# Lühike peenplaneerimise algoritmide ja süsteemide ülevaade

Riina Maigre

18. oktoober 2018. a.

## 1 Sissejuhatus

Peenplaneerimise ülesanne tootmises on etteantud ajahorisonti arvestades tootmisplaani genereerimine ehk tootmisprotsesside operatsioonide ajaline paigutamine ja ressurssidele vahel jaotamine. Ressursid võivad olla nii seadmed kui inimesed. Peenplaneerimise eesmärk on vähendada tootmiseks kuluvat aega ning maksumust. Samuti aitab peenplaneerimine pudelikaelaks olevaid seadmeid paremini hallata, tähtaegu täpsemalt määrata ning ette planeerida. Peenplaneerimise tulemused esitatakse tavaliselt Gantti graafikuna [1, 2].

Sellist peenplaneerimist nimetatakse ka töökoja-tüüpi planeerimiseks *(job shop scheduling)* ning see erineb tootmisliinide planeerimisest *(flow management)* selle poolest, et ressursse kasutavad tööd on tavaliselt tellimus- või partiipõhised ning seetõttu tihti muutuvad. Iga töö koosneb mitmest erinevast operatsioonist, mis vajab täitmiseks mingit ressurssi. Töödel on ka erinevad marsruudid. Peenplaneerimise teeb keeruliseks just erinevate toodete ning seetõttu ka tööde paljusus, kasutatavate marsruutide rohkus ning uute tööde (tellimuste) pidev lisandumine, mistõttu on töökoja-tüüpi peenplaneerimisele optimaalse lahenduse leidmine NP keerukusega probleem ehk lõplikus ajas lahendumatu.

Peenplaneerimise algoritme on võimalik jagada selle järgi, kui palju nad võimaldavad optimeerida. Kuna optimaalse lahenduse leidmine on NP keeruline juba siis kui seadmeid on üle kolme, siis ei sobi optimaalsed peenplaneerimise algoritmid reaalsete peenplaneerimis probleemide lahendamiseks ning mõistlik on üritada leida mitte optimaalseid vaid optimaalsuse lähedasi lahendusi. Paljude probleemide puhul kehtib aga seaduspära, et kui optimaalne probleem on NP täielik, siis on NP täielik ka optimaalsuse lähedane probleem [1]. Ouelhadj ja Petrovic [3] toovad välja, et paljud algoritmid reaalsetesse süsteemidesse ja olukordadesse ei sobi, sest ootamatud reaalajalised sündumused võivad muuta koostatud tootmisplaani kasutuks.

Järgnev on ülevaade mitmetest artiklites avaldatud algoritmidest, nendega saavutatud tulemustest ning nende puudustest. Lühidalt vaatame ka, mida on kirjanduses avaldatud reaalsete süsteemide kohta.

# 2 Algoritmid

Heuristilised algoritmid võimaldavad leida mõistliku ajaga piisavalt hea lahenduse, mis ei ole optimaalne. Heuristikad on tavaliselt suunatud mingi kindla mõõdiku parandamiseks. Heuristilised algoritmid on tööstuses kõige levinumad [1].

Peenplaneerimise ülesannete lahendamisel on üritatud kasutada mitmeid tehisintellekti algoritme. Põhjalikuma ülevaate annab näiteks Çalişe ja Bulkani ülevaateartikkel [4], kus autorid toovad välja, et üksi tehisintellekti algoritmidel põhinevad meetodid ei garanteeri optimaalset lahendust. Autorid toovad ka välja, et artiklites avaldatud algoritmid ei proovi tavaliselt lahendada reaalseid tööstuse ülesandeid vaid kasutavad konstrueeritud jõudlusteste (näiteks OR-Library [5]). Analüüsitud töödest ei selgu, millised kirjeldatud meetoditest on teistest paremad. Näiteks, puuduvad võrdlused teiste meetoditega sipelgameetodi *(ant colony optimization)* [6, 7] ja osakestepilve optimeerimise algoritmide kohta *(particle swarm optimization)* [8, 9].

