انجمن کامپیوتر ایران
Computer Society of Iran

یک الگوریتم ژنتیک چند هدفه سریع بدون بازدید مجدد پاسخ‌ها

غلامحسین اکباتانی فرد *
گیلان، دانشگاه آزاد اسلامی واحد لاهیجان، گروه مهندسی کامپیوتر
Ekbatanifard@stu-mail.um.ac.ir

رضا منصفی
خراسان رضوی، دانشگاه فردوسی مشهد، گروه مهندسی کامپیوتر
Monsefi@um.ac.ir

چکیده: در این مقاله یک الگوریتم ژنتیک چند هدفه سریع با امکان حذف بازدید مجدد پاسخ‌ها، ارائه شده است. الگوریتم پیشنهادی با به خاطر سپردن نقاطی که از قبل مورد جستجو قرار داده است از بازدید مجدد آن‌ها در نسل‌های بعدی جلوگیری نموده و در صورت مشاهده‌ی پاسخ تکراری با اعمال عملگر جهش وقفی به یک پاسخ غیرتکراری می‌رسد. در واقع الگوریتم پیشنهادی توابع برازش را برای پاسخ‌های تکراری بررسی نمی‌کند از این رو باعث کاهش هزینه محاسباتی الگوریتم خواهد شد همچنین با اعمال جهش وقفی، تنوع پاسخ‌ها را افزایش می‌دهد. شبیه‌سازی‌های انجام شده نشان داده است که الگوریتم پیشنهادی دارای همگرایی و تنوع بهتری در رسیدن به جواب‌های بهینه، نسبت به الگوریتم ژنتیک چند هدفه NSGA-II است.

واژه‌های کلیدی: الگوریتم ژنتیک، بهینه سازی چند هدفه، بهینه پارتو، بازدید مجدد، جبهه جلویی، غالب بودن

۱- مقدمه

وجود توابع هدف مختلف در یک مساله در واقع به جای نیل به یک جواب، موجب یک مجموعه جواب بهینه می‌شود که به آن جواب‌های بهینه پارتو[1] می‌گویند. در صورتی‌که هیچ قیدی مطرح نباشد هیچ یک از این جواب‌های بهینه به دیگری غالب نخواهد بود. این باعث می‌شود که کاربر دنبال این باشد که تا حد امکان بزرگترین مجموعه جواب پارتو را بیابد. روش‌های بهینه‌سازی کلاسیک برای حل یک مساله بهینه‌سازی چند هدفه پیشنهاد می‌کنند که مساله به یک مساله تک هدفه با تاکید بر یک جواب بهینه پارتو خاص در هر بار تبدیل شود. هنگامی‌که چنین روشی برای یافتن جواب‌های متعدد مورد استفاده قرار می‌گیرد، باید تعداد دفعات زیادی اجرا گردد با این امید که در هر بار اجرا جواب متفاوتی پیدا شود[۱].

در دهه گذشته چندین روش الگوریتم تکاملی (ژنتیک) چند هدفه پیشنهاد شده است[۱،۲،۳،۴،۵]. دلیل اصلی آن توانایی یافتن چندین راه‌حل بهینه پارتو، در یک اجرای الگوریتم است. NSGAII [۱] یکی از سریع‌ترین و جدیدترین الگوریتم‌های ژنتیک چند هدفه مبتنی بر مرتب سازی غیرغالب[2] است که از پیچیدگی زمانی مناسب $O(MN^2)$ (که M تعداد توابع هدف و N اندازه جمعیت است) برای مرتب‌سازی برخوردار است و همچنین دارای یک عملگر انتخاب است که با ادغام جمعیت والد و جمعیت فرزندان، یک مخزن جفت‌گیری[3] ایجاد می‌کند آنگاه N تا از بهترین جواب‌ها را با توجه به مقدار برازندگی و گسترش انتخاب می‌کند.

الگوریتم‌های تصادفی[4] مانند الگوریتم ژنتیک، مکانی (عنصری) را که قبلاً بازدید کرده و مورد بررسی قرار داده‌اند در حافظه نگه‌داری نمی‌کنند. البته جستجوی تابو[۶] به عنوان یک استثنا، از نتایج جستجوهای اخیر برای هدایت گام بعدی جستجو استفاده می‌کند. هر چند در روش جستجوی تابو نیز کل مکان‌های بازدید شده از قبل، در حافظه ثبت نمی‌شود. لذا برای روش‌های موجود، بازدید مجدد مکان‌ها ناگزیر است[۷].

ارزیابی مجدد تابع برازش برای عنصری مانند S که قبلاً مورد ارزیابی قرار گرفته است را بازدید مجدد[5] گویند. از آنجائی‌که در بسیاری از کاربردهای دنیای واقعی، به خصوص کاربردهایی که نیاز به بهینه‌سازی چند هدفه دارند، ارزیابی تابع برازش یکی از فرایندهای حساس محاسباتی در الگوریتم ژنتیک است، لذا واضح است که بازدید مجدد باعث اتلاف منابع محاسباتی می‌شود. به عنوان نمونه از جمله کاربردهای دنیای واقعی که محاسبه تابع برازش در آن‌ها هزینه‌بر و زمان‌بر هستند

[1] Pareto optimal

* نویسنده اصلی
[2] Non-dominated Sorting
[3] Mating pool
[4] Stochastic
[5] Revisit s

ثبت اشیائ سه بعدی در بینایی ماشین [۹] و مهندسی HVAC[۶] [۱۰] را می‌توان نام برد.

