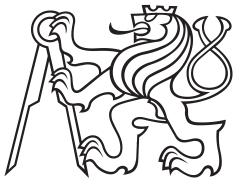
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra řídicí techniky

Stochastický simulátor výpadků elektráren v přenosové soustavě

Bc. Ondřej Svoboda

Vedoucí: Ing. Jan Zábojník
Obor: Kybernetika a robotika
Studijní program: Systémy a řízení
Květen 2017

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Svoboda Ondřej**

Studijní program: Kybernetika a robotika
Obor: Systémy a řízení

Název tématu: **Stochastický simulátor výpadků elektráren v přenosové soustavě**

Pokyny pro vypracování:

1. Seznamte se s problematikou modelování a simulace výpadků zdrojů.
2. Vytvořte generátor náhodných realizací výpadků elektráren z historických statistik poruchovosti.
3. Navrhněte algoritmus, který pro zadanou realizaci výpadků zajistí optimální skladbu zdrojů pokrývajících zatížení. Respektujte dodatečná omezení na množství kladných točivých rezerv a rozdílné způsoby vynuceného nasazování části elektráren.
4. Proveďte stochastickou simulaci na dodaném modelu České přenosové soustavy, otestujte vliv výpadku největšího bloku na stabilitu soustavy. Výsledky diskutujte.

Seznam odborné literatury:

[1] J. Zábojník, 'Modelovani vyroby a toku elektricke energie v evropske prenosove soustave', DP, FEL CVUT, 2012
[2] J. Zábojník, M. Dvořák, 'Power grid simulation model for long term operation planning', Applied Thermal Engineering, Volume 70, Issue 2, 22 Sept 2014, Pages 1294-1305, ISSN 1359-4311
[3] K. Sigman, 'Lecture Notes on Stochastic Modeling I', online: http://www.columbia.edu/~ks20/stochastic-I/stochastic-I.html
[4] E. Bauer, R. Adams, D. Eustace, 'Beyond Redundancy: How Geographic Redundancy Can Improve Service Availability and Reliability of Computer-Based Systems', 5 Oct, 2011, ISBN: 9781118104910

Vedoucí: Ing. Jan Zábojník

Platnost zadání: do konce letního semestru 2017/2018

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze, dne 21. 2. 2017

Poděkování

Rád bych poděkoval vedoucímu této práce Ing. Janu Zábojníkovi za jeho nedocenitelné rady, ochotu a vstřícnost. Bez jeho odborné i pedagogické pomoci by tato práce nemohla vzniknout. Také bych rád poděkoval své rodině a přátelům za podporu při mém studiu a psaní této diplomové práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 26. května 2017

Abstrakt

Tato diplomová práce se zabývá návrhem generátoru pseudo-stochastických výpadků bloků elektráren v elektrické přenosové soustavě a simulátoru, který s ohledem na tyto vygenerované výpadky a další dodatečná omezení zajistí pokrytí zatížení sítě vhodnou skladbou zdrojů. Práce dále obsahuje analýzu vlivu výpadku největšího bloku zadané sítě na stabilitu soustavy v závistlosti na hodině výpadku. Výstupem práce je funkční programová knihovna napsaná v jazyku C#.

Klíčová slova: pseudo-stochastické výpadky, unit commitment problém, elektrická přenosová soustava, simulátor

Vedoucí: Ing. Jan Zábojník
Český institut informatiky, robotiky a kybernetiky,
Jugoslávských partyzánů 1580/3,
Praha 6

Abstract

The purpose ot his master thesis is to design pseudo-stochastic failure generator of power plant units in electrical power grid and simulator, that will satisfy coverage of power demand with respect to generated failures and other additional restrictions. The work also includes power grid stability analysis of failure of the biggest source in grid depending on hour of failure. Outcome of this work is program library written in C#.

Keywords: pseudo-stochastic failures, unit commitment problem, electrical power grid, simulator

Title translation: Stochastic forced outage simulator of power plants in transmission network

Obsah

Obrázky

Tabulky

Úvod

V současné době je stále důležitější analyzovat stabilitu elektrické přenosové soustavy za různých krizových situací, protože nároky na tuto síť neustále rostou. Je nutné testovat, jaký vliv budou mít náhodné výpadky zdrojů na schopnost pokrytí zatížení a nebo v jakém období bude dopad výpadku největšího zdroje na stabilitu sítě nejvýraznější.

Zajištění optimální skladby zdrojů elektrické energie v elektrizační síti je obtížný optimalizační problém a jeho řešení je časově velice náročné. Jestliže ale nejdříve nalezneme alespoň jedno kvalitní řešení, které poskytneme softwarovému nástroji pro řešení optimalizačních úloh jako počáteční řešení, ze kterého při řešení optimalizační úlohy vychází, můžeme dosáhnout výrazné redukce stavového prostoru řešení problému a tím i zkrácení celkové doby výpočtu. Za tímto účelem byl navržen a implementován simulátor, který dokáže reagovat vhodnou skladbou zdrojů na požadavky soustavy a výše zmíněné výpadky s ohledem na všechna dodatečná omezení modelu soustavy. Při integraci navrženého simulátoru do optimalizačních nástrojů tak bude možné výrazně snížit čas, potřebný k nalezení optimálního řešení optimalizační úlohy, aniž by došlo k degradaci nalezených výsledků.

První kapitola obsahuje popis simulované sítě a vlastnosti prvků, které tato síť obsahuje. Ve druhé kapitole je popsán navržený generátor pseudonáhodných výpadků zdrojů v síti, který vytváří varianty výše zmíněných krizových situací. Ve třetí kapitole je detailně popsán navržený a implementovaný simulátor. Čtvrtá kapitola uvádí výsledky simulací výpadků největšího zdroje sítě v každé hodině a výsledky simulace s jednou sadou náhodných výpadků. V závěru jsou shrnuty dosažené výsledky a uvedeny návrhy na další možná vylepšení.

Kapitola 1

Struktura simulované sítě

V této kapitole je popsán model simulované soustavy, především pak jeho zdroje elektrické energie, jejich vlastnosti a omezení. Řešení problému výroby a přenosu elektrické energie v síti je velmi náročný problém z teorie grafů a proto byl použitý model v simulátoru koncipován jako jednouzlový. V případě, že zadaný model obsahuje více uzlů, jsou tyto uzly agregovány do jednoho jediného. Skladba zdrojů je tedy navrhována pouze v rámci tohoto uzlu bez ohledu na přenosové cesty. Použitý model se skládá z následujících prvků a jejich parametrů.

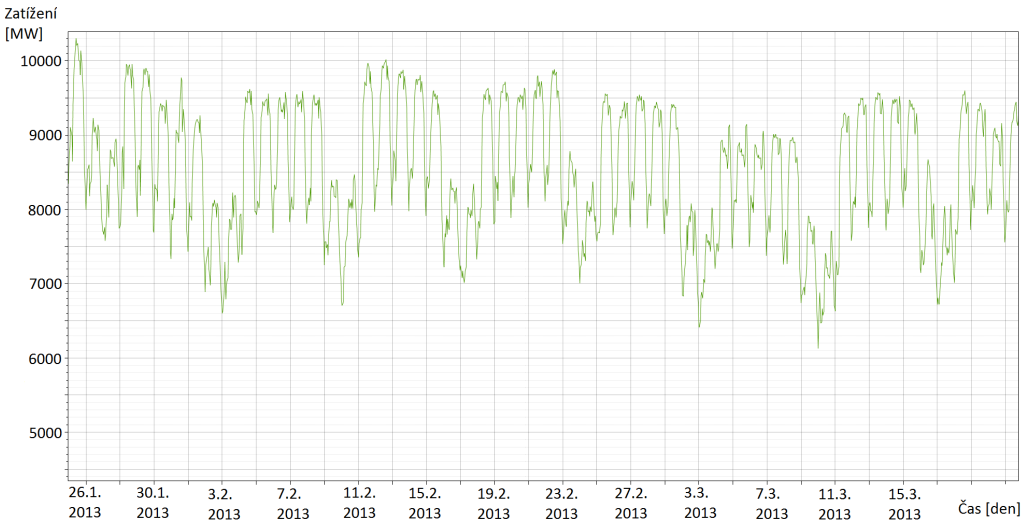
1.1 Uzel

Uzel v tomto modelu představuje rozvodnu VVN, do které jsou připojené elektrárny a která je prostřednictvím vedení VVN připojena ke zbytku přenosové soustavy. V uzlu jsou obsaženy tyto časové řady:

1. Vstupní
   - Load - Zatížení. Příklad zatížení je uveden na obrázku č. 1.1. Jedná se o historický průběh z roku 2013. Data byla převzata z webových stránek společnosti ČEPS, a.s.
   - RequiredPositivePrimaryReserve - Požadované kladné rezervy na nasazených zdrojích. Dostupná kladná regulační energie zdroje představuje rozdíl mezi aktuální a maximální možnou výrobou energie.

2. Výstupní

   - PrimaryReserveDelta - Nedodržené kladné rezervy
   - UndelPwr - Nedodaná energie (vypočtena jako velikost záporných hodnot rozdílu mezi vyrobenou energií a zadaným zatížením)



Obrázek 1.1: Ukázka zatížení v uzlu [1].

Pro každý uzel jsou jako vstupní parametry definovány ceny všech v uzlu obsažených paliv a emisí elektráren. V modelu jsou použity následující paliva: uran, hnědé uhlí, černé uhlí, topný olej, plyn (různé druhy), voda, vzduch a světlo. Ceny paliv jsou uvedeny v eurech za 1 MWht. Mezi uvažované emise patří oxid siřičitý (SO_2), oxid uhličitý (CO_2) a oxidy dusíku (NO_x). Ceny emisí jsou uvedeny v eurech za tunu.

1.2 Elektrárny

V této části jsou uvedeny vlastnosti, které platí pro všechny typy nebo skupiny typů elektráren. V rámci elektráren je uvažována pouze jedna vstupní časová řada. Jedná se o řadu MinReqPwr, která v případě nastavení MIN_RUN v omezení na výkon (viz níže) udává, jaký musí být minimální dodávaný výkon elektrárny.

Všechny elektrárny, nehledě na jejich typ, musí navíc splňovat několik omezení (omezení na chod a na dodanou energii). Jedná se o část sady omezení, převzatých z [4]. Zbylé restrikce byly zanedbány, což vede k výraznému zrychlení simulace, aniž by byla výrazně degradována kvalita získaných výsledků.

Omezení na chod

Dalším společným omezením na úrovni elektrárny je restrikce na chod bloků. V modelu je nazvána jako ECtype a lze nastavit na jednu ze 4 úrovní.

- EC - Provoz je řízen ekonomicky. Nasazování bloku není nijak omezeno.
- ALW_RUN - Všechny bloky elektrárny jsou trvale nasazeny.
- PART_ALW_RUN - Je trvale nasazena část bloků elektrárny.
- FIXED - Stav chodu bloků elektrárny je předem definován. Na tuto úroveň jsou obvykle nastaveny obnovitelné zdroje a přečerpávací elektrárny.

Omezení na dodanou energii

Provoz elektrárny může být dále regulován požadavky na výstupní výkon. V modelu je toto omezení uvedeno jako UDtype.

- EC - Výroba bloků je řízena ekonomicky. Výkon elektrárny není nijak omezen.
- MAX_RUN - Všechny bloky dané elektrárny musí po celou dobu jejich chodu vyrábět na maximální možný výkon.
- MIN_RUN - Součet výkonů všech bloků dané elektrárny musí být větší nebo roven hodnotě definované ve vstupní časové řadě MinReqPwr v dané hodině. Např. teplárenský provoz.
- SOFT_MAX_RUN - Všechny bloky dané elektrárny by měly po celou dobu chodu běžet na maximální možný výkon, avšak při přebytku energie v soustavě je možné jejich výkon snížit. Toto snížení je penalizováno.
- RES_MAX_RUN - Až na prioritu regulace stejné nastavení jako u typu SOFT_MAX_RUN. Snížení výkonu elektráren tohoto typu je možné pouze až po odstavení všech elektráren s typem SOFT_MAX_RUN.
- RES_FIX_RUN - Dodaný výkon všech bloků obnovitelného zdroje musí přesně odpovídat vstupní časové řadě ResReqPwr.
- FIXED - Dodaný výkon všech bloků dané elektrárny musí přesně odpovídat příslušným časovým řadám DelPwr, které jsou zadány jako vstup do simulace.

1.2.1 Tepelné elektrárny

Do kategorie tepelných elektráren jsou v tomto modelu zařazeny elektrárny jaderné, elektrárny spalující hnědé a černé uhlí, zemní, koksárenský a hutní plyn, bioplyn a lehké topné oleje.

Pro každý blok tepelné elektrárny je nutné vypočítat podle vzorce 1.1 jeho cenu vyprodukovaných emisí za 1 MWh_t a cenu za propálená paliva za 1 tepelnou MW. Tepelné elektrárny jsou v tomto modelu jediné, ve kterých se tyto ceny uvažují. Obnovitelné zdroje a akumulační elektrárny neprodukují žádné emise a cena jejich paliva je nulová.

$$C_{E_{MWht}} = \sum_{i}^{N} \frac{r_{ei} \cdot c_{et_i}}{1000} \qquad C_{F_{MWht}} = \sum_{j}^{M} r_{fj} \cdot c_{ft_j} \tag{1.1}$$

$C_{E_{MWht}}$	Cena emisí za 1 MWht
$C_{F_{MWht}}$	Ceny spotřebovaného paliva na výrobu 1 MWht
N	Počet blokem produkovaných emisí
M	Počet blokem pálených paliv
r_{ei}	Množství produkovaných emisí v kg/MWht
r_{fj}	Poměr daného typu spáleného paliva v 1 MW
c_{et_i}	Cena za 1 t emise
c_{ft_i}	Cena paliva za 1 vyrobenou MWht

1.2.2 Akumulační elektrárny

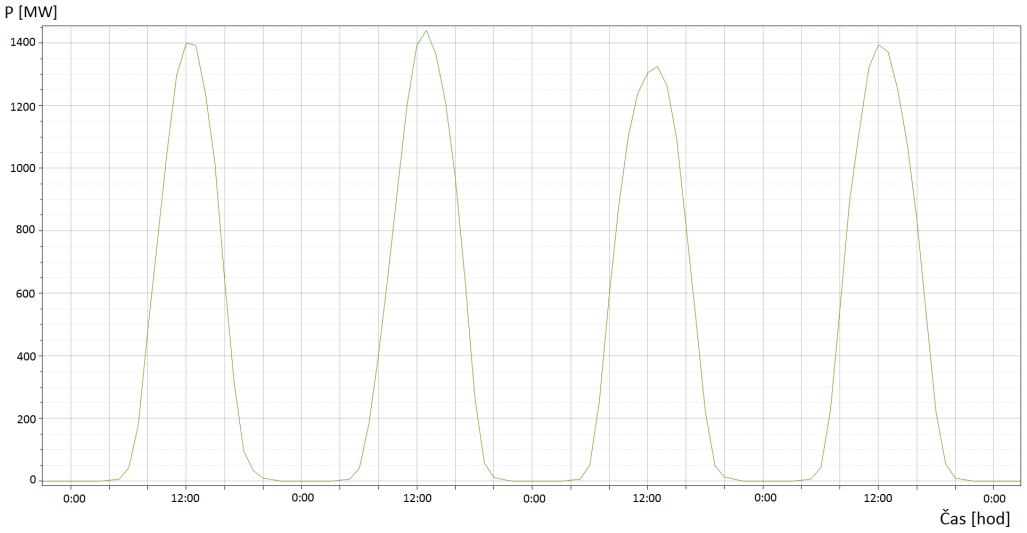
Akumulační elektrárny pracují na principu uchování energie, v tomto případě potenciální energie vody v uměle vytvořených nádržích s přítokem. Jejich mezihodinové změny stavu chodu probíhají s ohledem na hodinový krok simulace bez zpoždění a parametry MinUpTime a MinDownTime jsou nastaveny na nejkratší možný krok (1 hodinu). Jejich chod je již předem naplánován, omezení na chod i dodanou energii je nastaveno na úroveň FIXED. V simulátoru nejsou tyto elektrárny nijak optimalizovány.

1.2.3 Přečerpávací elektrárny

Přečerpávací elektrárny jsou speciálním případem vodních elektráren. Pracují na principu akumulace vody v umělých nádržích, kdy v době levné elektrické energie dochází k čerpání vody do rezervoáru a v době drahé energie naopak k vypouštění vody a tím ke generování energie. Naplánovat optimální provoz tohoto typu elektrárny je náročný optimalizační problém a proto se v simulátoru považují za předem naplánované (mají vždy nastavení na FIXED v omezení na chod i na dodanou energii). Při simulacích tedy vůbec nedochází k jejich regulaci a ani pro ně nejsou generovány výpadky.