## 2.1 Dispetšer-reeglid *(dispatching rules)*

Dispetšer-reeglid määravad ressurssi kasutava töö, siis kui see ressurss või teised ressursid vabanevad ja vastav töö tekib. Selline planeerimine sobib tehastesse, kus on suur hulk erinevaid tooteid või kus täpsem ja ettevaatavam peenplaneerimine on keeruline. Otsus, millist tööd ressurss järgmisena tegema hakkab tehakse selle põhjal, milline on töö täitmise kuupäev ning prioriteet ja kas see töö nõuab ümberseadistamist või mõne muu heuristika või reegli põhjal. Otsused tehakse lähtuvalt hetkeseisust ning ettepoole ei vaadata. Enamasti ei vaadata ka kogu tootmisprotsessi vaid ainult konkreetset ressurssi või tööd. Eelis on see, et dispetšer-reeglid ei vaja eriti arvutusressurssi. Seda tüüpi algoritmide puuduseks on, et kui ressursi koormatus läheneb 100%-le siis laovaru ja tooteühiku tootmisaeg hakkab lähenema lõpmatusele [1]. Jain ja Meeran [10] toovad välja, et kuna igas otsustuspunktis vaadatakse vaid ühte võimalikku operatsiooni võib lahendus olla optimaalsest väga kaugel (isegi 74% optimaalsest erinev). Tulemuste parandamiseks kombineeritakse erinevaid reegleid, mis vajab aga rohkem arvutusvõimsust. Nende puuduste tõttu sobivad dispetšer-reeglid esialgseks lahenduseks, mida kombineeritakse keerulisemate algoritmidega [10]. Süsteemidest kasutab näiteks Asprova[1] dispetšer-reegleid koos ettepoole ja tagasisuunas peenplaneerimise kombineerimisega [11].

## 2.2 Ettepoole ja tagasisuunas peenplaneerimine *(forward and backward scheduling)*

Ettepoole ja tagasisuunas peenplaneerimine on enamikes ERP süsteemides kõige rohkem kasutatavamad meetodid. Tagasisuunas planeerimisel [12] lähtutakse tellimuse tähtajast ning määratakse selle põhjal, millal toote tootmine peab algama. Ressursse kasutatakse, siis kui vaja, mis tähendab, et vahepeal on mõned

[1] `https://www.asprova.com/en/`

seadmed ootel. Töötab nii pika- kui lühiajalisel planeerimisel. Ettepoole planeerimisel (näiteks Sanko ja Kotkas [13]) lähtutakse kuupäevast, millal ressurss vabaneb ning määrtakse sellest tellimuse tähtaeg. Tööd planeeritakse seadmetele peale kohe kui need vabanevad. Suurim probleem on liigne laovaru ning et komponent-tootmise puhul saavad komponendid erineval ajal valmis. Ettepoole peenplaneerimine töötab lühiajalisel planeerimisel. Pikaajalisel planeerimisel võivad laovarud liiga suureks minna [14].

Ettepoole ja tagasisuunas peenplaneerimist on võimalik ka kombineerida. Seda teevad Asprova, MIE Trak[2], FrePPLe[3] ja paljud teised, kus esmajärjekorras üritatakse tagasisuunas peenplaneerimisega katta tellimused õigeaegselt. Kui tellimuse täitmine õigeaegselt ei ole võimalik, kasutatakse ettepoole planeerimist ning leitakse esimene võimalik tähtaeg, milleks toodet on võimalik toota [14].

Ettepoole ja tagasisuunas peenplaneerimine võimaldab järjestada nii operatsioone kui ka ressursse. Kui planeeritakse ressursse nende kasutamise järgi, siis nimetatakse sellist planeerijat pudelikaela planeerijaks, mis on vastavuses kitsenduste teooriaga *(theory of constraints)*[15]. Planeerimisel saab ajaühikutena kasutada minuteid, tunde, päevi või nädalaid. Mida suuremates ajaühikutes planeeritakse, seda vähem arvutusressurssi kulub. Samuti on kindlaks määratud ajahorisont, mille jaoks plaan tehakse. Enamasti on ajahorisondiks kuu, aga võib olla ka nädal või kvartal. Varasemad süsteemid arvutasid kogu tootmisplaani regulaarselt ümber. Uuemates süsteemides kasutatakse dünaamilisemat planeerimist, kus planeeritakse uuesti ainult osa tootmisplaanist[1].

