الگوریتمی جدید برای خزنده‌های وب جهت بهبود نتایج جستجو

اسماء قیصری[1] و علیرضا خلیلیان[2]

[1] کارشناس نرم‌افزار، دانشگاه فنی دکتر شریعتی، تهران
asma.gheisari@gmail.com

[2] کارشناس ارشد نرم‌افزار، دانشگاه علم و صنعت ایران، تهران
khalilian@comp.iust.ac.ir

چکیده

خزنده‌ها مهم‌ترین و اولین بخش موتورهای جستجو هستند که بهبود عملکرد آن‌ها تأثیر چشم‌گیری در بهبود نتایج نهایی دارد. محدودیت منابع نظیر زمان و حافظه، باعث می‌شود تا هر خزنده تنها ۸ تا ۱۲ درصد از حجم عظیم صفحات وب را در مخزن محلی خود نگه دارد. همه‌ی خزنده‌ها فرآیند خزش را در عمق خاصی مثلاً عمق ۵ متوقف می‌کنند. در این مقاله الگوریتمی پیشنهاد شده است تا صفحات بهتر، پیش از بقیه دانلود شوند. در نتیجه پس از توقف فرآیند خزش می‌توان مطمئن بود که بیشتر صفحات دانلود شده از بین بهترین صفحات قابل دسترس از URL شروع بوده‌اند. برای بررسی کارایی الگوریتم پیشنهادی، عملکرد آن با یکی از الگوریتم‌های جستجو مقایسه شده است.

کلمات کلیدی

خزنده ، موتور جستجو ، پیمایش، URL، Frontier

۱- مقدمه

در ابتدای پیدایش اینترنت، وب تنها یک شبکه‌ی کوچک بود که مجموعه‌ای نه چندان بزرگ از چند وب سایت ایستا در آن قرار داشت. با توجه به اینکه محتویات وب در ابتدا مقیاس کوچکی داشت، تنها توجه تعداد محدودی از افراد و سازمان‌ها را به سمت خود جلب کرده بود. با افزایش روز افزون حجم اطلاعات وب و به تبع آن افزایش نیاز کاربران به این اطلاعات مفید، نیاز به ابزاری که بتواند آخرین اطلاعات به‌روز را به‌صورت سازماندهی شده دسته‌بندی نماید و به کاربر امکان جستجو در بین این اطلاعات را دهد، بیشتر احساس می‌شود. هم‌اکنون کاربران اینترنت از موتورهای جستجوی قدرتمندی مثل Google و Yahoo برای پیدا کردن مجمو عه‌ای از اطلاعات در یک موضوع یا ساختار مشخص بهره می‌برند [1].

خزنده وب [2] مهم ترین و اولین بخش موتور جستجو است و با توجه به نقش بارزی که در طراحی موتور جستجو دارد، بهبود عملکرد آن از جنبه‌های مختلف، می‌تواند تاثیر چشم گیری بر بهبود عملکرد موتور جستجو داشته باشد. خزنده وب برنامه‌ایست که صفحات وب را پیدا کرده، به یک قالب استاندارد تبدیل، پیوندهای درون آنها را به صورت بازگشتی استخراج می‌کند و مجموعه اطلاعات به‌دست آمده را در اختیار بخش بعدی موتور جستجو که وظیفه‌ی اندیس گذاری را به‌عهده دارد، قرار می‌دهد [3].

از ابتدای پیدایش خزنده‌ها، بحث داشتن الگوریتم پیمایشی که بتواند مرتبط ترین و بیشترین صفحات را جمع آوری نماید، همیشه مطرح بوده است. بررسی‌های اخیر [4] نشان داده است که بیشتر خزنده‌های موتورهای جستجوی امروزی تنها موفق به کشف و اندیس‌گذاری کسر کوچکی از وب (۸ تا ۱۲ درصد) می‌شوند. در حالیکه محتویات این مجموعه‌ی ارزشمند برای کاربران اینترنت جایگاه فوق العاده مهمی دارد و کاربران انتظار دارند بتوانند به مفیدترین وب سایت‌هایی که به موضوع جستجویشان ارتباط دارد دست یابند [5].

خزنده در ابتدا تعدادی از صفحات را جمع آوری می‌کند و اندیس‌گذار آن‌ها را اندیس گذاری کرده و در مخزن موتور جستجو ذخیره می‌کند [1]. سپس هر چند وقت یک بار (مثلاً دو هفته) صفحات درون مخزن خود را مجدداً پیمایش می‌کند تا آخرین تغییرات صفحات را داشته باشد و مخزن محلی خود را با نسخه‌ی اصلی صفحات درون وب همگام کند. پس هر خزنده سیکل های متوالی دانلود دارد تا به این ترتیب مخزن محلی خود را به‌روز کند.

محدودیت منابع (سربار زمانی و محاسباتی) به ما اجازه نمی‌دهد که فرآیند بازگشتی پیمایش را تا عمق زیاد ادامه دهیم و باید آن‌را در

یک عمق خاص قطع کرد. از طرفی صرفه‌جویی در منابع مستلزم داشتن یک مخزن محلی به‌روز و با کیفیت می باشد.

