کارگاه های آموزشی مرکز اطلاعات علمی جهاد دانشگاهی

کارگاه آنلاین آشنایی با پایگاه های اطلاعات علمی بین المللی و ترفند های جستجو

کارگاه آنلاین
پروپوزال نویسی و پایان نامه نویسی

کارگاه آنلاین
بررسی مقابله ای متون (مقدماتی)

بهینه سازی بر مبنای شبیه سازی در بهره برداری از مخازن سدها: رویکرد یادگیری تقویتی

بهزاد شریف، دانشجوی کارشناسی ارشد عمران - محیط زیست دانشگاه علم و صنعت ایران×
سید جمشید موسوی، دانشیار دانشکده مهندسی عمران دانشگاه صنعتی امیرکبیر
b_sharif @ iust.ac.ir ،۰۹۳۲۹۳۹۲۹۴۱

چکیده

استفاده از برنامه ریزی پویای استوکستیک (SDP) در بهینه سازی مسائل بزرگ مقیاس بهره برداری از مخازن سدها به دلیل نیاز به گسسته سازی متغیرهای حالت و تصمیم، و در نتیجه مشکل ابعادی با محدودیتهای جدی مواجه است. روش یادگیری تقویتی (RL) یکی از تکنیکهای پیشرفته در حل مسائل تصمیم گیری متوالی در محیط استوکستیک و مبتنی بر شبیه سازی است. RL می تواند با پیدا کردن سیاست بهینه برای حالتهایی از سیستم که در واقعیت بیشتر رخ می دهند به جوابهایی نزدیک به جواب بهینه، در زمان قابل قبول نایل شود. در این مقاله، مساله بهینه سازی بهره برداری از سد مخزنی چراغ ویس واقع در استان کردستان به عنوان مطالعه موردی با استفاده از روش RL مطالعه شده و با روش SDP مقایسه گردیده است. نتایج نشان دهنده همگرایی مطلوب روش RL در نیل به جواب بهینه است.

کلید واژه ها: ***یادگیری تقویتی، برنامه ریزی پویا، بهره برداری از مخزن***

۱- مقدمه

استفاده از برنامه ریزی پویا (DP) در حل مسائل بهینه سازی غیر خطی برای بدست آوردن مقدار بهینه کلی در سیستمهای بهره برداری از مخزن، در مطالعات زیادی مورد بررسی قرار گرفته است. DP به سهولت و به طور موثر قادر به اداره خصوصیاتی چون غیرخطی بودن، احتمالی بودن، غیرمحدب بودن و مشتق‌ناپذیری توابع است، که از مشکلات اساسی در سایر روشهای بهینه‌سازی می‌باشند. Yakowitz [1] به طور جامع اقدام به دسته‌بندی کاربردهای مختلف DP در مسائل منابع آب نموده است. موسوی و کارآموز [۲] و موسوی و همکاران [۳] از DP در بهینه‌سازی بهره‌برداری از سیستمهای چند مخزنه بهره گرفتند.

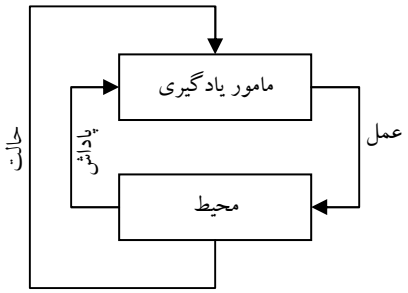
یکی از مشکلات استفاده از DP، نیاز به گسسته سازی متغیرهای حالت و تصمیم است. بنابراین در مسائل بزرگ مقیاس، با افزایش تعداد متغیرهای حالت و تصمیم، مسئله دچار مشکل ابعادی خواهد شد که از آن اصطلاحاً به عنوان نفرین بعد یاد می شود. روشهای یادگیری تقویتی، روشهایی نزدیک به برنامه ریزی پویا و مبتنی بر شبیه سازی هستند که می توانند با استفاده از تکنیکهایی، جواب نزدیک به بهینه را بدون محاسبه تابع ارزش تمام متغیرهای حالت و تصمیم بدست آورند.

این روش یکی از تکنیکهای پیشرفته در حل مسائل تصمیم گیری متوالی در محیط استوکستیک است. در این مقاله، روش RL در مسئله بهینه سازی بهره برداری از مخزن سد در محیط استوکستیک استفاده شده و نتایج آن با نتایج حاصل از DP احتمالاتی (SDP) مقایسه شده است.

