Univerza v Ljubljani

Fakulteta za elektrotehniko

Dejan Gradišar

RAČUNALNIŠKO PODPRTA GRADNJA MODELOV ZA POTREBE RAZVRŠČANJA PROIZVODNIH OPRAVIL

DOKTORSKA DISERTACIJA

Mentor: izred. prof. dr. Gašper Mušič

Ljubljana, 2006

Zahvala

Iskreno se zahvaljujem mentorju izred. prof. dr. Gašperju Mušiču za pomoč pri mojem podiplomskem študiju in za dragocene pripombe ter predloge pri nastajanju tega dela.

Zahvaljujem se tudi ostalim sodelavcem Laboratorija za modeliranje, simulacijo in vodenje ter Laboratorija za avtomatizacijo in informatizacijo procesov, ki so mi kakorkoli pomagali pri mojem delu. Za konstruktivne ideje in nasvete se zahvaljujem tudi Thomasu Löscherju.

Posebej pa bi se rad zahvalil mojim staršem, sestri in prijateljem, ki so me na moji študijski poti spremljali in me podpirali.

Povzetek

Ustrezna informacijska podpora je sedaj že nekaj časa nujna za uspešno poslovanje podjetja. Upravljanje proizvodnega podjetja se vrši na večih nivojih, ki pa mora biti čimbolj usklajeno. Ena pomembnejših funkcij je razvrščanje opravil v proizvodnji. Sistem za razvrščanje sprejema naloge s poslovnega nivoja, ki jih mora razvrstiti na razpoložljivo delovno okolje. Pri tem je potrebno upoštevati vrsto omejitev.

Disertacija obravnava problematiko razvrščanja opravil v proizvodnji in zagotavljanja matematičnih opisov proizvodnih sistemov, da je razvrščanje možno. Teorija razvrščanja raziskuje postopke izdelave učinkovite razvrstitve ob prisotnosti omejitev, na katere naletimo v praksi. V preteklosti so bila razvita številna pravila razvrščanja, ki obravnavajo različno kompleksne modele in zadostujejo različnim kriterijem in zahtevam po učinkovitosti. Tako je v nadaljevanju pričakovati večji napredek pri modeliranju problemov in ne toliko pri iskanju rešitve. Različne metode razvrščanja smo predstavili glede na obseg in strukturo modela.

Na proizvodni sistem lahko gledamo kot na diskretno dogodkovni sistem, katerega obnašanje je običajno zelo kompleksno. S pomočjo formalnih metod, s katerimi matematično predstavimo bistvene lastnosti sistema, je razumevaje takih sistemov bistveno lažje. V delu obravnavamo orodja, ki so zmožna poenotene obravnave proizvodnih sistemov prek celega njihovega življenjskega cikla, poudarek pa je na uporabi tako zgrajenih modelov za namene razvrščanja.

Glavni poudarek je na uporabi Petrijevih mrež. Izkaže se, da je za študij časovnega obnašanja sistema in reševanja problemov razvrščanja potrebno v osnovno definicijo Petrijevih mrež vpeljati tudi časovne zakasnitve. V literaturi se pojavlja več pristopov k vpeljavi časa. V delu uporabljamo časovne Petrijeve mreže, ki delujejo po načelu časa trajanja držanja. Model proizvodnega sistema, predstavljen s časovnimi Petrijevimi mrežami, lahko analiziramo z različnimi me-

todami analize. Pogosto se pojavljajo kompleksni primeri, ko se simulacija izkaže za najučinkovitejšo metodo analize. S simulacijo lahko opazujemo izvajanje Petrijeve mreže skozi čas. Da je simulacija možna, pa je ob pojavu konflikta treba uporabiti določene odločitvene strategije. V primeru obravnave proizvodnih sistemov kot odločitvene strategije uporabimo različna hevristična pravila. Tako simulator lahko uporabimo kot razvrščevalnik opravil. V delu je predstavljena simulacijska funkcija časovne Petrijeve mreže, ki je bila razvita v okolju Matlab.

Petrijeve mreže predstavljajo družino orodij, ki jih lahko uporabimo za različne probleme, ki se pojavljajo preko življenjskega cikla proizvodnih sistemov. Tako lahko z njimi študiramo različne poslovne strategije, ki se uporabljajo v sistemih za upravljanje in planiranje. S temi sistemi so določene naloge, ki jih je potrebno izvesti znotraj proizvodnega sistema. Proizvodi proces lahko razstavimo na osnovne aktivnosti. Prikazano je, kako s Petrijevimi mrežami predstavimo posamezne aktivnosti.

Sistemi za upravljanje in planiranje za svoje delovanje potrebujejo informacijo o strukturi izdelkov, statusu zalog in strukturi proizvodnega sistema. Te podatke lahko s pridom uporabimo v sistemih za razvrščanje, ki se uporabljajo v povezavi s sistemi za planiranje. Pri tem mora sistem za razvrščanje pridobiti tudi podatke o statusu proizvodnih virov s procesnega nivoja. Kadar imamo na razpolago omenjene podatke, lahko do neke mere avtomatiziramo postopek gradnje modela in s tem tudi olajšamo naknadno analizo. V disertaciji smo predstavili algoritem, ki upošteva zakonitosti tako na poslovnem nivoju kot tudi na nivoju razvrščanja, in ki omogoča gradnjo modela proizvodnega sistema s časovnimi Petrijevimi mrežami. Uporabnost prototipa smo predstavili na dveh različnih proizvodnih okoljih.

Tudi projekti so opisani z nizom aktivnosti, ki so med sabo smiselno povezane, tako da opravijo neko specifično nalogo. Pri planiranju in vodenju projektov je potrebno sprejeti vrsto odločitev. Obstaja več orodij, ki vsebujejo različne tehnike za pomoč pri delu s projekti. Obstaja več sorodnih pogledov na vodenje projektov in razvrščanje proizvodnih opravil. Tako so orodja za vodenje projektov do neke mere uporabna tudi za razvrščanje v proizvodnji. V delu smo preučevali določene lastnosti orodja Microsoft Project, sicer tipičnega predstavnika orodij za vodenje projektov. V podporo generiranju razvrstitve opravil je

bila znotraj orodja razvita rešitev, ki je bila zasnovana v obliki postopka v jeziku VBA. Postopek omogoča vključitev orodja v obstoječ informacijski sistem in njegovo uporabo pri razvrščanju. Uporabnost omenjenega postopka smo predstavili na primeru šaržnega procesa.

Abstract

Appropriate information technology support has become a prerequisite for a successful and efficient management of production companies, one through which all essential management functions are controlled. Among the most important ones is scheduling function, which has to interface with many other functions. Orders are released to the scheduling system and have to be translated into tasks, which then have to be processed by machines. Usually there are many constraints to be considered.

The thesis deals with the problem of task scheduling in a production system and assuring the mathematical descriptions of production systems. An adequate model is required in order to derive a feasible schedule. The theory of scheduling investigates the methods of producing effective schedules where all practical constraints are considered. In the past, many scheduling rules were developed which deal with different complex models and satisfy different criteria and requirements. In the future, more progress is therefore expected in the area of problem modelling, and not so much in the area of development of techniques for achieving a solution. Different scheduling techniques are presented with regard to the scope and the structure of a model.

Production systems can be seen as discrete-event systems whose behaviour is very complex. Formal methods usually improve the understanding of these systems, allow their analysis and help in their implementation. In the thesis, different mathematical methods are discussed that are usable along the life cycle design. Their abilities to model scheduling problems are investigated.

The main stress was given on the usage of the Petri nets. For the performance evaluation and scheduling problems of dynamical systems it is necessary to introduce time delays in the original definition of Petri nets. There are different ways of representing time in Petri nets. In the thesis timed Petri nets that use the holding duration principle are used. Production systems that are modelled with

timed Petri nets can be analyzed using different techniques. For complex models simulation turns out to be the most effective technique. With simulation the evolution of the marking trough time can be observed. Different heuristic rules can then be introduced when solving the situations in which conflicts occur. When production systems are considered, different rules are used to satisfy different predefined production objectives. In this way the schedule of process operations can be observed. In the thesis the simulation function of timed Petri net was presented and was developed in the Matlab environment.

Petri nets are a family of tools that provide a framework or working paradigm which can be used for many of the problems that appear during the lifecycle of production systems. They allow the examination of different production-management strategies. With production-management systems, the work orders which have to be carried out within the production system are determined. The production process can be broken down into basic production activities. The thesis presents the process of modelling these activities with timed Petri nets.

For their operation, production-management systems require data about the inventory status, process structure and product structure. These data, together with the information about the resource availability, form the basic elements of the manufacturing process and can be effectively used to build up a detailed model of the production system which is needed for scheduling purposes. Provided that some data are available, the modelling procedure can be automated to a certain extent. In the thesis an algorithm that considers the principles given in the production-management systems, and makes it possible to build a model of a production system using timed Petri nets is presented. The applicability of the algorithm was demonstrated on two different production environments.

The projects are also described through a set of activities that are connected with an intention to perform a specific task. During the planning and management of projects, many decisions have to be performed. Many tools encompassing many techniques exist which can help when dealing with projects. Many similar views on project management and production scheduling can be found. It follows that to some extent the project-management tools can also be used for production scheduling purposes. In this thesis some properties of a Microsoft Project are investigated, which is a typical representative of tools intended for project

management. To support a schedule generation, the procedure was developed using VBA language. The procedure makes it possible to integrate the tool into the existing information system, and to use it for the scheduling purposes. The applicability of the procedure was demonstrated on the example of the batch system.

Originalni prispevki disertacije

- **Pregled pristopov k reševanju problema razvrščanja, kjer so metode obravnavane glede na kompleksnost modela proizvodnega procesa.**

  Teorija razvrščanja raziskuje postopke izdelave učinkovitosti razvrstitev ob prisotnosti raznih proizvodnih omejitev. Teorija se intenzivno naslanja na matematična orodja oziroma na matematične modele, ki praktične probleme abstrahirajo v matematično obliko. Razvita so bila številna pravila razvrščanja, ki smo jih v delu obravnavali glede na kompleksnost modela.

- **Zasnova in izdelava prototipa programskega orodja za podporo modeliranju s časovnimi Petrijevimi mrežami.**

  Petrijeva mreža je družina formalizmov, ki podajajo okvir za obravnavo proizvodnih sistemov skozi cel življenjski cikel proizvodnega procesa. Celoten proizvodni sistem lahko razstavimo na osnovne aktivnosti, ki jih lahko modeliramo s časovnimi Petrijevimi mrežami. Zasnovali in izdelali smo prototip programskega orodja za opis modela proizvodnega procesa s časovnimi Petrijevimi mrežami.

- **Razvoj simulatorja časovnih Petrijevih mrež z vgrajenimi različnimi možnostmi razreševanja konfliktov v Petrijevi mreži, kar omogoča uporabo simulatorja za razvrščanje.**

  Pri obravnavi kompleksnih diskretno dogodkovnih sistemov postane uporaba analitičnih metod nemogoča ali je uporabna le v omejenih količinah. V takih primerih se poslužujemo simulacije. S simulacijo časovne Petrijeve mreže oponašamo njeno izvajanje skozi čas. Da je simulacija možna je potrebno ob pojavu konflikta vpeljati neke odločitvene strategije. V okolju Matlab smo razvili simulacijsko funkcijo časovne Petrijeve mreže, s

pomočjo katere lahko določimo sled označitve časovne Petrijeve mreže in s tem razvrstitev opravil v modeliranem problemu razvrščanja.

- **Analiza uporabnosti orodij za vodenje projektov za namene razvrščanja proizvodnih opravil.**

  Projekt je opisan kot niz nerutinskih aktivnosti, ki so med sabo povezane z namenom, da opravijo neko specifično nalogo. Orodje za vodenje projektov je nepogrešljiv pripomoček za doseganje optimalnega plana. Tudi proizvodni procesi so sestavljeni iz večih procesnih korakov oz. aktivnosti. V delu smo preučevali določene lastnosti orodja Microsoft Project, sicer tipičnega predstavnika orodij za vodenje projektov. Definirali smo preslikavo elementov, ki se pojavljajo pri upravljanju projektov, na problem razvrščanja.

- **Dva načina povezave obstoječih proizvodnih informacijskih sistemov s postopki razvrščanja:**

  - **avtomatska gradnja modela s Petrijevimi mrežami na podlagi obstoječih podatkov ter uporaba dobljenega modela za razvrščanje,**
  - **prilagoditev orodja za vodenje projektov na način, ki omogoča vključitev v obstoječ informacijski sistem in uporabo za razvrščanje.**

  V proizvodnih okoljih se običajno uporabljajo sistemi za planiranje, ki za svoje delovanje potrebujejo informacijo o strukturi izdelka, statusu zalog in strukturi proizvodnega sistema. Rezultat planiranja so naloge, ki jih je potrebno izvršiti za dosego nekega cilja. Takim sistemom je možno pripojiti tudi dodatno orodje za razvrščanje teh nalog.

  Razvili smo pripomoček za avtomatsko gradnjo modela s Petrijevimi mrežami na osnovi podatkov, ki so na voljo v sistemih za planiranje. Tak model lahko uporabimo za razvrščanje opravil v proizvodnji.

  V podporo generiranju razvrstitve opravil smo razvili tudi rešitev, integrirano znotraj Microsoft Projecta. Rešitev smo zasnovali v obliki postopka v jeziku VBA, s katerim zgradimo model proizvodnega sistema, ki ga lahko zopet uporabimo za razvrščanje opravil.

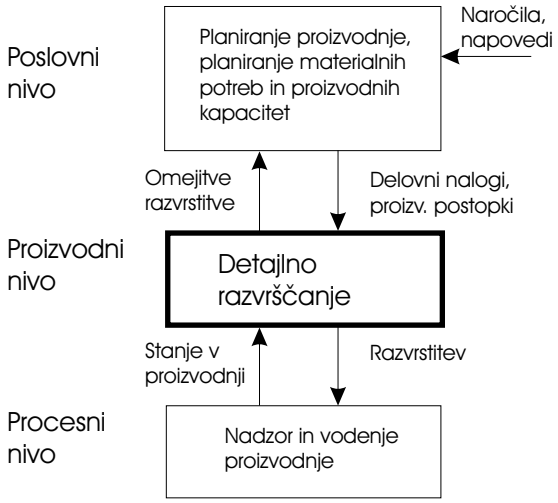
Vsebina

1. Uvod

Podjetja morajo iskati uspeh in s tem dejavnike uspeha predvsem v svojem okolju in ne samo v podjetju samem. Tako je za uspešno poslovanje podjetja že kar nekaj časa nuja, da je fleksibilno in prilagodljivo na spremembe na trgu. Pri tem fleksibilnost označuje sposobnost prilagajanja proizvodnega sistema na zahteve potrošnika, prilagodljivost pa govori o hitrosti preoblikovanja sistema v stanje, ko lahko spet učinkovito deluje v spremenjenem okolju. Načrtovanje in vodenje takega proizvodnega sistema je zelo zahtevna naloga, potrebna so različna znanja, izkušnje in ogromno informacij. Pri tem je nujno sodelovanje ljudi iz različnih področij. Zato je potreben pristop, ki vsem vpletenim omogoča komuniciranje in delo na skupnem cilju ter omogoča celovito obvladovanje vseh bistvenih funkcij poslovanja proizvodnega podjetja.

Upravljanje proizvodnje lahko razdelimo na tri nivoje [5]. Sprejemanje dolgoročnih odločitev na strateškem nivoju je proces vzpostavljanja skupnih ciljev in organizacijskih kriterijev. Tu se ukvarjamo z vprašanji, katere izdelke izdelovati, katere vire pri tem uporabiti, z zagotavljanjem vhodnih materialov ipd. S taktičnimi odločitvami izbiramo metode, s katerimi lahko dosežemo zastavljene kriterije. S srednjeročnimi plani sicer omejujemo, vendar podajamo stabilno (izvedljivo) osnovo za planiranje na obratovalni stopnji. Odločitve na stopnji obratovanja so potrebne za kratkoročno vodenje proizvodnih in upravljalnih procesov. Odločitve se sprejema z veliko mero pogostosti in na ta način dobimo detajlni kratkoročni plan. Primer takega odločanja je razvrščanje proizvodnih opravil na razpoložljive proizvodne vire.

Funkcija razvrščanja mora znotraj proizvodnega podjetja sodelovati še z mnogimi ostalimi funkcijami, ki so potrebne za upravljanje proizvodnje. Te povezave so močno odvisne od sistema, s katerim imamo opravka. Aktivnosti, ki so potrebne za vodenje proizvodnega podjetja, so z organizacijskim modelom [35] raz-



Slika 1.1: Umestitev razvrščanja

deljene v več ravneh, glej sliko 1.1.

Na *poslovnem nivoju* se izvaja osnovno, strateško planiranje proizvodnje. Tu se upošteva stanje materiala oz. zalog (*Inventory levels*), napovedi potreb ter potrebe po proizvodnih virih. Tako se načrta proizvodni postopek in se izvede dolgoročno dodeljevanje virov (*Resource allocation*). Med proizvajanjem izdelkov se na *procesnem nivoju* (*Shopfloor control*) ustvarjajo informacije o stanju proizvodnega procesa, porabi materiala in energije, kakovosti izdelkov, zastojih, okvarah itd. Informacije, potrebne za popravljanje planov, se v prirejeni obliki prenašajo višjim nivojem. *Proizvodni nivo* vodenja podjetja zapolnjuje vrzel med aktivnostmi vodenja poslovnega in procesnega nivoja. Na njem se izvaja vrsta ključnih aktivnosti, ki močno vplivajo na uspešnost podjetja. Njena bistvena aktivnost je funkcija planiranja in razvrščanja opravil v proizvodnji, ki se mora vključevati v celoten sistem informacijske podpore vodenja proizvodnega sistema. Način uvedbe sistema za podporo razvrščanju je odvisen od obravnavanega proizvodnega okolja in situacije, v kateri se nahaja.

Skozi življenjski cikel proizvodnega procesa nastopajo najrazličnejši tipi problemov. Potreba po fleksibilni in prilagodljivi proizvodnji zahteva skrajševanje življenjskega cikla. V tem pogledu lahko v splošnem probleme razdelimo na dve fazi. V fazi načrtovanja nastopajo problemi, ki se nanašajo na zasnovo in postavitev proizvodnega sistema. Na tej stopnji se izvajajo strateške in taktične odločitve. V fazi obratovanja pa se v glavnem pojavljajo problemi, ki se nanašajo

na dodeljevanje virov, razvrščanje, vodenje in nadzor sistema.

Različne tipe problemov lahko razlikujemo tudi po proizvodnem okolju, v katerem se pojavljajo [44, 86]. Le-ta lahko, glede na značilnosti razvrščanja, delimo na kosovno, procesno in šaržno proizvodnjo. V kosovni proizvodnji imamo opravka z mehansko obdelavo obdelovancev, material se ne spreminja. Taka proizvodnja je tipična za avtomobilsko industrijo, orodjarne, ipd. Razvrstitev naj bi zagotavljala razporeditev obdelovancev po virih, tako da so kapacitete uporabljene čimbolj učinkovito. V procesni industriji nastopa zvezen materialni tok skozi posamezne procesne enote. Tak primer se pojavlja v rafinerijah, papirnih industrijah, itd. Funkcija razvrščanja naj v tem primeru zagotovi stabilno in enakomerno proizvodnjo. V šaržni proizvodnji pa imam opravka s prekinjajočim materialnim tokom [31] in je značilna za farmacevtsko ali pa živilsko industrijo. Z razvrstitvijo zagotavljamo stabilno in enakomerno proizvodnjo, poleg tega pa naj zagotavlja tudi čimbolj učinkovito izrabo kapacitet, npr. preprečevanje ozkih grl.

Če se še enkrat vrnemo k sliki 1.1, vidimo, da je sistem za detajlno razvrščanje integriran v celoten informacijski sistem podjetja. Sistem za razvrščanje sprejema naročila prenesena in pretvorjena v opravila/naloge (*Jobs*) s prirejenimi izvedbenimi roki (*Due dates*). Običajno morajo biti naloge znotraj delovnega okolja procesirane s proizvodnimi viri (stroji) po danem zaporedju oz. razporedu, kot to zahteva proizvodni postopek. Z razvrščanjem želimo doseči optimalno proizvodnjo ob sočasnem upoštevanju vseh omejitev tehnološkega procesa, razpoložljivosti proizvodnih virov, vrste naročil in rokov izdelave. Tako dobimo razvrstitev, ki določa, kako se bodo izvajala posamezna proizvodna opravila. Razvrstitev se pošilja na procesni nivo, t.j. posameznim delovnim oddelkom v proizvodni organizaciji.

Postopke izdelave učinkovitih razvrstitev ob prisotnosti omejitev, na katere naletimo v praksi pri izvajanju operacij, raziskuje teorija razvrščanja. Teorija se intenzivno naslanja na matematična orodja, za kar potrebuje matematične modele, ki praktične probleme abstrahirajo v matematično obliko. Težava, ki nastopa, je, da preprosti modeli premalo natančno opisujejo realna dogajanja. Po drugi strani pa je kompleksne in bolj precizne modele težje uporabljati. Določanje eksaktne rešitve, ki predstavlja optimalno razvrstitev, v mnogih primerih velja

za NP-težak problem[1]. Zato se pri razvrščanju opravil v praksi ne uporabljajo eksaktni postopki, ki bi zagotovili optimalno rešitev. Običajno se zadovoljimo s hevrističnimi pristopi, ki omogočajo zadovoljivo rešitev v sprejemljivem času.

Področje teorije razvrščanja je zelo razvito, razvitih je bilo ogromno tehnik, s katerimi lahko pridemo do rešitve problema razvrščanja. Tako je po mnenju Abdeddaima s sod. [1] nadaljnji napredek pričakovati bolj v smeri iskanja pristopov k modeliranju problemov razvrščanja, ki bi zagotavljali jasen shematičen model. Lee s sod. [50] pa je mnenja, da je potrebno razvijati modele, ki nam služijo za razvrščanje, v razširjeni obliki, tako da vsebujejo vse več praktičnih omejitev.

Matematični model, ki nam naj bi služil za razvrščanje opravil, naj opisuje vse pomembne lastnosti proizvodnega sistema. Proizvodni sistem lahko obravnavamo kot diskretno dogodkovni sistem. Na žalost pa k modeliranju takih sistemov ne obstaja nek univerzalen pristop. Konvencionalna orodja (zvezne diferencialne in diferenčne (ne)enačbe) ne zadoščajo za opis takih sistemov. V ta namen je bilo razvitih več različnih matematičnih formalizmov, kot so Petrijeve mreže, časovni avtomati, disjunktivni graf. Predvsem Petrijeve mreže se izkažejo za uporabno orodje za analizo proizvodnih sistemov, saj model zapisan s Petrijevo mrežo lahko opisuje prednostne povezave med dogodki, dogodke, ki so med sabo konkurenčni in medsebojno odvisni, ker lahko modeliramo skupne vire in vse ostale omejitve, ki vplivajo na učinek proizvodnega sistema. Poleg tega pa je Petrijeve mreže mogoče uporabiti tudi na ostalih nivojih upravljanja proizvodnje.

Glede na to, da je funkcija razvrščanja vključena v informacijski sistem podjetja, je za gradnjo matematičnega modela smiselno uporabiti podatke, ki so na voljo v informacijskem sistemu, in tako zagotoviti enovito obravnavanje planiranja in razvrščanja. Iz sistemov za planiranje proizvodnje je mogoče preko delovnih nalogov pridobiti podatke o strukturi izdelka in proizvodnem postopku, ki zagotavlja njegovo izdelavo.

[1] Teorija kompleksnosti kvalificira več razredov kompleksnosti. Večino problemov razvrščanja se uvršča med NP-težke (*Non-deterministic Polynomial-time hard*) in strogo NP-težke [74]. Razred NP sestavljajo tisti odločitveni problemi, ki jih lahko z nedeterminističnim Turingovim strojem rešujemo z algoritmom polinomske časovne zahtevnosti, medtem ko strogo NP-težki problemi niso rešljivi v polinomskem času. Polinomski čas se nanaša na čas računanja problema, kjer čas $m(n)$ ni večji od polinomske funkcije reda n.

Doktorska disertacija obravnava problematiko razvrščanja opravil v proizvodnji in zagotavljanja formalnih predstavitev proizvodnih sistemov, s katerimi se lahko lotimo razvrščanja. Razdeljena je na šest poglavij.

Drugo poglavje podaja pregled metod razvrščanja. Problem razvrščanja je predstavljen s podatki o strukturi proizvodnega sistema, kriterijsko funkcijo, ki jo želimo optimizirati, in dodatnimi zahtevami in omejitvami procesa. Pristopi k reševanju problema razvrščanja so predstavljeni glede na strukturo proizvodnega sistema. Rezultat razvrščanja je običajno grafično predstavljen z gantogramom.

V tretjem poglavju je podan pregled matematičnih formalizmov, ki nam omogočajo matematično obravnavo problemov, ki nastopajo pri obravnavi proizvodnih sistemov. Poudarek je na orodjih, ki omogočajo poenoteno obravnavo proizvodnih sistemov tako v fazi načrtovanja kot tudi v fazi obratovanja. Predvsem v fazi obratovanja, ko se zahteva kvantitativna analiza sistemov, je potrebno razpolagati z orodji, ki omogočajo gradnjo časovnih modelov. Praktična uporabnost teh formalizmov je predvsem pogojena s prisotnostjo primernih računalniških orodij.

V četrtem poglavju je podrobneje obravnavana uporabnost Petrijevih mrež pri načrtovanju in vodenju proizvodnih sistemov. Petrijeve mreže se uporabljajo za raznolike namene znotraj področja proizvodnih sistemov. Kot matematično orodje vsebujejo številne lastnosti, ki jih lahko s pridom izkoristimo za ugotavljanje (ne)prisotnosti raznih funkcionalnih lastnosti sistema. Ker je uporaba analitičnih metod pogostokrat nemogoča, se pojavlja potreba po simulaciji. Petrijeve mreže lahko uporabimo tako za modeliranje raznih poslovnih strategij, kot tudi za, v primeru razširitve s časovno informacijo, detajlno modeliranje proizvodnih postopkov.

Peto poglavje obravnava okvirje za avtomatsko gradnjo modela proizvodnega sistema v obliki Petrijeve mreže, s pomočjo podatkov, ki nastopajo v poslovnih informacijskih sistemih. Predstavljen je postopek gradnje modela in njegova implementacija v okolju Matlab. Podana sta tudi dva praktična primera kosovne in šaržne proizvodnje, nad katerima je predstavljen postopek modeliranja in analize.

V šestem poglavju je obravnavana problematika razvrščanja proizvodnih opravil z orodji za vodenje projektov. Vodenje projektov je pomembna funkcija tudi

v proizvodni industriji v fazi načrtovanja izdelka. Projekti pa so si s proizvodnimi procesi podobni tudi v tem, da so sestavljeni iz večih proizvodnih korakov oz. aktivnosti. Kot primer takega orodja bomo predstavili Microsoft Project. S pomočjo postopka, vgrajenega v MS Project s pomočjo jezika VBA, smo razširili funkcionalnosti orodja tako, da nam je v pomoč pri detajlnem razvrščanju. Tudi v tem primeru se postopek oskrbuje s podatki o proizvodnem procesu iz poslovnih sistemov. Uporabnost predstavljenega postopka je prikazan na primeru večnamenskega šaržnega procesa.

V sedmem poglavju so podani zaključki v smislu prispevkov disertacije.

2. Problematika razvrščanja

Teorija razvrščanja raziskuje postopke izdelave učinkovitih razvrstitev ob prisotnosti omejitev, na katere naletimo v praksi pri izvajanju proizvodnih operacij. Teorija se intenzivno naslanja na matematična orodja, za kar potrebuje matematične modele, ki praktične probleme abstrahirajo v matematično obliko. Splošen matematični model, ki bi zajel vso raznolikost problemov razvrščanja, ne obstaja. Kompleksne modele, ki bi natančno opisovali proizvodno okolje, je težko uporabljati, saj je uporaba takih modelov pri reševanju problemov razvrščanja računsko zahtevna. Po drugi strani pa poenostavljeni modeli premalo natančno opisujejo realna dogajanja. Razvita so bila številna pravila razvrščanja, ki zadostujejo različnim kriterijem in zahtevam po učinkovitosti. Problematika razvrščanja je zelo raznolika, poleg tega pa se pogosto za podobne probleme uporabljajo tudi različni pristopi, specifični glede na znanstveno-raziskovalna področja, ki te probleme obravnavajo. Razvoj matematične teorije in teorije algoritmov razvrščanja, ki naj bi bila neodvisna od namena uporabe, je še najbolj opredeljen znotraj področja operacijskih raziskav (*Operations research*). Problem razvrščanja je obravnavan kot reševanje kombinatoričnega optimizacijskega problema. Teorija, ki obravnava deterministične modele, predpostavlja, da je potrebno razvrstiti končno število nalog, pri čemer mora minimizirati predoločen kriterij (*Objective*). Kriteriji, ki se uporabljajo, običajno upoštevajo podatke o nalogah, kot so čas izvajanja, čas sprostitve (*Release date*), rok izgotovitve (*Due date*) ali pa podatke o proizvodnih virih.

Da razvrščanje postane bolj dozorela disciplina, bi bilo potrebno več poudarka na modeliranju problemov in ne toliko na iskanju rešitve, ki je odvisna od specifične tehnike reševanja problema [1]. Tak pristop seveda ne spremeni kompleksnosti računanja, omogoča pa večjo svobodo pri izbiri metode.

Problemi razvrščanja, ki bodo omenjeni v nadaljevanju, bodo predstavljeni z

zapisom, povzetem po Pinedu [74]. Notacija zapisa modela je določena s tremi polji $\alpha|\beta|\gamma$, kjer α označuje strukturo proizvodnega sistema, β podaja dodatne zahteve in omejitve procesa in γ predstavlja kriterijsko funkcijo, katero je potrebno optimizirati. Pregled metod razvrščanja lahko najdemo tudi v [35], vendar v okviru tega dela izhajamo iz modela, tako bodo različne metode razvrščanja predstavljene glede na obseg in strukturo modela.

2.1 Kriteriji ovrednotenja razvrstitev

Za dan problem razvrščanja je možnih več različnih rešitev. Da ugotovimo, katera izmed njih je najbolj optimalna, jih je potrebno ovrednotiti. Razvrstitev je optimalna, če je vrednost kriterijske funkcije te razvrstitve minimalna v primerjavi z vsemi ostalimi izvedljivimi razvrstitvami. Kriteriji so v osnovi funkcije nalog in/ali virov. Te lastnosti so lahko konstantne ali pa tudi časovno spremenljive.

Najprej omenimo tip kriterijev, ki se nanašajo na čas trajanja izvajanja naloge znotraj proizvodnega procesa. Izvršni čas (*Makespan*) C_{max}, predstavlja skupen čas potreben za izvedbo vseh nalog v sistemu in je enak času zaključka zadnje operacije. Minimalni izvršni čas običajno pomeni tudi visok izkoristek proizvodnega sistema. Absoluten čas zaključka (*Total completiton time* ali *Total flowtime*) $\sum w_j C_j$, predstavlja vsoto zaključnih časov vseh nalog. Vsaki nalogi je lahko prirejen še faktor w_j, ki predstavlja pomembnost te naloge.

O izkoristku sistema govori tudi prepustnost sistema (*Throughput*), ki določa število nalog, ki se lahko izvršijo na sistemu v neki časovni enoti.

Drug tip kriterijev se nanaša na rok izgotovitve posamezne naloge. Zapoznelost (*Lateness*) L_{max}, ovrednoti zamujanje roka izgotovitve. Zapoznelost ima lahko tudi negativno vrednost, na primer, ko je naloga končana pred rokom. Kriterij kasnosti (*Tardiness*) T_{max}, je enak zapoznelosti, le da je vrednost kasnosti nič v primeru končanja pred rokom. Zgodnost (*Earliness*) E_{max}, označuje prehitevanje naloge, ki se zaključi pred rokom. Uporablja se v primeru, kadar stroški skladiščenja niso zanemarljivi.

V praksi so zahteve nekoliko kompleksnejše. Med sabo se običajno izključujejo in tako je potrebno najti kompromisno rešitev. Možna je kombinacija večih ele-

mentarnih kriterijev, ki so lahko tudi odvisni od časa. Obstoječe pristope lahko delimo na hkratne in hierarhične. Primer hkratnega pristopa bi bil tvorjenje kriterijske funkcije z linearno kombinacijo različnih elementarnih kriterijev. Pri hierarhičnih postopkih so kriteriji urejeni po pomembnosti. Najprej se optimizira prvi kriterij, v nadaljevanju se optimizira naslednje kriterije, kjer se išče optimalno rešitev izmed rešitev, določenih s predhodnim kriterijem.

2.2 Reševanje problema razvrščanja

Teorija razvrščanja sloni na modelu proizvodnega sistema. Model sistema pa je določen s strukturo proizvodnega sistema, izbiro kriterija ter dodatnih omejitev in zakonitosti v procesu. Za različne modele so bile razvite različne metode reševanja problema razvrščanja. Uporaba metod, ki zagotavljajo optimalno rešitev, je omejena na zelo okrnjene modele, ki zanemarijo vrsto zakonitosti. V nadaljevanju bodo predstavljeni različni modeli in od tega odvisni pristopi k reševanju. V prvi vrsti bodo pristopi deljeni glede na strukturo proizvodnega sistema. Za posamezne sestave so za različne kriterije predstavljene možnosti uporabe eksaktnih metod reševanja ter uporaba poenostavitev in aproksimativnih metod v primeru, ko analitično reševanje ni možno.

2.2.1 En stroj

Najprej so obravnavane rešitve razvrščanja na primeru modela enega stroja ($\alpha = 1$). Model enega stroja predstavlja najpreprostejše proizvodno okolje, ki je poseben primer vseh ostalih. Rezultati obravnave razvrščanja na enem stroju predstavljajo osnovo za razvoj hevrističnih pravil, ki se uporabljajo v kompleksnejših proizvodnih okoljih. V praksi so problemi razvrščanja v kompleksnih proizvodnih okoljih običajno razstavljeni na manjše podprobleme. Na primer, kompleksno proizvodno okolje z enim ozkim grlom lahko obravnavamo kot en stroj. V nadaljevanju bomo predstavili nekaj modelov enega stroja, ki ob gradnji razvrstitve upoštevajo različne kriterije.

Najprej obravnavajmo model s kriterijem, ki upošteva *skupni utežen čas zaključitve* ($1||\sum w_j C_j$). Čas trajanja naloge j je podan s p_j. Ta problem je opti-

malno rešljiv s pravilom, ki razporedi naloge tako, da so njihovi kvocienti w_j/p_j v padajočem vrstnem redu. Tako pravilo je v teoriji razvrščanja znano kot *najkrajši utežen čas izvajanja najprej* (*Weighted Shortest Processing Time first – WSPT*).

Za model enega stroja s kriterijem, ki poskuša minimizirati maksimalno zamujanje ($1||L_{max}$), optimalno razvrstitev zagotavlja algoritem, ki izvede najprej naloge z *najbolj zgodnim časom izgotovitve* (*Earliest Due Date first – EDD*). Ko imamo opravka s posplošenim problemom, kjer so naloge posredovane ob poljubnem času ($1|r_j|L_{max}$), postane problem določitve optimalne razvrstitve strogo NP-težak.

Ko imamo opravka z modelom enega stroja, kjer so nastavitveni časi odvisni od zaporedja ($1|s_{jk}|C_{max}$), je izvršni čas odvisen od razvrstitve. Reševanje takega problema je strogo NP-težko. V praksi so nastavitveni časi običajno vedno podani po kaki predoločeni strukturi, kar omogoča uporabo algoritmov, s katerimi je mogoče določiti optimalno rešitev v polinomskem času.

Problematika razvrščanja nalog na enem stroju, kjer je potrebno upoštevati kriterij kasnosti, $1||\sum T_i$, je podana v [92]. Problem velja za NP-težkega. Obstaja pa kar nekaj posebnih primerov, za katere obstajajo algoritmi, ki so rešljivi v polinomskem času. Tako se na primer pravilo *izvajanja najkrajšega časa najprej* (*Shortest Processing Time first – SPT*) izkaže za optimalnega, kadar imamo opravka z enakimi roki izgotovitve $d_i = d$.

V literaturi obstaja še več obravnav modelov enega stroja, ki upoštevajo različne kriterije (število kasnih nalog, izvršni čas z nastavitvenimi časi, odvisnimi od zaporedja,...). Vsi razviti algoritmi so uporabni le za zelo omejene primere in že manjša posplošitev problema rezultira v težko rešljive probleme.

Omenjene rešitve problemov se pogosto pojavljajo kot podproblemi v hevrističnih procedurah pri iskanju razvrstitve v kompleksnejših proizvodnih sistemih, npr. sestavu posamične obdelave (*Job shop*).

2.2.2 Paralelni stroji

S teoretičnega vidika so paralelni stroji ($\alpha = Pm$) posplošitev problema enega stroja. S praktičnega stališča pa je s takim modelom pogosto že mogoče opisati realno stanje. V nadaljevanju bomo obravnavali način reševanja danega problema

za različne kriterije.

Že reševanje problema *minimiziranja celotnega izvršnega časa* $Pm||C_{max}$ je NP-težko. Tako je bilo za ta primer razvitih kar nekaj hevrističnih pravil. Najenostavnejše je pravilo *izvajanja najdaljšega časa najprej* (*Longest Processing Time first – LPT*), ki ob času $t = 0$ naloži m najdaljših nalog na m strojev. V nadaljevanju se na prvi sproščen stroj naloži naslednja najdaljša še nerazvrščena naloga. V primeru, ko obstajajo tudi prednostne povezave, $Pm|prec|C_{max}$, postane iskanje rešitve še kompleksnejše. Ko obstaja več strojev, kot je nalog (neomejene kapacitete virov), je to klasičen problem s področja projektnega planiranja. Taki problemi se rešujejo z metodo kritične poti (CPM – *Critical Path Method*) ter tehnike ocenjevanja in pregledovanja projekta (PERT – *Program Evaluation and Review Technique*), [49]. Metodi zagotavljata optimalni razvrstitvi z enostavnim algoritmom, ki v začetku začne s sočasnim razvrščanjem nalog. Ko se prva izmed teh konča, nadaljuje z nalogami, katerim so bile vse predhodne naloge/operacije že končane. V primeru, ko imamo opravka z manjšim številom strojev, kot je nalog, je reševanje problema NP-težko. Uporaba omenjenih metod pri vodenju projektov bo prikazana v 6. poglavju.

Že v primeru enega stroja se izkaže pravilo SPT kot optimalno za minimiziranje kriterija *absolutnega časa zaključka*. Tudi ko imamo opravka z večjim številom vzporednih strojev, $Pm||\sum C_j$, velja pravilo SPT kot optimalno. Obstaja pa še več možnih načinov, kako doseči optimalno razvrstitev. Ko obravnavan sistem vsebuje tudi prednostne povezave, $Pm|prec|\sum C_j$, postane problem strogo NP-težak. Rešitev tega primera v polinomskem času je mogoča le za nekatere posebne poenostavljene primere.

Iskanje rešitve v primeru, ko se upošteva kriterij *roka izgotovitve*, velja za težko. Eden redkih sestavov, ki ga je moč rešiti v polinomskem času, je primer z dovoljenimi prekinitvami, $Pm|premp|L_{max}$, [74].

V literaturi se pojavljajo tudi kombinirani kriteriji, ki rešujejo problem vzporednih strojev. V [32] je predstavljen hierarhičen postopek za dva identična vzporedna stroja, kjer je najprej uporabljen algoritem, ki minimizira izvršni čas. V nadaljevanju se izbere tista izmed množice rešitev, ki minimizira absolutni čas zaključka. Problem je podan kot $P2||F_h(\sum C_i/C_{max})$. Podoben problem, $Pm||F_h(C_{max}/\sum C_i)$, je obravnavan v [33], kjer sta ista kriterija uporabljena

v obratnem vrstnem redu. Ta postopek zagotavlja razvrstitev z minimalnim izvršnim časom.

Ena izmed razširitev modela, ki upošteva več praktičnih omejitev, je problem, kjer je možno razvrščati eno nalogo na r strojih. Pogosto se pojavljajo problemi, ko se lahko naloga izvaja istočasno na večih strojih (r je pozitivno naravno število), ali pa ko se lahko več nalog izvaja na enem stroju ($0 < r \leq 1$, kjer je r realno število). Tako za klasičen primer razvrščanja, kjer lahko en stroj obdeluje le eno nalogo hkrati, velja $r = 1$. V [50] je podan izčrpen pregled omenjene problematike, ki se pojavlja v literaturi v zadnjih desetih letih.

2.2.3 Zaporedni stroji

Sestav tekoče obdelave (*Flow shop*) je sestavljen iz m zaporedno povezanih strojev z različno funkcionalnostjo ($\alpha = Fm$). Predvideva se, da so naloge sestavljene iz $n = m$ operacij in se izvajajo v istem zaporedju – imajo enako obdelovalno pot, ki vodi prek vseh m strojev. Posameznimi stroji so med sabo ločeni z vmesnimi skladišči (*Buffer*). Na vmesnem skladišču se nahajajo naloge, ki so bile končane na enem stroju in čakajo na izvajanje na naslednjem stroju. Za primer dveh strojev in vmesnega skladišča z neomejeno kapaciteto ($F2||C_{max}$), je bilo razvito pravilo, ki omogoča razvrstitev z minimalnim izvršnim časom, t.j. Jonhsonovo pravilo [41]. Naj bo čas izvajanja naloge j na prvem stroju a_j in na drugem b_j. Tako v primeru, ko velja $min\{a_i, b_j\} \leqslant min\{a_j, b_i\}$, Johnsonovo pravilo postavi nalogo i pred nalogo j. Johnsonovno pravilo se lahko razširi tudi na problem $F3||C_{max}$ in $F4||C_{max}$, vendar je v tem primeru potrebno zadostiti dodatnim pogojem glede razmerij trajanja operacij, kar njegovo praktično uporabnost močno zmanjša.

Običajno imajo skladišča med posameznimi proizvodnimi koraki omejeno kapaciteto. Upoštevati je potrebno, da se skladišče lahko napolni, kar povzroči blokiranje stroja na vhodni strani skladišča. V [74] je obravnavan kriterij izvršnega časa, kjer predvideva le vmesna skladišča s kapaciteto nič. Vmesna skladišča s končno pozitivno kapaciteto se modelira kot stroj, katerega procesni časi so za vse naloge enaki nič. Izvršni čas se lahko izračuna z določitvijo kritične poti v usmerjenemu grafu, ki je podrobneje predstavljen v 3. poglavju.