به‌منظور حل مشکل مطرح شده، در این مقاله روشی برای پیمایش گراف وب پیشنهاد شده است که به ما کمک می‌کند تا بهترین نتایج را اول از همه پیدا کنیم. با توجه به اینکه ساختار وب به شکل گراف می‌باشد (صفحات وب رئوس گراف و پیوند بین آن‌ها یال‌های گراف می باشند)، الگوریتم خزش را روی گراف بیان می‌کنیم.

در بیشتر الگوریتم‌هایی که تاکنون مطرح شده‌اند، تمرکز تنها بر روی رتبه‌ی صفحات بوده است. یعنی سعی بر این بوده تا صفحاتی که رتبه‌ی بالاتری دارند اول پیمایش شوند. ولی در روش پیشنهادی هزینه‌ی مسیر در نظر گرفته می شود، یعنی تمرکز بر این است که با حرکت از یک گره (صفحه) به گره دیگر چقدر امتیاز به دست آورده یا اینکه چقدر امتیاز از دست می دهیم. برای محاسبه‌ی هزینه‌ی مسیر از امتیاز گره والد و فرزند استفاده می شود و با این روش نه تنها در حین پیمایش، رتبه‌ی صفحات به‌حساب می‌آید بلکه هزینه‌ی مسیر برای صفحات کم ارزش (با رتبه‌ی پایین) افزایش یافته و به این ترتیب از مسیر پیمایش حذف می‌شوند.

ساختار مقاله در ادامه به شرح زیر است: بخش دوم، مروری بر ادبیات موضوع خواهد داشت. بخش سوم روش پیشنهادی را مطرح می‌نماید. بخش چهارم در مورد ارزیابی روش پیشنهادی بحث می‌کند. بخش پنجم هم به نتیجه‌گیری اختصاص دارد.

۲- مروری بر ادبیات موضوع و کارهای پیشین

یک مخزن وب، مجموعه‌ای بزرگ از صفحات وب و اندیس‌های مربوط به آنهاست که به منظور خاصی گرد آمده اند [4]. مهم ترین بخش خزنده زمانبند می‌باشد که URL بعدی را از لیست URLهای دیده نشده بر می‌دارد تا پردازش شوند. با توجه به عملکرد پایه‌ای خزنده‌ها می‌بینیم که در هر مرحله خزنده یک پیوند جدید را از صف Frontier (صفی که URLها را در خود نگهداری می‌کند) انتخاب می‌کند تا پردازش شود. انتخاب پیوند بعدی از Frontier کاملاً وابسته به الگوریتم پیمایش است که استفاده می‌شود. پس انتخاب پیوند بعدی از Frontier شبیه انتخاب یک کار توسط CPU از صف پردازش‌های آماده به اجرا در سیستم عامل است.

الگوریتم پایه‌ای پیمایش به این صورت است که لیستی از URLها را به عنوان ورودی می‌گیرد،در صف Frontier قرار داده و گام‌های زیر را مکرراً اجرا می‌کند [3]: (۱) یک URL از صف Frontier حذف می‌کند. (۲) آدرس IP میزبان آن‌را تعیین می‌کند. (۳) سند مربوط به آن‌را دانلود می‌کند. (۴) همه‌ی پیوندهای داخل سند بازیابی شده را استخراج کرده و به Frontier اضافه می‌نماید.

هر الگوریتم پیمایشِ وب پایه، به تعدادی از اجزای عملکردی نیاز دارد [3]: (۱) مؤلفه‌ای برای ذخیره‌ی لیست URLهای دانلود شده (Frontier). (۲) مؤلفه‌ای برای تبدیل نام میزبان به آدرس IP. (۳) مؤلفه‌ای برای دانلود اسناد با استفاده از پروتکل HTTP. (۴) مؤلفه‌ای برای استخراج پیوندها از سند HTTP بازیابی شده. (۵) مؤلفه‌ای برای تعیین اینکه آیا یک URL قبلاً به حساب آمده یا نه.

هورسویسی و همکارانش [6] در سال ۱۹۹۸ الگوریتم Shark-Search را ارائه دادند که از شباهت موضوعی یک تکنیک مدل فضای برداری بعنوان یک پارامتر در فرآیند مرتب سازی استفاده نمود. این شباهت از مقایسه بین کلمات کلیدی و محتوی صفحات وب ناشی می‌شود. اگر یک صفحه وبِ جمع آوری شده دارای یک امتیاز شباهت یا همان میزان انطباق بالا باشد، هم خود صفحه وب و هم URLهای پیدا شده در آن صفحه می‌توانند به یک موضوع مورد نظر مرتبط باشند.

دین و هنزینگر رابطه خواهر برادری صفحه را co-citation نامیدند [7]. از طرف دیگر رنی و مک کالم در سال ۱۹۹۹ معتقد بودند که خزنده، صفحات وب مرتبط را مستقیماً درک نمی‌کند بلکه آنها را پس از گردآوری صفحات وب غیر مرتبط کشف می‌نماید. بنابراین آنها روشی را شرح داده اند که به خزنده اجازه می‌دهد راهی را که به صفحات وب مرتبط ختم می‌شود را یاد بگیرند [8].