۲- یادگیری تقویتی (RL)

ایده اصلی استفاده از روش یادگیری تقویتی (Reinforcement Learning) یا RL که امروزه در کاربردهای مهندسی رواج یافته است، در واقع برگرفته شده از کار Turdnik (۱۹۱۱) است. وی رفتار حیوانها را از نگاه روانشناسانه مورد بررسی و مطالعه قرار داد. او اعتقاد داشت که اعمالی که توسط یک حیوان در یک موقعیت خاص منجر به نتایج خوبی شده است، تجربه مناسبی برای آن جاندار خواهد شد و اگر آن شرایط مجدداً، نیز تکرار شود آن حیوان تمایل بیشتری به انجام مجدد آن اعمال خواهد داشت. یادگیری تقویتی در واقع پیدا کردن بهترین نوع رفتار در موقعیتهای مختلف در یک سیستم پویا از طریق اندرکنش با محیط اطراف، بدون داشتن یک معلم مشخص است. RL راه حلی برای روند کنترل بهینه است که در آن مأمور یادگیری یا تصمیم گیرنده درصدد یافتن بهترین سیاست است. این سیاست بهینه، در واقع نگاشتی از حالتهای مختلف سیستم به بهترین تصمیمات قابل قبول برای دستیابی به توابع هدف بهینه مشخص برای افق برنامه ریزی است. بنابراین RL بسیار شبیه به روش معمول برنامه ریزی پویا می باشد. در RL سیاست بهینه را می توان به صورت مستقیم بر اساس شبیه سازی بدست آورد. اسامی دیگری برای RL موجود است که از جمله آنها می توان به « برنامه ریزی پویای مبتنی بر شبیه سازی» و «برنامه ریزی پویای عصبی» اشاره کرد.

چهار مولفه اصلی در RL وجود دارد: سیاست، پاداش، تابع ارزش و مدل. سیاست، نگاشتی از حالات به تصمیمهایی است که باید گرفته شود. پاداش، پاسخ فوری محیط به عمل انجام گرفته توسط مأمور یادگیری است. تابع ارزش که برای هر زوج از حالت – عمل (state-action) تعریف می شود، پاداش تجمعی از نقطه شروع RL است. به عبارت دیگر، تابع ارزش بر خلاف تابع پاداش، سود سیستم در هر زوج از حالت – عمل در یک دوره بلند مدت در RL می باشد. مدل در واقع یکی از مؤلفه های اختیاری در RL است که برای تعیین حالت بعد و پاداش بدست آمده بکار می رود. مدل برای حالتی که یادگیری تقویتی قرار است به صورت غیر بهنگام (off-line) انجام شود، اجباری است. شکل شماتیک مولفه های RL و نحوه ارتباط آنها با یکدیگر در زیر نشان داده شده است.



شکل ۱- *شماتیک الگوریتم یادگیری تقویتی*

در بیشتر الگوریتمهای RL، مقدار توابع ارزش همانند روش DP محاسبه می شود و به همین دلیل RL و DP شباهت زیادی به یکدیگر دارند. مقدار تابع ارزش بر اساس معادله بهینگی بلمن از رابطه زیر محاسبه می شود:

$$J^*(i)=\max_{a\in A(i)}\left[\bar{r}(i,a)+\lambda\sum_{j=1}^{|S|}p(i,a,j)J^*(j)\right]\quad \forall i\in S \qquad (۱)$$

که در آن $J^*(i)$، i امین مولفه بردار تابع ارزش مربوط به سیاست بهینه، $A(i)$ مجموعه اعمال قابل انجام در حالت i، $p(i,a,j)$ احتمال انتقال از حالت i به حالت j در نتیجه انجام عمل a، $\bar{r}(i,a)=\sum_{j=1}^{|S|}p(i,a,j)r(i,a,j)$ نشان دهنده مقدار بازگشت فوری مورد انتظار در حالت i ام در صورت انتخاب عمل a است که $r(i,a,j)$ نشان دهنده بازگشت فوری حاصله از انجام عمل a در حالت i و در نتیجه، رفتن به حالت j است. S، نشان دهنده مجموعه حالتها در زنجیره مارکوفی است و λ نشان دهنده ضریب تنزیل اقتصادی است. برای هر زوج (i,a) به عبارت داخل [ ] در معادله (۱) اصطلاحاً Q-Factor مربوط به آن زوج می گویند.

