شده CPU از C = 20 فقط منجر به یک انحراف 1.62٪ برای همان 585 نمونه شد. در واقع، برای این 585 نمونه، SS پیشنهادی (C = 20) بهتر از یا برابر با EDA در 508 مورد است. اگر چه آزمایشات بر روی کامپیوترهای مختلف انجام می شوند (کامپیوتر مورد استفاده توسط Wang و همکاران (2013) که سریعتر از 3.2 گیگاهرتز از2.5 گیگاهرتز هستند)، SS پیشنهادی ما با C = 20 دارای زمان CPU قابل مقایسه با خواسته های به EDA از Wang و همکاران. (2013) است. در نتیجه، می توانیم بیان کنیم که حتی اگر مقایسه به طور غیر مستقیم قرار گیرد، SS پیشنهادی راه حل بسیار بهتری را از EDA از Wang و همکاران. (2013) به دست می آورد. راه حل های جدید و بهبود یافته به دست آمده در این مقاله در http://soa. iti. es موجود هستند.

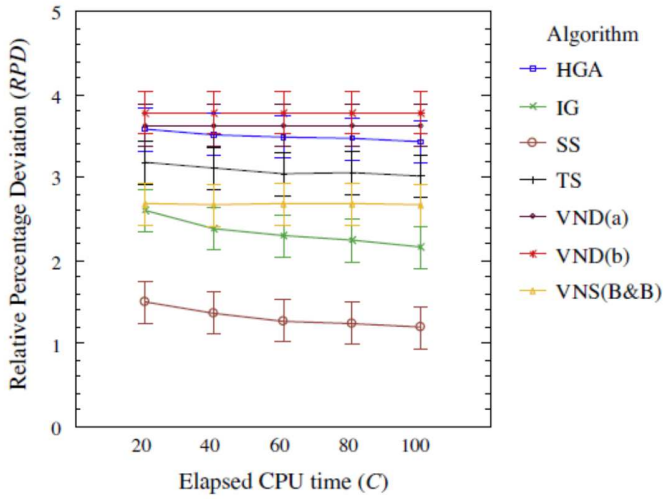
در حالی که تفاوت ها بین SS پیشنهادی و الگوریتم های فراابتکاری موجود گزارش شده در جدول 2 به وضوح به اندازه کافی از نظر آماری معنی دار بزرگ هستند، ما هنوز هم ANOVA را جهت چک کردن انجام می دهیم و

تفاوت های مشاهده شده در واقع از نظر آماری قابل توجه هستند. در ANOVA خلاصه شده اول، ما روش EM را در برابر NEH2 و NEHdf بررسی می نماییم. با توجه به دلایل فضا، ما نمودارهای میانگین را گزارش نمی کنیم، بلکه ما سوء ظن های خود را تایید می کنیم که میانگین انحراف درصد نسبی کلی EM از نظر آماری بهتر از NEH2 یا NEHdf نیست.

به منظور داشتن یک تصویر واضح تر، NEH1، NEH2، NEHdf و EM برای تجزیه و تحلیل آماری بعدی حذف می شوند. ANOVA دیگر در جایی انجام می شود که در آن F و C، همراه با نوع الگوریتم، عوامل کنترل شده هستند. تا کنون، مهم ترین اثر از عامل الگوریتم با نسبت F تقریبی 423 و P-مقدار بسیار نزدیک به صفر می آید. عامل F (همچنین به عنوان n و m در صورتی که ما آزمایش را افزایش دهیم) بسیار مهم هستند. در مقابل، عامل C بسیار معنی دار نیست. این یک نتیجه از الگوریتم های بسیاری است که عملکرد مشابه رل صرف نظر از زمان CPU به کار گرفته شده نشان می دهند. شکل. 5 نشان دهنده نمودار میانگین های الگوریتم عامل میانگین گیری شده در تمام 720 نمونه و تمام مقادیر C (3600 داده میانگین گیری شده در مرکز هر فاصله).



شکل. 5. نمودار میانگین ها برای الگوریتم های فراابتکاری (به استثنای EM). تمام میانگین ها دارای فواصل تفاوت صادقانه معنادار (HSD) Tukey در سطح اطمینان 95٪ هستند.