Aldenov model [13] analizira proizvodno linijo, ki je sestavljena iz dveh zaporednih postaj (strojev) in sta med sabo ločena z vmesnim skladiščem. Obe postaji sta podvrženi okvaram, ki lahko vplivajo na prepustnost proizvodne linije. Vsaka postaja je okarakterizirana s hitrostjo (*Processing rate*), velikostjo vmesnega pomnilnika in s parametri zanesljivosti, kot so stopnja okvare in stopnja popravljanja. Prepustnost linije, predstavljene z Aldenovim modelom, je opisana z analitično formulo, pri čemer je upoštevana vrsta predpostavk. V nadaljevanju obravnava primer z M postajami. Analiza takega sistema bi postala preveč kompliciran, zato podaja aproksimativno metodo, ki je osnovana na izsledkih obravnave dveh strojev.

Bolj splošna konfiguracija bi bila predstavljena z več zaporednimi stopnjami, kjer lahko vsako stopnjo sestavlja več vzporednih strojev (*Flexible flow shops – FFs*). Zaporedje izvajanja operacij je enako za vsako nalogo, medtem ko je obdelovalna pot odvisna od izbire strojev na posamezni stopnji. Kadar imamo opravka s konfiguracijo, kjer je na vsaki naslednji stopnji vsaj toliko strojev, kot jih je na predhodni stopnji, in ko so si vsi procesni časi naloge enaki na vseh strojih, velja, da je pravilo SPT optimalno za kriterij absolutnega časa zaključka [74].

Hevristični postopek za razvrščanje fleksibilnega sestava tekoče obdelave, ki minimizira izvršni čas, je predstavljen v [73]. Pristop razporedi vsako nalogo posebej skozi vse stopnje in nato začne z razvrščanjem naslednje naloge. Pred vsakim razporejanjem posamezne naloge se ugotovi ozko grlo sistema. Nato algoritem izbere tako nalogo, ki minimalno poveča čakalni čas na stopnji, kjer nastopa ozko grlo.

2.2.4 Sestav posamične obdelave (*Job shop*)

V primeru, ko je proizvodna pot (*Route*) naloge fiksna, vendar različna za posamezne naloge, govorimo o sestavu posamične obdelave (*Job shop* – $\alpha = Jm$). Klasičen problem razvrščanja posamične obdelave je podan z množico n nalog $J = \{J_j\}$, $j = 1, ..., n$, ki se morajo izvršiti na končnem številu m strojev $M = \{M_i\}$, $i = 1, ..., m$. Vsaka naloga je sestavljena iz končnega števila n_j operacij $O_j = \{o_{jk}\}$, $k = 1, ..., n_j$, kjer $o_{jk} \prec o_{jk+1}$ določa, da se lahko operacija

o_{jk+1} začne po zaključku operacije o_{jk}. Posamezna operacija se lahko izvaja le na predpisanem stroju, za kar potrebuje predpisan čas izvajanja p_{jk}. Na področju uporabe različnih pristopov k reševanju problema razvrščanja nad sestavom posamične obdelave obstaja ogromno literature. Nekaj preglednih prispevkov lahko najdemo v [11, 42, 74].

Nekoliko splošnejši primer bi bil sestav odprte obdelave (*Open shop*), kjer proizvodne poti niso fiksne ($\alpha = Om$). To pomeni, da zaporedje operacij ni vnaprej predpisano, ampak ga določi proces razvrščanja. Enake naloge imajo lahko različna zaporedja in različne proizvodne poti.

Razvitih je bilo nekaj algoritmov, ki zagotavljajo optimalno rešitev, vendar so namenjeni in uporabni le za omejene probleme razvrščanja. Kadar imamo opravka s sestavom posamične obdelave z dvema strojema in n nalogami ter želimo minimizirati izvršni čas, $J2||C_{max}$, lahko ta problem reduciramo v problem $F2||C_{max}$. Tu z $J_{1,2}$ označimo niz nalog, ki se morajo izvršiti na stroju M_1 in z $J_{2,1}$ naloge, ki se morajo izvršiti na stroju M_2. Velja, da imajo naloge iz $J_{1,2}$ na stroju M_2 večjo prioriteto kot naloge iz niza $J_{2,1}$ in obratno. V kakšnem zaporedju se bodo izvajale naloge iz $J_{1,2}$ na stroju M_1, lahko določimo z rešitvijo problema razvrščanja niza nalog $J_{1,2}$, kot problem $F2||C_{max}$, kjer je najprej uporabljen stroj M_1. Enako se določi zaporedje izvajanja nalog $J_{2,1}$. Predstavljen algoritem je eden redkih, ki je rešljiv v polinomskem času, za problem sestava posamične obdelave.

Za problem, ko imamo sestav odprte obdelave, ki vključuje dva stroja in n nalog ter želimo minimizirati izvršni čas, $O2||C_{max}$, se izkaže pravilo LAPT (*Longest Alternate Processing Time first*) za optimalnega [74]. Kadarkoli je stroj sproščen, pravilo LAPT izbere nalogo, ki čaka na izvajanje in ima najdaljši čas izvajanja ter jo naloži na drug stroj. To pravilo je eno izmed redkih, ki je rešljivo v polinomskem času, za sestave odprte obdelave. Za probleme, ko imamo opravka z večjim številom strojev, $Om||C_{max}$, obstajajo le sub-optimalne rešitve, ki so rešljive v polinomskem času.

Eksaktne metode temeljijo na upoštevanju celotnega prostora rešitev in tako zagotavljajo optimalnost dobljene rešitve. Zaradi počasne konvergence in potreb po veliki računalniški kapaciteti, so metode uporabne le za omejene in poenostavljene probleme.

Najpreprostejša metoda razvrščanja je *polno pregledovanje* (*Exhaustive enumeration*) celotnega prostora rešitev. Za vse možne rešitve se izračuna vrednost kriterijske funkcije. Optimalna je tista rešitev, ki ima najboljšo vrednost kriterijske funkcije. Metoda je relativno enostavna, toda ker število rešitev v odvisnosti od velikosti problema hitro raste, je metoda primerna le za probleme majhnih dimenzij.

Tehnika razveji in omeji (*Branch and bound*) z uporabo analize dosegljivosti najprej razveji problem (problemsko drevo) na več podproblemov, medtem ko z odstranjevanjem ne optimalnih poti problematiko omejuje. S temi algoritmi se računsko potratnost nekoliko zmanjša, vseeno pa je čas potreben za izračun običajno prevelik za uporabo v realnih okoljih. Tako se v praktičnih primerih metoda razvejaj in omeji večinoma uporablja v kombinaciji z aproksimativnimi postopki.

Uporaba *metod matematičnega programiranja* je v kompleksnih sistemih zelo omejena. Matematični modeli, ki jih je moč rešiti s takimi metodami, morajo običajno zanemariti veliko praktičnih omejitev. Matematično programiranje je splošna matematična metoda za reševanje statičnih optimizacijskih problemov, ki so podani s kriterijsko funkcijo $F(x)$, ki jo je potrebno minimizirati. $F(x)$ je skalarna funkcija večih spremenljivk x, glede na niz omejitev, podanih v obliki enačb in neenačb. Matematično programiranje, ki se pogosto pojavlja v teoriji razporejanja, je linearno programiranje (LP), kjer so vse nastopajoče funkcije linearne. V primeru, ko imamo opravka le z celoštevilčno spremenljivko x, govorimo o celoštevilčnem linearnem programiranju (ILP).

Večino problemov razvrščanja je tako kompleksnih, da jih ni mogoče enostavno formulirati v matematični obliki. Dejstvo, da je take probleme težko formulirati, otežuje uporabo eksaktnih metod razvrščanja. Tako so se pojavile številne tehnike, ki sicer ne garantirajo optimalne rešitve, omogočajo pa določitev sprejemljive rešitve v relativno kratkem času. V nadaljevanju bo omenjenih le nekaj reprezentativnih aproksimativnih metod, ki se pojavljajo najpogosteje.

Prioritetna pravila (priority rules) se zaradi enostavne implementacije in majhne računske zahtevnosti v praksi uporablja največ. Uporabo omenjenih pravil ponazorimo na naslednjem primeru. Predpostavimo, da ob času t obstaja množica še nerazvrščenih operacij $O(t)$. Algoritem priredi stroju eno izmed ope-

Tabela 2.1: Tabela nekaterih reprezentativnih pravil razvrščanja

Pravilo	Opis
SPT	Naloga z najkrajšim proizvodnim časom (*Shortest Processing Time*)
WSPT	Uteženi najkrajši proizvodni čas (*Weighted Shortest Processing Time*)
LPT	Naloga z najdaljšim proizvodnim časom (*Longest Processing Time*)
RND	Naključna izbira (*Random*)
EDD	Najzgodnejši rok izgotovitve (*Earliest Due Date*)

racij, ki se lahko začne izvajati. V primeru konflikta, ko se več operacij poteguje za isti stroj, je potrebno narediti odločitev. Pri tej odločitvi pa lahko uporabimo razna pravila, ki so predstavljena na primer v [10, 72]. S prioritetnimi pravili se v splošnem vsakemu opravilu dodeli prioriteta, ki določa vrstni red, po katerem jih algoritem razvrsti. Obstajajo pa tudi pravila, ki so vezana na stroje. Z njimi se odloča, katera operacija iz vrste čakajočih operacij, se bo na danem stroju izvajala. Nekaj osnovnih pravil, ki se pogosto pojavljajo v praksi, je naštetih v tabeli 2.1. Kot že omenjeno, so potrebe v praksi običajno podane s kompleksnimi kriterijskimi funkcijami, ki so sestavljene iz več elementarnih pravil.

Ena od aproksimativnih metod, ki se v praksi izkaže za dokaj učinkovito pri reševanju sestava posamične obdelave, je *metoda premikanja ozkega grla (Shifting bottleneck heuristic)*. Metoda temelji na predpostavki, da ozko grlo predstavlja le en stroj. Problem $J_M||C_{max}$ se razstavi na m podproblemov $1|r_j|L_{max}$ in se reši vsakega izmed njih. Na podlagi dobljenih rešitev se določi stroj, ki predstavlja ozko grlo. Na njem se fiksira zaporedje izvajanja operacij. Ostale stroje se ob upoštevanju fiksiranega stroja zopet reši kot $(m-1)$ problemov $1|r_j|L_{max}$. Postopek se nadaljuje, dokler niso vsi stroji fiksirani. V praksi pogosto nastopajo sistemi, katerih delovanje je odvisno od nastavitvenih časov. V tem primeru učinkovito rešitev predstavlja minimiziranje ozkega grla, izvedeno z minimiziranjem nastavitvenih časov [49]. Le-to pa je izvedeno z različnimi tehnikami razvrščanja, predstavljenimi že na enostavnejših proizvodnih strukturah.

S pojavom hitre in relativno poceni računalniške moči, so se pojavile nove tehnike razvrščanja. Enostavno je mogoče implementirati tehniko iskanja (*Se-*

arch technique). Ideja temelji na iterativnem izboljševanju začetne rešitve, kjer se v vsakem koraku optimizacije na razvrstitvi izvede določena sprememba izmed vnaprej definirane množice sprememb, ki ji pravimo okolica razvrstitve. Ko z nobeno spremembo v okolici ne moremo več izboljšati kriterijske funkcija, smo dosegli lokalni minimum. Avtor v [70] predlaga metahevristični pristop, ki dovoljuje spreminjanje razvrstitve tudi na način, ki njeno kvaliteto poslabša. S tem je omogočen pobeg iz lokalnega minimuma in nadaljnje iskanje boljših rešitev.

Tabu iskanje (*Tabu search*) je matematična optimizacijska metoda, ki pripada razredu tehnik lokalnega iskanja. Osnovna ideja tabu iskanja je v preiskovanju izvedljivih razvrstitev z nizom premikov. Premiki niso naključni, ampak so izbrani deterministično. V ta namen se uporablja tabu seznam, v katerega se uvrstijo premiki, ki so prepovedani. Optimizacija poteka tako, da se v vsaki iteraciji izvede tista sprememba, ki na kriterijsko funkcijo vpliva najbolje in hkrati ni na tabu seznamu. Po vsaki izvedeni spremembi se na tabu seznam uvrsti njeno inverzno spremembo, s čimer le-ta postane prepovedana v nadaljnjih iteracijah. Pregled uporabe metod tabu iskanja pri reševanju problematike razvrščanja lahko najdemo v [27].

Probleme razvrščanja je mogoče reševati tudi z genetskimi algoritmi (*Genetic algorithms*). Genetski algoritmi izhajajo iz genetike. Lastnosti organizmov so zapisane v genih, ki so nosilci informacij. Če želimo spremeniti lastnost organizma, na osnovi določenih pravil spremenimo gene. Na teh spoznanjih temelji teorija genetskih algoritmov, ki je pravzaprav prevedba opisanega dogajanja v računalniško okolje. Na ta način pridobljeni algoritmi so primerni za reševanje najrazličnejših optimizacijskih problemov. Uporaba omenjenih algoritmov za namene razvrščanja je predstavljena v [22, 55].

Tudi teorija mehke logike (*Fuzzy logic*) je bila uporabljena pri razvoju hibridnih pristopov k razvrščanju. Teorija mehke logike je lahko uporabna pri modeliranju in reševanju problemov razvrščanja, kjer nastopajo negotovi proizvodni časi, omejitve ali pa nastavitveni časi. Te nezanesljivosti so lahko predstavljene s pripadnostno funkcijo. Ti pristopi so običajno integrirani skupaj z kako drugo metodologijo, kot so na primer genetski algoritmi [24].

Umetne nevronske mreže (*Artificial neural networks*) predstavljajo poizkus prenosa principa delovanja nevronov v tehniko. Informacija se v nevronskih

mrežah procesira preko velikega števila preprostih procesnih elementov (nevronov), ki so med seboj povezani v mreže. Pravo uporabnost dobijo nevronske mreže z učenjem, kar pomeni, da so se sposobne prilagoditi vhodnim podatkom in podanim zahtevam. Pregled uporabe nevronskih mrež na področju razvrščanja lahko najdemo v [39, 84]. V teoriji razvrščanja se pojavlja več tipov nevronskih mrež. V [84] je podrobno predstavljena izvedena verzija Hopfieldovega modela nevronske mreže. Vsak stroj je predstavljen z $n \times n$ matriko, ki predstavlja nivo. Vsaka vrstica v matriki predstavlja operacijo posamezne naloge in vsak stolpec določa pozicijo te operacije v zaporedju, določenem za ta vir. Slabost takega modela je, da potrebuje veliko število nevronov in povezav. Pojavljajo pa se tudi hibridni pristopi, ki razna hevristična pravila kombinirajo skupaj z nevronskimi mrežami [100].

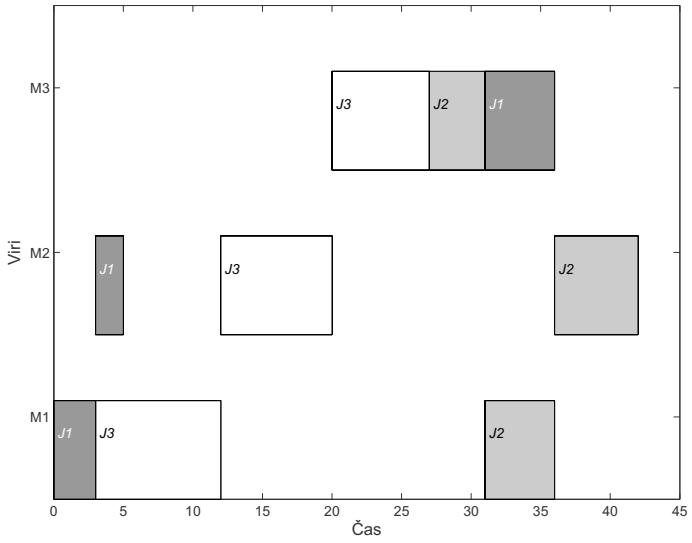
2.3 Prikaz razvrstitve

Za grafično predstavitev planov, razvrstitev in nadzor poteka dela, planov se v veliki meri uporablja gantogram (*Gantt Chart*). Ta način predstavitve je v l. 1919 predstavil Henry L. Gantt [26]. V proizvodnem razvrščanju se gantogram uporablja za prikaz razvrstitve operacij po strojih. Iz gantograma je mogoče oceniti izvršni čas, razpoložljivost virov, itd. Tako predstavlja nepogrešljiv pripomoček za razvrščanje.

Gantogram se gradi s podatki o stanju posameznih operacij. Vsaka operacija je podana s podatki v treh dimenzijah. Prva podaja informacijo o času, t.j. začetek in konec posamezne operacije. Z drugim podatkom so podani stroji, ki so potrebni za izvedbo posamezne operacije. In tretja dimenzija predstavlja nalogo, v kateri se operacija nahaja.

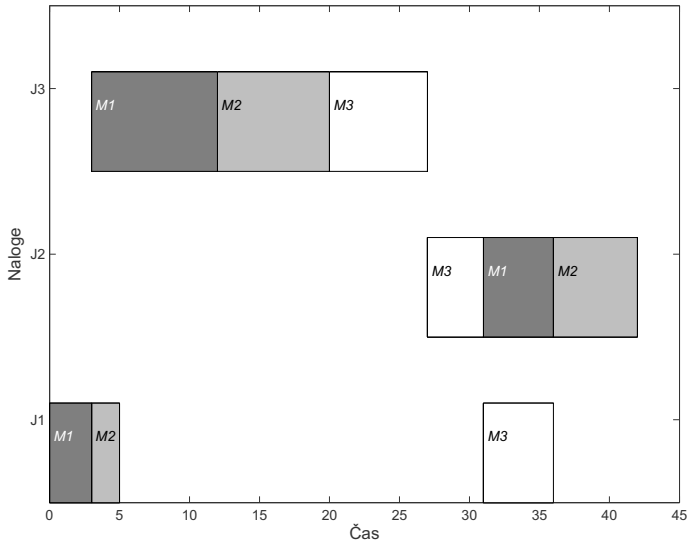
Operacije so v gantogramu predstavljene s pravokotniki. Trajanje operacije je predstavljeno z dolžino tega pravokotnika. Uporabljata se dve obliki gantograma. Prva se uporablja za prikaz statusa proizvodnih virov skozi čas. Tako so na ordinatno os naneseni viri, medtem ko abscisna os predstavlja čas. Primer takega gantograma je prikazan na sliki 2.1. V danem primeru je vsaka izmed operacij tudi označena, kateri nalogi pripada.

Druga oblika gantograma se uporablja za nadzor napredovanja dela. Naloge,



Slika 2.1: Gantogram – status strojev

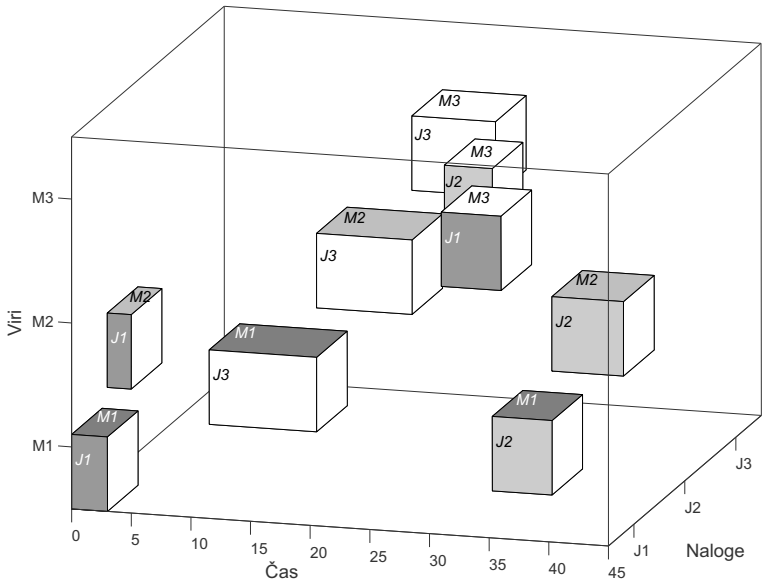
ki jih je potrebno izvršiti, so nanesene na ordinatno os. Gantogram, kot je prikazan na sliki 2.2, grafično prikazuje, kateri stroj uporablja naloga ob določenem trenutku.



Slika 2.2: Gantogram – spremljanje poteka dela

Iz gantograma je jasno razvidno, kdaj in s katero operacijo je posamezni stroj zaseden. Le-to je v pomoč operaterju, ki mu je na ta način predstavljena rešitev. Poleg tega pa se lahko iz gantograma kaj hitro opazi neoptimalne odseke v razvrstitvi, npr. ob okvari katerega izmed strojev.

Pojavilo se je tudi nekaj poskusov izboljšanja gantograma. Eden izmed njih je tridimenzionalni gantogram [43]. Tri dimenzije predstavljajo čas, naloge in stroje. Tu je operacija predstavljena s kvadrom, katerega dolžina je zopet odvisna od trajanje operacije. Kvader, ki prestavlja operacijo, je postavljen v prostor, glede na časovno obdobje, v katerem se izvaja, glede na vir, ki ga uporablja za svoje izvajanje, in glede na nalogo, kateri operacija pripada. Na sliki 2.3 je predstavljen primer predstavljenega gantograma. Z rotacijo gantograma preko časovne osi, lahko dobimo različne projekcije tridimenzionalnega gantograma. Med njimi standardna dvodimenzionalna gantograma in pa različne kombinacije obeh.



Slika 2.3: Poševna projekcija tridimenzionalnega gantograma

Razvrstitev je lahko prikazana tudi z acikličnim grafom, ki je rezultat problemov razvrščanja, ki so predstavljeni z disjunktivnim grafom. Disjunktivni graf je direktno integriran v področje diskretne matematike in je primeren za opis računalniških algoritmov. Omenjena ponazoritev je podrobneje predstavljena v naslednjem poglavju.

3. Modeliranje problemov razvrščanja v proizvodnji

Pri obravnavi sodobnih proizvodnih sistemov se pojavljajo raznoliki problemi preko njihovega celotnega življenjskega cikla. Že v fazi načrtovanja je pomembno vedeti čimveč o sistemu, ki ga želimo. Odločitve sprejete v tej fazi so običajno zelo pomembne, saj so stroški sprememb v kasnejših fazah vedno večji. Tudi v fazi obratovanja, kjer je potrebno obvladovati nešteto podatkov, je potrebno sprejemati ogromno odločitev.

Modeliranje in simulacija sta dva neločljiva postopka, katera vsebujeta kompleksne aktivnosti v zvezi s konstrukcijo modelov, ki predstavljajo realne objekte, in eksperimentiranje z modeli v smislu pridobivanja podatkov o obnašanju modeliranega procesa. Na proizvodni sistem lahko gledamo kot na diskretno dogodkovni sistem (*Discrete Event System – DES*). Obnašanje takih sistemov je običajno zelo kompleksno. S pomočjo formalnih metod je razumevanje takih sistemov lažje. Omogočajo nam njihovo analizo in so nam v pomoč pri njihovem načrtovanju. Poleg tega poenoten formalni zapis modela omogoča komunikacijo med vpletenimi osebami, ki opravljajo dela na različnih stopnjah delovanja proizvodnje. Primarna naloga simulacije diskretnih dogodkov vključuje podporo pri odločanju pri delu z diskretno dogodkovnimi modeli.

Model je matematična predstavitev pomembnih lastnosti sistema. Na področju sistemov diskretnih dogodkov ni univerzalnega in splošno sprejetega pristopa k modeliranju. Konvencionalna modelerska orodja, kot so zvezne diferencialne in diferenčne enačbe ali neenačbe ne zadoščajo za opis takih sistemov. Ti sistemi so lahko asinhroni, sestavljeni iz več konkurenčnih si komponent ali podsistemov, med njimi pa obstajajo kompleksne povezave. V ta namen je bilo razvitih več različnih matematičnih formalizmov.

V tem poglavju so predstavljena orodja, ki omogočajo poenoteno obravnavo proizvodnih sistemov tako v fazi načrtovanja kot tudi v fazi obratovanja. Za obravnavo teh sistemov v fazi obratovanja je pomembno, da je možna tudi gradnja časovnega modela. Vsem predstavljenim modelom je tudi skupno, da omogočajo grafično predstavitev in enostavno izvajanje simulacije nad modelom. Take modele je mogoče uporabiti za reševanje problemov razvrščanja, saj jih je mogoče enostavno uporabljati skupaj z različnimi algoritmi oziroma metodami razvrščanja. Osnovna ideja je v uporabi metod razvrščanja pri odločanju ob konfliktnih situacijah, ki jih model zazna.

3.1 Matematična orodja

Za določitev modela proizvodnega sistema, ki služi za razvrščanje dela, v splošnem potrebujemo vhodne podatke, ki so spisek operacij, ki sestavljajo posamezne naloge, čas izvajanja posamezne operacije, spisek prednostnih povezav, ki nastopajo med posameznimi operacijami ter spisek vseh potrebnih virov za vsako nalogo in operacijo. V nadaljevanju bo predstavljenih nekaj izmed formalizmov, ki se pogosto uporabljajo za matematično predstavitev problemov razvrščanja.

3.1.1 Disjunktivni graf (*Disjunctive graph model*)

Disjunktivni graf [83] je eden izmed načinov predstavitve problema razvrščanja. To je usmerjen graf $G = (V, C \cup D)$, kjer je:

- V množica vozlišč, ki predstavljajo operacije o_{jk} potrebne za izvedbo nalog. Niz vsebuje dve dodatni vozlišči za fiktivni operaciji z ničelnim časom izvajanja – izvor in ponor, ki predstavljata začetek (I) in konec (f) razvrstitve.
- C množica konjuktivnih povezav, s katerimi lahko povzamemo tehnološke omejitve. Smeri povezav določajo zaporedja izvajanja operacij znotraj posamezne naloge. Vsaki povezavi je dodana utež, ki določa čas izvajanja operacije, iz katere povezava izhaja.
- D množica disjunktivnih povezav, ki so dvosmerne. Med seboj povezujejo tiste operacije, ki potrebujejo za izvajanje isti stroj.

Kot primer obravnavajmo problem razvrščanja sestava posamične obdelave, kjer nastopajo trije stroji $M = M_1, M_2, M_3$ in tri naloge $J = J_1, J_2, J_3$. Vsaka naloga je sestavljena iz treh operacij. Potrebe nalog so predstavljene v tabeli 3.1. V tabeli 3.2 pa so definirani časi posameznih operacij, ki nastopajo v nalogah.

Tabela 3.1: Potrebe nalog

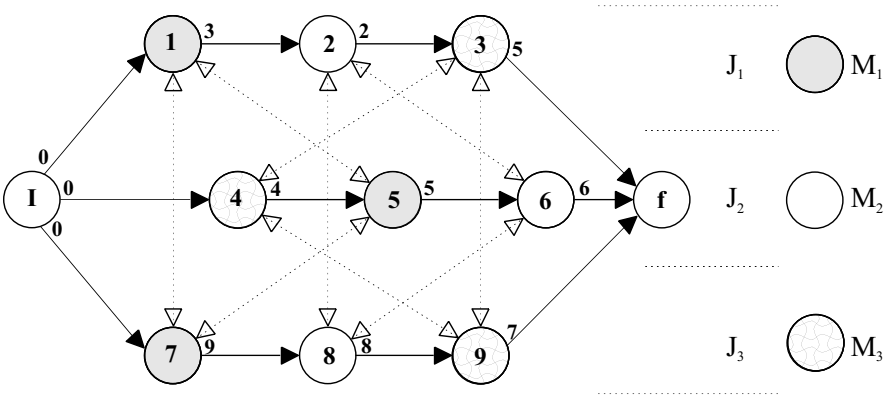
Proces	J_1	J_2	J_3
1	M_1	M_3	M_1
2	M_2	M_1	M_2
3	M_3	M_2	M_3

Tabela 3.2: Časi trajanja operacij

Operacija	o_{11}	o_{12}	o_{13}	o_{21}	o_{22}	o_{23}	o_{31}	o_{32}	o_{33}
Čas	3	2	5	4	5	6	9	8	7

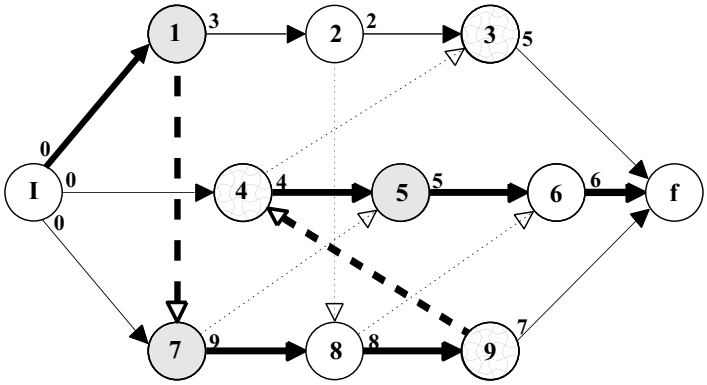
Problem je predstavljen s disjunktivnim grafom na sliki 3.1. Tako v grafu nastopa devet operacij, predstavljene so z vozlišči V. Poleg vozlišča je podan čas trajanja posamezne operacije. Množica C vsebuje konjuktivne povezave (prednostne omejitve), ki povezujejo operacije, ki pripadajo isti nalogi, npr. operacije označene z 1, 2 in 3, sestavljajo nalogo J_1. Množica D vsebuje disjunktivne povezave, ki povezujejo operacije, ki se izvajajo na istih strojih (omejitve končnih virov), npr. operacije označene z 1, 5 in 7 se izvajajo na stroju M_1.

Običajen problem razvrščanja sestava posamične obdelave je iskanje optimalne razporeditve opravil po strojih, v smislu iskanja razvrstitve z minimalnim izvršnim časom. Na posameznem stroju se lahko izvaja le po ena operacija. Določiti izvedljivo razvrstitev v diskjunktivnem grafu, pomeni izločanje vsaj ene povezave iz para disjunktivnih povezav. Dobimo usmerjeni aciklični graf, ki določa zaporedje izvajanja vseh konfliktnih operacij, ki za izvajanje potrebujejo iste stroje. Iz tega lahko določimo začetne čase vseh operacij. Izvršni čas določimo z izračunom kritične (najdaljše) poti, kar pa je lahko pri večjih sistemih računsko zahtevno [20]. Slika 3.2 predstavlja razvrstitev za primer dan



Slika 3.1: Problem razvrščanja predstavljen z disjunktivnim grafom

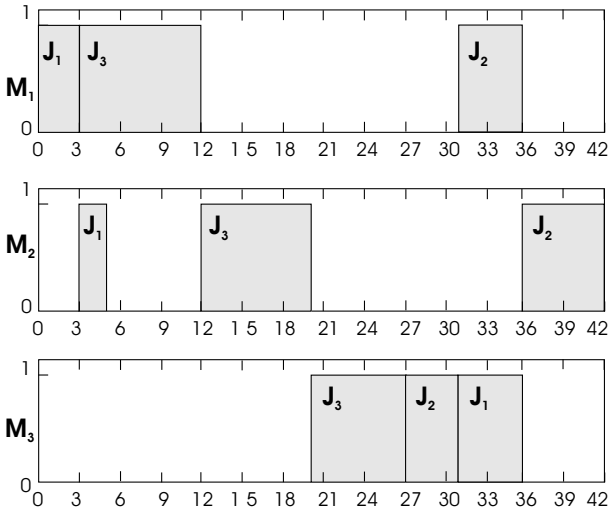
na sliki 3.1, kjer je kritična pot ilustrirana s povezavami z odebeljenimi črtami. Problem razvrščanja po kriteriju najkrajšega izvršnega časa se tako reducira na iskanje tistih disjunktivnih povezav, ki zagotavljajo, da bo najdaljša pot od izvora do ponora minimalna [11]. Ena izmed metod reševanja problema razvrščanja, ki se v praksi pogosto uporablja, je metoda premikanja ozkega grla.



Slika 3.2: Razvrstitev

Razvrstitev, ki je podana z acikličnim grafom na sliki 3.2, lahko predstavimo tudi s preglednejšim gantogramom, ki je prikazan na sliki 3.3.

Obstaja vrsta matematično programirljivih formulacij za reševanje problema razvrščanja sestava posamične obdelave. Pogosto se uporablja oblika *disjunktivnega programiranja*, ki se močno navezuje na disjunktivni graf. Tu se uporablja p_{jk} za označitev časa izvajanja naloge j na stroju k, t_{jk} pa določa začetni čas te operacije. Množica vseh operacij o_{jk} ustreza množici V v disjunktivnem grafu. Naslednji matematični program minimizira izvršni čas C_{max}, pri čemer



Slika 3.3: Razvrstitev prikazana z gantogramom

mora upoštevati naslednje omejitve:

$$C_{max} \geq t_{jk} + p_{jk} \qquad \forall o_{jk} \in V \tag{3.1}$$

$$t_{jk} \geq t_{jl} + p_{jl} \qquad \forall o_{jl} \prec o_{jk} \in C \tag{3.2}$$

$$t_{jk} \geq t_{il} + p_{il} \text{ ali } t_{il} \geq t_{jk} + p_{jk} \qquad \forall o_{jk} \text{ in } o_{il} \in V, i = 1...m \tag{3.3}$$

$$t_{jk} \geq 0 \qquad \forall o_{jk} \in V \tag{3.4}$$

Druga 3.2 in tretja 3.3 omejitev ustrezata množici konjuktivnih C in disjunktivnih D povezav v disjunktivnem grafu.

Dejstvo, da je problem razvrščanja moč formulirati kot disjunktivno programiranje, še ne pomeni, da obstaja standarden postopek za iskanje rešitve. Tako se uporabljajo postopki reševanja na osnovi hevrističnih metod. Pogosto se uporablja tehnika razvejaj in omeji [74] ali tehnika lokalnega iskanja.

Matrika grafa (*Graph matrix*) je podatkovna struktura, ki v matrični obliki predstavi disjunktivni graf [12]. Matrika grafa $G_{[(n+2)\times(n+2)]}$ predstavlja disjunktivni graf, pri čemer vrednost elementa g_{ij} določa medsebojno zvezo med i-to in j-to operacijo. Ker je na začetku (koncu) disjunktivnega grafa dodana fiktivna operacija, ki nima nobenega predhodnika (naslednika), in ker ne obstajajo lastne zanke, so lahko elementi g_{i0}, g_{0i}, $g_{i(n+1)}$, $g_{(n+1)i}$ in g_{ii} uporabljeni za shranjevanje dodatnih informacij. Iz matrike grafa je tako moč razbrati spisek operacij,

ki so predhodniki neki operaciji, spisek naslednjih in spisek nerazvrščenih operacij. Tako predstavitev proizvodnega sistema je mogoče ustvariti dokaj enostavno, enostavno pa je tudi osveževanje podatkov med iskanjem rešitve. Kompaktna oblika predstavitve podatkov o proizvodnem sistemu je še posebej primerna za pristope razvrščanja, ki med iskanjem rešitve shranjujejo vmesne podatke, kot je na primer metoda razvejaj in omeji.

3.1.2 Časovni avtomati (*Timed automata*)

Pogosto uporabljeno orodje za modeliranje sistemov diskretnih dogodkov so modeli avtomatov [66]. Z avtomati, ki ne upoštevajo časa, lahko določamo le zaporedje stanj. Taki modeli so primerni za kvalitativen ali logičen opis sistema. Iz modela lahko na primer razberemo, ali je neko stanje dosegljivo iz začetnega stanja, ali sistem lahko zaide v mrtvi tek ipd. Pri časovnih avtomatih pa upoštevamo tako zaporedje, kot tudi trajanje stanj oziroma čas nastopa dogodkov [3]. Omogočajo kvantitativni opis sistema in so primerni za določanje učinkovitosti in zmogljivosti sistema. Tako se v zadnjem obdobju časovni avtomat pogosto pojavlja kot orodje za modeliranje problemov, ki nastajajo ob razvrščanju [1].

Časovni avtomat je avtomat, kateremu je dodana časovna spremenljivka, ki uniformno raste z vsakim naslednjim stanjem. S takimi avtomati lahko opišemo vedenje časovno-odvisnih sistemov. Omogočajo gradnjo modela distribuiranega sistema, ki je sestavljen iz manjših podsistemov, na naraven način. Model zgrajen s časovnim avtomatom je izvršljiv in tako omogoča nadzor nad izvajanjem modela.

Obstaja več različic definicije časovnega avtomata. V primeru uporabe časovnega avtomata v namene razvrščanja [1], je časovna informacija podana nekoliko manj splošno, kot v standardni definiciji časovnih avtomatov. Tako je časovni avtomat definiran kot $A = (Q, C, I, \Delta, s, f)$, kjer je:

- Q množica stanj avtomata,
- C množica časovnikov (*Clocks*),
- I vztrajnostni pogoj (*Staying condition*), ki vsakemu stanju $q \in Q$ priredi

I_q v obliki $c \leq u$ za nek časovnik c in neko naravno število u,

- Δ funkcija prehajanja stanj,
- s in f sta začetno in končno stanje.

Izračun časovnika je določen s funkcijo $\mathbf{v} : C \rightarrow \mathbb{R}_+ \cup \{0\}$. Konfiguracija avtomata je tako podana s parom $(q, \mathbf{v})$, ki je sestavljen iz diskretnih stanj in izračunov časovnika. Obstaja neka podmnožica časovnikov $\rho \subseteq C$, ki proži funkcijo $Reset_\rho$. Ta funkcija je definirana za vsak izračun časovnika $\mathbf{v}$ in za vsak časovnik $c \in C$:

$$Reset_\rho = \begin{cases} 0 & if \ \ c \in \rho \\ \mathbf{v}(c) & if \ \ c \notin \rho \end{cases} \tag{3.5}$$

Funkcija $Reset_\rho$ resetira vse časovnike, ki se nahajajo v množici ρ, na nič, medtem ko ostanejo ostali nespremenjeni.

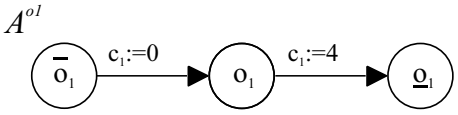
Časovni korak izvajanja avtomata je definiran kot $(q, \mathbf{v}) \rightarrow^t (q, \mathbf{v} + t \cdot \mathbf{1})$, $t \in \mathbb{R}_+$. Tako je izvajanje avtomata iz začetnega v n-to stanje predstavljeno kot končni niz korakov:

$\xi : (q_0, \mathbf{v}_0) \rightarrow^{t1} (q_1, \mathbf{v}_1) \rightarrow^{t2} ... \rightarrow^{tn} (q_n, \mathbf{v}_n)$

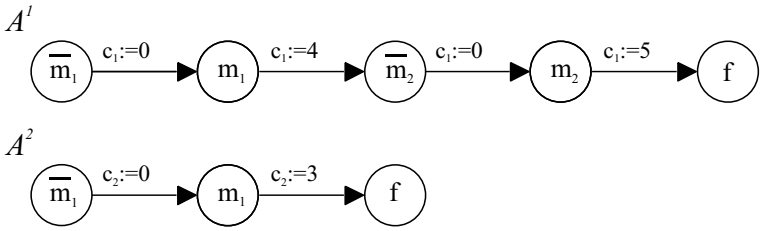
Časovne avtomate je mogoče uporabiti za modeliranje proizvodnih sistemov za namene razvrščanja [1]. Za vsako operacijo o je zgrajen avtomat z enim časovnikom c in tremi stanji $Q = \{\overline{o}, o, \underline{o}\}$, kjer $\overline{o}$ opredeljuje stanje čakanja na začetek operacije, o je aktivno stanje in $\underline{o}$ stanje, ko je operacija končana. Prehod iz stanja $\overline{o}$ v stanje o resetira časovnik na nič. Omenjeni prehod se lahko zgodi le v primeru, ko so avtomati, ki se nanašajo na predhodne operacije, v končnem stanju. Prehod iz stanja o v $\underline{o}$ pa se zgodi, ko velja $c = d(o)$, kjer $d(o)$ označuje čas trajanja operacije. Za vsako operacijo je tako določen avtomat $A = (\{o, \overline{o}, \underline{o}\}, \{c\}, I, \Delta, \overline{o}, \underline{o})$.

Primer časovnega avtomata za operacijo o_1 je podan na sliki 3.4. Kot lahko opazimo, je čas trajanja operacije štiri časovne enote.

Da dosežemo model s časovnim avtomatom za celoten problem razvrščanja, je potrebno zgraditi avtomat $A^i = (Q^i, C^i, I^i, \Delta^i, s^i, f^i)$ za vsako operacijo o_i. Za primer obravnavajmo problem z dvema strojema $M = \{m_1, m_2\}$ in dvema



Slika 3.4: Operacija modelirana s časovnim avtomatom



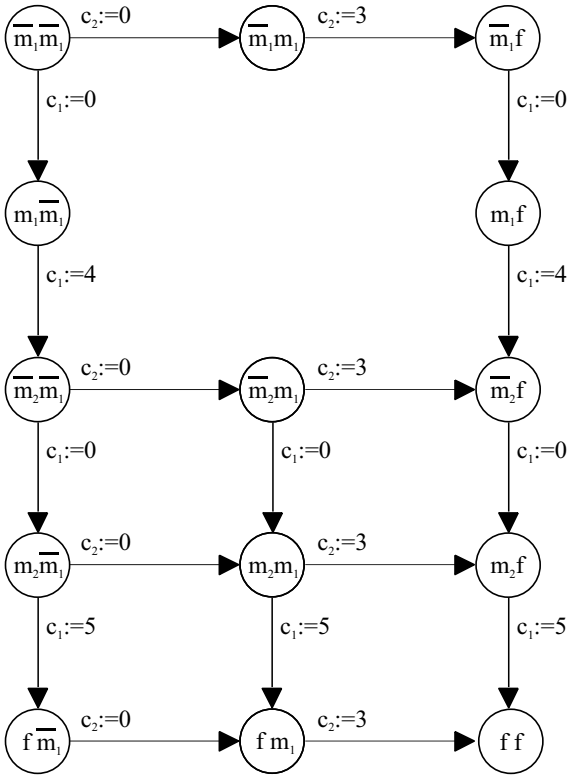
Slika 3.5: Nalogi modelirani s časovnima avtomatoma

nalogama $J = \{J_1, J_2\}$. Na sliki 3.5 sta s časovnim avtomatom predstavljeni dve nalogi $J_1 = \{o_1(m_1) \prec o_2(m_2)\}$ in $J_2 = \{o_3(m_1)\}$. Pri združevanju teh avtomatov je potrebna dodatna pozornost pri prednostnih povezavah med operacijami. Avtomatom so prehodi dovoljeni le, ko so njihovi predhodni avtomati v končnem stanju. Zgraditi je potrebno tak sestav avtomatov $A = (Q, C, I, \Delta, s, f)$, da so prepovedana vsa konfliktna stanja. Množica stanj je konfliktna, če vsebuje dve stanji q^j in q^k, za kateri velja $q^j = q^k = m$, kjer je $m \in M$ proizvodni vir, ki povzroča konflikt. Globalni avtomat, dobljen z združitvijo avtomatov A^1 in A^2 je prikazan na sliki 3.6.