## 2.3 Deterministlik simulatsioon *(deterministic simulation)*

Deterministlikus simulatsioonis simuleeritakse tehase tööd dispetšer-reeglitega. Tootmisplaanina kasutatakse simulatsiooni tulemust. Selline lahendus vajab, et inimene proovib läbi erinevaid reegleid ning valib välja parima plaani ning vajadusel modifitseerib seda [1].

## 2.4 Itereeritud pudelikaela lahendamise meetod *(shifting bottleneck procedure)*

Itereeritud pudelikaela lahendamise meetod (näiteks Adams jt [16] ning Balas ja Vazacopoulos [17]) jagab tootmisplaani koostamise mitmeteks ühe seadme alamprobleemiks ning lahendab iteratiivselt iga alamprobleemi. Balase ja Vazacopoulose meetod on võrreldes teiste itereeritud pudelikaela lahendamise meetodidega andnud paremaid tulemusi. Samas on see võrreldes teiste itereeritud pudelikaela lahendamise meetoditega ressursinõudlikum [10].

[2] `https://www.mie-solutions.com/`
[3] `https://frepple.com/`

## 2.5 Kitsenduste rahuldamine *(constraint satisfaction)*

Töökoja peenplaneerimise probleemi on püütud lahendada mitmete tehisintellekti algoritmidega, kuid häid tulemusi pole seni saavutatud. Jain ja Meeran [10] toovad välja, et kuigi kitsenduste lahendamist, on kasutatud paljudes süsteemides ([18, 19, 20]), pakkusid neist varasemad ainult juhiseid inimplaneerijale, hilisemad meetodid kombineerivad kitsenduste lahendamist heuristikatega ning saavad paremaid tulemusi, vajades samas palju arvutusressurssi.

## 2.6 Närvivõrgud

Kõige populaarsem närvivõrkudel põhinev peenplaneerimiseks kasutatav meetod Jaini ja Meerani põhjal on Hopfieldi võrk (näiteks Sabuncuoglu ja Gurgun [21]). Närvivõrkude treenimiseks kasutatakse andmeid otse tootmisest või siis dispetšer-reeglitega genereeritud tootmisplaane. Närvivõrkudel põhinevate meetodite suurimaks puuduseks on, et nad ei üldistu hästi teistele sarnastele proleemidele ning uued tootmisplaanid ei tohi erineda treeningandmetest rohkem kui 20%. Enamik närvivõrkudel põhinevaid meetodeid kipuvad jääma kinni lokaalsesse miinimumi ega tee globaalset optimeerimist [10].

## 2.7 Jaga ja valitse *(branch and bound)*

Jaga ja valitse [22, 23] algoritmil põhinevad peenplaneerimisalgoritmid on mõnedel juhtudel andnud häid tulemusi. Puuduseks on see, et neid ei ole võimalik rakendada suurte probleemide korral ning nad vajavad töötamiseks väga spetsiiflisi tuletus- ja valikureegelid, et kärpida lahenduste puus võimalikult kõrgelt [10].

## 2.8 Simuleeritud lõõmutamine *(simulated annealing)*

Lokaalse otsingu meetoditest kõige rohkem on peenplaneemiseks kasutatud simuleeritud lõõmutamist, näiteks van Laarhooven jt [24], Yamada jt [25], [26] ning Kolonko [27]. Jaini ja Meerani põhjal ei ole simuleeritud lõõmutamine, ilma teiste meetoditega kombineerimata, andnud peenplaneerimisel häid tulemusi. Suurim puudus simuleeritud lõõmutamisel on aga see, et paremate tulemuste saamine nõuab väga palju arvutusressurssi (vajalik võib olla teha sadu miljoneid iteratsioone), lisaks sõltub algoritm väga probleemist ning mitmed parameetrid tuleb väga täpselt valida [10].

