نشریه علمی ـ ترویجی محاسبات نرم
شماره‌ی سوم/ بهار و تابستان ۱۳۹۲/ صفحه ۱۲ـ۲۱

تاریخ دریافت مقاله: دی ۱۳۹۱

تاریخ پذیرش مقاله: دی ۱۳۹۲

پیاده‌سازی یک مولد فضای حالت نمادین برای توصیف سیسستم‌های تصادفی گسسته‌رخداد در چارچوب PDETool

رضا فتحی[۱]، محمد عبدالهی ازگمی[۲]

[۱]کارشناس ارشد نرم‌افزار، دانشکده مهندسی کامپیوتر، دانشگاه علم و صنعت، تهران، ایران

fathi_reza@comp.iust.ac.ir

[۲]دانشیار گروه نرم‌افزار، دانشکده مهندسی کامپیوتر، دانشگاه علم و صنعت، تهران، ایران

azgomi@iust.ac.ir

چکیده: بررسی مدل، یکی از مهم‌ترین روش‌های درستی‌یابی سیستم‌هاست. یکی از مشکلات بررسی مدل، تولید فضای حالت است و معمولاً به‌دلیل بزرگ‌بودن فضای حالت، مشکل انفجار حالت پیش می‌آید. دلیل انفجار فضای حالت، رشد نمایی اندازه فضای حالت با تعداد متغیرهای مدل است. از راهکارهای غلبه بر این مشکل، نگهداری فضای حالت به‌صورت ضمنی به‌جای نگهداری صریح آن‌هاست. نگهداری فضای حالت به‌صورت نمادین، هزینه‌ی ذخیره و پردازش فضای حالت را به مرتبه ذخیره‌سازی و پردازش گراف‌ها کاهش می‌دهد.

در این مقاله، پیاده‌سازی نوینی برای تولید فضای حالت نمادین از توصیف SDES در ابزار PDETool ارائه شده است. این ابزار با هدف فراهم‌سازی چارچوبی یکپارچه برای مدل‌سازی و تحلیل سیستم‌ها طراحی شده است و برمبنای موتور شبیه‌سازی SimGine و توصیف SDES عمل می‌کند. درواقع، از توصیف سیستم‌های تصادفی گسسته رخداد (SDES) به‌عنوان صورت‌بندی رابط استفاده می‌گردد که انواع مدل‌های صوری به آن تبدیل می‌شوند. در این روش، با استفاده از تولید فضای حالت نمادین به کمک گراف تصمیم دودویی مرتب کاهش‌یافته (ROBDD)، فضای حالت بسیار بزرگتری را می‌توان تولید و مدیریت کرد؛ درنتیجه با استفاده از این روش، تحمل‌پذیری بالاتری برای ابزار PDETool در مقابل مشکل انفجار حالت ایجاد شده است.

کلمات کلیدی: انفجار فضای حالت، گراف تصمیم دودویی مرتب کاهش‌یافته (ROBDD)، تولید فضای حالت نمادین، توصیف SDES، ابزار PDETool.

۱. مقدمه

بررسی مدل، پیش‌نیاز اساسی در تولید سیستم‌های مدرن است تا بتوان قبل از ایجاد سیستم، برآوردهای لازم از درستی، کارآیی، نقاط بحرانی و شکست سیستم به‌دست آورد. از طرفی، تولید فضای حالت که همه رفتارهای سیستم را پوشش دهد، پیش‌نیازی اساسی برای بررسی مدل‌هاست؛ ولی تولید فضای حالت برای مدل‌های واقعی که اندازه‌ی بزرگی دارند با مشکل انفجار فضای حالت روبه‌رو می‌شود. این مشکل به‌دلیل بزرگی بیش از اندازه فضای حالت پیش می‌آید.

به عبارت دیگر، با بزرگ‌ترشدن فضای حالت، علاوه بر مشکل ذخیره و نگهداری آن، با مشکل پیمایش مؤثر و کارای فضای حالت نیز روبه‌رو خواهیم شد؛ بنابراین، نیاز است که الگوریتم‌های مناسب پیمایش و عملیات روی فضای حالت، معرفی شوند. از طرفی، انتخاب الگوریتم‌های پیمایش به ساختمان داده‌ی مورد استفاده برای ذخیره‌سازی فضای حالت نیز بستگی دارد؛ بنابراین، در انتخاب ساختمان داده مناسب برای ذخیره فضای حالت باید دو عامل مهم را درنظر داشت. عامل اول فشرده‌بودن فضای حالت است. ساختمان داده انتخابی باید بتواند فضای حالت بسیار بزرگی را به‌صورت فشرده و با حداقل حافظه ممکن نگهداری کند. عامل دوم امکان پیمایش و انجام عملیات مورد نیاز با مرتبه زمانی پایین است تا امکان پیمایش با سرعت مناسب روی فضای حالت امکان‌پذیر شود. الگوریتم‌های کند یا انتخاب ساختمان داده‌ای که سرعت پیمایش و قدرت فشرده‌سازی مناسبی ندارد، تولید فضای حالت و بررسی آن را کند خواهد کرد؛ درنتیجه، بررسی مدل، بیش از حد متعارف زمان خواهد برد.

۲. کارهای مرتبط

فضاهای حالت به یکی از دو روش صریح یا نمادین نمایش داده می‌شوند. از مهم‌ترین روش‌های صریح، روش ساختمان داده‌های چندسطحی مثل آرایه‌های چندسطحی و ادغام‌بردارهای بیتی را می‌توان نام برد. ساختمان داده‌های چندسطحی از ساختار سیستم تحت نمایش، مثل شکستن یک شبکه پتری به زیرشبکه‌ها، برای نمایش فضای حالت استفاده می‌کنند [۱]. روش ادغام‌بردار بیتی تلاش می‌کند تا بردارهای بیتی مشترک را ادغام نماید تا بتوان فضای حالت بزرگ‌تری را نگهداری و نمایش داد [۲]. این ساختمان داده‌ها برای مدل‌های کوچک و تولید فضای حالت آن‌ها خوب هستند. برای اجتناب از مواجه‌شدن با مشکل انفجار فضای حالت در مواجهه با مدل‌های بزرگ‌تر، سه تکنیک عمده کوچک‌سازی ترکیبی [۳]، ترتیب جزئی [٤] و استفاده از تقارن سیستم [٥] در نمایش صریح فضای حالت توسعه داده شده است.