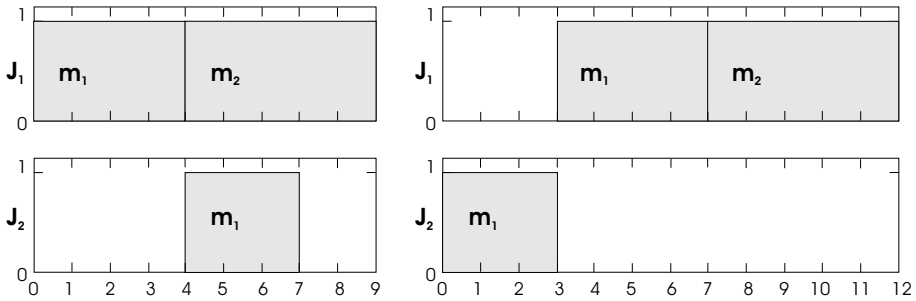
Za izvajanje avtomata A velja, da je zaključeno, če se izvajanje začne s stanjem $(s, \mathbf{0})$ in se konča s prehodom, ki nas vodi v stanje f. Vsaka zaključena izvedba ξ avtomata A določa izvedljivo razvrstitev S_ξ. Običajno obstaja več rešitev. Dve izmed možnih razvrstitev za obravnavan primer prikazuje slika 3.7. Tako je lahko iskanje optimalne rešitve kompleksen problem.

Rešitev k omenjenemu problemu je vpeljava hevrističnih postopkov. V [1] avtor predlaga uporabo algoritma najkrajše poti (*Shortest path algorithm*) za časovne avtomate za iskanje optimalne razvrstitve klasičnega problema razvrščanja sestava posamične obdelave.

V prispevkih [8, 9] avtorji uporabijo časovne avtomate za reševanje praktičnega problema razvrščanja v proizvodnji lakov. Za vsak recept (proizvodni postopek) je načrtan model s časovnim avtomatom. Uporabijo različna



Slika 3.6: Sestav posamične obdelave modeliran s časovnim avtomatom



Slika 3.7: Dve možni razvrstitvi za podan primer avtomata

hevristična pravila in na ta način zmanjšajo prostor iskanja rešitve.

3.1.3 Petrijeve mreže

Petrijeve mreže, imenovane po izumitelju, nemškemu matematiku Carlu Adamu Petriju, so se pojavile v začetku šestdesetih let [46]. Zgodovinski pregled in podroben pregled literature lahko najdemo v [69] ali v [106]. Kot grafično in matematično orodje služijo za modeliranje in analizo najrazličnejših sistemov, ki

so okarakterizirani kot konkurenčni in asinhroni. Z njimi je mogoče modelirati konfliktne dogodke. Omogočajo študij tako deterministčnih kot tudi stohastičnih sistemov.

V samem začetku so se Petrijeve mreže izkazale za uporabno orodje pri preučevanju informacijskih sistemov. Tudi proizvodni sistemi, ki jih srečujemo v sodobni industriji, posedujejo lastnosti, ki jih je mogoče obravnavati s Petrijevimi mrežami. Zato so široko uporabljene pri modeliranju in analizi različnih proizvodnih sistemov. Uspešno se uporabljajo tudi pri načrtovanju sekvenčnega vodenja.

Petrijeve mreže so splošnejše orodje kot avtomati, saj z njimi lahko modeliramo vzporedne aktivnosti in sinhronizacijo, kar z avtomati ni mogoče. Omogočajo matematično in grafično predstavitev sistemov, in pa tudi prikaz njihovega izvajanja.

Petrijeva mreža je struktura, sestavljena iz elementov dveh tipov, poimenovana kot mesto (P – *place*) in prehod (T – *transition*). Le elementa različnih tipov sta med sabo lahko povezana z usmerjenimi povezavami (*Arcs*). Večkratne povezave so označene z utežjo. V primeru, da je le-ta večja od ena, je grafično podana s številom, ki je podano ob usmerjeni povezavi. Mesto je grafično upodobljeno s krogom, prehod pa s črto oz. pravokotnikom. Označitev (M – *marking*) Petrijeve mreže določa njeno stanje. Označitev vsakemu mestu priredi nenegativno število žetonov (*Tokens*). Žetoni so grafično upodobljeni z ustreznim številom enakih malih črnih krogov (•). Petrijeva mreža je podana s strukturo in pa začetno označitvijo. Premikanje žetonov po Petrijevi mreži imenujemo izvajanje ali razvoj Petrijeve mreže.

Matematično Petrijeve mreže predstavimo z: $PN = (P, T, I, O, M_0)$, kjer je:

- $P = \{p_1, p_2, ..., p_g\}$ končna množica mest,
- $T = \{t_1, t_2, ..., t_h\}$ končna množica prehodov,
- $I : (P \times T) \rightarrow \mathbb{N}$ je funkcija vhodnih povezav. Če obstaja vhodna povezava z utežjo k, ki povezuje p_i na t_j, potem velja $I(p_i, t_j) = k$.
- $O : (P \times T) \rightarrow \mathbb{N}$ je funkcija izhodnih povezav. Če obstaja izhodna povezava z utežjo k, ki povezuje t_j na p_i, potem velja $O(p_i, t_j) = k$.

- $M : P \rightarrow \mathbb{N}$ je funkcija označitve, M_0 je začetna označitev.

Funkciji I in O določata uteži usmerjenih povezav. Prehod t_j je omogočen le v primeru, ko je izpolnjen pogoj $M(p_i) \geqslant I(p_i, t_j)$, $\forall p_i \in P$. Omogočen prehod se lahko proži. Sprožitev povzroči odvzem toliko žetonov v vsakem vhodnem mestu, kolikor je število povezav med posameznim vhodnim mestom in prehodom t_j. Hkrati se v vsakem izhodnem mestu ustvari toliko žetonov, kolikor je število povezav med prehodom t_j in posameznim izhodnim mestom. Ob proženju prehoda t_j, je nova označitev določena z enačbo (3.6).

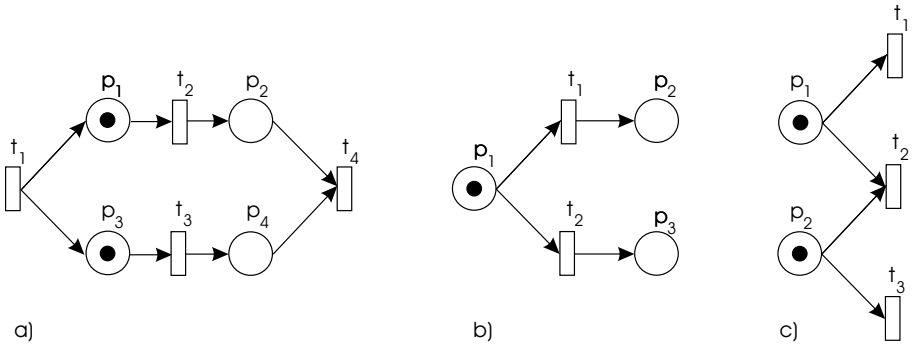
$$M'(p_i) = M(p_i) + O(p_i, t_j) - I(p_i, t_j) \ , \forall p_i \in P \tag{3.6}$$

Dinamika Petrijeve mreže je lahko podana tudi v obliki enačbe stanj [69]. Tako funkcija označitve M definira $g \times 1$ stolpni vektor $\mathbf{M}_k = [M(p_1), ..., M(p_g)]^T$, kjer je g število mest v Petrijevi mreži. Element na $i$tem mestu vektorja $\mathbf{M}_{k+1} (k \geq 0)$ podaja število žetonov v mestu p_i po k-tem proženju. k-ti prožilni vektor $\mathbf{u_k}$ je $h \times 1$ stolpni vektor, v katerem nastopa $h - 1$ ničel. Element, čigar vrednost je 1 in je na j-tem mestu, določa, da se v tem trenutku proži j-ti prehod. Poleg tega, definiramo še $g \times h$ vhodno matriko $\mathbf{I}$, kjer je element (i, j) določen z $I(p_i, t_j)$ in $g \times h$ izhodno matriko $\mathbf{O}$, kjer je element (i, j) določen z $O(p_i, t_j)$. Z uporabo teh vektorjev in matrik lahko zapišemo vektorsko enačbo stanj (3.7), ki določa označitev v naslednjem trenutku.

$$\mathbf{M}_{k+1} = \mathbf{M}_k + (\mathbf{O} - \mathbf{I}) \cdot \mathbf{u_k} \tag{3.7}$$

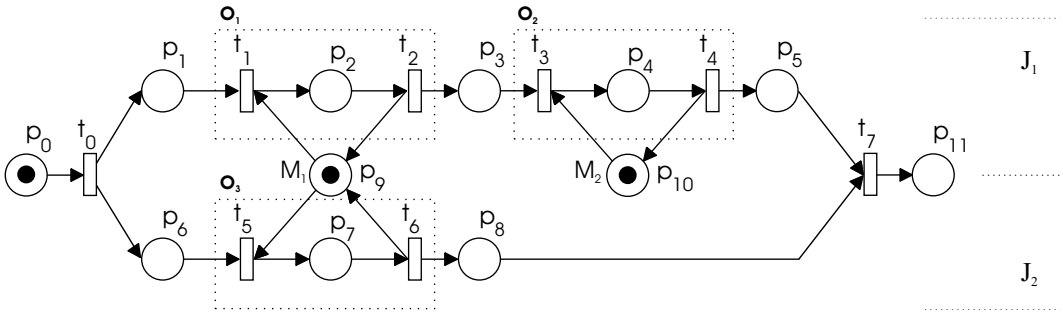
Pomembna lastnost Petrijevih mrež je, da omogočajo zapis in obravnavo situacij, ko nastopajo konkurenčni dogodki, konflikti ter zmede, ki se običajno pojavljajo v diskretno-dogodkovnih sistemih. S Petrijevimi mrežami lahko enostavno opišemo konkurenčne (vzporedne) dogodke (glej sliko 3.8a). Dva dogodka sta si konkurenčna, ko sta med sabo neodvisna in se lahko zgodita v kateremkoli zaporedju, t.j. prehod se lahko proži pred, za ali vzporedno z drugim prehodom. Dva dogodka sta v konfliktu, če se lahko zgodi en ali drug, vendar ne oba sočasno. Konflikt se pojavi med prehodi, ki so omogočeni z isto označitvijo, in kjer proženje enega prehoda onemogoči ostale prehode. Primer, ko s Petrijevo mrežo modeli-

ramo dva dogodka v konfliktu, je prikazan na sliki 3.8b. Situacijo, ko sta pojava konkurenčnosti in konflikta povezana, imenujemo zmeda (slika 3.8c).



Slika 3.8: Modeliranje pojavov, ko nastopa konkurenčnost (a), konflikt (b) in zmeda (c)

S Petrijevimi mrežami je mogoče na enostaven način izvajati kvalitativno analizo modeliranih sistemov. Uporabo Petrijevih mrež pri reševanju problematike razvrščanja ponazorimo z naslednjim primerom. Privzemimo, da imamo opravka s podobnim problemom kot pri primeru obravnave časovnih avtomatov. Opravka imamo z dvema strojema $M = \{M_1, M_2\}$, ki morata opraviti dve nalogi $J = \{J_1, J_2\}$, pri čemer je $J_1 = \{o_1(M_1) \prec o_2(M_2)\}$ in $J_2 = \{o_3(M_1)\}$. Na sliki 3.9 je obravnavan problem predstavljen s Petrijevimi mrežami.



Slika 3.9: Problem razvrščanja modeliran s Petrijevimi mrežami.

Med izvajanjem predstavljenega modela Petrijeve mreže opazujemo njeno označitev. Sled označitve v mestih, ki modelirajo operacije, predstavlja zaporedje operacij, ki jih je potrebno izvesti. Mesto p_9, ki modelira stroj M_1, uporabljata dve operaciji. To nam predstavlja konfliktno situacijo. Tako lahko hitro opazimo, da je možnih več različnih zaporedij dogodkov, ki nastopajo v danem sistemu. S pomočjo take rešitve pridemo do izvedljive razvrstitve opravil.

3.1.4 Matrični model

Tacconi s sod. [90] predstavlja matrični model, s katerim se da opisati diskretno dogodkovne sisteme, torej tudi proizvodne sisteme. Matrike potrebne za opis sistema so lahko pridobljene tudi direktno iz podatkovnih struktur proizvodnega sistema. Matrike predstavljajo model sistema in omogočajo njegovo analizo. Predvsem enostavna je simulacija takega modela.

Matrični model diskretno dogodkovnega sistema je predstavljen z nizom logičnih enačb. V zapisu, ki sledi, seštevanje označuje logično operacijo "ALI" in množenje logično operacijo "IN". Enačba matrike stanj (3.8) preveri pogoje za izvajanje naslednje naloge. Pogoji so podani z vektorjem stanja $\mathbf{x}$.

$$\mathbf{x} = \mathbf{F}_v \cdot \overline{\mathbf{v}}_c + \mathbf{F}_r \cdot \overline{\mathbf{r}}_c + \mathbf{F}_u \cdot \overline{\mathbf{u}}_c + \mathbf{F}_D \cdot \overline{\mathbf{u}}_D \tag{3.8}$$

Tu matrika $\mathbf{F}_v$ določa, v kakšnem zaporedju so naloge. Element $\mathbf{F}_v(i,j)$ ima vrednost 1, kadar je naloga j predhodnik naloge i. Vektor $\mathbf{v}_c$ govori o tem, katere naloge so končane. Z matriko $\mathbf{F}_r$ se določi potrebe po proizvodnih virih. Element matrike $\mathbf{F}_r(i,j)$ je enak 1, kadar je za izvedbo naloge i potreben vir z oznako j. Vektor $\mathbf{r}_c$ podaja vire, ki so razpoložljivi. Vhodna matrika $\mathbf{F}_u$ določa, kateri sestavni deli so potrebni. $\mathbf{F}_u(i,j)$ je enak 1, če je za vzpostavitev pravila i potreben izdelek z oznako j. Matrika $\mathbf{F}_D$ skupaj z vektorjem $\mathbf{u}_D$ podaja pravila za reševanje konfliktov, npr. ob uporabi deljenih virov.

Na osnovi teh pogojev, podanih z vektorjem stanj $\mathbf{x}$, se z enačbo (3.9) določi naloge ($\mathbf{v}_s$), ki se naj začno izvajati, in z enačbo (3.10) vire ($\mathbf{r}_s$), ki se naj sprostijo.

$$\mathbf{v}_s = \mathbf{S}_v \cdot \mathbf{x} \tag{3.9}$$

$$\mathbf{r}_s = \mathbf{S}_r \cdot \mathbf{x} \tag{3.10}$$

Z matriko $\mathbf{S}_v$ so podane naloge, ki se naj začno izvajati. Element $\mathbf{S}_v(i,j) = 1$ pomeni signal za začetek izvajanja naloge i, ko je pravilo j postavljeno na "1". V primeru, da je element matrike $\mathbf{S}_r(i,j)$ enak 1, se naj sprosti i-ti proizvodni vir, ko je postavljeno pravilo x_j na "1".

Izračunane zahteve se pošlje na proizvodni nivo. Z enačbo (3.11) se zahteve priredi na primerne izhodne signale.

$$\mathbf{y} = \mathbf{S}_y \cdot \mathbf{x} \tag{3.11}$$

Zapis matrične enačbe se da enostavno pretvoriti v zapis s Petrijevimi mrežami. To omogoča uporabo orodij analize Petrijevih mrež. Z njimi lahko tako analiziramo diskretno dogodkovne sisteme. S podanimi matričnimi enačbami lahko določimo matriko končanih aktivnosti $\mathbf{F}$ in matriko začetnih aktivnosti $\mathbf{S}$.

$$\mathbf{F} = [\mathbf{F}_u\ \mathbf{F}_v\ \mathbf{F}_r\ \mathbf{F}_y] \qquad \mathbf{S} = [\mathbf{S}_u\ \mathbf{S}_v\ \mathbf{S}_r\ \mathbf{S}_y] \tag{3.12}$$

Definirajmo X kot množico vseh možnih stanj vektorja stanj $\mathbf{x}$ in A kot množico vseh vektorjev $\mathbf{v}$ in $\mathbf{r}$. Potem nam zapis (A, X, F, S^T) predstavlja Petrijevo mrežo. Tu sta vpeljana ničelna vektorja $\mathbf{F}_y$ in $\mathbf{S}_u$, katerih namen je le zaključitev matrik $\mathbf{F}$ in $\mathbf{S}$. Tako matrika $\mathbf{F}$ predstavlja vhodno matriko in matrika $\mathbf{S}$ izhodno matriko.

Če označimo iteracijo diskretnega dogodka s k, enačba (3.13) predstavlja dinamični zapis diskretno dogodkovnega sistema.

$$\mathbf{M}_{k+1} = \mathbf{M}_k + (\mathbf{S^T} - \mathbf{F})^T \cdot \mathbf{x_k} \tag{3.13}$$

Omenjena enačba predstavlja tudi vektorsko enačbo stanj za izračun nove označitve Petrijeve mreže. Za popoln zapis, je potrebno določiti še prožilni vektor $\mathbf{x}_k$. V ta namen je uporabljena enačba (3.8), ki je lahko zapisana tudi kot:

$$\mathbf{x}_k = \overline{\mathbf{F}} \oplus \overline{\overline{\mathbf{M}}_k} \tag{3.14}$$

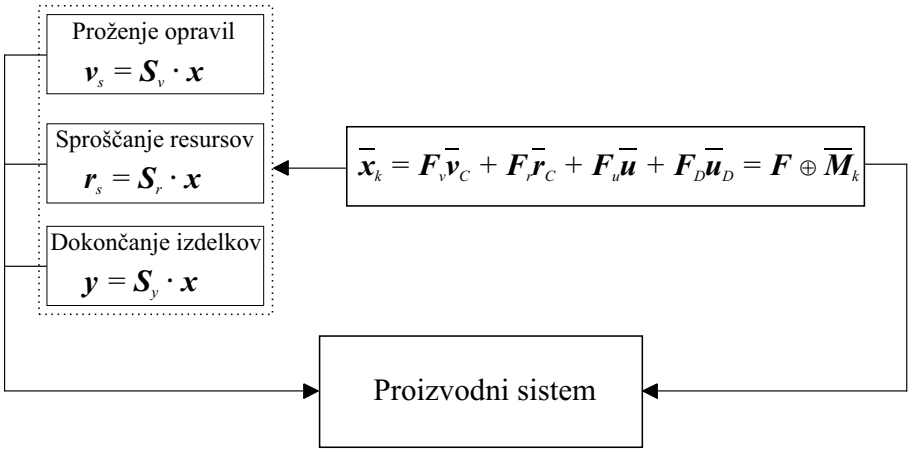
Simbol $\oplus$ označuje matrično logično množenje, znotraj katerega se namesto množenja in seštevanja zopet uporablja logični funkciji "IN" in "ALI". Z enačbama (3.13) in (3.14) tako lahko simuliramo diskretno dogodkovni sistem, podan z matričnim modelom.

V [75] je izračun prožilnega vektorja razdeljen na dva dela, kjer so zunanji

pogoji vpeljani ločeno, v obliki vhodnega vektorja $\mathbf{u}_D$.

$$\mathbf{x}_k = (\overline{\mathbf{F}} \oplus \overline{\overline{\mathbf{M}}}_k) \cdot \mathbf{u}_D \tag{3.15}$$

Prvi del enačbe (3.15) ($\overline{\mathbf{F}} \oplus \overline{\overline{\mathbf{M}}}_k$) tu zaobsega notranje pogoje. Na ta način je regulacijski del, predstavljen z $\mathbf{u}_D$, ločen od modela proizvodnega sistema.



Slika 3.10: Princip matričnega modela

Princip uporabe matričnega modela je grafično predstavljen s sliko 3.10. S predstavljenimi enačbami je enostavno mogoče izvesti simulacijo na tak način predstavljenega sistema. V [90] je podana tudi ideja simulacije časovnega matričnega modela. V ta namen je predlagan izračun novega stanja modela (3.13) v dveh korakih. V drugem delu uporabi še izračun, s katerim določi preostali čas, ko se operacija zaključi.

V literaturi se pojavlja veliko praktičnih primerov uporabe danega pristopa. Kot je predstavljeno v [90], je mogoče matrike, ki opisujejo model, zajemati direktno iz standardnih proizvodnih podatkovnih struktur, kot so kosovnica, sestavno drevo in pa podatki o potrebah po proizvodnih virih. V [14] je s pomočjo matričnega modela osnovano vodenje in nadzor fleksibilnega proizvodnega sistema. V [54] je obravnavano sprotno izogibanje pojavu zastoja. Sistem je predstavljen z matričnim modelom, kar omogoča uporabo simulacije in načrtovanje nadzornika diskretno dogodkovnega sistema. Huang s sod. [37] preučuje različne proizvodne strukture, za katere podaja formalni zapis z matričnim modelom. S kombiniranjem teh struktur omogoča predstavitev splošnega sestava tekoče obdelave. Tak

zapis sistema avtorji uporabijo za vzdrževanje razvrstitve ob pojavu različnih okvar.

3.1.5 Časovne Petrijeve mreže

S klasičnimi Petrijevimi mrežami je mogoče opisati logično strukturo in določiti obnašanje tako modeliranega sistema. Za detajlnejšo analizo pa bi potrebovali več informacije, kot nam ponujajo take Petrijeve mreže. Za študij časovnega obnašanja sistema in problema razvrščanja bi tako bilo potrebno modelirati tudi časovne zakasnitve.

Iz tega razloga se je v literaturi pojavilo več različnih pristopov k vpeljavi časa v klasične Petrijeve mreže. V 70-ih letih prejšnjega stoletja sta Merlin [63] in Ramchandani [79] neodvisno podala koncept vključitve časovnih zakasnitev v Petrijeve mreže. Podroben pregled načinov vpeljave časa v Petrijeve mreže lahko najdemo v [16] ali pa tudi v [25].

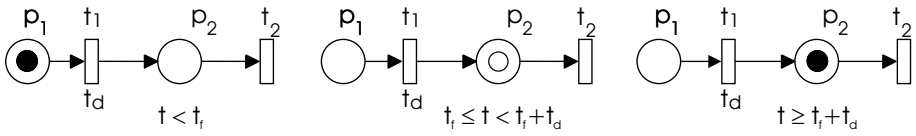
Časovne zakasnitve so lahko prirejene mestom in/ali prehodom. Časovne zakasnitve so lahko deterministične ali pa tudi stohastične [60]. Za obravnavo razvrščanja proizvodnih opravil zadošča že uporaba časovnih Petrijevih mrež, katerih časovne zakasnitve so deterministične. Pri obravnavi procesov je potrebno modelirati čase trajanja aktivnosti, ki jih prožijo dogodki. Ker so dogodki običajno modelirani s prehodi, je smiselno uporabiti koncept, ko so časovne zakasnitve aktivnosti predpisane s prehodi.

Kot podaja Bowden [16], obstajajo trije osnovni načini vključitve časa v Petrijeve mreže: čas trajanja proženja, držanja in omogočanja. V literaturi se sicer pojavlja več različnih poimenovanj vpeljav časa v Petrijeve mreže, odvisno od raziskovalnih skupin.

Princip časa trajanja proženja (*Firing duration*) je najprej uporabil Ramchandani [79] in se v raziskavah in aplikacijah pogosto uporablja. Časovne zakasnitve so predpisane prehodom. Ko postane prehod omogočen, se žeton v trenutku odstrani iz vhodnega mesta, vendar se ne pojavi v izhodnem mestu, dokler ne poteče čas, ki je predpisan prehodu. Izčrpen opis omenjenega principa in formalni zapis lahko najdemo v [109].

V primeru uporabe časa trajanja držanja (*Holding duration*) imamo opravka

z dvema tipoma žetonov: razpoložljivi in nerazpoložljivi žetoni. Le razpoložljivi žetoni lahko omogočajo prehod. Običajno je časovna zakasnitev predpisana prehodom. Ko se prehod sproži, se v trenutku odstrani žetone iz vhodnih mest in se ustvari nove žetone na izhodnih mestih. Na novo ustvarjen žeton se ustvari kot nerazpoložljiv, in to za toliko časa, kot je predpisano s prehodom, ki je ta žeton ustvaril. Ta princip je ilustriran s sliko 3.11, kjer so razpoložljivi žetoni predstavljeni z ustreznim številom polnih (črnih) malih krogov in nerazpoložljivi žetoni s praznimi malimi krogi. V primeru, ko je časovna zakasnitev, predpisana prehodu, različna od nič, je ta vrednost podana ob prehodu, npr. $f(t_1) = t_d$. Na sliki 3.11 je s t označen čas globalne ure in s t_f čas proženja prehoda.



Slika 3.11: Časovna Petrijeva mreža s proženjem po principu časa trajanja držanja

Tehniko, ki vpeljuje časovne zakasnitve po principu časa trajanja omogočanja (*Enabling duration*), je predstavil že Merlin [63]. V tem načinu je proženje prehoda izvedeno v trenutku, časovne zakasnitve pa so vpeljane tako, da mora biti prehod omogočen določen čas, preden se lahko sproži.

Principa trajanje proženja in trajanje držanja sta pod pogojem, da obakrat uporabljamo časovno zakasnitev predpisano prehodom, v bistvu enakovredni predstavitvi časa. Edina razlika je v tem, da so žetoni v prvem primeru zadržani v prehodu in v drugem v mestu. Vseeno pa je opis Petrijevih mrež s časom trajanja držanja bližji opisu nečasovnih Petrijevih mrež, saj v tem primeru prehodi niso aktivni preko časovne periode, žetoni pa med proženjem prehoda ne izginejo, ampak se takoj pojavijo v izhodnih mestih, podobno kot pri nečasovnih mrežah.

Poglavitna razlika med uporabo časa trajanja držanja in omogočanja se pokaže, ko opazujemo pojav konflikta ali zmede. V tem primeru je z eno označitvijo omogočenih več prehodov. V primeru uporabe časa trajanja omogočanja, lahko proženje enega prehoda prekine omogočanje ostalih prehodov, saj se je označitev, ki je omogočala prejšnjo situacijo, spremenila. Pri obravnavi proizvodnih procesov s prehodi modeliramo dogodke, ki prožijo aktivnosti. V

primeru, ko se bi označitev ustrezno spremenila, bi to pomenilo prekinitev aktivnosti, ki se je že začela izvajati. Tako je v primerih, ko predpostavimo, da se že začeta operacija ne more prekiniti, primernejša uporaba časa trajanja držanja.

Ker se v pričujočem delu ukvarjamo z modeliranjem proizvodnih procesov, kjer operacij običajno ne prekinjamo, bomo v nadaljevanju uporabljali princip trajanja držanja. Formalni zapis take Petrijeve mreže je:

$TPN = (P, T, I, O, s_0, f)$, kjer je pomen oznak naslednji:

- P, T, I, O predstavljajo enake oznake, kot so bile že definirane pri zapisu klasične Petrijeve mreže,
- s_0 je začetno stanje časovne Petrijeve mreže,
- $f : T \rightarrow \mathbb{R}_0^+$ je funkcija, ki vsakemu prehodu $t_j \in T$ priredi nenegativno realno vrednost, t.j. vsakemu prehodu je prirejena deterministična časovna zakasnitev. Časovne zakasnitve so lahko predstavljene z $1 \times h$ vrstičnim vektorjem $\mathbf{f}$, kjer je j-ti element vektorja določen z $f(t_j)$.

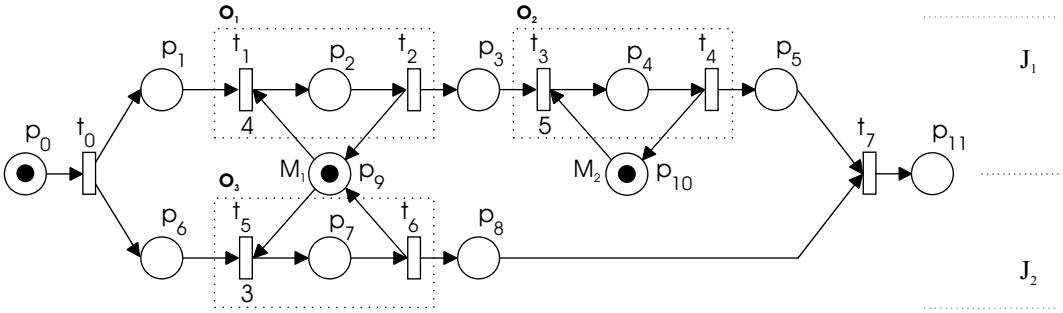
Stanje časovne Petrijeve mreže je kombinacija treh funkcij, kjer prva popisuje porazdelitev razpoložljivih žetonov v mestih, druga porazdelitev nerazpoložljivih žetonov in tretja predstavlja funkcijo, ki določa preostali čas držanja posameznih nerazpoložljivih žetonov. Stanje časovne Petrijeve mreže je tako podano s sledečim zapisom:

$s = (m, n, r)$ kjer je,

- $m : P \rightarrow \mathbb{N}$ funkcija označitve razpoložljivih žetonov, ki definira $g \times 1$ stolpni vektor $\mathbf{m}$, kjer je i-ti element vektorja določen z $m(p_i)$,
- $n : P \rightarrow \mathbb{N}$ funkcija označitve nerazpoložljivih žetonov, ki definira $g \times 1$ stolpni vektor $\mathbf{n}$, kjer je i-ti element vektorja določen z $n(p_i)$,
- r funkcija, ki vsakemu nerazpoložljivemu žetonu, ki nastopa v mestu, priredi preostali čas držanja (stanje pripadajoče lokalne ure oz. časovnika – *clock*). V primeru, da je število nerazpoložljivih žetonov v mestu p_i enako l, t.j. $n(p_i) = l$, funkcija $r(p_i)$ definira vektor l pozitivnih realnih števil $\mathbf{r}(p_i) = [r(p_i)[1], r(p_i)[2], ..., r(p_i)[l]]$; funkcija r je prazna za vsa tista mesta p_i, za katera velja $n(p_i) = 0$.

Prehod t_j je omogočen z dano označitvijo le v primeru, ko je zadoščen pogoj $m(p_i) \geqslant I(p_i, t_j)$, $\forall p_i \in P$. Proženje prehoda se smatra kot trenutno. Za vsak na novo ustvarjen žeton se ustvari časovnik, katerega začetno stanje je določeno s prehodom, ki je žeton ustvaril. Ko noben izmed prehodov ni več omogočen, se čas globalne ure poveča za vrednost najmanjšega časovnika. Takrat postane nerazpoložljiv žeton v mestu s tem časovnikom razpoložljiv, časovnik se odstrani, vrednosti ostalih časovnikov pa se zmanjšajo za toliko, kot se je spremenila globalna ura. Sledi ponovna kontrola pogoja omogočanja. Pravilo proženja je podrobneje razloženo v 4. poglavju.

Zopet obravnavajmo problem razvrščanja, ki je bil predstavljen že pri obravnavi časovnih avtomatov in klasičnih Petrijevih mrežah. S pomočjo časovnih Petrijevih mrež lahko upoštevamo tudi časovno komponento. Model je predstavljen na sliki 3.12, kjer lahko ob prehodih opazimo tudi podatek o časovni zakasnitvi. Postopek modeliranja bo podrobneje predstavljen v 4. poglavju.

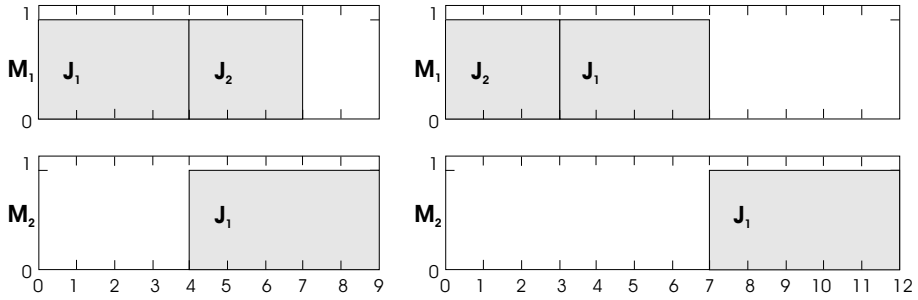


Slika 3.12: Problem razvrščanja modeliran s časovnimi Petrijevimi mrežami.

Z izvajanjem predstavljenega modela Petrijeve mreže dobimo različne možne scenarije, ki nam predstavljajo različne izvedljive razvrstitve za predstavljen problem razvrščanja. Tudi z uporabo časovnih Petrijevih mrež pridemo do enakih rezultatov, kot s pomočjo časovnih avtomatov, kjer smo obravnavali taisti primer. Rešitvi, ki jih dobimo s pomočjo modela časovnih Petrijevih mrež, sta predstavljeni na sliki 3.13.

3.1.6 Barvne Petrijeve mreže

Kaj hitro lahko opazimo, da postane grafična predstavitev Petrijevih mrež zelo kompleksna, ko poskušamo modelirati realne probleme. Glavni razlog je v tem, da



Slika 3.13: Dve možni razvrstitvi.

imamo opravka le z enim tipom žetona. Pri barvnih Petrijevih mrežah (*Coloured Petri nets – CPN*) lahko obravnavamo različne tipe žetonov; vsak tip predstavlja različno *barvo*, ki je prirejena žetonom. Barvne Petrijeve mreže je najprej definiral K. Jensen [40].

Da lahko definiramo CPN, je potrebno najprej predstaviti *večkratne množice* (*Multisets*). Za večkratne množice velja, da so podobne množicam, le da se lahko v večkratni množici posamezni elementi pojavijo več kot enkrat. Za primer, ko imamo množico $\{a, b, c\}$ in ji dodamo element b, imamo še vedno opravka z množico $\{a, b, c\}$. Če pa imamo opravka z večkratno množico, bi pri dodajanju elementa b dobili večkratno množico $\{a, b, b, c\}$.

Večkratna množica m, definirana nad neprazno množico S, je funkcija $m \in [S \mapsto \mathbb{N}_0]$. Nenegativno naravno število $m(s) \in \mathbb{N}_0$ predstavlja število elementov s v večkratni množici m, t.j. kardinalno število. Množico vseh večkratnih množic nad S označimo z S_{MS}.

Prehodi se lahko prožijo v različnih načinih, glede na barvo, s katero so opisani. To vpliva tudi na definicijo vhodne in izhodne funkcije. Vhodna funkcija barvne Petrijeve mreže je tako zdaj element $I(p,t) \in [C(t) \mapsto C(p_1)_{MS}]$. Tako je z izrazom $I(p,t)(c')(c)$ določeno število žetonov barve c, ki se naj odstranijo iz mesta p, ob proženju prehoda t v načinu c'. Podobno je definirana izhodna funkcija O.

Definicijo barvne Petrijeve mreže povzamemo po [6], ki je sicer poenostavljena varianta zapisa, kot ga je predstavil Jensen [40]. Kot take, nam zadovoljujejo za predstavitev problemov, ki se pojavljajo pri razvrščanju. Formalni zapis take barvne Petrijeve mreže je podan kot:

$PN = (P, T, I, O, C, M_0)$, kjer so:

- P in T končni množici mest in prehodov,
- I in O sta vhodni in izhodni funkciji,
- C je funkcija barv, ki nad množico $P \cup T$ določa končne neprazne množice,
- M je funkcija definirana nad P, ki določa začetno označitev tako, da velja $M_0(p) \in C(p)_{MS}$, $\forall p \in P$.

Vsak žeton je v taki strukturi določen kot par elementov (p, c), kjer je $p \in P$ in $c \in C(p)$. Označitev M pa je določena kot večkratna množica nad množico žetonov. Tako $M(p)(c)$ določa število žetonov, označenih z barvo c, v mestu p.

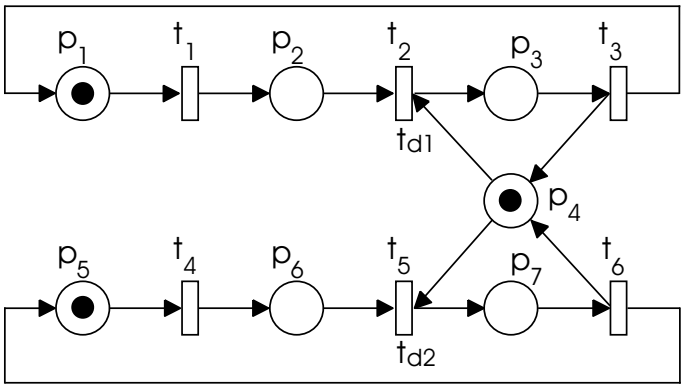
Prehod $t \in T$ je omogočen v označitvi M z ozirom na barvo $c' \in C(t)$ le in samo v primeru, ko velja $M(p)(c) \geq I(p,t)(c')(c)$, $\forall p \in P, c \in C(p)$. Tako omogočen prehod se lahko sproži. Po sprožitvi dobimo novo označitev, ki je določena z enačbo (3.16).

$$M'(p)(c) = M(p)(c) + O(p,t)(c')(c) - I(p,t)(c')(c), \forall p \in P, c \in C(p) \quad (3.16)$$

Podobno kot pri navadnih mrežah, so lahko tudi v primeru barvnih Petrijevih mrež vpeljane časovne zakasnitve. Vsakemu prehodu lahko priredimo več različnih časovnih zakasnitev, odvisno od števila načinov, v katerih se lahko posamezen prehod proži.

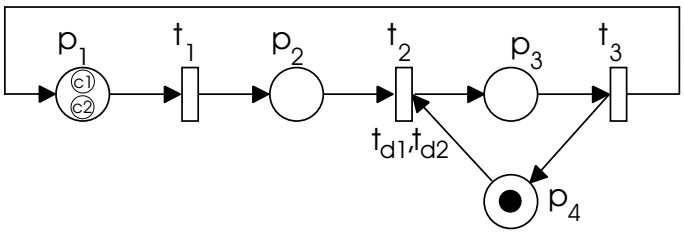
Transformacija barvnih Petrijevih mrež v navadne je enolična, medtem ko obratno to ne velja. Obstoj ekvivalentne navadne Petrijeve mreže je pomemben, ker nam omogoča posploševanje osnovnih konceptov in metod analize navadnih Petrijevih mrež tudi na barvne Petrijeve mreže.

Za ilustracijo uporabnosti barvnih Petrijevih mrež obravnavajmo primer z dvema napravama, ki izvajata isti postopek z istim virom. Model problema s klasičnimi Petrijevimi mrežami je podan na sliki 3.14. Mesta p_1, p_2, p_3 predstavljajo operacije potrebne za izvajanje prvega postopka, medtem ko mesta p_5, p_6, p_7 predstavljajo operacije drugega postopka.



Slika 3.14: Problem modeliran s klasičnimi Petrijevimi mrežami.

Glede na to, da sta postopka identična, bi lahko z barvnimi Petrijevimi mrežami problem modelirali z eno proizvodno potjo, za vsak postopek pa bi uporabili drugo barvo žetona. Glede na barve, bi posameznih prehodom določili še pravila proženja in časovne zakasnitve. Model s strukturo CPN je predstavljen na sliki 3.15.



Slika 3.15: Problem modeliran z barvnimi Petrijevimi mrežami.

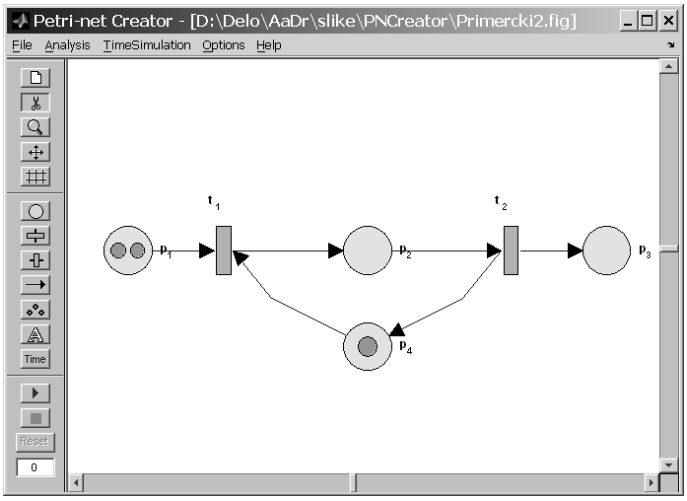
Barvne Petrijeve mreže so uporabne za predstavitev problemov razvrščanja, znotraj katerih se pojavljajo podsistemi s podobnim obnašanjem. Kadar imamo opravka s takimi simetričnimi pojavi, nam barvne Petrijeve mreže omogočajo kompaktnejši zapis modela. To pa nam pogostokrat pride prav pri obravnavi proizvodnih sistemov [2, 45, 82].

3.2 Programska orodja

Praktična uporabnost matematičnih formalizmov je predvsem pogojena s prisotnostjo primernih računalniških orodij. V nadaljevanju si bomo na kratko ogledali nekaj tipičnih pripomočkov.

3.2.1 Splošno-namenska orodja za Petrijeve mreže

Za učinkovito delo s Petrijevimi mrežami bi potrebovali orodje, ki bi nam ponujalo grafični urejevalnik in pripomočke za analizo tako modeliranega sistema. S pomočjo splošno-namenskih orodij je možno zgraditi različne pripomočke. Primer takega orodja je razvojno okolje Matlab, v katerem je bilo razvito orodje *Petri-Net Creator* [68]. Orodje je sestavljeno iz večih funkcij in iz uporabniškega vmesnika, ki omogoča enostavno gradnjo modelov navadnih Petrijevih mrež in njihovo analizo. Tako zgrajeni model je mogoče shraniti in pa tudi izvoziti v delovno okolje Matlaba, kjer je na voljo nadaljnji obravnavi s preostalimi funkcijami orodja. Integracija omenjenega orodja znotraj okolja Matlab predstavlja pomembno prednost, saj nam le-ta ponuja množico učinkovitih numeričnih metod, uporabo grafičnih vmesnikov ipd. Na sliki 3.16 je prikazan posnetek grafičnega vmesnika Petri-Net Creator, skupaj z enostavnim primerom modela Petrijeve mreže. Kot bomo predstavili v nadaljevanju (glej 4. poglavje), je bil za omenjeno orodje razvit postopek modeliranja proizvodnega okolja ter pripomoček za kvantitativno analizo časovnih Petrijevih mrež.



Slika 3.16: Petri-Net Creator.

Na trgu obstaja veliko različnih orodij, ki omogočajo simulacijo različnih Petrijevih mrež. Tudi raziskovalna skupina iz Romunije je razvila orodje za delo s Petrijevimi mrežami, ki je uporabno kot dodatek k Matlabu, poimenovano *Petri Net Toolbox* [62]. Za razreševanje konfliktnih situacij je na voljo uporaba prioritetnih nastavitev ali pa uporaba naključne izbire. Zelo uveljavljeno orodje za

delo z barvnimi Petrijevimi mrežami je programski paket *CPN* [81]. Ameriška raziskovalna skupina ISIS pa je razvila dodatek k Matlabu za nadzorno vodenje Petrijevih mrež, ki je osnovano s P-invariantno analizo [38].

