علوم و مهندسی کامپیوتر
نشریه علمی پژوهشی انجمن کامپیوتر ایران
مجلد ۵ شماره ۱ (الف)، ۱۳۸۶
صفحات ۴۱-۴۹
مقاله کوتاه

بررسی استفاده از وزن ثابت در روش جستجوی *WA

سید علی اکرمی‌فر　　غلامرضا قاسم‌ثانی

دانشکدهٔ مهندسی کامپیوتر، دانشگاه صنعتی شریف، تهران، ایران

چکیده

روش جستجوی *WA که یک روش جستجوی مبتنی بر *A می‌باشد، کاربردهای مختلفی در هوش مصنوعی پیدا کرده است. این روش یک روش نیمه بهینه است که در آن با انتخاب وزن مناسب، کارآیی جستجو به میزان قابل توجهی افزایش می‌یابد. بسیاری از تحقیقات گذشته که از این روش جستجو استفاده کرده‌اند، هر یک وزنی ثابت را برای جستجو در همه دامنه‌ها و همهٔ مسایل خود در نظر گرفته‌اند، در حالیکه شرایط و ویژگی‌های دامنه‌ها و مسایل با یکدیگر تفاوت دارد. در این مقاله با بررسی روش *WA نشان داده‌ایم که استفاده از یک وزن ثابت برای همهٔ شرایط مختلف شامل دامنه، مساله و تابع ابتکاری متفاوت، پذیرفته نیست. همچنین ضمن بررسی و توجیه علت کارایی بهتر *WA نسبت به *A، با استفاده از مثال‌ها و قضیه‌هایی، دامنهٔ تغییرات وزن و تاثیر آن بر عملکرد جستجو را بررسی نموده و روش جدید جستجوی *GBFS را به عنوان حد بالای روش جستجوی *WA تعریف کرده‌ایم.

کلمات کلیدی: هوش مصنوعی، جستجوی ابتکاری، روش جستجوی*A، روش جستجوی *WA.

۱- مقدمه

روش‌های جستجوی ابتکاری[۱]، که از یک تابع ابتکاری[۲] $h(S)$ برای تخمین فاصلهٔ هر وضعیت S از فضای جستجو تا وضعیت هدف بهره می‌گیرند در حل مسایل مختلف خصوصا دامنه‌های پیچیده و مشکل به صورت گسترده استفاده می‌شوند. یکی از مهمترین روش‌های جستوی ابتکاری *A نام دارد. این روش جستجوی پرکاربرد در هوش مصنوعی، ترکیبی از دو روش جستجوی حریصانه بهترین اول[۳] و جستجوی سطح اول[۴] می‌باشد[۱، ۲] و محاسن هر دو روش را در بر دارد. روش جستجوی حریصانه بهترین اول با انتخاب یک وضعیتی بعدی که منجر به کاهش تخمین فاصله تا هدف یا $h(S)$ می‌گردد، به جستجوی هدف می‌پردازد. علی‌رغم کاهش قابل توجه هزینهٔ جستجو، این روش یک روش جستجوی کامل و بهینه نیست. در مقابل روش جستجوی سطح اول وضعیت‌های بعدی را که فاصلهٔ $g(S)$ کمتری تا وضعیت اولیه دارند زودتر مورد بررسی قرار می‌دهد. اگرچه این روش بهینه و کامل است اما روشی ناکارامد و زمانبر بوده و حافظهٔ زیادی مصرف می‌کند.

روش جستجوی *A با کمینه کردن تابع تخمین طول پاسخ جستجو($f(S)$)، کوتاهترین مسیر را برای رسیدن از وضعیت اولیه به یک وضعیت هدف می‌یابد. تابع تخمین *A شبیه تابع تخمین روش جستجوی حریصانه بهترین اول است، با این تفاوت که روش حریصانه بهترین اول وضعیتی که کمترین تخمین $f(S)=h(S)$ را دارد زودتر گسترش می‌دهد. ولی *A با در نظر گرفتن فاصله وضعیت اولیه تا وضعیت جاری ($g(S)$)، وضعیتی را که $f(S)=g(S)+h(S)$ کمتری دارد انتخاب و گسترش می‌دهد. در مقایسه با روش‌های کلاسیک جستجو همچون سطح اول و یا عمق اول[۵]، روش *A در صورت انتخاب یک تابع ابتکاری مطلع[۶] و قابل قبول[۷]، با بررسی تعداد کمتری از وضعیت‌ها پاسخ بهینه را می‌یابد و در نتیجه سریعتر عمل می‌کند.

تاکنون روش‌های متعددی مبتنی بر *A طراحی شده‌اند. برای مثال می‌توان به روش‌های *IDA[۳]، *SMA[۴]، *Any-Time-A[۵]، Adaptive-*A[۶]، *Incremental-A[۷]، *LPA[۷]، *Dynamic-A(D*)[۸]، *Alpha-A[۹]، *LRT-A[۱۰]، *FSA[۱۱] و *WA[۱۲، ۱۳، ۱۴، ۱۵] اشاره نمود. این روش‌ها به صورت گسترده‌ای در کاربردهای رایانه و هوش مصنوعی مورد استفاده قرار گرفته‌اند. علت کاربرد گستردهٔ این روش‌ها، یافتن پاسخ مناسب در زمان مناسب است.

تمرکز این مقاله بر روش *WA است. روش *WA برای اولین بار در [۱۵] مطرح گردید. در این روش برای محاسبهٔ تابع f نسبت‌های غیریکسانی از g و h استفاده می شود. در روش *WA از یک عدد $W\geq 1$ به عنوان ضریب یا وزن

تخمین فاصله h استفاده می‌شود و تابع تخمین به صورت $f(s)=g(S)+W\times h(S)$ در نظر گرفته می‌شود[۱۶، ۱۷، ۱۸].

مهمترین بخش در WA* انتخاب وزن مناسب است. وزن نامناسب ممکن است تاثیر منفی بر کارایی جستجو یا کیفیت پاسخ داشته باشد. به عبارت دیگر وزن نامناسب می‌تواند زمان جستجو را افزایش دهد و از مسیری طولانی‌تر به جواب برسد. سیستم‌هایی که از WA* استفاده می‌کنند، معمولا وزن ثابتی را در نظر می‌گیرند. برای مثال سیستم HSPr که یک سیستم برنامه ریزی است از عدد ۵ برای حل همهٔ مسایل استفاده کرده است[۱۹]. همچنین سیستم YAHSP عدد ۳ را مورد استفاده قرار داده است[۲۰]. در آزمایشات ارائه شده در [۲۱] به صورت ثابت عدد ۲ و در [۲۲] نیز از عدد ۳ کمک گرفته است.

مهمترین دلیل انتخاب یک عدد ثابت برای همهٔ مسایل و همهٔ دامنه‌ها مناسب بودن این اعداد در برخی آزمایشات این سیستم‌ها بوده است. اما به صورت مشخص استدلال روشنی برای انتخاب یک عدد ثابت وجود ندارد و معلوم نیست که آیا یک عدد ثابت می‌تواند برای همهٔ دامنه‌ها و هر تابع ابتکاری مناسب باشد؟ همچنین در هیچ یک از این روش‌ها به دلیل ثابت بودن تابع ابتکاری، رابطهٔ تابع ابتکاری با وزن در نظر گرفته نشده است. در حالیکه ممکن است برای شرایط مختلف شامل دامنهٔ متفاوت، مسالهٔمتفاوت و یا تابع ابتکاری متفاوت، عدد مناسب نیز متفاوت باشد. هدف اصلی این مقاله نشان دادن این نکته است که انتخاب یک عدد ثابت برای همهٔ شرایط دامنه، مسایل و توابع ابتکاری متفاوت، صحیح نیست.

ترتیب ارائهٔ مطالب این مقاله به این صورت است که در بخش دوم WA* به صورت کامل معرفی می‌شود و برخی از ابعاد نظری، تعاریف و قضایای مرتبط با آن مطرح می‌گردد. در این بخش تنها به تعریف WA* اکتفا نشده و یافته‌های جدیدی راجع به آن معرفی شده است. در بخش سوم نیز نتایج آزمون تجربی جستجو در مسایل مختلف تشریح می‌گردد. بخش پایانی نیز به جمع‌بندی و نتیجه گیری و بررسی مختصر اهداف مقاله می‌پردازد.