تکنیک‌های تولید فضای حالت نمادین به‌طور سنتی روی سیستم‌های سخت‌افزاری تمرکز کرده بودند. این تکنیک‌ها را پاستر و همکاران برای اولین‌بار، برای شبکه‌های پتری به‌کار گرفتند [٦]. تکنیک‌های نمادین برپایه‌ی نمودارهای تصمیم، مثل نمودارهای تصمیم دودویی، به‌جای تلاش برای ذخیره‌ی حالت‌های کمتر تلاش می‌کنند که فضای حالت را با فشردگی بیشتری نگهداری کنند [۷]. فرم عمومی‌تر دیگرنمودارهای تصمیم، نمودار تصمیم چندمقداری [۸] است که در آن هر گره مقدار اطلاعات بیشتری می‌تواند نگهداری کند. در [۹,۱۰] با یک پیاده‌سازی مؤثر از نمودارهای تصمیم چندمقداری، فضای حالت بزرگ‌تری تولید و نگهداری شده است.

همچنین در این قسمت، به بررسی برخی از ابزارهای مهم در تولید فضای حالت و بررسی مدل پرداخته می‌شود. تقریباً همه‌ی ابزارهای مهم و شناخته‌شده در زمینه‌ی بررسی مدل، تولید فضای حالت نمادین را پشتیبانی می‌کنند و عمدتاً در روش‌های پیاده‌سازی و انواع نمودارهای تصمیم که پشتیبانی می‌کنند، با هم تفاوت دارند. برخی از این ابزارها، مانند ابزار Meddly [۱۱] سعی کرده‌اند کتابخانه‌ای برای تولید فضای حالت نمادین در اختیار کاربران قرار دهند. Meddly سعی کرده است انواع تبدیل‌ها از یک فرم نمودار تصمیم به فرم دیگری از نمودار تصمیم را فراهم کند. ابزار Möbius [۱۲] جزو ابزارهای معروف در زمینه‌ی مدل‌سازی و ارزیابی مدل‌ها محسوب می‌شود. این ابزار ابتدا از ساختمان داده جدول

درهم‌سازی استفاده می‌کرد؛ ولی با بزرگ‌شدن مدل‌ها و نیاز به نگهداری فضای حالت بیشتر برای افزایش تحمل‌پذیری در مقابل مشکل انفجار فضای حالت، از ساختمان داده نمودارهای تصمیم استفاده کرد. ابزار دیگر، PRISM [۱۳] از کتابخانه CUDD [۱٤] برای تولید فضای حالت استفاده می‌کند که یک کتابخانه در دسترس عموم BDD/MTBDD است.

۳. مفاهیم پایه

توصیف[1] SDES مدل یک روش صوری چندگانه است که امکان تبدیل مدل‌های تصادفی گسسته رخداد دیگر، مثل شبکه‌های پتری یا شبکه‌های فعالیت تصادفی به این توصیف وجود دارد تا بدین وسیله بتوان الگوریتم‌هایی را که روی این روش صوری اعمال می‌شود، روی همه‌ی آن‌ها اعمال کرد. توضیحی مختصر از توصیف SDES در بخش ۲-۱ و در بخش ۲-۲ نیز مفاهیم مربوط به نمودار تصمیم دودویی مرتب کاهش‌یافته که ساختمان داده مورد استفاده برای نگهداری فضای حالت نمادین است، آورده می‌شود.

۳-۱. توصیف واحد SDES

سیستم گسسته‌رخداد، سیستمی است که در طول یک بازه‌ی زمانی در یک حالت می‌ماند و به‌محض بروز رخدادی اتمیک، حالت سیستم فوراً تغییر می‌کند؛ به عبارت دیگر، حالت سیستم در فاصله‌ی زمانی دو رخداد متوالی تغییر نمی‌کند که این اصلی‌ترین تفاوت مدل‌های گسسته‌رخداد با مدل‌های پویای پیوسته‌رخداد است. توصیف SDES توصیفی واحد برای سیستم‌های گسسته‌رخداد است که مدل‌های بسیاری شامل اتوماتا، شبکه‌های صف، شبکه‌های پتری تصادفی و غیره می‌توانند براساس آن توصیف شود و به عبارتی، ترجمه‌ای از این مدل‌ها به SDES وجود دارد.

در [15]، یک سیستم تصادفی گسسته‌رخداد SDES به‌صورت یک چهارتایی فرمول ۱ تعریف شده است:

$$SDES = (SV^*, A^*, S^*, RV^*) \quad (۱)$$

که در آن:

- SV^* مجموعه‌ای محدود از متغیرهای حالت تعریف شده است.
- A^* مجموعه‌ای محدود از گذارها تعریف شده است.
- S^* تابعی است که یک ترتیب خاص به هریک از متغیرهای حالت SV^* و متغیرهای گذار Var^* موجود در یک مدل انتساب می‌کند.
- RV^* به‌عنوان متغیرهای پاداش، متناسب با ارزیابی‌های کیفی مدل تعریف شده است.

با انتساب مقادیر به هریک از متغیرهای حالت، همه حالت های ممکن از یک مدل SDES مشخص به شکل Σ در فرمول (۲) تعریف می‌شود:

$$\sum = \prod_{sv \in SV^*} S^*(sv) \quad (۲)$$

متغیر حالت sv_i، به عنصری غیرفعال از SDES، مثل مکانی از یک شبکه پتری یا صفی از مدل صف اطلاق می‌شود که تعریف صوری آن در فرمول (۳) آمده است.

$$sv = (Cond^*, Val_0{}^*) \quad (۳)$$
$$Cond^*: SV^* \times \sum \to \mathbb{B}$$

۳-۲. نمودارهای تصمیم دودویی کاهش‌یافته

نمودارهای تصمیم دودویی، گراف‌های ریشه‌دار، جهت‌دار و بدون دور هستند که نمایشی طبیعی برای توابع محدود ارائه می‌دهند. این نمودارها اولین‌بار در [۱٦] و [۱۷] مطرح شدند؛ ولی در ادامه توسط برآیان در [۱۸] عمومیت پیدا کردند؛ کسی که ساختار داده‌ای آن را بهبود بخشید و الگوریتم‌هایی برای پردازش مؤثر آن‌ها ارائه کرد. یک نمودار تصمیم دودویی با مجموعه محدودی از متغیرهای دودویی مرتب‌شده ساخته می‌شود تا تابع دودویی تعریف‌شده روی این متغیرها را نمایش دهد. فرض کنید تابع نمایش داده شده توسط درخت تصمیم دودویی روی k متغیر $x_k > \ldots > x_1$ را به‌صورت $fB: \mathbb{B}^K \to \mathbb{B}$ نمایش داده شود.