3.2.2 Orodja za vodenje projektov

Tudi projekti so, podobno kot proizvodni procesi, sestavljeni iz različnih med sabo povezanih dejavnosti (nalog), ki za svojo izvedbo potrebujejo nek čas in nek nabor virov. Tako lahko tudi proizvodne procese obravnavamo z orodji za vodenje projektov.

Tudi pri delu s projekti, ki jih je potrebno najprej načrtati in potem slediti njihovem izvajanju, dandanes praktično ne gre brez računalniške podpore. Obstaja kar nekaj takih orodij, primat najbolj uveljavljenega orodja na tem področju pa v zadnjem času zavzema orodje *Microsoft Project* [64]. K popularnosti omenjenega orodja prispeva tudi dejstvo, da je kot standardna komponenta vključen v zadnjo izdajo programskega paketa družine *Microsoft Office 2003*.

MS Project omogoča gradnjo projektnega plana. Vanj se vnese vse potrebne operacije, skupaj s predvidenim časom izvajanja in viri, ki so potrebni za njihovo izvedbo. Na enostaven način se poda tudi povezave med posameznimi operacijami. Aktivnosti so lahko podane v obliki strukturirane členitve dela (*Work Breakdown Structure – WBS*). Pri tem lahko posamezne operacije poljubno povezujemo. Posameznim operacijam so lahko prirejeni tudi stroški, ki nastopijo ob njihovem izvajanju.

Za analizo projektov, podanih s MS Projectom, so na voljo uveljavljene tehnike na področju upravljanja projektov, metodi PERT in CPM. Da je upoštevana tudi omejenost virov, ki so potrebni za izvedbo projekta, ima MS Project na razpolago tudi funkcijo avtomatske zakasnitve operacij (*Resource levelling*). V ta namen operacije, ki povzročajo preobremenitev, prerazporedi ali jih razcepi. Projekt in njegov plan je prikazan z različnimi pogledi, t.j. gantogram, histogram virov, itd.

S pomočjo dodatnih modulov, ki jih MS Project vsebuje, lahko tako orodje prilagodimo za delovanje v podporo razvrščanju proizvodnih opravil in ga integriramo v celoten informacijski sistem. Omenjen pristop, skupaj z ilustrativnim

praktičnim primerom, je predstavljen v poglavju 6.

3.2.3 Komercialna orodja za razvrščanje

Na trgu obstaja veliko komercialnih ponudnikov programskih orodij, ki naj bi zagotavljala podporo pri razvrščanju opravil v proizvodnih sistemih. Običajno se taka orodja vključuje v že obstoječe informacijske sisteme, ki so vpeljani v proizvodnji, vendar ne omogočajo funkcionalnosti razvrščanja. Pregled nekaterih orodij lahko najdemo v [35].

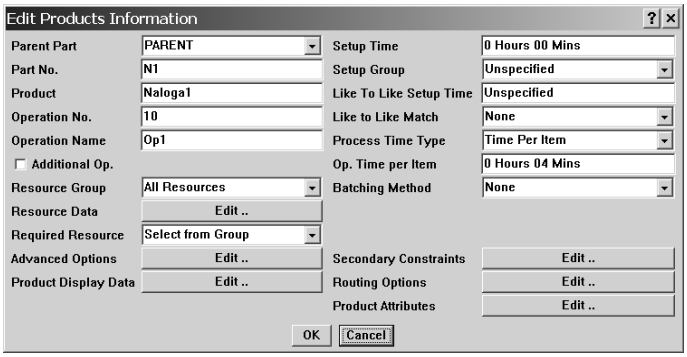
Osnovne naloge teh orodij so razvrščanje proizvodnih opravil, prikaz in spremljanje izvajanja razvrstitve ter generiranje poročil. Poleg avtomatskega razvrščanja lahko operater plan upravlja tudi ročno. Razvrstitve dela se razpošlje na ustrezna delovna mesta.

Obstaja več različnih algoritmov, ki jih orodja uporabljajo za generiranje razvrstitve [23]. Najenostavnejši je algoritem, ki deluje v načinu, ko čas napreduje v sklopih (*Block-time approach*). Algoritem razvršča naloge na razpoložljive proizvodne vire eno za drugo, t.j. najprej vse operacije prve naloge, nato naslednje, itd. Operacije se lahko razvršča naprej po časovni osi, lahko pa tudi nazaj ali kot kombinacija obeh načinov. Z različnimi pravili se določa, v kakšnem vrstnem redu se bodo naloge razvrščale. Uporaba algoritma, ki deluje na dogodkovno voden način (*Event-driven approach*), velja za naprednejši pristop. V tem primeru pa se za vsak prosti vir posebej ugotovi nabor operacij, ki bi se v določenem trenutku lahko pričele izvajati. Izmed njih se izbere tisto, ki jo izbere uporabljeno pravilo, t.j. razna hevristična pravila. Orodja, ki uporabljajo tak pristop, so že precej dražja.

Kot primer takega programskega orodja si bomo nekoliko podrobneje pogledali *Preactor* [77]. Ponujen je kot družina orodij, ki se med sabo razlikujejo glede na obseg funkcionalnosti in kompleksnosti algoritmov in pa seveda glede na ceno. Enostavnejši paketi so sicer cenejši, vendar pa ponujajo le najbolj osnovne pripomočke, ki so potrebni pri razvrščanju majhnih in enostavnih problemov. Za kompleksnejše probleme je potrebno zagotoviti naprednejši sistem, ki pa je seveda tudi cenovno manj ugoden.

Kako uporabljati omenjeno orodje, bomo prikazali na že znanem problemu,

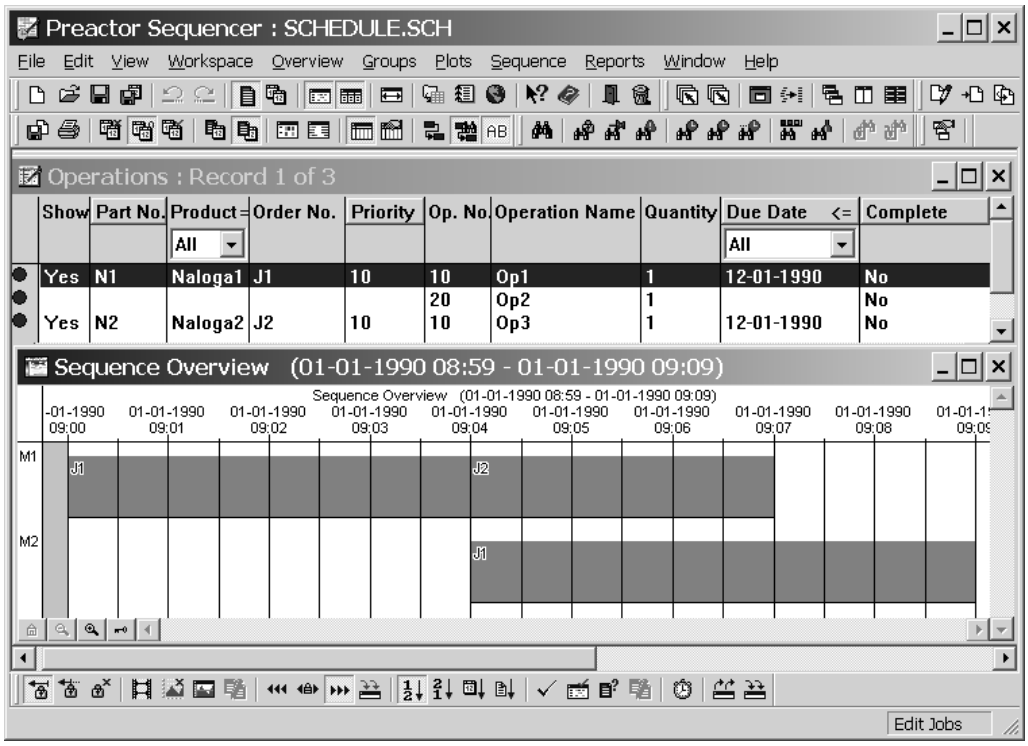
kjer razvrščamo dve nalogi $J_1 = \{o_1(M_1) \prec o_2(M_2)\}$ in $J_2 = \{o_3(M_1)\}$ na dva proizvodna vira $M = \{M_1, M_2\}$. Da lahko zgradimo plan razvrstitve opravil, moramo v podatkovno bazo najprej vnesti podatke o proizvodnem sistemu. Najprej podamo proizvodne vire, ki nastopajo v proizvodnem sistemu, v nadaljevanju pa je potrebno definirati že vse izdelke, ki jih je možno izdelati. Za prikaz našega primera definiramo dva proizvodna vira in dve nalogi, podani kot dva izdelka. Vsaka naloga je sestavljena iz več operacij. Kako definiramo operacijo o_1, je prikazano na sliki 3.17.



Slika 3.17: Definicija operacije o_1.

Orodje Preactor ponuja več različnih algoritmov, s katerimi lahko razvrščamo operacije. Grafična interaktivna planerska tabla, prikazana na sliki 3.18, nam omogoča uporabo teh avtomatskih algoritmov in pa tudi interaktivno poseganje planerja. Za dan problem razvrščanja uporabimo metodo razvrščanja naprej in na ta način dobimo razvrstitev, kot je podana z gantogramom na sliki 3.18. Iz danega pogleda lahko pregledujemo tudi vse naloge (izdelke) in operacije, ki nastopajo v razvrstitvi.

V praksi se taka orodja vključuje v informacijske sisteme, ki so že vpeljani v industriji. Da lahko Preactor vključimo v tak sistem, potrebujemo najbolj napreden sistem iz družine, t.j. *Preactor APS*, skupaj s standardnim modulom SMC (*Static Material Control*), ki skrbi za povezave med posameznimi naročili in na ta način zagotavlja sestavne elemente za končni izdelek. Študija, kako vključiti orodje Preactor v poslovni informacijski sistem Gosoft, je predstavljena v [28].



Slika 3.18: Razvrstitev operacij, ki jo določi orodje Preactor.

4. Uporaba Petrijevih mrež pri načrtovanju in vodenju proizvodnih sistemov

Proizvodne sisteme lahko smatramo kot konkurenčne in asinhrone diskretno dogodkovne sisteme. Obnašanje takih sistemov je običajno zelo kompleksno. Tako za njihovo analizo in v pomoč pri njihovi implementaciji potrebujemo formalne metode. V splošnem lahko probleme, ki nastopajo pri obravnavi proizvodnih sistemov, razdelimo na dve fazi. Prva je faza načrtovanja, ko se gradi nov proizvodni sistem oziroma se v sistemu vršijo večje spremembe. V fazi obratovanja pa so glavni problemi dodeljevanje virov, razvrščanje, vodenje in nadzor sistema.

Petrijeve mreže predstavljajo močno grafično in matematično modelersko orodje. Služijo za modeliranje in analizo najrazličnejših sistemov, ki so okarakterizirani kot konkurenčni in asinhroni. Petrijeva mreža je družina formalizmov, ki podajajo okvir za obravnavo proizvodnih sistemov skozi cel življenjski cikel proizvodnega procesa [85]. Zato so široko uporabljene pri modeliranju in analizi raznih proizvodnih sistemov [78, 89, 106, 111].

Znotraj področja proizvodnih sistemov so bile Petrijeve mreže uporabljene v različne namene [85, 105]. Med njimi omenimo:

- Zgodovinsko gledano so se raziskave najprej osredotočale na kvalitativno analizo modelov proizvodnih sistemov. Analiza dosegljivosti pokaže, ali sistem lahko doseže neko stanje. Ostale lastnosti Petrijevih mrež se uporablja za ugotavljanje stabilnosti diskretno dogodkovnih sistemov, cikličnega obnašanja in prisotnosti mrtvih stanj.
- Kasneje so se pojavile potrebe tudi po časovnih modelih, ki predstavljajo kvantitativen opis sistema. V ta namen se uporablja časovne Petrijeve mreže. Pri obravnavi proizvodnih operacij, ki so stohastične narave, se uporabljajo stohastične Petrijeve mreže.

- Ko modeli, zaradi svoje kompleksnosti, ne omogočajo več učinkovite matematične analize, se pojavi potreba po uporabi simulacije, s katero je mogoča kvalitativna in kvantitativna analiza diskretno dogodkovnih sistemov.

- Modeli Petrijevih mrež so lahko v pomoč pri izvedbi vodenja proizvodnih sistemov. Z njimi lahko predstavimo izvršljive specifikacije logičnih krmilnikov [36, 67]. Slabost načrtovanja sekvenčnega vodenja z lestvičnimi diagrami je v oteženem pregledovanju takega zapisa ter v težavnem morebitnem naknadnem spreminjanju sistema. Po drugi strani pa je sekvenčno vodenje, ki je načrtano s Petrijevimi mrežami, preprosto za implementacijo in vzdrževanje.

- Petrijeve mreže, v kombinaciji s še drugimi pristopi, služijo kot koristen pripomoček pri proizvodnem planiranju in razvrščanju. Model v obliki Petrijeve mreže obsega tako prednostne povezave kot tudi omejitve proizvodnih virov. Z vpeljavo hevrističnih pravil lahko razvijemo optimalne oz. sub-optimalne razvrščevalnike. S Petrijevimi mrežami lahko opišemo tudi razne poslovne strategije, ki se uporabljajo pri izdelavi proizvodnega plana.

- Petrijeve mreže se v veliki meri uporabljajo tudi za upravljanje delovnih tokov (*Workflow management*) [94, 95]. Upravljanje delovnih tokov se ukvarja s procesi, ki nastopajo v administrativnih okoljih.

Uporaba Petrijevih mrež v proizvodnih aplikacijah predstavlja raziskovalno področje, ki je že kar nekaj časa aktivno, vseeno pa obstaja še precej odprtih vprašanj. Tako velja, da je avtomatizacija postopka modeliranja zahtevno opravilo. Če se zadovoljimo z dejstvom, da je kreativnost gradnje modela nekoliko omejena, lahko do neke mere omogočimo avtomatsko generiranje modela in s tem tudi olajšamo naknadno analizo.

Pri uporabi formalnih metod za analizo proizvodnih sistemov je pomemben model, ki mora biti podan dovolj natančno. Obstajajo različni koncepti določanja pravilnosti. V osnovi velja, da je model sistema pravilen, ko je model, določen po specifikacijah, ekvivalenten njegovi implementaciji. Pravilnost modela je lahko zagotovljena tudi, če tak sistem zadovoljuje zahtevane vnaprej določene karakteristike.

Da lahko Petrijeve mreže koristno uporabimo na področju obravnave proizvodnih sistemov, je njuno, da si najprej pogledamo, kakšne lastnosti Petrijevih mrež lahko študiramo in kakšne vrste analiz nam omogočajo.

4.1 Analiza

Petrijeve mreže, kot matematično orodje, omogočajo preverjanje številnih lastnosti. V kontekstu modeliranih proizvodnih sistemov te lastnosti omogočajo ugotavljanje (ne)prisotnosti raznih funkcionalnih lastnosti sistema. Model, zgrajen s časovno Petrijevo mrežo, lahko uporabljamo za izvajanje kvalitativne in kvantitativne analize sistemov.

4.1.1 Lastnosti Petrijevih mrež

Ločimo lahko dva tipa lastnosti Petrijevih mrež. To so vedenjske (*Behavioural properties*) in strukturne lastnosti (*Structural properties*). Vedenjske lastnosti so odvisne od začetne označitve Petrijeve mreže, strukturne pa le od njene topologije in so neodvisne od začetne označitve. Na tem mestu bomo omenili le nekaj lastnosti Petrijevih mrež, ki jih lahko koristno uporabimo za analizo delovanja proizvodnih sistemov.

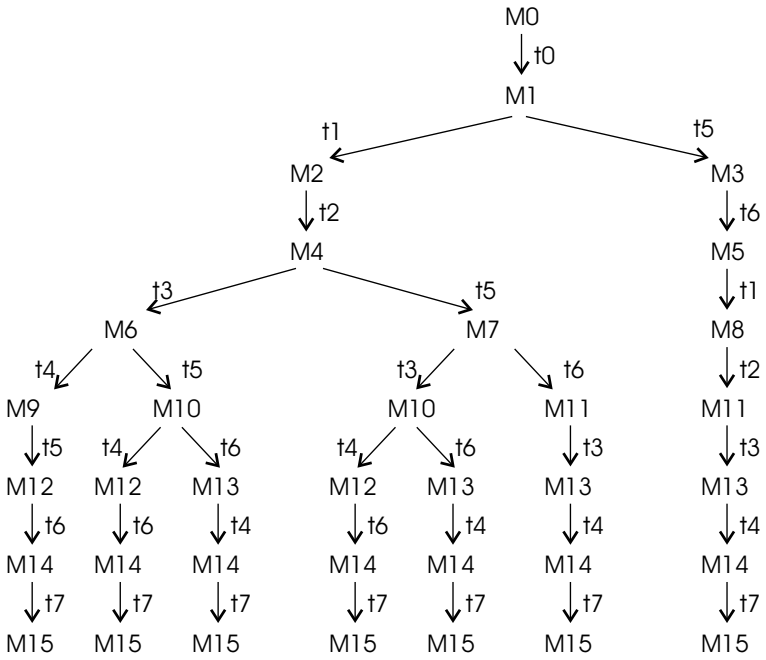
Za neko označitev Petrijeve mreže M rečemo, da je *dosegljiva* iz začetne označitve M_0, če obstaja sekvenca proženja prehodov, ki nas pripelje iz začetne označitve do M. Množica R predstavlja vsa dosegljiva stanja iz dane začetne označitve. Neko mesto p, ki se nahaja v Petrijevi mreži je *B-omejeno*, če zanj velja $M(p) \leq B$, $\forall M \in R$ in $B \in \mathbb{N}$. Celotna Petrijeva mreža je *B-omejena*, če so vsa mesta v mreži B-omejena. Če velja, da je Petrijeva mreža omejena za vse začetne označitve, pravimo da je *strukturno omejena*. Koncept omejenosti je pogosto interpretiran kot stabilnost diskretnih proizvodnih sistemov. Petrijeva mreža je *konservativna*, če obstaja vektor $\mathbf{w} = [w_1, ..., w_g]^T$ pri $w_i > 0$, $i = 1, ...g$, tako da velja $w^T\mathbf{M} = w^T\mathbf{M}_0$, $\forall M \in R$. Taka lastnost drži v primerih, ko modeliramo proizvodne stroje. Kadar žetoni predstavljajo proizvodne vire, se njihovo število skozi izvajanje Petrijeve mreže ne spreminja, t.j. število proizvodnih virov se ne spreminja. Prehod t je *živ*, če za vsako označitev $M \in R$ obstaja sekvenca,

ki pripelje do označitve, ki omogoča proženje prehoda t. Prehod t je *mrtev*, če sistema iz neke označitve ne moremo pripeljati do označitve, ki bi omogočala proženje tega prehoda t. Petrijeva mreža vsebuje *zastoj* (*Deadlock*), če zanjo obstaja označitev $M \in R$, ki ne omogoča nobenega prehoda. Zastoj se lahko pojavi ob nepravilnem prirejanju virov.

4.1.2 Analiza dosegljivosti

V namene razvrščanja je nadvse uporabna analiza dosegljivosti. Pri taki analizi želimo iz nekega začetnega stanja izpeljati vsa dosegljiva stanja. Na ta način dobimo drevo dosegljivosti (*Reachability tree*). Ko imamo opravka s končnim številom stanj, lahko s tako analizo ugotovimo vse vedenjske lastnosti, ki smo jih omenili predhodno.

Vozlišča v drevesu dosegljivosti predstavljajo označitve, dosegljive iz začetnega stanja, povezave med vozlišči pa predstavljajo proženje prehodov, ki vodijo v naslednjo označitev.



Slika 4.1: Drevo dosegljivosti za primer s slike 3.9.

Za primer si poglejmo drevo dosegljivosti za primer modela Petrijeve mreže, ki je predstavljen na sliki 3.9. Drevo izhaja iz začetne označitve $M0$. Iz drevesa

dosegljivosti je moč opaziti, da je v tem stanju omogočen le prehod $t0$. Proženje tega prehoda vodi v označitev $M2$, ki pa omogoča dva prehoda – drevo se razveji. Na enak način lahko iz drevesa spremljamo nadaljnje spreminjanje stanja Petrijeve mreže.

Opazimo, da ima drevo dosegljivosti za omenjen primer končno število nivojev. Kaj hitro pa lahko pridemo do problemov, ko imamo opravka z drevesom z neskončnim številom nivojev. V ta namen je bilo razvito prekrivno drevo (*Coverability tree*), ki upošteva stanja, ki so se v strukturi drevesa že predhodno pojavljala. Z omenjenim drevesom pa je mogoče analizirati tudi Petrijevo mrežo, ki je neomejena. V ta namen se uporabi simbol ω, ki predstavlja element neskončnosti. Njegova lastnost je, da za vsako končno naravno število k velja $\omega > k$, $\omega \leq \omega$ in $\omega + k = \omega$. Iz prekrivnega drevesa lahko ugotovimo, ali je mreža omejena, saj v tem primeru simbol ω v drevesu ne nastopa. V nadaljevanju podajmo algoritem, ki generira končno prekrivno drevo [69].

Algoritem 4.1

1. Izhodišče drevesa naj bo označitev $M0$ in pripadajoče vozlišče poimenuj z oznako 'nov'.
2. Dokler obstajajo vozlišča z oznako 'nov', se naj izvajajo naslednji koraki:
   (a) Izberi novo označitev;
   (b) Če je M identična označitvi, ki se je že pojavila na poti iz izhodišča drevesa, potem vozlišče označi z oznako 'star' in pojdi na naslednjo novo označitev;
   (c) Dokler iz označitve M obstajajo omogočeni prehodi, se naj izvaja naslednje:
      i. Če v označitvi M ni omogočen noben prehod, vozlišče označi z 'mrtvi konec';
      ii. Določi označitev M', ki rezultira s proženjem t v označitvi M;
      iii. Če na poti iz izhodišča drevesa obstaja taka označitev M'', da velja $M'(p) \geq M''(p)$ za vsako mesto p in $M' \neq M''$ (označitev M'' je prekrita), potem nadomesti $M'(p)$ z ω za vsak p, za katerega velja $M'(p) > M''(p)$;

iv. Za označitev M' vpelji vozlišče in med vozliščema M in M' ustvari povezavo t. Vozlišču dodaj oznako 'nov';

(d) Pojdi na 2. korak.

Z drevesom dosegljivosti oziroma s prekrivnim drevesom lahko analiziramo, katera stanja so dosegljiva iz nekega začetnega stanja. Če bi imeli na razpolago celotno drevo, bi lahko poiskali tudi optimalno pot. Poleg tega lahko na osnovi celotnega drevesa dosegljivosti razberemo vse vedenjske lastnosti mreže. Običajno pa je za realne probleme tako drevo preveliko, da bi ga generirali v celoti. Ob uporabi hevrističnih pravil, lahko drevo omejimo in iščemo le preko najobetavnejših delov drevesa. Na ta način omogočimo določitev sub-optimalne rešitve.

4.1.3 Invariantna analiza

Invariantna analiza sloni na strukturnih lastnostih Petrijeve mreže [78]. Strukturo mreže podaja incidenčna matrika $\mathbf{A}$, ki je določena z zapisom $\mathbf{A} = \mathbf{O} - \mathbf{I}$. $g \times 1$ stolpni vektor $\mathbf{x}$ predstavlja *P-invarianto*, če velja $\mathbf{A}^T \cdot \mathbf{x} = 0$. Neničelni elementi v P-invarianti predstavljajo uteži prirejene mestom. Tako je utežena vsota žetonov v teh mestih konstanta za vse dosegljive označitve. Sledi, da so mesta, utežena z neničelnimi elementi P-invariante, omejena. $h \times 1$ stolpni vektor $\mathbf{y}$ predstavlja *T-invarianto*, če velja $\mathbf{A} \cdot \mathbf{y} = 0$. Neničelni elementi v T-invarianti predstavljajo število proženj določenega prehoda, ki pripada prožilni sekvenci, ki nas privede iz začetnega stanja m_0 nazaj v začetno stanje. Invariantna analiza omogoča analizo nekaterih lastnosti Petrijevih mrež.

Najprej izračunajmo P-invarianto za primer, ko obravnavamo Petrijevo mrežo s slike 3.9. Dobimo več možnih rešitev, izmed katerih podajmo: $Z_1 = [0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0]^T$ in $Z_2 = [0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0]^T$. Opazimo lahko, da izračun, podan z enačbo $S = z_1 \cdot M(p_1) + z_2 \cdot M(p_2) + ... + z_{12} \cdot M(p_{12})$, ostane konstanten za vsako začetno označitev M_0 in vsako P-invarianto Z. Velja tudi posplošitev, ki pravi, da za vsako označitev M, dosegljivo iz začetne označitve M_0, velja izraz $Z \cdot M_0 = Z \cdot M$. Iz tega sledi, da so mesta, katerih označitve so v izrazu $Z \cdot M$ utežene z neničelnimi koeficienti, vselej omejena ne glede na začetno označitev mreže.

Če poskušamo v nadaljevanju za isti primer izračunati še T-invarianto, opazimo, da za ta primer T-invarianta ne obstaja. Iz tega lahko sklepamo, da Petrijeve mreže ni mogoče pripeljati nazaj v začetno označitev, potem ko je bil eden izmed prehodov prožen.

Pri analizi kompleksnih Petrijevih mrež pa se lahko uporabljajo tudi različne metode redukcije mreže. Del mreže se poenostavlja tako, da ob tem ohranjamo določene lastnosti. Metode redukcije pa so običajno zelo omejene, saj je poenostavitev težko izvedljiva ali pa sploh ni možna.

4.1.4 Simulacija časovne Petrijeve mreže

Uporaba analitičnih metod je zaradi kompleksnosti sistemov pogostokrat nemogoča ali je uporabna le v omejenih okoliščinah. V takih primerih se lahko poslužujemo simulacije, kljub dejstvom, da je taka analiza računalniško in časovno potratna.

Diskretno dogodkovna simulacija je proces, preko katerega model oponaša vedenje diskretno dogodkovnega sistema [57]. S simulacijo izvajanja Petrijeve mreže lahko opazujemo sled označitve mreže skozi čas. Da je simulacija možna, pa je potrebno vpeljati neke odločitvene strategije ob pojavu konflikta. V primeru obravnave proizvodnih sistemov, lahko kot odločitvene strategije uporabimo različna hevristična pravila. S preučevanjem mest, ki predstavljajo proizvodne vire, lahko opazujemo razvrstitev proizvodnih operacij, t.j. kdaj in kateri vir je potreben za izvedbo določene naloge. Na ta način lahko opazujemo, kako bi se sistem obnašal pri uporabi različnih pravil in s tem zadovoljeval različnim proizvodnim kriterijem.

V okolju Matlab smo zgradili simulacijsko funkcijo časovne Petrijeve mreže [29], s pomočjo katere lahko določimo sled označitve časovne Petrijeve mreže in s tem razvrstitev opravil v modeliranem problemu razvrščanja. Simulator je sposoben obravnavati situacije, ko nastopi konflikt. Za razreševanje konfliktnih situacij smo vgradili različna prioritetna pravila, omogoča pa še vgradnjo poljubnih dodatnih pravil. Prostor rešitve problema razvrščanja, ki je predstavljen s časovno Petrijevo mrežo, je tako bistveno zmanjšan. Pomembno je le, da izberemo primerno pravilo za problem, ki ga obravnavamo.

Simulacijska funkcija določa novo stanje časovne Petrijeve mreže $s_{k+1} = (\mathbf{m}_{k+1}, \mathbf{n}_{k+1}, r_{k+1})$ v vsakem računskem koraku po sledečem postopku:

1. Določi označitev klasične (nečasovne) Petrijeve mreže, t.j., upoštevani so le razpoložljivi žetoni. Za nečasovno Petrijevo mrežo se izračuna ustrezen prožilni vektor $\mathbf{u_k}$. Na tem mestu se tudi razrešuje konfliktne situacije – vpeljava prioritetnih opravil.

   - $\mathbf{m}_k \rightarrow \mathbf{u_k}$.

2. V primeru, da ni prožen noben prehod ($\mathbf{u_k} = 0$), čas simulacije teče dalje – vrednosti časovnikov se zmanjšajo za toliko, kot je trenutno najmanjša vrednost časovnika.

   - $\mathbf{M}_k = \mathbf{m}_k + \mathbf{n}_k$,
   - $T_s = \min_{p_i \in P}(\min_l(\mathbf{r}_k(p_i)[l]))$ in $\mathbf{r}(p_i)$ ni prazen,
   - $\mathbf{r}_{k+1}(p_i) = \mathbf{r}_k(p_i) - T_s$, $\forall r(p_i)[l]$, $\forall p_i \in P$ in vektor $\mathbf{r}(p_i)$ ni prazen – odstrani funkcije za žetone, katerih vrednost postane nič,
   - $n_{k+1}(p_i) = dim(\mathbf{r}_{k+1}(p_i)), \forall p_i \in P$ in vektor $\mathbf{r}(p_i)$ ni prazen,
   - $\mathbf{m}_{k+1} = \mathbf{M}_k - \mathbf{n}_{k+1}$.

3. V preostalih primerih ($\mathbf{u_k} \neq 0$), čas ostane nespremenjen. Prožilni vektor $\mathbf{u_k}$ definira novo stanje časovne Petrijeve mreže, t.j. razporeditev žetonov po mestih in ustrezna začetna stanja časovnikov na novo ustvarjenih žetonov.

   - $\mathbf{M}_k = \mathbf{m}_k + \mathbf{n}_k$,
   - $\mathbf{M}_{k+1} = \mathbf{M}_k + (\mathbf{O} - \mathbf{I}) \cdot \mathbf{u_k}$,
   - $n_{k+1}(p_i) = n_k(p_i) + O(p_i, t_j) \cdot u_{kj}$, $\forall p_i \in P$ in $\forall t_j \in \{t \in T : f(t) > 0\}$,
   - $\mathbf{r}_{k+1}(p_i) = \mathbf{r}_k(p_i)$; $\mathbf{r}_{k+1}(p_i)[h] = f(t_j)$, za $h = n_k(p_i) + 1,\ \ldots\ ,\ n_{k+1}(p_i)$.
   - $\mathbf{m}_{k+1} = \mathbf{M}_{k+1} - \mathbf{n}_{k+1}$.

Simulacija se konča, ko v mreži ni več nerazpoložljivih žetonov ali v primeru, ko globalna ura doseže nek vnaprej določen končni čas simulacije.

S podanim algoritmom se proženje prehodov izvede v trenutku. To je rešeno na način, da se čas ne spreminja, dokler je kateri izmed prehodov omogočen. Predpostavimo pa, da je prepovedana situacija, ko se ciklično pojavlja omogočanje in proženje prehodov. Pri tem cikličnost razumemo kot nepretrgano zaporedje omogočenih prehodov, katerih čas zakasnitve je enak 0. Tak cikel bi zaustavil napredovanje simulacije. Taka situacija v primeru modeliranja proizvodnih sistemov ni možna oziroma nastopi le ob nepravilnem modeliranju.

4.2 Področja uporabe

Odvisno od tega, s katero stopnjo življenjskega cikla imamo opravka, potrebujemo različne podatke o delovanju sistema. V fazi načrtovanja potrebujemo približne ocene, s pomočjo katerih lahko hitro ocenimo, katere alternative ne zadovoljujejo predpisanih zahtev. V nadaljevanju so izbrane alternative izboljšane, potrebni so bolj natančni zapisi modelov in natančnejši postopki analize učinkovitosti.

4.2.1 Vrednotenje učinkovitosti in optimizacija

Med pripravljalno fazo, ko se izvaja postopek načrtovanja, smo osredotočeni na povezave med podsistemi in na njihov kvantitativni vpliv na celotno obnašanje sistema. Ta analiza je potrebna za odločitve v zvezi z združevanjem virov v skupine, izbiro opreme, razporeditvijo virov, strategijo o proizvodnih postopkih, itd. Petrijeve mreže so orodje, s katerim lahko izvajamo take analize. Pomembna lastnost Petrijevih mrež pa je tudi dejstvo, da lahko te modele in rezultate analiz na enostaven način uporabimo tudi v nadaljnjih fazah načrtovanja. Z navadnimi Petrijevimi mrežami lahko opišemo logično strukturo in obnašanje sistema. Za časovno analizo sistema pa je potrebno uporabiti časovne Petrijeve mreže. Z njimi lahko ugotavljamo različne lastnosti proizvodnje, kot je proizvodna zmogljivost sistema, izkoristek proizvodnih virov, odkrivanje ozkih grl itd.

Obstaja več različnih optimizacijskih tehnik, ki jih lahko delimo na razvojno naravnane (*Generative* – na osnovi danih kriterijev dinamične omejitve generirajo niz odločitev) ali pa tehnike na osnovi ocenjevanja (*Evaluative* – za določen niz odločitev ocenimo učinkovitost sistema). Pogosto se pojavljajo metode, ki

temeljijo na analizi grafa dosegljivosti. Ker problem iskanja rešitve po celotnem prostoru kaj hitro postane NP-težak, se te analize uporabljajo v kombinaciji z različnimi tehnikami razvrščanja (hevristična pravila, razne tehnike iskanja, itd). Za enostavne modele je za ocenjevanje določenih kriterijev mogoče uporabiti tudi invariantno analizo. Tak pristop uporabi Zuberek [110], ki s pomočjo redukcije in analize P-invariance ugotavlja in nadzoruje prepustnost sistema in njegov obhodni čas (*Cycle time*). V industrijski praksi so problemi običajno zelo obsežni, tako se v takih primerih najpogosteje uporablja simulacija.

Kot podaja Zimmermann s sod. [108], obstajajo naslednji tipični problemi, ki se običajno pojavljajo v fazi načrtovanja proizvodnih sistemov:

- Problem izbire proizvodnih virov, pri čemer je potrebno upoštevati njihovo kapaciteto, stroške in podvrženost napakam.
- Uravnavanje proizvodnje izdelkov, ko je potrebno zagotoviti ustrezno razmerje različnih končnih izdelkov.
- Pogosto se pojavlja potreba po uporabi vmesnih skladišč. V tem primeru se je potrebno odločiti o tem, kako jih namestiti in kakšne naj bodo njihove velikosti.
- V proizvodnih sistemih se pogosto pojavljajo proizvodni viri, kot so avtomatsko vodeni vozički (AGV), tekoči trakovi ipd, ki služijo za transport obdelovancev oz. materiala. Tudi te vire je potrebno smiselno namestiti in določiti njihovo število ali pa tudi njihovo hitrost delovanja.

V praksi se pri optimizaciji proizvodnih sistemov pogostokrat pojavlja kombinacija omenjenih problemov.

4.2.2 Planiranje proizvodnje

V delu se ukvarjamo z modeli proizvodnega procesa. Do sedaj smo predpostavljali, da so naloge, ki jih mora proizvodni sistem izgotoviti, dane. V tem razdelku pa bomo obravnavali tudi proces planiranja proizvodnje, ki definira naloge. Planiranje proizvodnje določa dolgoročni proizvodni plan, s katerim želimo čimbolj

optimalno zadostiti zahtevam, ki jih podajajo končni potrošniki oz. porabniki. Pri tem se uporabljajo različne poslovne strategije.

Očitno je, da se pri gradnji celotnega modela proizvodnega procesa pojavlja tudi potreba po upoštevanju informacije o tem, katera strategija je uporabljena znotraj določenega proizvodnega postopka. Na osnovi tako zgrajenega modela lahko preučujemo vpliv posameznih strategij na celotno delovanje proizvodnega sistema. Obstaja več pristopov k planiranju in vodenju proizvodnje, ki so se uveljavili v proizvodnji industriji [97]. Med njimi sta osnovna pristopa, ki delujeta po načelu potiskanja in vlečenja, obstaja pa še mnogo drugih izvedenih različic.

Vodenje proizvodnega sistema po načelu *potiskanja* (*Push control*) sloni na ideji pospeševanja nalog v proizvodnji. Tako je delovanje proizvodnega sistema omejeno navzven z razpoložljivostjo surovin in s kapaciteto vmesnih in končnih skladišč. Z uporabo take strategije, surovine "potiskajo"proizvodnjo. Na ta način so minimizirani dostavni časi, kar omogoča hitrejše odzivanje na potrebe potrošnikov. Vse to pa vpliva na povečanje stroškov proizvodnje v teku (*Work In Progress – WIP*), ki pa so običajno tudi zelo pomembni. Sistemi planiranja potreb po materialih (*Material Requirement Planning – MRP*) so sistemi, ki delujejo po načelu potiskanja. Za generiranje proizvodnih planov običajno uporabljajo napovedi potreb in kosovnice.

Načelo potiskanja obravnava vsako delovno mesto kot samostojen, zaprt podsistem, izoliran od okolja, ki svoje delo opravlja, ne da bi ga pri tem zanimalo, kaj delajo drugi. Koncept MRP se lahko uspešno uporabi takrat, kadar je možno napovedovanje potreb in kadar so zagotovljeni pravilni in zanesljivi osnovni podatki o proizvodnji (vsaj definicije izdelkov in proizvodni proces). Kljub večim pomanjkljivostim je planiranje z MRP v praksi univerzalno in daleč najbolj uporabljan model planiranja in vodenja proizvodnje.

Po drugi strani pa pri vodenju proizvodnje po načelu *vlečenja* (*Pull control*), potrebe potrošnikov sprožajo proizvodnjo, t.j. "vlečejo proizvodnjo". V tem primeru je pobudnik izvedbe dela na nekem delovnem mestu naslednje delovno mesto v proizvodni verigi, ki takrat, ko mora izvesti neko operacijo, zahteva od predhodnika, da izvede njemu dodeljeno operacijo in obdelovance preda na naslednje delovno mesto. Načelo vlečenja vsako delovno mesto obravnava kot člen proizvodne verige in želi čim bolj kontinuiran in nemoten tok skoznjo. S takim

postopkom so stroški WIP zmanjšani na minimum na račun zmanjšanja kvalitete dostave izdelkov potrošnikom, t.j. pojavljajo se lahko zakasnitve v dostavi. Postopku, ki uporablja tak način vodenja proizvodnje, pravimo proizvodnja ob pravem času oz. proizvodnja brez zalog (*Just-In-Time production – JIT*).

Sistem *Kanban* je tipičen predstavnik vodenja po načelu vlečenja. Sistem Kanban je način oskrbovanja delovnih mest z materialom oz. obdelovanci. Bil je razvit v tovarni Toyota in temelji na ideji, da krmilimo materialni tok v proizvodnji, ki uporablja načelo JIT. Za nemoteno delovanje proizvodnje so potrebna vmesna skladišča. Delni proces porabnik vzame z odlagalnega mesta le toliko blaga, kolikor ga potrebuje v danem trenutku, delni proces proizvajalec pa manjkajočo količino nadomesti z novo proizvodnjo. Tako dobimo več samostojnih zaporedno nanizanih regulacijskih zank. Sam izraz kanban pomeni kartico, ki opredeljuje količino, ki jo porabnik zahteva od proizvajalca. Te kartice krožijo znotraj proizvodnih enot in določajo vse količine, ki se izdelujejo v procesu. Število kanbanov določa velikost vmesnih skladišč.

Obstaja ogromno literature, ki primerja omenjeni strategiji in poudarja prednosti ene ali druge v določenih situacijah [15, 88]. V proizvodnih sistemih se v praksi več uporablja sisteme MRP, medtem ko sistemi Kanban omogočajo boljše rezultate, kadar jih je le mogoče uporabiti. Z namenom zedinjenja prednosti obeh omenjenih strategij, se je v literaturi pojavilo več različnih hibridnih algoritmov vodenja. Spearman [87] je predlagal hibridno proizvodno strategijo CONWIP (*CONstant Work In Progress*). Tudi v tem primeru je, podobno kot v Kanban sistemu, sistem prožen z zahtevo po izdelavi nekega izdelka. Za vodenje proizvodnje se ne uporablja kanban kartic, ampak se razvrščanje nalog izvaja na način, kot je to narejeno s sistemi MRP. Tako sistem CONWIP omogoča boljši nadzor WIPa kot sistem MRP in učinkovitejše zaporedje izvajanja nalog, kot to omogoča Kanban. Zhang in Chen [104] sta na osnovi CONWIPa razvila model za enojno proizvodno linijo, ki je primeren za reševanje z celoštevilčnim linearnim programiranjem. Z modelom določita optimalno proizvodno zaporedje in velikost ene šarže. V [17] avtorji preučujejo izdelčni proizvodni sistem, kateremu so vhodni materiali oskrbovani iz dveh proizvodnih linij. Pri tem je vodenje osnovano z metodo CONWIP. Za koordinacijo operacij na teh dveh linijah zopet uporabijo celoštevilčno linearno programiranje. Hibridni algoritem za večstopenjsko in

večlinijsko izdelčno proizvodnjo lahko najdemo v [7].

Ob uporabi različnih poslovnih strategij se pojavljajo različna vprašanja glede optimiziranja proizvodnje. Določiti je potrebno mesta, kjer bodo vmesna skladišča, njihovo velikost, katero strategijo je najbolj smiselno uporabiti. V ta namen se pojavlja več različnih pristopov modeliranja. Pojavljajo se pristopi s čakalnimi vrstami (*Queueing network*) [61] ali pa modeli v obliki matematičnih programov [17, 104]. Predvsem uporaben pa je pristop s Petrijevimi mrežami [96, 106], saj jih je mogoče uporabiti na tudi na ostalih področjih v proizvodnji. Pristopi k modeliranju poslovnih strategij s pomočjo Petrijevih mrež bodo podrobneje podani v podpoglavju 4.3.1.

4.2.3 Razvrščanje

Vrednotenje učinkovitosti sistema se vrši z analizo časovno odvisnih karakteristik sistema, razvrščanje pa je aktivnost, ki skrbi za njegovo učinkovito delovanje. S časovnimi Petrijevimi mrežami lahko modeliramo tako funkcionalnost, časovne omejitve, kot tudi omejitve proizvodnih virov. Z njimi se določi omogočene prehode, nato se s pomočjo razvrščanja odloča o tem, kateri izmed omogočenih prehodov se naj prožijo. Obstaja več koristnih lastnosti v pristopu k razvrščanju s Petrijevimi mrežami. Z njimi lahko obravnavamo razne kompleksne povezave med nalogami, proizvodnimi potmi in raznimi proizvodnimi viri. Velja pa tudi, da je razvrstitev, generirana s Petrijevimi mrežami, dogodkovno vodena in ne povzroča zastojev. Ob neprimerni izbiri hevristične funkcije za izračun razvrstitve, pa nam lahko to predstavlja računsko zelo kompleksno nalogo.

Kot pripomoček k razvrščanju lahko uporabimo tudi invariantno analizo. Z analizo P-invariance lahko iščemo konfliktne prednostne povezave [93]. V primeru, da bi take povezave obstajale, bi bil problem razvrščanja nerešljiv. S T-invarianco lahko ugotavljamo obstoj prožilne sekvence, ki ne daje nobenega učinka, t.j. prožilna sekvenca, ki nas pripelje zopet do začetnega stanja.