۲- بررسی نظری Weighted A*

۲-۱ معرفی WA*

روش جستجوی WA* از این نظر که با کمینه کردن یک تابع تخمین $f(S)$ پاسخ را می‌یابد، شبیه روش A* است. تفاوت WA* با A* در نحوهٔ تعریف تابع $f(S)$ است. تابع $f(S)$ در A* برای هر وضعیت S، از مجموع فاصلهٔ وضعیت اولیه تا S که با $g(S)$ نمایش داده می‌شود و تخمین فاصله از وضعیت S تا وضعیت هدف که با $h(S)$ نمایش داده می‌شود، به‌دست می‌آید. تعریف نمادین این تابع به صورت $f(S)=g(S)+h(S)$ می‌باشد. تفاوت تابع $f(S)$ مربوط به WA* با تابع $f(S)$ در A* در یک عدد ثابت (وزن) است که به عنوان ضریب $h(S)$ استفاده می‌شود. شکل کلی این تابع به صورت $f(S)=g(S)+W\times h(S)$ می‌باشد. با توجه به تعریف این تابع مشاهده می‌شود که روش A* خود یک روش WA* است که در آن W=1 می‌باشد.

در روش A* برای رسیدن به پاسخ بهینه، تابع h به صورت *قابل قبول* انتخاب می‌گردد. تابعی که تخمین فاصله آن همواره از اندازهٔ واقعی کمتر باشد، یک تابع قابل قبول است. در صورت قابل قبول بودن تابع ابتکاری $h(S)$، روش جستجوی A* پاسخ بهینه را می‌یابد[۱]. از آنجاییکه وزن W در A* به صورت پیش‌فرض برابر ۱ است، لذا برای افزایش سرعت جستجو از وزن‌های بزرگتر از ۱ استفاده می‌شود. افزایش وزن می‌تواند باعث کاهش زمان (و افزایش سرعت جستجو) شود؛ در مقابل ممکن است یک جواب غیر بهینه را پیدا نماید. اگر تابع ابتکاری h یک تابع قابل قبول باشد و با استفاده از ضریب ثابت W تابع $W\times h(S)$ نیز قابل قبول باشد، آنگاه روش WA* نیز جواب بهینه را می‌یابد. اما در عمل ممکن است ضریب W تابع $h(S)$ را غیرقابل قبول کند و در نتیجه جوابی نیمه بهینه به‌دست آید. در این صورت استفاده از آن ضریب مشخص، ناقض قابل قبول بودن خواهد بود. بنابراین مانند سایر مقالات مرتبط[۱۹] در این مقاله مطلع بودن تابع $h(S)$ نسبت به قابل قبول بودن آن ترجیح دارد. لذا از همین ابتدا شرط قابل قبول بودن را از تابع ابتکاری WA* برداشته و تنها در موارد مورد نیاز آن را فرض خواهیم کرد.

۲-۲ بررسی ابعاد WA*

استفاده از وزن W به عنوان ضریب تخمین فاصله h باعث می‌شود که رفتار WA* با A* تفاوت نماید. با انتخاب وزنی مناسب WA* بسیار سریعتر از A* پاسخ را می‌یابد. علت اصلی این مساله اینست که وفاداری WA* به فاصله وضعیت اولیه تا وضعیت جاری $g(S)$ نسبت به مورد مشابه در A* کمتر است. وفاداری روش جستجو به فاصلهٔ وضعیت جاری نسبت به مبدأ باعث می‌شود تا پاسخ بهینه به دست آید. این وفاداری باعث می شود که A* به جای جستجو در عمق، تمایل بیشتری به جستجو در سطح داشته باشد؛ در حالیکه استفاده از وزن تمایل به جستجو در عمق را افزایش می‌دهد. در صورت مطلع بودن تابع ابتکاری پاسخ سریعتر به دست می‌آید. اما این پاسخ ممکن است بهینه نباشد. میزان وفاداری WA* به وزن W بستگی دارد و با افزایش وزن احتمال کاهش کیفیت وجود دارد.

درک عمیقتر رفتار WA* مستلزم بررسی ابعاد آن، خصوصا بررسی اهمیت و جایگاه وزن W در آن می‌باشد. معمولا تخمین فاصله‌های دور در محیط‌های واقعی خطای بسیار بیشتر و حتی غیر قابل مقایسه با تخمین فاصله‌های نزدیک دارد. برای مثال اگر شیئی در فاصلهٔ ۲/۵ کیلومتری باشد، ابزار یا عامل تخمین بسته به خوشبینانه یا بدبینانه عمل کردن ممکن است آن شیئ را در فاصلهٔ ۲ کیلومتری یا ۳ کیلومتری در نظر بگیرد. در حالیکه اگر همین ابزار یا عامل بخواهد فاصلهٔ شیئی را که در فاصلهٔ ۱۰/۵ متری قرار دارد برآورد نماید، احتمالا فاصله را ۱۰ یا ۱۱ متر تخمین خواهد زد. درحالت اول خطای تخمین در حد کیلومتر و در حالت دوم خطای تخمین در حد متر است. این مساله می‌تواند باعث ایجاد اعتماد بیشتر به تخمین‌های نزدیکتر یک تابع تخمین فاصله درمقابل تخمین‌های دورتر همان تابع گردد. با توجه به اینکه الگوریتم WA* وضعیت‌هایی را که تخمین کمتری ارائه می‌کنند، زودتر جستجو می‌کند، بنابر این استفاده از یک ضریب (W) باعث می‌شود که تخمین‌های نزدیکتر و کمتر، اهمیت بیشتری پیدا کنند.

یک علت دیگر استفاده از وزن برای تخمین فاصله می‌تواند این باشد که افزایش وزن W میزان قابل قبول بودن را در برخی از توابع ابتکاری افزایش می‌دهد. طبق تعریف اگر دو تابع ابتکاری h_1 و h_2 هردو قابل قبول باشند، میزان قابل قبول بودن h_2 از h_1 بیشتر است اگر و فقط اگر به ازای هر وضعیت در یک مسالهٔ مشخص تخمین فاصله توسط تابع h_2 بیشتر از h_1 باشد. اگر یک تابع ابتکاری همواره تخمینی کمتر از مقدار واقعی ارائه نماید، میزان تخمین آن برای فواصل دورتر بسیار کمتر است و لذا خطای بیشتری دارد. اگر افزودن یک ضریب W خود باعث غیر قابل قبول بودن نشود، یک تابع ابتکاری جدید ($W\times h(S)$) با مقبولیت بیشتر را ایجاد خواهد نمود. حتی در صورت غیر قابل قبول بودن در برخی از وضعیت‌ها، در بسیاری از موارد دیگر مقبولیت این تابع نسبت به تابع ابتکاری اصلی بیشتر است و بنابر این ممکن است در مقابل از دست دادن مقداری از کیفیت جواب، سرعت جستجو را به اندازهٔ قابل توجهی افزایش دهد.

۲-۳ روش‌های نمایش WA*

ابتدایی‌ترین شکل WA* که از دو وزن مختلف برای ضریب h و g استفاده می‌کرد برای اولین بار در [۱۳] معرفی شد. پس از آن روش‌های بعدی که مبتنی بر یک وزن بودند، به وجود آمدند و رایج شدند. معمولا دو شکل مختلف برای نمایش تابع تخمین در در روش‌های تک وزنی WA* مطرح است. شکل اول نمایش این تابع به صورت $f_A=g+W\times h,\ W\geq 0$ [۲۲،۱۴] و شکل دوم نمایش آن نیز به صورت $f_B=(1-\omega)\times g+\omega\times h,\ 0\leq\omega\leq 1$ می‌باشد[۱۴، ۱۷]. در حالت اول مقدار W می‌تواند بزرگتر از صفر باشد، اما مقادیر مؤثری که برای افزایش کارایی جستجو صورت می‌گیرند، برای مقادیر بزرگتر از یک است. در شکل دوم نمایش تابع f نیز مقادیر موثری که باعث افزایش کارایی می‌شوند، در بازه $\frac{1}{2}\leq\omega\leq 1$ در نظر گرفته می‌شود.

این دو روش از نظر عملکرد معادلند. یعنی به ازای هر ω یک و تنها یک W در بازه‌های تعریف شده وجود دارد که روش WA* برای جستجو و حل هر مسالهٔ مشخص توسط توابع f_A و f_B مسیر یکسانی را پیمایش می‌کند. برای نشان دادن معادل بودن بایستی بازای هر $0\leq\omega<1$ تنها و فقط تنها یک $W\geq 0$ بیابیم که دو روش مشابه هم رفتار نمایند. به عبارت دیگر باید داشته باشیم:

$$\forall S_1,\forall S_2,\exists W\geq 0 \mid f_A(S_1)<f_A(S_2) \Leftrightarrow \exists\omega\in[0,1) \mid f_B(S_1)<f_B(S_2)$$

یا

$$g(S_1)+W\times h(S_1)<g(S_2)+W\times h(S_2)$$
$$\Leftrightarrow (1-\omega)\times g(S_1)+\omega\times h(S_1)<(1-\omega)\times g(S_2)+\omega\times h(S_2)$$

و یا

$$g(S_1)+W\times h(S_1)<g(S_2)+W\times h(S_2)$$
$$\Leftrightarrow g(S_1)+\frac{\omega}{(1-\omega)}\times h(S_1)<g(S_2)+\frac{\omega}{(1-\omega)}\times h(S_2)$$

بنابر این با فرض $0\leq\omega<1$ و $W\geq 0$ داریم[۱۴]:

$$W=\frac{\omega}{(1-\omega)}\quad or\quad \omega=\frac{W}{1+W} \qquad (۱)$$

در این مقاله شکل اول نمایش WA* مورد استفاده قرار گرفته است.