اگراول و همکارانش [2] معتقد بودند که اکثر صفحات وبی که تحت یک فهرست والد یکسان قرار داشته و صفحات خواهر و بردار یا همزاد نامیده می‌شوند، به یک موضوع مورد نظر مرتبط خواهند بود. احتمال اینکه مابقی صفحات وب تحت فهرست یکسان به موضوع مورد نظر ارتباط داشته باشند، بالا است.

در سال ۲۰۱۰ رادهاکیشان، یاسر ساروک و سلواکومار یک خزنده‌ی وب با الگوریتم جستجوی بر اساس متن CRAYSE را ارائه دادند [9] که الگوی متنی را داخل صفحات به‌صورت بازگشتی جستجو می‌کند و البته جستجو و پیمایش را به صورت همزمان انجام می‌دهد.

۳- روش پی‌شنهادی

در این بخش روشی برای پیمایش، در راستای بهبود نتایج جستجو و صرفه جویی در منابع ارائه می‌شود هر بار گرهی را انتخاب می‌کنیم که کم‌ترین هزینه‌ی مسیر برای رسیدن به آن پرداخت می‌شود. این روش فرضیاتی دارد: (۱) الگوریتم رتبه‌بندی از قبل اعمال شده و امتیاز گره‌ها مشخص شده است. (۲) با توجه به اینکه هنوز صفحات دانلود نشده‌اند، ما اطلاعی از وضعیت آن‌ها نداریم و فرض می کنیم که الگوریتم رتبه‌بندی استفاده شده جز آن دسته از الگوریتم هاییست که امتیاز صفحات را پیش از دانلود آن‌ها و با توجه به ساختار صفحات محاسبه می کند (مثلاً ساختار پیونددهی آنها، کیفیت صفحاتی که به آن‌ها پیوند داده‌اند، تعداد پیوندهایی که به آن‌ها وارد شده است و غیره). البته از این روش می‌توان برای سیکل‌های پیمایش بعدی هم استفاده کرد. یعنی یک‌بار صفحات را پیمایش کرده و امتیاز آن‌ها را محاسبه کنیم و دفعات بعد روش پیشنهادی را اعمال کنیم. به این ترتیب می‌توانیم از هر نوع الگوریتم رتبه‌بندی استفاده کنیم، چون این

بار صفحات از پیش دانلود شده‌اند. در جریان اجرای این روش، هزینه‌ی حرکت از یک گره به گره دیگر را برای هر یال حساب می‌کنیم. برای محاسبه‌ی هزینه‌ی یال‌ها (هزینه‌ی حرکت از یک گره به گره دیگر) از این فرمول استفاده می‌کنیم:

$$\left\lceil \frac{rank(parent)}{rank(child)} * rank(parent) \right\rceil \qquad (۱)$$

جمله‌ی *rank(parent)* به امتیاز گره والد و جمله *rank(child)* به امتیاز گره فرزند اشاره می‌کنند. فرمول (۱) یک نسبت خوب برای محاسبه‌ی هزینه‌ی حرکت از یک گره به گره دیگر است. اگر از گرهی با امتیاز بالا به گرهی با امتیاز پایین حرکت کنیم، هزینه‌ی یال افزایش می‌یابد و برعکس.

این الگوریتم سعی می‌کند گره‌های با امتیاز بالا را پیش از بقیه انتخاب کند (یا به عبارت دیگر هزینه ی مسیر را حداقل کند). هزینه‌ی مسیر برای هر گره جمع هزینه‌ی تمام یال‌های سر راه آن گره (از گره شروع تا گره فعلی) می‌باشد. اگر در مسیری قرار بگیریم که گره‌ها امتیاز کمی داشته باشند، هزینه‌ی مسیر خود به خود افزایش یافته و مسیر حرکت طوری تغییر می‌کند که گره‌های با ارزش‌تر پیمایش می‌شوند. به این ترتیب گره‌های کم ارزش خود به خود از مسیر حرکت خزنده حذف می شوند. هزینه‌ی مسیر هر گره در واقع جمع هزینه‌ی یالهاییست که از گره شروع باید طی کنیم تا به آن گره برسیم.

مسأله: با توجه به اینکه این الگوریتم همیشه از مسیرهای پرهزینه دوری می‌کند، ممکن است این سوال بوجود آید که آیا مسیرهای پرهزینه همیشه ما را به سمت گره‌های بی کیفیت هدایت می‌کنند؟

در بحث خزنده‌ها ثابت شده است [10] که گره‌های با امتیاز پایین با احتمال بالایی ما را به سمت گره‌های بی کیفیت هدایت می‌کنند. فرض کنید به دنبال مطلبی در مورد جنگ جهانی دوم هستیم. حالا به سایتی برخورد می‌کنیم که در مورد مُد می‌باشد. پس امتیاز پایینی به این گره انتساب می‌یابد. چقدر احتمال دارد که این سایت، پیوندی به مطلبی در مورد جنگ جهانی دوم داشته باشد؟ ساختار پیوند دهی و بسیاری از فاکتورهای دیگر که در امتیازدهی صفحات وب اعمال می‌شوند، نشان می‌دهند که از مسیرهای پرهزینه باید دوری کرد. شبه کد روش پیشنهادی در شکل (۱) دیده می‌شود.