1.2.4 Obnovitelné zdroje

Mezi obnovitelné zdroje patří v tomto modelu elektrárny větrné a solární. Tyto zdroje neprodukují žádné emise a jejich paliva (světlo, vítr) mají nulovou cenu. Jejich výstupní výkon je definován vstupní časovou řadou ResReqPwr. Ukázka výstupního výkonu obnovitelného zdroje je uvedena na obrázku č. 1.2. Jedná se o historický průběh z roku 2013. Data byla převzata z webových stránek společnosti ČEPS, a.s. Pro zdroje tohoto typu nejsou generovány výpadky.



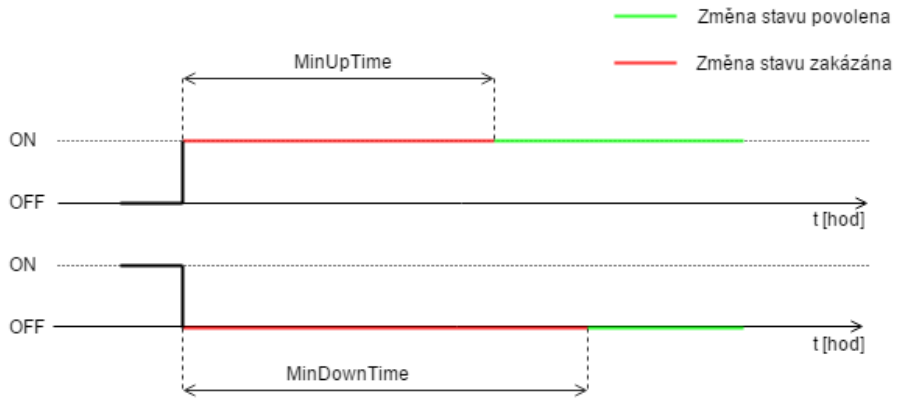
Obrázek 1.2: Příklad dodaného výkonu fotovoltaickými elektrárnami [2].

Omezení na dodanou energii je u obnovitelných zdrojů zpravidla nastaveno na RES_FIX_RUN, kdy výstupní časová řada DelPwr musí přesně odpovídat vstupní řadě ResReqPwr (nelze regulovat jejich výkon), nebo na RES_MAX_RUN, kdy je v případě nadvýroby a dalších splněných požadavků možné snížit velikost výkonu oproti hodnotám ResReqPwr.

1.3 Bloky

Blok je nejmenším a dále nedělitelným prvkem modelu. Důležitými vlastnostmi bloku jsou počáteční stav (počáteční podmínka simulátoru) a schopnost bloku poskytovat kladné točivé rezervy. Dále obsahuje množství parametrů, které se při simulaci využívají k ocenění bloků.

Jsou zde také definována dodatečná omezení na provoz, jako například minimální a maximální elektrický výkon (minPwr a maxPwr), který může blok vyrábět, a nebo omezení na minimální dobu provozu a odstavení, které přesně definuje minimální čas, po který musí nově nasazený blok zůstat v provozu a minimální čas, po který musí zůstat nově odstavený blok mimo provoz. Grafické znázornění poslední uvedené restrikce je uvedeno na obrázku č. 1.3. Tyto parametry jsou v modelu uvedeny jako MinUpTime a MinDownTime. Tuto vlastnost musí simulátor striktně dodržovat. Jediný případ, ve kterém může dojít k jejímu porušení, nastává při výpadku bloku, kdy je blok po detekci poruchy odstaven okamžitě.



Obrázek 1.3: Omezení na minimální dobu provozu a odstavení.

Pro blok jsou v modelu definovány následující časové řady:

1. Vstupní
   - ResReqPwr - Předem definovaná dodaná energie obnovitelného zdroje.
   - Outages - Plánované servisní odstávky.
   - Failures - Uživatelem definované výpadky (pokud nejsou definovány, plánují se generátorem v simulátoru).

2. Výstupní

   - DelPwr - Tato časová řada slouží k uložení výstupního výkonu bloku po provedení regulačních zásahů, pro bloky s omezením na výkon nastaveným na úrovni "FIXED"(popř. "RES_FIX_RUN"nebo "RES_MAX_RUN") je využívána jako vstupní časová řada s požadovanou sekvencí.
   - U - Do této časové řady se ukládá sekvence stavu bloku, jak jej nasadil simulátor, pro bloky s omezením na chod nastaveným na úrovni "FIXED"představuje vstupní požadovanou časovou řadu.

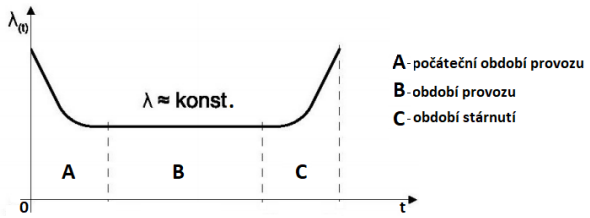
Kapitola 2

Generátor výpadků

Pro analýzu stability přenosové soustavy je vhodné znát její chování v případě výpadků zdrojů elektrické energie. Nejzajímavější jevy představují následující situace. První z nich umožní zkoumání reakce soustavy při náhodných výpadcích všech zdrojů v průběhu celé simulace. Ve druhé situaci dochází ke hledání hodiny, ve které bude mít výpadek konkrétního testovaného zdroje největší dopad na stabilitu. Z tohoto důvodu byl navržen a implementován generátor, který pro každý blok elektrárny a definované časové období vytvoří požadovanou sadu výpadků.

2.1 Model dostupnosti zdroje

Pro vytvoření jednoduchého modelu dostupnosti je nutné zavést několik omezení. Prvním z nich je uvažování všech bloků za provozované v tzv. období provozu. To je časový úsek života prvku, ilustrovaný na obrázku č. 2.1, kdy jsou již odstraněny poruchy vlivem návrhu a výroby a vznikají zde pouze náhodné poruchy. Protože je v elektrárnách prováděna pravidelná údržba, lze považovat intenzitu jejich poruch v období provozu za konstantní. Jako aproximace provozu bloků v období provozu bylo zvoleno exponenciální rozdělení, které taktéž uvažuje konstantní intenzitu poruch a uvažuje tedy pouze vnější náhodné poruchy a zanedbává vliv stárnutí zařízení [3].

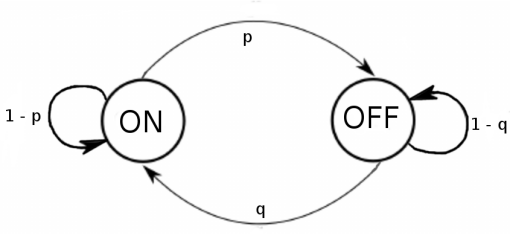


Obrázek 2.1: Vanová křivka popisující úseky života prvků elektrizační soustavy [3]

Ke každému bloku elektrárny jsou ze zadání dodány jako vstupní parametry hodnoty střední doby mezi poruchami (MTBF = Mean Time Before Failure) a střední doby do opravy (MTTR = Mean Time To Repair). Pro vygenerování náhodných výpadků zdrojů bylo potřeba z těchto dat sestavit tzv. model dostupnosti [5]. Takovýto model je možné reprezentovat pomocí Markovova procesu. Řetězec, znázorněný na obrázku č. 2.2, obsahuje pouze dva stavy - první stav (ON) reprezentuje blok v provozu a druhý stav (OFF) blok ve výpadku. Pravděpodobnosti přechodu p a q byly vypočítány pomocí následujících vztahů.

$$p = \frac{1}{MTBF} \qquad q = \frac{1}{MTTR} \tag{2.1}$$

p	Pravděpodobnost nastání výpadku
$1 - p$	Pravděpodobnost pokračování provozu
q	Pravděpodobnost obnovy zdroje
$1 - q$	Pravděpodobnost přetrvání výpadku



Obrázek 2.2: Model dostupnosti zdroje reprezentovaný jako dvoustavový Markovův proces

2.2 Navržený generátor

Pro vytvoření náhodných výpadků byla navržena třída FailGenerator, která umožňuje vygenerovat dva druhy výpadků - náhodné výpadky nebo hodinový výpadek zadaného zdroje, který se v každé variantě posune právě o jednu hodinu. Obě možnosti jsou implementovány jako paralelní cykly, které tak umožňují paralelní generování různých variant výpadků. Jediným vstupním parametrem kontruktoru této třídy je degreeOfParallelism, kterým lze nastavit maximální počet v jednu chvíli běžících vláken ve výše zmíněných cyklech.

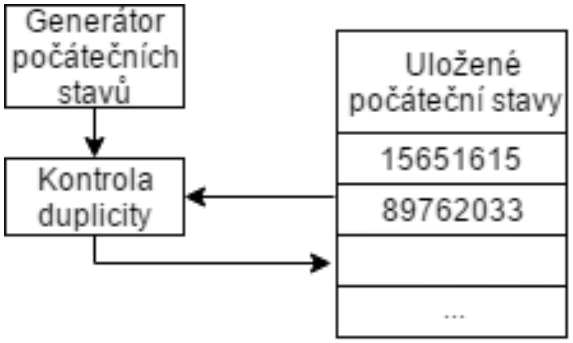
V případě zadání výpadků uživatelem je volána metoda LoadPreplannedFailures, ve které dochází k překopírování těchto výpadků do příslušných simulátorem zpracovávaných struktur. Pokud nejsou výpadky předem zadány, je možné vygenerovat dva následující typy.

2.2.1 Generování náhodných výpadků

Pro vygenerování náhodných výpadků slouží metody třídy GetRandomFailures. Jejími vstupními parametry jsou:

- objekt třídy AnalyzedScDto - je společně s dalšími DTO objekty popsán v následující kapitole,
- numberOfCases - počet požadovaných sad výpadků,
- customSeeds - nepovinný parametr, poskytuje možnost vložit vlastní sadu výchozích stavů generátoru, namísto těch zde generovaných.

V případě, že uživatel nezadá vlastní sadu počátečních stavů (customSeeds), dochází po validaci všech vstupních parametrů k jejich vytvoření. Generují se pomocí knihovního generátoru náhodných čísel. Tyto počáteční stavy jsou unikátní v rámci dané sady a při jejich vytváření se provádí test duplicity. Po převzetí uživatelské sady počátečních stavů nebo jejich vygenerování zde následuje paralelní cyklus, který probíhá přes zadaný počet variant nebo všechny uživatelem zadané počáteční stavy. V cyklu je volána metoda GenerateRandomFailures obsahující sekvenční cyklus, který prochází přes všechny bloky v zadaném scénáři.



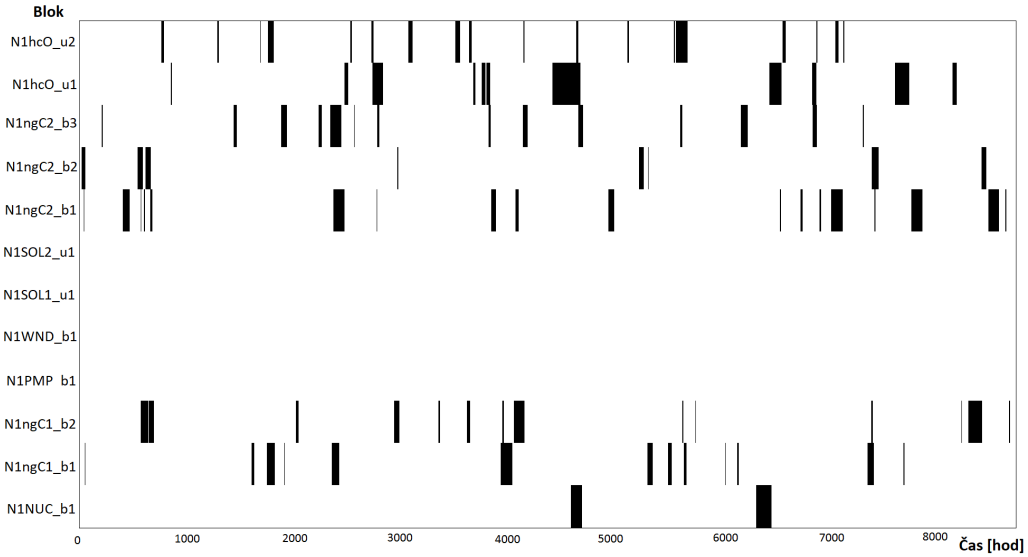
Obrázek 2.3: Princip generování počátečních stavů generátoru

V této variantě generátor respektuje omezení na dodanou energii a chod pro nastavení typu FIXED, tzn. pro bloky v tomto módu se výpadky negenerují, zachovávají se zafixované stavy, a v metodě GenerateRandomFailures je pro ně vygenerováno pole s defaultní hodnotou false. Dále se výpadky také negenerují pro všechny obnovitelné zdroje, nehledě na nastavení jejich restrikcí. Pro všechny ostatní typy zdrojů a jiná nastavení než FIXED se generují výpadky dle navrženého modelu dostupnosti, popsaného obrázkem č. 2.2 a rovnicemi 2.1, zavoláním metody GenerateProcessStates objektu typu SequenceGenerator. Tato metoda je popsána pseudokódem č. 1.

Algoritmus 1 Pseudokód generování sekvencí náhodných výpadků

```
1:  function GenerateProcessStates(mtbf, mttr)
2:    Výpočet pravděpodobností přechodu ze vstupních statistik
3:    for Procházení přes délku intervalu do
4:      Generování pravděpodobnosti přechodu
5:      if Blok je v provozu then
6:        if Pravděpodobnost přechodu > pravděpodobnost výpadku then
7:          Blok je ve výpadku
8:        else
9:          Blok je v provozu
10:       end if
11:     else
12:       if Pravděpodobnost přechodu > pravděpodobnost opravy then
13:         Blok je znovu v provozu
14:       else
15:         Blok zůstává ve výpadku
16:       end if
17:     end if
18:   end for
19: end function
```

Sada výpadků každého zdroje je v každé variantě uložena ve slovníku jako dvojice (řetězec, jednorozměrné pole typu boolean) a tento slovník je uložen v objektu typu FailSeriesDto společně s příslušným počátečním stavem náhodného generátoru pro jednoznačnou identifikaci sady. Výstupem metody GetRandomFailures je pole objektů typu FailSeriesDto (každý člen znamená jednu sadu vygenerovaných výpadků). Vygenerované výpadky jsou zobrazeny na mapě výpadků na obrázku č. 2.4. Černé obdélníky představují blok ve výpadku, bílé provozuschopnost.



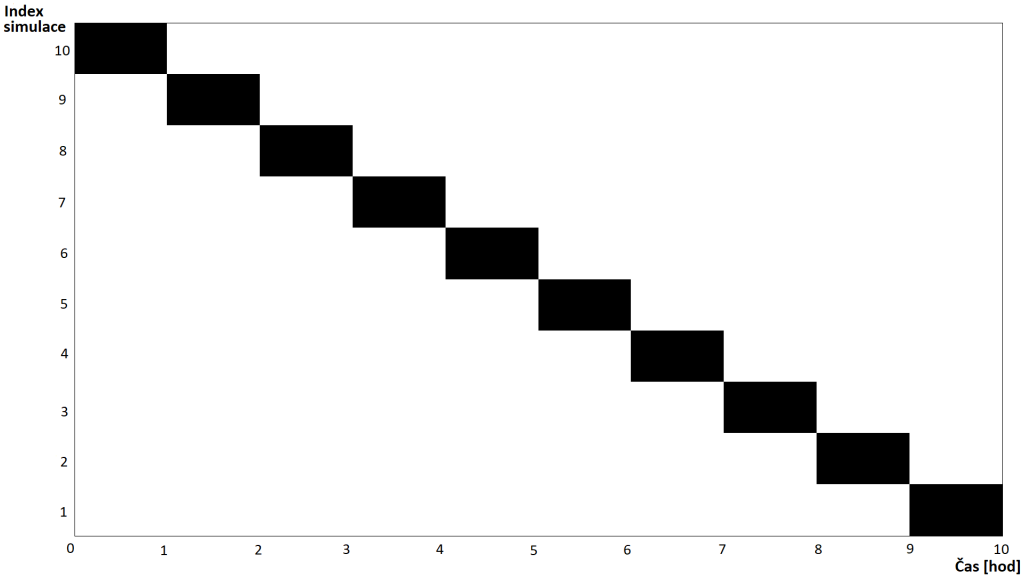
Obrázek 2.4: Mapa jedné vygenerované sady výpadků

2.2.2 Generování postupného výpadku

Za účelem testovaní, ve které konkrétní hodině simulace bude mít vliv výpadku konkrétního zadaného zdroje nejvýznamnější dopad na schopnost dodat požadované množství energie, je navržená metoda GetMovingHourFailures. Vstupní parametry metody tvoří objekt třídy AnalyzedScDto, obsahující parametry simulované sítě a počet generovaných variant, který zároveň definuje i délku generované sady. Úkolem této metody je provést validaci zadaných vstupních dat a spravovat paralelizaci generování výpadku prostřednictvím paralelního cyklu.

