Aalto-yliopisto
Perustieteiden
korkeakoulu

Applying modified policy iteration to multi-component system maintenance scheduling

Petri Koivisto

11.5.2022

Ohjaaja: *DI Jussi Leppinen*
Valvoja: *Prof. Antti Punkka*

Systeemianalyysin
laboratorio

Taustaa

- Tekniset järjestelmät kuluvat käytössä ja niitä täytyy huoltaa säännöllisesti
  - Järjestelmän komponenttien uusiminen
- Huollon suunnittelulla voidaan vaikuttaa merkittävästi pitkän aikavälin kustannuksiin
  - Samalla pyritään säilyttämään tietty vaadittu luotettavuustaso
- Huoltojen aikataulutukseen voidaan käyttää huollon aikataulutuksen optimointimallia (Leppinen, 2020)

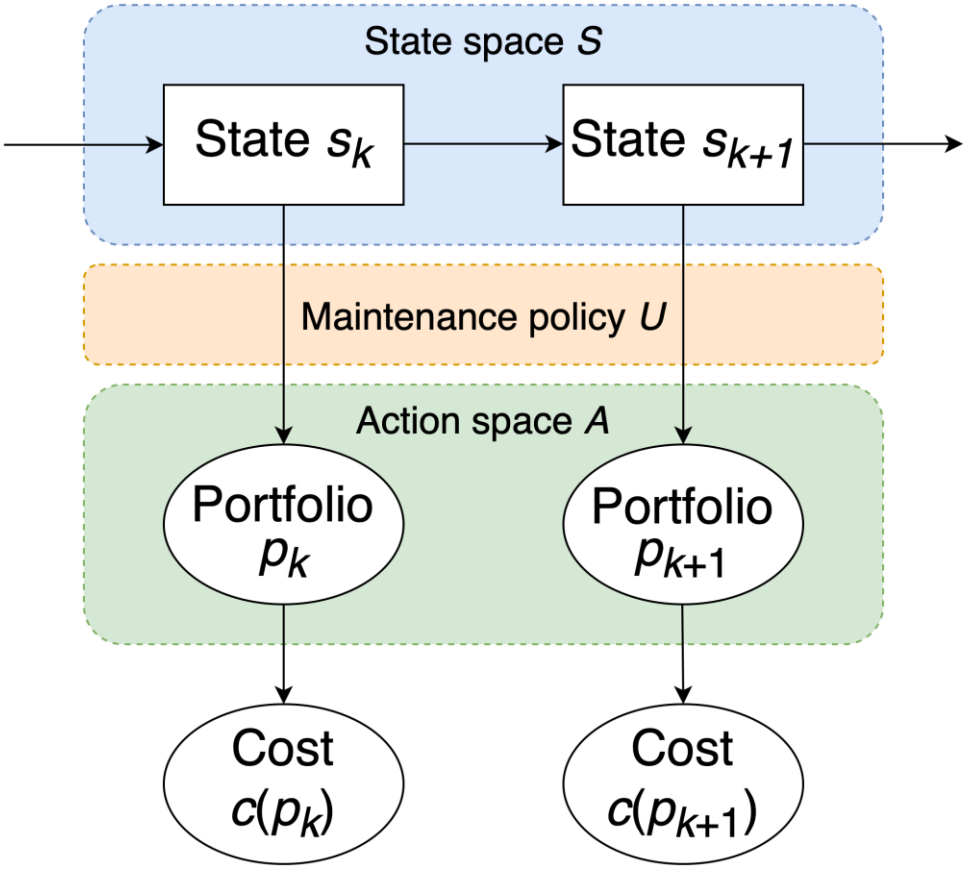
Huollon aikataulutuksen optimointimalli

- Lähtötiedot:
  - n komponenttia
  - Jokaisella komponentilla oma vikaantumisajankohtaa kuvaava todennäköisyysjakauma (esim. Weibull-jakauma)
    - Vikaantumistaajuus oletetaan kasvavaksi
  - Sarjajärjestelmä: minkä tahansa komponentin vikaantuminen johtaa koko järjestelmän vikaantumiseen
  - Huoltoja tehdään vain diskreeteillä ajanhetkillä t_k huoltovälin ollessa Δt
  - Järjestelmän luotettavuuden täytyy olla huoltoväliin t_{k+1} asti vähintään $\rho \in (0, 1)$
  - Järjestelmän tilaa hetkellä t_k kuvaa tilavektori $s_k \in \mathrm{R}^{2n}$
  - Koska vikaantumistaajuus oletetaan kasvavaksi, niin mahdollisten uniikkien tilavektorien määrä on äärellinen
    - Tila-avaruuden koko $|S|$