1. Stochastic Discrete Event Systems

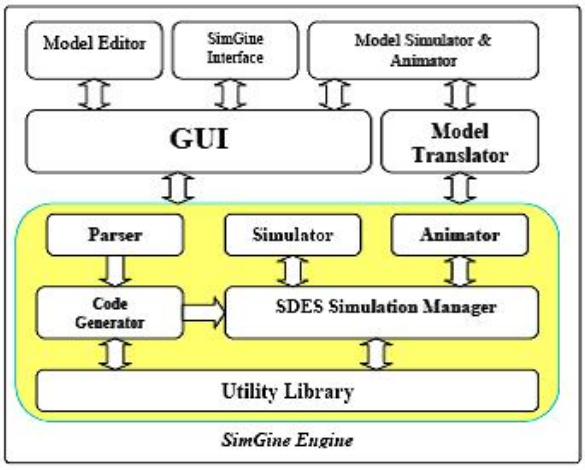
تعریف: یک نمودار تصمیم دودویی مرتب[1]، کاهش‌یافته نامیده می‌شود، اگر برای هر جفت (v,w) در بین گره‌های موجود در نمودار تصمیم دودویی که $v \neq w$ باشد، شرط $f_v \neq f_w$ برقرار باشد [۱۹].

بنابراین، در یک OBDD کاهش‌یافته (ROBDD[2]) هر مشتق[3] مرتب‌شده دقیقاً با یک گره نمایش داده می‌شود. این کلیدی‌ترین خصوصیت برای اثبات جامع[4] و متعارف[5] بودن ساختمان داده ROBDD برای نمایش توابع گذار است. جامع‌بودن بدین معنی است که هر تابع گذار می‌تواند با یک OBDD نمایش داده شود و استانداردبودن بدین معنی است که هر دو OBDD برای یک تابع مشابه خواهد بود، اگر گره‌ها را دوباره نام‌گذاری کنیم؛ به عبارتی دیگر، برای یک تابع همیشه یک OBDD کاهش‌یافته حاصل خواهد شد [۲۰،۲۱].

٤. ابزار PDETooL

ابزار PDETool [۲۲،۲۳] چارچوبی برای ارزیابی مدل‌های گسسته‌رخداد تصادفی است که برمبنای موتور شبیه‌سازی SimGine [۲٤] و توصیف SDES عمل می‌کند. این ابزار امکاناتی برای ساخت، شبیه‌سازی ساده و نمایش به‌صورت قدم به قدم شبیه‌سازی و همچنین شبیه‌سازی رخدادهای نادر در مدل‌های گسسته‌رخداد تصادفی متنی و گرافیکی فراهم می‌آورد. هسته SimGine[6] یک موتور شبیه‌سازی برمبنای توصیف SDES است. برای شبیه‌سازی یک مدل (مانند شبکه‌های پتری تصادفی، جبر پردازه‌های تصادفی و مدل‌های دیگر)، کافی است ترجمه‌ای از مدل به زبان ورودی این موتور صورت گیرد. بعد از ترجمه مدل، موتور SimGine شبیه‌سازی را انجام می‌دهد. این موتور شبیه‌سازی به‌عنوان هسته ابزار PDETool استفاده شده است.

معماری کلی ابزار در شکل (۱) آمده است.



شکل (۱): معماری کلی چارچوب PDETool و ارتباط آن با موتور شبیه‌سازی SimGine [8]

۵. الگوریتم تولید فضای حالت نمادین

در این قسمت، الگوریتم پیشنهادی تولید فضای حالت نمادین تشریح خواهد شد که در ابزار PDETool به‌منظور تولید فضای حالت، پیاده‌سازی شده است. شبه کد شکل (۳) الگوریتم تولید فضای حالت برای مدل SDES را بیان می‌کند.

```
NODE
{intvar;
   long low;
   long high;
   long parent;
   long refs;}
```

(الف)utable node staructure

```
HNODE
   {int var;
   long low;
   long high;
   long donotcare;
   long ulow;
   long uhigh;
   long refs;}
```

(ب) htable node structure

شکل (۲): ساختمان داده گره‌ها الف) نمودار تصمیم دودویی مرتب کاهش‌یافته و ب) نمودار تصمیم مرتب

1. Ordered Binary Decision Diagram(OBDD)
2. Reduced Ordered Binary Decision Diagrams
3. Cofactor
4. Universal
5. Canonical
6. SIMulationenGINE

تابع fnSSG مدل تعریف‌شده به زبان SDES را به‌عنوان ورودی دریافت نموده و ساختمان داده‌ای از نوع نمودار تصمیم دودویی را که حاوی مجموعه حالات قابل دسترس مدل ورودی است، به‌عنوان خروجی برمی‌گرداند. ساختمان داده گره‌های ROBDD در شکل (۲) آمده که از ساختمان داده الف برای ساخت جدول utable که ساختار ROBDD را نگهداری می‌کند و از ساختمان داده ب برای نگهداری جدول htable استفاده می‌شود تا به کمک آن بتوان در هنگام ساخت نمودار تصمیم، از ایجاد توابع تکراری جلوگیری کرد.

```
    Algorithm #1: state space generation
fnSSG(Input: SDES model) Output: ROBDD
    begin
1:      ROBDD ss,out,newGens;
2:      Initialize(ss,out,newGens);
3:      ss. findOrAdd(model.initialstate);
4:      out. findOrAdd(model.initialstate);
5:      while out isn't empty do
6:      begin
7:          for each state s in out do
8:          begin
9:              for each action a in model do
10:             begin
11:                 if a is enabled in s then
12:                 begin
13:                     newState=s.execute(a);
14:                     if  ss.findOrAdd(newState)==false
then
15:                         newGens.findOrAdd(newState);
16:                 end if
17:             end for
18:         end for
19:         out=newGens;
20:         newGens=null;
21:     end while
22:     return ss;
    end function;
```

شکل (۳): الگوریتم تولید فضای حالت نمادین

شکل (٤) الگوریتم جستجو و افزودن حالت به نمودار تصمیم دودویی مرتب کاهش‌یافته را نشان می‌دهد. تابع findOrAdd حالتی را که آرایه‌ای از مقادیر دودویی است، به‌عنوان ورودی دریافت کرده و مقداری دودویی برمی‌گرداند که این مقدار، اگر این حالت از قبل در نمودار موجود باشد، مقدار true و در غیر این صورت مقدار false خواهد بود.