V každém průchodu tohoto cyklu je volána metoda GenerateMovingHourFailures, jejímž úkolem je vytvořit stejný slovník výpadků zdrojů, jako ve variantě náhodných výpadků. Avšak zde jsou pole výpadků u všech bloků vyplněny výchozí hodnotou false a pouze u zadaného bloku je v příslušnou hodinu, která je dána indexem průchodu cyklu, nastavena hodnota true (tedy že nastal výpadek). Cílem je analyzovat reakci sítě v konkrétní kritické hodině. Z tohoto důvodu jsou zde generovány výpadky o délce přesně jedné hodiny. Pozorování vlivu delších výpadků již není tak významné, protože s každou další hodinou je množina nasaditelných zdrojů rozšiřována o další bloky, které nově splňují omezení na minimální dobu provozu a odstavení a tím klesá dopad vlivu výpadku.

Výstupem této metody je, stejně jako v případě generování náhodných výpadků, instance třídy FailSeriesDto. Tento objekt je uložen do pole na pozici, která je dána indexem průchodu cyklu. Tím je zaručeno správné ukládání vygenerovaných sad výpadků při paralelním zpracování. Toto pole je vráceno metodou GetMovingHourFailures jako výstup generátoru.

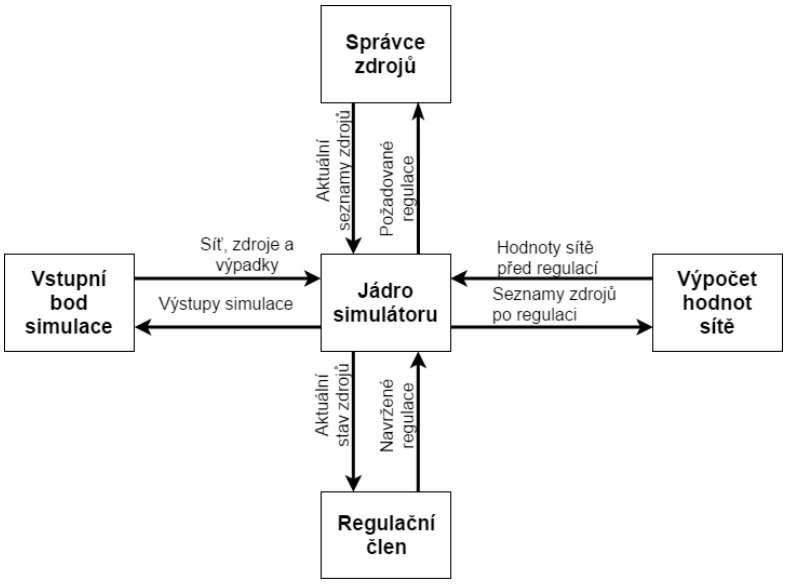


Obrázek 2.5: Vzor výpadku nejvýznamnějšího zdroje napříč vygenerovanými sadami

Kapitola 3

Simulátor skladby zdrojů

Pro analýzu vlivu výpadků na bilanci elektrizační soustavy byl navržen a implementován simulátor skladby zdrojů, který pro zadaný časový interval simuluje idealizované chování elektráren připojených do jednoho uzlu přenosové soustavy a reaguje na potřeby pokrytí zatížení a vygenerovaných výpadků jejich vhodnou skladbou. Simulátor je navržen tak, aby bylo možné simulovat paralelně více variant výpadků jednoho scénáře. Pracuje s referencemi na načtená vstupní data, která se během simulace neupravují, pouze se z nich čtou potřebné parametry. O zmíněnou paralelizaci se stará metoda RunSimulation třídy Simulation, která obsahuje paralelní cyklus, ve kterém vytváří instanci třídy SimulationCore, spouští simulaci dané varianty a po jejím skončení uloží výstupy simulace do výstupního objektu. Simulátor se skládá z jádra a tří modulů, které poskytují potřebné služby.



Obrázek 3.1: Struktura simulátoru

Jádro simulátoru tvoří metoda SimulateCase třídy SimulationCore. Tato metoda obsahuje cyklus, který probíhá přes požadovanou délku simulace s krokem rovným jedné hodině. V každé hodině probíhá stejná sekvence úkonů, popsaných pseudokódem č. 2.

Algoritmus 2 Pseudokód jádra simulátoru

```
1:  function SimulateCase
2:    for Přes simulovaný interval do
3:      Aktualizace stavu zdrojů (vypadlé, nasaditelné, odstavitelné, ...)
4:      Výpočet a uložení současných hodnot uzlu (load, delPwr, ...)
5:      Zrušení regulací na zdroje s omezením SOFT_MAX_RUN a RES_MAX_RUN
6:      Výpočet současných hodnot uzlu po zrušení regulací obnovitelných zdrojů
7:      Zajištění pokrytí zatížení a rezerv (regulace)
8:      Uložení změn skladby zdrojů a jejich výkonů
9:      Výpočet nových hodnot uzlu
10:   end for
11: end function
```

3.1 Správce zdrojů

Správce zdrojů má na starosti udržovat zdroje rozdělené do daných skupin v podobě seznamů a tyto seznamy pravidelně (každý simulační krok) aktualizovat. Pro tuto činnost byla implementována třída Sources. Jako vstupní parametry tato třída vyžaduje seznam všech zdrojů simulovaného uzlu a slovník výpadků daných zdrojů. Při vytvoření instance této třídy se v konstruktoru volá metoda InitiateRunningSources, která provede prvotní roztřídění bloků na nedostupné zdroje a zdroje v chodu.

3.1.1 Rozdělení zdrojů

Všechny bloky v simulátoru jsou rozděleny do pěti skupin podle diagramu na obrázku č. 3.2. Je patrné, že zdroj se může nacházet i ve více kategoriích v jeden čas. Tyto skupiny jsou v knihovně implementovány jako seznamy. Způsob zacházení s blokem je definován typem seznamů, ve kterých se právě nachází.

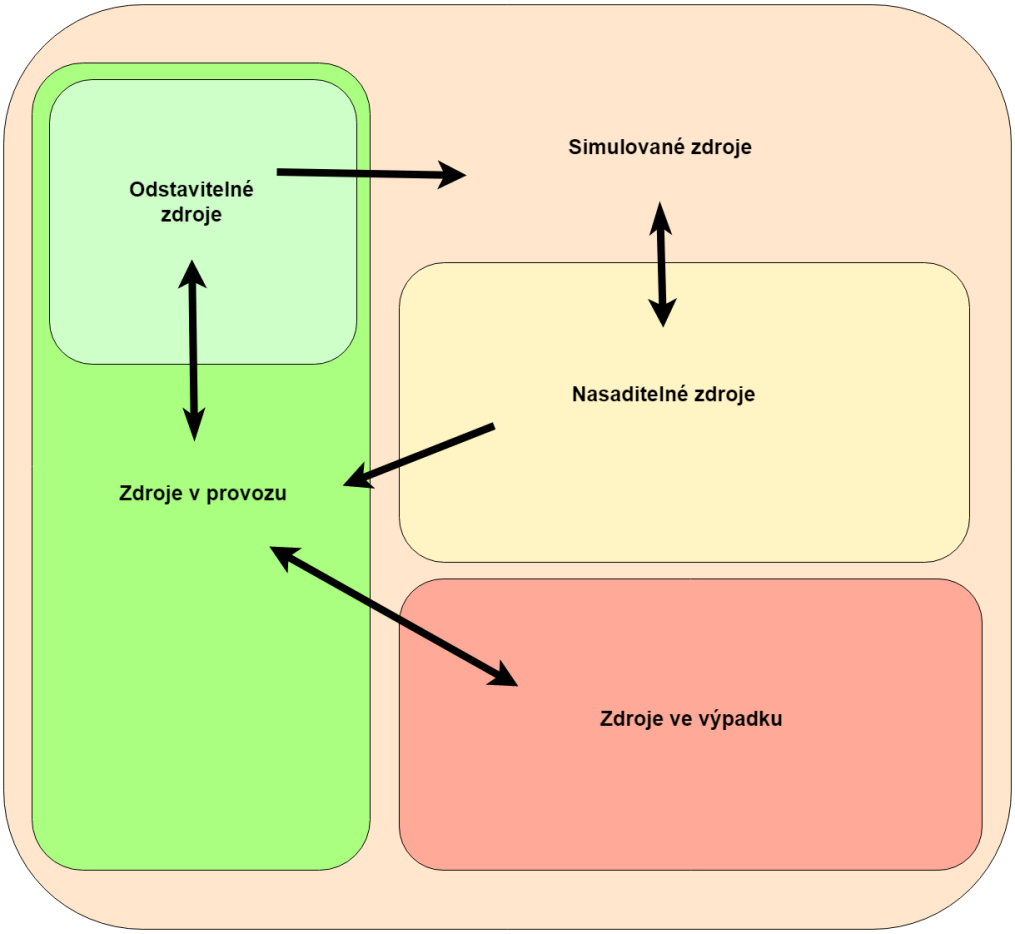
1. **Bez kategorie**
   Zdroje v této kategorii nejsou obsaženy v žádném seznamu, obecně se jedná o odstavené zdroje, které v danou hodinu nelze nasadit (z důvodů poruchy nebo omezení na minimální počty hodin provozu a odstavení).

2. **Nasaditelné**
   Jsou to odstavené zdroje, které nejsou ve výpadku, splňují omezení na minimální počty hodin provozu a odstavení, je možné je nasadit do provozu.

3. **V provozu**
   Zdroje, které se aktivně podílí na pokrytí zatížení v uzlu.

4. **Odstavitelné**
   Zdroje v provozu, které splňují omezení na minimální počty hodin provozu a odstavení, nejsou ve výpadku a je možné je odstavit.

5. **Porouchané**
   Zdroje, na kterých nastala během chodu porucha. Po opravení poruchy jsou automaticky vráceny zpět do provozu.



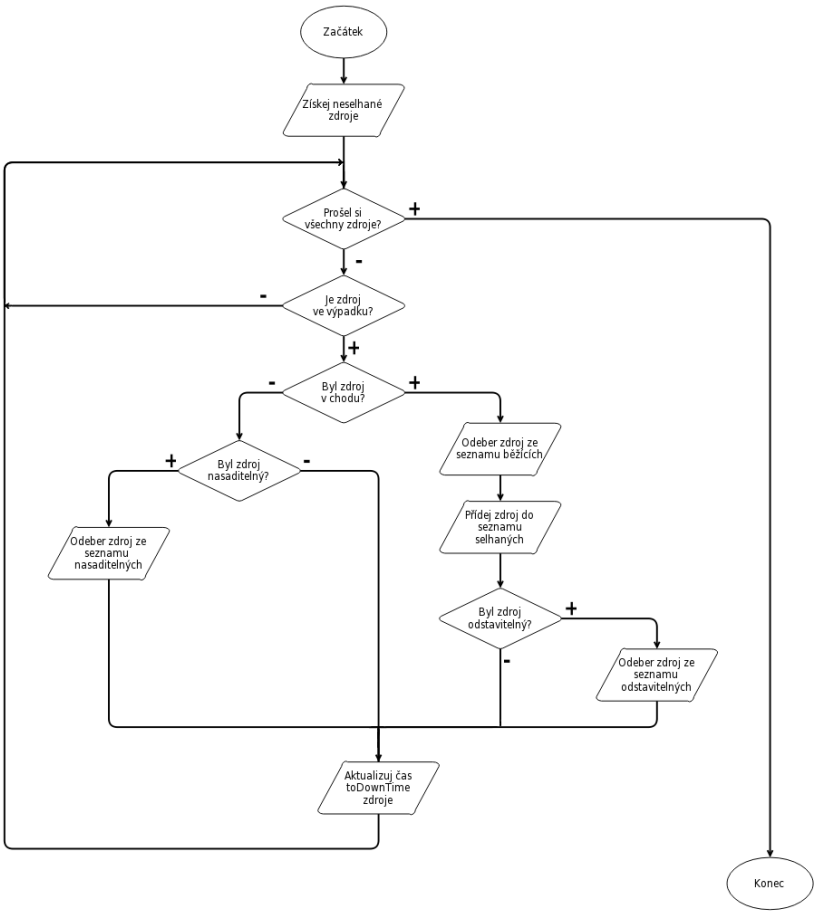
Obrázek 3.2: Diagram rozdělení zdrojů do seznamů

3.1.2 Mezihodinová aktualizace zdrojů

Pro aktualizaci stavu všech zdrojů simulace v každé hodině slouží metoda UpdateSources, která provádí aktualizaci jednotlivých seznamů.

1. **Aktualizace zdrojů ve výpadku.**
   Skládá se ze dvou částí, uvedení opravených zdrojů zpět do provozu a odstavení nově porouchaných zdrojů.

   V první části (metoda SendSourcesToOutage) projde v cyklu všechny bloky v seznamu vypadlých zdrojů a zkontroluje příslušnou časovou řadu výpadků, zda již výpadek neodezněl. Pokud výpadek odezněl, odstraní daný zdroj z porouchaných zdrojů, převede jej zpět do seznamu běžících zdrojů a aktualizuje čas přechodu bloku do chodu (toUpTime). Bližší popis metody je znázorněn na vývojovém diagramu na obrázku č. 3.3.



Obrázek 3.3: Vývojový diagram metody SendSourcesToOutage

V druhé části (metoda RecoverRepairedSources) se v cyklu projdou všechny bloky, které nejsou v seznamu vypadlých zdrojů. Pokud je testovaný blok v seznamu běžících popř. odstavitelných zdrojů, je z těchto seznamů odstraněn a přidán do seznamu zdrojů ve výpadku. V případě, že je testovaný blok odstavený, zjistí se, zda není v seznamu nasaditelných zdrojů. Pokud ano, je z něj odstraněn. V obou případech je aktualizován čas přechodu bloku do odstávky (toDownTime).

2. **Aktualizace nasaditelných zdrojů.**
Zde, v metodě UpdateDeployableSources, se v cyklu prochází všechny zdroje, které:

   a) nejsou v seznamu běžících zdrojů,
   b) nejsou již v seznamu nasaditelných zdrojů,
   c) nejsou v poruše,
   d) jejich omezení na chod nebo na výkon není nastaveno na úroveň "FIXED".

   U těchto zdrojů se otestuje, zda již neuběhl povinný minimální čas odstávky MinDownTime (zda je splněno omezení na minimální dobu provozu a odstavení zdroje). Pokud tento čas uplynul, je blok přidán do seznamu nasaditelných zdrojů.

3. **Aktualizace odstavitelných zdrojů.**
O tuto činnost se stará metoda UpdateAdjustableSources, skládající se ze dvou cyklů. V prvním z nich jsou procházeny všechny běžící bloky, které již nejsou v seznamu odstavitelných zdrojů a nejedná se o obnovitelné zdroje. Aby mohl být testovaný blok zařazen do odstavitelných zdrojů, musí splňovat omezení na minimální dobu provozu (MinUpTime) a jeho omezení na chod musí být nastaveno na jednu z následujících úrovní:

   a) EC (Economy),
   b) PART_ALW_RUN (kdy navíc musí zároveň dodržovat toto omezení).

   V druhé fázi dochází ke kontrole dodržování výše zmíněného omezení na chod úrovně PART_ALW_RUN. V cyklu jsou testovány všechny odstavitelné zdroje s tímto omezením.

   Jestliže tento seznam odstavitelných zdrojů obsahuje takové množství bloků příslušné elektrárny, že by odstavení všech těchto bloků vedlo k porušení restrikce, jsou z tohoto seznamu postupně odebírány. Je ponechán pouze takový počet bloků příslušné elektrárny, aby byl i po jejich odstavení zachován minimální požadovaných počet.

4. **Aktualizace zdrojů s fixními stavy nebo výkony.**
Tato část se týká všech bloků, jejichž omezení na chod nebo výkon je nastaveno na úroveň "FIXED"(popř. RES_FIX_RUN nebo RES_MAX_RUN pro výkon obnovitelných zdrojů). Je důležité zmínit, že ani jedna z následujících dvou metod nekontroluje omezení na minimální dobu chodu a odstavení, jejich dodržení je na autorovi zadaných časových řad.

Pro aktualizaci zdrojů s fixovanými stavy slouží metoda UpdateFixedStates. V cyklu prochází všechny příslušné bloky a podle časové řady U mění jejich umístění v seznamech zdrojů. Metoda UpdateFixedPowerSources aktualizuje tabulku výkonů běžících zdrojů s fixovanými výkony podle časové řady ResREqPwr (pro obnovitelné zdroje), nebo DelPwr (pro všechny ostatní typy). Zároveň aktualizuje i jejich umístění v seznamech, při nulovém výkonu se zdroj považuje za odstavený, naopak při výkonu $P > 0$ se považuje za běžící. Přehledný popis je uveden v pseudokódu č. 3.