روش WA* در دو حد نهایی خود، قابل تبدیل به روش‌های جستجوی سطح اول و حریصانهٔ بهترین اول می‌باشد. در تابع f_A اگر $W=0$ و همچنین در تابع f_B اگر $\omega=0$ باشد، آنگاه این توابع به صورت $f_A=g,\ f_B=g$ تبدیل می‌شوند و روش جستجو، روش جستجوی سطح اول خواهد بود. همچنین اگر $\omega=1$ و $W\to\infty$ آنگاه $f_B=h$ و $f_A\to h$ خواهد بود[۸]. به عبارت دیگر اگر در f_A وزن W به سمت بینهایت میل کند، روش جستجو به روش جستجوی حریصانهٔ بهترین اول نزدیک می‌شود. برای بررسی این موضوع می‌دانیم که دوتابع $f_A=g+W\times h$ و $f_A'=\frac{g}{W}+h, W>0$ در روش A* هم‌ارزند، زیرا به ازای هر s_1 و s_2 داریم:

$$f_A(S_1)>f_A(S_2)$$
$$\Leftrightarrow g(S_1)+W\times h(S_1)>g(S_2)+W\times h(S_2), W>0$$
$$\Leftrightarrow \frac{g(S_1)}{W}+h(S_1)>\frac{g(S_2)}{W}+h(S_2)$$
$$\Leftrightarrow f_A'(S_1)>f_A'(S_2), W>0$$
$$\Leftrightarrow f_A\equiv f_A'$$

در نتیجه داریم:

$$W\to\infty\Rightarrow\frac{g}{W}\to 0, f_A'\to h, f_A\to h \qquad (۲)$$

این مطالب نشان می‌دهد که روش WA* یک روش جستجو بین روش جستجوی سطح اول و روش جستجوی حریصانهٔ بهترین اول است. تغییر W باعث نزدیکتر یا دورتر شدن روش جستجو به یکی از این دو روش می‌گردد.

۲-۴ معرفی روش GBFS*

در هر مسالهٔ مشخص، برای هر تابع ابتکاری مشخص h می‌توان یک مقدار برای W پیدا نمود که برای اعداد بزرگتر از آن نتیجهٔ جستجو تفاوتی نکند. اگر W به اندازهٔ کافی بزرگ باشد، آنگاه در هر مرحله بین چند وضعیت، آنکه h کمتری دارد انتخاب می‌شود و پارامتر g عملا بی‌تاثیر است. تنها در وضعیت‌هایی که h برابری دارند، وضعیتی که g کمتری دارد انتخاب می‌گردد. این روش جستجو را GBFS* می‌نامیم. تفاوت GBFS* با جستجوی حریصانهٔ بهترین اول (GBFS) در این نکته است که GBFS در صورتی که چند وضعیت h برابر داشته باشند به صورت نامعین[۹] یکی از آن‌ها را انتخاب می‌کند و پارامتر g هیچ تاثیری ندارد؛ در حالیکه انتخاب در GBFS* قطعیت بیشتری دارد و در شرایطی از g استفاده می‌کند. به عبارت دیگر:

if $h(s_1)=h(s_2)$ select $(g(s_1)<g(s_2))?(s_1):(s_2)$
else select $(h(s_1)<h(s_2))?(s_1):(s_2)$

قضیه ۱: اگر به ازای W_i مشخص روش WA* مانند روش GBFS* عمل کند، آنگاه به ازای هر وزن W_j که $W_j>W_i$ نیز مانند روش GBFS* عمل می‌کند.

این که به ازای W_i روش GBFS* باشد، پس به صورت نمادین داریم:

$$f=g+W_i\times h$$
$$\forall S_1,S_2 \mid ((h(S_1)<h(S_2))\ or\ ((h(S_1)=h(S_2))\ and\ (g(S_1)<g(S_2)))\Rightarrow (f(S_1)<f(S_2))$$

حال باید نشان دهیم که با بزرگتر شدن W، رابطهٔ فوق در روش جستجو باز هم برقرار است. چون W_j از W_i بزرگتر است پس داریم: $W_j>W_i\Rightarrow W_j=W_i+\Delta w$. تابع جستجو نیز به صورت $f'=g+W_j\times h$ تعریف می‌شود. حال باید نشان دهیم که:

$$f'=g+W_j\times h, W_j>W_i$$
$$\forall S_1,S_2 \mid ((h(S_1)<h(S_2))\ or\ ((h(S_1)=h(S_2))\ and\ (g(S_1)<g(S_2)))\Rightarrow (f'(S_1)<f'(S_2))$$

مساله را به دو بخش تقسیم می‌کنیم. بخش اول به صورت زیر اثبات می‌شود:

$$f = g + W_I \times h, f' = g + W_j \times h, W_j > W_i$$
$$\forall S_1, S_2 \mid (h(S_1) < h(S_2)) \Rightarrow f(S_1) < f(S_2)$$
$$\Rightarrow g(S_1) + W_i \times h(S_1) < g(S_2) + W_i \times h(S_2)$$
$$\Rightarrow g(S_1) + W_i \times h(S_1) + \Delta w \times h(S_1) < g(S_2) + W_i \times h(S_2) + \Delta w \times h(S_2)$$
$$\Rightarrow f'(S_1) < f'(S_2)$$

اثبات بخش دوم نیز به همین صورت است:

$$f = g + W_I \times h, f' = g + W_j \times h, W_j > W_i$$
$$\forall S_1, S_2 \mid ((h(S_1) = h(S_2)) \; and \; (g(S_1) < g(S_2)))$$
$$\Rightarrow f(S_1) < f(S_2)$$
$$\Rightarrow g(S_1) + W_i \times h(S_1) < g(S_2) + W_i \times h(S_2)$$
$$\Rightarrow g(S_1) + W_i \times h(S_1) + \Delta w \times h(S_1) < g(S_2) + W_i \times h(S_2) + \Delta w \times h(S_2)$$
$$\Rightarrow f'(S_1) < f'(S_2)$$

استفاده از روش GBFS* کمک می‌کند که حدی برای تغییرات وزن داشته باشیم. این حد دامنهٔ تغییرات وزن را تعیین خواهد کرد.

۲-۵ بررسی نقش وزن در روش WA*

در این بخش تاثیر تغییر وزن بر رفتار WA* را بررسی می‌کنیم. همانگونه که اشاره شد GBFS* شاخصی برای تعیین حد بالای W است. فرض کنید T کمترین وزنی است که در آن WA* به GBFS* تبدیل می‌شود. طبق فرض مساله با هر W بزرگتر یا مساوی T روش WA* به روش GBFS* تبدیل می‌گردد. بنابر این داریم:

$$\forall W > T, \forall S_1, \forall S_2, h(S_1) < h(S_2)$$
$$\Rightarrow g(S_1) + W \times h(S_1) < g(S_2) + W \times h(S_2)$$

چون $h(S_1) < h(S_2)$ است، پس رابطه زیر برقرار است:

$$W > \frac{g(S_1) - g(S_2)}{h(S_2) - h(S_1)}$$

درنتیجه

$$T \geq \frac{g(S_1) - g(S_2)}{h(S_2) - h(S_1)} \qquad (۳)$$

می‌دانیم که $g+h$ تخمینی از طول مسیر از وضعیت اولیه به وضعیت نهایی است. در هریک از وضعیت‌های میانی S_i، این تخمین ممکن است نسبت به اندازهٔ کمترین فاصله (فاصلهٔ بهینه $d_{optimal}$) مقداری خطا داشته باشد. فرض کنید حداکثر خطا برای وضعیت S_i شامل E_o و E_u می‌باشد. منظور از E_o حداکثر مقدار خطایی است که $g+h$ در هر وضعیت نامشخص بیشتر از اندازهٔ فاصلهٔ بهینه می‌تواند ارزیابی نماید. همچنین E_u حداکثر مقدار خطایی است که تابع $g+h$ در هر وضعیت نامشخص کمتراز اندازهٔ فاصلهٔ بهینه[۱۰] می‌تواند ارزیابی نماید. به عبارت دیگر :

$$\forall S_i, d_{optimal} - E_u \leq g(S_i) + h(S_i) \leq d_{optimal} + E_o,$$
$$E_o \geq 0, E_u \geq 0$$

$$d_{optimal} - E_u - h(S_i) \leq g(S_i) \leq d_{optimal} + E_o - h(S_i) \qquad (۴)$$