شکل. 6. نمودار میانگین ها برای تعامل بین حد زمانی پردازنده C و الگوریتم فراابتکاری (به استثنای EM). تمام میانگین ها دارای فواصل تفاوت صادقانه معنادار (HSD) Tukey در سطح اطمینان 95٪ هستند.

همانطور که ما می بینیم، تفاوت های نشان داده شده در جدول 2 ، برای اکثر روش،ها از نظر آماری قابل توجه هستند. SS ازآماری بهتر از IG است که به نوبه خود بهتر از VNS(B & B) است که باعث بهبود در TS می شود که دوباره بهتر از HGA است. با این حال، HGA با VND(a) از نظر آماری معادل است. این، آزمایشات Gao و Chen (2011a) را تایید می کند که نشان داد که با گذشت زمان CPU مشابه، HGA در واقع بدتر از VND(a) بود. پیاده سازی ما از HGA، نتایج کمی بهتر را از VND(a) نشان می دهد هنگامی که در همان زمان CPU اجرا می شود، اما این تفاوت از نظر آماری معنی دار نبود. علاوه بر این، به یاد داشته باشید که حتی اگر در سراسر این مقاله VND(a) و VND(b) برای همان زمان CPU همانند تمام روش های دیگر اجرا شوند، در واقع آنها کسری از زمان را مصرف می نماید. به عنوان یک نتیجه، HGA نمی تواند برتر از VND(a) توصیه شود. در نهایت، VND(a) از نظر آماری بهتر از VND(b) بود، که دوباره با آزمایش های انجام شده در Naderi و Ruiz (2010) موافق است.

مطالعه تعامل بین حد زمان پردازنده CPU و الگوریتم، که در شکل 6 نشان داده شده است، نیز جالب است. ما تایید می کنیم که برای بسیاری از الگوریتم ها، راه حل های بهتر با زمان CPU اضافی به دست نمی آیند. SS از نظر آماری بهتر از IG برای همه دوره های زمانی است و این در میان تمام روش ها دیده می شود، تنها مواردی که

به تدریج با زمان بیشتر CPU بهبود می یابند، IG و SS هستند، اگر چه بیشتر زمان ها، این تفاوت ها به اندازه کافی از آماری معنی دار هستند. با این حال توجه داشته باشید که عرض فواصل تفاوت معنادار صادقانه Tukey، با افزایش تعداد نتایج کوتاه می شود. اگر ما هر الگوریتم (تکرار) را چند بار بیشتر برای هر نمونه و مقدار C اجرا کنیم، تفاوت ها برای IG و SS با افزایش C قابل توجه خواهد بود.

به عنوان نکته آخر، آزمایش های متمرکز کوچک بین IG و SS برای C = 60، 80 و 100 و F = 7 (موارد که در آن، با توجه به میانگین گزارش شده در جدول 2، IG بهتر از SS است) نشان می دهد که تفاوت های کوچک به اندازه کافی از نظر آماری معنی دار نیست. به عنوان یک قانون کلی، اختلاف بین دو میانگین باید روی تعداد زیادی از موارد سازگار باشد و / یا به اندازه کافی برای معنی دار بودن از نظر آماری بزرگ باشد. این واقعیت، به دور از نامطلوب بودن در آزمایش آماری، ستون فقرات واقعی قابلیت تعمیم ANOVA است. یک روش A باید از روش دیگر B و روی تعداد زیادی از موارد آزمون به منظور تعمیم نتایج به موارد و جمعیت های دیگر به طور قابل توجهی بهتر باشد. در غیر این صورت، عملکرد 0.1٪ بهتر روی یک تعداد کمی از موارد آزمایشگاه در یک روش نسبت به روش دیگر B نمی تضمین نمی کند که در خارج از آزمایشگاه این تفاوت ها صادق می باشند.