در DP برای هر مقدار حالت، یک تابع ارزش تخصیص داده می شود، در حالیکه در RL برای هر زوج حالت–عمل، یک تابع ارزش داریم. برای درک بهتر این موضوع فرض کنید که در یک مسئله تصمیم گیری مارکوفی، سه حالت و دو عمل ممکن در هر حالت داریم. در DP، بردار تابع ارزش $\vec{J}^*$ سه عضو همانند ذیل خواهد داشت:

$$\vec{J}^*=\{J^*(1),J^*(2),J^*(3)\}$$

در حالیکه در RL، ۶ مقدار Q-Factor وجود خواهد داشت؛ زیرا ۶ زوج حالت – عمل موجود است. بنابراین اگر $Q(i,a)$ نشان دهنده مقدار Q-Factor مربوط به حالت i و عمل a باشد:

$$\vec{Q}=\{Q(1,1),Q(1,2),Q(2,1),Q(2,2),Q(3,1),Q(3,2)\}$$

برای زوج حالت – عمل (i,a) مقدار Q-Factor متناظر از رابطه زیر محاسبه می شود.

$$Q(i,a)=\sum_{j=1}^{|S|}p(i,a,j)\left[r(i,a,j)+\lambda J^*(j)\right] \qquad (۲)$$

حال، با ترکیب روابط (۱) و (۲) خواهیم داشت:

$$J^*(i)=\max_{a\in A(i)}Q(i,a) \qquad (۳)$$

معادله (۳)، رابطه میان تابع ارزش یک حالت و Q-FACTOR های مرتبط با یک حالت را نشان می دهد. بنابراین اگر Q-FACTOR ها شناخته شده باشند، می توان تابع ارزش یک حالت را از رابطه (۳) بدست آورد. برای مثال، برای حالت i با دو عمل، اگر مقدار Q-FACTOR ها برابر با $Q(i,1)=95$ و $Q(i,2)=100$ باشد:

$$J^*(i)=\max\{95,100\}=100$$

با استفاده از معادله (۳) رابطه (۲) را می توان به صورت زیر نوشت:

$$Q(i,a)=\sum_{j=1}^{|S|}p(i,a,j)\left[r(i,a,j)+\lambda\max_{b\in A(j)}Q(j,b)\right] \qquad (۴)$$

رابطه (۴) در واقع به عنوان نسخه Q-FACTOR رابطه بهینگی بلمن قابل تعبیر است.

محاسبه مقادیر Q-FACTOR روش تکرار ارزش (Value Iteration Method)

الگوریتمی که در ذیل ارائه می شود معادل الگوریتم تکرار ارزش متداولی است که در DP مورد استفاده قرار می گیرد. این الگوریتم شامل مراحل زیر است.

گام اول: مقدار شماره گام زمانی k ام را برابر ۱ قرار داده و بردار دلخواه $\vec{Q}_0$ انتخاب می شود. برای مثال، برای تمام $i\in S$ و $a\in A(i)$:

$$Q^0(i,a)=0$$

مقدار ε (معیار توقف) بزرگتر از صفر قرار داده می شود.

گام دوم: برای هر $i\in S$ تابع ارزش به شکل زیر محاسبه می شود:

$$Q^{k+1}(i) \leftarrow \sum_{j=1}^{|S|} p(i,a,j)\left[r(i,a,j) + \lambda \max_{b \in A(j)} Q^k(j,b) \right]$$

گام سوم: برای هر $i \in S$ تابع ارزش بهینه به شکل زیر محاسبه می شود:

$$J^{k+1}(i) = \max_{a \in A(i)} Q^{k+1}(i,a), \quad J^k(i) = \max_{a \in A(i)} Q^{k+1}(i,a)$$

در ادامه اگر $\left\| \vec{J}^{k+1} - \vec{J}^k \right\| < \varepsilon(1-\lambda)/2\lambda$ به گام ۴ رفته ودر غیر این صورت k به اندازه یک واحد افزایش می یابد و کنترل به گام ۲ برمی گردد.

گام چهارم: برای تمام $i \in S$ تصمیم بهینه بدین صورت محاسبه می شود: $d(i) \in \arg\max_{b \in A(j)} Q(i,b)$

$\hat{d}$ سیاست ε – بهینه نامیده می شود. در صورت ارضای رابطه شامل عبارت ε در گام سوم، برای تمامی حالتها، الگوریتم متوقف می گردد. معادله گام ۲ از رابطه(۴) استخراج شده است. تشخیص معادل بودن این الگوریتم با روش تکرار ارزش معمول چندان دشوار نیست. به جای برآورد کردن مقدار تابع ارزش، این الگوریتم Q-FACTOR ها را برآورد می کند. در RL نیز Q-FACTOR ها برآورد می شوند ولی الگوریتم به روز رسانی معادله کمی متفاوت با رابطه ای است که در گام ۲ بیان شد.