Huollon aikataulutuksen optimointimalli

- Huoltopäätökset:
  - Jokaisella huoltohetkellä t_k tehdään päätös siitä, että mitä komponentteja huolletaan
    - Huoltoportfolio $p_k \subseteq \{1, \ldots, n\}$ sisältää huollettavat komponentit
  - Kaikki huoltoportfoliot tyhjää portfoliota lukuun ottamatta aiheuttavat kustannuksen $c(p_k)$
  - Ohjaus (engl. *policy*) U antaa jokaiselle tilalle portfolion
    - Funktio $U(s_k) = p_k$
  - Tavoitteena löytää ohjaus, joka minimoi pitkän aikavälin kumulatiivisen odotusarvoisen kustannuksen
    - Tulevaisuuden kustannukset diskontataan kertoimella $\lambda \in (0, 1)$
  - Siirtymätodennäköisyys tilasta toiseen riippuu vain systeemin nykytilasta
    - Kyseessä siis Markov-päätöksentekoprosessi (Puterman, 1994)

Markov-päätöksentekoprosessi

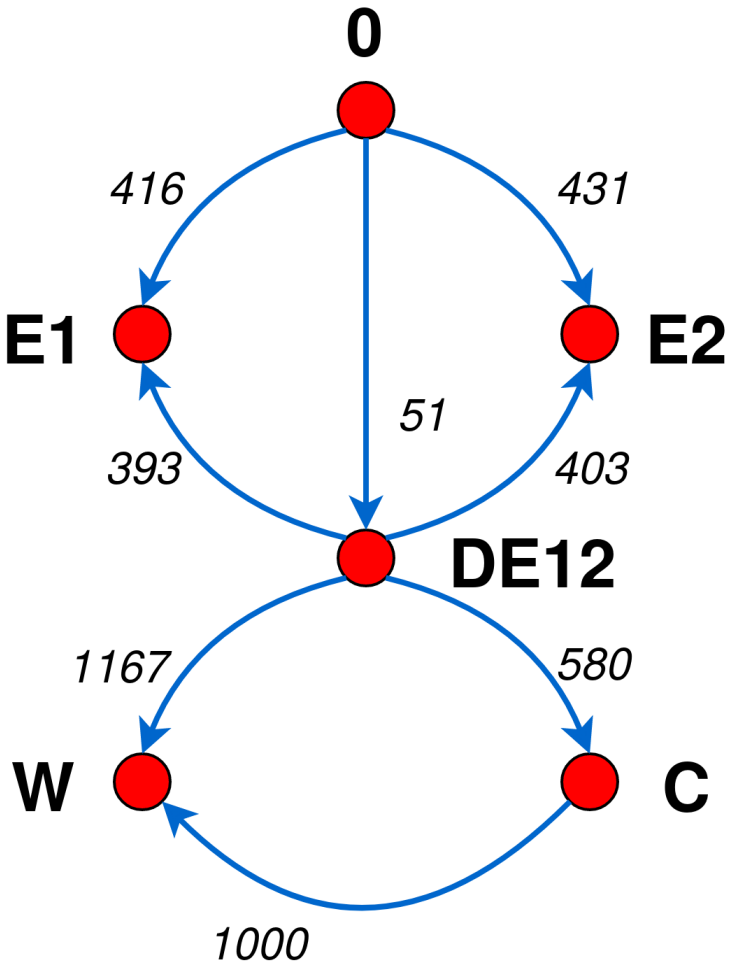


Markov-päätöksentekoprosessi

- Markov-päätöksentekoprosessin optimaalinen ohjaus voidaan rataista ohjauksen iterointialgoritmilla (engl. *policy iteration algorithm, PI*)
  - Toteutettu Leppisen diplomityössä 2020
- Haasteena on, että algoritmissa täytyy kääntää $|S|\times|S|$-kokoinen matriisi
- Kandidaatintyössä implementoitiin muokattu ohjauksen iterointialgoritmi (engl. *modified policy iteration algorithm, MPI*)
  - Vältetään yhtälöryhmän täsmällisen ratkaisun etsiminen lähestymällä ratkaisua approksimatiivisesti