در قوانین NFL[۷] [۷،۸] آمده است که اگر توزیع مساله یکنواخت باشد تمامی الگوریتم‌ها (چه تصادفی و چه معین) دارای یک میانگین کارایی خواهند بود. یک الگوریتم P، دارای بازدید مجدد، همان دنباله از نقاط متمایز را جستجو می‌کند که یک الگوریتم P'، دارای عدم بازدید مجدد (هنگامی که نقاط بازدید شده در دنباله بیرون کشیده شده‌اند)، جستجو می‌کند. لذا با توجه به این که P' دارای همان کارایی ولی با ارزیابی تابع کمتر نسبت به الگوریتم P است . می‌توان نتیجه گرفت که P' نسبت به P برتر است. پس سودمند است که همیشه بازدید مجدد را حذف کنیم.

در این مقاله ما یک الگوریتم ژنتیک چند هدفه ارائه می‌کنیم که بازدید مجدد را حذف می‌کند. و به منظور حذف کامل بازدید مجدد از یک درخت BSP[۸] به عنوان آرشیو استفاده می‌کنیم. الگوریتم پیشنهادی با سرعت و کارایی بیشتری به جواب‌های بهینه/ نزدیک به بهینه همگرا شده و از تنوع[۹] مطلوبی نیز برخوردار است.

در ادامه‌ی مقاله در بخش ۲ الگوریتم ژنتیک چند هدفه با حذف بازدید مجدد ارائه گردیده است. نتایج شبیه‌سازی و ارزیابی نتایج در بخش ۳ آورده شده است. در بخش ۴ نتیجه‌گیری و نهایتاً مراجع در انتهای مقاله آورده شده است.

۲- الگوریتم ژنتیک چندهدفه سریع با حذف بازدید مجدد

۲-۱ تعریف مفاهیم

در ابتدا تعدادی از مفاهیم را که در بهینه سازی چند هدفه مطرح است، معرفی می‌کنیم.

مساله بهینه‌سازی چند هدفه: یک بردار تصمیم n بعدی، $x=\{x_1,\ldots,x_n\}$، در فضای پاسخِ X داده شده است. یافتن یک بردار x^*ی که تعداد m تا از توابع هدف داده شده، $F(x^*)=\{f_1(x^*),\ldots,f_m(x^*)\}$، را بیشینه کند. معمولاً فضای پاسخِ X توسط یک مجموعه از قیود، $g_j(x^*)=p_j$ برای $j=1,2,\ldots,k$، محدود شده‌است.

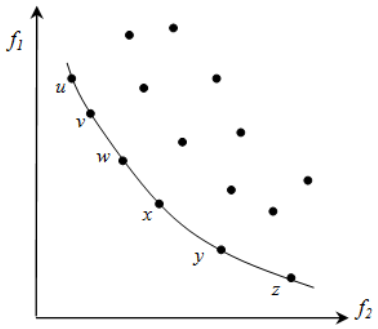
غالب بودن[۱۰]: گفته می‌شود یک بردار مانند $\mathbf{u}=(u_1,\ldots,u_n)$ بر بردار مانند $\mathbf{v}=(v_1,\ldots v_n)$ غالب است ($\mathbf{u}\prec\mathbf{v}$) اگر و تنها اگر $\mathbf{u}$ به صورت جزئی کمتر از $\mathbf{v}$ باشد. یعنی برای هر i که $i=1,\ldots,n$ است $u_i \preceq v_i$ باشد و وجود داشته باشد iی، $i=1,\ldots,n$ ، که برای آن $u_i < v_i$ است.

پاسخ بهینه پارتو: یک پاسخ x_u بهینه پارتو است اگر و تنها اگر هیچ x_v نباشد که برای آن $F(x_v)=\mathbf{v}=(v_1,v_2,\ldots,v_n)$ بر $F(x_u)=\mathbf{u}=(u_1,u_2,\ldots,u_n)$ غالب باشد.

مجموعه بهینه پارتو و جبهه جلویی[۱۱]: به یک مجموعه غیر غالب در فضای پاسخِ X، مجموعه بهینه پارتو می‌گوییم و با X_p نشان می‌دهیم به طوری‌که

$$X_p=\{\vec{x} \mid X_p\subseteq X \text{ و } \vec{x} \text{ در } X \text{ غیر غالب است}\}$$

مقادیر توابع هدف مطابق با X_p در فضای پاسخ، جبهه جلویی نامیده می‌شود و با Y_p نشان داده می‌شود و به صورت زیر نمایش داده می‌شود (شکل۱). $Y_p=F(X_p)=\{f(\vec{x}) \mid \vec{x}\in X_p\}$



شکل۱. نمایش مجموعه نقاط جبهه جلویی

بازدید مجدد: ارزیابی تابع برازش برای i امین عنصر x_i در دنباله $(x_1, x_2, \ldots)$ بازدید مجدد است اگر x_jی وجود داشته باشد که $j<i$ و $x_i=x_j$ است.