V namene razvrščanja se bolj pogosto uporablja analiza dosegljivosti. Teoretično lahko z iskanjem optimalne poti v drevesu dosegljivosti določimo tudi optimalno razvrstitev za tako modeliran sistem. Optimalna pot predstavlja zaporedje proženja prehodov. Toda že pri majhnih modelih imamo opravka z obsežnim dre-

vesom dosegljivosti, kar nam predstavlja računsko zahteven problem. Tako so se uveljavile razne hevristične metode, ki nam omogočajo določitev sub-optimalne rešitve.

Enega izmed prvih prispevkov na tem področju je predstavil Carlier s sod. [18], ki je uporabil polinomski algoritem za izračun (sub-)optimalne sekvence proženja. V [52], kjer je problem razvrščanja fleksibilnega proizvodnega sistema modeliran s časovnimi Petrijevimi mrežami, je uporabljena hevristična funkcija in na ta način je omejen prostor iskanja. Isti avtorji omenjen pristop uporabijo na primeru proizvodnega sistema [51]. V zadnjem času se pojavljajo tudi razni pristopi, ki uporabljajo tehnike umetne inteligence. Yu s sod. [102, 103] problem razvrščanja predstavi kot iskanje optimalne poti v grafu dosegljivosti. Iskanje optimalne poti je izvedeno s pomočjo hevurističnih metod, osnovanih na tehnikah umetne inteligence, tako da se upošteva le najbolj obetajoča vozlišča v grafu dosegljivosti.

Za proučevanje vedenjskih lastnosti se lahko uporablja tudi simulacija časovnih Petrijevih mrež, s katerimi je predstavljen problem razvrščanja. Med simulacijo se med večimi omogočenimi prehodi naključno odločamo o tem, katerega bomo prožili. Na ta način lahko opazujemo sled označitve skozi čas, kar že predstavlja izvršljivo razvrstitev. Namesto naključnega proženja omogočenih prehodov, pa lahko uporabimo tudi različne hevristične funkcije [107]. Le-te se uporabi v situacijah, ko se omogočeni prehodi pojavljajo v konfliktu. Z različnimi hevurističnimi pravili, ki jih uporabimo za odločanje v takih konfliktnih primerih, lahko proizvedemo razvrstitve, ki minimizirajo različne kriterije razvrstitve. Podrobna obravnava kriterijev in kateri izmed njih nam v določeni situaciji najbolj ustreza, je podana v 2. poglavju. Delovanje takega simulatorja smo predstavili v podpoglavju 4.1.4.

4.3 Modeliranje proizvodnje

Proizvodni sistem predstavlja skupek različnih operacij, virov in prednostnih povezav med njimi. Kako se naj operacije izvajajo, je lahko določeno z različnimi poslovnimi strategijami. Proces planiranja procesov in razvrščanje proizvodnje se običajno še vedno izvajata ločeno. Mehanizem Petrijevih mrež omogoča mo-

deliranje vseh teh elementov, ki nastopajo v proizvodnih sistemih, na enostaven in pregleden način [89, 93, 111]. Obstaja tudi nekaj pristopov, ki težijo k skupni obravnavi planiranja in razvrščanja [21, 48, 101].

Splošen postopek načrtovanja modela s Petrijevimi mrežami bi lahko predstavili z naslednjimi koraki. Najprej je potrebno identificirati operacije, proizvodne vire in razne omejitve, ki nastopajo v proizvodnem sistemu. V nadaljevanju je potrebno ugotoviti medsebojne odvisnosti med identificiranimi dogodki, operacijami. Vire grupiramo po skupinah, glede na njihovo uporabo. Za deljene vire je potrebno določiti strategijo izbiranja teh virov. Ko je znana še informacija o proizvodnih poteh za posamezne izdelke, lahko načrtamo zasnovo Petrijeve mreže. V nadaljevanju označimo mesta, ki predstavljajo status proizvodnih virov in stanje proizvodnega postopka. Določiti je potrebno tudi postavitev žetonov glede na začetno stanje procesa in kapaciteto posameznih virov. Na koncu je potrebno preveriti, ali model zadovoljivo predstavlja obravnavan sistem in ga modificirati, če je to potrebno.

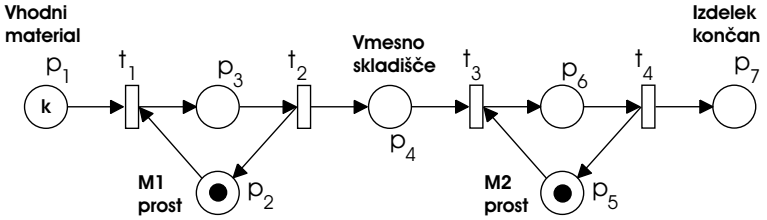
Za opis proizvodnih sistemov, uporabimo časovne Petrijeve mreže, ki so predstavljene v 3. poglavju. Z mesti predstavimo proizvodne vire, naloge ali pa proizvodne operacije. S prehodi modeliramo razne odločitve in pravila, ki se uporabljajo za dodeljevanje oz. sprostitev virov ali za zagon oz. končanje naloge. Žetoni podajajo stanje modeliranega sistema, t.j. stanje proizvodnih virov, pretok materiala.

4.3.1 Modeliranje poslovnih strategij

Naprej si nekoliko podrobneje poglejmo, kako uporabiti Petrijeve mreže za opis raznih strategij, ki se uporabljajo v sistemih za upravljanje in planiranje proizvodnih sistemov. Poleg tega, da lahko z njimi predstavimo različne poslovne strategije, omogočajo tudi upoštevanje različnih parametrov sistema, kot so čas trajanja posameznih operacij, kapaciteto virov ipd. Na ta način omogočajo natančno analizo vpliva posameznih strategij na delovanje proizvodnega sistema. Osnovni princip modeliranja s Petrijevimi mrežami in primerjavo različnih strategij lahko najdemo v [15, 96].

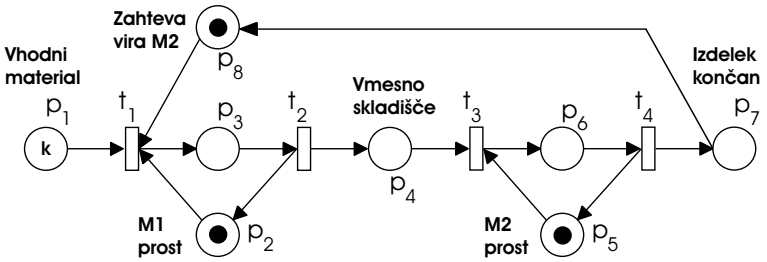
Kako se lotiti modeliranja dveh skrajnih strategij, vodenja po načelu potiska-

nja in vlečenja, s Petrijevimi mrežami, bomo prikazali z naslednjim enostavnim primerom. Predpostavimo, da sta za izdelavo izdelka potrebna dva procesna koraka. Najprej je potrebna obdelava na proizvodnem viru $M1$ in nato še na $M2$. Na sliki 4.2 je predstavljen model, ki ponazarja vodenje po načelu potiskanja. Predpostavljeno je, da je vhodni material na voljo v neomejenih količinah, kar je v danem primeru predstavljeno s k žetoni v mestu p_1. S takim vodenjem se začne proizvajanje na stroju $M1$ ne glede na stanje stroja $M2$. Pri tem se pogostokrat pojavi potreba po večji kapaciteti vmesnega skladišča. V predstavljenem primeru se to zgodi, kadar je proizvodni čas na $M2$ večji kot na $M1$.



Slika 4.2: Modeliranje strategije vodenja po načelu potiskanja s Petrijevo mrežo.

Vodenje po načelu vlečenja, predstavljeno s Petrijevo mrežo, je prikazano na sliki 4.3. V tem primeru se začne proizvodna operacija na viru $M1$ šele, ko dobi zahtevo z naslednje stopnje v proizvodnem procesu; to je proizvodni vir $M2$. V tem primeru je kapaciteta vmesnega skladišča ves čas enaka ena. Proizvodni postopek na viru $M1$ se ne more začeti, preden ne dobi zahteve s strani $M2$, zato mesto p_8 modelira začetni pogoj te zahteve.



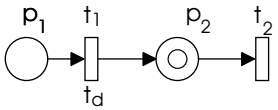
Slika 4.3: Modeliranje strategije vodenja po načelu vlečenja s Petrijevo mrežo.

Modeliranje različnih izvedenk vodenja po načelu vlečenja s pomočjo Petrijevih mrež lahko najdemo v [19, 82]. Poleg osnovne izvedbe modela vodenja po načelu vlečenja (BSCS), je predstavljenih še več izvedenih Kanban sistemov (SKCS, IKCS in IEKCS sistemi vodenja).

4.3.2 Modeliranje proizvodnih aktivnosti

Celoten proizvodni proces lahko razstavimo na osnovne aktivnosti. Na tem mestu bomo predstavili, kako s Petrijevimi mrežami modelirati elemente, ki se kot sestavni deli pojavljajo v proizvodnih sistemih.

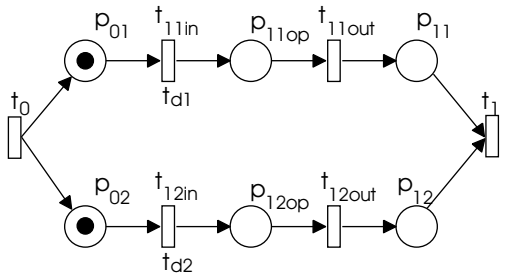
Na operacijo, kot najosnovnejši element proizvodnega sistema, lahko gledamo kot niz dogodkov in aktivnosti. Kot smo dejali, dogodke modeliramo s prehodi in aktivnosti s prisotnostjo žetona v mestu oziroma s proženjem prehoda. Dogodek se lahko zgodi, ko so vsi vhodni pogoji zagotovljeni, t.j. vsa vhodna mesta vsebujejo zadostno število žetonov. Tako lahko osnovno operacijo modeliramo s časovnimi Petrijevimi mrežami, ki so bile predstavljene v 3. poglavju. Operacija je predstavljena z dvema prehodoma in enim mestom. Njen model je predstavljen na sliki 4.4. Mesto p_1, ki sicer ni del operacije, predstavlja le vhodne pogoje. Ko je zadoščeno vsem vhodnim pogojem, t.j. prisotnost materiala, razpoložljivost vira, ipd, nastopi dogodek, ki določa začetek operacije. Ta dogodek je predstavljen s prehodom t_1. Temu prehodu je prirejena tudi časovna oznaka t_d, s katero je določen čas trajanja operacije. Za ta čas je žeton, ki se ob sprožitvi ustvari v mestu p_2, nerazpoložljiv. Torej, ob uporabi te vrste časovnih Petrijevih mrež je aktivnost predstavljena s prisotnostjo nerazpoložljivega žetona v mestu p_2. Ko poteče čas t_d, postane žeton razpoložljiv in pogoj za končanje operacije, ki je modeliran s prehodom t_2, je izpolnjen.



Slika 4.4: Operacija, predstavljena s strukturo časovne Petrijeve mreže.

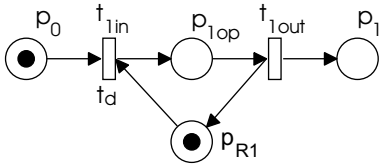
Tudi princip paralelnosti se da enostavno opisati s Petrijevimi mrežami. Model dveh paralelnih operacij je predstavljen na sliki 4.5. Prehod t_0 postavlja vhodne pogoje za paralelno izvajanje operacij. V mestih p_{01} in p_{02} lahko operaciji čakata na razpoložljive proizvodne vire. Začetka vsake izmed vzporednih operacij sta predstavljena s prehodoma t_{11in} in t_{12in}. Ko sta končani obe operaciji, se na njunih izhodnih mestih (p_{11} in p_{12}) pojavita razpoložljiva žetona. Prehod t_1 služi za sinhronizacijo obeh operacij.

Za izvedbo operacije je običajno potreben proizvodni vir, ki ima neko omejeno



Slika 4.5: Dve paralelni operaciji, predstavljeni s časovno Petrijevo mrežo.

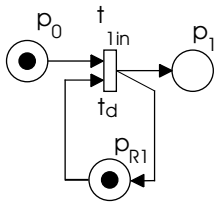
kapaciteto. Na sliki 4.6 je vir modeliran z mestom p_{R1}. Kapaciteta je določena z začetnim številom žetonov v mestu. Vir je razpoložljiv za izvajanje operacije, ko je v njem zadostno število razpoložljivih žetonov. Mesto p_0 v tem primeru označuje zagotovitev ostalih pogojev za začetek operacije, npr. prisotnost vhodnega materiala. Ko sta izpolnjena oba pogoja, je prehod t_{1in} omogočen in operacija se lahko začne izvajati. V mestu p_{1op} se za čas trajanja operacije pojavi nerazpoložljiv žeton, po poteku tega časa pa se s proženjem prehoda t_{1out} operacija zaključi. V mestu p_{R1} se zopet pojavi žeton in vir se lahko uporabi za delo na naslednji operaciji. Prisotnost žetona v mestu p_1, lahko predstavlja (pol)izdelek, ki je bil narejen z modelirano operacijo. Na ta način lahko vir takoj po koncu operacije sprostimo, med tem ko (pol)izdelek čaka v tem vmesnem skladiščem z neomejeno kapaciteto na izvajanje naslednje operacije.



Slika 4.6: Operacija, ki uporablja vir z omejeno kapaciteto.

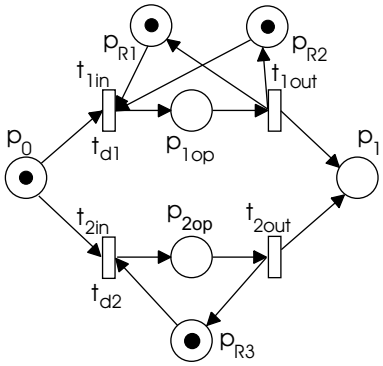
Operacijo z virom pa lahko predstavimo tudi z ekvivalentno strukturo, predstavljeno na sliki 4.7. S prehodom t_{1in} je določen čas trajanja operacije. V tem primeru ne obstaja mesto, kjer bi se nahajal nerazpoložljiv žeton za čas trajanja operacije. Namesto tega se za ta čas pojavita dva nerazpoložljiva žetona, en v mestu, ki predstavlja vir p_{R1}, in drugi v mestu p_1. Po času t_d, postaneta žetona razpoložljiva, t.j. vir je zopet razpoložljiv in (pol)izdelek je pripravljen za nadaljnjo obdelavo. Na ta način dobimo ekvivalenti model operacije, ki je predstavljen v kompaktnejši obliki. To nam sicer poenostavi izračun, ki je potreben pri ana-

lizi modela, po drugi strani pa v primeru, ko imamo opravka s kompleksnejšimi operacijami (predstavljenimi v nadaljevanju), taka predstavitev ne omogoča jasnega prikaza izvajanja operacije. Zato se v pričujočem delu poslužujemo principa modeliranja operacij, kot je to predstavljeno na sliki 4.6.



Slika 4.7: Alternativno modeliranje operacije, ki uporablja vir z omejeno kapaciteto.

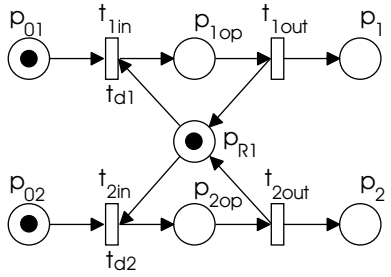
Pogosto se pojavlja situacija, ko operacija za svoje izvajanje potrebuje skupino večih različnih proizvodnih virov. Vsak izmed virov je predstavljen s svojim mestom, vsi pa so povezani s prehodoma, ki začenjata in končujeta operacijo. Poleg tega pa se pogosto pojavlja tudi situacija, ko se lahko lahko operacija izvrši na različnih alternativnih (skupinah) proizvodnih virov. Primer, ko je operacijo mogoče izvršiti na dveh različnih skupinah proizvodnih virov je predstavljen s strukturo prikazano na sliki 4.8. Čas trajanja operacije, je v primeru, da se uporablja vir R_3, določen s prehodom t_{2in}. V drugem primeru, ko pa se izbere skupino virov, sestavljeno iz R_1 in R_2, pa je čas trajanja določen s prehodom t_{1in}.



Slika 4.8: Operacija, ki se lahko izvaja na dveh različnih skupinah virov.

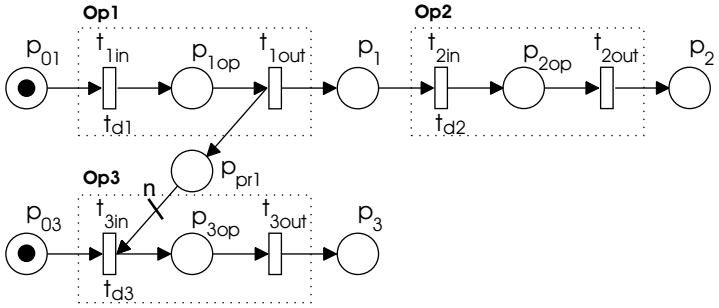
Zelo pogost pa je tudi pojav, ko se za en proizvodni vir poteguje več različnih operacij – deljeni viri. Tak primer bi bil uporaba avtomatsko vodenega vozička (AGV) v proizvodnih sistemih ali pa mešalnega reaktorja v šaržnih sistemih.

Primer, ko za en proizvodni vir konkurirata dve operaciji, je prikazan na sliki 4.9.



Slika 4.9: Deljeni viri.

S prednostnimi omejitvami so določene tehnološke omejitve in pa zaporedje izvajanja operacij. Primer dveh zaporednih operacij je na sliki 4.10 predstavljen z operacijama, označenima kot $Op1$ in $Op2$. Pogosto pa je potrebno upoštevati tudi razne tehnološke omejitve. Na sliki 4.10 je predstavljen primer, kjer se mora izvršiti operacija $Op1$ še pred začetkom operacije $Op3$. V ta namen je med prehodom t_{1out}, ki označuje konec operacije $Op1$, in prehodom t_{3in}, ki označuje začetek operacije $Op3$, dodano mesto p_{pr1}. Utežitev povezave n, ki povezuje mesto p_{pr1} in prehod t_{2in}, določa, koliko komponent se mora izdelati z operacijo $Op1$, preden se lahko prične operacija $Op3$.



Slika 4.10: Prednostne omejitve.

4.3.3 Simulator/razvrščevalnik

Za učinkovito uporabo Petrijevih mrež na področju dela s proizvodnimi sistemi potrebujemo računalniško orodje. V ta namen je bila razširjena funkcionalnost orodja Petri-Net Creator [68]. Pri tem je bilo orodje prilagojeno tudi za delovanje v novejši različici okolja Matlab 7.01 [58].

V Petri-Net Creator je bila vključena tudi možnost dela s časovnimi Petrijevimi mrežami. Za lažje delo pri delu s časovnimi Petrijevimi mrežami, je bilo izpopolnjeno tudi shranjevanje in nalaganje modelov, zgrajenih v Petri-Net Creatorju. S funkcijo *[PN]=getdata()* se iz grafične predstavitve Petrijeve mreže sestavi podatkovna struktura, ki vsebuje vse potrebne informacije o časovni Petrijevi mreži. Podatkovna struktura časovne Petrijeve mreže je določena z naslednjimi elementi:

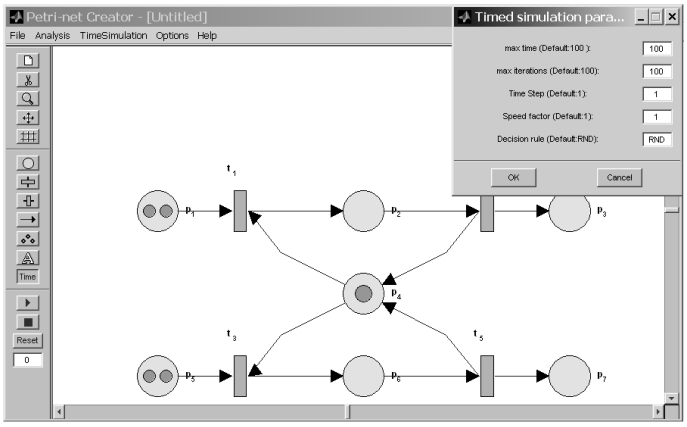
$PN.P$ je množica oznak mest Petrijeve mreže;
$PN.T$ je množica oznak prehodov Petrijeve mreže;
$PN.I$ je funkcija vhodnih povezav;
$PN.O$ je funkcija izhodnih povezav;
$PN.Pxy$ vertikalna in horizontalna pozicija mest;
$PN.Txy$ vertikalna in horizontalna pozicija prehodov;
$PN.Time\_T$ časovne zakasnitve prirejene prehodom;
$PN.M$ stolpni vektor začetne označitve vseh žetonov;
$PN.m$ stolpni vektor začetne označitve razpoložljivih žetonov;
$PN.n$ stolpni vektor začetne označitve nerazpoložljivih žetonov;
$PN.r$ matrika stanja časovnikov, ki vsakemu nerazpoložljivemu žetonu priredi čas držanja.

Funkcija *PlotPN(PN)* na osnovi take podatkovne strukture nariše Petrijevo mrežo.

Vgrajena je bila funkcija, ki omogoča simulacijo take Petrijeve mreže, predstavljena v podpoglavju 4.1.4. V to funkcijo je mogoče vključiti tudi različna hevristična pravila, ki se jih uporabi ob pojavu konfliktnih situacij. Na ta način lahko rešujemo različne probleme razvrščanja. Za predstavitev orodja in delo z nekaterimi praktičnimi primeri so bili implementirani algoritmi naključnega izbiranja, izbiranje po pravilu najkrajšega in najdaljšega časa izvajanja (SPT, LPT). Katero pravilo bomo izbrali, določimo z nastavitvijo simulacijskih parametrov, glej sliko 4.11. Poleg tega, lahko tu nastavljamo še ostale simulacijske parametre, kot so maksimalni čas simulacije, maksimalno število iteracij, časovni korak in hitrost izvajanja simulacije.

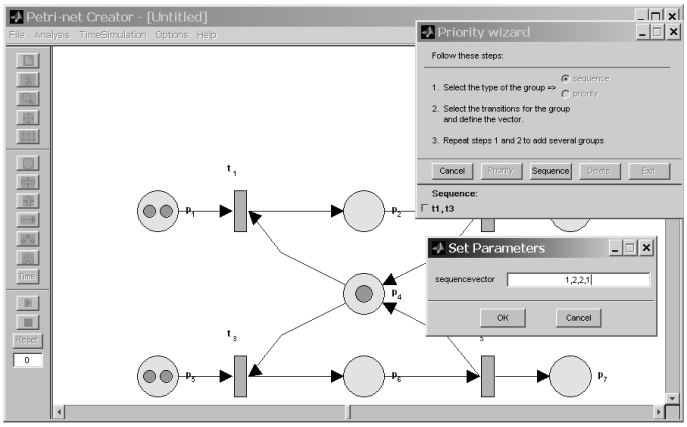
Na tem mestu omenimo še eno pomanjkljivost simulacijske funkcije, ki jo opazimo, ko jo uporabimo za razvrščanje proizvodnih operacij. Ko imamo opravka

z operacijo, ki jo lahko izvršimo na različnih skupinah virov (glej sliko 4.8), se pojavi konfliktna situacija, ki jo razrešimo s pomočjo nekega prioritetnega pravila in se tako odločimo za najugodnejšo skupino virov, vendar le med trenutno prostimi. Pojavi pa se lahko tudi situacija, ko bi bilo bolj smiselno počakati na kako drugo skupino virov, ki bi bila sproščena nekoliko kasneje.



Slika 4.11: Nastavitev simulacijskih parametrov.

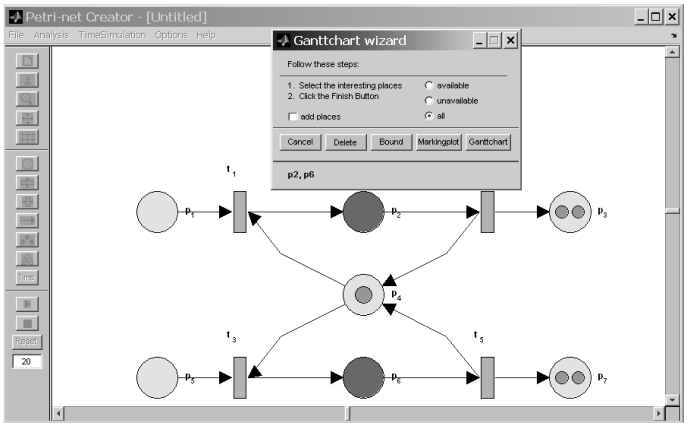
Vgrajena je bila tudi dodatna funkcionalnost, ki nam je v pomoč pri odločanju, kjer lahko prehodom v konfliktu določimo prioritete ali zaporedje izbiranja [58, 59]. Na ta način lahko simulacijsko funkcijo uporabimo znotraj optimizacijskega postopka, za določanje kriterijske funkcije. Na sliki 4.12 je predstavljen primer, ko za izbrana prehoda t_1 in t_3 definiramo, v kakšnem zaporedju se naj izvajata.



Slika 4.12: Nastavitev zaporedja izvajanj ali prioritet.

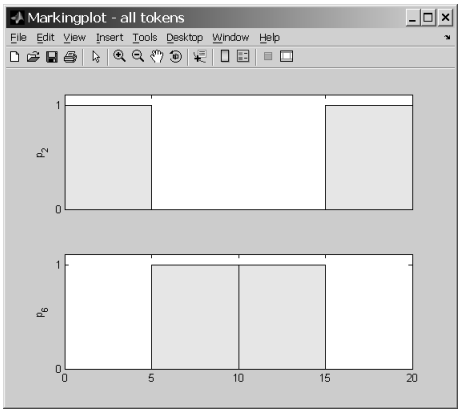
Rezultat simulacije lahko kasneje tudi grafično predstavimo. V ta namen je

vgrajen pripomoček za gradnjo gantogramov. Kot je prikazano na sliki 4.13, najprej izberemo mesta, za katera nas zanima časovni potek označitve.



Slika 4.13: Priprava grafičnega prikaza.

Na sliki 4.14 je predstavljen gantogram za dan primer, ob prej podani zahtevi o zaporedju izvajanja prehodov.



Slika 4.14: Rezultat.

Za omenjeno okolje je bil razvit pripomoček, s katerim je možno avtomatsko modeliranje proizvodnega sistema na osnovi podatkov iz poslovnih sistemov. Princip takega modeliranja bo predstavljen v 5. poglavju.

4.3.4 Primer modeliranja proizvodnega postopka tabletiranja

V nadaljevanju bomo predstavili primer modeliranja enostavnega proizvodnega postopka, ki je sestavljen iz več operacij. Obravnavali bomo proizvodni proces

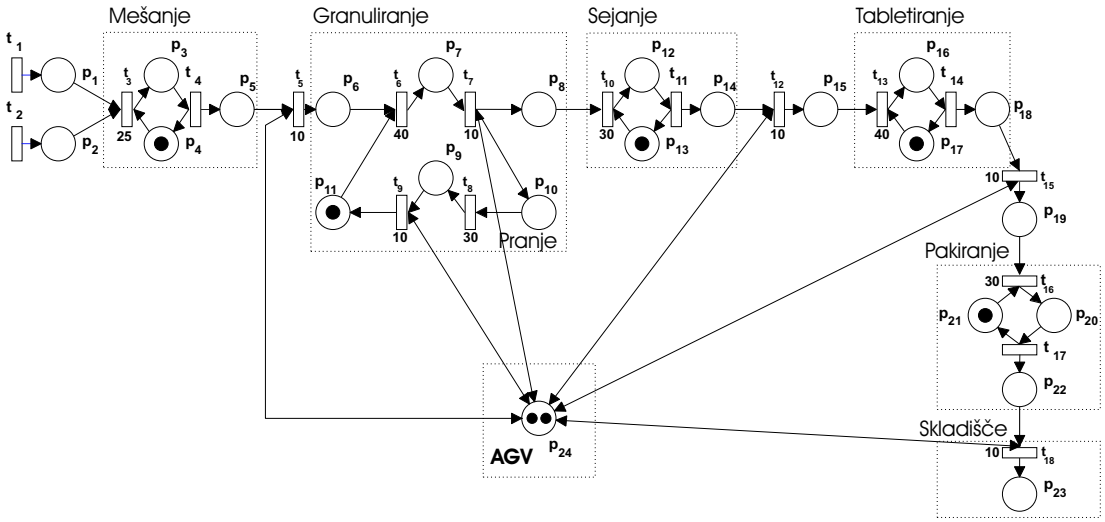
izdelave tablet. Proces je sestavljen iz več postaj: mešalnika, granulatorja, sejalnika, tabletirke, pakirnice in skladišča. Med postajami se material prenaša z avtomatsko vodenimi vozički (AGV).

Vhod procesa nam predstavlja mešalnik, kamor se vnaša natančno odtehtana količina vhodne surovine, za katero je predpostavljeno, da jo je v neomejenih količinah. Ko je mešanje zaključeno, se material z vozičkom prenese na postajo granuliranja. Tu se delci med sabo vežejo z namenom, da se izboljšajo njihove pretočne lastnosti. Po končanem granuliranju je potrebno pranje granulatorja. Na naslednji postaji se surovina preseje, čemur sledi tabletiranje, kjer se surovina stiska v obliko tablet. Tablete se potem zapakira in shrani v skladišču končnih izdelkov.

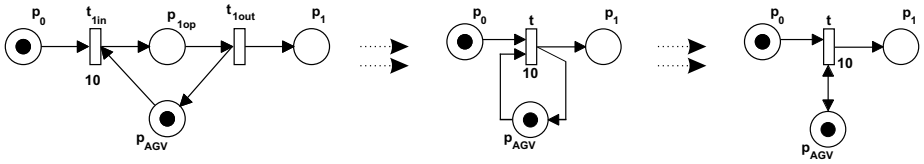
Modeliranje

Proizvodno linijo smo modelirali s časovno Petrijevo mrežo. Rezultat modeliranja je prikazan na sliki 4.15. Vsaka postaja je predstavljen kot operacija z enim proizvodnim virom. Na sliki so posamezne postaje označene z okvirjem. Razpoložljivost posameznih postaj je izražena z mesti p_4 – mešalnik, p_{11} – granulator, p_{13} – sejalnik, p_{17} – tabletirka in p_{21} – pakirnica. Število končnih produktov v skladišču je izraženo s številom žetonov v mestu p_{23}. Med postajami se material prevaža z vozičkom, zato je vsaka postaja povezana z mestom p_{24}, ki modelira ta voziček. Število žetonov v tem mestu določa število razpoložljivih vozičkov. Oznake časov posameznih operacij so podane ob prehodih, ki te operacije modelirajo.

Glede na to, da imamo opravka z dokaj enostavnim problemom, tu uporabimo alternativni princip modeliranja operacij, ki uporabljajo vir z omejeno kapaciteto (AGV), kot je bilo to že prikazano na sliki 4.7. Vsaka operacija, ki jo izvaja avtomatsko voden voziček, je predstavljena s poenostavljeno strukturo, prikazano na sliki 4.16. S prehodom je določen čas, potreben za pretovorjenje materiala. Ko se prehod sproži, se ustvari en nerazpoložljiv žeton v mestu, ki je povezan na naslednjo operacijo in en nerazpoložljiv žeton v mestu, ki definira razpoložljivost vozička. Oba žetona postaneta razpoložljiva po času, kot je to določeno s prehodom.



Slika 4.15: Model proizvodnje tablet.

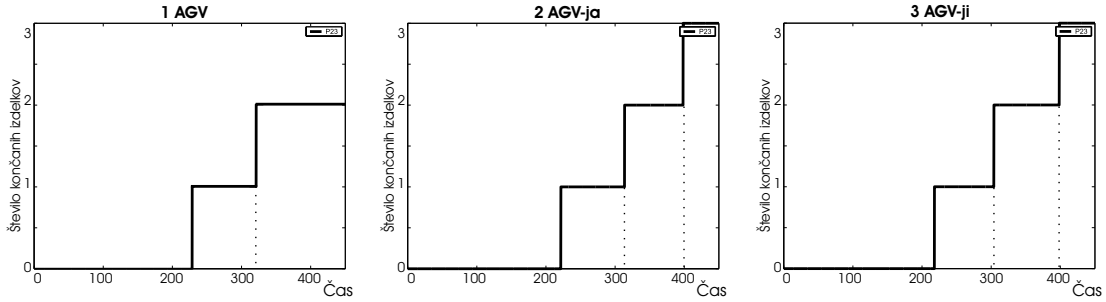


Slika 4.16: Poenostavitev modela vozička.

Analiza

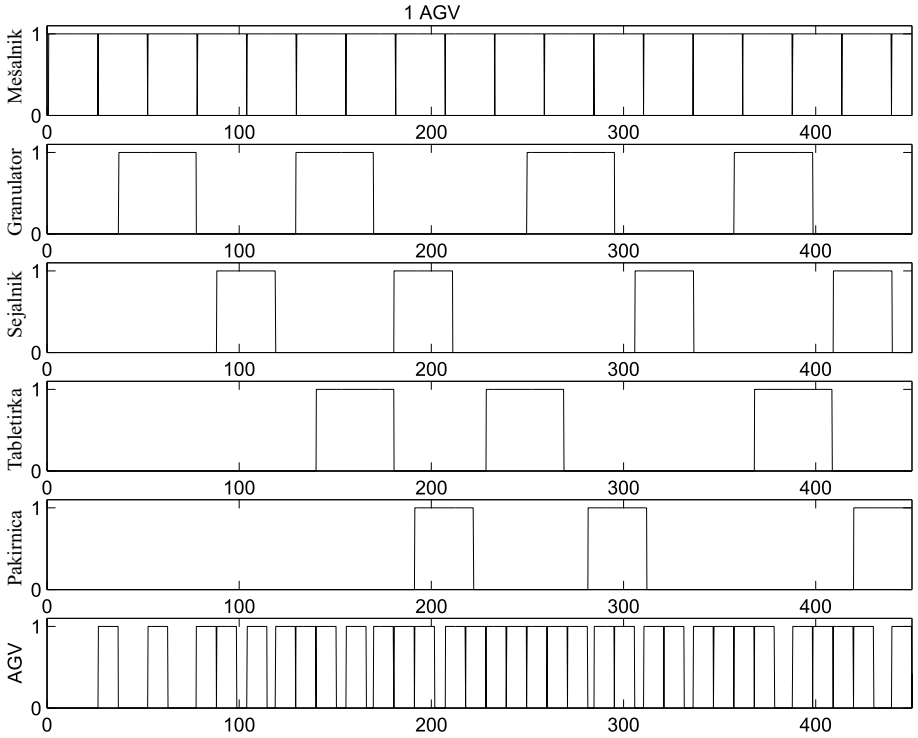
Predpostavljamo, da imamo dano proizvodno linijo, postavlja pa se vprašanje, koliko vozičkov potrebujemo za čimbolj učinkovito izrabo razpoložljive kapacitete dane proizvodne linije. Preveliko število vozičkov močno poveča stroške opreme in povzroča probleme koordinacije med njimi. Po drugi strani pa s premajhnim številom vozičkov razpoložljive kapacitete proizvodne linije ne izkoristimo dovolj.

Učinkovitost predstavljenega procesa, ob uporabi različnega števila vozičkov, analiziramo s pomočjo simulacije. Uporabimo simulacijsko orodje, predstavljeno v podpoglavju 4.1.4. Za grobo oceno o učinkovitosti lahko upoštevamo število izgotovljenih izdelkov v nekem časovnem obdobju. Tako opazujemo število žetonov v mestu p_{23}, ki označuje število končnih izdelkov v skladišču, slika 4.17. Opazimo, da je za neko časovno obdobje, število proizvedenih končnih izdelkov skoraj enako ob uporabi dveh ali treh vozičkov, medtem ko z enim vozičkom v danem obdobju proizvedemo bistveno manj končnih izdelkov. Iz rezultatov lahko sklepamo, da je v dani situaciji najbolj smotrna uporaba dveh vozičkov AGV.



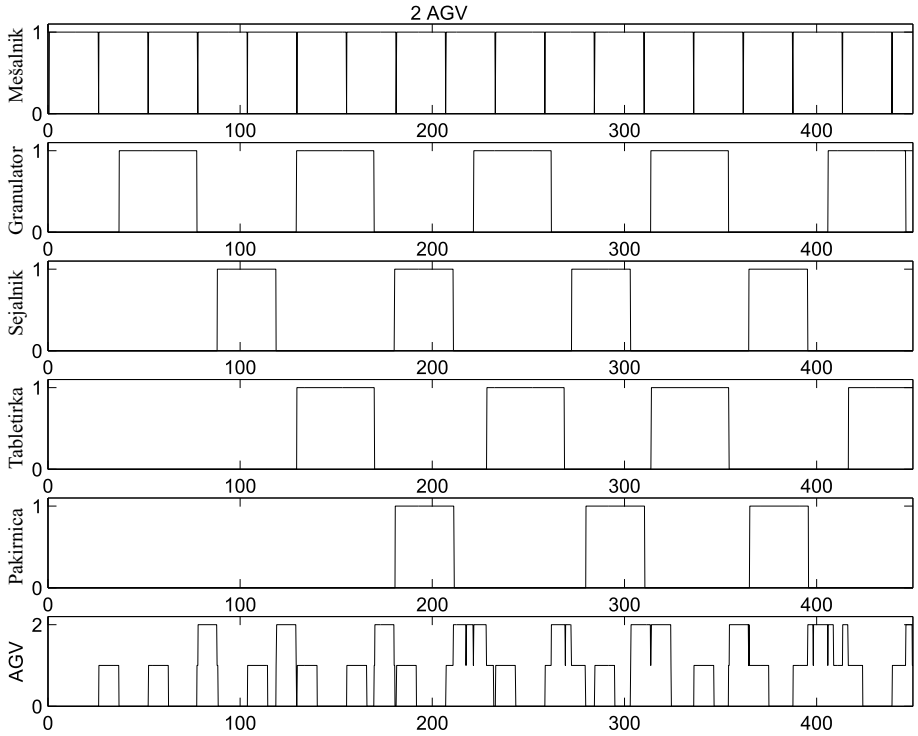
Slika 4.17: Analiza učinkovitosti pri uporabi različnega števila AGV-jev.

S simulacijo modela časovne Petrijeve mreže, ki je predstavljen na sliki 4.15, lahko opazujemo sled označitve skozi čas. V primeru, ko imamo opravka z modeli proizvodnih procesov, bi bilo koristno opazovati tista mesta, ki predstavljajo obremenitev posameznih proizvodnih virov skozi čas. Slike 4.18, 4.19 in 4.20 prikazujejo razvrstitev dela po vseh virih, ki nastopajo v proizvodnem sistemu. Na ta način lahko opazujemo vpliv količine vozičkov na obremenitev tudi vseh ostalih proizvodnih virov.

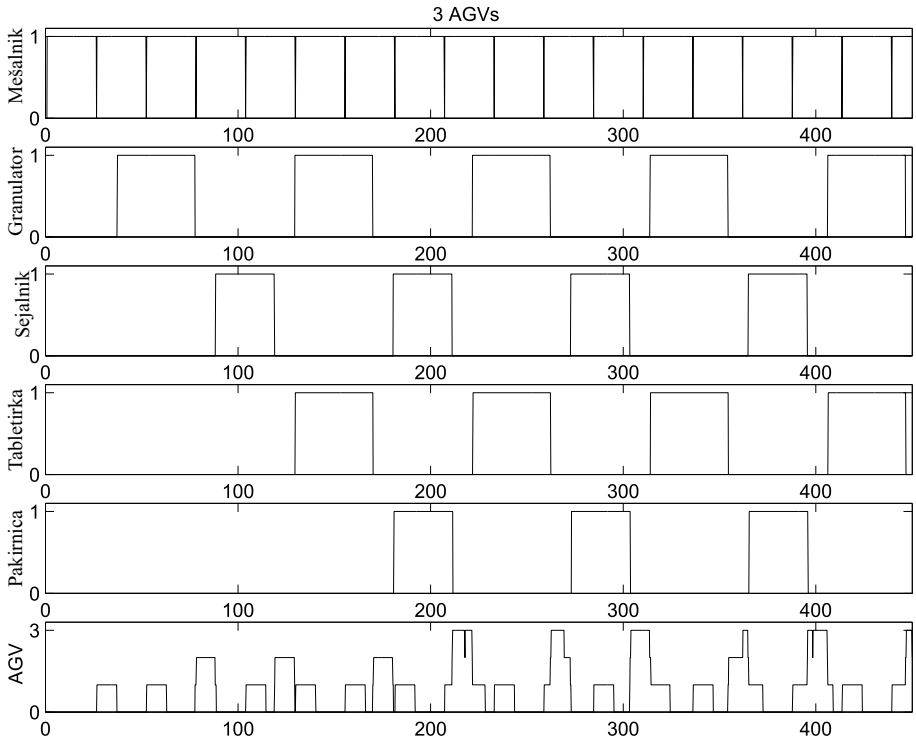


Slika 4.18: Gantogram razvrstitve opravil v proizvodnem postopku tabletiranja ob uporabi enega vozička.

Razvrstitev opravil ob uporabi enega vozička je prikazana na sliki 4.18. Najnižje je predstavljena časovna obremenitev vozička AGV. Iz gantograma lahko opazimo, da je voziček AGV skoraj ves čas zaseden. Po drugi strani pa lahko na sliki 4.19, ki predstavlja razvrstitev opravil ob uporabi dveh vozičkov, opazimo, da večkrat nastopa situacija, ko sta potrebna oba vozička. To kaže na upravičenost uporabe dveh vozičkov. Da pa uporaba treh vozičkov ni smiselna, pa se lahko prepričamo iz gantograma, ki je predstavljen na sliki 4.18. Vidimo lahko, da polna obremenitev vozičkov ne nastopa pogostokrat. To potrjuje že prej ugotovljeno dejstvo, da je najbolj smiselna uporaba dveh vozičkov.



Slika 4.19: Gantogram razvrstitve opravil v proizvodnem postopku tabletiranja ob uporabi dveh vozičkov.



Slika 4.20: Gantogram razvrstitve opravil v proizvodnem postopku tabletiranja ob uporabi treh vozičkov.

5. Računalniško podprto modeliranje z uporabo podatkov iz sistemov za planiranje proizvodnje

V praksi se za planiranje proizvodnje najpogosteje uporabljajo sistemi za planiranje materialnih potreb (*Material Requirement Planning – MRP*). Taki sistemi so običajno označeni kot sistemi, ki delujejo po načelu potiskanja. Kot je definirana taka strategija v [53], so v tem primeru naloge na vhodu sistema postavljene v vrsto. V kakšnem vrstnem redu se bodo naloge izvajale, je določeno z izbranim pravilom razvrščanja.