شکل (۵) الگوریتم پیداکردن توابع موجود در ROBDD را نشان می‌دهد. این تابع دنبال تابع خواسته‌شده به‌صورت معکوس در جدول htbale می‌گردد. اگر پیدا کرد که اندیس تابع مربوطه در جدول utable را برمی‌گرداند تا برای پیاده‌سازی توابع جدید استفاده شود و اگر پیدا نکند، تابع مورد نیاز را ساخته و اندیس مورد اشاره به تابع را برمی‌گرداند. لازم به ذکر است که این تابع فقط در صورتی فراخوانی می‌شود که قسمتی از تابع در جدول utable پیدا نشود.

```
Algorithm #2: find or Add a State to ROBDD
findOrAdd(Input: state s) Output: boolean;
   begin
1:    intvarnumber=s.length;
2:    booleab find=true;
3:    index=2;
4:    while 1≤varnumer≤model.varnumbers and index>1
do
5:    begin
6:      if s[varnumber-1]==0 then
7:          direction=low
8:      else
9:          direction=high
10:     end if
11:     if uTable[index].direction>0 then
12:      begin
13:        index=uTable[index].direction;
14:        varnumber=uTable[index].var;
15:      end
16:     else
17:      begin
18:         find=false;
19:         addhuTable(s,varnumber,index);
20:         break;
21:      end if
22:   end while
23:  return find;
end function
```

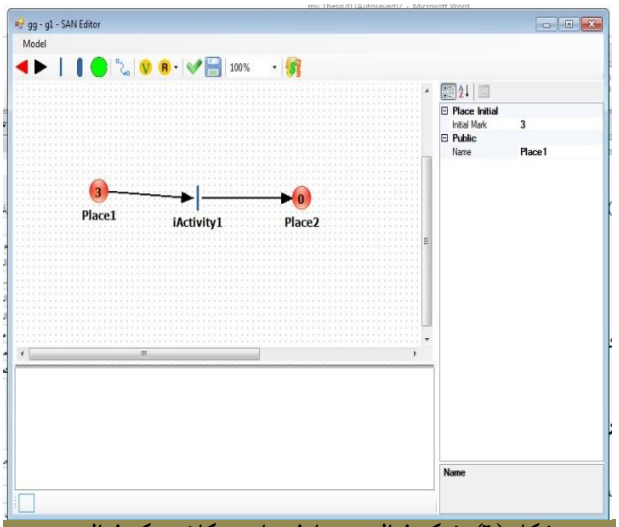
شکل (٤): الگوریتم جستجو و افزودن حالت به نمودار تصمیم دودویی مرتب کاهش‌یافته

```
Algorithm #3: add a state reverse to htable and state to utable with creating linkage form htable to utable rows
addhuTable(Input: state s,int varnumber,long index)
    begin
1:      hindex=2;
2:      var=1;
3:      while  var<varnumber do
4:      begin
5:          if state[var-1]==0 then
6:              direction=low;
7:          else
8:              direction=high;
9:          if htable[hindex].direction==0 then
10:             createNewuhNodes(hindex,state,var);
11:         hindex=htable[hindex].direction;
12:         var=htable[hindex].var;
13:     end while
14:     if utable[index].refs>1 then
15:         index= fnCopy(state,index);
16:     if state[varnumber-1]==0 then
17:         utable[index].direction=htable[hindex].ulow;
18:     else
19:
utable[index].direction=htable[hindex].uhigh;
20:     if utable[index].low==utable[index].high!=0
AND index>2 then
21:         shrink(index,hindex,state);
22:     end if
 End function
```

شکل (۵): الگوریتم افزودن معکوس حالت به htable و ایجاد آن در utable

٦. مثال عملی

برای نمونه، مثالی کوچک در ابزار PDETool پیاده‌سازی کرده و فضای حالت تولیدشده توسط ابزار را نمایش می‌دهیم. شکل (٦) یک مثال شبکه فعالیت تصادفی است که شامل دو مکان و یک کنش است. حالت اولیه مثال، شامل ۳ نشانه در مکان اول است. کنش مثال، یک کنش آنی است و برای سادگی، هیچ پیچیدگی به این مثال نیفزودیم. شکل (۷) فضای حالت تولیدشده برای مثال شکل (٦) را نشان می‌دهد که شامل ٤ حالت خواهد شد.