فاصله جمعیتی[۱۲]: برای اینکه یک تخمینی از چگالی پاسخ‌ها در اطراف یک پاسخ خاص در جمعیت داشته باشیم. میانگین فاصله دو نقطه در دو سمت این نقطه خاص را به ازای توابع هدف مختلف محاسبه می‌کنیم. چگونگی این تخصیص در الگوریتم ۱ آورده شده است. f_y^{max} و f_y^{min} به ترتیب بیشینه و کمینه مقادیری است که تابع هدف y می‌تواند

[6] Heating, Ventilating, Air Conditioning
[7] No Free Lunch
[8] Binary Space Partitioning
[9] Diversity
[10] Dominance
[11] Front
[12] Crowding Distance

داشته باشد. T[i].y نیز مقدارِ تابع هدفِ y به ازای iامین عضو مجموعه‌ی T است.

الگوریتم۱: تخصیص فاصله جمعیتی به مجموعه T

۱. تعداد اعضای مجموعه T را در m قرار بده
۲. برای هر i : =0: $T[i]_{\text{فاصله}}$
۳. برای هر تابع هدف y اعمال زیر را انجام بده
۳. ۱. T را بر اساس مقادیر تابع y به صورت صعودی مرتب کن
۳. ۲. $T[m]_{\text{فاصله}} := \infty$ و $T[1]_{\text{فاصله}} := \infty$
۳. ۳. برای i از ۲ الی m-1 دستور زیر را انجام بده
۳. ۳. ۱. $T[i]_{\text{فاصله}} := T[i]_{\text{فاصله}} + (T[i+1].y-T[i-1].y)/(f_y^{max} - f_y^{min})$

عملگر مقایسه‌گر جمعیتی: از نظر این عملگر بین دو عضو از اعضای یک جمعیت، آن عضوی انتخاب می‌شود که مقدار رتبه‌ی غیرغالب (الگوریتم ۳) آن کمتر است و در صورتی که این عدد به ازای دو عضو برابر بود. آن عضوی انتخاب می‌شود که مقدار فاصله جمعیتی آن بزرگتر باشد. از این عملگر در فرآیند انتخاب در الگوریتم ژنتیک استفاده می‌شود.

۲-۲ حذف بازدید مجدد پاسخ‌ها

در الگوریتم پیشنهادی، ابتدا یک جمعیت اولیه P0 به صورت تصادفی ایجاد می‌شود. برای اجتناب از بازدید مجدد عناصر (اعضاء)، از یک درخت BSP به عنوان آرشیوی برای نگه‌داری اعضای جمعیت تولید شده، استفاده می‌شود. در واقع درخت BSP به عنوان یک ساختار داده‌ای مناسب برای اعمال پرس‌وجوی سریع مورد استفاده قرار می‌گیرد، پرس و جویی مبنی بر اینکه آیا عضو تولید شده قبلاً بازدید شده است یا خیر؟

الگوریتمی که برای ایجاد چنین آرشیوی استفاده می‌شود [۷] در الگوریتم۲ آورده شده است. در این الگوریتم کل فضای جستجو طبق یک وضوح[13] (دقت) خاصی مورد بررسی قرار می‌گیرد و در ابتدا کل فضا به عنوان فضای گره ریشه قرار داده می‌شود. پس از آن با ایجاد اعضای جمعیت و تولید نودهای فرزند فضای جستجو تقسیم می‌شود به طوری‌که اجتماع زیربازه فرزندان یک گره، برابر با فضای والد خواهد شد. هرگاه یک عضو از جمعیت تولید می‌گردد برای اینکه در درخت قرار گیرد یک بررسی انجام می‌شود که آیا عضو جدید تکراری است یا نه؟ در صورت تکراری بودن، در زیربازه‌ی گره‌ی مورد نظر، که جستجو به آن رسیده است، عملگر جهش اعمال می‌شود تا یک عضو غیرتکراری در زیربازه مورد نظر تولید شود (جهش وفقی[14]).