اگر بخواهیم کوچکترین T را با شرایط مساله پیدا کنیم، در نامساوی (۳) بایستی بزرگترین مقدار سمت راست نامساوی را به گونه ای تعیین کنیم که بیشینه شود. این کار با پیدا کردن بزرگترین مقدار برای کسر و کوچکترین مقدار برای مخرج آن میسر است. با توجه به (۳) و (۴) داریم:

$$T \geq \frac{g(S_1) - g(S_2)}{h(S_2) - h(S_1)} \geq \frac{(d_{optimal} + E_o - h(S_1)) - (d_{optimal} - E_u - h(S_2))}{h(S_2) - h(S_1)} = 1 + \frac{E_o + E_u}{h(S_2) - h(S_1)}$$

همچنین با این فرض که تخمین فاصله توسط تابع ابتکاری یک عدد صحیح باشد،کمترین مقدار $h(S_2) - h(S_1)$ برابر ۱ است. چراکه طبق فرض این دو برابر نیستند و $h(S_1) < h(S_2)$ می‌باشد. پس داریم:

$$T \geq 1 + E_o + E_u \qquad (۵)$$

بنابراین با داشتن حداکثر و حداقل مقدار خطای تابع تخمین A*، می‌توانیم کوچکترین وزن T را که روش WA* می‌تواند به GBFS* تبدیل شود را به دست آوریم. همچنین این رابطه نشان می‌دهد که مقدار T بزرگتر از ۱ است.

اگر تابع ابتکاری h هیچ خطایی نداشته باشد و همواره فاصلهٔ واقعی را برگرداند، آنگاه $E_u = E_o = 0$ و $T=1$. بنابر این روش‌های GBFS، GBFS* و A* معادل خواهند بود. چگونگی محاسبهٔ E_u و E_o یک مسالهٔ باز است و نیازمند بررسی بیشتر است.

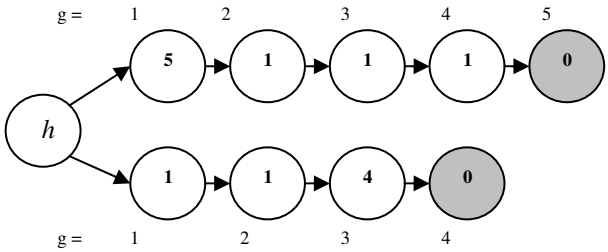
دامنهٔ تغییرات وزن در روش جستجوی WA* از ۱ تا T است. به ازای هر W متعلق به بازهٔ [1..T] روش WA* ممکن است رفتاری متفاوت داشته باشد. به عبارت دیگر ممکن است کیفیت و سرعت جستجو با تغییر W تغییر یابد. با توجه به اینکه مقدار T به دامنه، تابع ابتکاری و مسالهٔ وابسته است و در موارد متفاوت لزوما ثابت نیست، انتخاب یک وزن ثابت برای تمام شرایط درست به نظر نمی‌رسد.

در یک تابع ابتکاری مطلع، خطای تخمین نسبت به پیشرفت مراحل جستجو روندی کاهنده دارد. معمولا در فواصل دورتر از وضعیت هدف تخمین‌ها نادقیق‌تر و خطاها بیشتر و در فواصل نزدیکتر به هدف تخمین‌ها دقیق‌تر است. البته این به تابع ابتکاری نیز وابسته است. تابع ابتکاری HSP چنین است. بنابر این مقدار خطای تخمین طول نسبت به طول بهینه در فواصل نزدیکتر به هدف کمتر است. در این صورت بیشترین مقدار خطا در وضعیت‌های ابتدایی و کمترین آن در وضعیت‌های نهایی و نزدیک به هدف است که به سمت صفر می‌رود. با این اوصاف مناسب به نظر می‌رسد که W نیز ثابت نبوده و یک روند کاهنده داشته باشد. اگر بتوانیم وزن W [1..T] را به گونه ای مشخص کنیم که جستجو سریعتر و مناسبتر انجام شود، آنگاه چون T به سمت ۱ میل می‌کند، W نیز به سمت ۱ میل خواهد نمود. درنتیجه وزن W نسبت مستقیم با تخمین h در هر وضعیت دارد. لذا مناسبتر است که W به جای یک عدد ثابت تابعی از شرایط مختلف به خصوص تخمین فاصله h باشد.

مسایل بزرگتر تخمین‌های بزرگتری از فاصله دارند. توابع ابتکاری با توجه به ماهیت دامنه‌های مختلف رفتار متفاوتی دارند. در برخی از دامنه‌ها خطای بیشتر و در برخی از دامنه‌ها خطای کمتری دارند. دقت تخمین نیز در فواصل نزدیک به هدف با فواصل دورتر متفاوت است. همهٔ این موارد دال بر وابسته بودن وزن مناسب به تابع ابتکاری، دامنه و مساله است.

۲-۶ تغییرات وزن در روش *WA

در این بخش تغییر وزن بر عملکرد روش جستجوی *WA مورد بررسی قرار می‌گیرد. همانگونه که اشاره شد، دامنهٔ تغییرات W از عدد ۱ تا عدد T است. این تغییرات بر اندازهٔ جواب و زمان جستجو تاثیر دارد. بررسی‌های تجربی (در بخش بعد) نشان می‌دهد که به طور کلی افزایش وزن می‌تواند باعث کاهش کیفیت و افزایش سرعت گردد، اما این یک اصل نیست. مثال شکل ۱ که بر اساس یک تابع ابتکاری فرضی طراحی شده است، نشان می‌دهد افزایش وزن ممکن است نتیجه را بهتر کند.



شکل ۱- مثالی از جستجوی *WA که با افزایش وزن از ۱ به ۴، اندازهٔ پاسخ کاهش می یابد

در این مثال به ازای W=4 مسیر طولانی تر به طول ۱۰ پیمایش می‌گردد؛ در حالیکه با W=6 مسیر کوتاه تر به طول ۷ پیمایش می‌گردد. علت این امر اینست که تابع ابتکاری مزبور مطلع نیست. توابع ابتکاری مطلع‌تر تخمین‌هایی متناسب با فاصله تا هدف ارائه می‌کنند.

اگر فرض کنیم که وزن مساوی صفر باشد، همانگونه که اشاره شد روش جستجو سطح اول خواهد بود. در این صورت کوتاهترین جواب ممکن به دست می‌آید. در روش سطح اول عناصر به صورت سطح به سطح بررسی و گسترش می‌یابند. حال با در نظر گرفتن یک وزن W روش جستجو *WA خواهد بود. در این حالت امکان بررسی و مقایسهٔ وضعیت‌هایی از سطوح پایینتر با سطوح بالاتر حاصل می‌شود. با افزایش بیشتر W عمق بررسی نیز افزایش می‌یابد. بنابر این همانگونه که آزمایشات تجربی نشان می‌دهد، در اکثر موارد با افزایش وزن پاسخ در عمقی بیشتر به دست می‌آید. اگرچه همانگونه که در مثال فوق اشاره شد این یک اصل نیست و آزمایشات نیز وجود استثناءهایی را تایید می‌کنند.

تغییرات وزن در *WA نشان می‌دهد که یک تقابل[11] بین سرعت و کیفیت جستجو وجود دارد. افزایش وزن معمولا باعث افزایش سرعت و کاهش کیفیت می‌گردد. سوال اصلی در انتخاب وزن بسیار وابسته به میزان اهمیت کیفیت یا سرعت در کاربرد مورد نظر می‌باشد. اما به صورت کلی می‌توان دو روش را به عنوان مبنا انتخاب نمود. در روش اول درصدی کاهش کیفیت مد نظر گرفته و وزنی که سریعترین پاسخ را می‌یابد محاسبه شود. در روش دوم از میان وزن‌هایی که سریعترین پاسخ‌ها را ارائه می‌کنند، آنکه به بهترین کیفیت منجر می‌شود مورد توجه قرار گیرد. مثال‌های بررسی شده در بخش بعد این انتخاب‌ها را به خوبی نشان می‌دهند.