Sistemi za planiranje (MRP) zagotavljajo materialne potrebe in določajo proizvodne plane, t.j. naloge, ki jih je potrebno izvršiti. Za svoje delovanje potrebujejo informacijo o strukturi izdelka, statusu zalog in strukturi proizvodnega sistema [56, 97]. Struktura izdelka je določena s kosovnico, medtem ko lahko iz proizvodnega postopka razberemo strukturo proizvodnega sistema. Običajno je možno takim sistemom za planiranje pripojiti tudi dodatno orodje za razvrščanje. Prav tako pa tudi ta orodja za svoje delovanje potrebujejo prej omenjene podatke. V tem primeru je smiselno, da oba sistema uporabljata to podatkovno strukturo.

Kis s sod. [48] uporabi dvo-nivojske časovno objektne Petrijeve mreže (*Chameleon system*) za skupno obravnavo planiranja procesov in razvrščanja proizvodnje. Czerwinski [21] v svojem delu za problem iskanja razvrstitve v kompleksnih sistemih vpelje izpopolnjeno Lagrangeovo relaksacijsko metodo. Pri tem pa uporabi zaporedje izdelave posameznih izdelkov in sestavnih delov, ki je podano s kosovnico. Pristop predstavljen v [101] se oskrbuje s proizvodnimi podatki iz kosovnice proizvodnje (*Bill Of Manufacture – BOMfr*, [34]), ki združuje podatke iz kosovnice in proizvodnega postopka. BOMfr določa zaporedje proizvodnih operacij, potrebnih za izgotovitev vmesnih ali končnih izdelkov, skupaj z materialom

in proizvodnimi viri, ki so potrebni pri vsaki operaciji. Proizvodni podatki so nato uporabljeni za določanje proizvodnih nalog, le-te pa se potem razvršča na razpoložljive proizvodne vire.

V sistemu MRP je struktura izdelka podana v obliki kosovnice (*Bill Of Material – BOM*) in struktura procesa v obliki proizvodnega postopka (*Routing*). Kosovnica se uporablja za določanje potreb po surovinah, sestavnih delih in določa zapovrstje, v katerem se obdeluje in sestavlja sestavne dele. Izdelava posameznega sestavnega dela je nadalje sestavljena iz zaporedja operacij. Za izvedbo vsake posamezne operacije je potreben nek proizvodni vir, ki za to potrebuje nek določen čas.

Naše delo sloni na uporabi podatkov o strukturi izdelka in podatkov o strukturi procesa. Ti podatki, skupaj z informacijo o razpoložljivih proizvodnih virih, tvorijo osnovne elemente, potrebne za modeliranje proizvodnega procesa. Petrijeve mreže nam omogočanje modeliranje proizvodnih struktur tako na poslovnem, kot tudi na detajlnem proizvodnem nivoju. V nadaljevanju je predstavljen algoritem, ki upošteva zakonitosti na obeh nivojih in omogoča avtomatsko gradnjo modela v obliki Petrijeve mreže.

5.1 Razred obravnavanih proizvodnih procesov

V nadaljevanju bomo podrobno opredelili razred problemov, ki jih obravnavamo v tem delu. Različne probleme razvrščanja, ki se pojavljajo v proizvodnih okoljih, lahko predstavimo s Petrijevimi mrežami.

V našem primeru preučujemo proizvodne sisteme, ki za svoje delovanje že uporabljajo poslovne sisteme, s katerimi se izvaja planiranje proizvodnje. Obravnavamo problematiko, ko imamo opravka s sistemi za planiranje MRP, ki delujejo po načelu potiskanja. Sistem generira delovne naloge, s katerimi so podane zahteve po izdelavi izdelkov. Kako proizvesti posamezen izdelek, je določeno z nalogo, ki je opisana s proizvodnim postopkom. Vsako nalogo, ki je sestavljena iz več proizvodnih operacij, lahko izvedemo v proizvodnem sistemu ob uporabi različnih skupin proizvodnih virov. Za izdelavo izdelka pa je lahko potrebnih tudi več polizdelkov. Spisek potrebnih polizdelkov je podan s kosovnico. V tem primeru, naloga, ki poskrbi za izdelavo tega polizdelka, predstavlja podnalogo, ki je

potrebna za izdelavo glavnega izdelka. Na ta način dobimo problem razvrščanja opravil v proizvodnji, ki ga lahko predstavimo kot sestav odprte obdelave in ga lahko podamo kot:

- izvesti je potrebno n nalog: $J = \{J_j\}, j = 1, ..., n$,
- na voljo je m proizvodnih virov: $M = \{M_i\}, i = 1, ..., m$,
- vsaka naloga J_i je sestavljena iz n_j operacij: $O_j = \{o_{jk}\}, k = 1, ..., n_j$,
- vsaka operacija za svoje izvajanje potrebuje neko skupino virov $S_{jkl} \in R$; l določa število alternativnih skupin virov,
- čas izvajanja vsake operacije, ki jo izvaja skupina virov S_{jkl}, je določen s T_{jkl},
- s precedenčnimi povezavami ponazorimo, da se mora neka naloga izvršiti pred neko drugo.

V primeru, ko naloga zahteva tudi polizdelke, se za vsak polizdelek razbere, katero nalogo je za to potrebno izvesti. Pri tem pa veljajo tudi določene predpostavke, kot so:

- Proizvodni viri so vedno na voljo in se ne kvarijo.
- Vsak vir lahko sočasno opravlja omejeno število operacij. Omejitev je definirana s kapaciteto vira.
- Prekinjanje operacij ni dovoljeno.
- Ko operacija konča z delom na nekem polizdelku, mora čimprej sprostiti vire. V ta namen so operacije med sabo povezane z vmesnim skladiščem neomejene kapacitete. Pri šaržnih procesih se pojavljajo primeri, ko se mora operacija vršiti na istem viru, ki je bil uporabljen že pri predhodni operaciji. V tem primeru je vir sproščen šele po zadnji operaciji.
- Vsi proizvodni časi so deterministični in znani vnaprej.
- Delovnemu nalogu je poleg količine izdelkov prirejen tudi čas začetka izvajanja. Nalogi, ki so časovno usklajeni, so obravnavani združeno.

Postopek modeliranja in reševanja problema razvrščanja, ki ga predstavljamo, je uporaben za različna proizvodna okolja. V prvi vrsti je predvsem uporaben v kosovni proizvodnji. Poslovni sistemi, ki skrbijo za obratovanje take proizvodnje, običajno delujejo s podatki, kakršni so uporabljeni tudi v našem primeru. Seveda pa ima tak način modeliranja tudi določene omejitve. Modeliramo lahko le vnaprej predvidene situacije, tako je kreativnost gradnje modela precej okrnjena. V primeru, da imamo opravka z drugačno situacijo, kot je predvideno z našim algoritmom, jo poskušamo prilagoditi. V tem primeru je potrebno paziti na to, da nam tako zgrajeni model predstavlja zadovoljivo sliko obravnavanega procesa in je še vedno upravičena njegova uporaba. Tako bi za primer, ko imamo opravka s šaržnim proizvodnim okoljem, potrebovali nekaj manjših prilagoditev v podatkovni strukturi, ki se običajno uporablja pri planiranju proizvodnje.

5.2 Kosovnica

Kosovnica je spisek surovin in elementov, ki sestavljajo izdelek – nadrejeno komponento. Kosovnica je lahko v različnih oblikah [34, 91]. Kosovnica, uporabljena v tem delu, je predstavljena z:

$BOM = (R, E, q, pre)$, kjer velja:

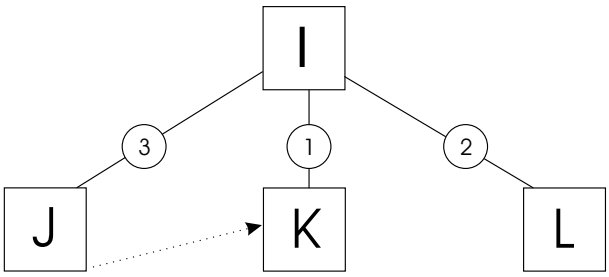
- $R = \{r_1\}$ je nadrejena komponenta.
- $E = \{e_1, ..., e_i\}$ je končna množica podrejenih komponent,
- $q : E \to \mathbb{N}$ je funkcija, ki definira količino posameznih podrejenih komponent e_i. **q** predstavlja $i \times 1$ stolpni vektor, katerega i-ti element je $q(e_i)$.
- $pre : (E \times E) \to \{0, 1\}$ je prioritetna funkcija. Določa prioritetno matriko **pre**, kjer $\mathbf{pre}(i, j) = 1$ označuje, da ima i-ta komponenta prednost pred j-to. Matrika je lahko predstavljena tudi z usmerjenim grafom.

R je nadrejena komponenta in predstavlja izdelek, ki je sestavljen iz podrejenih komponent $e_i \in E$. Število potrebnih podrejenih komponent je določeno z vektorjem **q**. V primeru, ko mora biti neka komponenta izvedena pred drugo, je to določeno s prioritetno matriko **pre**. Kosovnica, ki jo uporabljamo v našem primeru, zahteva, da se izdelajo vse predhodne komponente, preden se lahko začne

izdelava naslednje komponente. Za matriko **pre** velja, da so vrednosti njenih diagonalnih elementov enake nič, t.j. komponenta ne more imeti prednosti sama pred sabo. Velja tudi, da nikakršni cikli v prioritetni matriki niso dovoljeni, saj le-ti v povezavi s kosovnicami nimajo smisla. Prisotnost ciklov bi sicer lahko preverjali z analizo pripadajočega usmerjenega grafa.

V primeru, ko je katera izmed podrejenih komponent sestavljena iz večih podrejenih komponent, se uporablja enaka definicija za opis njihove medsebojne odvisnosti. Komponente na najnižji stopnji gradnje (surovine) nimajo kosovnic.

Kosovnica je v praksi pogosto predstavljena v grafični obliki (slika 5.1), kot sestavno drevo [99] ali v tabelarični obliki (tabela 5.1). V prikazanem primeru kosovnica opisuje zgradbo komponente I, ki je sestavljena iz treh podrejenih komponent, J, K in L. Pri tem velja omejitev, ki določa, da morajo biti izdelane najprej vse komponente J, preden se začne proizvajanje komponente K. V grafični predstavitvi so take omejitve ponazorjene s črtkasto črto, ki v danem primeru povezuje komponento J z K. Omejitve so jasno podane tudi v tabelarični predstavitvi z matriko prednostnih povezav.



Slika 5.1: Grafična predstavitev kosovnice.

Tabela 5.1: Tabelarična predstavitev kosovnice

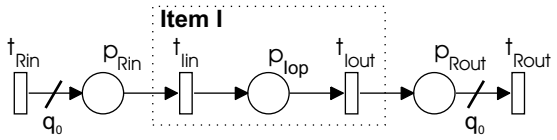
Nadr. komp.	Podr. komp.	Količina	Prednostne povezave		
I	J	3	0	1	0
	K	1	0	0	0
	L	2	0	0	0

Predstavljen primer kosovnice lahko matematično opišemo na naslednji način:

$$BOM = (R, E, \mathbf{q}, \mathbf{pre}), \text{ kjer so } R = \left\{ I \right\},$$

$$E = \left\{ J \quad K \quad L \right\}, \quad \mathbf{q} = \begin{bmatrix} 3 & 1 & 2 \end{bmatrix} \quad \text{in} \quad \mathbf{pre} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Predpostavimo, da vsaka komponenta v kosovnici predstavlja eno operacijo. Kot je definirano že predhodno, lahko operacijo predstavimo s Petrijevo mrežo – z dvema prehodoma in enim mestom, slika 4.4. Pred operacijo sta dodana še prehod t_{Rin} in mesto p_{Rin} ter za operacijo p_{Rout} in t_{Rout}, kar nam omogoča, da z uteženo povezavo (q_0) predpišemo količino potrebnih komponent. Komponenta, predstavljena s strukturo Petrijeve mreže, je prikazana na sliki 5.2.



Slika 5.2: Predstavitev komponente s strukturo Petrijeve mreže.

Končni izdelek je določen s strukturo večih kosovnic. Tako je gradnja celotne Petrijeve mreže iterativni postopek, ki začne z izvorno kosovnico in nadaljuje, dokler niso bile upoštevane vse podrejene komponente. V primeru, ko je komponenta sestavljena iz podrejenih komponent, se uokvirjen del Petrijeve mreže odstrani in nadomesti s strukturo, ki predstavlja relacije med podrejenimi komponentami. Kadar je podrejenih komponent več, se uporabi struktura paralelnih aktivnosti, slika 4.5.

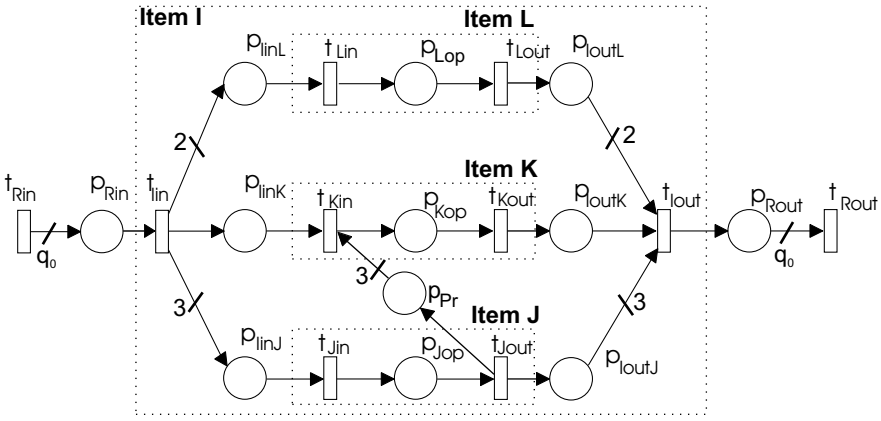
Substitucija komponente s podrejenimi komponentami je določena z naslednjimi koraki:

1. Odstrani se mesto p_{Xop} in pripadajoči vh/izh povezavi.

2. Definira se Petrijeva mreža za podrejene komponente, kot je to določeno s kosovnico. Pri tem se upošteva tudi prednostne povezave.

3. Odstranjeno mesto p_{Xop} je nadomeščeno z zgrajeno strukturo, pri čemer se vhodni in izhodni prehodi zlijejo z obstoječimi. Model kosovnice se vstavi v glavno strukturo Petrijeve mreže.

Rezultat gradnje Petrijeve mreže za obravnavano kosovnico je prikazan na sliki 5.3.



Slika 5.3: Kosovnica predstavljena s strukturo Petrijeve mreže.

5.3 Proizvodni postopek

Za vsako komponento, ki se nahaja v proizvodnem procesu in ne predstavlja vhodne surovine, je podan proizvodni postopek. Proizvodni postopek je postopek transformacije vhoda v izhod in je podan kot zaporedje delovnih operacij. V njem je navedeno tehnološko zaporedje delovnih operacij, procesni časi, potrebni za izvedbo teh operacij in proizvodni viri, na katerih se lahko operacije izvedejo.

Proizvodni postopek je v praksi običajno podan v obliki tabele (*Routing table*). Tabela vsebuje glavo, ki opredeljuje komponento, ki se izdeluje, in vrstice, kjer so opisane delovne operacije, potrebne za izdelavo. Za vsako operacijo obstaja ena vrstica.

Proizvodni postopek za komponento J je podan s tabelo 5.2. V danem primeru sta potrebni dve operaciji za izvedbo komponente. Prva operacija $Op10$ se lahko izvaja na dveh različnih virih, pri čemer vsak izmed virov potrebuje različen procesni čas. Druga operacija pa zahteva dve skupini virov, R_1 in tri vire R_3.

Substitucija ene komponente kosovnice s proizvodnim postopkom je določena z naslednjimi koraki:

1. Odstrani se mesto p_{Xop} in pripadajoči vh/izh povezavi,

Tabela 5.2: Proizvodni postopek za komponento J

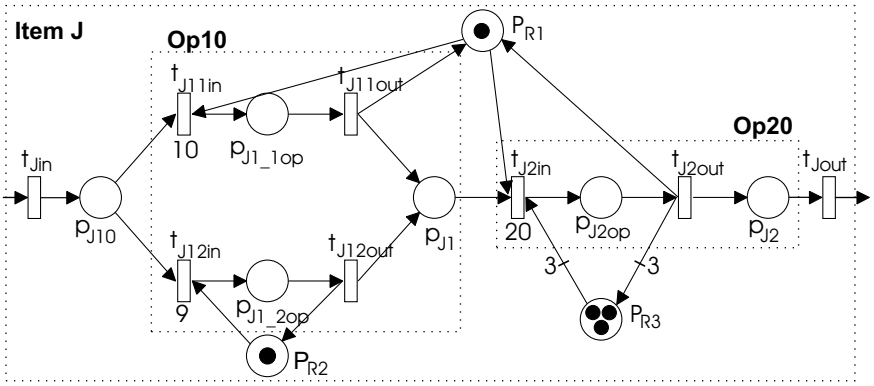
	Operacija	Proc. čas	Proizv. viri
J	Op10	10s/9s	R1/R2
	Op20	20s	R1, 3R3

2. Definira se Petrijeva mreža, kot je to določeno s proizvodnim postopkom.

3. Odstranjeno mesto p_{Xop} je nadomeščeno z zgrajeno strukturo, pri čemer se vhodni in izhodni prehodi zlijejo z obstoječimi. Model proizvodnega postopka se vstavi v glavno strukturo Petrijeve mreže.

Vsak proizvodni vir, ki nastopa v proizvodnem postopku, je predstavljen z mestom. Kapaciteta vira je določena z začetnim številom žetonov v mestu.

V drugem koraku, ko se kreira model proizvodnega postopka, se najprej razpozna operacije iz tabele, ki opredeljuje proizvodni postopek, in se nato za vsako operacijo posebej določi njen model s časovno Petrijevo mrežo, kot je to predstavljeno v razdelku 4.3.2. Vsaka operacija je povezana z mesti, ki predstavljajo proizvodne vire, potrebne za izvedbo teh operacij.



Slika 5.4: Petrijeva mreža proizvodnega postopka za komponento J.

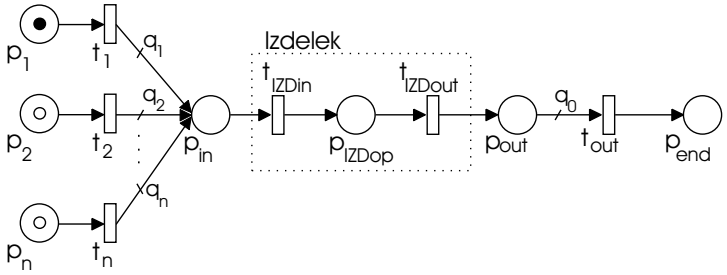
Petrijeva mreža, prikazana na sliki 5.4, je rezultat modeliranja operacij podanih s tabelo 5.2. Predstavljen model je nato vstavljen v glavno Petrijevo mrežo (slika 5.3), kjer nadomesti mesto p_{Jop}, ki predstavlja komponento J.

Proizvodni postopki so podmodeli, ki so vstavljeni v strukturo, določeno s

kosovnico. Možno pa je tudi, da so posamezne aktivnosti v proizvodnem postopku zopet določene s kosovnico.

5.4 Delovni nalog

Z delovnim nalogom (DN) je podano, koliko in katere izdelke je potrebno izdelati. Običajno se obdeluje izdelke v večjih skupinah, ki jim pravimo šarže. Na vsak delovni nalog lahko gledamo kot eno komponento (izdelek), ki ga je potrebno izdelati. Tako ga lahko z mrežo predstavimo, kot smo to že prikazali s sliko 5.2, le da je v tem primeru potrebno dodati še po eno mesto na začetku in na koncu modela, ki označujeta začetek in konec dela. q_0 pri tem določa število šarž, ki jih je potrebno izdelati. Da pa lahko obravnavamo tudi probleme, ko se časi začetka izvajanja za različne količine izdelkov med sabo razlikujejo, vpeljemo splošno strukturo, ki je predstavljena na sliki 5.5. Zahteve za šarže izdelkov, ki se naj začnejo sočasno, združimo in za vsako tako skupino začetni čas modeliramo z mesti $p_1, p_2, ..., p_n$ in žetoni, ki se nahajajo v njih. Z vrednostjo časovnika, ki je prirejen nerazpoložljivemu žetonu, je določeno, kdaj se naj šarže začno izvajati. Z utežmi $q_1, q_2, ..., q_n$ določimo število šarž, ki se naj začno proizvajati ob posameznih časovnih trenutkih. Pri tem velja, da je vsota $q_1 + q_2 + ... + q_n$ enaka številu vseh šarž q_0, ki jih je potrebno izdelati za obravnavan delovni nalog. Prisotnost žetona v mestu p_{end} označuje, da je delovni nalog zaključen.



Slika 5.5: Petrijeva mreža delovnega naloga.

5.5 Postopek gradnje modela

Torej z delovnim nalogom je podano, katere izdelke je potrebno izdelati. Za vsak delovni nalog se najprej določi model Petrijeve mreže. V nadaljevanju se potem za vsak izdelek naknadno vključuje dodatne podrobnosti proizvodnega postopka. Procedura modeliranja je podana z algoritmom 5.1.

Algoritem 5.1

[R, q, st] = *readWO*()
For i = 1 **to** *length*(R)
    E = *readBOM*(R(i))
    PN = *placePN*(R(i), E, q(i), [ ], st(i), x0, y0)
    PN = *routing*(PN, R(i))
end

Najprej se prebere podatke iz delovnega naloga (funkcija *readWO()*). Nalogi, ki jih je potrebno narediti, so dani z R, vektor **q** podaja količino posameznih izdelkov in vektor **st** določa začetni čas izvajanja posameznega izdelka. Vsak izdelek predstavlja eno komponento, za katero se s funkcijo *placePN()* zgradi Petrijeva mreža, predstavljena na sliki 5.2. Korak, kjer se kliče funkcijo *routing*(), je bolj podrobno opisan z algoritmom 5.2:

Algoritem 5.2

function [PN] = *routing*(PN, R)
[E, q, pre] = *readBOM*(R)
datRoute = *readRouting*(R)
for i = 1 **to** *length*(datRoute.Op)
    if datRoute.Resources ==BOM
        PN1 = *placePN*(R, E, q, pre, [ ])
        PN = *insertPN*(PN, PN1)
        for j = 1 **to** *length*(E)
            PN1 = *routing*(PN1, E(j))
        end

```
    else
        PN = constructPN(datRoute(i))
        PN = insertPN(PN, PN1)
    end
end
```

Najprej se iz podatkovne baze prebere podatke o kosovnici in proizvodnem postopku (funkciji *readBOM()* in *readRouting()*). Za vsako operacijo iz proizvodnega postopka algoritem preveri, če je narejena iz podrejenih komponent. V tem primeru se uporabi funkcija *placePN()* in se na ta način zgradi model Petrijeve mreže, glede na kosovnico za to komponento. Če je potrebno, se doda še prednostne povezave. Rezultirajočo mrežo se s pomočjo funkcije *insertPN()* vstavi v glavni model. V primeru, da operacija predstavlja proizvodno operacijo, se kliče funkcijo *constructPN*(). Tu se razpozna osnovne aktivnosti in se zgradi njihov model. Skupaj združene operacije predstavljajo proizvodni postopek, ki je nato vstavljen v glavni model. Vsi podatki o potrebnih virih in proizvodnih časih so pridobljeni iz tabele, ki podaja proizvodni postopek.

Ko je model za izdelavo posameznega izdelka zgrajen, se preveri, če rezultirajoča Petrijeva mreža res podaja model, kot je to zahtevano s podanimi podatki o proizvodnem procesu. Določene lastnosti tako zgrajenega modela lahko preverimo s pomočjo analize P-invariant. Velja, da se pojavi več P-invariant. Njihovo število je določeno z vsoto vseh virov, proizvodnih poti izdelka in prednostnih povezav, ki nastopajo v modelu. Za vsako P-invarianto, ki je posledica pojavljanja proizvodnega vira, velja da je njej pripadajoča utežena vsota žetonov vedno enaka kapaciteti vira. Pri ostalih invariantah pa velja, da je ta vsota določena s skupnim številom zahtevanih šarž izdelkov. Če je možno, se model naknadno poenostavi tako, da odstranjevanje vozlišč ne vpliva na obnašanje modela.

Na ta način dobimo model proizvodnega sistema, ki deluje po principu potiskanja in nam omogoča detajlno razvrščanje proizvodnih opravil v danem proizvodnem sistemu.

5.6 Izvedba algoritma v okolju Matlab

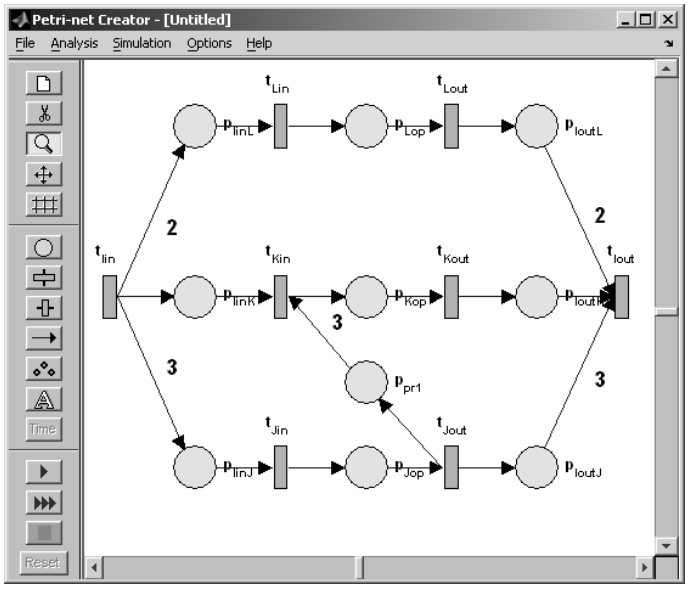
Algoritem, predstavljen v prejšnjem poglavju, je bil implementiran v okolju Matlab. Zgrajene so bile funkcije, ki določajo strukturo Petrijeve mreže, glede na podatke podane s kosovnicami in proizvodnimi postopki. Omenjene funkcije so prirejene za delovanje z orodjem Petri-Net Creator. Da lahko zgrajeno strukturo predstavimo v orodju Petri-Net Creator, moramo vsakemu vozlišču, ki nastopa v Petrijevi mreži, dodati tudi informacijo o poziciji.

Algoritem 5.1 je izveden s funkcijo *PN = algoritem1()* in algoritem 5.2 s *PN = routingPN(PN, R)*. Za delovanje omenjenih algoritmov je potrebnih več dodatnih funkcij, ki jih bomo predstavili v nadaljevanju.

Funkcija *[PN, E] = placePN(R, E, q, pre, st, x0, y0)* je uporabljena za generiranje modela delovnega naloga ali kosovnice s Petrijevo mrežo v okolju Matlab, odvisno od vhodnih podatkov. V primeru, ko se uporabi funkcijo za generiranje začetnega modela, ki predstavlja delovni nalog, so potrebni podatki o tem, za kater delovni nalog gre ('R'), kateri izdelek je s tem delovnim nalogom zahtevan in koliko teh izdelkov je potrebnih. S parametrom 'st' je določen začetni čas izvajanja tega delovnega naloga. Z 'x0' in 'y0' podamo začetne koordinate modela, ki jih potrebuje Petri-net Creator. Podobno pa lahko funkcijo uporabimo tudi za modeliranje kosovnice. Pri tem so potrebni tudi podatki o prednostnih povezavah 'pre', medtem ko parametra 'st' tu ne potrebujemo. Za primer poglejmo kosovnico komponente 'I', ki je sestavljena iz treh podrejenih komponent, med katerimi obstaja tudi prednostna povezava. S pomočjo funkcije *placePN()* je bila omenjena kosovnica modelirana v obliki Petrijeve mreže. Poleg strukture mreže nam omenjena funkcija vrne tudi podatke o podrejenih komponentah E. Model kosovnice, vstavljen v orodje Petri-Net Creator, je predstavljen na sliki 5.6.

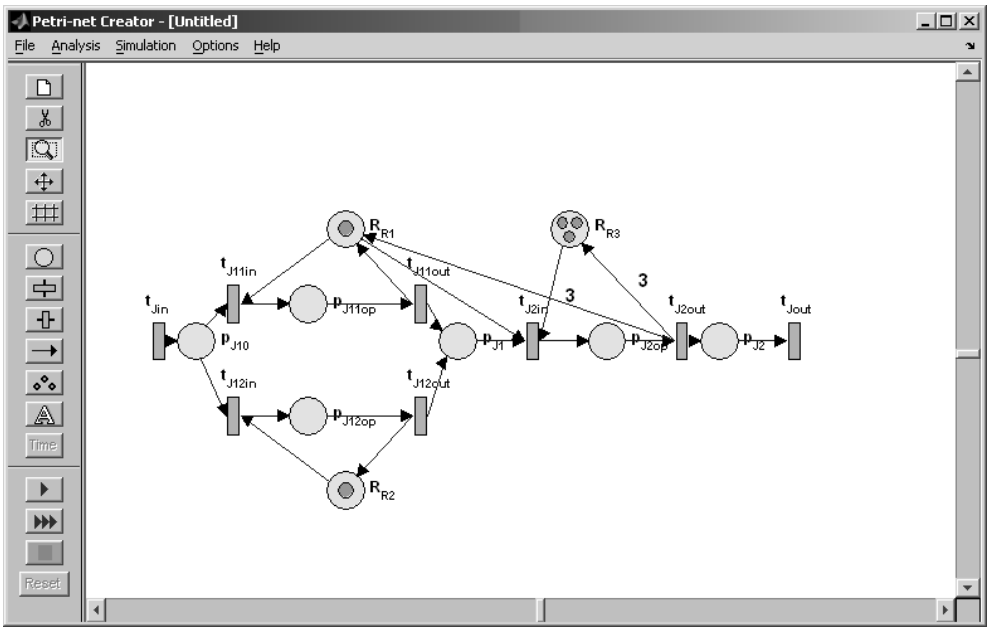
Druga pomembna funkcija, ki nastopa v predstavljenem algoritmu modeliranja, je funkcija, ki modelira proizvodni postopek: *[PN] = constructPN(datRoute)*. Funkcija na osnovi podatkov o proizvodnem postopku *datRoute* za komponento R zgradi model v obliki Petrijeve mreže. Primer modela ene komponente (naloge), sestavljene iz dveh operacij, je predstavljen na sliki 5.7.

Pri uporabi funkcije *constructPN()* se za vsako operacijo modelirajo potrebni viri, ne glede na to, če je bil ta vir modeliran že predhodno. Tako se lahko zgodi,



Slika 5.6: Modeliranje kosovnice.

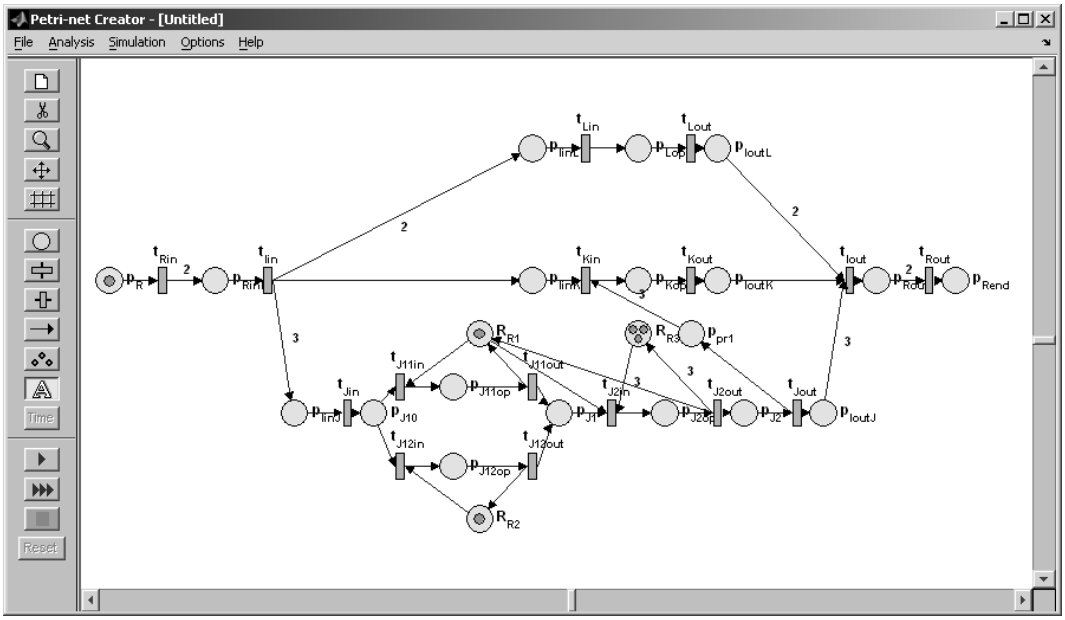
da se posamezen vir v modelu pojavi večkrat. Za detekcijo in rešitev omenjenega problema se uporabi funkcijo: *PN = PopraviR(PN)*.



Slika 5.7: Modeliranje proizvodnega postopka.

Podrejene komponente in proizvodni postopki nam predstavljajo podmodele, ki jih je potrebno vstaviti na ustrezno mesto v modelu nadrejene komponente. V

ta namen je bila zgrajena funkcija *[PN] = insertPN(PN1, PN2, x)*. Parameter *PN1* predstavlja glavni model Petrijeve mreže, v katerega vstavljamo podmodel (*PN2*). Mesto, ki ga želimo nadomestiti s podmodelom *PN2*, je določeno z zaporedno številko x. Pozicije elementov, ki nastopajo v strukturi Petrijeve mreže, se pri vrivanju podmodela smiselno spremenijo. Funkcija vrne model rezultirajoče Petrijeve mreže *PN*. Posnetek delno zgrajenega modela, potem ko je bilo nekaj podmodelov že vstavljenih v glavni model Petrijeve mreže, je predstavljen na sliki 5.8.



Slika 5.8: Vstavljanje podrejenih v nadrejene komponente.

Podatki, ki jih uporablja predstavljen postopek modeliranja, se nahajajo v več različnih datotekah. Funkcija *readBOM* vrne podatke o kosovnicah. Podatke o posamezni komponenti R dobimo s klicem funkcije *[E, q, pre] = readBOM(R)*. Tu nam E predstavlja množico podrejenih komponent e_i, q predstavlja vektor, ki določa potrebno število podrejenih komponent e_i in *pre* vrne prioritetno matriko. S funkcijo *[Op, d, r] = readRouting(R)* pridemo do proizvodnih podatkov. Tu *Op* predstavlja zaporedno številko operacije, d čas trajanja operacije in r potrebne vire za izvedbo operacije. Delovni nalogi so podani preko funkcije *[R, q, st] = readWO()* in na ta način dobimo zahteve po izdelkih, ki jih je potrebno narediti (R), koliko jih je potrebno (q) in neobvezen podatek, ki govori o tem, kdaj se naj izdelek začne izvajati (st).

Tako zgrajen model lahko enostavno analiziramo znotraj orodja Petri-Net Creator. S funkcijo *PlotPN(PN)* lahko model Petrijeve mreže predstavimo grafično. Kot že omenjeno, lahko Petri-Net Creator uporabljamo tudi kot razvrščevalnik, ki že ima vgrajena različna pravila razvrščanja. S funkcionalnostmi, predstavljenimi v tem poglavju, pa je mogoče razvrščevalnik integrirati tudi v že obstoječe informacijske sisteme. S pomočjo Petri-Net Creatorja lahko izvajanje modela, predstavljenim s časovnimi Petrijevimi mrežami, spremljamo tudi v grafični obliki. Seveda pa zahteva tak način izvajanja več računalniškega časa, tako tak način ni uporaben za analizo večjih modelov.

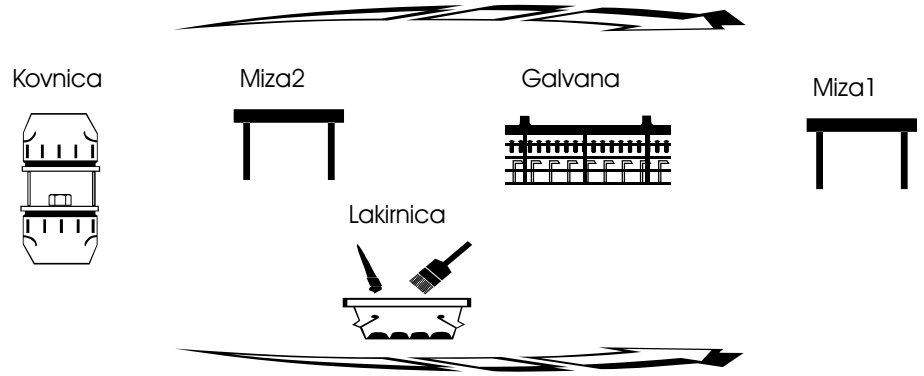
Pri obravnavi kompleksnih sistemov postane model nepregleden, ko ga želimo predstaviti v okolju Petri-net Creator. V tem primeru zgrajen model analiziramo brez prikaza njegove Petrijeve mreže in opazujemo le časovni potek označitve interesantnih mest, npr. mesta, ki predstavljajo proizvodne vire.

5.7 Primer 1: Kosovna proizvodnja

V nadaljevanju bomo obravnavali primer problematike razvrščanja nalog v kosovni proizvodnji. Gre za proizvodni obrat, kjer se proizvaja množica različnih okovij. Proizvodni postopki posameznih izdelkov (okovij) so do določene stopnje izdelave enaki in se kasneje kombinirajo v različne končne izdelke. Proizvodnja je razdeljena na več proizvodnih oddelkov, skozi katere poteka proizvodni proces. Izdelki enega proizvodnega oddelka se shranjujejo v vmesnih skladiščih in so obravnavani kot vhodni material v oddelku, ki sledi. Obstoječa informacijska tehnologija v podjetju zaobsega poslovni sistem (sistem *SAP*), ki se uporablja v podporo planiranju proizvodnje, obstaja pa tudi sistem za spremljanje proizvodnje. Za detajlno razvrščanje nalog po obstoječih proizvodnih virih pa bi bil potreben tudi sistem za razvrščanje. Podatke, ki so potrebni za razvrščanje, lahko pridobimo iz že obstoječih informacijskih sistemov.

Primer, ki ga bomo obravnavali, predstavlja le del proizvodnje okovja. Na poenostavljenem proizvodnem procesu bomo demonstrirali postopek modeliranja proizvodnega procesa, ki smo ga predstavili v prejšnjem podpoglavju. Proizvodni sistem nam predstavlja več proizvodnih virov, ki so prikazani na sliki 5.9. Tako se proizvodni postopek vsakega izdelka začne v kovnici in se običajno nadaljuje

preko postopkov sestavljanja, barvanja, galvanizacije in se zaključi na končni stopnji montaže in pakiranja.



Slika 5.9: Proizvodni sistem.

Končni izdelek enega oddelka predstavlja vhodni izdelek na drugem oddelku. Tako je naloga razvrščanja zagotavljanje časovne usklajenosti delovnih nalogov in s tem zagotavljanje zadostne količine polizdelkov na vhodih posameznih oddelkov proizvodnje. Tako dobljena razvrstitev je izvršljiva. Z uporabo hevrističnih pravil pa lahko zagotovimo razvrstitev, ki je do neke mere optimalna glede na vnaprej določene kriterije delovanja proizvodnje.

Podatki, ki jih potrebujemo za izgradnjo modela, so na voljo v sistemu za planiranje proizvodnje. To so delovni nalogi, kosovnice in proizvodni postopki.

Preko delovnih nalogov so podane zahteve za izdelavo različnih končnih izdelkov. Iz delovnega naloga lahko razberemo še potrebno količino izdelkov in čas začetka izvajanja naloga. Za naš primer so zahteve podane s tabelo 5.3.

Vsak izdelek je sestavljen iz ene ali večih komponent. Strukturo vseh izdelkov lahko razberemo iz kosovnice, ki je predstavljena v tabeli 5.4. Ko med posameznimi komponentami ni nobene prednostne povezave, so vsi elementi prioritetne matrike enaki nič. V tem primeru matriko enostavno predstavimo z zapisom [0].

Za izdelavo vsake komponente, ki nastopa v kosovnici, so potrebni različni proizvodni koraki. Vsi ti koraki so podani s proizvodnim postopkom in so predstavljeni s tabelami 5.5, 5.6 in 5.7. Opis podatkov v tabeli je podrobneje podan med opisom postopka modeliranja.

Tabela 5.3: Delovni nalogi za zahtevane izdelke

Izdelek	Koda	Količina	Začetni čas
Vogalna spona L	CL	2	0
Vogalna spona L	CL	2	80
Vogalna spona D	CR	2	0
Nosilec spone	CH	2	0
Kotnik	AB	2	0

Tabela 5.4: Kosovnice posameznih izdelkov

Nadr. komp.	Podr. komp.	Količina	Prednostne povezave
BOM_CCL	AngBL (A)	1	
	Spona (C)	1	
	Matica (N)	1	[0]
	Vijak (S)	1	
BOM_CCR	AngBR (A)	1	
	Spona (C)	1	
	Matica (N)	1	[0]
	Vijak (S)	1	
BOM_ABL	AngB (B)	1	0 1
	Nosilec (H)	1	0 0
BOM_ABR	AngB (B)	1	0 1
	Nosilec (H)	1	0 0
BOM_HC	NosilecC (HC)	1	
	Opornik (P)	1	[0]
	VijakC (S)	1	

5.7.1 Modeliranje proizvodnje okovja

Model Petrijeve mreže je zgrajen s podatki iz kosovnic in proizvodnih postopkov. Postopek gradnje modela bo prikazan na delu proizvodnega postopka, kjer se

Tabela 5.5: Proizvodni postopki izdelka 'Vogalna spona L'.

	Operacija	Proc. čas	Proizv. viri
CL	Op1	–	BOM_CCL
	Op2	10	Miza1
AngBL	Op11	–	BOM_ABL
	Op12	10	Miza2
	Op13	20	Galvana1
Spona	Op21	3	V1
	Op22	20	Galvana2
Matica	Op31	2	V1
	Op32	20	Galvana2
Vijak	Op41	20	Galvana2
AngB	Op111	8	V1
Nosilec	Op221	8	V2

Tabela 5.6: Proizvodni postopki izdelka 'Nosilec spone'.

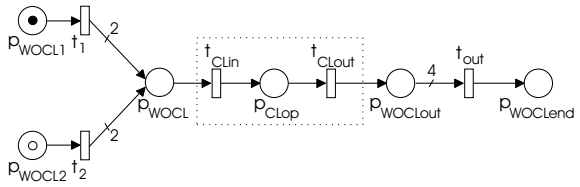
	Operacija	Proc. čas	Proizv. viri
CH	Op1	–	BOM_HC
	Op2	5	Miza1
NosilecC	Op11	15	Galvana2
	Op12	10	Lakirnica
Opornik	Op21	10	Lakirnica
VijakC	Op31	15	Galvana2

proizvaja levi tip vogalne spone ('CL'). Kot lahko opazimo iz tabele 5.3, obstajata za dan izdelek dva delovna naloga, izmed katerih se prvi začne izvajati takoj, medtem ko se naslednji začne izvajati 80 časovnih enot kasneje. Ko uporabimo prvi korak predstavljenega algoritma, dobimo strukturo Petrijeve mreže, ki je prikazana na sliki 5.10. Začetno število žetonov v mestu p_{WOCL} določa zahtevano število končnih izdelkov. Pri tem stanja časovnikov, ki so prirejeni posameznim

Tabela 5.7: Proizvodni postopki izdelka 'Kotnik'.