[13] Resolution
[14] Adaptive Mutation

الگوریتم ۲. الگوریتم حذف بازدید مجدد

۱. ایجاد گره ریشه (root)
۲. RF:= 0 (RF نشانه بازدید مجدد است)
۳. Flag(root):="باز"
۴. خلق عنصر جدید z، توسط الگوریتم ژنتیک
۵. C_node:="گره ریشه" (C_node نود جاری است)
۶. اگر Flag(C_node)= = "باز" آنگاه:
۶. ۱. اگر C_node دو فرزند مانند a وb داشت آنگاه:
۶. ۱. ۱. اگر (z= =a) یا (z= =b) آنگاه: RF:= ۱
۶. ۱. ۲. (بُعدی که a وb در آن بیشترین فاصله را باهم دارند) j:=
۶. ۱. ۳. اگر z در بعد j به a نزدیک‌تر بود آنگاه: C_node:= a
۶. ۱. ۴. در غیر این‌صورت: C_node:= b
۶. ۱. ۵. برو به مرحله شماره ۶.
۶. ۲. در غیر این‌صورت (اگر Flag(C_node)= ="بسته")
۶. ۲. ۱. اگر RF= = 0 آنگاه:
۶. ۲. ۱. ۱. Z را در یک نود فرزندی به C_node اضافه کن
۶. ۲. ۱. ۲. اگر زیر بازه نود فرزند یگانه نباشد آنگاه:
۶. ۲. ۱. ۲. ۱. Flag(Child_node):= "باز"
۶. ۲. ۱. ۳. در غیر این صورت: Flag(Child_node):= "بسته"
۶. ۲. ۱. ۴. پایان.
۶. ۲. ۲. در غیر این‌صورت (اگر RF≠0):
۶. ۲. ۲. ۱. توسط عمل جهش، به طور تصادفی در زیربازه بازدید نشده عضو z را تولید و در نود فرزند اضافه کن.
۶. ۲. ۲. ۲. اگر زیربازه نود فرزند یگانه است آنگاه:
۶. ۲. ۲. ۲. ۱. Flag(Child_node):="بسته"
۶. ۲. ۲. ۳. در غیر این‌صورت : Flag(Child_node):="باز"
۶. ۲. ۲. ۴. پایان.
۷. در غیر این‌صورت (اگر Flag(C_node)= ="بسته"):
۷. ۱. C_node:= "گره والد"
۷. ۲. اگر دو نود فرزند نود جاری "بسته" هستند آنگاه:
۷. ۲. ۱. Flag(C_node):="بسته"
۷. ۲. ۲. زیر درخت تحت نود جاری را از درخت هرس کن
۷. ۲. ۳. برو به مرحله شماره ۶.
۷. ۳. در غیر این‌صورت : نود فرزند باز C_node:=
۷. ۴. برو به مرحله شماره ۶

در حین جستجو، اگر به نودی برسیم که یگانه است، نود یگانه[15]نودی است که در زیربازه‌ی آن (با توجه به وضوح تعریف شد) عضو دیگری نتوان انتخاب کرد، الگوریتم به یک سطح بالاتر (گره والد)

[15] Singleton

برمی‌گردد و مسیر فرزند دیگر را انتخاب می‌کند. در صورتی که گره فرزند دیگر نیز یگانه باشد آنگاه زیر درخت مورد نظر از کل درخت هرس می‌شود. زیرا مقادیری که در زیر درخت وجود دارند مقادیری هستند که بازدید شده‌اند. و به این ترتیب با هرس نمودن زیردرخت از افزایش بیش از اندازه درخت جلوگیری می‌شود.

الگوریتم ۳. الگوریتم مرتب‌سازی غیر غالب برای جمعیت P

۱. برای هر $p \in P$ اعمال زیر را انجام بده
۱. ۱. $S_p := \emptyset$; $n_p := 0$;
۱. ۲. برای هر $q \in P$ اعمال زیر را انجام بده
۱. ۲. ۱. اگر $(p \prec q)$ آنگاه : $S_p := S_p \cup \{q\}$
۱. ۲. ۲. در غیر این‌صورت اگر $(q \prec p)$ آنگاه: $n_p := n_p + 1$
۱. ۳. اگر $(n_p = = 0)$ آنگاه : $F_1 := F_1 \cup \{p\}$; $P_{rank} := 1$;
۲. $i := 1$
۳. مادامی‌که ($F_i \neq \emptyset$) اعمال زیر را انجام بده
۳. ۱. $H := \emptyset$
۳. ۲. برای هر $p \in F_i$ اعمال زیر را انجام بده
۳. ۲. ۱. برای هر $q \in S_p$ اعمال زیر را انجام بده
۳. ۲. ۱. ۱. $n_q := n_q - 1$
۳. ۲. ۱. ۲. اگر $n_q = = 0$ آنگاه: $H = H \cup \{q\}$; $q_{rank} := i+1$;
۳. ۳. $i := i+1$
۳. ۴. $F_i := H$

۳-۲ الگوریتم ژنتیک چند هدفه بدون بازدید مجدد

پس از بررسی جمعیت اولیه P0، و حذف بازدید مجدد، جمعیت بر اساس روش مرتب سازی غیر غالب[۱] (الگوریتم ۳) مرتب می‌شود. به هر عضو از جمعیت یک مقدار برازش (رتبه) که برابر با سطح غیر غالب بودن اوست تخصیص داده می‌شود. رتبه یک برای بهترین سطح، رتبه دو برای دومین بهترین سطح و الی آخر.