۳- بررسی تجربی

در بخش ۲ بررسی نظری *WA انجام گرفت و برخی از ویژگی‌ها و ابعاد آن مشخص شد. در این بخش با معرفی و تحلیل نتیجهٔ تعدادی آزمایش، تاثیر وزن‌های مختلف را بر کیفیت و زمان جستجوی *WA بررسی می‌کنیم. این نکته را که وزن مناسب برای شرایط مختلف تفاوت دارد، با چند مثال مناسب می توان نشان داد. به صورت کلی در انجام این آزمایش‌ها، سه هدف اصلی زیر مد نظر بوده است:

الف- آیا انتخاب وزن مناسب برای *WA به تابع ابتکاری وابسته است؟

ب- آیا انتخاب وزن مناسب برای *WA در مسایل مختلف تفاوتی دارد؟

ج- آیا انتخاب وزن مناسب برای *WA برای دامنه‌های مختلف تفاوتی ندارد؟

آزمایشات بر روی یک دستگاه رایانه Pentium(R) 4 CPU، ۲/۴ گیگا هرتز با ۲۵۶ مگابایت حافظه اجرا شده است. همچنین برای جستجو و حل مسایل، سیستم برنامه ریزی HSPr مورد استفاده قرار گرفت[۱۹]. سیستم HSPr یک برنامه ریز ابتکاری است که از *WA برای جستجو استفاده نموده است. طراحان این سیستم به صورت کلی عدد ۵ را به عنوان وزن توصیه نموده‌اند. خانوادهٔ HSP جزو اولین سیستم‌های برنامه ریزی ابتکاری هستند که ایدهٔ آن‌ها توسط سیستم‌های کارآمد بعدی همچون FF[۲۳] و YAHSP[۲۰] تکمیل شده و مورد استفاده قرار گرفته است.

برای تحقق اهداف آزمایش و مشاهده تفاوت تاثیرگذاری وزن های مختلف، به حداقل ۲ دامنه، ۲ مساله و ۲ تابع ابتکاری نیاز داریم. زیرا در این مقاله هدف رد این فرضیه است که یک وزن برای همهٔ شرایط مناسب است. در این پژوهش دو دامنهٔ کلاسیک برج هانوی و دنیای مکعب‌ها را به عنوان دامنه‌های مورد بررسی در نظر گرفته ایم. از هریک از این دامنه‌ها تعدادی مساله با اندازه‌های مختلف را انتخاب نمودیم و آن‌ها را با وزنهای مختلف توسط HSPr حل کردیم. این سیستم برنامه‌ریزی از روش *WA برای جستجو استفاده می‌کند. تعریف مسایل به صورت STRIPS که یک روش استاندارد نمایش دامنهٔ در برنامه ریزی هوش مصنوعی است، انجام شد[۲۴]. همچنین از دوتابع ابتکاری هزینهٔ تجمعی و هزینهٔ بیشینه که در [۱۹] تعریف شده استفاده کردیم.

۳-۱ معرفی توابع ابتکاری HSPr

همانگونه که اشاره شد برنامه ریز HSPr از *WA برای جستجو استفاده می‌کند. این برنامه‌ریز دو تابع ابتکاری هزینهٔ تجمعی[12] و هزینهٔ بیشینه[13] را معرفی کرده و مورد استفاده قرار داده است. در این توابع ابتدا یک مسالهٔ ساده شده[14] به سرعت حل می‌شود و به کمک حل آن تخمین فاصله برای مسالهٔ واقعی ارائه می‌گردد. در روش هزینهٔ تجمعی هزینهٔ رسیدن به یک هدف یک واحد بیشتر از مجموع هزینه‌های اهداف پیش‌نیاز آن در نظر گرفته می‌شود. اما در روش هزینهٔ بیشینه هزینهٔ رسیدن به یک هدف یک واحد بیشتر از مقدار بیشینه هزینه‌های اهداف پیش‌نیاز آن در نظر گرفته می‌شود. تابع ابتکاری حاصل از روش هزینهٔ بیشینه یک تابع *قابل قبول* می‌باشد و همواره پاسخ‌هایی کوچکتر از مقدار واقعی فاصله تا هدف بر می‌گرداند در مقابل تابع هزینه تجمعی یک تابع *مطلع* است و مقدار خروجی آن با فاصلهٔ واقعی تناسب بیشتری دارد. این تابع لزوما قابل قبول نیست و ممکن است تخمین‌هایی بزرگتر از مقدار واقعی ارائه نماید. معرفی و توضیحات بیشتر دربارهٔ این توابع در [۱۹، ۲۰] آمده است.

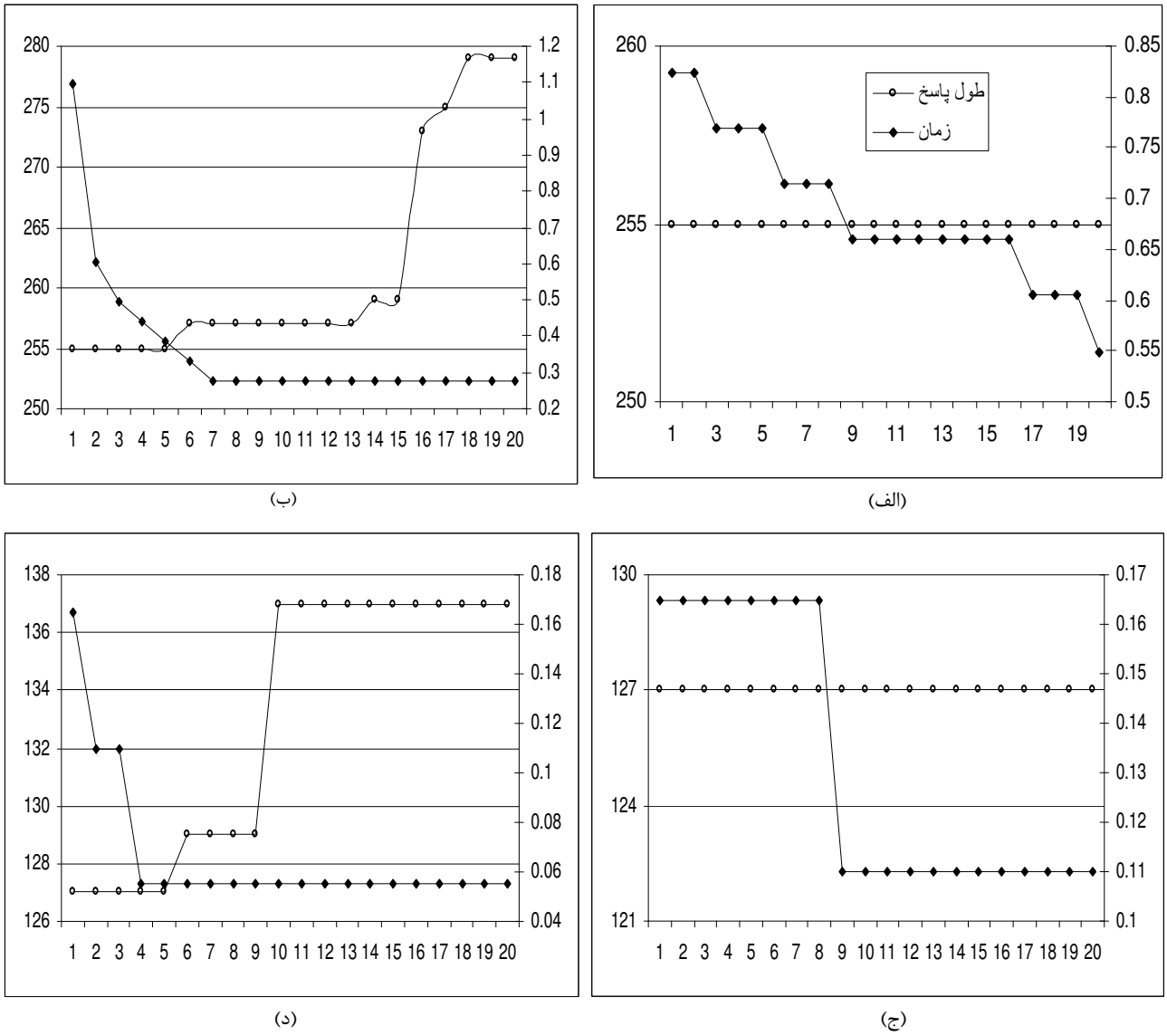
۳-۲ دامنهٔ هانوی

در دامنهٔ معروف هانوی، دو مساله، اولی هانوی با ۳ میله و ۸ دیسک و دومی هانوی با سه میله و ۷ دیسک مورد استفاده قرار گرفته است. هردوی این مسایل با ۲ تابع ابتکاری هزینهٔ تجمعی و هزینهٔ بیشینه توسط روش *WA با وزن‌های مختلف (۱ تا ۲۰) حل شده‌اند. شکل ۲ نتایج جستجوی این دو مساله توسط توابع ابتکاری یاد شده را نشان می‌دهد. نتایج بررسی هریک از آزمایش‌ها به صورت زیر است:

۳-۲-۱ نتایج مسالهٔ اول هانوی با تابع ابتکاری هزینهٔ تجمعی

در این مسالهٔ هانوی که شامل ۳ میله و ۸ دیسک می‌باشد، تابع ابتکاری هزینه تجمعی مورد استفاده قرار گرفته است. با توجه به شکل ۲- الف مشاهده می‌شود تابع ابتکاری هزینهٔ تجمعی معمولا بیش تخمین دارد، $E_u = 0$ خواهد بود. در نتیجه حداکثر خطای تابع در این مساله برابر ۵۲ است.