شکل (٦): شبکه فعالیت تصادفی با دو مکان و یک فعالیت

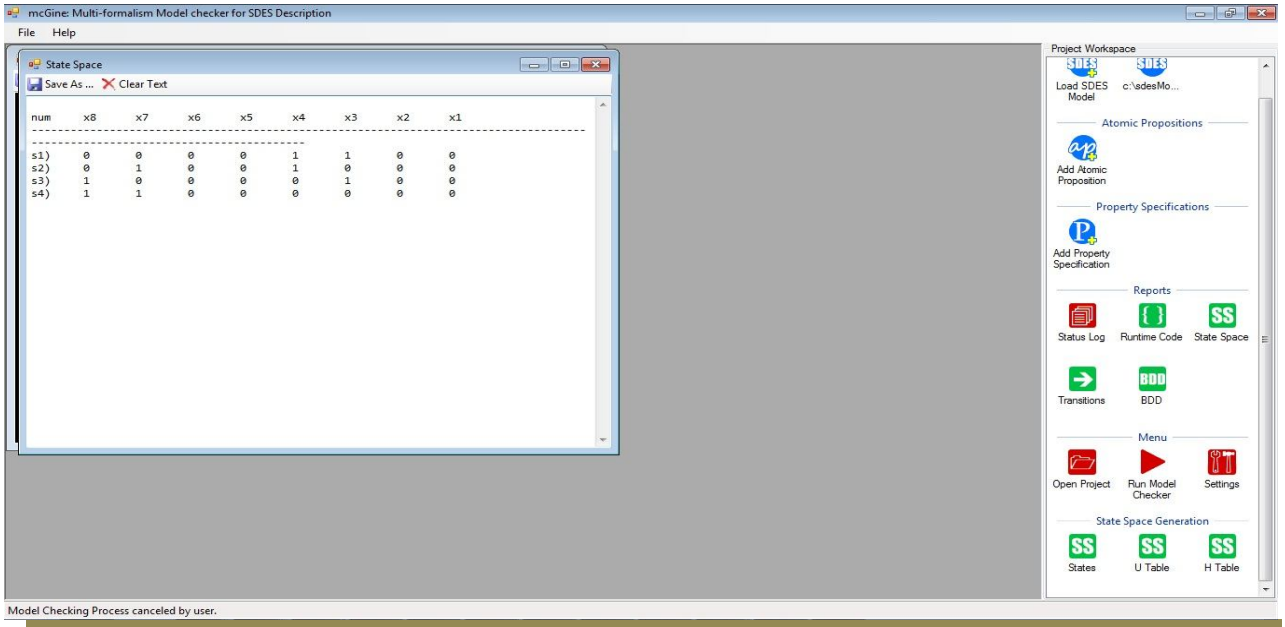
اگر حالات را به‌صورت "p1p2" نشان دهیم، چهار حالت به‌صورت {"۰۳"،"۱۲"،"۲۱"،"۳۰"} نشان داده می‌شود. برای کدکردن مکان‌ها از تابع کدینگ استفاده کرده‌ایم که به مکان اول چهار بیت به‌صورت "$X_2X_1X_4X_3$" اختصاص داده شده است و برای مکان دوم نیز به‌صورت "$X_6X_5X_8X_7$" بیت‌ها اختصاص داده شده‌اند. همان‌طور که در

شکل (۷) نیز نشان داده شده است، پس از اجرای دستور تولید فضای حالت نمادین، چهار حالت مطابق جدول (۱) ایجاد می‌شود.

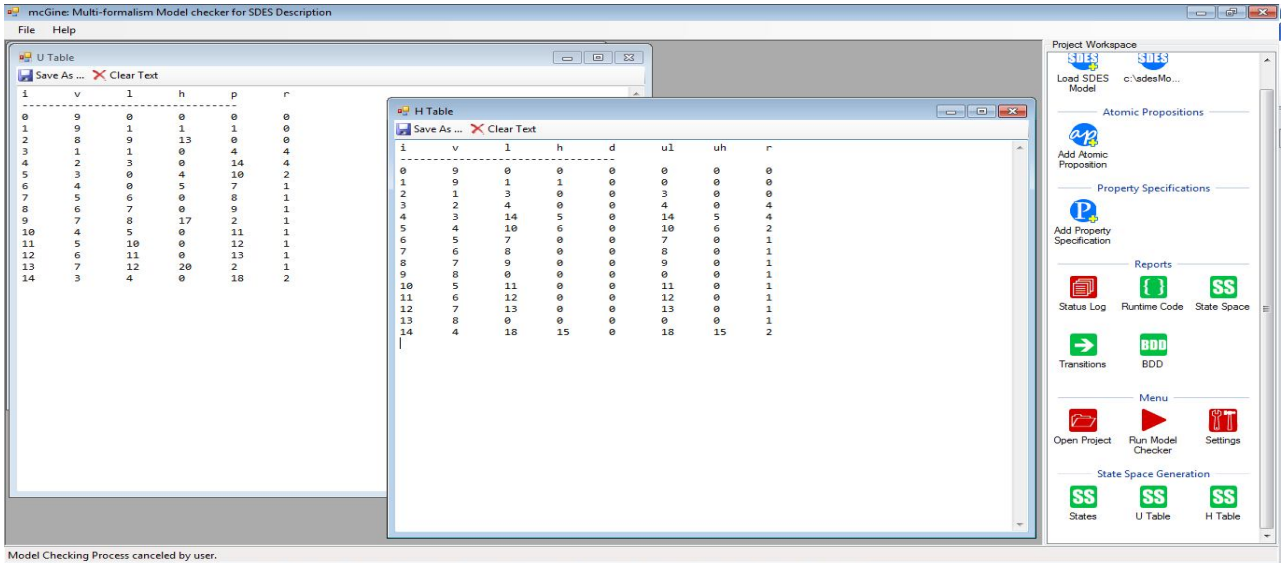
همچنین در شکل (۸)، اطلاعات مربوط به دو جدول utable و htable مورد نیاز برای تولید و نگهداری فضای حالت نشان داده شده است.

جدول (۱): حالات تولیدشده مربوط به شکل (٦) با کدینگ مربوطه

p_1p_2=30	p1=$X_2X_1X_4X_3$=0011 p2= $X_6X_5X_8X_7$=00000
p_1p_2=21	p1=$X_2X_1X_4X_3$=0010 p2=$X_6X_5X_8X_7$=00001
p_1p_2=12	p1=$X_2X_1X_4X_3$=0001p2=$X_6X_5X_8X_7$=00010
p_1p_2=03	p1=$X_2X_1X_4X_3$=0000 P2=$X_6X_5X_8X_7$=00011



شکل (۷): فضای حالت تولیدشده برای شکل (٦)



شکل (۸): جدول utable و htable استفاده‌شده برای تولید فضای حالت شکل (۷)