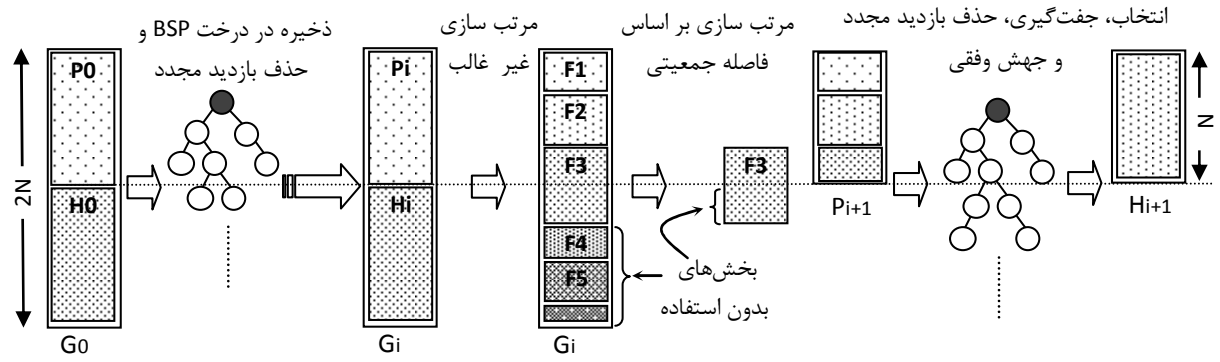
فرض کنید که هدف از بهینه سازی، حداقل کردن توابع هدف باشد. توسط عملگرهای انتخاب و جفت‌گیری، حذف بازدید مجدد و جهش وفقی نسل جدید فرزندان، با نام H_0، با اندازه N، از جمعیت اولیه P0 ایجاد می‌شود. از این‌رو نخبه‌گرایی به واسطه مقایسه نسل جاری با بهترین اعضاء غیرغالبِ یافت شدهِ از نسل قبل، اعمال می‌شود. سپس نسل جدید ایجاد شده با نسل والد ادغام و نسلی جدید با اندازه 2N به نام G0 تولید می‌گردد.

آنگاه G0 بر اساس روش مرتب سازی غیر غالب مرتب می‌شود و به صورت مجموعه‌هایی،که اعضای هر مجموعه دارای یک مقدار رتبه هستند، در F قرار می‌گیرد. ادامه مراحل کار برای نسل‌های بعدی اندکی متفاوت‌تر خواهد بود، که مراحل آن درالگوریتم۴ آمده است.

الگوریتم۴. الگوریتم ژنتیک چندهدفه با حذف بازدید مجدد

۱. $G_i := H_i \cup P_i$
۲. G_i را به صورت غیر غالب مرتب کن و در F قرار بده ($F=\{F_1, F_2, \ldots\}$)
۳. $k:=1$; $P_{i+1}:=\emptyset$;
۴. مادامی‌که $|P_{i+1} \cup F_k| \leq N$ مراحل زیر را انجام بده
۴. ۱. فاصله جمعیتی در F_k را محاسبه کن
۴. ۲. $P_{i+1} := P_{i+1} \cup F_K$
۴. ۳. $k := k+1$
۵. اگر $|P_{i+1}| < N$ آنگاه:
۵. ۱. F_k را بر اساس عملگر مقایسه‌گر جمعیتی مرتب کن
۵. ۲. $x := N - |P_{i+1}|$
۵. ۳. $P_{i+1} := P_{i+1} \cup F_K[1:x]$
۶. یک جمعیت جدید با اعمال جفت‌گیری، حذف بازدید مجدد و جهش وفقی از P_{i+1} ایجاد کن
۷. جمعیت جدید را که در درخت BSP نیز قرار گرفته در H_{i+1} قرار بده
۸. $i := i+1$ و برو به مرحله شماره ۱

برای ایجاد نسل بعدی، اعضایی که به مجموعه غیرغالب F_1، بهترین اعضا در دو جمعیت ادغام شده، تعلق دارند انتخاب می‌شوند. اگر اندازه F_1 از N، اندازه تعیین شده برای اعضای یک نسل، کمتر باشد آنگاه همه اعضای مجموعه F_1 برای رفتن به نسل جدید (P_{i+1}) انتخاب می‌شوند. مابقی اعضای نسل جدید نیز از مجموعه‌های غیر غالب بعدی به ترتیب رتبه آنها انتخاب می‌شود. یعنی اعضای مجموعه F_2 انتخاب بعدی خواهد بود و پس از آن مجموعه F_3 و قس علی هذا. این فرآیند ادامه پیدا می‌کند تا اینکه یا N عضو نسل جدید انتخاب گردد و یا اینکه مجموعه‌ی F_k دیگری را نتوان به صورت کامل به نسل جدید منتقل کرد (یعنی $|P_{i+1} \cup F_k| > N$ و $|P_{i+1}| < N$). اگر حالت دوم اتفاق افتد یعنی نسل جدید اعضای آن پر نشود آنگاه مجموعه غیرغالب بعدی انتخاب شده و اعضای آن توسط عملگر مقایسه‌گر جمعیتی مرتب می‌شود. سپس مابقی اعضای مورد نیاز برای نسل بعدی از ابتدای مجموعه غیرغالبِ مرتب شده انتخاب می‌شود. پس از ایجاد نسل جدید P_{i+1} ، با اعمال عملگرهای انتخاب، جفت‌گیری، حذف بازدید مجدد و جهش وفقی نسل H_{i+1} ایجاد می‌شود (شکل ۲).