Tulokset (1/4)

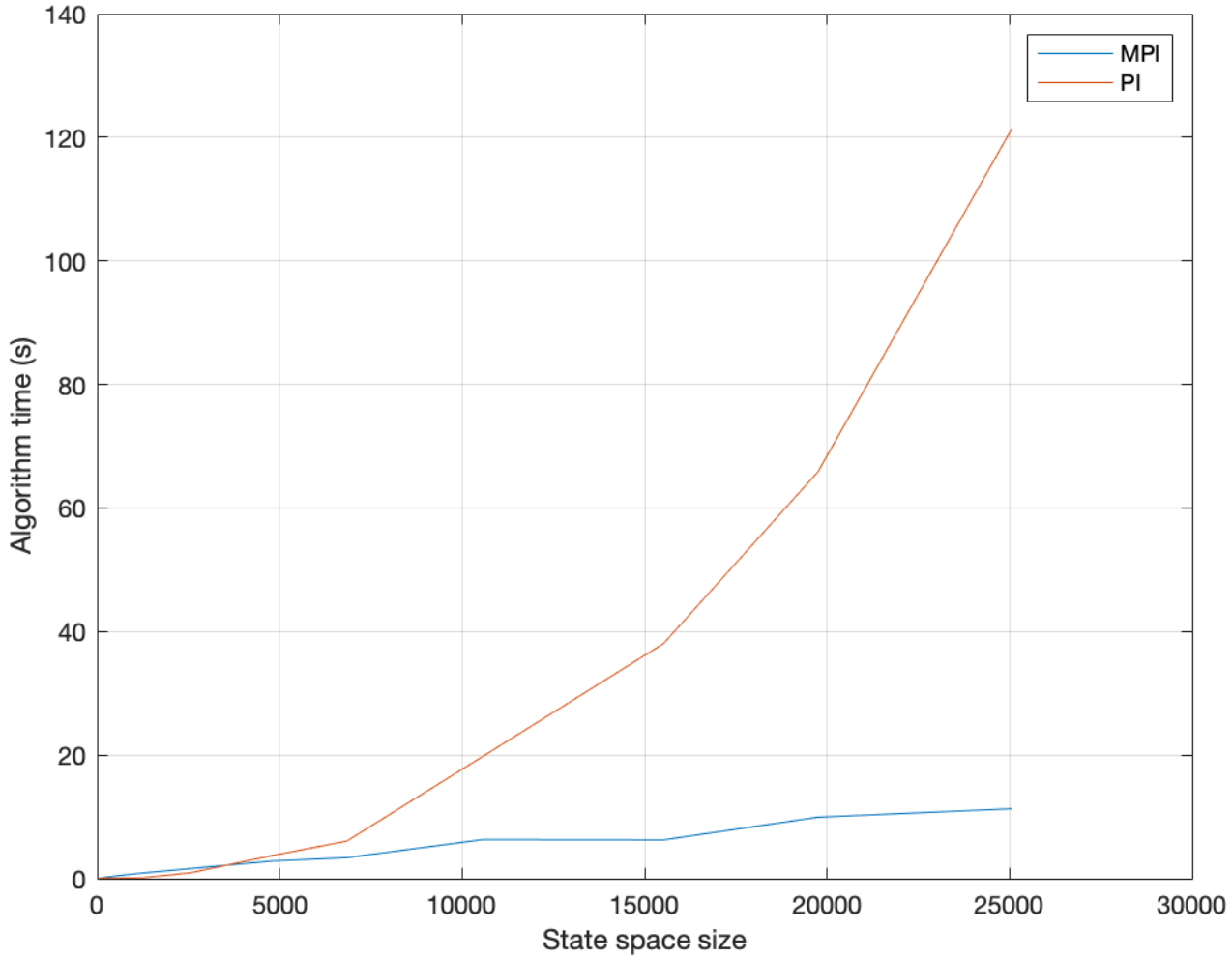
- Työssä verrattiin PI ja MPI algoritmeja laskemalla optimaalisia huoltoaikatauluja testijärjestelmälle
- Algoritmien implementaatiot toteutettiin MATLABilla ja laskennat työasemalla, jossa Intel Core i5-11600, 2.80 GHz, 32 GB RAM
- Laskennat suoritettiin varioimalla eri parametrien arvoja, ja algoritmien laskenta-aikoja ja tuloksia verrattiin