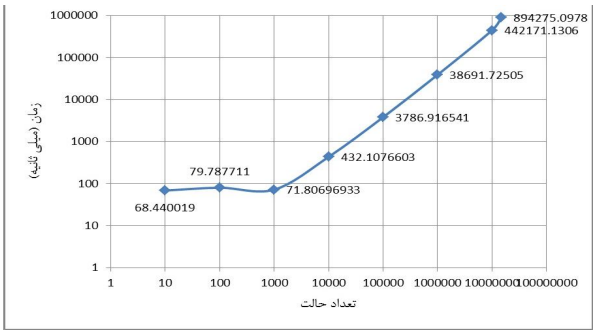
۷. ارزیابی روش پیاده‌سازی‌شده

برای نمایش رفتار برنامه در تولید فضای حالت، نمودار حالت-زمان برای اجراهای مختلف آن رسم شد. حالات مورد استفاده برای افزودن به فضای حالت، از اجرای تابع تصادفی تولید شده‌اند. اعداد گرفته‌شده حاصل ۵۰۰ بار اجرای برنامه برای هر نقطه از نمودار و میانگین‌گیری از آن‌هاست. بدین صورت که مثلاً برای به‌دست‌آوردن زمان لازم برای افزودن ۱۰۰۰ حالت به یک ROBDD با ۲٤ متغیر دودویی که یک نقطه از نمودار شکل (۹) را تشکیل داده است، ۵۰۰ بار عمل افزودن ۱۰۰۰ حالت تصادفی به فضای حالت انجام شد. همه‌ی این حالات با کمک تابع Random تولید شده است. هر بار نیز بعد از افزودن ۱۰۰۰ حالت، ROBDD راه‌اندازی مجدد می‌شد و زمان راه‌اندازی مجدد نیز محاسبه می‌شد. بعد از تکرار این عمل به تعداد مورد نظر، میانگین زمان‌های به‌دست‌آمده برای

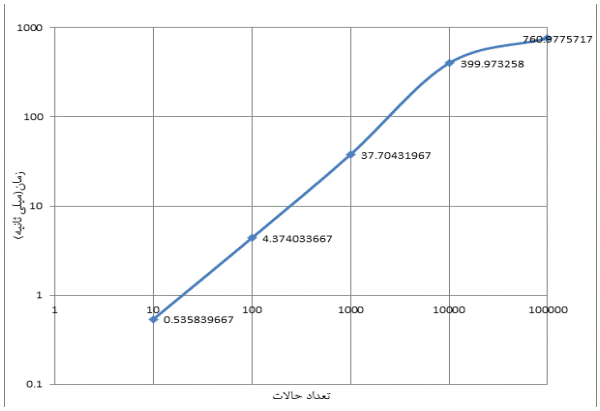
تخمین میانگین زمان مورد نیاز برای افزودن این تعداد حالت است.

به دلیل اینکه زمان راه‌اندازی نمودار تصمیم در همه آزمایش‌ها منظور شده است، زمان‌ها در اجراهای واقعی برای مدل‌های عملی دیگر نیز مشابه خواهد بود. پایداری خروجی با اجرای بیش از ۳۰ بار برای همه خروجی‌ها، واریانس کمتر از ۰.۰۱ به‌دست می‌داد که نشانگر همگرایی بالای خروجی به‌دست‌آمده است.

نمودارهای زیر حاصل اجرای برنامه است که محور x آن تعداد حالات افزوده شده است و محور y آن نمایشگر میانگین زمان مورد نیاز برای افزودن این حالات به ROBDD است. واحد زمان میلی ثانیه است. هر دو محور با لگاریتم مبنای ۱۰ نشان داده شده است. نمودار اولی برای نمودار تصمیم دودویی کاهش‌یافته با ۲٤ متغیر دودویی و نمودار دومی حاصل کار با نمودار تصمیم دودویی کاهش‌یافته با ۱٤ متغیر دودویی است. داده‌های به‌دست‌آمده از اجرای برنامه روی پردازنده core i5-480M با سرعت پردازش ۲.٦٦ GHz انجام شد که با حافظه ٤ RAMG اجرا شده است.



شکل(۹): نمودار تعداد حالات افزوده و زمان برای ROBDD با ۲٤ متغیر تصادفی دودویی



شکل(۱۰): نمودار تعداد حالات افزوده و زمان برای ROBDD با ۱٤ متغیر تصادفی دودویی

برای مقایسه‌ی روش پیشنهادی با نرم‌افزارهای موجود در این زمینه، بسته نرم‌افزاری Buddy [۱۱،۲۵] را انتخاب کرده‌ایم. Buddy بسته نرم‌افزاری است که در دانشگاه IT University of Copenhagen توسعه یافته است. این بسته نرم‌افزاری در کنار بسته نرم‌افزاری CUDD [۱٤] از دانشگاه Colorado، جزو بسته‌های نرم‌افزاری اصلی و معروفی است که برای تولید و مدیریت نمودارهای تصمیم دودویی استفاده می شود؛ برای نمونه، ابزاری مثل JavaBDD [۲٦] از دانشگاه تورنتو و Jedd [۲۷] و SableJBDD [۲۸] از دانشگاه McGill از هر دوی بسته‌های نرم‌افزاری Buddy و CUDD، برای تولید نمودارهای تصمیم دودویی استفاده می‌کند.

روش پیشنهادی و بسته نرم‌افزاری Buddy 2.2 را به‌عنوان نمونه در حل مسئله n وزیر با هم مقایسه می‌کنیم. مشخصات سخت‌افزاری مورد استفاده برای حل این مسئله برای هر دو روش یکسان بود. سخت‌افزار مورد استفاده، شامل سیستمی کامپیوتری با مشخصات پردازنده core i5-480M با سرعت پردازش 2.66 GHz بود که 4G حافظه RAM داشت. ابزار PDETool تحت سیستم‌عامل ویندوز ۷ اجرا شده است. این ابزار تحت چارچوب .NET. با زبان C# توسعه یافته است. بسته نرم‌افزاری Buddy 2.2 تحت سیستم‌عامل Ubuntu 11.4 اجرا شده است. این بسته نرم‌افزاری با زبان C/C++ توسعه یافته است.