شکل۲. فرآیند الگوریتم ژنتیک پیشنهادی

۳- نتایج شبیه سازی و ارزیابی

در این بخش ابتدا تعدادی از مسائل تست را که برای مقایسه روش پیشنهادی با الگوریتم NSGA-II مورد استفاده قرار گرفته است توضیح می‌دهیم. سپس معیارهای کارایی که برای مقایسه روش پیشنهادی با روش NSGA-II استفاده شده است، معرفی می‌گردد. نتایج به دست آمده از شبیه سازی نیز در ادامه آورده شده است.

۳-۱ مساله‌های تست

مساله‌های تستی که در این مقاله استفاده شده است از تعدادی از مقالات شاخص قبلی که در مقوله الگوریتم‌های تکاملی چند هدفه نوشته شده‌اند [۱۴،۱۵،۱۶،۱۷،۱۸و۱۹] انتخاب شده‌است. همه مساله‌های تست انتخاب شده دارای دو تابع هدف هستند. این مسائل در جدول ۱ آورده شده‌است. ستوت n در جدول نمایانگر بعد توابع است.

۳-۲ معیارهای کارایی

برخلاف روش‌های بهینه‌سازی تک هدفه (مبتنی بر یک تابع هدف)، در بهینه سازی چند هدفه دو مقصود وجود دارد. الف) همگرایی به مجموعه بهینه پارتو و ب) نگهداشت تنوع در پاسخ‌های مجموعه بهینه پارتو[۱]. این دو عامل را نمی‌توان با یک معیار کارایی، به طور مناسب اندازه‌گیری کرد.

متریک‌های کارایی متفاوتی در [۱۱،۱۲و۱۳] پیشنهاده شده است. در این مقاله ما از دو متریک که برای ارزیابی مقصودهای بالا مناسب است استفاده می‌کنیم.

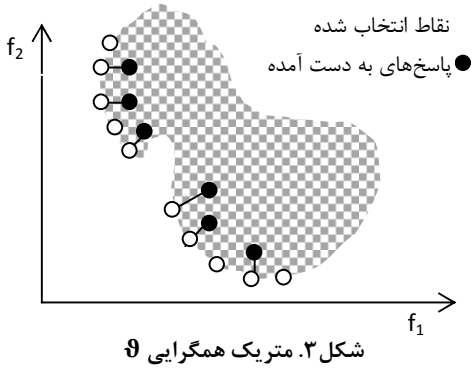
متریک اول، میزان همگرایی به یک مجموعه پاسخِ بهینه‌ی پارتوی مشخص را اندازه‌گیری می‌کند و با ϑ نشان داده می‌شود. پس الگوریتم‌های

جدول۱. مسائل تست استفاده شده در این مقاله

شماره مساله	n	محدوده متغیرها	توابع هدف
(۱)	۳	[-۴ ,۴]	$f_1(x) = 1 - \exp(-\sum_{i=1}^{3}(x_i - \frac{1}{\sqrt{3}})^2)$ $f_2(x) = 1 - \exp(-\sum_{i=1}^{3}(x_i + \frac{1}{\sqrt{3}})^2)$
(۲)	۲	$[-\pi, \pi]$	$f_1(x)=[1+(A_1-B_1)^2+(A_2-B_2)^2]$ $f_2(x)=[(x_1+3)^2+(x_2+1)^2]$ $A_1=0.5\sin 1-2\cos 1+\sin 2-1.5\cos 2$ $A_2=1.5\sin 1-\cos 1+2\sin 2-0.5\cos 2$ $B_1=0.5\sin x_1-2\cos x_1+\sin x_2-1.5\cos x_2$ $B_2=1.5\sin x_1-\cos x_1+2\sin x_2-0.5\cos x_2$
(۳)	۳	[-۵ , ۵]	$f_1(x) = \sum_{i=1}^{n-1}(-10\exp(-0.2\sqrt{x_i^2+x_{i+1}^2}))$ $f_2(x) = \sum_{i=1}^{n}(\lvert x_i\rvert^{0.8} + 5\sin x_i^3)$
(۴)	۳۰	[۰ ,۱]	$f_1(x)=x_1$ $f_2(x)=g(x)\,[1-\sqrt{x_1/g(x)}]$ $g(x)=1+9(\sum_{i=2}^{n} x_i)/(n-1)$
(۵)	۳۰	[۰ ,۱]	$f_1(x)=x_1$ $f_2(x)=g(x)\,[1-(x_1/g(x))^2]$ $g(x)=1+9(\sum_{i=2}^{n} x_i)/(n-1)$

چند هدفه‌ای که روی مسائلی با پاسخ بهینه پارتو معین تست شوند، برایشان این متریک قابل اندازه‌گیری است. لذا، چنین متریکی را نمی‌توان برای هر مساله دلخواهی به کار برد.