## 2.9 Tabu otsing *(tabu search)*

Jaini ja Meerani hinnangul on parima lahenduse aja ja kvaliteedi tabu otsingut kasutades saanud Nowicki ja Smutnicki [28]. Tabu otsingut on kombineeritud ka teiste meetoditega, näiteks jaga ja valitse algoritmiga [29], mis on andud veelgi paremaid tulemusi. Põhilise miinusena tuuakse välja, et sarnaselt teiste lokaalse otsingu meetoditega vajab tabu otsing parameetrite väga täpset probleemile häälestamist [10].

## 2.10 Geneetilised algoritmid

Geneetilised algoritmid [30], [31] töötavad kitsendusi rahuldava esialgsete lahenduste hulga peal. Geneetilised algoritmid sobivad kui otsinguruum on suur ja võimalike vastuste hulk on väike, mistõttu reaalsete ülesannete lahendamisel annavad kehvasid tulemusi. Mõnevõrra paremaid tulemusi on saadud geneetilise lokaalse otsinguga [32], kuid ka see algoritmi variatsioon ei suuda väga suurte ja keeruliste ülesannete korral optimaalse lähedast tulemust anda. Põhjaliku ülevaate geneetiliste algoritmide rakendamisest on teinud Mattfeld [32].

## 2.11 Agentidel põhinevad arhitektuurid

Mitmeid eelmainitud algoritme on katsetatud ka agentidel põhinevatel arhitektuuridel [1, 3], kus iga seadet vaadatakse agendina, mis töötab inimestega samal tasemel. Selline multi-agent süsteem on kergesti ümberseadistatav, mis võimaldab ressursse väga lihtsalt lisada või eemaldada. Kuna agent-arhitektuur on detsentraliseeritud, siis suudab selline arhitektuur teoreetiliselt paremini reageerida muudatustele. Arvatakse, et ühel päeval võivad sellised multi-agent süsteemid asendada seni kasutuselolevad planeerimistarkvarad, kuid praegusel hetkel ei ole need süsteemid veel reaalselt kasutatavad [1].

## 2.12 Muutuva naabrusega otsing *(variable neighborhood search)*

Muutuva naabrusega otsing on üks uuemaid optimeerimisprobleemide lahendamiseks välja pakutud metaheuristikaid, mis väldib lokaalsesse miinimumi kinni jäämist naabruse muutmisega. Modifitseeritud muutuva naabrusega otsingut on rakendanud peenplaneerimise ülesannete lahendamiseks näiteks Sevkli ja Aydin [33]. Meetod leidis optimaalse lahenduse mitmetele väga raskeks peetud OR-Library konstrueeritud jõudlustestidele.

## 2.13 Hübriidalgoritmid

Jaini ja Meerani põhjal annavad parima tulemuse hübriidsed optimaalse lähedased algoritmid, mis kasutavad tabu otsingut, itereeritud pudelikaela eemaldamise meetodit, geneetilist lokaalset otsingut või simuleeritud lõõmutamist.

Näiteks, kombineerivad Gonçalves jt [34] geneetilist algoritmi, millega määratakse prioriteedid, mida kasutatakse parameetriseeritud tootmisplaanide koostamiseks ning lokaalset otsingut, millega parandatakse tulemust. Algoritmi võrreldi konstrueeritud jõudlustestides, kus see leidis optimaalse või peaaegu optimaalse tulemuse kõigis testides ja jäi alla vaid Nowicki ja Smutnicki [28] algoritmile. Li jt [35] kombineerivad geneetilist algortimi ja tabu otsingut. Algoritm lahendas mitmeid konstrueeritud jõudlusteste varasematest algoritmidest paremini. Zhang jt [36] pakuvad välja geneetilisel algoritmil põhineva muutuva naabrusega otsingu. Algoritmi on võrreldud teiste avaldatud algoritmidega konstrueeritud

jõudlusteste kasutades ning autorite väitel saavuati paremaid tulemusi kui mitmeid teised hübriidalgoritmid.