Tulokset (2/4)

- Ensiksi muutettiin luotettavuuskynnyksen ρ arvoa
  - Mitä pienempi luotettavuusraja, sitä suurempi on mahdollisten tilojen määrä
  - Ei muuta mallin muita parametreja, antaa kuvan algoritmien käyttäytymisestä tila-avaruuden kasvaessa

Luotettavuusraja ρ	Tila-avaruuden koko $\|S\|$	Iteraatioiden määrä MPI	Ajoaika MPI (s)	Iteraatioiden määrä PI	Ajoaika PI (s)
0.999	40	24	0.05	1	0.06
0.99	550	18	0.45	4	0.09
0.98	1225	16	0.89	7	0.12
0.96	2560	14	1.63	9	0.94
0.93	4780	13	2.83	9	3.70
0.90	6840	11	3.38	8	6.06
0.85	10570	13	6.31	9	19.79
0.80	15520	9	6.26	10	38.06
0.75	19750	11	9.94	9	65.86
0.70	25060	10	11.30	8	121.38

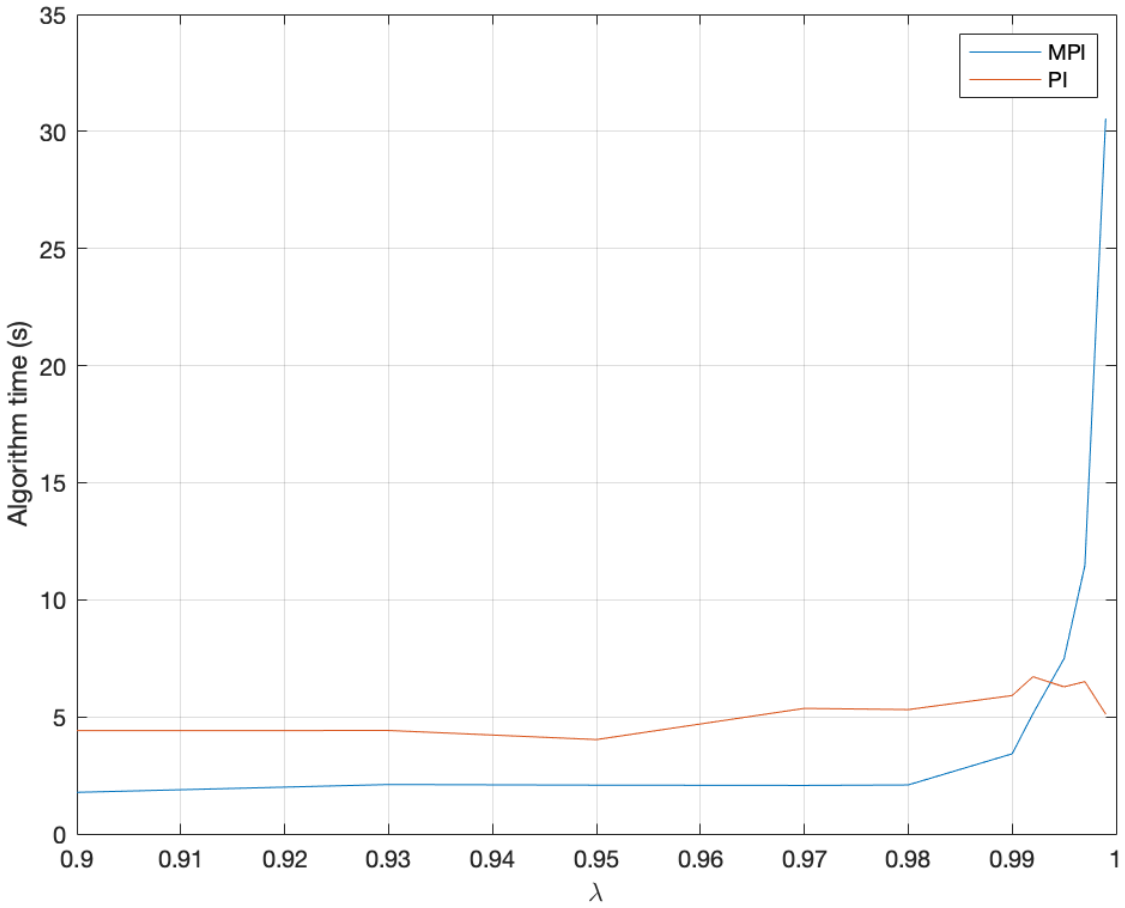


Algoritmien suoritusajat tila-avaruuden koon funktiona. Testissä $\lambda = 0.99$.