Algoritmus 3 Pseudokód metody UpdateFixedPowerSources

```
1:  function UpdateFixedPowerSources(fixedPowerSources, actualTime)
2:    for přes seznam fixedPowerSources do
3:      if Aktualizovaný blok je v seznamu běžících zdrojů then
4:        Aktualizuj výkon bloku ve slovníku výkonů běžících zdrojů
5:      else if Blok není v sezamu běžících zdrojů then
6:        if Požadovaný výkon P > 0 then
7:          Přidej blok do seznamu běžících zdrojů
8:          Aktualizuj výkon bloku ve slovníku výkonů běžících zdrojů
9:          Aktualizuj čas toUpTime bloku
10:       end if
11:     end if
12:   end for
13: end function
```

5. **Aktualizace zdrojů s omezením PART_ALW_RUN a MIN_RUN.**
Pro zajištění splnění omezení na chod úrovně PART_ALW_RUN a omezení na výkon úrovně MIN_RUN byla navržena metoda EnsureMustRunRestrictions. V cyklu jsou procházeny všechny simulované elektrárny a u těch, které mají toto nastavení, se testuje, zda jsou omezení dodržována. V případě, že testovaná elektrárna s nastavením PART_ALW_RUN nedodržuje minimální počet běžících bloků nebo elektrárna s nastavením MIN_RUN nedodržuje minimální požadovaný výkon, jsou ze seznamu nasaditelných zdrojů vybrány všechny bloky testované elektrárny (pokud se v tomto seznamu nachází, pokud ne, elektrárna se přeskočí) a postupně jsou po jednom uváděny v chod, dokud toto omezení (jedno nebo druhé podle daného typu) není splněno.

3.1.3 Aktualizace regulovaných zdrojů

Pro uložení změn zdrojů, provedených v regulačním členu, byla navržena metoda ManageSources. Jako vstupní parametry přijímá aktuální čas simulace a objekt třídy UpdatedSourcesDto (popsán později). Metoda postupně volá následující tři metody.

- SetUpSources
  Tato metoda se stará o všechny bloky, na které byl regulačním členem vznesen požadavek na nasazení. Pokud je takovýto blok v seznamu nasaditelných zdrojů (v případě, že není, je vyvolána výjimka - v regulačním členu došlo k manipulaci s nedostupným blokem), provede následující operace:

  a) přidání zadaného bloku do seznamu běžících zdrojů,
  b) odebrání bloku ze seznamu nasaditelných zdrojů,
  c) aktualizace času jeho náběhu,
  d) vytvoření záznamu ve slovníku běžících zdrojů.

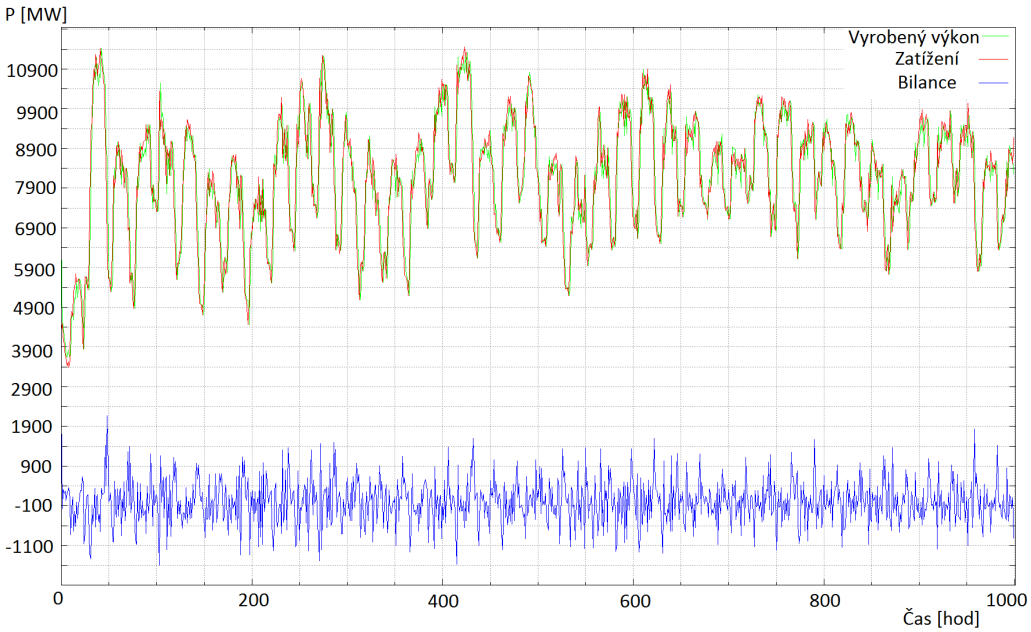
- ShutDownSources
  Všechny bloky, které jsou regulačním členem označeny k odstavení, jsou zpracovány touto metodou. Pokud je takto označený blok v seznamu odstavitelných zdrojů (v případě, že není, je stejně jako v předchozím bodu vyvolána výjimka - v regulačním členu byl vznesen požadavek na odstavení bloku, který být odstaven nesmí), je odebrán ze všech seznamů, je mu aktualizován čas odstavení toDownTime a také dojde k odebrání jeho záznam ze slovníku výkonů běžících zdrojů.

- ManageSourcesPower
  Pro aktualizaci slovníku výkonů všech (i nově) běžících zdrojů slouží metoda ManageSourcesPower. Skládá se z cyklu, v němž jsou procházeny všechny klíče slovníku aktualizovaných bloků s novými hodnotami výkonů. Po otestování, zda je upravený blok obsažen v seznamu běžících zdrojů, je jeho nová hodnota výkonu přepsána do slovníku výkonů všech běžících zdrojů objektu třídy Sources.

3.2 Regulační člen

Nejdůležitější částí simulátoru je třída Regulations. Metody této třídy zajišťují, aby bylo pokryto zatížení uzlu a požadované rezervy. V ideálním případě by po zásahu regulačního členu měla být bilance uzlu rovna nule (zatížení je pokryto a nevzniká ani žádná nadbytečná výroba) bez porušení požadovaných rezerv. Vstupními parametry konstruktoru této třídy je seznam simulovaných elektráren, počáteční hodnota pro výběr kandidátů a proměnná, která udává, zda má být zohledněn faktor náhody. Podrobný význam všech těchto vstupních parametrů je vysvětlen později.

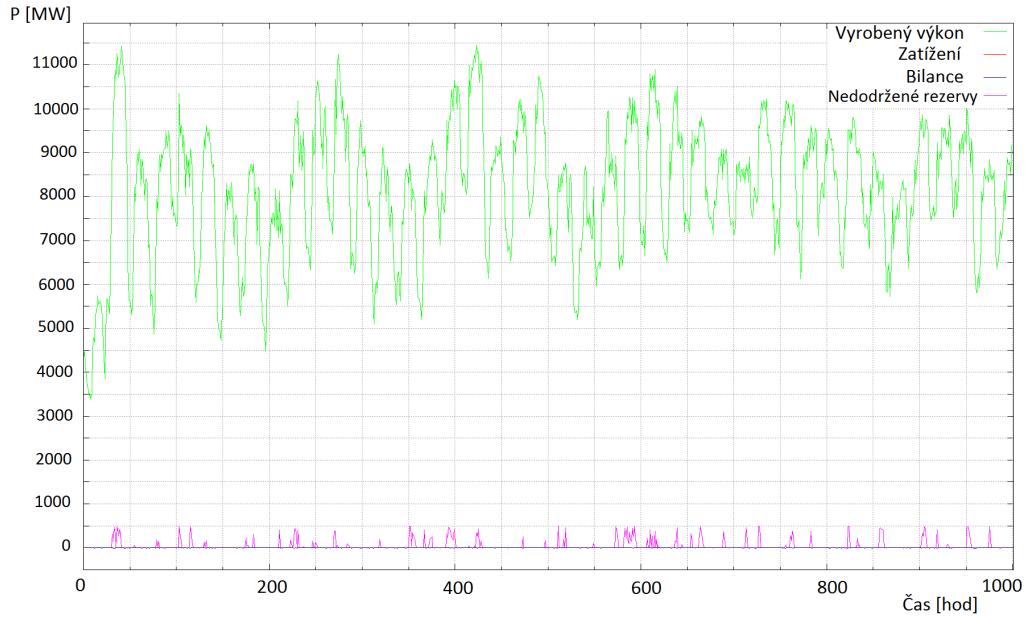
Funkčnost tohoto členu je zobrazena na grafech na obrázcích č. 3.4 a 3.5. Obrázek č. 3.4 popisuje stav bilance před provedením regulačních zásahů. Z průběhu bilance (tmavě modrá) je patrné, že v naprosté většině simulačních kroků dochází ke vzniku nedostatečné výroby nebo nadvýroby.



Obrázek 3.4: Stav uzlu zaznamenaný před zásahem regulačního členu

Z grafu na obrázku č. 3.5, který popisuje situaci po zásahu regulačního členu, je patrné, že bilanci (tmavě modrá) se podařilo v celé délce simulace (1000 hodin) regulačním členem uregulovat na nulu, oproti původnímu průběhu (obrázek č. 3.4).

Také je vidět, že ne v každé hodině se podařilo zajistit pokrytí požadovaných rezerv (velikost nedodržených rezerv je v grafu růžově).



Obrázek 3.5: Stav uzlu zaznamenaný po zásahu regulačního členu

Pro provedení regulačních zásahů je z jádra simulátoru volána metoda ManageActualBalance, ve které se testuje bilance a podle ní se rozhoduje, zda se bude navyšovat nedostatečná výroba nebo eliminovat nadbytečná výroba a také, jestli bude prováděn pokus o pokrytí požadovaných rezerv.

3.2.1 Pokrytí zátěže při nedostatečné výrobě

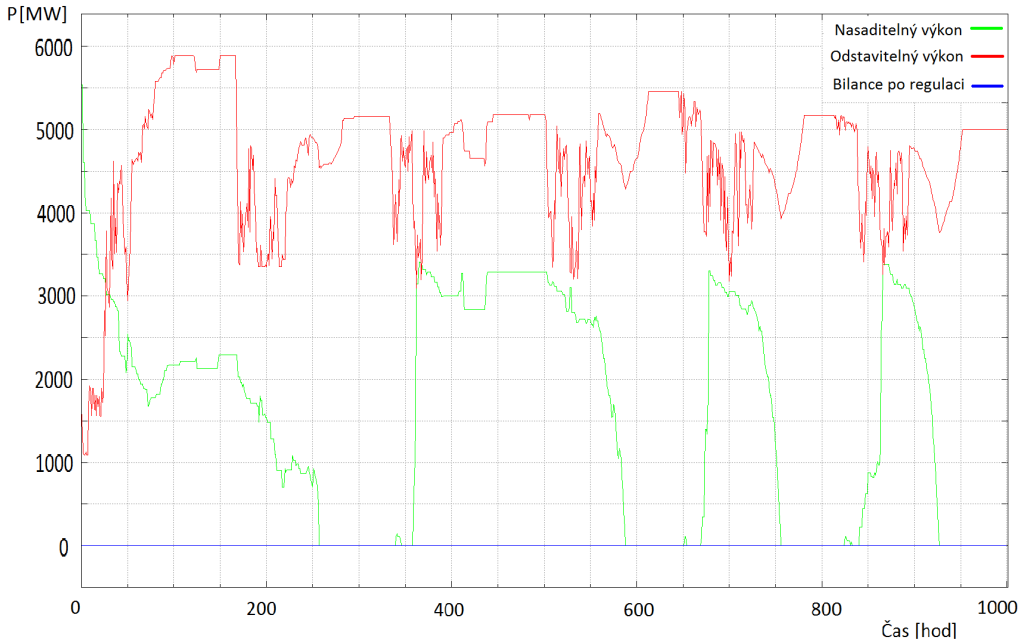
Při vzniku nedostatečné výroby vůči aktuálnímu zatížení (záporná bilance) je volána metoda CoverInsufficientProduction. Jako vstupní parametry metoda přijímá seznam nasaditelných zdrojů, seznam aktuálně běžících zdrojů, slovník jejich aktuálních výkonů, pole s aktuálními hodnotami dodávaného výkonu, zatížení v uzlu, bilance, požadovaných rezerv (actualNodeStats - přesný popis pole je uveden v tabulce č. 3.1) a aktuální čas simulace. Její úkolem je zajistit dostatečné zvýšení součtu výkonů běžících zdrojů. K tomu má dvě možnosti, buď zvýšit výkon (pokud je to možné) na aktuálně běžících zdrojích anebo nasadit (opět pokud je to možné) nové zdroje.

Prioritu má v tomto případě zvyšování výkonu již běžících zdrojů, protože nasazení odstaveného zdroje přináší další výdaje. V prvním kroku je tedy zvyšován výkon na zdrojích, jejichž omezení na výkon je nastaveno na úrovně EC

0	Aktuálně dodávaný výkon do uzlu všemi běžícími zdroji
1	Současné zatížení v uzlu
2	Aktuální bilance uzlu (dodávaný výkon - zatížení)
3	Velikost požadovaných rezerv

Tabulka 3.1: Popis pole hodnot uzlu actualNodeStats

a MIN_RUN (u zdrojů jiných úrovní již výkon zvýšit nelze). K provedení tohoto úkonu je volána metoda IncreasePower. Pokud nelze zvýšit výkon na žádných běžících zdrojích s tímto nastavením nebo byl volný výkon nedostatečný k velikosti záporné hodnoty bilance, přejde metoda k druhému kroku. Tím je uvedení do provozu potřebné množství nasaditelných bloků a to zavoláním metody ManageUnitsToDeploy. Velikost nasaditelného a odstavitelného výkonu je uvedena na grafu na obrázku č. 3.6.



Obrázek 3.6: Graf nasaditelného a odstavitelného výkonu během simulace

Pokud se podaří nasadit dostatečné množství zdrojů, aby byla bilance kladná, testuje se, zda není větší než nula. Pokud ano, provede se pokus o její korekci na nulu. Ten se skládá z cyklu, který má definovaný maximálně dva průchody. V obou průchodech je volána metoda DecreasePower, které je ale předáván v každém průchodu jiný seznam zdrojů. Při prvním průchodu jsou předány bloky, jejichž omezení na výkon je nastaveno na úrovně EC a MIN_RUN. V druhém průchodu je předán seznam bloků s omezení na výkon úrovní SOFT_MAX_RUN a RES_MAX_RUN. Důvodem tohoto pořadí jsou penalizace za snižování výkonu zdrojů s nastavením na SOFT_MAX_RUN a RES_MAX_RUN zatímco výkon zdrojů nastavených na úrovně EC a MIN_RUN můžeme při dodržení příslušných omezení snižovat bezplatně. Výstupem metody je zaktualizované pole hodnot uzlu

actualNodeStats, upravované všemi službami, které byli volány. Výše zmíněné metody IncreasePower, ManageUnitsToDeploy a DecreasePower jsou blíže popsány v sekci "Využívané služby".

3.2.2 Pokrytí zátěže při nadbytečné výrobě

V případě, že je zatížení v uzlu plně pokryto, ale vzniká nadvýroba (bilance je větší než nula), je volána metoda CoverOverProduction. Vstupními parametry metody jsou seznam nasaditelných zdrojů, seznam právě běžících zdrojů, slovník jejich aktuálních výkonů, pole s aktuálními hodnotami uzlu (actualNodeStats) a aktuální čas simulace. Tato metoda má za úkol tuto nadbytečnou výrobu eliminovat snížením sumy výkonů právě běžících zdrojů.

V první části je proveden pokus o odstavení bloků ze seznamu odstavitelných bloků. Tím chceme snížit velikost nadbytečné výroby a zároveň zmenšit počet běžících bloků a tím snížit cenu provozu. K tomuto účelu je zavolána metoda ManageUnitsToAdjust, které je tento seznam s dalšími parametry předán. Pokud nelze odstavit žádné zdroje, aniž by došlo k přechodu bilance do záporných hodnot, nebo je-li i po odstavení určitého množství zdrojů bilance stále větší než nula, přechází metoda do druhé části. Tato část má stejnou strukturu jako v případě metody CoverInsufficientProduction. Také se skládá z cyklu, který prochází pole, obsahující upravené seznamy běžících zdrojů, které jsou předány volané metodě DecreasePower. Bližší popis je uveden pomocí pseudokódu č. 4.