اندازهٔ پاسخ بهینه برای این مساله ۲۵۵ است و سریعترین پاسخ بهینه به ازای وزن ۵ به دست می‌آید. همچنین بهترین پاسخ سریع در وزن ۷ به دست آمده



شکل ۲- نتایج تست دو مساله در دامنهٔ هانوی: در هر شکل محور افقی تغییرات وزن را نشان می‌دهد. ستون سمت چپ اندازهٔ طول مسیر جستجو از وضعیت اولیه تا وضعیت هدف را نشان می‌دهد و ستون سمت راست زمان جستجو را نمایش می‌دهد. (الف) نتایج حل مسالهٔ اول با تابع ابتکاری هزینهٔ بیشینه (ب) نتایج حل مسالهٔ اول با تابع ابتکاری هزینهٔ تجمعی (ج) نتایج حل مسالهٔ دوم با تابع ابتکاری هزینهٔ بیشینه (د) نتایج حل مسالهٔ دوم با تابع ابتکاری هزینهٔ تجمعی

که افزایش وزن تا ۷ به صورت نمایی زمان جستجو را کاهش می‌دهد. با افزایش بیشتر وزن، سرعت تغییری نمی‌کند و ثابت می‌ماند؛ ولی طول مسیر یافت شده به صورت پله‌ای بزرگتر می‌شود. همانگونه که قبلا اشاره شد، افزایش وزن ممکن است اندازهٔ جواب را افزایش دهد. این مورد به صورت مشخص برای وزن‌های ۶، ۱۴، ۱۶، ۱۷ و ۱۸ آمده است. اختلاف اندازهٔ جواب در وزن‌های ۱۵ و ۱۶ برابر ۱۴ است که نشان می‌دهد تغییر وزن ممکن است تاثیر چشمگیری بر اندازهٔ پاسخ داشته باشد. بر اساس آزمایشات تکمیلی برای وزن‌های بزرگتر از ۵۳ روش جستجو عملا *GBFS می‌باشد و با بزرگتر شدن وزن کیفیت و سرعت جستجو تغییر نمی‌کند. طبق رابطهٔ شمارهٔ (۵) بخش ۵-۲ ($T \geq 1 + E_u + E_o$)، و با توجه به اینکه این است. همانگونه که در شکل مشاهده می شود بهترین پاسخ سریع، کوتاه ترین پاسخ در میان پاسخ‌هایی است که کمترین زمان جستجو را دارند.

۳-۲-۲ نتایج مسالهٔ اول هانوی با تابع ابتکاری هزینهٔ بیشینه

این مساله همان مسالهٔ قبلی است با این تفاوت که در جستجو تابع ابتکاری هزینه بیشینه مورد استفاده قرار گرفته است. با توجه به شکل ۲- ب مشاهده می‌شود که افزایش وزن تا عدد ۲۰ به صورت خطی-پله‌ای زمان جستجو را کاهش می‌دهد.

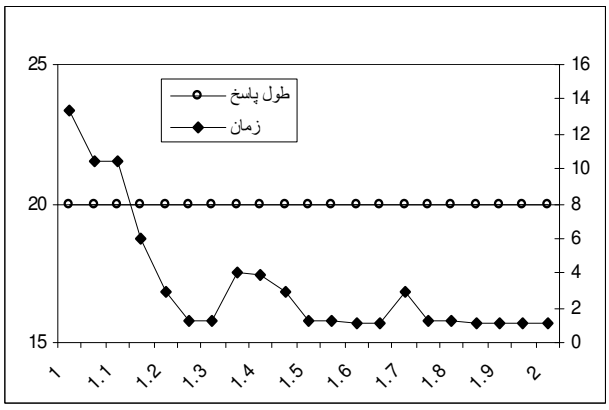
با افزایش بیشتر وزن، سرعت تغییری چندانی نمی‌کند. همچنین کیفیت جستجو یا طول مسیر یافت شده ثابت می‌ماند. بر اساس آزمایشات بیشتر برای

وزن‌های بزرگتر از ۴۵ روش جستجو عملا *GBFS می‌باشد. به عبارت دیگر برای وزن‌های بزرگتر از ۴۵ جستجو در مسیر ثابتی انجام می‌شود وکیفیت و سرعت تغییر نمی‌کند. طبق فرمول شمارهٔ (۵) بخش ۲-۵ ($T \geq 1 + E_u + E_o$)، و با توجه به اینکه این تابع ابتکاری معمولا کم تخمین دارد، E_o=0 خواهد بود. در نتیجه حداکثر خطای تابع در این مساله برابر ۴۴ است.

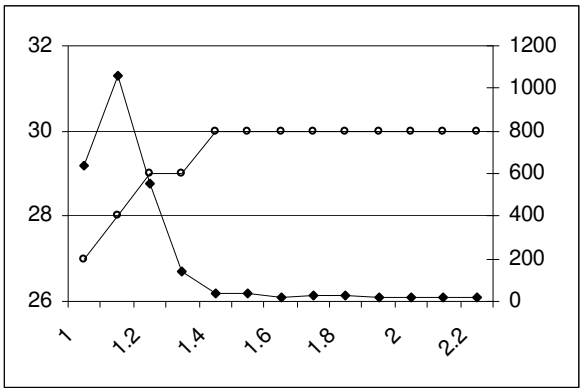
اندازهٔ پاسخ بهینه برای این مساله ۲۵۵ است و سریعترین پاسخ بهینه به ازای وزن ۲۰ به دست می‌آید. همچنین بهترین پاسخ سریع در وزن‌های بالاتر از ۱۹ نمی‌کند. علت ثابت بودن کیفیت به دلیل ویژگی قابل قبول بودن تابع ابتکاری مورد استفاده است.

اندازهٔ پاسخ بهینه برای این مساله ۱۲۷ است و سریعترین پاسخ بهینه به ازای وزن ۹ به دست می‌آید. همچنین بهترین پاسخ سریع نیز در وزن ۹ به دست آمده است.

بررسی و مقایسهٔ این مثال‌ها نشان می‌دهد که تاثیر وزن در کیفیت و زمان جستجو برای مسایل مختلف تفاوت دارد. از مقایسهٔ نتایج دو تابع ابتکاری در شکل



(الف)



(ب)

شکل ۳- نتایج تست دو مساله در دنیای مکعب‌ها: در هر شکل محور افقی تغییرات وزن را نشان می‌دهد. ستون سمت چپ اندازهٔ طول مسیر جستجو از وضعیت اولیه تا وضعیت هدف را نشان می‌دهد و ستون سمت راست زمان جستجو را نمایش می‌دهد. (الف) نتایج حل مسالهٔ دوم با تابع ابتکاری هزینهٔ تجمعی (ب) نتایج حل مسالهٔ اول با تابع ابتکاری هزینهٔ تجمعی

به‌دست آمده است.

۳-۲-۳ نتایج مسئله دوم هانوی با تابع ابتکاری هزینه تجمعی

در این مسالهٔ هانوی که شامل ۳ میله و ۷ دیسک می‌باشد نیز مانند مسالهٔ شکل ۲- الف تابع ابتکاری هزینه تجمعی مورد استفاده قرار گرفته است. با توجه به شکل ۲- ج مشاهده می‌شود که افزایش وزن تا عدد ۴ به صورت قابل توجهی زمان جستجو را کاهش می‌دهد. با افزایش بیشتر وزن، سرعت تغییری نمی‌کند و ثابت می‌ماند؛ ولی طول مسیر یافت شده به صورت پله‌ای بزرگتر می‌شود. این مورد به صورت مشخص برای وزن‌های ۶ و ۱۰ مشاهده می‌شود. اختلاف اندازهٔ جواب در وزن‌های ۹ و ۱۰ برابر ۸ است. با توجه به آزمایشات بیشتر برای وزن‌های بزرگتر یا مساوی ۲۸، روش جستجو عملا *GBFS می‌باشد و با بزرگتر شدن وزن کیفیت و سرعت جستجو تغییر نمی‌کند.