Tulokset (3/4)

- Algoritmeja verrattiin myös eri diskonttauskertoimilla λ
  - Testeissä $\rho = 0.9$ ja $|S| = 6840$

Diskonttauskerroin λ	Iteraatioiden määrä MPI	Ajoaika MPI (s)	Iteraatioiden määrä PI	Ajoaika PI (s)
0.9	6	1.78	7	4.41
0.93	7	2.11	9	4.41
0.95	7	2.08	8	4.03
0.97	7	2.07	9	5.36
0.98	7	2.09	8	5.31
0.99	11	3.42	8	5.91
0.993	16	5.13	10	6.71
0.995	23	7.49	9	6.28
0.998	35	11.47	9	6.50
0.999	92	30.56	7	5.12



Algoritmien suoritusajat diskonttikertoimen funktiona. Kyseisessä testissä $\rho = 0.9$ ja $|S| = 6840$

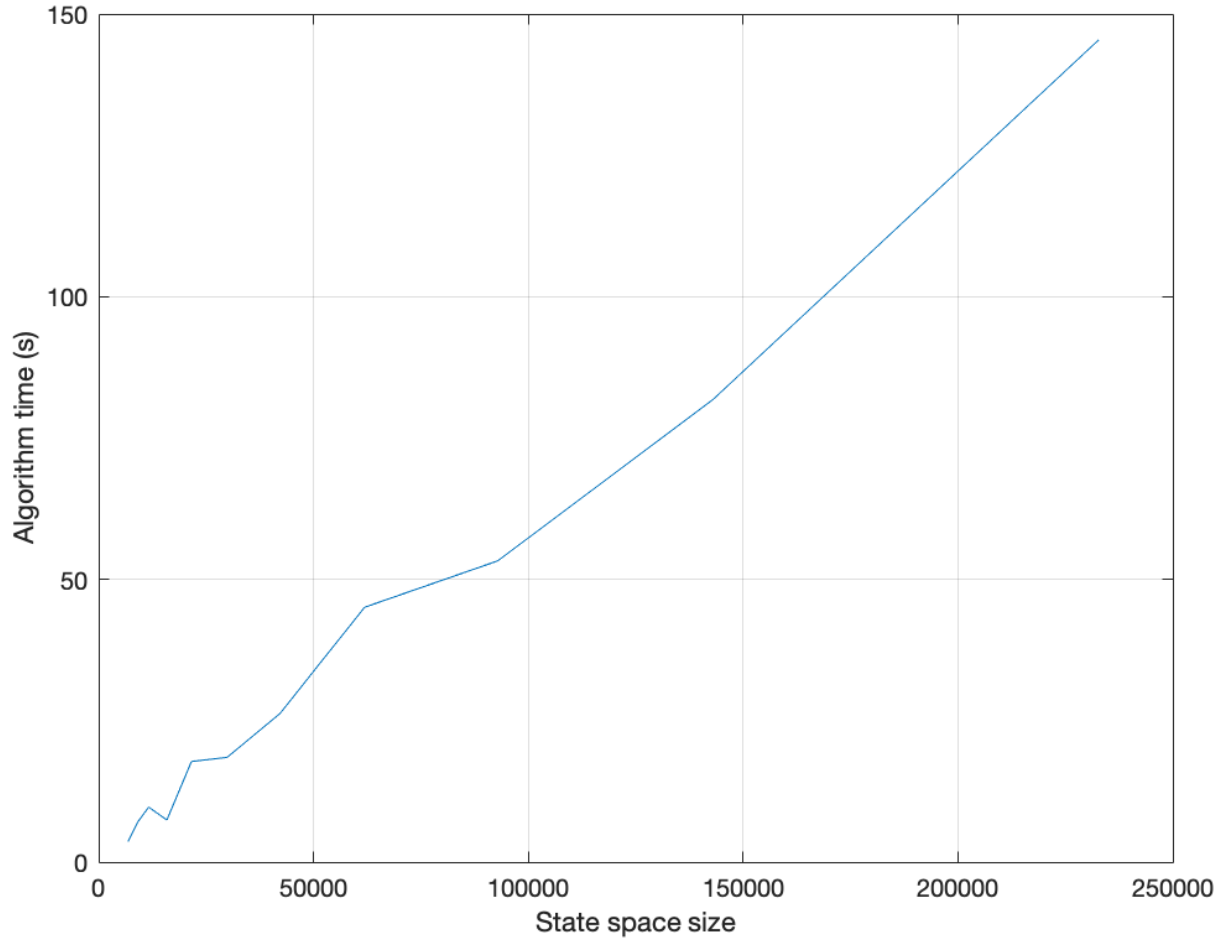
MPI-algoritmin muistinkäyttö

- MPI-algoritmin implementaatiossa voidaan käyttää hyödyksi Markov-päätöksentekoprosessin siirtymämatriisin harvuutta
- Koska $|S| >> n$ ja jokaisesta tilasta voidaan siirtyä vain $n + 1$ tilaan, niin siirtymämatriisi $P \in \mathrm{R}^{|S| \times |S|}$ on hyvin harva
  - Implementaatiossa voidaan säilöä vain nollasta poikkeavat todennäköisyydet ja niiden indeksit
  - MPI-algoritmissa täytyy laskea matriisi-vektorituloja, joiden myös laskenta nopeutuu, kun nollalla kertominen jää pois
  - PI-algoritmissa täytyy ratkaista lineaarinen yhtälöryhmä, joka on muotoa $(I - \lambda P)v(U) = c(U)$

Tulokset (4/4)

- Kun testataan algoritmien suorituskykyä tila-avaruuden kokoa kasvattamalla, huomataan nopeasti, että muisti loppuu kesken siirtymämatriisi *P*:n säilömiseen
- MPI-algoritmin suorituskykyä suurilla tila-avaruuksilla testattiin muuttamalla mallin huoltovälin Δt pituutta