الگوریتم رابینز – مونرو

الگوریتم رابینز–مونرو، الگوریتمی قدیمی از دهه پنجاه میلادی است که بوسیله آن میانگین یک متغیر تصادفی از روی نمونه های تولید شده از آن برآورد می شود. میانگین یک متغیر تصادفی را می توان با میانگین گیری مستقیم بدست آورد. فرض کنید i امین نمونه از متغیر تصادفی تولید شده X، s^i باشد و مقدار امید ریاضی این نمونه ها $E(X)$ باشد. مقدار میانگین حاصله از رابطه $\sum_{i=1}^{n} s^i \Big/ n$ با رفتن n به سمت بی نهایت با احتمال ۱ به مقدار واقعی میانگین همگرا می شود. (این مسئله از قانون مهم اعداد بزرگ به دست می آید). به عبارت دیگر، با احتمال ۱، $E(X) = \lim_{n \to \infty} \sum_{i=1}^{n} s^i \Big/ n$.

می توان از این رابطه، الگوریتم رابینز–مونرو را استخراج کرد. فرض کنید مقدار برآورد شده X در n امین تکرار – بعد از تولید n نمونه – X^n باشد. بنابراین $X^n = \sum_{i=1}^{n} s^i \Big/ n$. در نتیجه:

$$X^{n+1} = \frac{\sum_{i=1}^{n+1} s^i}{n+1} = \frac{\sum_{i=1}^{n} s^i + s^{n+1}}{n+1} = \frac{X^n n + s^{n+1}}{n+1} = \frac{X^n n + X^n - X^n + s^{n+1}}{n+1} = \frac{X^n (n+1)}{n+1} - \frac{X^n}{n+1} + \frac{s^{n+1}}{n+1}$$

$$= X^n - \frac{X^n}{n+1} + \frac{s^{n+1}}{n+1} = (1-\alpha^{n+1})X^n + \alpha^{n+1} s^{n+1} \quad if \quad \alpha^{n+1} = 1/n+1$$

در نهایت خواهیم داشت:

$$X^{n+1} = (1-\alpha^{n+1})X^n + \alpha^{n+1} s^{n+1} \qquad (۵)$$

اگر α^{n+1} برابر با $1/(n+1)$ باشد، این روش معادل میانگین گیری ساده خواهد بود.

الگوریتم رابینز–مونرو و برآورد Q-FACTOR ها

الگوریتم رابینز–مونرو را می توان برای برآورد Q-FACTOR ها بکار برد. هر Q-FACTOR در واقع میانگین یک متغیر تصادفی است. معادله بهینگی بلمن به شکل زیر نیز قابل بیان است:

$$Q(i,a) = \sum_{j=1}^{S} p(i,a,j)\left[r(i,a,j) + \lambda \max_{b \in A(j)} Q(j,b) \right] = E\left[r(i,a,j) + \lambda \max_{b \in A(j)} Q(j,b) \right] \qquad (۶)$$

که E عملگر امید ریاضی و مقدار داخل کروشه در معادله (۶) یک متغیر تصادفی است. بنابراین اگر نمونه هایی از متغیر تصادفی از طریق شبیه سازی به دست آورده شود، می توان به جای بهره گیری از معادله (۶) برای برآورد Q-FACTOR ها (همانطور که در نسخه Q-FACTOR روش تکرار ارزش ها آمده است) می توان از الگوی رابینز–مونرو برای ارزیابی Q-FACTOR ها استفاده کرد. با استفاده از الگوریتم رابینز–مونرو (رابطه ۵)، رابطه (۶) برای هر زوج حالت–عمل به صورت زیر خواهد بود:

$$Q^{n+1}(i,a) \leftarrow (1-\alpha^{n+1})Q^n(i,a) + \alpha^{n+1}\left[r(i,a,j) + \lambda \max_{b\in A(j)} Q^n(j,b)\right] \qquad (۷)$$

جالب ترین مسئله در مورد این الگوریتم این است که در این روش احتمال انتقال حالت وجود ندارد. به عبارت دیگر اگر این الگوریتم بتواند Q-FACTOR های بهینه را در شبیه ساز تولید کند، لازم نیست که احتمالات انتقال نهفته در مدل مارکوفی محاسبه شود و تنها به یک شبیه ساز برای سیستم نیاز است. مکانیسم نشان داده شده در رابطه (۷) باعث جلوگیری از تشکیل ماتریس احتمال انتقال در RL می شود. نیاز به برآورد این ماتریس در مدلهای بزرگ مقیاس با تعداد متغیرهای تصادفی قابل ملاحظه، تحت عنوان بلای مدلسازی (curse of modeling) شناخته می شود که RL این مشکل را حل می کند.