برای روشن شدن عملکرد الگوریتم، مثالی ارائه می‌کنیم: شکل‌های (۲) الی (۶) اجرای الگوریتم را روی یک گراف (شبکه وب) نمونه نشان می‌دهند. این گراف نمونه همان شکل (۶) است که گره‌ها، دایره‌ها هستند و به‌دلیل کمبود فضا رسم نکرده‌ایم. در این شکل‌ها عدد سمت راست روی هر یال هزینه‌ی مسیر از گره شروع تا گره فعلی می‌باشد و عدد سمت چپ هزینه‌ی حرکت از هر گره به گره بعد است. مثلاً هزینه‌ی مسیر از گره شروع(A) تا گره G برابر ۲۵ است و هزینه‌ی حرکت از گره G به K برابر سقف ۶/(۸*۸) و ۱۱ می‌شود. پس هزینه‌ی مسیر از گره شروع تا گره K برابر ۳۶ است (۲۵+۱۱). هر بار گره با کم‌ترین هزینه‌ی مسیر را انتخاب کرده، سپس امتیاز و هزینه‌ی مسیر گره‌های فرزند آن را محاسبه و از بین برگ‌های موجود برگ با کم‌ترین هزینه‌ی مسیر را انتخاب می‌کنیم و به همین شکل ادامه می‌دهیم. ملاحظه می‌شود که با استفاده از تخمین هزینه‌ی مسیر و تلاش برای حداقل کردن آن، می‌توان توزیع مناسب تری به‌دست آورده و با احتمال بیشتری گره‌های با کیفیت را اول از همه به دست آوریم.

به کادر قرمز رنگ در شکل (۶) توجه کنید. مشاهده می‌شود که هزینه‌ی مسیر برای رسیدن به این گره‌ها بالاست و در نتیجه از فرآیند خزش حذف شدند و اگر فرآیند خزش را ادامه دهیم، قطعاً آخرین گره‌هایی که پیمایش می‌شوند همین‌ها هستند. این الگوریتم به تعداد گام‌هایی که در یک مسیر وجود دارد توجه نمی‌کند، بلکه مجموع هزینه‌های آن‌ها را در نظر می‌گیرد. پس با حرکت در یک مسیر، هزینه مسیر همیشه افزایش می‌یابد. لذا می‌توان نتیجه گرفت که الگوریتم همیشه گره‌ها را به‌ترتیب افزایش هزینه‌ی مسیر باز می‌کند. در نتیجه اولین گره هدفی که برای باز کردن انتخاب می‌شود، جواب بهینه‌ی مورد نظر می‌باشد (می‌دانیم که الگوریتم پیشنهادی، تنها آزمون هدف را برای گره‌هایی انجام می‌دهد که برای باز شدن انتخاب می‌شوند).

۴- ارزیابی

برای ارزیابی الگوریتم پیشنهادی، الگوریتم اول سطح به‌عنوان حد پایین کارایی در نظر گرفته شده است [3]. می‌خواهیم بررسی کنیم که چه تعداد گره‌های موجود در گراف، گره هدف می‌باشند. یعنی ما در این گراف به دنبال گره‌های هدف می‌باشیم و این گره‌های هدف در واقع جواب‌های بهینه‌ی ما می‌باشند. پس برای بررسی کارایی الگوریتم پیشنهادی، آن‌را با الگوریتم اول سطح مقایسه می‌کنیم. برای اثبات کارایی این الگوریتم برخلاف روش اول سطح نمی‌توانیم از b (فاکتور انشعاب) و d (عمق) استفاده کنیم. چون در روش پیشنهادی عمق جواب بهینه اصلاً معلوم نیست. پس فرض می‌کنیم Z هزینه‌ی رسیدن به جواب بهینه و ϵ حداقل هزینه‌ی هر گام (هزینه‌ی رفتن از هر گره به گره دیگر) است. بنابراین حداکثر عمق جواب بهینه $d=[Z/\epsilon]$ بوده و پیچیدگی زمانی این الگوریتم در بدترین حالت $O(b^{[Z/\epsilon]+1})$ خواهد بود. این زمان از $O(b^d)$ که پیچیدگی زمانی الگوریتم اول سطح است بهتر می‌باشد. علت این است که الگوریتم پیشنهادی می‌تواند (و اکثر اوقات این‌گونه است) یک زیر درخت بزرگ از گام‌های کوچک را قبل از یک زیر درخت شامل مسیرهای بزرگ با گام‌های شاید مفید، پیمایش کند. پس در نظر گرفتن هزینه‌ی مسیر چند فایده‌ی مهم دارد: (۱) با افزایش هزینه‌ی مسیر، گره‌های بی کیفیت خود به خود حذف می‌شوند. (۲) هدایت خزش به سمت گره‌های بهتر. (۳) بهترین گره‌ها را اول پیدا می‌کند و در مواقعی که محدودیت منابع وجود دارد و مجبور به قطع پیمایش در عمق کم می‌باشیم، بسیار مناسب است. (۴) اگر URL شروع خوب نباشد، ما را به گره‌های بی کیفیت هدایت کرده و هزینه‌ی مسیر به سرعت شروع به افزایش می‌کند.

```
input: The web graph whose nodes are Internet pages and an edge from node m to node n shows that there is a link to n in m.
output: Some traversed nodes in the graph that have a minimum path cost according to Equation (1).
declare:
    Frontier: The queue in which the traversing nodes are added, initially seed, such that seed is a starting node. This
              queue has two functions, Inqueue() and Dequeue() to insert and delete nodes in it.
    Current: The current node that is being processed
    Node: Each immediate adjacent node connected to the current node that is not visited yet
algorithm WebCrawling begin
  while (Frontier.IsNotEmpty()) do begin
    Current = Frontier.Dequeue_Least_PathCost_Node()
    for each Node in Current.ImmediateAdjacentNodes that is not visited yet  do begin
      Node.PathCost = Ceil((Current.Rank * Current.Rank) / Node.Rank)
      Frontier.Inqueue(Node)
    end for
  end while
end WebCrawling
```