# 3 Süsteemid

Simeonova jt [2] on kirjeldanud, kuidas realiseerida sama peenplaneerimisülesanne Excelis[4], Lekinis[5] ning Asprovas. Excel võimaldas peenplaneerimise odavalt lahendada, kuid oli raske kasutada ning nõudis teadmiseid valemitest. Lekin, mis on hariduslikul eesmärgil tehtud tööriist peeplaneerimise algoritmide õpetamiseks ja väljatöötamiseks, võimaldas planeerida suurte piirangutega ning ei võimaldanud optimeerimist. Lekini puuduseks on ka see, et seda ei ole aastast 2010 edasi arendatud. Asprova võimaldas genereerida ja hallata tootmistellimusi ning ostutellimusi. Asprova tehtud tootmisplaan ei olnud optimaalne ning mõned tellimused ei jõudnud tähtajaks valmis. Genereeritud tootmisplaani oli võimalik optimeerida rakendades Asprova funktsionaalust, mis võimaldab näiteks ühes operatsioonis kogust vähendada. Peamiseks miinuseks toodi hind.

Lisaks ERP-süsteemidele saab peenplaneerimiseks kohandada laiaotstarbelisi planeerijaid, näiteks vabavaralist OptaPlanner-it[6], kus marsruudid ning ressursside eelistused tuleb kirjeldada kitsendustena. OptaPlanneri puhul on tegemist planeerimissüsteemiga, mis kasutab oma kitsenduste lahendajas mitmeid juba mainitud meetodeid, nagu tabu otsingut ja simuleeritud lõõmutamist. Kuna tegemist on laiaotstarbelise planeerijaga, siis erinevalt ERPidest ei ole OptaPlanneril mugavat kasutajaliidest.

Võrdluseid süsteemide funktsionaalsuste vahel on tehtud kirjanduse põhjal. Dios ja Framinan [37] on võrrelnud süsteeme selle põhjal, kuidas nad võimaldavad lahendatava ülesande kirjeldamist, kas süsteem võimaldab tootmisplaani ümberplaneerimist, kui palju kasutatakse inimest eksperdina, kas ja kuidas hinnatakse tulemuse kvaliteeti ning millisel kujul esitatakse väljund. Tulemusena tuuakse välja, et üllatavalt paljud süsteemid kasutavad väga lihtsaid algoritme (näiteks ettpoole planeerimist) või põhinevad palju inimeste otsustel, ega kasuta jooksvalt ümberplaneerimist. Samuti on enamik süsteeme kindla probleemi spetsiifilised ning ei ole kasutatavad kui probleem natukegi erineb.

Reaalses tootmises kasutatavad ERP süsteemid täpseid detaile oma tööpõhimõtete kohta ei jaga, kuid dokumentatsioonid mainivad ära ettepoole ja tagasisuunas planeerimise kombineerimise [14, 38, 11]. Kuna ERP süsteemide puhul on enamasti tegemist tasulise tarkvaraga, mille tootmises juurutamine ja testimine on töömahukas, siis ei ole avaldatud võrdlusi erinevate ERPide planeerimistulemuste kohta. Raskuseid, mis pilvepõhise ERP süsteemi kasutuselevõtuga kaasnevad on kirjeldanud Gupta jt [39].