روش تکرار ارزش در RL

فرض کنید که شبیه ساز عمل a را در مرحله i انتخاب کند و سیستم در نتیجه عمل i به حالت j برود. در طول دوره زمانی که شبیه ساز از حالت i به j می رود، الگوریتم RL از اطلاعات درون شبیه ساز استفاده می کند. این اطلاعات شامل $r(i,a,j)$ است که بازگشت حاصله از رفتن از حالت i به حالت j در نتیجه انجام عمل a است. وقتی شبیه ساز به حالت j می رسد از $r(i,a,j)$ برای به روز رسانی مقدار $Q(i,a)$ اقدام می کند. بنابراین به روز رسانی، پس از انتقال کامل از حالت i به حالت j رخ می دهد. با توجه به مطالب ذکر شده، روش RL شامل مراحل زیر است.

گام ۱: شماره تکرار الگوریتم (k=1) و سپس تمام مقادیر Q-FACTOR را برابر صفر قرار می دهیم. به عبارت دیگر برای همه (l,u) که $l \in S$ و $u \in A$ قرار می دهیم: $Q(l,u) \leftarrow 0$

گام ۲: فرض کنید سیستم در حالت i باشد. عمل a را با احتمال $1/A(i)$ (که $A(i)$ مجموعه تصمیمات ممکن در حالت i ام است) انتخاب کرده و سیستم شبیه سازی می شود.

گام ۳: فرض کنید که حالت بعدی سیستم برابر با j باشد. $r(i,a,j)$ بازگشت فوری بدست آمده در انتقال از حالت i به حالت j، تحت عمل a می باشد که توسط شبیه ساز به دست می آید. در این شرایط $V(i,a)$ که بیانگر تعداد دفعاتی است که هر زوج حالت–عمل (i,a) مورد امتحان قرار می گیرد را یک واحد افزایش می دهیم. این مقدار همان n در رابطه (۷) می باشد که به فاکتور برخورد (Visit Factor) موسوم است. k را به اندازه یک واحد افزایش داده و مقدار $\alpha = 1/V(i,a)$ محاسبه می شود.

گام ۴: $Q(i,a)$ با استفاده از معادله زیر به روز رسانی می گردد:

$$Q(i,a) \leftarrow (1-\alpha)Q(i,a) + \alpha\left[r(i,a,j) + \lambda \max_{b\in A(j)} Q(j,b)\right]$$

گام ۵: اگر $k < k_{\max}$، $i \leftarrow j$ و به گام ۲ رفته، وگرنه گام ۶ انجام می شود.

گام ۶: برای هر $l \in S$ انتخاب می کنیم: $d(l) \in \arg\max_{b\in A(l)} Q(l,b)$

سیاست (راه حل) بدست آمده توسط الگوریتم، بردار $\hat{d}$ خواهد بود. گامهای مذکور تا $k = k_{\max}$ تکرار می شود.

روشهای انتخاب عمل (Action Selection Methods)

در بخش قبل و پس از توضیح روش الگوریتم رابینز – مونرو در این خصوص توضیح داده شد که برای میانگین گیری، تمام اعمال با احتمال مساوی شانس انتخاب شدن را دارند. اگر تمام اعمال برای تمام حالتهای ممکن به تعداد دفعات زیادی تکرار شود، چند مشکل پدید می آید. اول آن که اگر بخواهیم تمام حالتها و اعمال به تعداد زیاد (مناسب) تکرار شوند تا میانگین گیری به واقعیت نزدیک باشد، تعداد گامهای زمانی زیادی در شبیه سازی باید انجام شود. دوم اینکه با این کار، عملاً روش انتخاب سیاست بهینه بر مبنای تکرار بلند مدت شبیه سازی خواهد بود و هیچ تلاشی در جهت هوشمند کردن انتخاب اعمال مناسب در هر حالت وجود نخواهد داشت. یکی از ساده ترین روشها برای هوشمند کردن انتخاب اعمال مناسب، انتخاب کردن عملی است که دارای بیشترین میزان ارزش برآورد شده تا به حال است. یعنی به صورت کاملاً « حریصانه » (greedy)، در t امین گام زمانی، عمل a^* به نحوی انتخاب شود که $Q_t(a^*)=\max_a Q_t(a)$. در این روش همواره از اطلاعات کنونی برای حداکثر کردن پاداش فوری استفاده می شود و توجهی به اعمالی که دارای ارزش کنونی کمتری هستند و ممکن است واقعاً دارای ارزش تجمعی بیشتری باشند نمی شود. روش جایگزین برای روش «حریصانه» این است که در بیشتر اوقات به صورت حریصانه اعمال را انتخاب کنیم، ولی گاهی اوقات، با احتمالی کوچک، ε، عملی دیگر را به صورت تصادفی و بدون توجه به توابع ارزش محاسبه شده انتخاب کنیم. این روشها که بسیار نزدیک به روشهای حریصانه هستند را روشهای ε –حریصانه $(\varepsilon-greedy)$ می نامند. مزیت این روش این است که با افزایش تعداد تکرارها، هر از چند گاهی اعمال دیگری نیز غیر از اعمال حریصانه انتخاب می شوند و این به همگرایی ارزش تخمین زده شده به ارزش واقعی کمک می کند. زیرا ممکن است این اعمال که در کوتاه مدت دارای ارزش کمتری هستند، در ادامه ارزشی بیشتری از اعمال حریصانه داشته باشند.