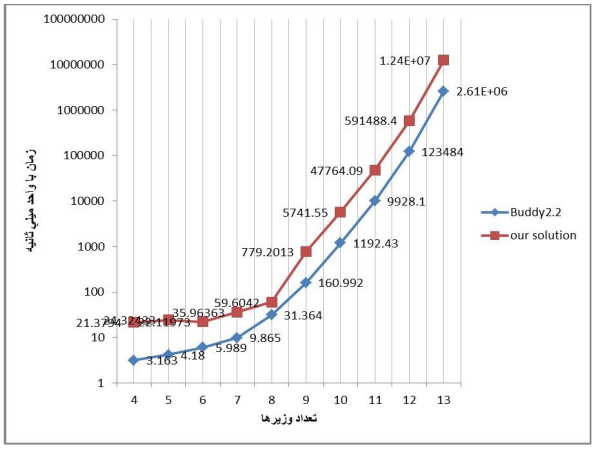
داده‌های حاصل از اجرای برنامه برای تعداد وزیرهای ٤ تا ۱۳ وزیر در جدول ۲ آورده شده و در شکل (۱۱) به‌صورت نموداری نیمه لگاریتمی ترسیم شده است. در نمودار مربوطه، محور افقی تعداد وزیرها (n) را نشان می‌دهد و محور عمودی با واحد میلی ثانیه است که مدت زمان لازم برای حل مسئله n وزیر را نشان می‌دهد. واحد محور عمودی به‌صورت لگاریتمی در مبنای ۱۰ است. خط قرمز، سرعت اجرای راه‌حل پیشنهادی را نشان می‌دهد و خط آبی سرعت اجرای Buddy را نشان می‌دهد.

از نمودار پیداست که سرعت اجرای Buddy بهتر از روش پیشنهادی است. چندین عامل در سریع‌تربودن آن دخیل است.

دلیل اصلی، پیاده‌سازی مناسب و بهبودیافته Buddy است که تأثیر زیادی در سرعت اجرای برنامه دارد. برای کمتر از ۸ وزیر، سرعت Buddy نسبت به راه‌حل پیشنهادی خیلی بالاتر است. دلیل آن این است که روش پیشنهادی برای حالات بسیار زیاد درنظر گرفته شده است و با زیادشدن حالات، کارآیی خود را بهتر نشان می‌دهد. برای تعداد حالت کم، زمان راه‌اندازی اولیه باعث می‌شود تا روش پیشنهادی خیلی کندتر به‌نظر بیاید. با افزایش تعداد وزیرها و به‌تبع آن افزایش تعداد حالات، روش پیشنهادی نیز سرعت نمایی متناسب با سرعت Buddy دارد. برای ۹ وزیر که مسئله ۳۵۲ حالت دارد، روش پیشنهادی ٤.٨٤ برابر نسبت به Buddy کندتر است و برای ۱۳ وزیر که ۷۳۷۱۲ حالت دارد ٤.۷۵ برابر کندتر است. نمودار نشان می‌دهد که با افزایش فضای حالت و زیادشدن تعداد حالات، روش پیشنهادی به‌سرعت حل مسئله Buddy نزدیک‌تر می‌شود.

جدول (۲): داده‌های مربوط به حل مسئله n وزیر

queen number	number of states	Time(miliSec) of Buddy2.2	Time(miliSec) of our solution
4	2	3.163	21.3794
5	10	4.18	24.32433
6	4	5.989	22.11973
7	40	9.865	35.96363
8	92	31.364	59.6042
9	352	160.992	779.2013
10	724	1192.43	5741.55
11	2680	9928.1	47764.09
12	14200	123484	591488.4
13	73712	2.61E+06	1.24E+07



شکل (۱۱): نمودار حل مسئله n وزیر

۸. تحلیل ریاضی بدترین حالت

بزرگ‌ترین و بدترین فضا وقتی ایجاد می‌شود که هیچ‌کدام از حالات نتوانند کاهش یابند. این وقتی اتفاق می‌افتد که دقیقاً نصف فضای حالت، یعنی 2^{Xn-1} حالت که امکان کاهش ندارد، اتفاق بیفتد که در آن Xn تعداد متغیرهای مورد نیاز برای ساخت نمودار تصمیم دودویی است و احتمال این رخداد برابر فرمول ٤ خواهد بود.

$$\binom{2^{Xn}}{2^{Xn-1}} / 2^{Xn} \qquad (٤)$$

در بدترین حالت، تعداد $2^{Xn-1}+1$ گره برای ساخت نمودار تصمیم دودویی مورد نیاز خواهد بود. همچنین احتمال کاهش نمودار با ورود حالتی جدید برابر $\left(\frac{1}{2}\right)^{Xn}$ خواهد بود که برابر با احتمال قرارگیری حالت در یکی از موقعیت‌های کاهش‌دهنده‌ی نمودار است.

۹. نتیجه

در عمل، تولید فضای حالت حتی برای مدل‌های کوچک نیز به‌دلیل مشکل انفجار فضای حالت خیلی سخت است. این مشکل به این دلیل به‌وجود می‌آید که اندازه‌ی فضای حالت نسبت به تعداد متغیرهای مدل رشد نمایی دارد. از راهکارهای غلبه بر این مشکل، نگهداری فضای حالت به‌صورت ضمنی به‌جای نگهداری صریح آن‌هاست. برای تخفیف این مشکل در ابزار PDETooL، از روش تولید و نگهداری نمادین فضای حالت به شکل ROBDD کمک گرفته شد. به‌دلیل استفاده از BDD برای پیداکردن توابع مورد نیاز یا افزودن توابع جدید، عملیات افزودن حالت جدید با همان مرتبه جستجوی حالت انجام می‌شود که برابر $O(\Sigma_{sv\epsilon SV^*}\lceil \log_2(count(S^*(sv)))\rceil)$ است که از مرتبه لگاریتمی در مبنای ۲ است. این مرتبه برابر ارتفاع نمودار تصمیم دودویی است. خاصیت کاهش‌یافته‌بودن نمودار در همان زمان افزودن حالت‌های جدید رعایت می‌شود و دیگر نیازی نیست تا توابع کاهش‌دهنده که از مرتبه بالایی برخوردار هستند، بعد از ساخت نمودار بر روی آن اعمال شوند.