[4] `http://production-scheduling.com/`

[5] `http://web-static.stern.nyu.edu/om/software/lekin/`

[6] `http://www.optaplanner.org/`

# 4 Kokkuvõte

Töökoja-tüüpi peenplaneerimise ülesanne on teaduslikult huvitav ja keeruline probleem. Seda on väga palju uuritud ning proovitud lahendada mitmesuguste optimeerimis-, planeerimis- ja otsingualgoritmidega. Ülesanne on endiselt lahendamata, see tähendab, et ei ole leitud optimaalse planeerimise algoritmi, mis leiaks mõistliku ajaga reaalsetele ülesannetele optimaalse tootmisplaani. Optimaalsuse lähedase planeerimisega algoritmidest on kirjanduse põhjal parimaid tulemusi on andnud hübriidalgoritmid, mis kombineerivad mitut erinevat algoritmi ja meetodit. Viimase kümne aasta jooksul välja pakutud algoritmid on just hübriidalgoritmid. Kahjuks ei ole aga avaldatud neid algoritme võrdlevaid artikleid. Algoritme, mis kirjanduse andmetel annavad häid tulemusi, on tavaliselt testitud konstrueeritud operatsioonianalüüsi jõudlustestidega, mitte reaalsete tootmisülesannetega. Ka konstrueeritud jõudlustestides ei anna enamik algoritme häid tulemusi. Seega, on reaalsetes ülesannetes ja süsteemides võimalik kasutada vaid heuristilisi optimaalse lähedasi planeerimismeetodeid, nagu ettepoole ja tagasisuunas peenplaneerimine.

# Viited


[1] Albert D. Baker. A survey of factory control algorithms that can be implemented in a multi-agent heterarchy: dispatching, scheduling, and pull. *Journal of Manufacturing Systems*, 17(4):297–320, 1998.

[2] Ivana Simeonovova, Aleksandar Gyurov, and Simeon Simeonov. Modern methods for manufacturing planning and scheduling. *MM Science Journal.[Online]*, pages 635–639, 2015.

[3] Djamila Ouelhadj and Sanja Petrovic. A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling*, 12(4):417, Oct 2008.

[4] Banu Çaliş and Serol Bulkan. A research survey: review of ai solution strategies of job shop scheduling problem. *Journal of Intelligent Manufacturing*, 26(5):961–973, Oct 2015.

[5] John E Beasley. Or-library: distributing test problems by electronic mail. *Journal of the operational research society*, 41(11):1069–1072, 1990.

[6] P. Surekha and S. Sumathi. Solving fuzzy based job shop scheduling problems using GA and ACO. 2010.

[7] J. Heinonen and F. Pettersson. Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem. *Applied Mathematics and Computation*, 187(2):989–998, 2007.

[8] Zhigang Lian, Bin Jiao, and Xingsheng Gu. A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan. *Applied mathematics and computation*, 183(2):1008–1017, 2006.

[9] Tsung-Lieh Lin, Shi-Jinn Horng, Tzong-Wann Kao, Yuan-Hsin Chen, Ray-Shine Run, Rong-Jian Chen, Jui-Lin Lai, and I-Hong Kuo. An efficient job-shop scheduling algorithm based on particle swarm optimization. *Expert Systems with Applications*, 37(3):2629–2636, 2010.

[10] Anant Singh Jain and Sheik Meeran. A state-of-the-art review of job-shop scheduling techniques. Technical report, Technical report, Department of Applied Physics, Electronic and Mechanical Engineering, University of Dundee, Dundee, Scotland, 1998.

[11] Dispatching. `https://lib.asprova.com/onlinehelp/en/AS2003HELP00765000.html`. [WWW; viimati vaadatud 23. aprill 2018].

[12] D. Sun and L. Lin. A dynamic job shop scheduling framework: a backward approach. *The International Journal of Production Research*, 32(4):967–985, 1994.

[13] Jelena Sanko and Vahur Kotkas. Ontology-Driven Scheduling System for Manufacturing. *Baltic Journal of Modern Computing*, 4(3):508–522, 2016.

[14] Backward or forward scheduling? `https://frepple.com/learn/backward-forward-scheduling/`. [WWW; viimati vaadatud 23. aprill 2018].

[15] Eliyahu M Goldratt. *Theory of constraints*. North River Croton-on-Hudson, 1990.

[16] Joseph Adams, Egon Balas, and Daniel Zawack. The shifting bottleneck procedure for job shop scheduling. *Management science*, 34(3):391–401, 1988.

[17] Egon Balas and Alkis Vazacopoulos. Guided local search with shifting bottleneck for job shop scheduling. *Management science*, 44(2):262–275, 1998.