ارزیابی عملکرد

برای تعیین کیفیت راه حل ارائه شده توسط مسئله، باید بعد از بدست آوردن سیاستهای بهینه، مسئله را شبیه سازی نمود. این شبیه سازی، با استفاده از سیاست یاد گرفته شده در آخرین گام الگوریتم انجام می شود. برای پیدا کردن مقدار تابع ارزش حالت i، باید شبیه سازی را از حالت i شروع کرد و پاداشهای بدست آمده در طول یک دوره شبیه سازی را محاسبه نمود. این کار باید در چندین تکرار انجام گردیده و از جوابها میانگین گیری شود.

۳- مطالعه موردی

در ابتدا برای بررسی صحت عملکرد الگوریتم یادگیری تقویتی، از مثالی فرضی استفاده شده که توسط Loucks et al.[4] در زمینه کاربرد برنامه ریزی پویای استوکستیک در حل مسئله بهینه سازی بهره برداری از یک مخزن به کار رفته است. آنها مقدار حجم مخزن در هر فصل را به عنوان متغیر حالت و میزان رهاسازی آب در هر فصل را به عنوان متغیر تصمیم در نظر گرفتند. پارامتر تصادفی مسئله، میزان دبی ورودی به مخزن است. تابع هدف مسئله، حداقل کردن مجموع مربعات فاصله میزان رهاسازی آب و حجم مخزن به ترتیب با میزان نیاز و حجم مطلوب مخزن در هر ماه است.

$$\min \quad \sum_{t=1}^{T}\left(S_t-S^*\right)^2+\left(R_t-R_t^*\right)^2$$

که S_t حجم مخزن در فصل t، S^* حجم مطلوب مخزن، R_t میزان رهاسازی آب در فصل t و R^* میزان رهاسازی آب مطلوب (نیاز پایین دست) می باشد. به دلیل اینکه مسئله تنها دارای ۴ کلاس برای متغیر حالت و ۴ کلاس برای متغیر تصمیم است، برای مقایسه بهتر الگوریتمهای SDP و RL ابتدا روش RL برای این مسئله ساده بکار گرفته شد و نتایج با جواب حاصله از مدل SDP مقایسه شد. نتایج حاصل کاملا با نتایج SDP مطابقت داشت. در ادامه و پس از اطمینان از

صحت عملکرد الگوریتم، در یک مثال واقعی بزرگتر از این الگوریتم استفاده گردید. مطالعه موردی بر روی سد مخزنی چراغ ویس انجام شده است. این سد در نزدیکی شهر سقز در استان کردستان در دست احداث است. متغیرهای حالت و تصمیم همانند مسئله فوق تعریف شدند. تنها تفاوت در نحوه فرمول بندی تابع هدف مسئله است که حداقل کردن مجموع وزنی مربعات فاصله میزان رهاسازی آب و حجم مخزن به ترتیب با میزان نیاز و حجم مطلوب مخزن در هر ماه می باشد.