شکل (۱) : شبه کد الگوریتم پیشنهادی

می‌توان با بررسی میزان و سرعت افزایش هزینه‌ی مسیر به این نتیجه رسید که آیا URL شروع خوب بوده یا نه. در صورتی که URL شروع خوب نباشد می‌توان خزش را متوقف کرده و آن را با URL شروع دیگری ادامه دهیم. به این ترتیب کارایی و کیفیت مجموعه صفحات دانلود شده به‌طور چشم گیری افزایش می‌یابد.

۵- نتیجه‌گیری

پیش از دانلود یک صفحه وب نمی‌توان وضعیت آن از قبیل امتیاز، میزان مطلوبیت یا اینکه آیا صفحه از آخرین سیکل دانلود تا کنون تغییر کرده است یا نه را ارزیابی کرد. در این مقاله الگوریتمی ارائه شده است که با محاسبه هزینه مسیر هنگام پیمایش صفحات، سعی می‌کند صفحات بهتر پیش از بقیه دانلود شوند. در نتیجه پس از توقف فرآیند خزش، می‌توانیم مطمئن باشیم که بیشتر صفحات دانلود شده، از بین بهترین صفحات قابل دسترس از URL شروع، بوده‌اند.

مراجع

[1] Pinkerton, B., *WebCrawler: Finding What People Want*, University of Washington, PhD Thesis, 2000.

[2] Aggarwal, C., Garawi, F., Yu, P., "Intelligent crawling on the world wide web with arbitrary predicates", In: Proceedings of the 10th International World Wide Web Conference, Hongkong, p. 96–105, 2001.

[3] Dorado, I., G., *Focused Crawling: algorithm survey and new approaches with a manual analysis*, Master Thesis, 2008.

[4] Sharma, S., *Web-Crawling Approaches in Search Engines*, Master Thesis, 2008.

[5] Pant, G., Srinivasan, P., Menczer, F., "Crawling the Web", *Web Dynamics*, edited by M. Levene and A. Poulovassilis, Springer, pp. 153-177, 2004.

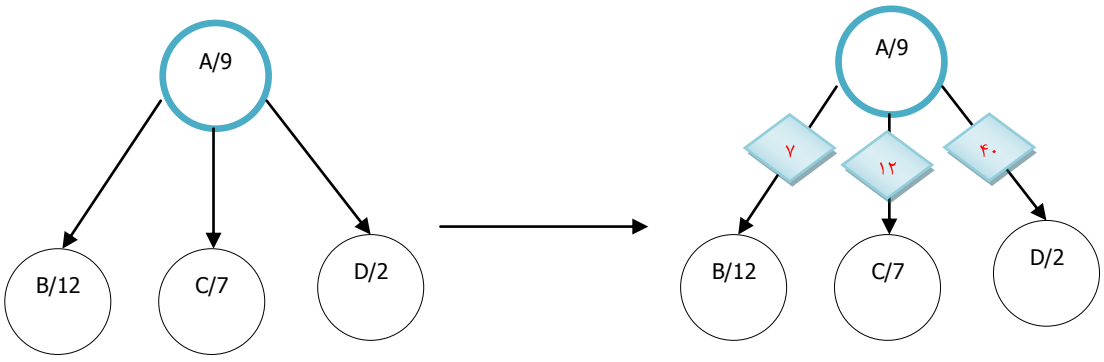
[6] Hersovici, M., Jacovi, M., Maarek, Y.,S., Pelleg, D., Shtalhaim, M., Ur, S., "The shark-search algorithm-an application: tailored web site mapping", In: Proceedings of the Seventh International World Wide Web Conference, Brisbane, Australia, April, p. 317–26, 1998.

[7] Dean, J., Henzinger, M., R., "Finding related pages in the world wide web", In: Proceedings of the Eighth International World Wide Web Conference, Toronto, Canada, p. 389–401, 1999.