	Operacija	Proc. čas	Proizv. viri
AB	Op1	8	V2
	Op2	15	Galvana1
	Op3	5	Miza1

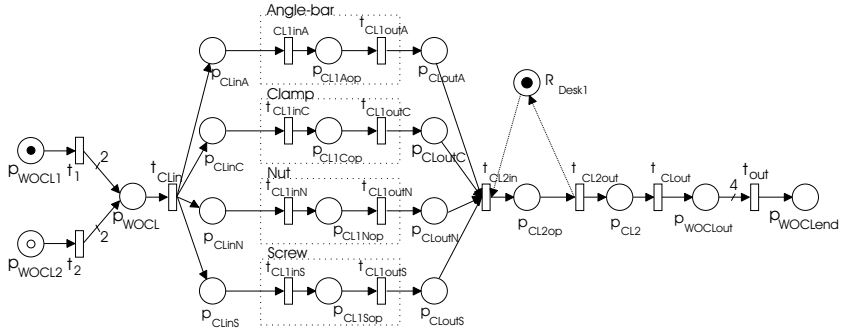
žetonom, označujejo čas začetka proizvajanja posameznega izdelka.



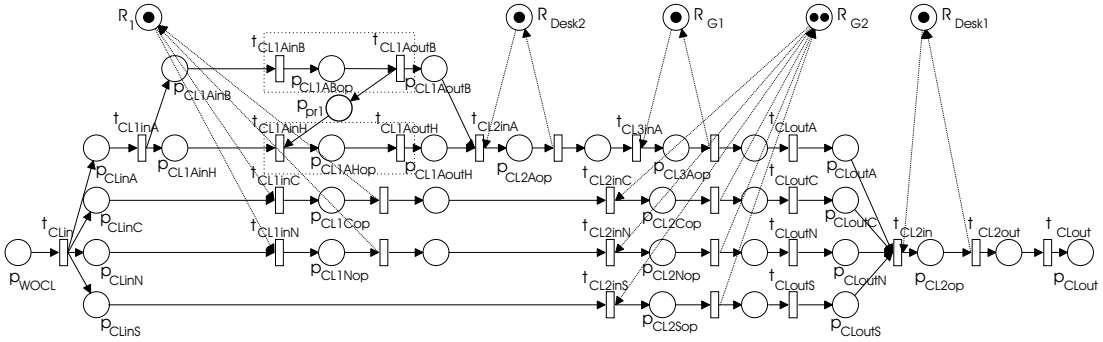
Slika 5.10: Osnovni model izdelka 'Vogalna spona L'.

Proizvodni podatki o izdelku so prebrani iz tabele 5.5. Prva operacija v tabeli ($Op1$) določa, da je potrebno najprej zagotoviti sestavne elemente, kot je to predpisano s kosovnico 'BOM_CCL'. Ta kosovnica se prebere iz tabele 5.4 in kot lahko opazimo, so potrebni štirje sestavni elementi ('AngBL', 'Spona', 'Matica' in 'Vijak'). Vsakemu sestavnemu elementu je predpisana oznaka, ki je podana v oklepaju, poleg imena komponente. Z oznakami so v modelu Petrijeve mreže predstavljene posamezne komponente. Ko so proizvedeni vsi ti sestavni deli, se lahko prične operacija sestavljanja – $Op2$. Situacija, ko so predstavljeni elementi vgrajeni v model, je prikazana na sliki 5.11.

V nadaljevanju algoritem za vsako komponento, ki predstavlja sestavni del, razpozna proizvodni postopek, po katerem je zgrajena. Proizvodni postopki za izgradnjo posameznih komponent, ki so potrebne za izdelavo izdelka 'Vogalna spona L', so podani v tabeli 5.5. Za izdelavo 'Vijaka' je potrebna ena operacija, za izdelavo 'Spone' in 'Matice' sta potrebni dve in za izdelavo levega kotnika z nosilcem ('AngBL') tri operacije. Rezultat vključevanja teh podatkov v naš model je predstavljen na sliki 5.12. Da je model preglednejši, je nekaj oznak prehodov in mest izpuščenih. Prav tako so izpuščena mesta in prehodi, ki označujejo začetek in konec delovnega naloga.

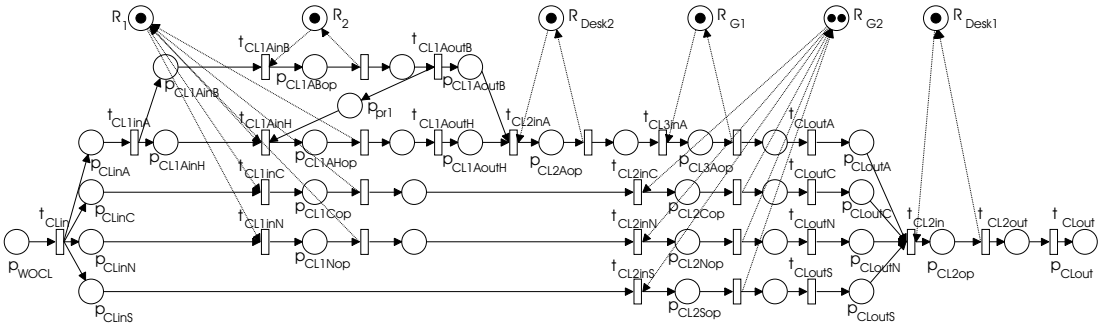


Slika 5.11: Model izdelka 'Vogalna spona L' ob upoštevanju proizvodnega postopka komponente CL.



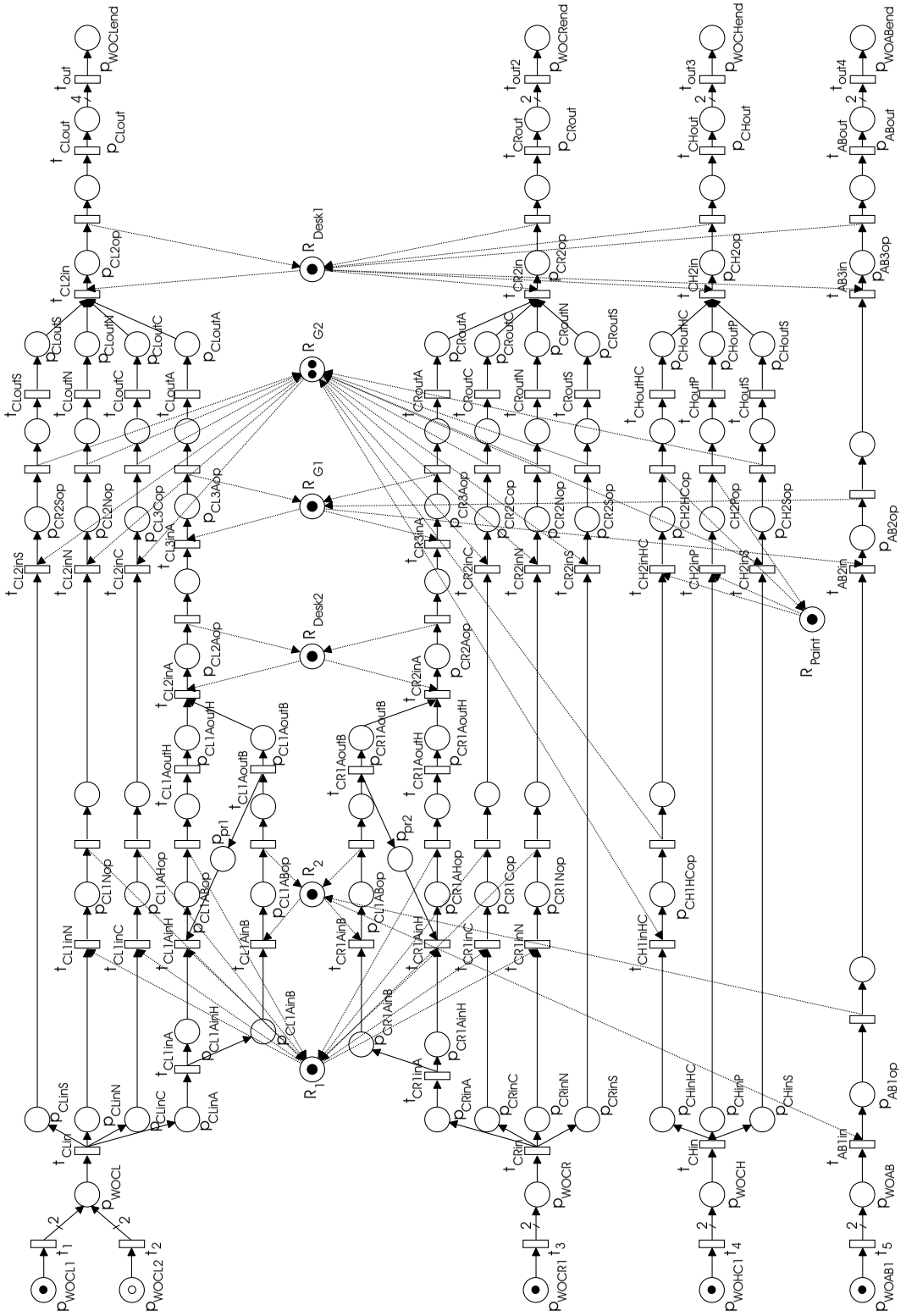
Slika 5.12: Model izdelka 'Vogalna spona L' ob upoštevanju proizvodnih postopkov komponent $AngBL$, $Spona$, $Matica$ in $Vijak$.

Iz modela lahko opazimo, da je za izdelavo kotnika z nosilcem ('AngBL') potrebno zagotoviti še dva sestavna dela ('AngB' in 'Nosilec'). Ko so v naš model vključeni vsi ti podatki, dobimo strukturo Petrijeve mreže, ki je predstavljena na sliki 5.13.



Slika 5.13: Celotni model izdelka 'Vogalna spona L'.

Enak postopek gradnje modela se izvede tudi za vse ostale izdelke, ki so



Slika 5.14: Celotni model proizvodnega postopka.

zahtevani z delovnimi nalogi. Celoten model proizvodnega postopka, zgrajen s Petrijevo mrežo, je predstavljen s sliko 5.14.

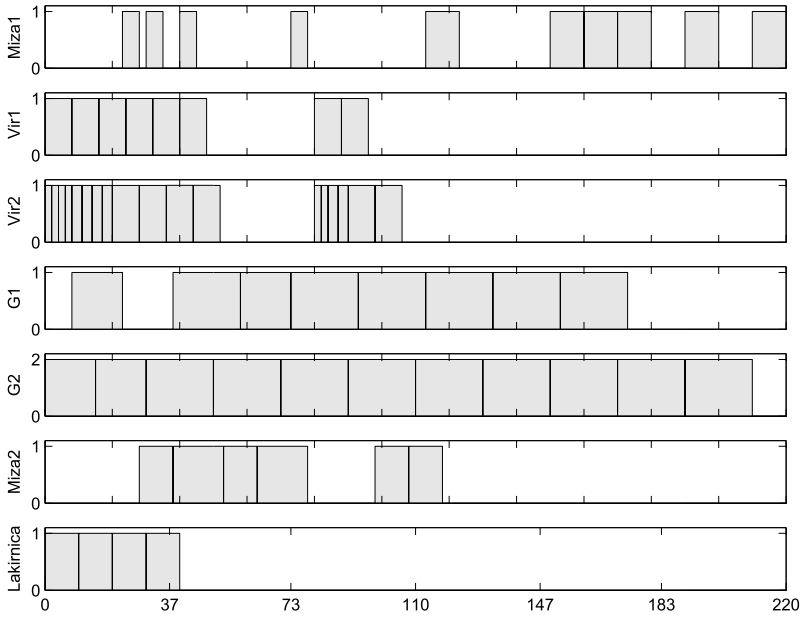
V nadaljevanju s pomočjo analize P-invariant preverimo pravilnost našega modela. Ugotovimo, da je prisotnih triindvajset P-invariant. Od tega se jih sedem nanaša na proizvodne vire, ki se nahajajo v modelu, štirinajst P-invariant se nanaša na proizvodne poti, poleg tega pa sta prisotni tudi dve invarianti, ki sta posledici prednostnih povezav. Iz ugotovljenega lahko sklepamo, da v modelu nismo ugotovili nepravilnosti.

5.7.2 Rezultati

Problem razvrščanja opravil v proizvodnem procesu smo s pomočjo časovnih Petrijevih mrež predstavili v matematični obliki. Pri tem je upoštevan postopek proizvodnje na vseh nivojih, t.j. delovni nalogi so med sabo usklajeni. Pri uporabi algoritmov razvrščanja pa so upoštevane vse operacije, ki nastopajo v proizvodnem sistemu. S pomočjo razvrščevalnika, ki lahko vključuje različna hevristična pravila, smo za dan problem dobili različne razvrstitve. Običajna zahteva v proizvodnji je minimiziranje izvršnega časa. Pravilo SPT se med pravil, ki smo jih preizkusili, izkaže kot optimalno. Kot rezultat dobimo razvrstitev, ki za izvedbo vseh nalog potrebuje 220 časovnih enot. Uporabljen razvrščevalnik omogoča tudi pregledno vizualno predstavitev razvrstitve dela preko razpoložljivih virov, kot je to predstavljeno na sliki 5.15. Na ta način lahko tudi spremljamo, ali je ob zahtevanih trenutkih na voljo dovolj sestavnih delov in polizdelkov ter predvidimo ukrepanje v nasprotnem primeru.

5.8 Primer 2: Šaržna proizvodnja

Predstavili pa bomo še en primer, kjer je predstavljen algoritem modeliranja uporabljen še na proizvodnem sistemu, ki je šaržnega tipa. Pregled področja, ki obravnava razvrščanje šaržnih procesov lahko najdemo v [65]. V primeru obravnavamo modelno napravo večnamenskega šaržnega procesa, ki je relativno enostavna, vendar vsebuje vse kompleksne elemente industrijskega okolja. Podrobno predstavitev procesa lahko najdemo v [76].

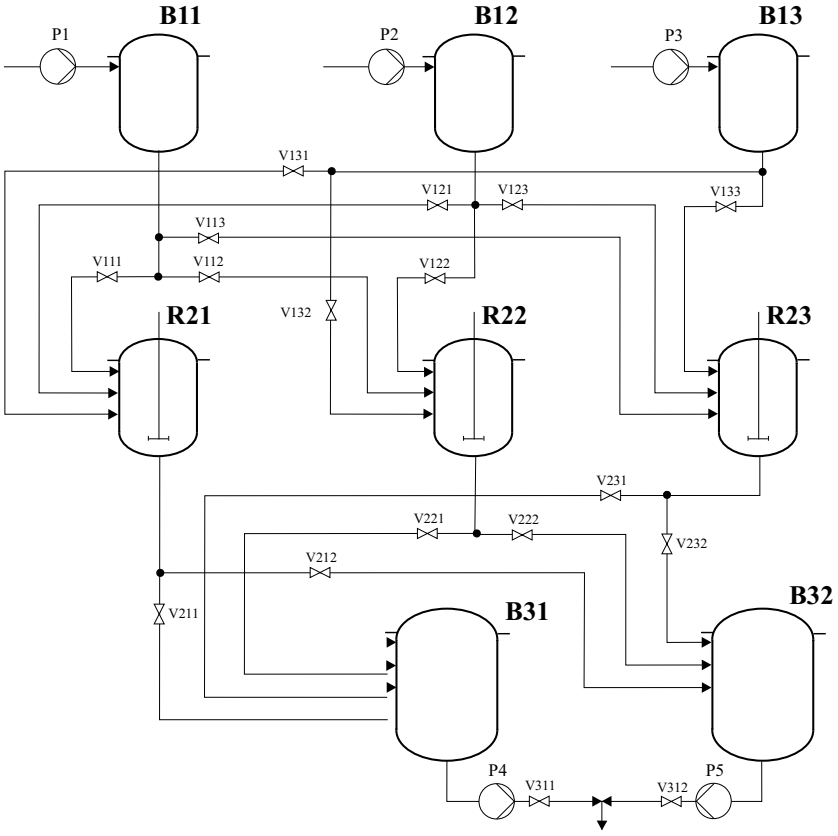


Slika 5.15: Razvrstitev opravil v kosovni proizvodnji, določena s pomočjo prioritetnega pravila SPT.

V splošnem je problematika vodenja šaržnih procesov formalizirana s standardom S88.01 [4]. Okvir za razvrščanje opravil v šaržnih procesih, ki upošteva standard S88.01, je obravnavan v [71]. Tudi v takih proizvodnih sistemih pa se v podporo upravljanju proizvodnje spet pogostokrat uporabljajo standardna orodja MRP [98]. S standardom S88.01 so procesne surovine in njihove količine, potrebne za izdelavo nekega produkta, določene s podatkovno strukturo, imenovano formula (*Formula*). Na ta način je formula ekvivalenta kosovnici, ki se uporablja v sistemih MRP. Uporaba Petrijevih mrež za modeliranje in analizo proizvodnih okolij šaržnega tipa je podrobno obravnavana v preglednem prispevku [31].

V nadaljevanju podajmo kratek opis obravnavanega procesa, ki proizvaja dva različna produkta (substanci) iz treh različnih tekočin (surovin). Surovine predstavimo z 'rdečo', 'rumeno' in 'belo' barvo, medtem ko produkta predstavimo z 'modro' in 'zeleno'. Kemijska reakcija, ki predstavlja ozadje delovanja celotnega sistema, je nevtralizacija klorovodikove kisline (HCl) z natrijevim hidroksidom (NaOH). Klorovodikovi kislini se doda dva različna indikatorja, prvi jo obarva rdeče drugi pa rumeno. Skupaj z 'belim' natrijevim hidroksidom predstavljajo surovine. Če nevtraliziramo eno šaržo 'rdeče' klorovodikove kisline z eno šaržo 'belega' natrijevega hidroksida, dobimo 'moder' produkt, kjer je sprememba barve

povezana s primešanim indikatorjem in spremembo pH vrednosti, če pa nevtraliziramo eno šaržo 'rumene' klorovodikove kisline z eno šaržo 'belega' natrijevega hidroksida, dobimo 'zelen' produkt.



Slika 5.16: Večnamenski šaržni proces.

Naprava se sestoji iz treh različnih nivojev, glej sliko 5.16. Na prvem nivoju so rezervoarji, v katerih hranimo surovine. Rezervoar B11 hrani rumeno, B12 rdečo in B13 belo surovino. Na drugem nivoju se nahajajo reaktorji R21, R22 in R23, v katerih mešamo vhodne surovine in tako proizvajamo dva različna produkta. Proizvodni postopek je tak, da najprej napolnimo reaktor z eno šaržo rumene ali rdeče surovine in nato še z eno šaržo bele surovine. Po končanem mešanju reaktor spraznimo v tretji nivo, kjer hranimo produkte. To sta dva rezervoarja: B31, ki je namenjen modremu produktu in B32, ki je namenjen zelenemu produktu. Vsak od njiju lahko hrani največ tri šarže produktov in ju lahko izpraznimo, ko sta polna. Posode v šaržnem procesu so prostorsko porazdeljene, tako so tudi povezave med posodami različne. Temu primerno se razlikujejo tudi časi praznjenja in polnjenja le teh. Na potek proizvodnje v sistemu lahko vplivamo s

črpalkami P1-P5 in ventili V111-V311.

Predstavljena naprava omogoča definiranje množice različnih problemov razvrščanja. Na tem mestu bomo obravnavali problem, ko imamo zahtevo po želeni količini produktov ter začetni čas postopka. Določiti pa želimo, kdaj naj bodo na voljo vhodne surovine ter kdaj se postopek konča. Delovni nalog, podan s tabelo 5.8, definira problem, ki zahteva, da je potrebno proizvesti šest šarž modrega in šest šarž zelenega produkta. Sledi, da potrebujemo šest šarž rumene in rdeče surovine ter dvanajst šarž bele surovine.

Tabela 5.8: Delovni nalogi za zahtevane produkte

Izdelek	Koda	Količina	Začetni čas
Modra surovina	MO	6	0
Zelena surovina	ZE	6	0

Proizvodni postopek in pa tudi sama struktura šaržnega procesa je podana že v sistemu za planiranje. Tako formula, predstavljena s tabelo 5.9, podaja spisek surovin, ki so potrebne za izdelavo končnega produkta. Za proizvodnjo 'modrega' produkta (B) potrebujemo eno šaržo 'bele' (WM) surovine, vendar pri tem velja, da mora biti še prej na voljo ena šarža 'rumene' (Y) surovine. To je v formuli podano s prednostno povezavo. Na enak način je definirana tudi formula za 'zelen' produkt. Produkti 'Y', 'R' in 'WZ' so predhodno pripravljeni iz vhodnih surovin, ki jih označimo z 'Y0', 'R0' oz. 'W0'.

Proizvodna postopka za izdelavo posameznega produkta sta podana v tabelah 5.10 in 5.11. Prva operacija $Op10$ v proizvodnem postopku izdelave 'modrega' produkta določa, da je najprej potrebno proizvesti (vmesni) produkt, ki je določen s formulo 'B'. V nadaljevanju sta potrebni še dve operaciji, s katerima zmešana produkta shranimo v rezervoar B31 in ta rezervoar izpraznimo. Dejstvo, da se rezervoar prazni šele, ko se napolni s tremi šaržami vmesnega produkta 'B', je predstavljeno z zapisom $(3 \times 1)\cdot$B31. V primeru, ko imamo opravka s šaržnimi procesi, se pogosto pojavlja situacija, ko je potrebno neko nalogo dokončati s skupino virov, ki smo jo začeli uporabljati že v predhodni nalogi. Tak primer se pojavi pri obravnavi našega problema, ko nastopi uporaba reaktorja. Viri,

Tabela 5.9: Formule produktov

Nadr. komp.	Podr. komp.	Količina	Prednostne povezave
B	Y	0	0 1
	WM	1	0 0
G	R	0	0 1
	WZ	1	0 0
Y	Y0	1	0
R	R0	1	0
WM	W0	1	0
WZ	W0	1	0

ki se zvezno uporabljajo v večih nalogah, so podani v oklepajih. Viri pa so tudi dodatno označeni z zaporedno oznako v oglatem oklepaju, s pomočjo katere lahko določimo, kdaj naj bodo ti viri sproščeni. Tako pomeni oznaka [$R1\mathbf{3}$], ki jo ima operacija praznjenja $R2x$ v $B31$ in je pripojena reaktorju, da operacija sicer potrebuje omenjeni vir, vendar bo le-ta sproščen šele, ko se pojavi operacija z oznako [$R1\mathbf{1}$]. V tabeli najdemo še vse potrebne informacije za postopke, ki so potrebni za izvedbo posameznih komponent, ki nastopajo v zahtevani formuli.

Tabela 5.10: Proizvodni postopki izdelave 'modrega' produkta.

	Operacija	Proc. čas	Proizv. viri
MO	Op10	–	BOM_B
	Op20 prazni R2x v B31	12/12/12	[R13](R21/R22/R23), B31
	Op30 črpaj iz B31	30	$(3 \times 1)\cdot$B31
Y	Op10 prazni B11 v R2x	15/12/12	[R11](R21/R22/R23), [BY2](B11)
WM	Op10 prazni B13 v R2x	10/9/13	[R12](R21/R22/R23), [BW2](B13)
Y0	Op10 črpaj Y v B11	12	[BY1](B11)
W0	Op10 črpaj W v B13	12	[BW1](B13)

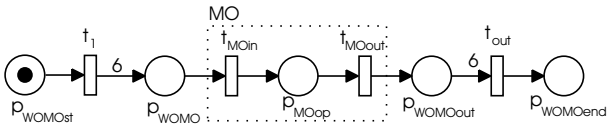
Tabela 5.11: Proizvodni postopki izdelave 'zelenega' produkta.

	Operacija	Proc. čas	Proizv. viri
ZE	Op10	–	BOM_G
	Op20 prazni R2x v B32	12/12/12	[R23](R21/R22/R23), B32
	Op30 črpaj iz B32	30	(3×1)·B32
R	Op20 prazni B12 v R2x	11/13/14	[R21](R21/R22/R23), [BR2](B12)
WZ	Op20 prazni B13 v R2x	10/9/13	[R22](R21/R22/R23), [BW2](B13)
R0	Op10 črpaj Y v B12	12	[BR1](B12)
W0	Op10 črpaj W v B13	12	[BW1](B13)

5.8.1 Modeliranje večnamenskega šaržnega procesa

Predstavljen proizvodni sistem modeliramo s časovnimi Petrijevimi mrežami. S tem modelom lahko določimo razvrstitev opravil, ki so potrebna za izdelavo zahtevanih končnih produktov.

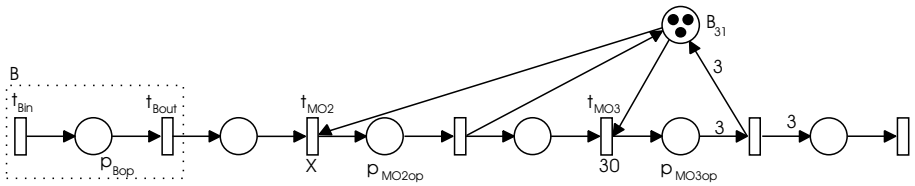
Model izdelave modrega produkta zgradimo s pomočjo predstavljenega algoritma. Iz delovnega naloga (tabela 5.8) se ugotovi, katere produkte je potrebno proizvesti in koliko jih je potrebnih. S prvim korakom modeliranja 'modrega' produkta dobimo mrežo predstavljeno na sliki 5.17.



Slika 5.17: Osnovni model 'modrega' produkta.

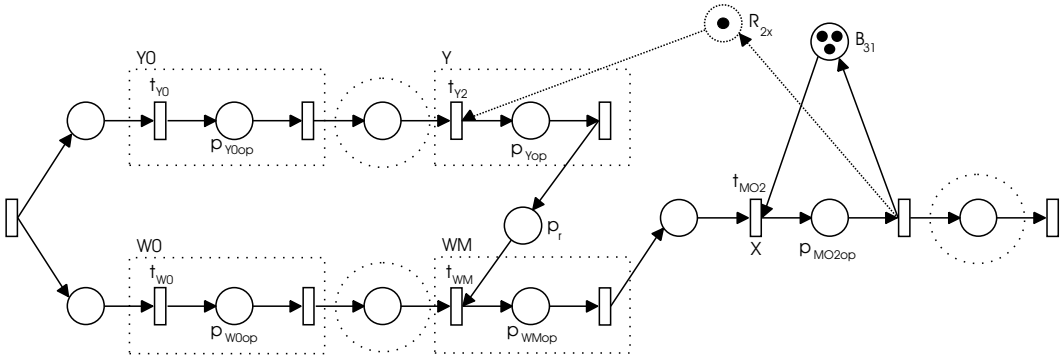
V nadaljevanju se v ta model (del v okvirju) dodaja dodatne informacije, ki jih pridobimo iz tabele o proizvodnem postopku, tabela 5.10. Doda se informacijo o praznjenju reaktorja in črpanju produkta iz rezervoarja R31. Ker operacija $Op20$ zahteva uporabo virov, ki so določeni s predhodno operacijo, ji na tem mestu še ne dodamo vseh podrobnosti. Dobimo model, predstavljen na sliki 5.18.

Nato je v model vključena informacija o mešanju, določena s formulo za kom-



Slika 5.18: Model proizvodnega postopka komponente 'MO'.

ponento 'B', in o ostalih potrebnih komponentah (tabela 5.9). Poleg tega pa se na tem mestu določi tudi način uporabe reaktorja, ki ga uporabljamo še v naslednji nalogi (slika 5.19).

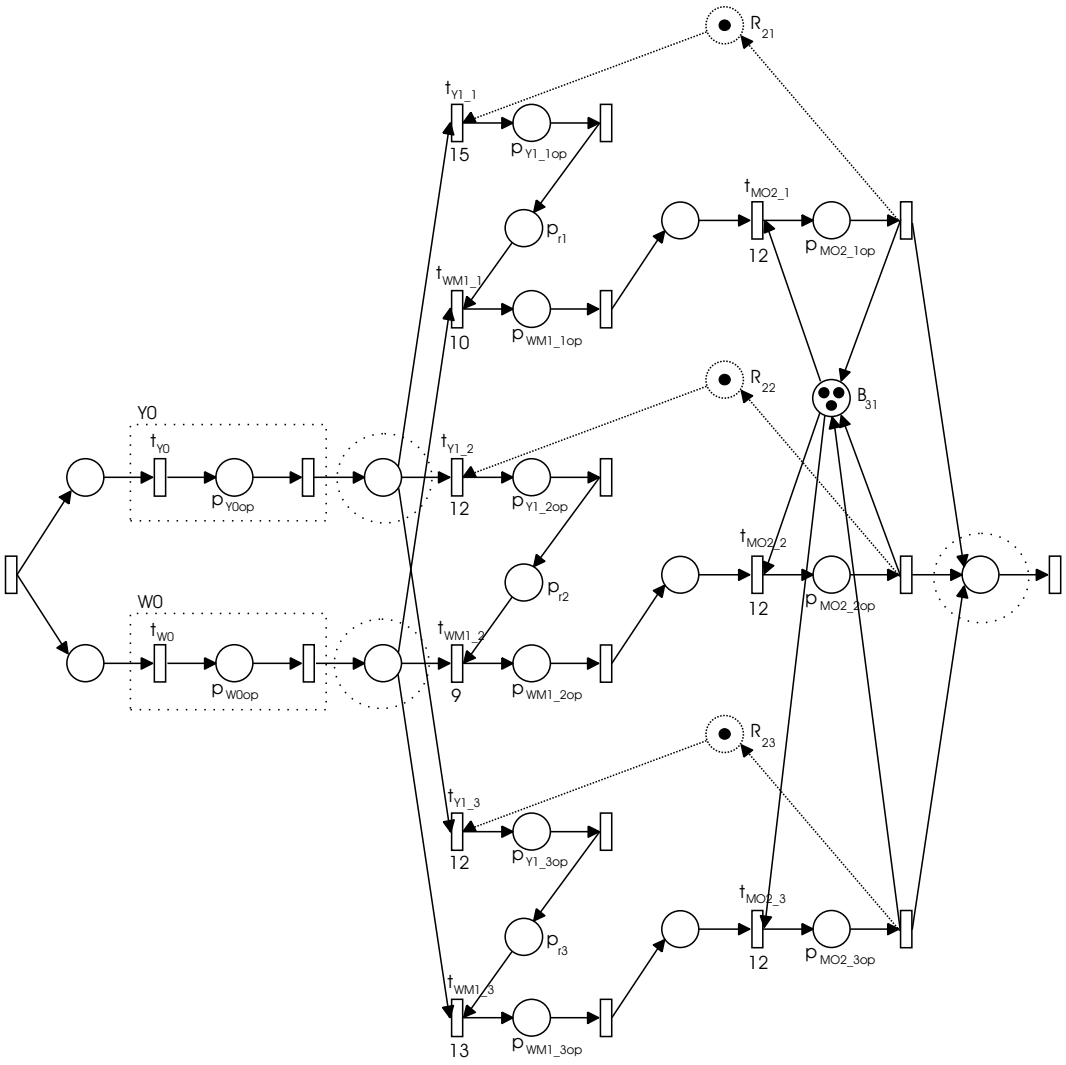


Slika 5.19: Shematski prikaz komponente 'M'.

Hkrati pa se doda tudi informacija o tem, da je operacije, ki jih izvaja reaktor, mogoče izvajati na treh različnih reaktorjih. Tako dobimo model, ki je predstavljen na sliki 5.20.

S tem postopkom dobimo detajlni model postopka izdelave 'modrega' produkta. Enak postopek je bil izveden še za postopek izdelave 'zelenega' produkta. Rezultirajoč model celotnega večnamenskega šaržnega procesa je predstavljen na sliki 5.21.

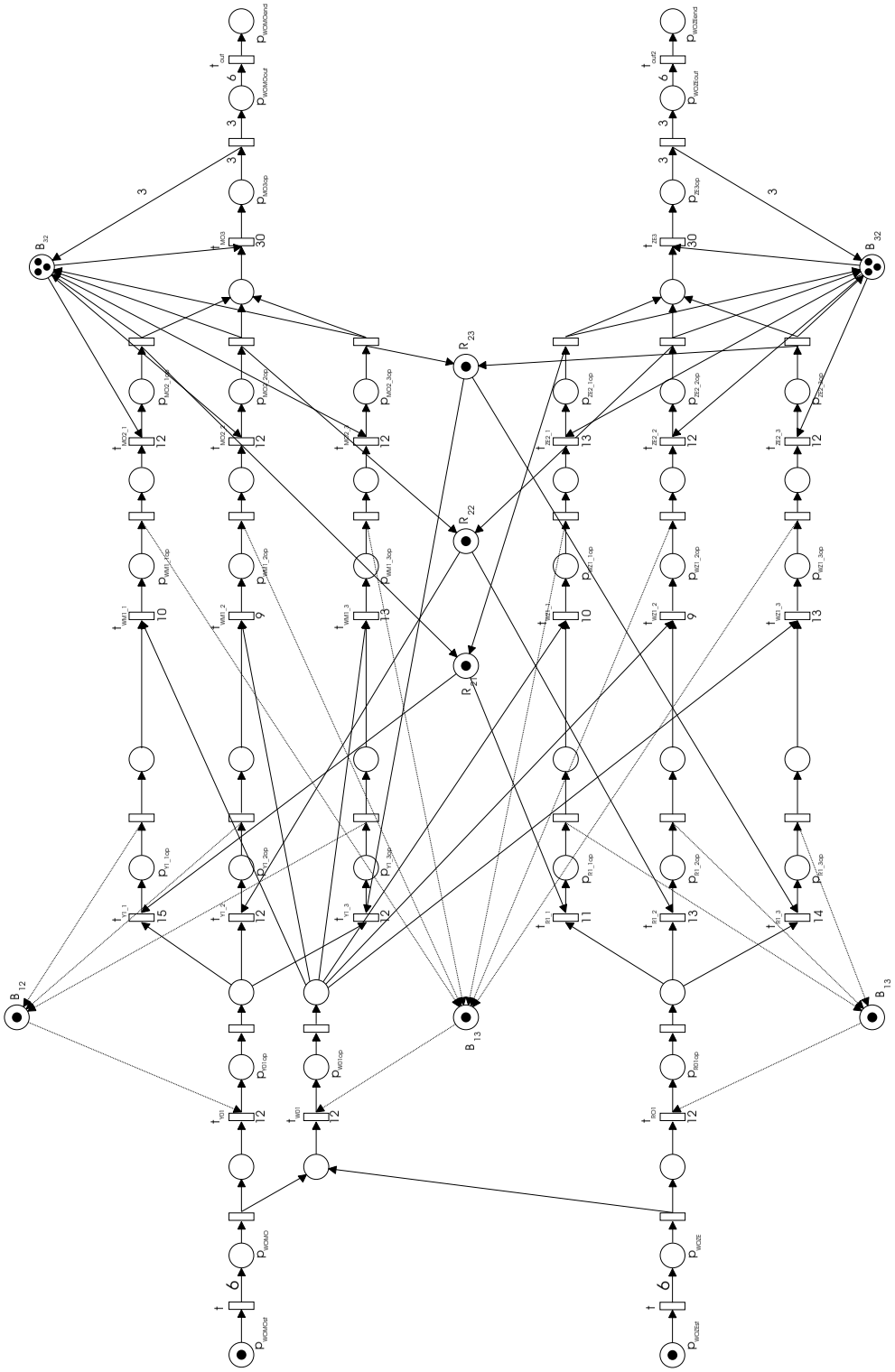
Tudi v tem primeru smo rezultirajoči model analizirali z invariantno analizo. Ugotovimo, da je prisotnih enajst P-invariant, pri čemer se jih osem nanaša na proizvodne vire in tri na proizvodne poti.



Slika 5.20: Model proizvodnega postopka komponente 'M'.

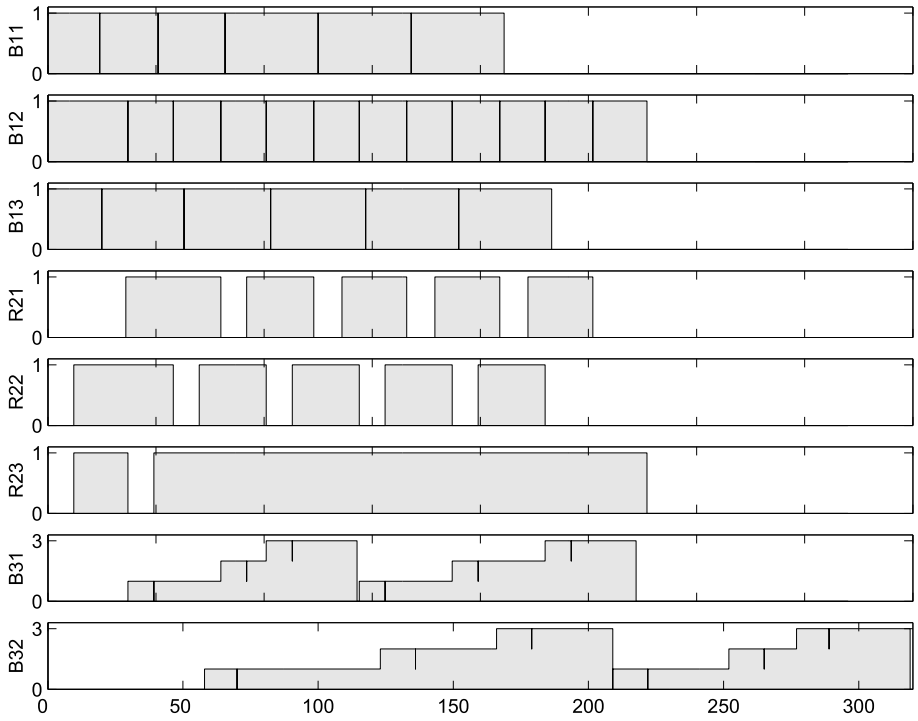
5.8.2 Rezultati

Nad zgrajenim modelom, predstavljenim s časovnimi Petrijevimi mrežami, uporabimo razvrščevalnik. Pri razvrščanju opravil, ki nastopajo pri izdelavi želenih produktov, preizkusimo različna pravila razvrščanja. Časi izvajanja celotnega postopka (izvršni časi) za različne algoritme so predstavljeni v tabeli 5.12. Rezultate razvrščanja s hevrističnimi pravili primerjamo z algoritmom razvejaj in omeji, ki ga avtor uporabi v [76]. Opazimo, da se tudi v tem primeru pravilo SPT izkaže kot najboljše, saj je izvršni čas izvajanja postopka najkrajši, 315 s. Z razvrstitvijo, predstavljeno na sliki 5.22, pa lahko tudi enostavno ugotovimo, ob



Slika 5.21: Končni model večnamenskega šaržnega procesa.

katerih trenutki morajo biti na razpolago različne vhodne surovine.



Slika 5.22: Razvrstitev.

Tabela 5.12: Rezultati

Algoritem	Čas trajanja
SPT rule	315 s
LPT rule	331 s
Razvejaj in omeji [76]	323 s

6. Razvrščanje proizvodnih opravil z orodji za vodenje projektov - MS Project

Planiranje in vodenje projektov se ukvarja z upravljanjem s projekti. Projekt je opisan kot niz nerutinskih aktivnosti, ki so med sabo povezane z namenom, da opravijo neko specifično nalogo. Za dosego optimalnega plana projekta so potrebne številne medsebojno povezane odločitve. V podporo pri odločanju so bile upravljavcu razvite specifične tehnike za delo s projekti.

Vodenje projektov je pomembna funkcija tudi v proizvodni industriji v fazi načrtovanja novega izdelka.

Končni cilj projekta (izdelek) je nerutinski. Nerutinski cilj pa je možno doseči tudi z nizom rutinskih aktivnosti. Tipičen tak primer v proizvodnem okolju je sestav posamične obdelave. Proizvodnja, kjer se za vsak načrtan izdelek proizvaja le po en ali neka omejena količina kosov, je lahko opredeljena kot projektno usmerjena proizvodnja. Projekti so si podobni s proizvodnimi procesi v tem, da so sestavljeni iz več procesnih korakov oz. aktivnosti.

6.1 Vodenje projektov

Projekt bi lahko definirali kot enkratna, začasna, praviloma zahtevna in zapletena skupina nalog potrebna za izdelavo novega izdelka ali storitve, ki mora biti končana v določenem časovnem obdobju. Vsak projekt v osnovi sestavljajo dejavnosti, kot so planiranje, organizacija in nadzor ter razpoznavanje in razvrščanje virov, ki so potrebni za izvedbo projekta. Vsak projekt lahko razgradimo na posamezne aktivnosti (opravila, naloge). Vsaka aktivnost ima predpisan čas izvajanja. Pri gradnji projekta je treba najprej narediti seznam opravil, predpisati predpostavljen čas izvajanja, razpoznati odvisnosti in zakonitosti med posame-

znimi opravili in določiti potrebne vire. Strukturirana členitev dela (*Work Breakdown Structure – WBS*) je tehnika, s pomočjo katere razdelimo projekt na vedno manjše, obvladljive enote, dokler niso definirane v dovolj majhnih podrobnostih, da omogočajo določanje porabe časa in stroškov in s tem izvajanje nadaljnjih aktivnosti projekta. Strukturo WBS je velikokrat možno ponovno uporabiti v drugem projektu, ker je veliko projektov do določene mere podobnih. Elementi, ki so najnižje v strukturi, se imenujejo aktivnosti.

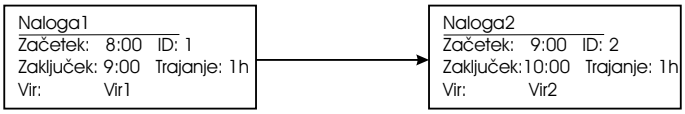
Orodja za vodenje projektov nam omogočajo opisati medsebojne odvisnosti med aktivnostmi. Podati je treba vire, ki skrbijo, da bodo aktivnosti končane v predvidenem času. Nato je treba najti rešitev, kako te aktivnosti časovno uskladiti, da je izvajanje projekta čimbolj optimalno. V ta namen obstaja kar nekaj pripomočkov, ki ponujajo analizo plana in pomoč pri reševanju preobremenitve virov.