Algoritmus 4 Pseudokód metody CoverOverProduction

```
1:  function CoverOverProduction(adjustableSources, runningSources, runningSourcesPower, actualNodeStats, actualTime)
2:    if Pokud není seznam odstavitelných zdrojů prázdný then
3:      Pokus se odstavit vhodné zdroje voláním metody ManageUnitsToAdjust
4:    end if
5:    Získej běžících zdroje s omezením na chod typu EC nebo MIN_RUN.
6:    Ulož získaný seznam do prvního místa pole.
7:    Získej běžící zdroje s omezením na chod typu SOFT_MAX_RUN nebo RES_MAX_RUN.
8:    Ulož druhý získaný seznam do druhého místa pole.
9:    for Přes velikost pole do
10:     if Pokud není seznam v i místě pole prázdný then
11:       Sniž výkon vybraných zdrojů voláním metody DecreasePower
12:     end if
13:   end for
14: end function
```

Výstupem této metody je také zaktualizované pole hodnot uzlu actualNodeStats, upravované všemi službami, které byly volány. V ideálním případě je dodávaný výkon roven zatížení v uzlu a bilance je nulová. Výše zmíněné metody ManageUnitsToAdjust a DecreasePower budou blíže popsány v sekci 3.2.4.

3.2.3 Pokrytí povinných rezerv

Tento model pracuje s kladnými točivými rezervami. Pro jejich zajištění byla navržena metoda EnsurePositiveReserves. Jejím úkolem je udržet dostatečný volný výkon na běžících zdrojích, které podporují rezervy. V případě, že není dostatek volného výkonu k pokrytí požadovaných rezerv, nasadí tato metoda nutný počet zdrojů a to tak, aby nebyla změněna hodnota bilance z předchozích regulačních zásahů. Může nastat případ, kdy nebude možné zajistit dostatečné množství volného výkonu k pokrytí požadovaných rezerv (např. v danou hodinu nebude možné nasadit další zdroje, podporující rezervy). V takové situaci vznikají nedodržené rezervy, které jsou penalizovány a sníží tak kvalitu řešení simulace.

Metoda EnsurePositiveReserves je tvořena cyklem, který prochází přes následující seznamy zdrojů:

1. Zdroje, které byly dosud v provozu, jsou určeny k odstavení a podporují rezervy.
2. Zdroje, které jsou nasaditelné, není dosud požadováno jejich nasazení a podporují rezervy.

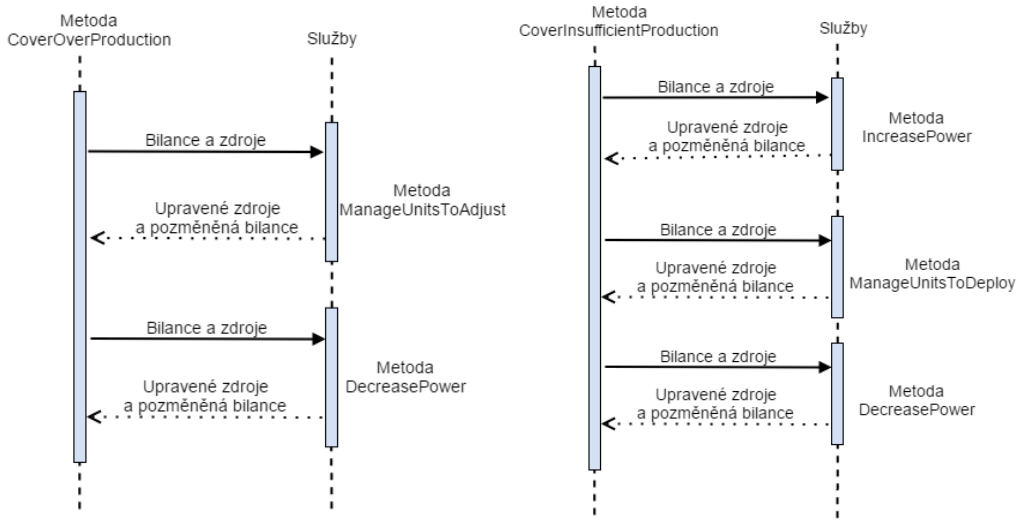
Cyklus v každém průchodu vybere příslušný seznam a provede sekvenci akcí:

1. Výpočet aktuálních rezerv na běžících zdrojích. Ty jsou vypočteny jako součet maximálních výkonů bloků, které podporují rezervy, mínus suma jejich aktuálního výkonu.
2. V případě nedostatečně pokrytých rezerv, nasazení dostatečného počtu zdrojů z obou seznamů (k tomuto účelu je volána metoda ManageUnitsToDeploy).
3. Korekce bilance pomocí redukce výkonu na běžících zdrojích. Bilanci je nutné doregulovat na původní hodnotu. Tento krok opět obsahuje cyklus, který prochází pole, obsahující dva seznamy zdrojů. První z nich obsahuje nasazené bloky, jejichž elektrárny, do do kterých tyto bloky náleží, mají

omezení na výkon nastaveno na úrovně EC nebo MIN_RUN. Druhý seznam obsahuje běžící bloky, jejichž elektrárny mají omezení na výkon nastaveno na úrovně SOFT_MAX_RUN a RES_MAX_RUN. K samotnému snižování výkonu je zavolána metoda DecreasePower, již jsou tyto seznamy postupně předány.

3.2.4 Využívané služby

Pro návrh samotných změn ve skladbě bloků v seznamech a výkonech jednotlivých zdrojů byly navrženy čtyři základní metody, které jsou volány jako služby ve výše popsaných případech. Každá z nich pracuje s heuristikou, uvedenou v následující sekci. Sekvence jejich volání při nadvýrobě i nedostatečné výrobě jsou uvedeny na obrázku č. 3.7.



Obrázek 3.7: Diagram volání služeb

Navržená heuristika a selekce kandidátů

Aby bylo možné provést cenově optimální výběr zdrojů (při nasazování, odstavování, snižování nebo zvyšování výkonu), bylo nutné navrhnout vhodnou ztrátovou funkci. Funkce, která je použita v simulátoru, byla částečně převzata z [6] a následně upravena pro tento model soustavy. Zohledňuje následující náklady, vzniklé provozem (nebo změnou provozu) bloku:

c_{fix}	Fixní cena bloku v €/hod
c_{var}	Variabilní cena za 1 vyrobený MW
c_{ems}	Cena produkovaných emisí za 1 MWh_t
c_{fuel}	Cena propáleného paliva za 1 MWh_t
c_{ud}	Cena v €za start/odstavení zdroje

Výpočet této funkce, uvedené v rovnici č. 3.1, se pro jednotlivé služby liší členem $k * c_{ud}$, který zohledňuje změnu stavu chodu bloku. Parametr $k \in \{-1, 0, 1\}$ je konstanta, která říká, zda započítat cenu za nasazení/odstavení bloku a jestli ji přičítat nebo odčítat.

$$L(c) = c_{fix} + c_{var} + c_{ems} + c_{fuel} + k * c_{ud} \tag{3.1}$$

$$\text{kde } k = \begin{cases} +1 & \text{v případě nasazování zdrojů} \\ -1 & \text{při odstavování bloků} \\ 0 & \text{jinak (pokud se pouze mění výkon běžících zdrojů)} \end{cases}$$

Výběr kandidátů k provedení dané změny probíhá v každé službě zvlášť, ale postup je velice podobný, proto bude popsán jednotně zde. Stejně jako použitá funkce byl i postup selekce částečně převzat z [6]. Ve všech případech je nejdříve volána metoda GetSourcesCosts, která obsahuje navrženou heuristiku a provede ohodnocení všech zdrojů v seznamu, který ji daná služba předá jako vstupní parametr. Bloky s jejich ohodnocením jsou vráceny jako slovník, obsahující jméno daného bloku a jeho cena jako typ double.

a) Selekce kandidátů na nasazení nebo zvýšení jejich výkonu
V tomto případě je cílem vybrat bloky tak, abychom dosáhli potřebného navýšení produkce, ale takové, jejichž celková provozní cena je co nejmenší.

$$\arg\min_{u} L \tag{3.2}$$

Ohodnocené zdroje se vybírají po skupinách pomocí cyklu, který prochází přes proměnnou $\alpha_{inc} \in [\alpha_{init}; 10]$, kde α_{init} konstanta, která udává velikost první skupiny kandidátů a je zadána uživatelem jako vstupní parametr konstruktoru třídy Regulations. V každém dalším průchodu se tato proměnná zvýší o 1. Horní hranice cen zdrojů v daném průchodu je dána vzorcem č. 3.3. Proměnná c_{max} je největší cena zdrojů, ohodnocených ztrátovou funkcí a c_{min} je nejmenší cena těchto zdrojů.

$$\alpha_{up} = c_{min} + \alpha_{inc} \cdot (c_{max} - c_{min}) \tag{3.3}$$

b) Selekce kandidátů na odstavení nebo snížení jejich výkonu
Na rozdíl od prvního případu je nyní cílem výběr nejdražších bloků, což vede ke snížení celkové ceny produkce.

$$\arg\max_{u} L \tag{3.4}$$

Stejně jako v předchozím případě, probíhá výběr zdrojů po skupinách přes výše popsaný cyklus, avšak rozdíl nastává ve výpočtu hranice (v tomto případě spodní). Ta je dána vzorcem č. 3.5.

$$\alpha_{up} = c_{max} - \alpha_{inc} \cdot (c_{max} - c_{min}) \tag{3.5}$$

Důležitou volbou je možnost zahrnout do procesu výběru faktor náhody. Ten se zavádí pomocí proměnné randomFactor, která je zadána jako vstupní parametr konstruktoru třídy Regulations. Pokud je požadován faktor náhody, probíhá výběr generováním náhodného čísla (v intervalu od 1 do počtu prvků v seznamu kandidátů), které udává pořadí vybraného bloku. V opačném případě je vybrán vždy první blok v daném seznamu (po selekci je vybraný blok z tohoto seznamu odebrán).

Metoda IncreasePower

Pro zvýšení výkonu již běžících zdrojů při nedostatečné výrobě je volána metoda IncreasePower, které je jako vstupní parametry předán seznam běžících zdrojů, jejich současný výkon, aktuální stav uzlu a čas simulace. Jak již bylo zmíněno, v prvním kroku dojde k ohodnocení zdrojů, které provádí metoda GetSourcesCosts. V tomto případě je této metodě předán seznam právě běžících zdrojů (chceme navýšit výkon za co nejmenší nárůst ceny). Dále se provádí selekce kandidátů. V okamžiku, kdy je vhodný kandidát vybrán, provede se přepočet chybějícího výkonu k pokrytí zatížení a vyčte se současný výkon tohoto bloku. Obě tyto hodnoty jsou předány společně s instancí vybraného kandidáta a proměnnou hasUndelivered, která říká, zda se jedná o případ nedostatečné výroby nebo nadvýroby, jako vstupní parametry metodě GetNewRunningPower.

Úkolem této metody je zajistit respektování omezení na minimální a maximální výkon bloku, tedy aby nově nastavený nenulový výkon bloku byl vždy pouze v intervalu $[minPwr; maxPwr]$. Kromě metody IncreasePower je ke stejnému účelu volána také z metod DecreasePower a ManageUnitsToDeploy. Bližší popis metody je znázorněn pomocí pseudokódu č. 5.

Po výpočtu nové hodnoty výkonu zdroje je tato hodnota otestována, zda byl výkon zdroje změněn. Pokud ano, je starý výkon ve slovníku výkonů běžících zdrojů přepsán novou hodnotou a také je zaktualizováno pole hodnot uzlu. Následně je zpracovávaný blok odstraněn ze seznamu skupiny kandidátů a přidán do seznamu již zpracovaných bloků (aby při příštím výběru již nebyl zařazen mezi možné kandidáty).

Algoritmus 5 Pseudokód metody GetNewRunningPower

```
1:  function GetNewRunningPower(selectedUnit, requiredPower, oldRunningPower, hasUndelivered)
2:    if Výroba nedostatečná then
3:      if Chybějící výkon je větší než volný výkon zdroje then
4:        Nový výkon zdroje je roven jeho maximálnímu možnému výkonu bloku
5:      else if Chybějící výkon < minimální možný výkon zdroje a aktuální
              výkon je roven nule
        then
6:        Nový výkon zdroje je roven minimálnímu možnému výkonu bloku
7:      else
8:        Nový výkon je zdroje je roven součtu současného a chybějícího výkonu
9:      end if
10:   else
11:     if Přebytečný výkon je větší než snížitelný výkon na zdroji then
12:       Nový výkon zdroje je roven minimálnímu možnému výkonu bloku
13:     else
14:       Nový výkon zdroje je roven rozdílu současného a přebytečného výkonu
15:     end if
16:   end if
17: end function
```

Metoda DecreasePower

Tato metoda je určena pro snížení výkonu na již běžících zdrojích v případě nadbytečné výroby. Její vstupními parametry jsou stejné jako v případě metody IncreasePower. Stejně tak struktura metody je podobná, hlavními rozdíly mezi těmito metodami jsou:

a Selekce kandidátů
Zde se snažíme snížit výrobu o nadbytečný výkon a zároveň maximálně snížit cenu produkce, vybíráme tedy bloky s nejdražším provozem.

b Výpočet nového výkonu zdroje
Stejně jako v předchozí metodě je volána metoda GetNewRunningPower,

které jsou předány totožné parametry. Rozdíl je v proměnné hasUndelivered, která má nyní hodnotu false. Takto se předá metodě informace, že se jedná o požadavek na snížení výkonu zdroje a je nutné testovat dodržení minimálního možného výkonu bloku.

c Dodržování omezení na chod
Protože snižování výkonu zdrojů může vést k porušení restrikce na výkon, konkrétně nastavení na úroveň MIN_RUN, je nutné dodržování tohoto omezení testovat. K tomu je určena metoda TestMinRunRestriction. Této metodě jsou předány jako vstupní parametry seznam právě běžících zdrojů, slovník jejich výkonů, blok, jehož výkon má být změněn, jeho navrhovaný výkon, současné hodnoty uzlu a aktuální čas simulace.

Zde je využit seznam všech simulovaných elektráren. V první části je nalezen seznam všech zdrojů, které náleží do elektrárny testovaného bloku. Pokud je součet výkonů všech těchto bloků, od kterých je odečteno navrhované snížení výkonu testovaného bloku, větší nebo rovno minimální požadované hodnotě dané vstupní časovou řadu MinReqPwr, je navrhovaná hodnota výkonu validní. V opačném případě není přípustná a je nutný její přepočet. Blok přepočtu má dvě varianty, pro tento případ (snížení výkonu bloku) se provede pouze navýšení hodnoty nového výkonu o chybějící výkon, pro splnění omezení. Je zde také ošetřen případ, kdy by byl chybějící výkon pro dodržení omezení větší než volný výkon zdroje (výkon může být zvýšen nanejvýš do maximálního možného výkonu bloku).

Metoda ManageUnitsToDeploy

Pokud je potřeba vybrat nové zdroje pro nasazení do provozu nebo vybrat bloky s plánovanou odstávkou, které je nutné nechat v provozu, je volána metoda ManageUnitsToDeploy. Oproti předchozím dvěma metodám patří mezi vstupní parametry navíc proměnná typu bool, která definuje za jakým účelem budou bloky nasazovány, a seznam zdrojů, které mohou být nasazeny. Záměrně nebyl výše uvedený seznam nazván jako nasaditelné zdroje, protože metoda může být volána ve dvou případech. Bloky jsou regulovány za účelem pokrytí:

1. zatížení v uzlu - seznam obsahuje nasaditelné zdroje
2. požadovaných rezerv - seznam může obsahovat buď nasaditelné zdroje nebo zdroje, které byly v předchozí hodině v provozu a jsou nyní určeny k odstavení

Ohodnocení všech zdrojů probíhá již popsaným způsobem voláním metody GetSourcesCosts, které je ale tentokrát předán seznam zdrojů, určených k nasazení (chceme nasadit co možná nejlevnější zdroje). Stejně tak probíhá i selekce vhodných kandidátů. Po zvolení vhodného zdroje je zde nutné otestovat, zda není omezení na chod elektrárny, do níž tento blok náleží, nastaveno na MAX_RUN. Pokud ano, musí být výkon zdroje striktně nastaven na jeho maximální hodnotu. V opačném případě probíhá výpočet výkonu standardně voláním metody GetNewRunningPower.