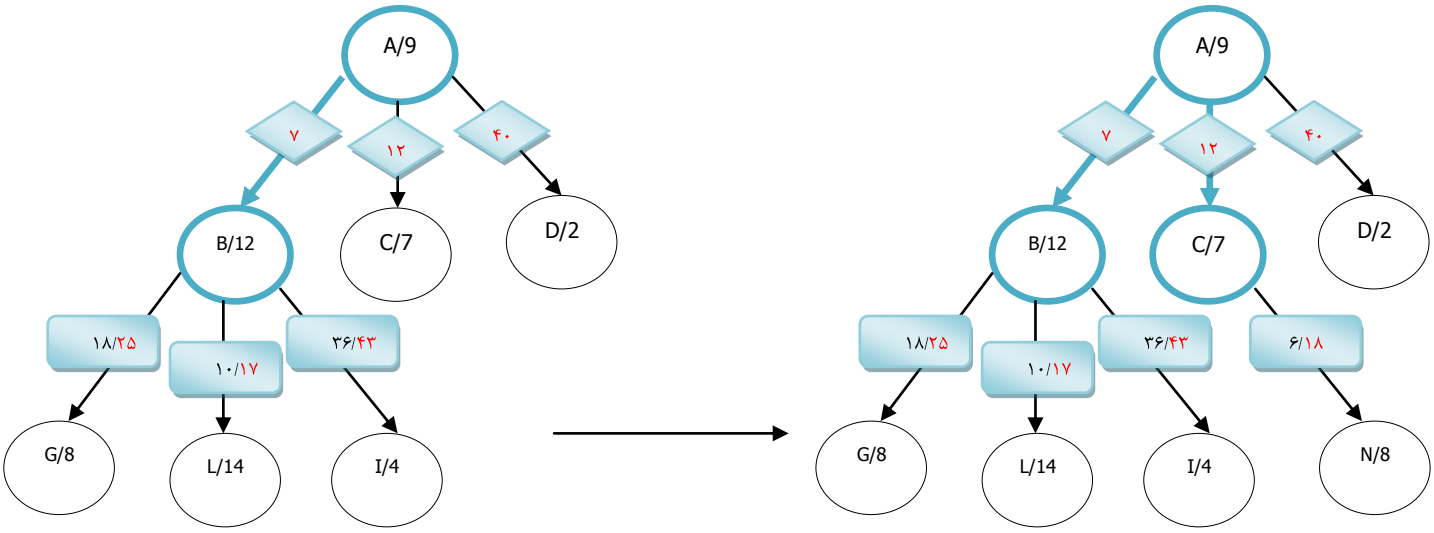
[8] Rennie, J., McCallum, A., "Efficient web spidering with reinforcement learning", In: Proceedings of the 16th International Conference on Machine Learning, Bled, Slovenia, p. 335–43, 1999.

[9] Radhakishan, V., Farook, Y., Selvakumar, S., "CRAYSE: design and implementation of efficient text search algorithm in a web crawler", ACM SIGSOFT Software Engineering Notes, Vol. 35(4), 2010.

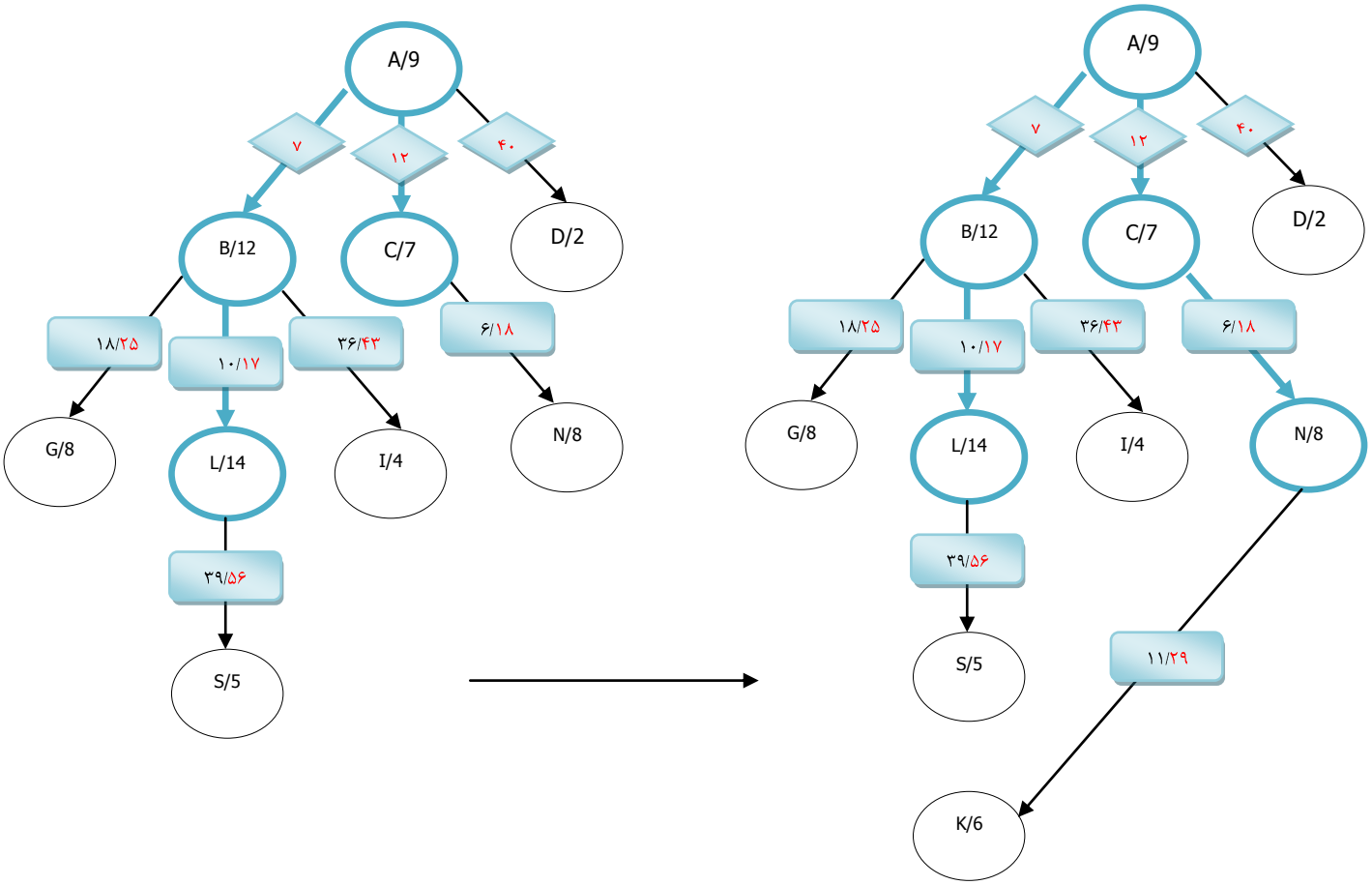
[10] "GoogleGuide", www.googleguide.com/google_works.html, 2007.