- Esimerkiksi kun $\rho = 0.9$, huoltovälin puolittaminen kasvattaa tila-avaruuden 34-kertaiseksi ja koko *P*:n säilöminen vaatisi 400 GB muistia käytettäessä tuplatarkkuusliukulukuja

Huoltoväli Δt	Diskonttauskerroin λ	Tila-avaruuden koko $\|S\|$	Iteraatioiden määrä	Ajoaika (s)
1	0.99	6840	11	3.63
0.95	0.990	9090	16	7.07
0.90	0.991	11635	17	9.72
0.85	0.991	15875	10	7.44
0.80	0.992	21600	17	17.79
0.75	0.992	29885	13	18.52
0.70	0.993	42185	13	26.26
0.65	0.993	61890	15	45.07
0.60	0.994	92875	12	53.27
0.55	0.994	143040	12	81.86
0.50	0.995	232755	13	145.43



MPI-algoritmin suoritusaikoja tila-avaruuden koon funktiona huoltoväliä Δt muutettaessa.

Yhteenveto

- MPI-algoritmi vaikuttaa tulosten perusteella varteenotettavalta vaihtoehdolta optimaalisen huoltoaikataulun ratkaisemiseen
  - Monissa tapauksissa ratkaisee ongelman huomattavasti nopeammin ja vähemmän muistia käyttäen kuin tavallinen PI-algoritmi
  - Tuottaa kaikissa testeissä täsmälleen saman ohjauksen kuin PI-algoritmi
- Erityisesti komponenttien lisääminen järjestelmään kasvattaa tila-avaruuden kokoa ”todennäköisesti eksponentiaalisesti”
  - Jos diskonttauskerroin λ pysyy samana, niin MPI-algoritmin pitäisi sopia tilanteeseen paremmin

Lähteet

- J. Leppinen. A Dynamic Optimization Model for Maintenance Scheduling of a Multi-Component System. Diplomityö, Aalto-yliopisto, 2020
- M. L. Puterman. Markov Decicion Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, Inc., 1994.