با پیاده‌سازی این روش در ابزار PDETool که یک روش صوری چندگانه است، امکان تولید فضای حالت نمادین برای روش‌های صوری دیگر، مثل شبکه‌های پتری تصادفی، شبکه‌های فعالیت تصادفی و غیره که ترجمه‌ای از آن‌ها به SDES وجود دارد، فراهم شد. پیشنهاد می‌شود به‌عنوان کار آینده، روی تکمیل و افزودن عملگر و عملکردهای جدید به نمودار تصمیم دودویی مرتب کاهش‌یافته کار تحقیقاتی بیشتری انجام گیرد. همچنین پیاده‌سازی و کار روی نمودار تصمیم چندمقداری و نمودارهای تصمیم دودویی چندپایانه‌ای می‌تواند تکمیل‌کننده این راه باشد. استفاده از نمودارهای تصمیم چندمقداری ارتفاع نمودار تصمیم را کم می کند و پیاده‌سازی آن را به مدل واقعی که مدل‌سازی‌شده است، شبیه‌تر می کند. بررسی دیگری که می تواند انجام شود، استفاده از الگوریتم‌های اشباع برای تولید فضای حالت است.

مراجع

[1] G. Ciardoand AS. Miner, Storage alternatives for large structured state spaces, in 9th International Conference on ModelingTechniques and Tools for Computer Performance Evaluation, Vol. 1245, Springer, PP. 44--57,Jun.1997.

[2] P. Buchholz, Hierarchical structuring of superposed GSPNs ,in 7th International Workshop on Petri Nets and Performance Models (PNPM'97), France, IEEE Computer Society Press, pp. 81--90, Jun.1997.

[3] S. Graf, B. Steffen, and G. Luttgen, Compositional minimization of finite state systems using interface specifications, Formal Aspects of Computing,pp. 607—616, 1996.

[4] P. Godefroid, Partial-order methods for the verification of concurrent systems: an approach to the state-explosion problem, vol. 1032 of Lecture Notes in Computer Science, Springer-Verlag, 1996.

[5] K. Jensen, ColouredPetrinets, in Petri Nets: Central Models and Their Properties, PartI, Proceedings of an Advanced Course, vol. 254 of Lecture Notes in Computer Science, Bad Honnef, Germany, Springer-Verlag, pp. 248--299, September1987.

[6] E. Pastor, O. Roig, J. Cortadella, and R. Badia,Petrinet analysis using Boolean manipulation, in 15th International Conference on the Application and Theory of PetriNets(ICATPN'94), vol. 815 of Lecture Notes in Computer Science, Springer-Verlag, pp.416--435, June1994.

[7] R. E. Bryant. Graph-based algorithms for boolean function manipulation. IEEE Trans. Comp., Vol. 100, No. 8, PP.677--691, 1986.

[8] T. Kam, T. Villa, R. Brayton, and A. Sangiovanni-Vincentelli, Multi-valued decision diagrams: Theory and applications,Multiple-ValuedLogic,Vol. 4, pp.9--62, 1998.

[9] AS. Miner, G. Ciardo, Efficient reachability set generation and storage using decision diagrams, Springer, PP. 6—25, 1999.

[10] G. Ciardo, R. Marmorstein, and R. Siminiceanu, The saturation algorithm for symbolic state-space exploration, Vol. 8, No. 1, International Journal on Software Tools for Technology Transfer, Springer, 2006.

[11] J. Babar and A. S. Miner, "Meddly: Multi-terminal and Edge-Valued Decision Diagram LibrarY," in proc. of the 2010 Seventh International Conference of Quantitative Evaluation of Systems (QEST), 2010, pp. 195--196.

[12] Mobius. Model Based Environment for Validation of System Reliability, Availability, security, and Performance. [Online] Illinois University, 04 27, 2012. https://www.mobius.illinois.edu/index.html.

[13] M. Kwiatkowska,G. Norman, and D. Parker, Probabilistic Symbolic Model Checking with PRISM: A Hybrid Approach, International Journal on Software Tools for Technology Transfer (STTT), Vol. 6, pp. 128--142, Springer, 2004.

[14] F. Somenzi, CUDD: Colorado University Decision Diagram Package, Public Software, Colorado University, Boulder, 1997. http://vlsi.colorado.edu/~fabio/.

[15] A. Zimmermann, Stochastic Discrete-Event Systems: Modeling, Evaluation and Applications. Springer, 2008.

[16] C. Lee, "Representation of Switching Circuits by Binary-Decision Programs," vol. 38, pp. 985--999, Jul. 1959.

[17] S. Akers, "Binary Decision Diagrams," IEEE Transactions on Computers, vol. 27, no. 6, pp. 509--516, Jun. 1978.

[18] R. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," IEEE Transactions on Computers, vol. 35, no. 8, pp. 677-691, Aug. 1986.

[19] C. Baier and J. P. Katoen, Principles of Model Checking(Representation and Mind Series). The MIT Press, 2008.

[20] D. Parker and A. Miner, "Symbolic representations and analysis of large state spaces," in Validation of Stochastic Systems, 2004., pp. 296-338.

[21] G. Ciardo, G. Luttgen, and AS. Miner, "Exploiting Interleaving Semantics in Symbolic State-Space Generation," in Formal Methods in System Design, vol. 31, 2007, pp. 63--100.

[22] A. Khalili, A. JalalyBidgoly, and M. AbdollahiAzgomi, "PDETool: A Multi-formalism Modeling Tool for Discrete-Event Systems Based on SDES Description," Lecture Notes in Computer Science, vol. 5606, pp. 343-352, Jun. 2009.

[23] A. JalalyBidgoly, A. Khalili, and M. AbdollahiAzgomi, "Implementation of Coloured Stochastic Activity Networks within the PDETool Framework," in Proc. of 3rd Asia Int'l Conf. on Modelling& Simulation (AMS09), 2009, pp. 710-715.

[24] SimGineHomePage. [Online]. http://pdel.iust.ac.ir/Projects/SimGine.html

[25] [Online]. http://vlsicad.eecs.umich.edu/BK/Slots/cache/www.itu.dk/research/buddy/index.html

[26] [Online]. ftp://ftp.cs.toronto.edu/pub/shou/Research/javaBDD/javabdd_src/JavaBDD/JavaBDD.html.

[27] [Online]. http://www.sable.mcgill.ca/jedd/.

[28] [Online]. http://www.sable.mcgill.ca/~fqian/SableJBDD/.