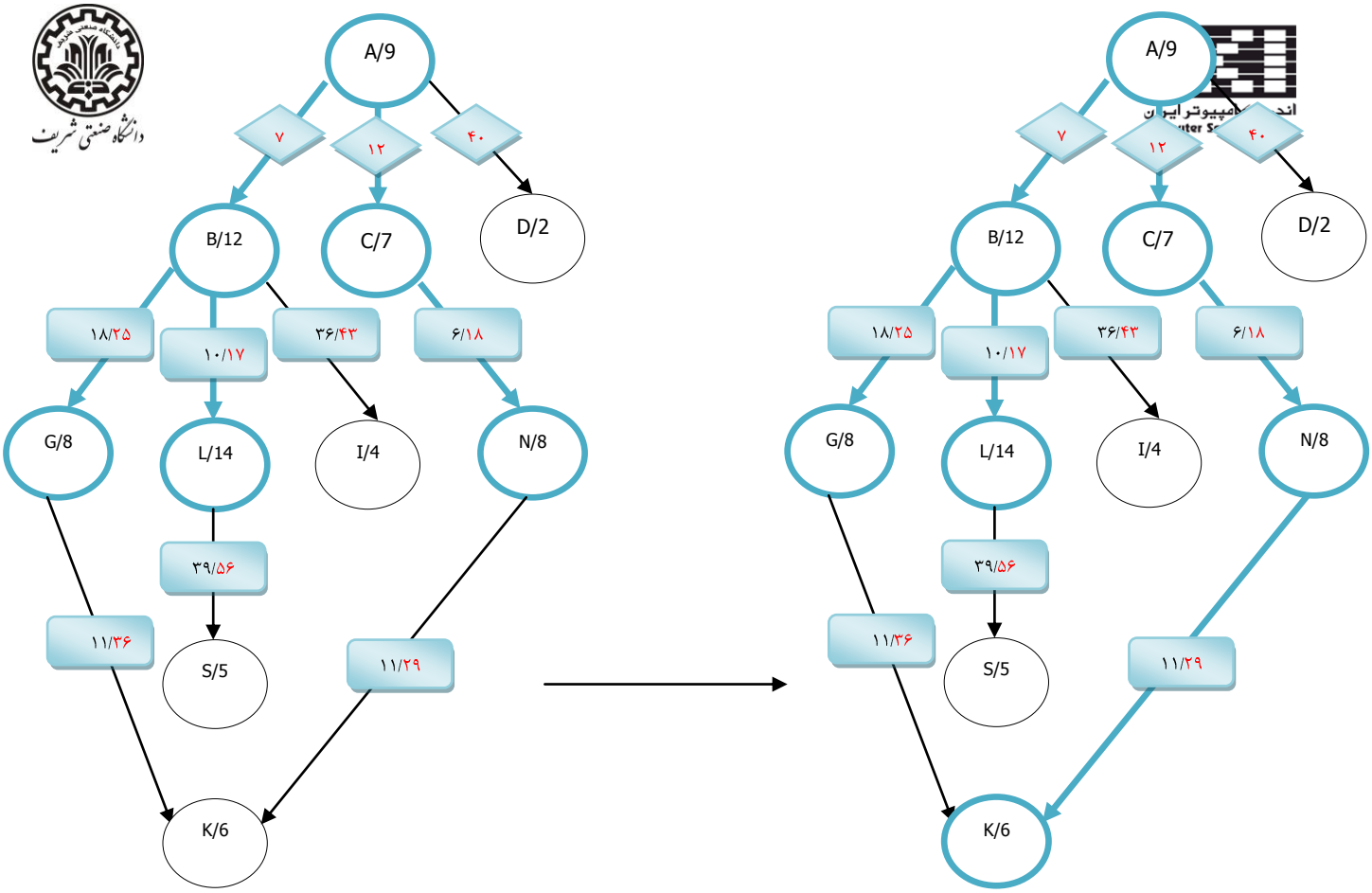
شکل (۲) : مرحله اول اجرای الگوریتم پیشنهادی