اندازهٔ پاسخ بهینه برای این مساله ۱۲۷ است و سریعترین پاسخ بهینه به ازای وزن ۴ و ۵ به دست می‌آید. همچنین بهترین پاسخ سریع در وزن ۴ و ۵ به‌دست آمده است.

۳-۲-۴ نتایج مسالهٔ دوم هانوی با تابع ابتکاری هزینهٔ بیشینه

این مساله همان مسالهٔ استفاده شده در بخش ۳-۲-۳ است، با این تفاوت که در جستجو تابع ابتکاری هزینه بیشینه مورد استفاده قرار گرفته است. با توجه به شکل ۲- د مشاهده می‌شود که به ازای وزن‌های بزرگتر از ۸ سریعترین پاسخ‌ها حاصل می‌شود. با افزایش بیشتر وزن، سرعت تغییری نمی‌کند و ثابت می‌ماند. همچنین کیفیت جستجو یا طول مسیر یافت شده به ازای همهٔ وزن‌ها ثابت می‌ماند. آزمایش‌ها نشان می‌دهند که برای وزن‌های بزرگتر از ۲۱ روش جستجو عملابه *GBFS تبدیل می‌شود و با بزرگتر شدن وزن، کیفیت و سرعت تغییر

۲-الف با ۲-ب و ۲-ج با ۲-د، تغییر وزن مناسب در توابع ابتکاری مختلف مشاهده می‌شود. همچنین از مقایسهٔ رفتار دو مسالهٔ مختلف در شکل‌های ۲-الف با ۲-ج و ۲-ب با ۲-د، متفاوت بودن وزن مناسب در مسایل مختلف مشاهده می‌شود. بنابر این ممکن است در یک مساله و در یک وزن مشخص بهترین جواب سریع را یافت و در مساله‌ای دیگر با یک وزن متفاوت به آن رسید. در نتیجه انتخاب وزن مناسب به نوع مساله و تابع ابتکاری بستگی دارد. در ادامه با بررسی مسایل دنیای مکعب‌ها تاثیر تغییرات وزن بر روش *WA را بر یک دامنه متفاوت توسط بررسی خواهیم نمود.

۳-۳ دنیای مکعب‌ها

دردنیای مکعب‌ها، دو مساله به ترتیب شامل ۱۹ و ۱۵ مکعب مورد آزمایش قرار گرفته است. هردوی این مسایل با ۲ تابع ابتکاری هزینهٔ تجمعی و هزینهٔ بیشینه توسط روش *WA با وزن‌های مختلف (۱ تا ۲) حل شده‌اند. در این دامنه، تابع ابتکاری هزینهٔ بیشینه بسیار زمانبراست و عملا به نتیجه نمی‌رسد. شکل ۳ نتایج جستجوی این دو مساله توسط تابع ابتکاری هزینهٔ تجمعی را نشان می‌دهد. نتایج بررسی هریک از آزمایش‌ها به صورت زیر است:

۳-۳-۱ نتایج مسالهٔ اول مکعب‌ها با تابع ابتکاری هزینهٔ تجمعی

در حل این مسالهٔ دنیای مکعب‌ها که شامل ۱۹ مکعب می‌باشد، تابع ابتکاری هزینه تجمعی مورد استفاده قرار گرفته است. با توجه به شکل ۳-الف مشاهده می‌شود که افزایش وزن تا ۱/۴ به جز در یک مورد (وزن ۱/۱)که زمان افزایش یافته است، به صورت نمایی زمان جستجو را کاهش و در مقابل اندازهٔ جواب را

افزایش می‌دهد. با افزایش بیشتر وزن، سرعت و همچنین اندازهٔ پاسخ تغییری نمی‌کند و ثابت می‌ماند. همانگونه که مشاهده می‌شود در وزن ۱ تنها حالت با پاسخ بهینه اتفاق افتاده و برای سایر وزن‌ها اندازهٔ جواب افزایشی است.

بر اساس آزمایشات تکمیلی برای وزن‌های بزرگتر از ۱/۹ روش جستجو عملا به GBFS* تبدیل می‌شود و با بزرگتر شدن وزن، کیفیت پاسخ و سرعت جستجو تغییر نمی‌کند. طبق رابطهٔ (۵) بخش ۲-۵ ($T \geq 1 + E_u + E_o$)، و با توجه به اینکه این تابع ابتکاری هزینهٔ تجمعی معمولا بیش تخمین دارد، $E_u = 0$ خواهد بود. در نتیجه حداکثر خطای تابع در این مساله برابر ۲۹ است.

همانگونه که مشاهده می‌شود. بر خلاف دامنهٔ هانوی که تغییرات در مقیاس ۱ باعث تغییر در سرعت و کیفیت می‌شد، در این دامنه مقیاس تغییرات در حد ۰/۱ است.

۳-۳-۲ نتایج مسالهٔ دوم مکعب‌ها با تابع ابتکاری هزینهٔ تجمعی

در مسالهٔ دوم از دنیای مکعب‌ها که شامل ۱۵ مکعب است (شکل ۳- ب) نیز تابع ابتکاری هزینه تجمعی مورد استفاده قرار گرفته است. تابع هزینهٔ بیشینه تنها برای وزن‌های بزرگتر از ۹ در زمانی طولانی پاسخ را می‌یابد. با توجه به شکل مشاهده می‌شود که افزایش وزن تا ۱/۳ ، به صورت نمایی زمان جستجو را کاهش می‌دهد. در این مثال اندازهٔ جواب افزایش نیافته است. با افزایش بیشتر وزن، سرعت نوساناتی دارد که ناشی از نوع تابع ابتکاری است. در این مساله نیز مقیاس تغییرات در حد ۰/۱ است. اگرچه برای وزن‌های بزرگتر یا مساوی ۱/۸ یک پایداری در زمان جستجو مشاهده می‌شود، اما بررسی‌های بیشتر نشان می‌دهد که برای وزن‌های بزرگتر از ۲/۳ روش جستجو عملا به روش GBFS* تبدیل می‌شود و لذا با بزرگتر شدن وزن، کیفیت راه حل و سرعت جستجو تغییر نمی‌کند. در اینجا نیز طبق رابطهٔ (۵) بخش ۲-۵ ($T \geq 1 + E_u + E_o$)، و با توجه به اینکه این تابع ابتکاری هزینهٔ تجمعی معمولا بیش تخمین دارد، $E_u = 0$ خواهد بود. در نتیجه حداکثر خطای تابع در این مساله برابر ۱۹ است. ضمنا پاسخ بهینه نیز ۱۹ است.

در نتیجه برای دامنه‌های مختلف، توابع ابتکاری متفاوت و مسایل مختلف وزن مناسب تفاوت دارد و تعیین وزن مناسب خود می‌تواند یک کار پژوهشی باشد.

۴- نتیجه‌گیری و جهت‌گیری آینده

سیستم‌هایی که از *WA استفاده می‌کنند معمولا وزن ثابتی در نظر می‌گیرند و آن چه که کمتر به آن پرداخته می‌شود، نحوهٔ انتخاب وزن ثابت است. در این مقاله با بررسی روش *WA موارد زیر را مشخص نمودیم:

الف- میزان خطا در تخمین‌های نزدیک کمتر از میزان خطا در تخمین‌های دور است. بنابر این افزودن وزن به روش *A باعث می‌شود که اهمیت تخمین‌های کوچکتر نسبت به تخمین‌های بزرگتر افزایش یابد.

ب- به دلیل ارتباط وزن با اندازهٔ تخمین تابع ابتکاری و تغییر این تخمین در حین جستجو، به نظر می‌رسد استفاده از یک تابع کاهشی متناسب با تخمین فاصله برای وزن مناسبتر از یک عدد ثابت باشد.

ج- افزایش وزن معمولا باعث کاهش کیفیت پاسخ و افزایش سرعت جستجو می‌گردد.

د- در حل هر مسالهٔ جستجو توسط *WA حدی برای افزایش وزن وجود دارد که پس از آن کیفیت و زمان جستجو تغییری نمی‌کند. بر اساس این حد یک روش جستجوی وزنی به نام *GBFS را در این مقاله تعریف نمودیم.