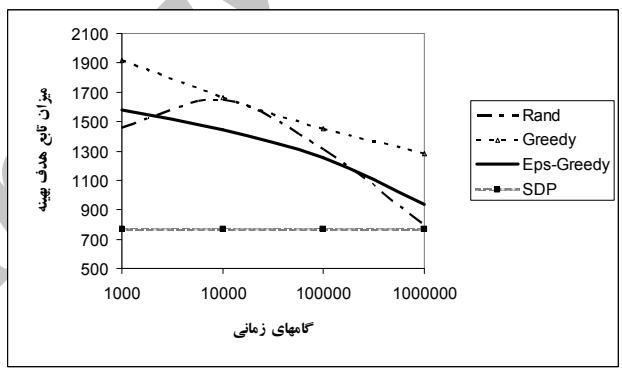
$$\min \quad \sum_{t=1}^{T}\left(S_t - S^*\right)^2 + 2*\left(R_t - R_t^*\right)^2$$

حجم حداقل بهره برداری و حجم مخزن در رقوم نرمال به ترتیب برابر ۱۷ و ۷۰ م.م.م. و حجم مطلوب مخزن برابر ۵۳ م.م.م. است. حجم ذخیره مخزن با فواصل ۱ م.م.م. گسسته سازی شده است. مقادیر گسسته متغیر تصمیم (میزان آب رهاسازی شده در هر ماه)در ۰ م.م.م. تا ۶۰ م.م.م.، با فواصل ۰/۵ م.م.م. در نظر گرفته شده است. با استفاده از آمار آبدهی ۳۵ ساله این رودخانه، ۱۰۰۰ داده تصادفی مستقل با فرض توزیع لوگ نرمال تولید و سپس احتمالات ماهانه جریان ورودی محاسبه گردید. مقادیر نیازهای ماهانه در جدول ذیل ارائه شده است:

ماه	مهر	آبان	آذر	دی	بهمن	اسفند	فروردین	اردیبهشت	خرداد	تیر	مرداد	شهریور
نیاز (م.م.م)	۵/۴۳۲	۴/۰۵۳	۳/۵۱۲	۳/۰۹۰	۳/۳۰۰	۳/۴۶۰	۴/۰۰۷	۶/۱۷۰	۱۱/۷۷۰	۱۴/۴۸۸	۱۲/۳۳۳	۸/۳۵۱

۴- بررسی و تحلیل نتایج

از مدل SDP و نیز مدل RL برای بدست آوردن سیاست بهینه استفاده شد و پس از هر اجرا، شبیه سازی بهره برداری مخزن با توجه به سیاست حاصله انجام شد و میزان تابع هدف حاصله بدست آمد. برای انتخاب عمل، سه روش انتخاب تصادفی اعمال (الگوریتم اصلی رابینز-مونرو)، انتخاب اعمال حریصانه و انتخاب اعمال به صورت ε -حریصانه، با مقدار ε برابر ۰/۱ ، مورد بررسی قرار گرفت. مدلهای RL چندین بار اجرا شد و نتایج از میانگین گیری جوابها حاصل شده و مقایسه شد. مقایسه جوابها در شکل (۲) نشان داده شده است.



شکل ۲- نتایج *شبیه سازی بهره برداری از مخزن سد چراغ ویس با استفاده از الگوریتم RL*

در شکل (۲)، خط افقی نشان دهنده نتیجه مدل SDP است. روشهای مختلف RL برای تعداد گامهای زمانی مختلف اجرا شده اند و نتایج آنها به صورت نیمه لگاریتمی ترسیم شده است. روشهای ε -حریصانه با سرعت مناسبی به سمت جواب بهینه همگرا می شوند و این روند با افزایش تعداد گامهای زمانی همچنان ادامه می یابد. در انتخاب تصادفی اعمال که طبق الگوریتم اصلی رابینز-مونرو انجام می شود، در ابتدا دچار واگرایی می شود و سپس مجددا به سمت جواب بهینه همگرا می شود. علت واگرایی را در این حالت می توان با مقایسه الگوریتم SDP و RL بررسی کرد. در SDP معمولاً از یک حالت شروع کرده و تمام حالتها به ترتیب مورد بررسی قرار می گیرد ولی در شبیه ساز نمی توان تضمین کرد که ترتیب رخ دادن حالتها منظم باشد. برای مثال در DP، حالتها به صورت ۱، ۲، ۳، ۱، ۲، ۳، ... رخ می دهند اما در RL ممکن است ترتیبی