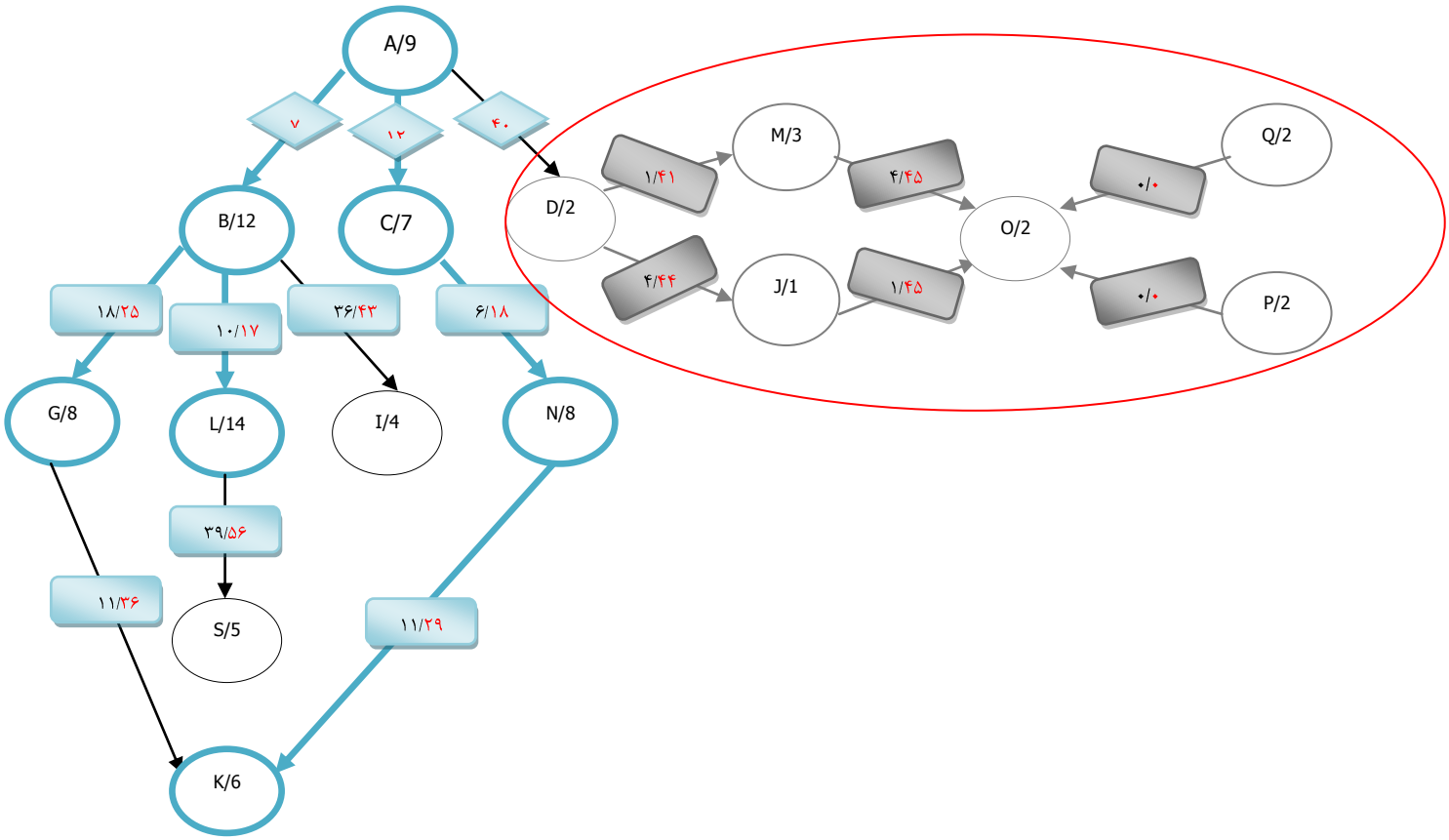
شکل (۳) : مرحله دوم اجرای الگوریتم پیشنهادی



شکل (۴) : مرحله سوم اجرای الگوریتم پیشنهادی



شکل (۵) : مرحله چهارم اجرای الگوریتم پیشنهادی



شکل (۶) : مرحله پنجم اجرای الگوریتم پیشنهادی