5. نتیجه گیری ها و تحقیقات آینده

مسئله جابجایی جریان فروشگاه توزیع شده (DPFSP)، یک گسترش جالب چند کارخانه ای از جریان فروشگاه منظم است که به تازگی توسط Naderi و Ruiz (2010) پیشنهاد شده است. این نویسندگان در ابتدا شش مدل جایگزین صحیح خطی مختلط برنامه نویسی و همچنین دو الگوریتم اکتشافی ساده (NEH1 و NEH2) را بر اساس الگوریتم های اکتشافی جریان فروشگاه Nawaz و همکاران (1983) تکمیل شده با شغل کارآمد برای قوانین انتساب کارخانه ارائه نمودند. این نویسندگان همچنین دو الگوریتم فرود همسایگی متغیر ساده VND (a) و VND(b) را ارائه نمودند. پس از این شغل اولیه، تعدادی از نویسندگان، تعدادی از روش ها را ارائه نمودند و به طور عمده آنها را در برابر بهترین روش در زمان مقایسه نمودند - VND(a). در این تحقیق پیگیرانه، ما دوباره DPFSP را مورد بررسی قرار دادیم و روش موثر پراکندگی جستجو (SS) را پیشنهاد نموده ایم. مشخصه اصلی از SS ارائه شده،

یک RefSet ترکیبی ساخته شده از راه حل های کامل و همچنین بردار انتساب شغل به کارخانه است. روش ترکیبی راه حل، ترکیبی از همه راه حل های کامل با تمام بردارهای انتساب شغل به کارخانه است. این منجر به یک استراتژی موثر می شود، همانطور که روش بهبود راه حل در جایگشت های شغل در هر کارخانه کار می کند و روش ترکیبی به بررسی شغل های مختلف موثر برای انتسابات کارخانه می پردازد. همراه با یک روش به روز رسانی مجموعه مرجع دقیق و مکانیزم تنظیم مجدد، SS پیشنهادی، آخرین وضعیت این عملکرد را نشان می دهد.

ما یک تجزیه و تحلیل محاسباتی دقیق را انجام دادیم که در آن بسیاری از روش های موجود از نوشته ها با دقت در یک مجموعه جامع از 720 نمونه پیاده سازی و آژمایش شده اند. تقریبا 165 روز از زمان CPU در آزمون ها به کار گرفته شده است که در آن تمام الگوریتم ها در 5 بار توقف مختلف آزمایش می شوند. نتایج محاسباتی با تجزیه و تحلیل های آماری بی عیب با استفاده از طراحی آزمایشات و تجزیه و تحلیل تکنیک های واریانس همراه شد. نتایج نشان می دهد که SS پیشنهادی بهتر با اختلاف آماری گسترده ای از تمام روش های موجود عمل می کند، از جمله روش هایی که بسیار به تازگی پیشنهاد شده اند. کمک دیگر از این مقاله، مقایسه میان سایر روش های موجود است. بسیاری از الگوریتم ها، زمانی که در یک سناریوی کاملا قابل مقایسه آزمایش می شوند، اغلب نشان دهنده عملکردی هستند که در آزمایش های اولیه مشاهده نشده بود. برای مثال، ما نشان داده ایم که عملکرد الگوریتم EM، با وجود ادعای نویسندگان اصلی، رقابتی نیست. از سوی دیگر، پیاده سازی مجدد ما از روش VNS(B & B) عملکرد امیدوار کننده ای را نشان می دهد حتی اگر نویسندگان اصلی این روش را در آخرین مطالعه منتشر شده خود مقایسه نکرده باشند.

DPFSP، یک مسئله زمانبندی اخیرا پیشنهاد شده است و راه های زیادی برای تحقیقات آینده را باز می کند. هیچ تحقیق گزارش شده ای در مورد DPFSP با اهداف دیگر به غیر از تفاوت زمان بین شروع و اغاز یک شغل وجود ندارد. علاوه بر این، این مسئله نباید تعمیم داده شود، زیرا همه کارخانه ها به طور کامل یکسان نیستند. جنبه های دیگر می تواند شامل تسطیح بار در میان کارخانه ها و یا توجه به محدودیت مهم عمر واقعی مانند عملیات مونتاژ و زمان راه اندازی باشد.