برای استفاده از متریک همگرایی ما ابتدا یک مجموعه (M) از پاسخ‌ها را از بین پاسخ‌های بهینه پارتو جبهه جلویی که دارای فاصله یکنواختی هستند انتخاب می‌کنیم. آنگاه برای هر پاسخی که توسط الگوریتم به دست می‌آید حداقل فاصله اقلیدسی آن را با اعضای مجموعه M محاسبه می‌کنیم. میانگین این فاصله‌ها متریک اول ϑ (متریک همگرایی)، را به دست می‌دهد (شکل ۳).

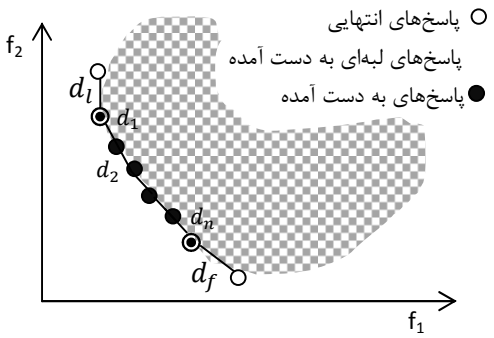


شکل۳. متریک همگرایی ϑ

متریک دوم Δ، برای اندازه‌گیری گسترشِ جواب‌های به دست آمده از الگوریتم است. در واقع ما به دنبال مجموعه پاسخ‌هایی هستیم که کل ناحیه بهینه پارتو را پوشش دهد. برای این منظور ابتدا فاصله اقلیدسی d_i بین پاسخ‌های متوالی در مجموعه پاسخ‌های غیرغالبِ بدست آمده را محاسبه می‌کنیم. آنگاه میانگین این فاصله‌ها $\bar{d}$ را به دست می‌آوریم. پس از آن با محاسبه پاسخ‌های انتهایی و اندازه‌گیری فاصله آنها از پاسخ‌های لبه‌ای در مجموعه‌ی پاسخ‌های به دست آمده، d_f و d_l، توسط رابطه ۱ غیریکنواختی در توزیع پاسخ‌ها رامحاسبه می‌کنیم.

$$\Delta = \frac{d_l + d_f + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_l + d_f + (N-1)\bar{d}} \qquad (۱)$$

به طوری‌که N تعداد پاسخ‌های پارتوی یافته شده است. پارامترهای استفاده شده در رابطه (۱) در شکل ۴ نمایش داده شده است. یک توزیع مناسب باعث خواهد شد که همه d_i ها برابر $\bar{d}$ و $d_l = d_f = 0$ شود. و این شرایط باعث خواهد شد که Δ برابر صفر شود.



شکل۴. متریک تنوع Δ

۳-۳ ارزیابی نتایج

در شبیه‌سازی‌های انجام شده تعداد اعضای یک نسل N= ۱۰۰ و تعداد کل ارزیابی توابع به اندازه ثابت ۴۰۱۰۰ [۷] (به عنوان معیار خاتمه الگوریتم) در نظر گرفته شده است. جدول ۲ میانگین و واریانس متریک همگرایی ϑ به دست آمده از دو الگوریتم NSGA-II و الگوریتم پیشنهادی را نشان می‌دهد. نتایج به دست آمده نشان می‌دهد که الگوریتم پیشنهادی نسبت به الگوریتم NSGA-II بهتر همگرا می‌شود.

جدول۲ . میانگین و واریانس متریک همگرایی ϑ

الگوریتم / مساله تست		(۱)	(۲)	(۳)	(۴)	(۵)
NSGA-II	میانگین	۰/۰۰۱۹	۰/۰۱۵۵	۰/۰۲۸۹	۰/۰۳۳۴	۰/۰۷۲۳
	واریانس	۰	۰/۰۰۰۰۰۱	۰/۰۰۰۰۱۸	۰/۰۰۴۷۵	۰/۰۳۱۶
الگوریتم پیشنهادی	میانگین	۰/۰۰۱۱	۰/۰۰۹۸	۰/۰۲۶۳	۰/۲۹۱	۰/۰۶۰۹
	واریانس	۰	۰	۰/۰۰۰۰۰۳	۰/۰۰۱۲	۰/۰۲۲۱

همچنین میزان واریانس نیز در ده بار اجرا که محاسبه شده است مقدار کمی را نشان داده است.

جدول ۳ واریانس و میانگین متریک تنوع Δ را برای اجراهای متعدد دو الگوریتم روی مسائل تست مختلف نشان می‌دهد. همان‌طور که از نتایج به دست آمده معلوم است الگوریتم پیشنهادی از نظر متریک تنوع پاسخ‌ها نیز کارایی بهتری داشته است.