[18] Cheng-Chung Cheng and Stephen F Smith. Applying constraint satisfaction techniques to job shop scheduling. *Annals of Operations Research*, 70:327–357, 1997.

[19] Wim Nuijten and Claude Le Pape. Constraint-based job shop scheduling with ILOG SCHEDULER. *Journal of Heuristics*, 3(4):271–286, 1998.

[20] Norman Sadeh and Mark S. Fox. Variable and value ordering heuristics for the job shop scheduling constraint satisfaction problem. *Artificial intelligence*, 86(1):1–41, 1996.

[21] Ihsan Sabuncuoglu and Burckaan Gurgun. A neural network model for scheduling problems. *European Journal of Operational Research*, 93(2):288–299, 1996.

[22] Peter Brucker, Bernd Jurisch, and Bernd Sievers. A branch and bound algorithm for the job-shop scheduling problem. *Discrete applied mathematics*, 49(1-3):107–127, 1994.

[23] Paul Martin and David B Shmoys. A new approach to computing optimal schedules for the job-shop scheduling problem. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 389–403. Springer, 1996.

[24] Peter JM Van Laarhoven, Emile HL Aarts, and Jan Karel Lenstra. Job shop scheduling by simulated annealing. *Operations research*, 40(1):113–125, 1992.

[25] Takeshi Yamada, Bruce E. Rosen, and Ryohei Nakano. A simulated annealing approach to job shop scheduling using critical block transition operators. In *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, volume 7, pages 4687–4692. IEEE, 1994.

[26] Takeshi Yamada and Ryohei Nakano. Job-shop scheduling by simulated annealing combined with deterministic local search. In *Meta-Heuristics*, pages 237–248. Springer, 1996.

[27] Michael Kolonko. Some new results on simulated annealing applied to the job shop scheduling problem. *European Journal of Operational Research*, 113(1):123–136, 1999.

[28] Eugeniusz Nowicki and Czeslaw Smutnicki. A fast taboo search algorithm for the job shop problem. *Management science*, 42(6):797–813, 1996.

[29] S. Thomsen. Metaheuristics combined with branch & bound. Technical report, 1997.

[30] B. Norman. A random keys genetic algorithm for job shop scheduling. *Engineering Design & Automation*, 3(2):145–156, 1997.

[31] Ulrich Dorndorf and Erwin Pesch. Evolution based learning in a job shop scheduling environment. *Computers & Operations Research*, 22(1):25–40, 1995.

[32] Dirk C. Mattfeld. *Evolutionary search and the job shop: investigations on genetic algorithms for production scheduling.* Springer Science & Business Media, 2013.

[33] Mehmet Sevkli and M Emin Aydin. A variable neighbourhood search algorithm for job shop scheduling problems. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 261–271. Springer, 2006.

[34] José Fernando Gonçalves, Jorge José de Magalhães Mendes, and Mauricio GC Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European journal of operational research*, 167(1):77–95, 2005.

[35] Xinyu Li and Liang Gao. An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. *International Journal of Production Economics*, 174:93–110, 2016.

[36] Guohui Zhang, Lingjie Zhang, Xiaohui Song, Yongcheng Wang, and Chi Zhou. A variable neighborhood search based genetic algorithm for flexible job shop scheduling problem. *Cluster Computing*, Jan 2018.

[37] Manuel Dios and Jose M. Framinan. A review and classification of computer-based manufacturing scheduling tools. *Computers & Industrial Engineering*, 99:229–249, 2016.

[38] What is forward and backward scheduling. `https://www.mie-solutions.com/estimating/what-is-forward-and-backward-scheduling/`. [WWW; viimati vaadatud 23. aprill 2018].

[39] Shivam Gupta, Subhas C Misra, Akash Singh, Vinod Kumar, and Uma Kumar. Identification of challenges and their ranking in the implementation of cloud ERP: A comparative study for SMEs and large organizations. *International Journal of Quality & Reliability Management*, 34(7):1056–1072, 2017.