همانند زیر رخ دهد: ۱، ۳، ۱، ۲، ۲، ۳، ۱، ... یعنی ابتدا یکی از QF ها در حالت ۱ به روز می شود، سپس به حالت ۳ می رسیم، یک QF را به روز کرده و به حالت ۱ بر می گردیم، سپس یک QF را در حالت ۱ به روز می کنیم و به همین ترتیب ادامه می دهیم. به روز کردن QF در یک حالت، طبق رابطه (۷) نیازمند داشتن QF از حالت دیگری است و آخرین برآورد از QF ها تاکنون، مورد استفاده قرار می گیرد. در این نوع به روز رسانی، در هر زمان امکان دارد یک QF بسیار بیشتر از بقیه QF ها به روز رسانی شود. نتیجه این است که به روز رسانی بسیار نامنظم خواهد شد و این بی نظمی موجب ایجاد خطا می شود. روشهای مختلفی برای کاهش اثر این خطا پیشنهاد شده است [۵]. با افزایش تعداد گامهای زمانی، این روش جوابهای بهتری را در مقایسه با دو روش دیگر نشان داده است زیرا در مسئله بهره برداری از یک مخزن، تعداد زیادی از زوجهای حالت-عمل در واقعیت رخ می دهند، و در روش انتخاب تصادفی اعمال نیز با افزایش تعداد تکرارها، چندین بار با زوجهای حالت و عمل مختلف به صورت تصادفی روبرو می شویم. در حالیکه انتخاب حریصانه یا ε - حریصانه، درصد بالایی از این زوجهای حالت-عمل را در نظر نمی گیرد.استفاده از دو روش اخیر در مسائلی که بسیاری از زوجهای حالت عمل در واقعیت کمتر رخ می دهد، نتایج بهتری خواهد داشت.

۵- خلاصه و نتیجه گیری

روشهای بهینه سازی مبتنی بر شبیه سازی در کارهای اخیر مورد توجه بسیاری قرار گرفته است. روشهای یادگیری تقویتی که گاهی از آنها به عنوان روش برنامه ریزی پویای مبتنی بر شبیه سازی یاد می شود، بدون نیاز به ماتریس احتمال انتقال و با استفاده از یک شبیه ساز که داده ها در آن تولید شده و شبیه سازی به صورت مستقیم و رو به جلو انجام می شود، با استفاده از تجربیات مثبت و منفی حاصله، مقدار توابع ارزش را برای حالتها و تصمیمهای مختلف به روز می کند. مهمترین مزیت روش RL این است که چون مبتنی بر شبیه سازی می باشد، آن دسته از زوجهای حالت-عمل که در واقعیت کمتر با آنها روبرو می شویم، در RL کمتر مورد توجه و به روز رسانی قرار می گیرند. به همین دلیل برای این حالتها پیدا کردن سیاست بهینه عملاً لزومی ندارد. در حالیکه در SDP برای تمام حالتها، اعم از حالتهایی که سیستم واقعی تحت عملکرد مطلوب خود زیاد با آنها روبرو می شود و یا حالتهایی که احتمال رخ دادن آنها کم است، سیاست بهینه محاسبه می شود. بنابراین در مسائل بزرگ مقیاس، صرفه جویی زیادی در زمان مورد نیاز برای محاسبات خواهیم داشت. در این مقاله روش RL در مسئله بهینه سازی بهره برداری از سد چراغ ویس بکار گرفته شده و نتایج حاصل با مدل SDP کلاسیک مقایسه شد. نتایج نشان می دهد که روش RL می تواند به شکل موثری به جوابهای نزدیک به جواب مدل SDP همگرا شود. مزیت عمده استفاده از این روش در سیستمهای چند مخزنه می باشد که مبنای ادامه این مطالعه خواهد بود.

۶- مراجع

[1] Yakowitz, S. (1982), Dynamic programming applications in water resources. Water Resources Research, 18 (4), 673-696.

[2]Mousavi,S.J., Karamouz, M. (2003), Computational Improvement for Dynamic Programming Models by Diagnosing Infeasible Storage Combinations, Journal of Advances in Water Resources, Elsevier, Vol. 26, 851-859.

[3] Mousavi, S. J., Zanoosi,A., and Afshar, A. (2004) ,Optimization and Simulation of a Multiple Reservoir System Operation, Journal of Water Supply: Research and Technology (AQUA), 56(6),409-424

[4] Loucks, D. P., Stedinger. J. R. and Haith, D. A. (1981). Water resources system planning and analysis, Prentice Hall, Englewood Cliffs, New York

[5] Sutton R. S., and Barto A. G. 1998, Reinforcement Learning. The MIT press, Cambridge, Massachusetts.

[6] Gosavi A. 2003, Simulation-based optimization: parametric optimization techniques and reinforcement learning. Kluwer academic publisher, Norwel Massachusetts.

کارگاه های آموزشی مرکز اطلاعات علمی جهاد دانشگاهی

کارگاه آنلاین آشنایی با پایگاه های اطلاعات علمی بین المللی و ترفند های جستجو

کارگاه آنلاین
پروپوزال نویسی و پایان نامه نویسی

کارگاه آنلاین
بررسی مقابله ای متون (مقدماتی)