جدول ۳ . میانگین و واریانس متریک تنوع Δ

الگوریتم / مساله تست		(۱)	(۲)	(۳)	(۴)	(۵)
NSGA-II	میانگین	۰/۳۹۹	۰/۵۶۲	۰/۶۰۳	۰/۴۰۱	۰/۵۱۹
	واریانس	۰/۰۰۰۶۳	۰/۰۰۲۸۶	۰/۰۰۰۹۹	۰/۰۰۱۸	۰/۰۰۴۷
الگوریتم پیشنهادی	میانگین	۰/۳۷۸	۰/۴۵۲	۰/۴۱۱	۰/۳۹۰	۰/۴۳۰
	واریانس	۰/۰۰۰۵۸	۰/۰۰۲۷	۰/۰۰۱۹	۰/۰۰۰۲۳	۰/۰۰۵۰

۴- نتیجه‌گیری

در این مقاله یک الگوریتم ژنتیک چند هدفه با حذف بازدید مجدد ارائه شده است. در الگوریتم پیشنهادی برای حذف بازید مجدد پاسخ‌ها و افزایش کارایی الگوریتم در رسیدن به جواب‌های بهینه از درخت BSP استفاده شده است. جهت بررسی و مقایسه کارایی، الگوریتم پیشنهادی و الگوریتم چند هدفه NSGA-II روی چند مساله تست استاندارد اعمال گردید. و طبق دو معیار تنوع و میزان همگرایی با هم مقایسه شدند. مشاهده شده است که الگوریتم پیشنهادی دارای همگرای بهتری در رسیدن به پاسخ‌های بهینه پارتو است. همچنین الگوریتم پیشنهادی در مقایسه با الگوریتم NSGA-II تنوع بهتری در پاسخ‌ها از خود نشان داده است.

۵- سپاسگزاری

این اثر با استفاده از اعتبارات پژوهشی دانشگاه آزاد اسلامی به انجام رسیده است.

مراجع

1. Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan , "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II", IEEE transactions on evolutionary computation, vol. 6, no. 2, april 2002.
2. Mitra, K., Deb, K., and Gupta, S. K. (1998). Multiobjective dynamic optimization of an industrial Nylon 6 semibatch reactor using genetic algorithms. Journal of Applied Polymer Science, 69(1), 69–87. 8. Rudolph, G. (1999) Evolutionary.
3. Horn, J. and Nafploitis, N., and Goldberg, D. E. (1994) A niched Pareto genetic algorithm for multi-objective optimization. In Michalewicz, Z., editor, Proceedings of the First IEEE Conference on Evolutionary Computation, pages 82–87, IEEE Service Center, Piscataway, New Jersey.
4. Srinivas, N. and Deb, K. (1995) Multi-Objective function optimization using non-dominated sorting genetic algorithms, Evolutionary Computation, 2(3):221–248.
5. Zitzler, E. and Thiele, L. 'Multiobjective optimization using evolutionary algorithms—A comparative case study". In Eiben, A. E., B¨ack, T., Schoenauer, M., and Schwefel, H.-P., editors, Parallel Problem Solving from Nature, V, pages 292–301, (1998), Springer, Berlin, Germany.
6. F. Glover and M. Laguna, Tabu search. Norwell, MA: Kluwer, 1997.
7. Shiu Yin Yuen, Chi Kin Chow, "A Genetic Algorithm That Adaptively Mutates and Never Revisits", IEEE transactions on evolutionary computation, vol. 13, no. 2, april 2009.
8. D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," IEEE Trans. Evol. Comput., vol. 1, no. 1, pp. 67–82,Apr.1997.
9. C. K. Chow, H. T. Tsui, and T. Lee, "Surface registration using a dynamic genetic algorithm," Pattern Recognition, vol. 37, no. 1, pp.105–117, 2004.
10. K. F. Fong, V. I. Hanby, and T. T. Chow, "HVAC system optimization for energy management by evolutionary programming," Energy and Buildings, vol. 38, no. 3, pp. 220–231, 2006
11. K. Deb, "Multiobjective Optimization Using Evolutionary Algorithms". Chichester, U.K.: Wiley, 2001.
12. ---, "On the performance assessment and comparison of stochastic multiobjective optimizers," in Parallel Problem Solving from Nature IV, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds. Berlin, Germany: Springer-Verlag, 1996, pp. 584–593
13. E. Zitzler, "Evolutionary algorithms for multiobjective optimization: Methods and applications," Doctoral dissertation ETH 13398, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 1999
14. ---- , "Multiobjective optimization and multiple constraint handling with evolutionary algorithms—Part II: Application example," IEEE Trans. Syst., Man, Cybern. A, vol. 28, pp. 38–47, Jan. 1998
15. F. Kursawe, "A variant of evolution strategies for vector optimization," in Parallel Problem Solving from Nature, H.-P. Schwefel and R. Männer, Eds. Berlin, Germany: Springer-Verlag, 1990, pp. 193–197.
16. C. Poloni, "Hybrid GA for multiobjective aerodynamic shape optimization," in Genetic Algorithms in Engineering and Computer Science, G. Winter, J. Periaux, M. Galan, and P. Cuesta, Eds. New York: Wiley, 1997, pp. 397–414
17. J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in Proceedings of the First International Conference on Genetic Algorithms, J. J. Grefensttete, Ed. Hillsdale, NJ: Lawrence Erlbaum, 1987, pp. 93–100.
18. D.Van Veldhuizen, "Multiobjective evolutionary algorithms: Classifications, analyzes, and new innovations," Air Force Inst. Technol., Dayton, OH, Tech. Rep. AFIT/DS/ENG/99-01, 1999
19. E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," Evol. Comput., vol. 8, no. 2, pp. 173–195, Summer 2000.