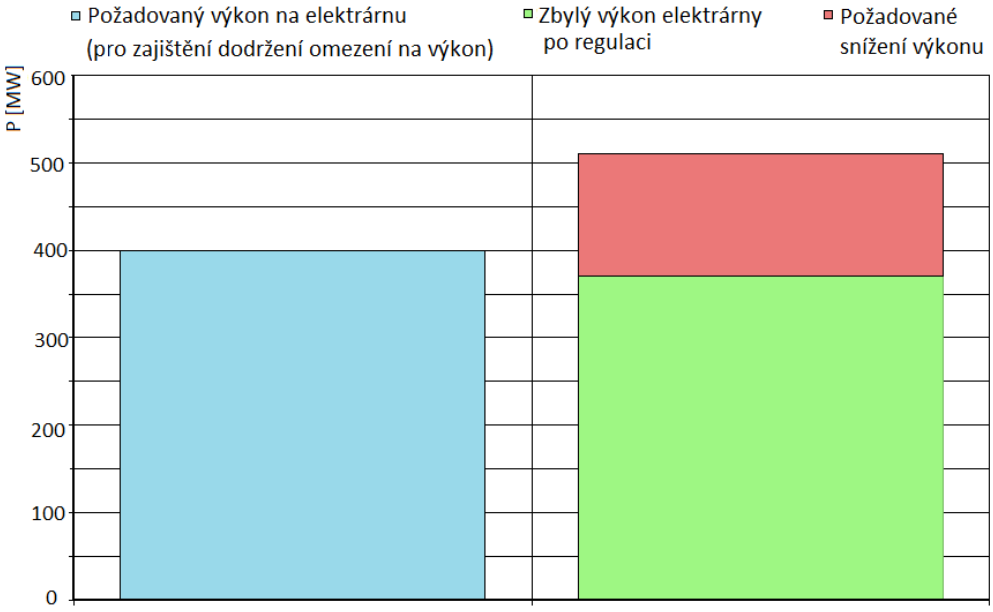
V poslední fázi dochází k otestování, zda zpracovávaný blok již nebyl upravován, přesněji, jestli tento blok nebyl této hodině v předchozí fázi regulace označen k odstavení. Pokud je takto označen, je požadavek na jeho odstavení zrušen a je aktualizován jeho výkon ve slovníku výkonů běžících zdrojů. Pokud se jedná o blok ze seznamu nasaditelných zdrojů (blok v předchozí hodině nebyl v chodu), je z tohoto seznamu odebrán a uveden do provozu. Také je přidán jako nový záznam do slovníku výkonu běžících zdrojů s požadovanou hodnotu výkonu. Nakonec je tento zdroj odebrán ze seznamu kandidátů, přidán do seznamu již zpracovaných zdrojů a také jsou přepočítány hodnoty v uzlu.

Metoda ManageUnitsToAdjust

Metoda ManageUnitsToAdjust byla navržena pro výběr bloků k odstavení v případě nadbytečné výroby. Tato metoda přijímá všechny parametry jako metoda DecreasePower a navíc ještě seznam odstavitelných zdrojů. Ohodnocení zdrojů a výběr kandidátů na odstavení probíhá stejným způsobem, jako u metody DecreasePower. Metodě GetSourcesCosts je předán seznam odstavitelných zdrojů a selekce probíhá od nejdražších z nich.

Stejně jako v metodě DecreasePower zde může dojít k porušení restrikce na výkon u elektráren, do nichž testované bloky náleží, a jsou nastaveny na úroveň MIN_RUN. Taktéž je volána metoda TestMinRunRestriction, které jsou předány tytéž parametry. První část testování je opět stejná. Provede se výpočet součtu výkonů bloků dané elektrárny a testuje se dodržení omezení. Pokud by bylo omezení po odstavení testovaného bloku porušeno, jak je znázorněno na obrázku č. 1.3, je nutné provést korekci. Ta spočívá v navýšení výkonu zbylých běžících bloků téže elektrárny. To je docíleno voláním metody IncreasePower, které je předán seznam, obsahující tyto bloky a uměle vytvořené hodnoty uzlu. Avšak toto lze provést pouze v případě, že volný výkon těchto zdrojů stačí k pokrytí chybějícího výkonu. V opačném případě je blok označen jako neodstavitelný.

Zdroj může být odstaven pouze pokud nebyl metodou TestMinRunRestriction označen jako neodstavitelný a v případě, že by jeho odstavení neznamenalo nedostatečnou výrobu (bilanci menší než nula). Pokud jsou tyto podmínky splněny, zdroj je označen k odstavení a jsou aktualizovány hodnoty uzlu.



Obrázek 3.8: Ilustrace zpracovávaných výkonů metodou TestMinRunRestriction

3.3 Výpočet hodnot v uzlu

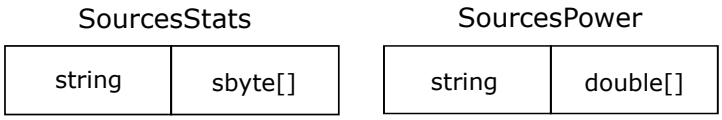
Předposlední důležitou částí simulátoru je třída StatsCalculator. Tato třída má dva důležité úkoly, spočítat v každé hodině simulace hodnoty stavu simulovaného uzlu a uložit tyto údaje. Jako vstupní parametry konstruktoru přijímá objekt s právě simulovaným uzlem (instance třídy AnalyzedScDto) a objekt s vygenerovanými výpadky pro daný případ (instance třídy FailSeriesDto).

V konstruktoru také dochází k přípravě struktur pro uložení průběhu simulace. První z nich je alokování dvourozměrného pole pro ukládání stavu uzlu s rozměry 10 × délka simulace. Jeho popis je dán tabulkou 3.2. Pole obsahuje stav uzlu před i po zásahu regulačního členu. Dále jsou vytvářeny slovníky pro ukládání stavu bloků (sourcesStats) a jejich výkonů (sourcesPower). Tyto slovníky mají podobnou formu, skládají se z klíče (jména bloku v typu řetězec) a jednorozměrného pole (pro stavy typu sbyte, pro výkony typu double) o velikosti délky simulace.

Oba slovníky jsou po vytvoření naplněny jmény všech bloků v uzlu jako klíči a k nim jsou vytvořeny prázdná výše zmíněná pole.

0	Minimální dodatelný výkon aktuálně běžícími zdroji (před regulací)
1	Aktuálně dodávaný výkon (před regulací)
2	Maximální dodatelný výkon aktuálně běžícími zdroji (před regulací)
3	Bilance v uzlu (před regulací)
4	Aktuálně dodávaný výkon (po regulaci)
5	Bilance v uzlu (po regulaci)
6	Maximální dodatelný výkon nasaditelnými zdroji (po regulaci)
7	Maximální odstavitelný výkon (po regulaci)
8	Nedodržené rezervy (po regulaci)
9	Nedodaná energie (po regulaci)

Tabulka 3.2: Popis pole s výstupy simulace



Obrázek 3.9: Pár klíč-hodnota slovníků sourcesStats a sourcesPower

Pro záznam stavu uzlu před regulačním zásahem byla navržena metoda SaveActualNodeStats. Jako vstupní parametry přijímá seznam všech běžících zdrojů, slovník jejich výkonů a aktuální čas simulace. Tato metoda ukládá hodnoty do prvních 4 míst pole 3.2. Pro získání aktuálně dodávaného výkonu, zatížení v uzlu a výpočtu bilance je volána metoda GetActualNodeBalance (přijímá stejné parametry jako volající metoda). Aktuálně dodávaný výkon je dán jako součet výkonů (získaných ze slovníku výkonů) všech běžících zdrojů. Aktuální zatížení je načteno z objektu s vlastnostmi simulovaného uzlu a bilance je vypočítána jako jednoduchý rozdíl těchto hodnot (kladná hodnota bilance znamená nadbytečnou výrobu, záporná nedostatečnou výrobu).

$$\text{bilance} = \text{dodávaný výkon} - \text{zátěž} \tag{3.6}$$

Pro získání minimálního a maximálního dodatelného výkonu slouží metoda CalculatePower, která v cyklu projde všechny běžící zdroje a sčítá jejich minimální (MinPwr) a maximální (MaxPwr) dodávatelný výkon. Výjimka vzniká u obnovitelných zdrojů. Pokud mají tyto bloky nastaveno omezení na úroveň "RES_FIX_RUN", je pro minimální i maximální dodatelný výkon použita požadovaná hodnota z časové řady ResReqPwr. Pro ostatní úrovně tohoto omezení je rozdíl v tom, že minimální dodávatelný výkon je 0.

Po provedení všech regulačních zásahů je potřeba tyto změny ve skladbě zdrojů a jejich nastaveném výkonu uložit. Také je nutné přepočítat všechny hodnoty v uzlu. O tuto činnost se starají následující metody:

- SaveSourcesStates
  Tato metoda přijímá jako vstupní parametry seznam všech běžících zdrojů, seznam zdrojů ve výpadku a aktuální čas simulace Obsahuje pouze jeden cyklus, ve kterém projde všechny klíče slovníku sourcesStats. Pokud se blok s testovaným klíčem nachází v seznamu běžících zdrojů, je v tomto slovníku k tomuto klíči a příslušné hodině uložena hodnota 1. Pokud se nenachází v běžících zdrojích, testuje se dále, zda není obsažen v seznamu zdrojů ve výpadku. Pokud je zdroj ve výpadku, je uložena hodnota -1, jinak se zdroj považuje za odstavený a je uložena hodnota 0.

$$\text{sourcesStats[klíč][hodina]} = \begin{cases} +1 & \text{v provozu} \\ -1 & \text{v výpadku} \\ 0 & \text{jinak (odstavený)} \end{cases}$$

- SaveSourcesPower
  Vstupními parametry metody jsou seznam běžících zdrojů, slovník jejich výkonů (sourcesPower) a aktuální čas. Stejně jako předchozí metoda se skládá z cyklu, který projde všechny klíče slovníku sourcesPower. Pokud se blok se jménem rovným testovanému klíči nachází v seznamu běžících zdrojů, je k tomuto klíči a příslušné hodině uložena hodnota tohoto bloku ze slovníku výkonů. V opačném případě je uložena nula.

$$\text{sourcesPower[klíč][hodina]} = \begin{cases} runningSourcesPower[unitName] & \text{v chodu} \\ 0 & \text{jinak} \end{cases}$$

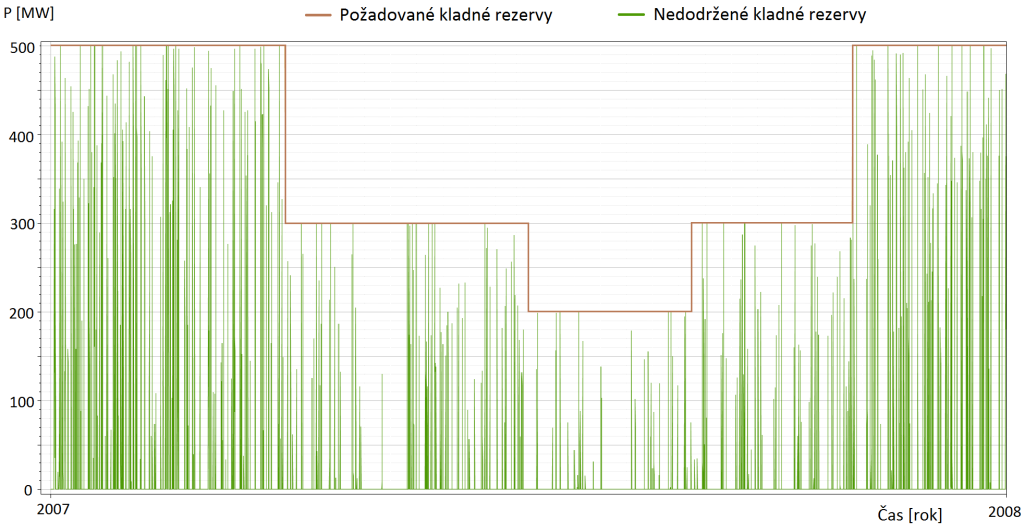
- SaveCorrectedNodeStats
  Pro uložení stavu simulovaného uzlu po provedení zásahů regulačním členem byla navržena metoda SaveCorrectedNodeStats. Vstupní parametry jsou seznamy nasaditelných, odstavitelných a běžících zdrojů, slovník výkonů běžících zdrojů a aktuální čas simulace. Zde jsou vypočítávány a ukládány hodnoty do zbývajících řádků pole 3.2 v dané hodině.

  Pro výpočet dodaného výkonu běžícími zdroji je opět volána metoda CalculatePower, tentokrát ale s regulačním členem upraveným seznamem běžících zdrojů. Bilance po zásahu je vypočítána stejně jako před zásahem, jen s novou hodnotou dodaného výkonu. Pro výpočet odstavitelného výkonu a maximálního nasaditelného výkonu je také volána metoda CalculatePower. V prvním případě je předán parametrem seznam odstavitelných zdrojů a v druhém

seznam nasaditelných zdrojů s požadavkem na zohlednění maximálního výkonu místo. Poslední dvě pozice v poli patří nedodrženým kladným točivým rezervám a nepokrytému zatížení. Velikost nedodržených rezerv je vypočítána jako součet volného výkonu na všech běžících zdrojích mínus velikost požadovaných rezerv. Pokud je tato hodnota kladná nebo rovna nule, je uložena nula, pokud je záporná, ukládá sa absolutní hodnota nedodržených rezerv.

$$rl = \begin{cases} \left| \sum_{i=1}^{N} P_{ra_i} - mr \right| & pro \sum_{i=1}^{N} P_{ra_i} - mr < 0 \\ 0 & jinak \end{cases} \tag{3.7}$$

r_u	Velikost nedodržených rezerv v danou hodinu
r_m	Hodnota požadovaných rezerv v danou hodinu
P_{ra_i}	Volný výkon běžícího zdroje s povolenými kladnými rezervami
N	Počet běžících bloků s povolenými rezervami



Obrázek 3.10: Graf požadovaných a nedodržených rezerv v uzlu

Velikost nepokrytého zatížení je rovna absolutní hodnotě záporných hodnot bilance. Platí:

$$\text{nepokryté zatížení} = \begin{cases} \text{bilance} & \text{pokud bilance} < 0 \\ 0 & \text{jinak} \end{cases}$$

Pro navrácení všech uložených dat po skončení simulace slouží metoda GetSimulatedCaseData. Jejím jediným úkolem je vytvořit instanci třídy SimulatedNodeDto

a naplnit tento objekt přes konstruktor uloženými hodnotami. Třída SimulatedNodeDto se skládá z:

- id scénáře
- id uzlu
- polem s hodnotami uzlu v průběhu simulace
- slovníku stavů chodu zdrojů
- výkonů zdrojů

Tato instance je vrácena jádrem simulátoru jako celkový výstup simulace se všemi uloženými hodnotami stavu simulovaného uzlu a zdrojů.

3.4 Validace výstupních dat

Důležitou součástí simulátoru je kontrola dodržování požadovaných restrikcí na zdroje. Zde se určuje, zda je dané řešení simulovaného scénáře přípustné a jak moc je kvalitní. V případě omezení na minimální dobu provozu a odstavení bloku se jedná o tzv. tvrdé omezení, to znamená, že pokud je porušeno, dané řešené simulovaného scénáře je nepřípustné. Simulátor je obecně navržen tak, aby toto omezení striktně dodržoval (kromě zmíněné výjimky z důvodu výpadku, kterou zohledňuje i validátor). Proto pokud vznikne případ, kdy restrikce není dodržena, jedná se o chybu návrhu simulátoru. Naopak omezení na chod a výkon zdroje jsou tzv. měkká omezení. To znamená, že mohou vzniknout případy, kdy je toto omezení porušeno. Čím více těchto porušení vznikne, tím méně kvalitní je dané řešení (nedodržení těchto restrikcí je finančně penalizováno).

3.4.1 Omezení na minimální dobu výroby a odstavení

Testování dodržení tohoto omezení provádí metody třídy UpDownRestrictionValidator. Pro spuštění validace je volána metoda ValidateRestrictions, která je navržena tak, aby v paralelním cyklu otestovala výstupy simulací všech variant simulovaných výpadků. V každém průchodu cyklu je volána metoda TestCase, které je předán jeden konkrétní výstup simulace.

Metoda TestCase otestuje v cyklu všechny zdroje v dané variantě a pro každý z nich projde jeho stavy přes celou délku simulace a hledá nejkratší interval, kdy byl blok odstaven, a nejkratší interval, po který blok běžel. Po projití celé délky simulace otestuje, zda jsou nalezené nejkratší časové intervaly větší nebo rovny nařízeným omezením. Metoda je blíže popsána pomocí pseudokódu č. 6.