همچنین یکی از مواردی که در این مقاله بررسی شد، ارتباط نسبت تغییرات سرعت و کیفیت متناسب با تغییرات وزن بود. در دامنهٔ هانوی با تغییر وزن در مقیاس سرعت و کیفیت تحت تاثیر قرار می‌گیرد در حالیکه در دنیای مکعب‌ها این تغییرات با تغییرات وزن در مقیاس ۰/۱ همراه است.

با توجه به مباحث نظری و آزمایشات عملی در مسایل دامنهٔ هانوی و دنیای مکعب‌ها نشان دادیم که وزن مناسب برای حل هر مساله توسط *WA، با تابع ابتکاری، دامنه و اندازهٔ مساله ارتباط دارد. بنابر این انتخاب وزن ثابت برای همهٔ مسایل و همهٔ دامنه‌ها توسط محققین قبلی لزوما بهترین انتخاب نیست. به عبارت دیگر انتخاب یک وزن ثابت نمی‌تواند برای همهٔ شرایط مختلف (دامنه‌ها، توابع ابتکاری و مسایل مختلف) همواره مناسب باشد. برای استفادهٔ بهتر از وزن در روش *WA یک راه حل در نظر گرفتن وزن مشخص متناسب با شرایط مشخص است. راه‌حل دیگر تغییر تدریجی وزن در هنگام حل مساله می‌باشد. در این زمینه می‌توان از ایدهٔ ذوب فلزات(SA) به خوبی بهره گرفت.

به عنوان یک کار بعدی می‌توان تحقیقی را آغاز نمود که در آن از روش‌های یادگیری ماشین برای تعیین و یادگیری وزن متناسب با شرایط مختلف بهره برد. یادگیری یک وزن ثابت منطبق با شرایط هر مساله و یا یادگیری یک تابع که به صورت پویا وزن را تغییر دهد می‌تواند به عنوان دو گزینهٔ اصلی یادگیری مورد توجه قرار گیرد. با توجه به اینکه مقدار تخمین فاصله به دامنه، مساله و تابع ابتکاری ارتباط و وابستگی زیادی دارد، می‌توان از آن به عنوان یک پارامتر اصلی ورودی تابع یادگیری استفاده نمود.

قدردانی

این کار با حمایت مرکز تحقیقات مخابرات ایران انجام گرفته است.

مراجع

[1] S. J. Russell, and P. Norvig, "Artificial Intelligence: A Modern Approach, Second edition, " *Prentice-Hall*, 2003.

[2] E. Rich, and K. Knight, "Artificial Intelligence, Second edition, " *McGraw-Hill*, 1991.

[3] R. E. Korf, "Iterative-deepening A*: An optimal admissible tree search," *Proc. of the Ninth Intl Joint Conf. on Artificial Intelligence (IJCAI85)*, pp 1034-1036, 1985.

[4] S. J. Russell, "Efficient memory-bounded search methods," *Proce. 10th European Conf. on Artificial Intelligence (ECAI92)*, pp 1-5, 1992.

[5] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* with Provable Bounds on Sub-Optimality," *Advances in Neural Information Processing Systems 16 (NIPS)*, MIT, Cambridge, MA, 2004.

[6] S. Koenig, and M. Likhachev, "Adaptive A* [poster abstract]," *Proc. of the Intl Conf. on Autonomous Agents and Multi-Agent Systems*, pp 1311–1312, 2005.

[7] S. Koenig, and M. Likhachev, "Incremental A*," *Proc. of the Neural Information Processing Systems*, pp 1539-1546, 2001.

[8] A. Stentz, "The focuseed D* algorithm for real-time replanning," *Proc. of the Intl Joint Conf. on Artificial Intelligence*, pp 1652–1659, 1995.

[9] B. Reese, "AlphA* : An ε-admissible Heuristic Search Algorithm," url: http://citeseer.ist.psu.edu/390892.html

[10] R. Korf, "Real-time heuristic search," *Artificial Intelligence*, vol. 42, no. 23, pp 189–211, 1990.

[11] X. Sun, and S. Koenig, "The Fringe-Saving A* Search Algorithm - A Feasibility Study," *Proc. of the twentieth Intl Joint Conference on Artificial Intelligence (IJCAI07),* pp 2391-2397, 2007.

[12] S. Pearl, "Heuristics: Intelligent Search Strategies for Computer Problem Solving," *Addison-Wesley*, 1984.

[13] I. Pohl, "The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting, and computational issues in heuristic problem solving," *Proc. of Third Intl Joint Conf. on Artificial Intelligence (IJCAI 73)*, pp 20-23, 1973.

[14] R. E. Korf, "linear space best first search," *Artificial intelligence*, vol. 62, no. 1, pp 41-78, 1993.

[15] I. Pohl, "Heuristic search viewed as path finding in a graph," *Artificial Intelligence*, vol. 1, no. 3, pp 193-204, 1970.

[16] Y. Kitamura, M. Yokoo, T. Miyaji, and S. Tatsumi, "Multi-state commitment search," *Proc. of the Intl Conf. on Tools for Artificial Intelligence*, pp 431–439, 1998.

[17] D. Furcy and S. Koenig. "Scaling up WA* with Commitment and Diversity [Poster Abstract]," *Proc. of the nineteenth Intl Joint Conf. on Artificial Intelligence (IJCAI05),* pp 1521-1522, 2005.

[18] B. Bonet, and H. Geffner, "HSP: Heuristic Search Planner," *AIPS98 Planning Competition*, Pittsburgh, PA, 1998.

[19] B. Bonet, and H. Geffner, "Planning as Heuristic Search: New Results," *Proc. of ECP-99*, pp 360-372, 1999.

[20] V. Vidal, "A Lookahead Strategy for Heuristic Search Planning," *Proc. of 14th Intl Conf. on Automated Planning and Scheduling (ICAPS-04),* pp 150-160, 2004.

[21] R. Zluo, and E. A. Hansen, "Breadth-First Heuristic Search," *Artificial intelligence*, vol. 170, no. 4-5, pp 385-408, 2006.

[22] W. Ruml, M. B. Do, "Best-first utility-guided search," *Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, pp 2378-2384, 2007.

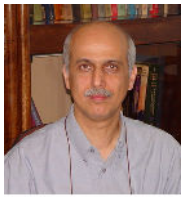
[23] J. Hoffmann, and B. Nebel, "The FF planning system: Fast plan generation through heuristic search," *Journal of Artificial Intelligence Research*, vol. 14, pp 253-302, 2001.

[24] R. E. Fikes, N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, no (3-4) pp 189-208, 1971.

سید علی اکرمی‌فر کارشناسی و کارشناسی ارشد خود را در مهندسی نرم‌افزار به ترتیب در سال‌های ۱۳۷۶ و ۱۳۷۸ از دانشگاه صنعتی شریف دریافت کرد. وی هم اکنون در حال انجام مرحلهٔ پژوهشی دورهٔ دکتری در رشتهٔ هوش مصنوعی با گرایش برنامه‌ریزی می‌باشد. زمینه‌های تحقیقاتی مورد علاقهٔ وی در رشتهٔ کامپیوتر عبارتند از روش‌های جستجوی ابتکاری، برنامه‌ریزی در هوش مصنوعی، یادگیری ماشین و الگوریتم‌های پیشرفته.

آدرس پست الکترونیکی ایشان عبارتست از:

akrami@ce.sharif.edu



غلامرضا قاسم ثانی کارشناسی خود را در سال ۱۳۶۴ در رشتهٔ علوم کامپیوتر از دانشگاه شهید بهشتی دریافت نمود. ایشان کارشناسی ارشد خود را در سال ۱۳۶۷ از دانشگاه اسکس انگلستان در رشتهٔ سیستم‌های هوشمند مبتنی بر دانش دریافت کرد. سپس در سال ۱۳۷۱ موفق به اخذ درجهٔ دکترا از همین دانشگاه در رشتهٔ علوم کامپیوتر گردید. ایشان هم اکنون عضو هیات علمی و دانشیار دانشگاه صنعتی شریف هستند. زمینه‌های تحقیقاتی مورد علاقهٔ ایشان در رشتهٔ کامپیوتر عبارتند از برنامه‌ریزی در هوش مصنوعی و پردازش زبان‌های طبیعی.

آدرس پست الکترونیکی ایشان عبارتست از:

sani@sharif.edu

---

[1] Heuristic Search Algorithms

[۲] در برخی از منابع تابع مکاشفه‌ای نیز نامیده شده است.

[3] Greedy best first search

[4] Breadth first

[5] Depth first

[6] Informative

[7] Admissible

[۸] علامت → نشانهٔ میل کردن به یک حد است.

[9] non-deterministic

[10] Underestimate

[11] Tradeoff

[12] Additive Cost Heuristic function

[13] Max Cost Heuristic function

[14] Relaxed problem