Algoritmus 6 Pseudokód metody pro validaci omezení na minimální dobu výroby a odstavení

```
1:  function TestCase(sourcesStates)
2:    Inicializuj proměnné (downTime, upTime, minDownTime, minUpTime)
3:    for přes seznam sourcesStates do
4:      for přes délku simulace do
5:        if Minulý stav je roven současnému then
6:          if Současný stav je roven 1 then
7:            Inkrementuj upTime o 1
8:          else
9:            Inkrementuj downTime o 1
10:         end if
11:       else
12:         if Předchozí stav je roven 1 then
13:           if downTime < minDownTime then
14:             minDownTime = downTime
15:           end if
16:         else
17:           if Aktuální stav je výpadek then
18:             upTime = int.MaxValue
19:           end if
20:           if upTime < minUpTime then
21:             minUpTime = upTime
22:           end if
23:         end if
24:       end if
25:       Do předchozího stavu přiřaď aktuální
26:     end for
27:     Ulož výsledek validace
28:   end for
29: end function
```

3.4.2 Omezení na chod a výkon

Pro otestování dodržení omezení na chod a výkon zdrojů slouží metody třídy UcEdRestrictionValidator. Stejně jako v případě třídy UpDownRestrictionValidator je pro spuštění validace volána metoda ValidateRestrictions, která je také navržena na paralelní zpracování všech variant výpadků a v každém průchodu cyklu je volána

metoda TestCase, která testuje konkrétní variantu. Metoda TestCase obsahuje cyklus, který prochází všechny bloky a u každého z nich testuje dodržení obou omezení.

Pro validaci omezení na chod elektrárny v tomto cyklu volána metoda VerifyUcType. Testování jednotlivých úrovní omezení je následující:

a) EC
Pro tuto úroveň se neprovádí žádné testování, chod elektrárny není nijak omezen.

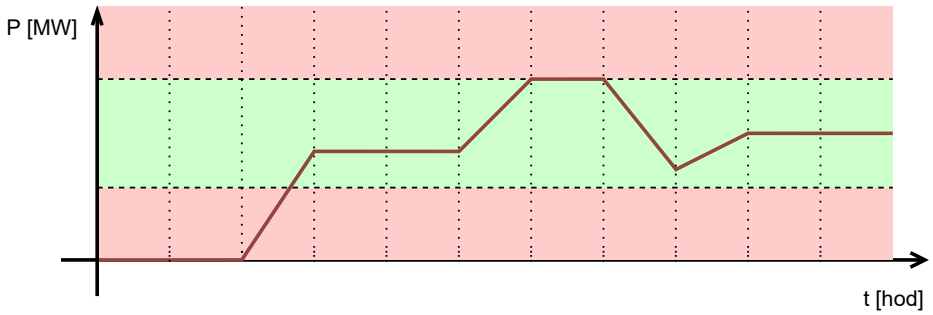
b) ALW_RUN
Při této úrovni se testuje, zda sekvence chodu bloků neobsahuje odstávku (stav roven 0).

c) PART_ALW_RUN
U tohoto nastavení se v každé hodině simulace testuje, zda v jednu chvíli byl v chodu alespoň minimální požadovaný počet bloků.

d) FIXED
Zde dochází k porovnání zaznamenané sekvence stavů s původní sekvencí požadovaných stavů.

Dodržení omezení na výkon elektrárny je testováno voláním metody VerifyEdType. Pro jednotlivé úrovně omezení se testuje:

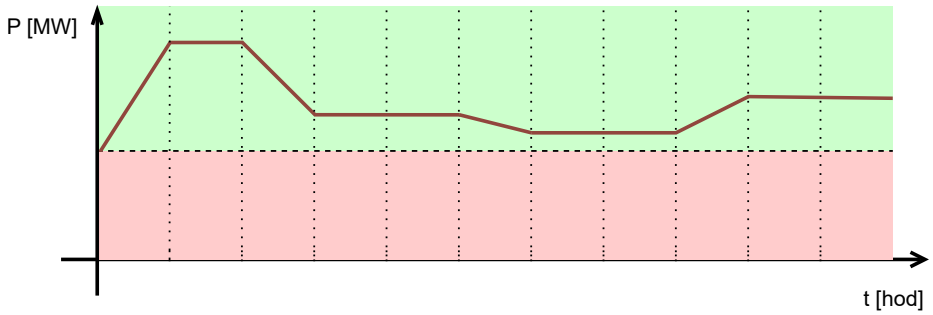
a) EC
Při nastavení an úroveň Economy se testuje, zda je celá sekvence výkonu zdroje v intervalu od nuly do maximálního možného výkonu bloku a v časech, kdy byl zdroj v provozu, zda byl výkon zdroje větší nebo roven minimálnímu možnému výkonu bloku.



Obrázek 3.11: Povolená pásma výkonu při omezení na úrovni EC

b) MAX_RUN
Při této úrovni se testuje, zda sekvence výkonu zdrojů obsahuje jinou hodnotu než maximální možný výkonu bloku. V případě, že je nalezena jiná hodnota, omezení bylo porušeno.

c) MIN_RUN
V této úrovni se prochází celá simulace a v každé hodině se spočítá součet výkonu všech bloků dané elektrárny. Pokud byl součet menší než požadovaný minimální výkon, omezení bylo porušeno.



Obrázek 3.12: Povolená pásma výkonu při omezení na úrovni MIN_RUN

d) FIXED a RES_FIX_RUN
Pro omezení s fixním předem definovaným výkonem (RES_FIX_RUN pro obnovitelné a FIXED pro ostatní zdroje) se testuje, zda jejich zaznamenaná sekvence výkonu přesně odpovídá zadaným časovým řadám (ResReqPwr pro obnovitelné a DelPwr pro ostatní typy elektráren).

e) SOFT_MAX_RUN a RES_MAX_RUN
U těchto dvou nastavení dochází pouze ke kontrole, zda sekvence výkonů neobsahuje hodnotu menší než nula nebo větší než maximální hodnotu v zadané časové řadě (ResReqPwr pro obnovitelné a DelPwr pro ostatní typy elektráren).

Kapitola 4

Simulace vlivu výpadků na zadaný model

V této kapitole jsou popsány modelové simulace dvou variant uvažovaných výpadků (při postupném výpadky zadaného zdroje a při jedné sadě vygenerovaných náhodných výpadků) a zhodnocena kvalita získaných výstupů z obou typů simulací.

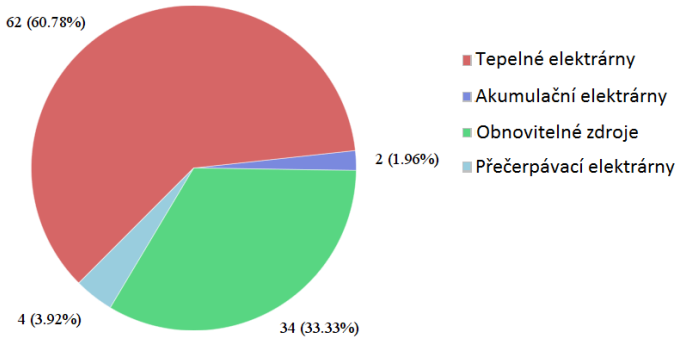
4.1 Model přenosové soustavy

V provedených simulacích byl použit model České elektrizační soustavy s fixními přeshraničními toky, které jsou reprezentovány příslušnými časovými řadami Load. Model se skládá z následujících prvků:

- 5 zón (CZ, AT, DE, PL, SK)
- 31 uzlů (27 českých a 4 zahraniční)
- 61 elektráren
- 110 bloků

Čtyři z těchto uvedených zón (AT, DE, PL, SK) aproximují přeshraniční výměnu pevně zadaným importem z historických dat. Každá z těchto zón je reprezentována jedním uzlem bez zdrojů, který v sobě obsahuje informaci o přeshraničním toku (časová řada Load). Zóna CZ obsahuje zjednodušený 400KV model České přenosové soustavy. Simulátor neuvažuje prvky typu zóna (ani linky), z každé z nich jsou načteny jejich uzly a pracuje se pouze s nimi. Bližší popis zón a linek je uveden v

bakalářské práci [7]. V modelu jsou zastoupeny všechny typy elektráren. Jejich zastoupení je uvedeno v grafu na obrázku č. 4.1.

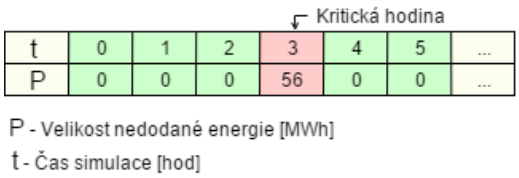


Obrázek 4.1: Zastoupení typů elektráren v zadaném modelu

V tomto modelu jsou dále nastaveny požadované kladné točivé rezervy v rámci celé zóny CZ. Jejich velikost je konstantních 1100 MW. Rezervy podporuje 54 bloků z celkových 110. Všechny tyto bloky patří pouze k tepelným elektrárnám, neboť produkce vodních zdrojů je v modelu již zafixována.

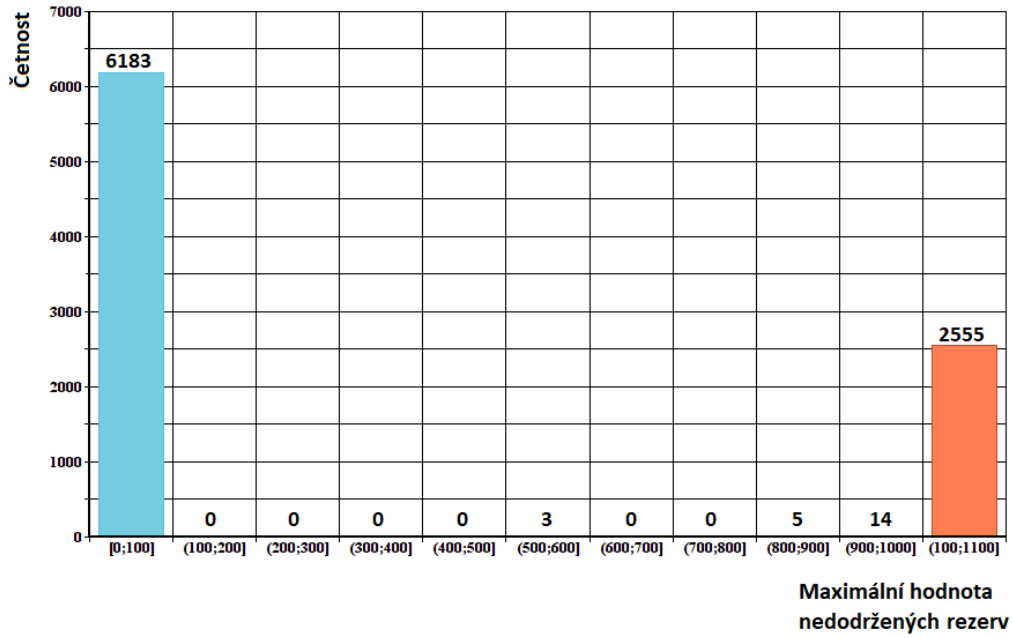
4.2 Analýza vlivu postupného výpadku

Pro analýzu stability elektrické přenosové soustavy je důležité testovat jak se projeví výpadek nejvýznamnějšího zdroje sítě v jednotlivých hodinách. Analýzou výsledků lze následně identifikovat kritické hodiny, kdy může dojít k vzniku nedodané energie nebo nedodržení požadovaných rezerv, a reagovat na tyto nedostatky posílením skladby zdrojů, případně nakoupením dodatečné regulační energie v zahraničí. Pro zadaný model elektrické soustavy byly vygenerovány výpadky zdroje s nejvyšším maximálním výkonem - v tomto případě se jedná o blok TEMELIN_b1 elektrárny TEMELIN s maximálním výstupním výkonem 1000 MW. Výpadky byly vygenerovány pro jeden nepřestupný rok začínající pondělím (tj. 8760 hodin), dostaneme tedy 8760 případů k simulaci. Číslo simulovaného případu odpovídá hodině, ve které nastal výpadek testovaného zdroje.



Obrázek 4.2: Ilustrace hledání kritické hodiny vzniku nedodané energie

V ani jednom případu nedošlo k porušení omezení na chod, výkon ani minimální dobu provozu a odstávky. V několika málo případech pouze došlo ke snížení výkonu obnovitelného zdroje s nastavením omezení na výkon na úroveň RES_MAX_RUN. Z každého odsimulovaného případu byla získána maximální hodnota nedodané energie a nedodržených rezerv. V žádném ze simulovaných případů nedošlo ke vzniku nedodané energie, zatížení bylo vždy plně pokryto. Požadované kladné rezervy ale nebyly 100% dodrženy ani v jednom případě. Četnosti maximálních hodnot nedodržených rezerv jsou uvedeny v histogramu na obrázku č. 4.3.

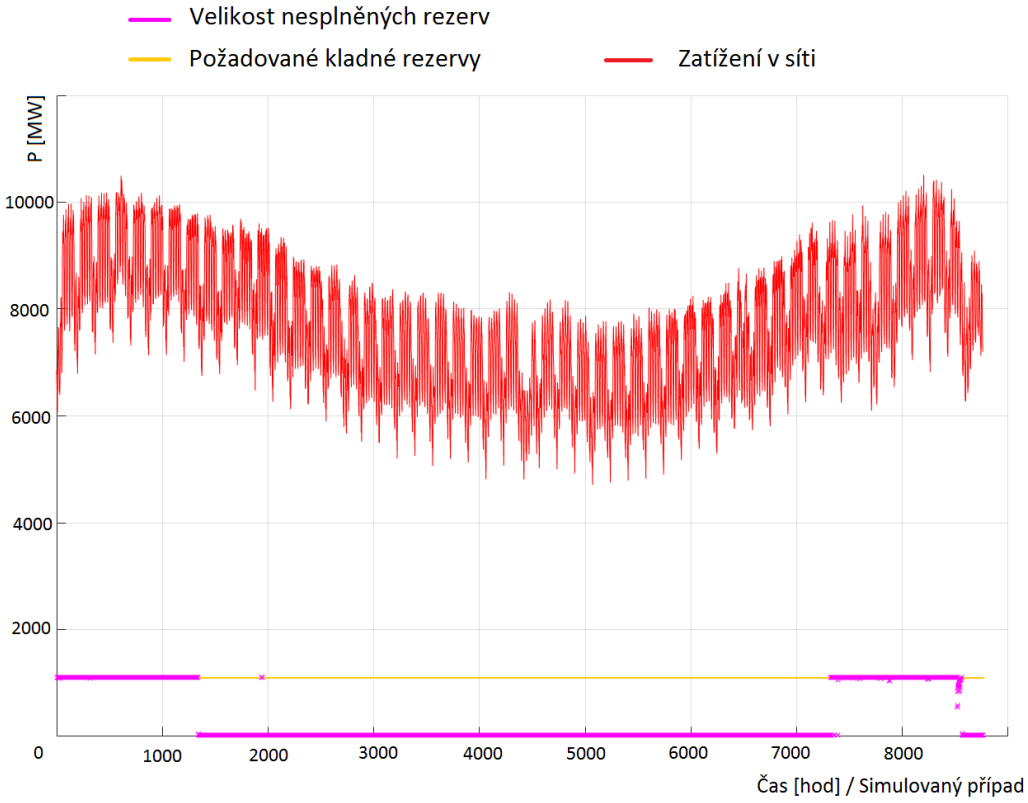


Obrázek 4.3: Histogram maximálních hodnot nedodržených rezerv simulovaných případů

Ačkoliv zde žádná nedodaná energie nevznikla, v 28% (tj. 2496) simulovaných případů nastalajedna nebo více hodin, kdy nebyla pokryta požadovaná hodnota kladných točivých rezerv vůbec tzn. velikost nedodržených rezerv dosáhla hodnoty 1100 MW. Tato skutečnost ještě nepředstavuje kritickou situaci, neboť v dané hodině stále existuje část zdrojů, které neposkytují rezervy a mají dostatek volné kapacity k pokrytí zatížení. V případě výpadku dalšího zdroje ovšem nebude zaručena okamžitá reakce, která byla zajištěna rezervovaným modulačním výkonem na nasazených zdrojích ve formě rezerv.

Graf na obrázku č. 4.4 ukazuje souvislost mezi celkovým zatížením v zóně a velikostí nedodržených rezerv. Je vidět, že k nejvýraznějším nedostatkům docházelo v intervalech simulovaných případů $(0; 1338)$ a $(7326; 8519)$. Tedy v případech,

kdy nastal plánovaný výpadek testovaného zdroje v období vysokého zatížení. Z těchto získaných výstupů simulací vyplývá, že je z hlediska pokrytí zatížení je model dostatečně robustní proti výpadku nejvýznamnějšího zdroje, avšak měl by být zadaný model posílen a více zdrojů, které podporují kladné rezervy.



Obrázek 4.4: Graf maximálních hodnot nedodržených rezerv v jednotlivých případech

Celkový čas simulace všech 8760 scénářů byl 8h. Průměrný čas jedné roční simulace, obsahující všechny zdroje v soustavě, je 3,2 vteřiny. Oproti modelu [4], který obdobný roční scénář simuluje 20 minut, představuje dosažený simulační čas zrychlení 375x. Tento čas může být dále snížen při použití procesoru s více jádry, neboť simulační knihovna byla navržena pro použití na vícejádrových procesorech.

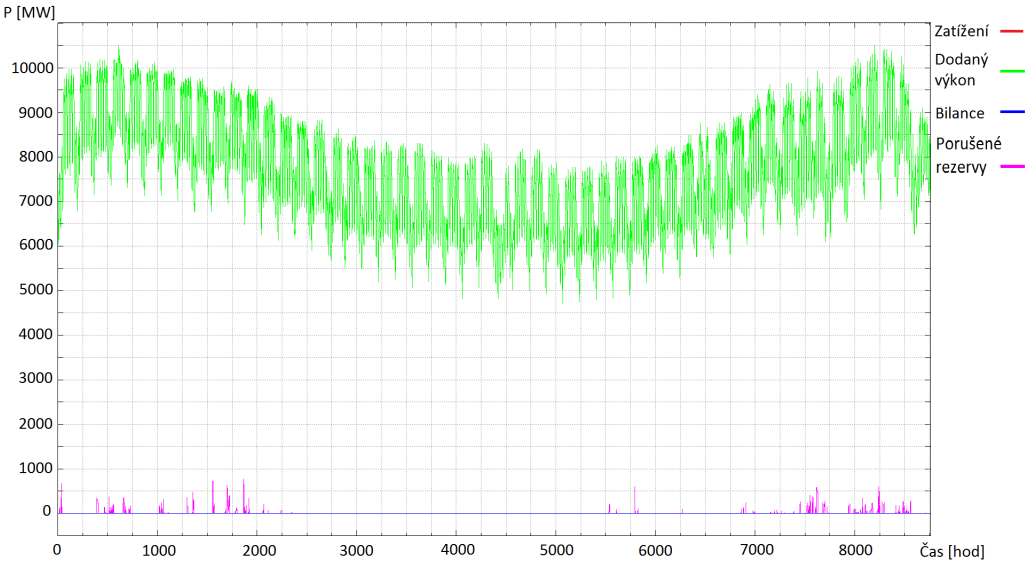
4.3 Analýza vlivu stochastických výpadků

V této části byl zadaný model podroben simulaci s vygenerovanými stochastickými výpadky. K vytvoření této sady výpadků byl použit počáteční stav 93350832 knihovního generátoru náhodných čísel. Celá mapa vygenerovaných výpadků je uvedena v příloze B. Délka simulace byla zvolena shodně s předchozím případem,

tedy 8760 hodin. Simulace byla provedena v deterministické variantě bez faktoru náhody (blíže popsáno v sekci 3.2.4), čímž byla zajištěna její opakovatelnost.

Validace omezení na minimální dobu provozu a odstávky bloků dopadla kladně, při simulaci nedošlo k porušení této restrikce. Také nebylo porušeno omezení na chod a výkon elektrárny. Nastal pouze jediný případ, kdy musel být snížen výkon obnovitelného zdroje s nastavením RES_MAX_RUN omezení na výkon a to pouze o 3,46 MW.

Při simulaci s touto sadou výpadků nedošlo v žádné hodině ke vzniku nedodané energie ani žádné nadbytečně vyrobené energie. Tato skutečnost je znázorněna grafem na obrázku č. 4.5 v tabulce č. 4.1. Z grafu je ale patrné, že opakovaně docházelo k nesplnění požadovaných kladných rezerv, především v období, kdy bylo vysoké zatížení v síti, a to až o velikost požadovaných rezerv, tedy 1100 MW. Z toho vyplývá stejný závěr, jako v předchozím případě, že by měl být model posílen o bloky, které podporují kladné rezervy.



Obrázek 4.5: Graf bilance, zátěže a vyráběného výkonu v průběhu simulace

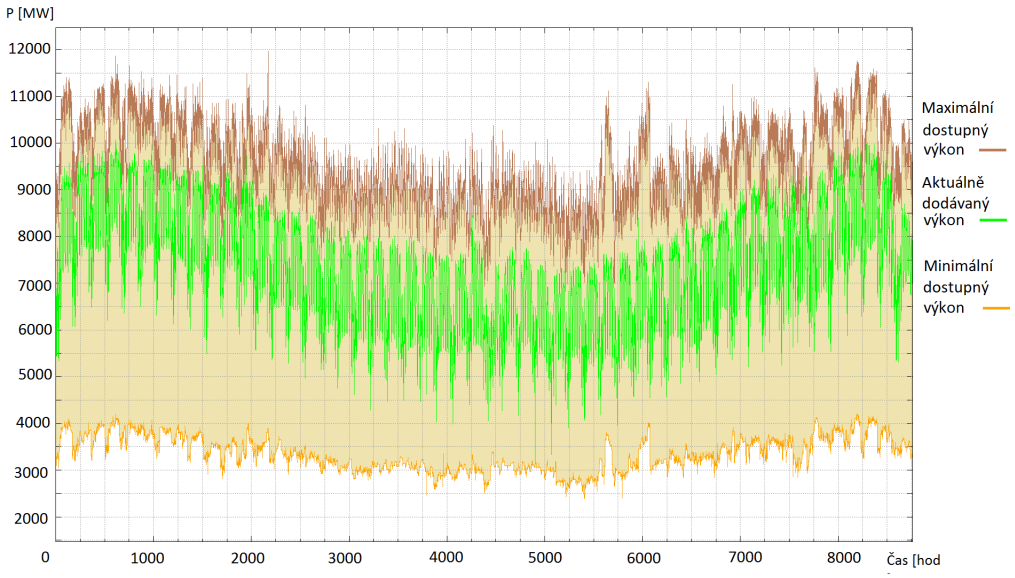
Sledovaný výstup	Modus	Průměr
Velikost nedodané energie [MW]	0,00	0,00
Velikost nedodržených rezerv [MW]	1100,00	143,08

Tabulka 4.1: Hodnoty sledovaných výstupů simulace

Graf na obrázku č. 4.6 znázorňuje stav výkonu zdrojů před regulačním zásahem. Z tohoto grafu je možné vyčíst dvě důležité informace.

- Velikost volného výkonu na zdrojích, které jsou aktuálně v chodu (rozdíl mezi maximálním dostupným výkonem a aktuálně dodávaným výkonem).
- Velikost snížitelného výkonu aktuálně běžících zdrojů (rozdíl mezi aktuálně dodávaným výkonem a minimálním dostupným výkonem)

Tyto dvě hodnoty udávají o jakou hodnotu lze zvýšit (snížit) aktuálně dodávaný výkon bez nutnosti nasadit (odstavit) jiné zdroje. Z grafu je patrné, že některých hodinách nezbývá mnoho volného výkonu. Pokrytí požadovaných kladných rezerv je v těchto hodinách závislé na nasaditelných zdrojích podporujících kladné rezervy).



Obrázek 4.6: Graf znázorňující hranice regulovatelnosti vyráběného výkonu aktuálně běžících zdrojů

Kapitola 5

Závěr

V rámci této diplomové práce byla navržena a implementována knihovní aplikace skládající se ze dvou částí. Prvním z nich je generátor výpadků, který umožňuje vytvořit sérii poruch zadaného zdroje, kdy je její vznik v každé variantě přesně o hodinu posunut, nebo zadané množství sad náhodně generovaných výpadků vybraných zdrojů. Generátor umožňuje zadání části výpadků uživatelem. Druhou částí je simulátor, který je schopný tyto vytvořené výpadky aplikovat na zadaný model elektrizační soustavy a odsimulovat jeho chování v zadaném časovém intervalu. Simulátor byl navržen tak, aby respektoval zadaná omezení na zdroje, které byly obtížné na implementaci a představují marginální část z celkového výpočetního času, avšak díky tomu jsou výstupy simulací kvalitnější.

Implementovaná knihovna byla otestována na zadaném modelu České elektrizační soustavy. Byly použity obě dvě varianty výpadků, jedna sada náhodně generovaných selhání a série postupných výpadků největšího zdroje modelu. I přes mírné zjednodušení oproti modelu [4] bylo dosaženo průměrného zrychlení roční simulace z 20 minut na 3,2 vteřiny. Při analýze výstupních dat simulací bylo zjištěno, že zadaný model obou případech obsahuje nedostatečné množství zdrojů poskytujících kladné točivé rezervy. Model soustavy by tedy měl být doplněn o další zdroje které podporují rezervy nebo by měly být povoleny rezervy u některých stávajících zdrojů.

Hlavním přínosem této práce je možnost provést rychlou analýzu stability a odolnosti modelu přenosové sítě vůči různým krizovým situacím, které jsou modelovány zvoleným typem výpadků. Velkou výhodou je možnost identifikace rizikových hodin, ve kterých dochází k poklesu kladných točivých rezerv, případně i k porušení výkonové bilance. Další výhodou je možnost využití výstupů této aplikace a použít je jako počáteční řešení, ze kterých mohou vycházet další optimalizační nástroje a snížit tak čas, potřebný k nalezení optimálního řešení skladby zdrojů.

Literatura

[1] Čeps, a.s., "Reálné zatížení naměřené v české elektrizační soustavě," 2013. Dostupné z `http://www.ceps.cz/CZE/Data/Vsechna-data/Stranky/Zatizeni.aspx`.

[2] Čeps, a.s., "Reálný průměr dodaného výkonu FVE elektráren do české elektrizační soustavy," 2013. Dostupné z `http://www.ceps.cz/CZE/Data/Vsechna-data/Stranky/Vyroba.aspx`.

[3] J. Tůma, S. Rusek, Z. Martínek, I. Chemišinec, and R. Goňo, *Spolehlivost v elektroenergetice*. CONTE spol. s r.o., ČVUT Praha", 2006.

[4] J. Zábojník, "Modelování výroby a toku elektrické energie v evropské přenosové soustavě," diplomová práce, České vysoké učení technické v Praze, 2012.

[5] E. Bauer, R. Adams, and D. Eustace, "Beyond redundancy: How geographic redundancy can improve service availability and reliability of computer-based systems," 2011.

[6] A. Viana, J. P. de Sousa, and M. A. Matos, "Fast solutions for uc problems by a new metaheuristic approach," *Electric Power Research*, no. 78, 2008.

[7] O. Svoboda, "Hybridní úložiště dat pro simulátor přenosové sítě," bakalářská práce, České vysoké učení technické v Praze, 2015.

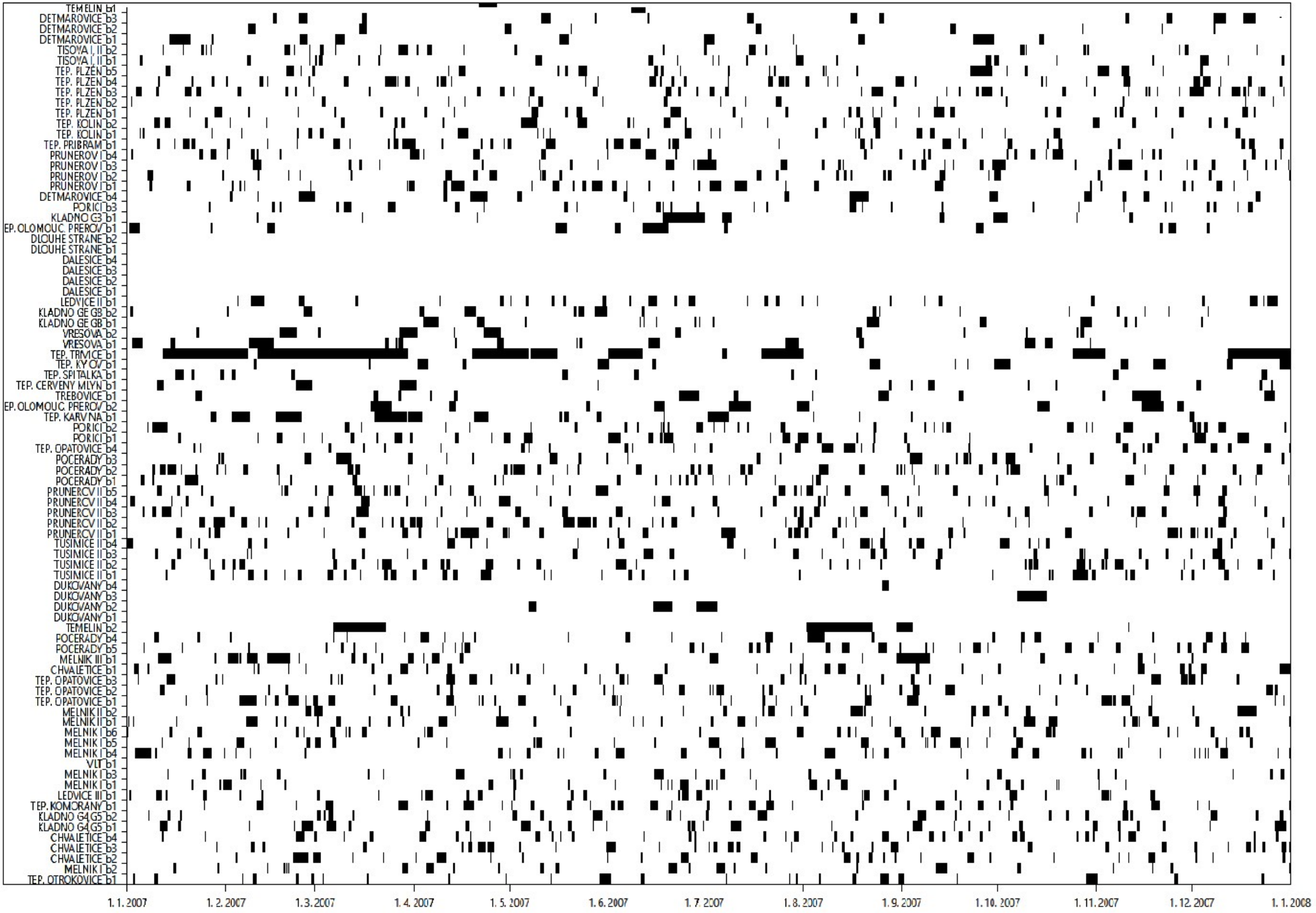
Příloha A

Obsah přiloženého CD

1. Text této diplomové práce
2. Projekt Microsoft Visual Studia 2015 s popsaným knihovním modulem
3. Model není přiložen. Je dostupný pouze na vyžádání u vedoucího práce.

Příloha B

Mapa náhodných výpadků


TEMELIN_b1
DETMAROVICE_b3
DETMAROVICE_b2
DETMAROVICE_b1
TISOVA_I, II_b2
TISOVA_I, II_b1
TEP. PLZEN_b5
TEP. PLZEN_b4
TEP. PLZEN_b3
TEP. PLZEN_b2
TEP. PLZEN_b1
TEP. KOLIN_b2
TEP. KOLIN_b1
TEP. PRIBRAM_b1
PRUNEROV I_b4
PRUNEROV I_b3
PRUNEROV I_b2
PRUNEROV I_b1
DETMAROVICE_b4
PORICI_b3
KLADNO G3_b1
EP. OLOMOUC. PREROV_b1
DLOUHE STRANE_b2
DLOUHE STRANE_b1
DALESICE_b4
DALESICE_b3
DALESICE_b2
DALESICE_b1
LEDVICE II_b1
KLADNO GE GB_b2
KLADNO GE GB_b1
VRESOVA_b2
VRESOVA_b1
TEP. TRMICE_b1
TEP. KYOV_b1
TEP. SPITALKA_b1
TEP. CERVENY MLYN_b1
TREBOVICE_b1
EP. OLOMOUC. PREROV_b2
TEP. KARVINA_b1
PORICI_b2
PORICI_b1
TEP. OPATOVICE_b4
POCERADY_b3
POCERADY_b2
POCERADY_b1
PRUNEROV II_b5
PRUNEROV II_b4
PRUNEROV II_b3
PRUNEROV II_b2
PRUNEROV II_b1
TUSIMICE II_b4
TUSIMICE II_b3
TUSIMICE II_b2
TUSIMICE II_b1
DUKOVANY_b4
DUKOVANY_b3
DUKOVANY_b2
DUKOVANY_b1
TEMELIN_b2
POCERADY_b4
POCERADY_b5
MELNIK III_b1
CHVALETICE_b1
TEP. OPATOVICE_b3
TEP. OPATOVICE_b2
TEP. OPATOVICE_b1
MELNIK II_b2
MELNIK II_b1
MELNIK I_b6
MELNIK I_b5
MELNIK I_b4
VLT_b1
MELNIK I_b3
MELNIK I_b1
LEDVICE III_b1
TEP. KOMORANY_b1
KLADNO G4G5_b2
KLADNO G4G5_b1
CHVALETICE_b4
CHVALETICE_b3
CHVALETICE_b2
MELNIK I_b2
TEP. OTROKOVICE_b1
1. 1. 2007
1. 2. 2007
1. 3. 2007
1. 4. 2007
1. 5. 2007
1. 6. 2007
1. 7. 2007
1. 8. 2007
1. 9. 2007
1. 10. 2007
1. 11. 2007
1. 12. 2007
1. 1. 2008