Problem razvrščanja projektov se običajno nanaša na razvrščanje nalog, med katerimi veljajo določene zakonitosti, t.j., prednostne povezave. Problem lahko predstavimo kot problem razvrščanja vzporednih strojev, katerih kapaciteta je neomejena. Kriterij, ki se tu običajno uporablja, je minimiziranje izvršnega časa, ob upoštevanju vseh zakonitosti, ki nastopajo med nalogami. Torej lahko običajen problem razvrščanja projektov podamo z zapisom $Pm|prec|Cm$. Pogosto pa se tudi pri razvrščanju projektov pojavljajo viri, katerih kapaciteta je omejena, npr. osebje, ki skrbi za izvršitev določene naloge. V tem primeru je problem razvrščanja težji.

Najbolj razširjeno matematično orodje za planiranje in vodenje projektov je mrežno planiranje[1] [47]. Mrežno planiranje temelji na prikazu diskretnih opravil (nalog, dogodkov) s pomočjo mrežnega diagrama (*Network diagram*). Le-ta v grafični obliki predstavlja zaporedje in medsebojno odvisnost opravil, ki so potrebna za realizacijo projekta. Vsaka naloga je grafično predstavljena kot pravokotnik, ki vsebuje informacijo o zaporedni številki naloge, nazivu naloge, trajanju, predvidenem začetku, predvidenem koncu in o tem, kateri viri so potrebni za njeno izvedbo, glej sliko 6.1.

Prvi metodi mrežnega planiranja sta bili razviti v 50-ih letih prejšnjega sto-

[1]Matematična osnova mrežnega planiranja je teorija grafov, ki je sestavni del vede operacijskih raziskav.



Slika 6.1: Mrežni diagram

letja. Kot prvo so leta 1957 v kemični tovarni Du Pont razvili metodo kritične poti (*Critical Path Method – CPM*). Z metodo CPM poskušamo določiti neko pričakovano trajanje projekta. Izhodišče za časovno analizo po metodi CPM so določeni normativni časi trajanja opravil d_i. Zaključni (izvršni) čas projekta EF lahko tako določimo, ko vemo začetni čas projekta ST, ter začetne in končne čase posameznih opravil (ST_i in EF_i). Predpostavimo, da imamo opravka s projektom z N opravili, ki so postavljena v takem zaporedju, da ima lahko i-to opravilo le predhodnike, ki so postavljeni v zaporedje pred tem opravilom. Za vsako opravilo lahko določimo najzgodnejši rok zaključka:

$$EF_i = d_i + \max(EF_j \mid j < i), j \text{ je predhodnik naloge } i. \tag{6.1}$$

Tako nam izraz $LS_i = EF_i - d_i$ predstavljan najzgodnejši rok začetka i-tega opravila. Opazovati pa je potrebno tudi najkasnejši rok začetka opravil LS_i, ki še zagotavlja, da bo projekt končan v času EF.

$$EF_i = \min(EF_j \mid j < i) - d_i, j \text{ je predhodnik naloge } i. \tag{6.2}$$

Za vsako opravilo lahko sedaj definiramo lebdeči čas (*Slack time*), $S_i = ES_i - LS_i$. Opravilo, čigar $S_i = 0$, je kritično. Pot v mrežnem diagramu, ki povezuje začetno in končno opravilo, in ki je izmed vseh poti časovno najdaljša, je kritična pot. Torej to je pot, ki povezuje kritična opravila. Kritična pot s svojim časom trajanja definira čas trajanja projekta. Metoda CPM skrbi za premikanje opravil na kritični poti in na ta način minimiziranje izvršnega časa projekta.

Leto kasneje je bila za potrebe ameriške vojne mornarice razvita tehnika ocenjevanja in pregledovanja projekta (*Program Evaluation & Review Technique – PERT*). Metoda PERT upošteva tudi naključne čase trajanja aktivnosti, s predpostavljeno verjetnostno porazdelitvijo. Na osnovi pesimistične, realne in optimistične ocene trajanja posameznih aktivnosti s posebnimi izračuni ugotavljamo

njihovo najverjetnejše oziroma pričakovano trajanje. Pri tem je pomembno, da se optimistična in pesimistična ocena ne razlikujeta preveč, saj velike razlike kažejo verjetnost, da bo dejansko trajanje aktivnosti odstopalo od pričakovanega.

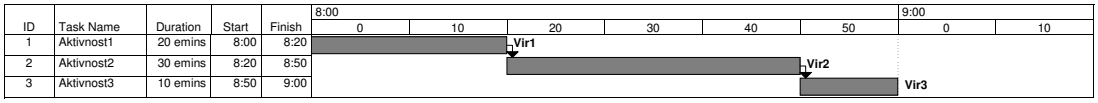
Na podlagi omenjenih metod se je, glede na različne značilnosti projektov, razvila cela vrsta podobnih metod za planiranje projektov, ki se med seboj sicer razlikujejo glede na uporabljeno tehniko in namen uporabe, vse pa temeljijo na nazornem grafičnem prikazovanju aktivnosti in njihovih medsebojnih odvisnosti. Te metode mrežne analize omogočajo časovno analizo projekta na različnih osnovah, optimalno razporejanje izvajalnih kapacitet, optimizacijo stroškov in ugotavljanje verjetnosti realizacije projekta. Obstaja veliko literature, kjer je mogoče najti podrobnejši opis omenjenih metod, nekaj osnov je podanih v [49, 80]. Sodobni komercialni programski paketi uporabljajo metode, ki so kombinacija metod PERT in CPM in vključujejo možnosti za izbiro aktivnosti z naključnimi časi trajanja. Ponavadi so te metode poimenovane s terminom CPM/PERT.

6.2 Primerjava med proizvodnim razvrščanjem in vodenjem projektov

V nadaljevanju bomo predstavili notacijo, ki na enak način obravnava tako proizvodno razvrščanje kot tudi vodenje projektov. Ekvivalent kosovnicam je pri uporabi vodenja projektov členitev WBS. Oba zapisa omogočata strukturni opis elementov, ki sestavljajo izdelek oz. storitev. Projekti in proizvodni procesi so sestavljeni iz posamičnih procesnih korakov oz. aktivnosti. Opravila na višjem nivoju v strukturi WBS tako predstavljajo nalogo, opravila na nižjem nivoju pa predstavljajo operacije, ki to nalogo sestavljajo. Vsaka operacija potrebuje določen čas trajanja za izvedbo. Niz zaporednih operacij je vzpostavljen s povezovanjem opravil. Vsakemu opravilu podamo vire, ki jih potrebuje za izvedbo.

Za prikaz razvrstitve se v obeh primerih uporablja grafična predstavitev s pomočjo gantograma. Le-ta omogoča pregleden časovni nadzor nad uporabo virov in analizo predlagane razvrstitve. Daje tudi možnost interaktivnega posega-nja v dano razvrstitev. S tem daje planerju možnost vpliva na razvrstitev glede na njegove lastne izkušnje. Trajanje opravila je predstavljeno z dolžino pravokotnika v gantogramu. Usmerjene povezave so uporabljene za ilustrativen prikaz

medsebojne odvisnosti opravil. Uporabljata se dve obliki gantograma. Prva se uporablja za nadzor napredovanja dela zlasti pri orodjih za razvrščanje projektov. Ordinatna os kaže naloge, abscisna pa prikazuje čas (glej sliko 6.2). Druga oblika prikaza se ponavadi uporablja pri razvrščanju opravil v proizvodnem postopku. Tu ordinatna os prikazuje proizvodne vire, na katere nanašamo opravila, abscisna pa čas.



Slika 6.2: Gantogram, ki ga uporabljajo orodja za vodenje projektov

Za vodenje projektov je značilno, da pravokotniki v gantogramu pomenijo aktivnosti in uporabnik jim dodeli potrebne vire. Dodeljevanje virov k operacijam ne upošteva končne zmogljivosti virov. S povezavami in prerazporejanjem pa lahko zagotovimo razvrstitev, ki upošteva omejene zmogljivosti virov.

6.3 MS Project kot orodje za razvrščanje

Na tem mestu bomo poudarili nekaj lastnosti MS Projecta, sicer tipičnega predstavnika orodij za vodenje projektov, ki jih lahko koristno uporabimo v proizvodnem razvrščanju.

Z MS Projectom lahko predstavimo nalogo sestavljeno iz večih operacij. V ta namen so aktivnosti združene po stopnjah v obliki strukture WBS. Stopnje in pod-stopnje projekta pomenijo hierarhijo med aktivnostmi, kar lahko določimo s strukturirano členitvijo. Posamezne stopnje lahko predstavimo s preglednimi opravili (nalogami). Le-ta so sestavljena iz sestavnih opravil (operacij).

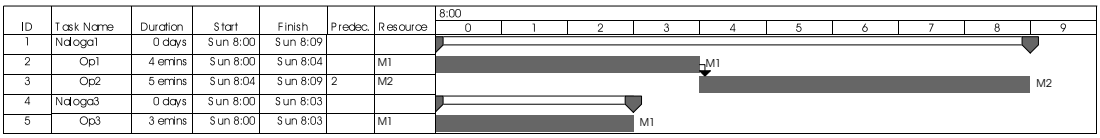
Za vsako opravilo podamo čas izvajanja. Za izvedbo zaporednega izvajanja operacij se uporablja povezovanje opravil. Na primer s povezavo *konec-začetek* (*Finish-to-Start – FS*) definiramo, da se lahko naslednje opravilo začne šele, ko se prejšnje konča. Združevanje opravil je lahko časovno različno, npr. kolikor mogoče zgodaj (*As Soon As Possible – ASAP*). V bazi podatkov MS Projecta so zbrani tudi vsi podatki o virih, le-te pa potem priredimo k posameznim opravilom.

Z vsemi temi zmožnostmi MS Projecta lahko opišemo tudi proizvodni pro-

ces, ki je podan v obliki kosovnice (strukture BOM). Za predstavitev in zapis posameznih elementov, ki omogočajo opis proizvodnega sistema, je na voljo več urejevalnikov oz. grafičnih vmesnikov, ki nam omogočajo vnos podatkov v bazo in pregled nad njimi. Obstajata dva tipa pogledov: pogled nad nalogami (gantogram, spremljanje plana, koledar, ...) in pogled nad viri (histogram vira, tabela virov...).

MS Project omogoča avtomatsko izravnavo preobremenjenih virov. Opravila, ki posamezne vire preobremenjujejo, zakasni ali razdeli na več segmentov. Funkcija ne zagotavlja optimalne rešitve, saj uporablja le omejen niz pravil, kot so: prednostne povezave, dopustne zakasnitve, prioritete in dane omejitve opravil. Zato je lahko zelo koristno, da se rezultirajoča razvrstitev na koncu preveri in po potrebi naredijo popravki.

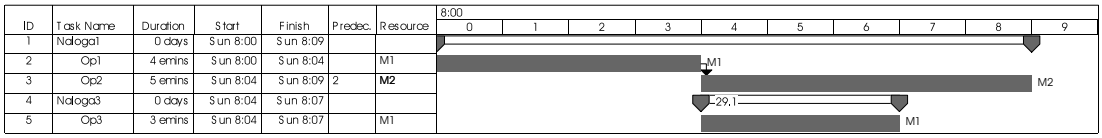
Z enostavnim primerom pokažimo, kako bi lahko uporabili Microsoft Project za reševanje problema razvrščanja sestava posamične obdelave. Poglejmo si zopet problem razvrščanja dveh nalog na dva vira, ki je bil že večkrat obravnavan v 3. poglavju. Predstavljen problem je prikazan na sliki 6.3. Na gantogram se naložita obe nalogi. Vsaka izmed nalog je sestavljena iz večih operacij, podane so kot sestavna opravila. Poleg tega je vsaki operaciji dodana še informacija o času trajanja in povezava s potrebnimi viri, ki so že prej definirani v bazi podatkov o virih.



Slika 6.3: Razvrstitev sestava posamične obdelava – razpoložljivost virov ni upoštevana

Opazimo, da rezultirajoča razvrstitev ne upošteva končne kapacitete virov. Tako bi v istem časovnem obdobju vir M_1 potrebovali dve različni operaciji. Za razrešitev tega problema preobremenjenosti uporabimo funkcijo avtomatske izravnave virov. Na ta način dobimo izvedljivo rešitev, ki je predstavljena na sliki 6.4.

Orodje MS Project omogoča tudi analizo PERT. Na ta način lahko ocenimo najboljši možen, pričakovan in najslabši možen potek dela. Na različne načine



ID	Task Name	Duration	Start	Finish	Predec.	Resource
1	Naloga1	0 days	Sun 8:00	Sun 8:09		
2	Op1	4 emins	Sun 8:00	Sun 8:04		M1
3	Op2	5 emins	Sun 8:04	Sun 8:09	2	M2
4	Naloga3	0 days	Sun 8:04	Sun 8:07		
5	Op3	3 emins	Sun 8:04	Sun 8:07		M1

Slika 6.4: Izvedljiva razvrstitev sestava posamične obdelava

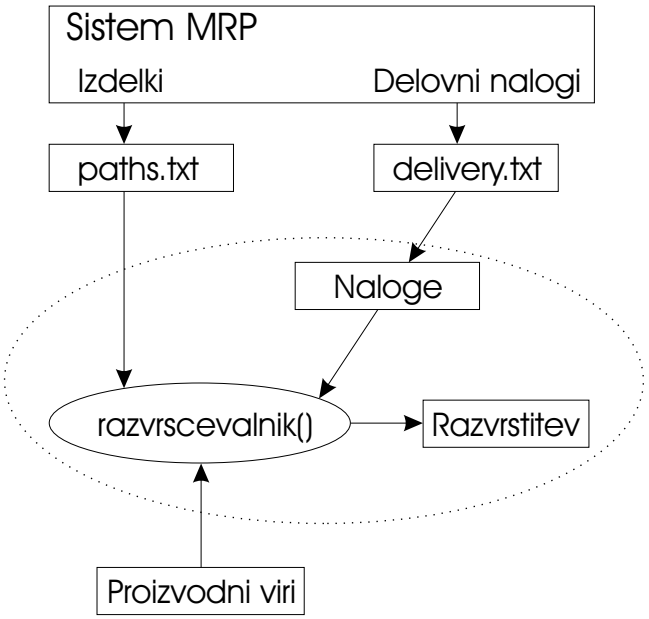
so lahko prikazana opravila, ki so na kritični poti. Tako nam orodje služi le kot pripomoček pri iskanju minimiziranja izvršnega časa. S pomočjo MS Projecta lahko projekt predstavimo tudi s pomočjo mrežnega diagrama.

V okolju MS Project se lahko uporabljajo tudi naslednja programska orodja: Microsoft Basic for Applications (VBA), Component Object Model (COM) dodatki, OLE DB in Extensible Markup Language (XML). Z njimi lahko MS Project povežemo s poljubnimi informacijskimi sistemi.

6.4 Izvedba razvrščanja z orodjem MS Project

V podporo generiranju razvrstitve opravil smo razvili rešitev, integrirano znotraj orodja MS Project [30]. Rešitev prilagodi in razširi funkcionalnosti orodja za vodenje projektov, tako da je v pomoč pri detajlnem razvrščanju. Rešitev smo zasnovali v obliki postopka v jeziku VBA, ki smo ga vgradili v orodje MS Project in uporablja njegov objektni model.

S predlaganim postopkom lahko zgradimo model, katerega uporabimo za določitev razvrstitev opravil v proizvodnem sistemu. Postopek gradnje modela temelji na isti filozofiji kot postopek, ki smo ga predstavili in uporabili pri gradnji modela s časovnimi Petrijevimi mrežami. Za gradnjo modela za razvrščanje uporabimo podatke, ki so prisotni v že obstoječih informacijskih sistemih, ki se uporabljajo v nekem proizvodnem okolju. Podatke o razpoložljivih proizvodnih virih pridobimo iz proizvodnje, medtem ko prejemamo zahteve po želenih končnih izdelkih preko delovnih nalogov, ki so generirani v sistemih za vodenje na višjih nivojih (MRP, ERP). Iz teh sistemov lahko običajno pridobimo tudi podatke o strukturi izdelkov in proizvodnih postopkih, ki so potrebni za njihovo izdelavo. Postopek najprej pripravi podatke o virih, v nadaljevanju pa na gantogram naloži naloge, ki so potrebne za izvedbo delovnih nalogov – razvrščanje. Shema, ki nam ilustrira predlagan postopek, je podana na sliki 6.5.



Slika 6.5: Shema postopka gradnje razvrstitve

Pri razvrščanju je potrebna informacija o proizvodnih poteh, ki določajo izvedbo izdelkov/storitev. Te informacije so podane v datoteki *Paths.txt* in pomenijo podatke o potrebnih operacijah, njihovih časih izvajanja ter virih, ki jih je treba prirediti posameznim operacijam. V MS Projectu so operacije, ki so potrebne za izdelavo izdelka, predstavljene tako, da jih premaknemo na nižji nivo v strukturi WBS. Zaporedja operacij in odvisnosti med njimi so določene s povezavami.

Z delovnimi nalogi je določeno, kateri in koliko končnih izdelkov/storitev je zahtevanih. Za izdelavo nekega izdelka je potrebno izvesti določene naloge, podane s proizvodnim postopkom. Glede na zahteve, ki jih podajajo delovni nalogi, se izračunajo tudi potrebe po vhodnih materialih. V našem primeru so potrebe po vhodnih materialih že podane z datoteko *Delivery.txt*. Zaporedje, v katerem so podani vhodni materiali, že vpliva na izvajanje celotnega proizvodnega postopka. To zaporedje predpiše planer. Prek delovnih nalogov so lahko podani tudi dostavni časi in želeni zaključni časi.

Razvrščevalnik deluje v načinu, ko čas napreduje v sklopih. Vsak izdelek (nalogo) v celoti razvrsti na proizvodni plan in nato prične z naslednjim izdelkom [23]. Razvrščevalnik iz datoteke *Delivery.txt* razbere, katere surovine ima na vo-

ljo, ugotovi, kateri izdelek lahko izvede, in zanj prebere vse potrebne podatke v datoteki *Paths.txt*. Odločitev o tem, kateri izdelek naj se izvaja v naslednjem trenutku, povzroči odločitvena funkcija, ki je predpisana vnaprej in je v našem primeru vgrajena znotraj postopka. Na gantogram se naloži naloga, ki izdelek izdela z vsemi pripadajočimi operacijami. Tako je določeno tudi, kateri viri bodo uporabljeni. Vsaka naloga je označena s svojo zaporedno številko. Med gradnjo razvrstitve je treba upoštevati tudi pretok materiala. Zakonitosti, ki jih je treba upoštevati pri uporabi virov, so preprosto modelirane s povezavami med posameznimi operacijami.

Obremenitev virov skozi čas lahko pregledujemo s histogramom virov. Za razrešitev problema preobremenjenosti posameznih virov je na voljo MS Projectova funkcija avtomatske izravnave virov (*Resource levelling*). Tako dobimo razvrstitev, ki je izvedljiva, ni pa nujno, da je tudi optimalna. Pregleden uporabniški vmesnik močno olajša delo planerju in mu tudi omogoča poseganje v dano razvrstitev ter vpeljavo lastnega znanja in izkušenj.

6.5 Primer uporabe na večnamenskem šaržnem procesu

Uporabnost predstavljenega postopka bomo prikazali na že znanem primeru večnamenskega šaržnega procesa. Podrobnejši opis omenjenega procesa je predstavljen v podpoglavju 5.8.

Tudi tu obravnavamo problem, ko imamo podano zahtevo po želeni količini končnih produktov in začetni čas postopka, določiti pa želimo, kdaj naj bodo na voljo vhodne surovine in kdaj se postopek konča. Proizvesti je treba šest šarž 'modrega' in šest šarž 'zelenega' produkta. Sledi, da potrebujemo šest šarž 'rumene' in 'rdeče' surovine in dvanajst šarž 'bele'. Postopek reševanja problema razvrščanja opravil na danem procesu temelji na uporabi orodja MS Project.

Najprej je treba opredeliti proizvodne vire in jih vnesti v bazo MS Projecta. Vnaprej so podani tudi izdelki, ki jih je z danim sistemom mogoče proizvajati. Izdelka sta v našem primeru 'modri' in 'zeleni' produkt. Vsak produkt je mogoče izdelati po različnih proizvodnih poteh. Vsaka od teh proizvodnih poti pomeni izdelek in je podana v datoteki *Paths.txt*. Datoteka vsebuje podatke o tem, katere operacije so potrebne za določitev izdelka, njihove procesne čase in potrebne pro-

```
Y1,15,10,12,V111,V131,V211,R21
Y2,12,09,12,V112,V132,V221,R22
Y3,12,13,12,V113,V133,V231,R23
R1,11,10,13,V121,V131,V212,R21
R2,13,09,12,V122,V132,V222,R22
R3,14,13,12,V123,V133,V232,R23
```

Slika 6.6: Proizvodni podatki, podani z datoteko *Paths.txt*

izvodne vire. Izpis datoteke za obravnavan primer je dan na sliki 6.6. Na primer, prva vrstica definira proizvodni postopek izdelave modre šarže preko reaktorja $R21$. Proizvodni postopek je označen z oznako "Y1", za tem so našteti procesni časi posameznih operacij, in na koncu potrebni viri posameznih operacij. Zadnji vir določa kater izmed treh reaktorjev bo izbran.

Delovne naloge z višjih nivojev vodenja prevzamemo preko datoteke *Delivery.txt*. Količine posameznih vhodnih materialov so določena z delovnimi nalogi. Datoteka je sestavljena tako, da vsebuje zaporedje vhodnih materialov — dostava rumene (Y), rdeče (R) in bele (W) surovine. Dano zaporedje ureja planer in tako že vpliva na generiranje razvrstitve. Kot dodatne omejitve so lahko podane tudi informacije o dostavnih časih posameznih vhodnih surovin. Delni izpis zahtev po vhodnih materialih, ki podajajo obravnavan primer, je predstavljen s sliko 6.7. Ker predpostavljamo da so vhodne surovine vedno na voljo, so vsi dostavni časi enaki 0:00.

Za dane vhodne podatke lahko s pomočjo našega postopka določimo razvrstitev opravil. Glede na dano zaporedje dostav vhodnih surovin se na gantogram nalagajo naloge, ki so potrebne za izvedbo naročila. Vsako nalogo sestavlja več operacij, katerim so predpisani časi izvajanja in potrebni viri. Zakonitosti med posameznimi operacijami so opisane s povezavami med njimi. Operacije, ki so potrebne za izvedbo proizvoda, so med sabo zaporedno povezane s povezavami '*Finish to Start*'. S povezavami pa lahko predpišemo tudi omejitve, ki nastopajo pri uporabi virov. Na primer, ko je reaktor enkrat uporabljen za izdelavo nekega produkta, mora le-tega dokončati, preden je spet na voljo za drugo vhodno

```
Y0:00
R0:00
Y0:00
Y0:00
R0:00
R0:00
W0:00
```

Slika 6.7: Izpisek podatkov a dostavi vhodnih materialov, ki so dani z datoteko *Delivery.txt*

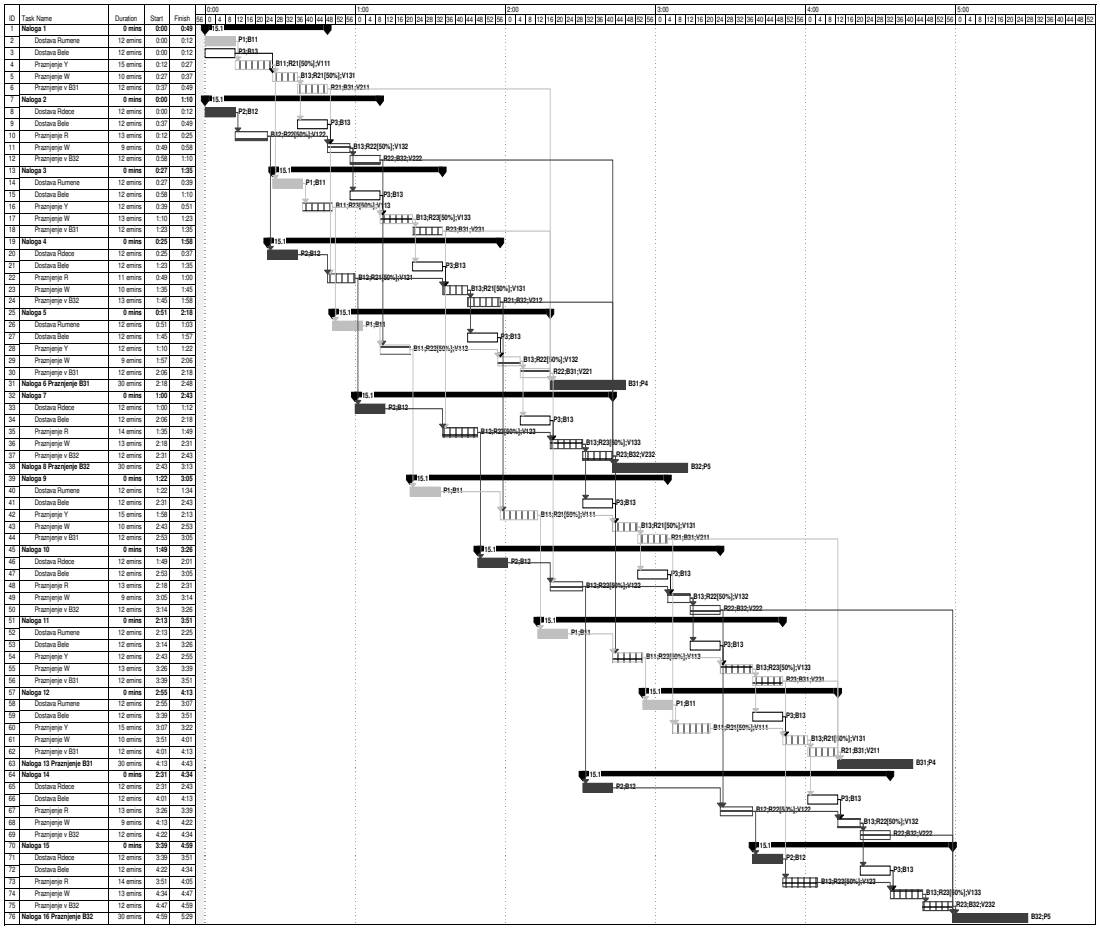
surovino. Enako velja tudi za druge vire, ki shranjujejo surovine in proizvode. Poleg reaktorjev so to tudi vsi rezervoarji, ki nastopajo v obravnavanem šaržnem procesu.

Odločitvena pravila, po katerih se izbirajo viri za izvedbo naročila, so podana znotraj našega postopka. Uporaba vhodnih rezervoarjev pa je predpisana z datoteko *Delivery.txt*. Vsak rezervoar je predpisan za shranjevanje ene od vhodnih surovin. Na voljo pa so trije reaktorji, ki se lahko enakovredno uporabljajo za izvedbo kateregakoli proizvoda. Pravilo izbiranja virov, ki je uporabljeno v našem postopku, določa, da se reaktorji uporabljajo izmenično, drug za drugim (R21, R22, R23,...).

V naš postopek je vgrajen tudi nadzor nad pretokom materiala. Ko je rezervoar, ki shranjuje proizvode, poln, ga je treba pred nadaljnjim izvajanjem postopka izprazniti. Ko so vse naloge naložene, še vedno obstaja možnost, da so kateri od virov v določenih časovnih obdobjih preobremenjeni. V ta namen se uporabi funkcija orodja MS Project, izravnava virov, ki operacije, ki vir preobremenjujejo, časovno prerazporedi.

Kot rezultat razvrščanja dobimo razvrstitev opravil, ki jo prikazuje gantogram na sliki 6.8. Ves postopek bi se po dani razvrstitvi končal v petih minutah in 29 sekundah (329 s). Za preglednejšo predstavitev so pravokotniki, ki pomenijo proizvodne aktivnosti, različno obarvani glede na to, katera surovina se obravnava, in označeni z različnimi vzorci glede na to, kateri reaktor se uporablja. Z

razpoložljivim orodjem lahko nadzorujemo tudi obremenitev virov in po potrebi spreminjamo predlagano razvrstitev. Izvršni čas je ob uporabi tega postopka sicer nekoliko daljši kot z algoritmi, ki smo jih uporabili v primeru predstavljenem v 5.8 podpoglavju, vendar je rešitev vseeno sprejemljiva.

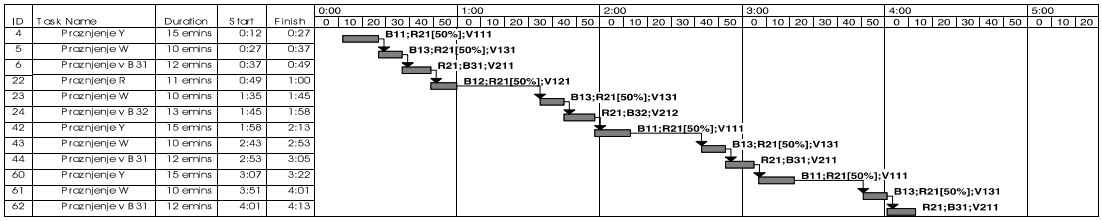


ID	Task Name	Duration	Start	Finish
1	**Naloga 1**	**0 mins**	**0:00**	**0:49**
2	Dostava Rumene	12 emins	0:00	0:12
3	Dostava Bele	12 emins	0:00	0:12
4	Pranjenje Y	15 emins	0:12	0:27
5	Pranjenje W	10 emins	0:27	0:37
6	Pranjenje v B31	12 emins	0:37	0:49
7	**Naloga 2**	**0 mins**	**0:00**	**1:10**
8	Dostava Rdece	12 emins	0:00	0:12
9	Dostava Bele	12 emins	0:37	0:49
10	Pranjenje R	13 emins	0:12	0:25
11	Pranjenje W	9 emins	0:49	0:58
12	Pranjenje v B32	12 emins	0:58	1:10
13	**Naloga 3**	**0 mins**	**0:27**	**1:35**
14	Dostava Rumene	12 emins	0:27	0:39
15	Dostava Bele	12 emins	0:58	1:10
16	Pranjenje Y	12 emins	0:39	0:51
17	Pranjenje W	13 emins	1:10	1:23
18	Pranjenje v B31	12 emins	1:23	1:35
19	**Naloga 4**	**0 mins**	**0:25**	**1:58**
20	Dostava Rdece	12 emins	0:25	0:37
21	Dostava Bele	12 emins	1:23	1:35
22	Pranjenje R	11 emins	0:49	1:00
23	Pranjenje W	10 emins	1:35	1:45
24	Pranjenje v B32	13 emins	1:45	1:58
25	**Naloga 5**	**0 mins**	**0:51**	**2:18**
26	Dostava Rumene	12 emins	0:51	1:03
27	Dostava Bele	12 emins	1:45	1:57
28	Pranjenje Y	12 emins	1:10	1:22
29	Pranjenje W	9 emins	1:57	2:06
30	Pranjenje v B31	12 emins	2:06	2:18
31	**Naloga 6 Pranjenje B31**	30 emins	2:18	2:48
32	**Naloga 7**	**0 mins**	**1:00**	**2:43**
33	Dostava Rdece	12 emins	1:00	1:12
34	Dostava Bele	12 emins	2:06	2:18
35	Pranjenje R	14 emins	1:35	1:49
36	Pranjenje W	13 emins	2:18	2:31
37	Pranjenje v B32	12 emins	2:31	2:43
38	**Naloga 8 Pranjenje B32**	30 emins	2:43	3:13
39	**Naloga 9**	**0 mins**	**1:22**	**3:05**
40	Dostava Rumene	12 emins	1:22	1:34
41	Dostava Bele	12 emins	2:31	2:43
42	Pranjenje Y	15 emins	1:58	2:13
43	Pranjenje W	10 emins	2:43	2:53
44	Pranjenje v B31	12 emins	2:53	3:05
45	**Naloga 10**	**0 mins**	**1:49**	**3:26**
46	Dostava Rdece	12 emins	1:49	2:01
47	Dostava Bele	12 emins	2:53	3:05
48	Pranjenje R	13 emins	2:18	2:31
49	Pranjenje W	9 emins	3:05	3:14
50	Pranjenje v B32	12 emins	3:14	3:26
51	**Naloga 11**	**0 mins**	**2:13**	**3:51**
52	Dostava Rumene	12 emins	2:13	2:25
53	Dostava Bele	12 emins	3:14	3:26
54	Pranjenje Y	12 emins	2:40	2:55
55	Pranjenje W	13 emins	3:26	3:39
56	Pranjenje v B31	12 emins	3:39	3:51
57	**Naloga 12**	**0 mins**	**2:55**	**4:13**
58	Dostava Rumene	12 emins	2:55	3:07
59	Dostava Bele	12 emins	3:39	3:51
60	Pranjenje Y	15 emins	3:07	3:22
61	Pranjenje W	10 emins	3:51	4:01
62	Pranjenje v B31	12 emins	4:01	4:13
63	**Naloga 13 Pranjenje B31**	30 emins	4:13	4:43
64	**Naloga 14**	**0 mins**	**2:31**	**4:34**
65	Dostava Rdece	12 emins	2:31	2:43
66	Dostava Bele	12 emins	4:01	4:13
67	Pranjenje R	13 emins	3:26	3:39
68	Pranjenje W	9 emins	4:13	4:22
69	Pranjenje v B32	12 emins	4:22	4:34
70	**Naloga 15**	**0 mins**	**3:39**	**4:59**
71	Dostava Rdece	12 emins	3:39	3:51
72	Dostava Bele	12 emins	4:22	4:34
73	Pranjenje R	14 emins	3:51	4:05
74	Pranjenje W	13 emins	4:34	4:47
75	Pranjenje v B32	12 emins	4:47	4:59
76	**Naloga 16 Pranjenje B32**	30 emins	4:59	5:29

Slika 6.8: Razvrstitev opravil v šaržnem sistemu, prikazana na gantogramu

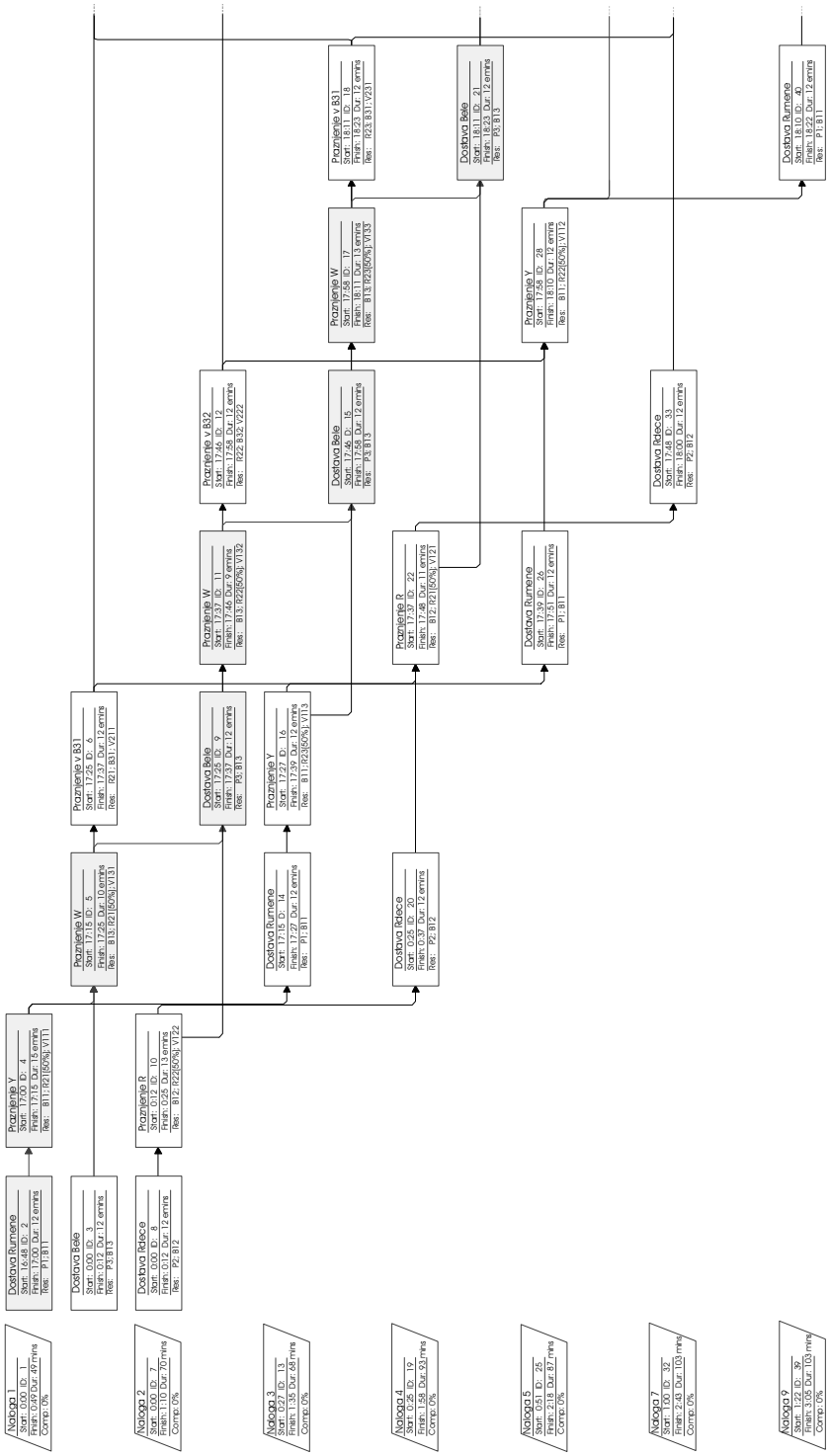
MS Project omogoča tudi pogled, s katerim lahko opazujemo, kako opravila časovno obremenjujejo posamezni vir. Na sliki 6.9 so prikazana vsa opravila, ki obremenjujejo vir R21. Tak pogled je zelo koristen pri nadzoru proizvodnih opravil. Preprosto lahko dobimo vpogled nad opravili, ki so ozko grlo. Z danim orodjem je mogoče operirati le do minute natančno. Ker pa časovna dinamika našega problema zahteva obravnavo v sekundah, smo ga časovno prenormirali iz *mm:ss* v *hh:mm*.

Projekt, zgrajen v MS Projectu pa lahko predstavimo tudi s pomočjo



Slika 6.9: Prikaz časovne obremenitve vira R21

mrežnega diagrama. Del mrežnega diagrama, ki predstavlja plan dela v šaržnem procesu, je predstavljen na sliki 6.10. Na prikazanem diagramu lahko opazimo opravila, ki se nahajajo na kritični poti. Le-ta so predstavljena z osenčenimi pravokotniki.



Slika 6.10: PERT analiza

7. Zaključek

Delo obravnava problematiko razvrščanja v proizvodnih sistemih. Obstaja ogromno tehnik, ki se uporabljajo za reševanja problemov razvrščanja. V delu je predstavljenih več različnih postopkov reševanja različnih problemov razvrščanja.

Za učinkovito analizo proizvodnih sistemov potrebujemo matematični model. Obstaja več načinov, kako matematično formalno predstaviti problematiko, ki se pojavlja na področju proizvodnih sistemov. V delu smo podali pregled matematičnih formalizmov, s katerimi lahko predstavimo problem razvrščanja. Z njimi je omogočeno učinkovito iskanje rešitve – razvrstitve opravil. Predvsem Petrijeve mreže so zelo uveljavljeno matematično orodje na tem področju. Omogočajo kvalitativno analizo diskretno dogodkovnih sistemov, ob vpeljavi časovne informacije pa je mogoča tudi kvantitativna analiza. S tako definiranimi Petrijevimi mrežami je bil predstavljen postopek modeliranja značilnih aktivnosti, ki nastopajo v proizvodnih sistemih. Razvit pa je bil tudi postopek, ki ob uporabi teh sestavnih elementov, zgradi model celotnega proizvodnega sistema. Postopek predvideva, da je sistem za razvrščanje vključen v informacijski sistem podjetja, preko katerega lahko pridobi več koristnih informacij o strukturi in organizaciji proizvodnega postopka. S pomočjo orodja Petri-net Creator je tako zgrajen model mogoče natančno analizirati.

V nadaljevanju pa smo raziskali tudi, kako bi se pri reševanju problematike razvrščanja lahko posluževali orodij za vodenje projektov. Značilno za projekte je, da so si s proizvodnimi procesi podobni v več pogledih. Oboji so sestavljeni iz več procesnih korakov oz. aktivnosti. Struktura izdelka je v proizvodnem okolju podana s kosovnico, medtem ko se za členitev projekta uporablja tehnika WBS. Kot tipično orodje za vodenje projektov smo nekoliko podrobneje analizirali Microsoft Project. Predstavili smo postopek, ki omenjeno orodje vključuje v informacijski sistem podjetja in omogoča detajlno razvrstitev opravil v proizvodnji.

Najpomembnejše prispevke tega dela lahko strnemo v naslednjih točkah:

- Podali smo podroben pregled najpogostejših pristopov k reševanju problemov razvrščanja glede na kompleksnost modela proizvodnega procesa.
- Zasnovali in izdelali smo prototip programskega orodja za opis modela proizvodnega procesa s časovnimi Petrijevimi mrežami. Tak model je nato mogoče uporabiti za namene razvrščanja opravil v obravnavanem proizvodnem sistemu.
- Za orodje Matlab je bila razvita simulacijska funkcija, s katero lahko analiziramo modele predstavljene s časovnimi Petrijevimi mrežami. V simulacijsko funkcijo so vgrajena tudi razna hevristična pravila, funkcionalnost, ki omogoča prirejanja prioritet, poleg tega pa omogoča vgradnjo še raznih drugih pravil.
- Izvedena je bila študija uporabnosti orodij za vodenje projektov za namene razvrščanja proizvodnih opravil. Definirali smo preslikavo elementov, ki se pojavljajo pri upravljanju projektov, na problem razvrščanja. V delu smo preučevali orodje *Microsoft Project*, sicer tipičnega predstavnika orodij za vodenje projektov.
- Izboljšali smo možnost vključevanja postopkov razvrščanja v informacijski sistem podjetja. Poudarek je bil na avtomatizirani gradnji modela iz podatkov, dobljenih s poslovnega nivoja. Razvit je bil postopek avtomatske gradnje modela s Petrijevimi mrežami, ki uporablja podatke iz poslovnih informacijskih sistemov. Podobno je bilo prilagojeno tudi orodje Microsoft Project, kar nam omogoča vključitev orodja v obstoječ informacijski sistem. S pomočjo jezika VBA smo zgradili postopek, s katerim lahko modeliramo proizvodni sistem.

Omenjeni prispevki so bili predstavljeni na različnih praktičnih problemih.

Z nastajanjem te doktorske disertacije so se pojavljali tudi novi problemi in izzivi, ki kažejo možne smeri nadaljnjih raziskav. Model Petrijeve mreže, ki ga dobimo s postopkom avtomatske gradnje, je sicer nazorno predstavljen, vendar zaradi tega precej obsežen. Da bi lahko te modele učinkovito uporabljali za

razvrščanje, bi bila potrebna dodatna obdelava modela, s katero bi odstranili vozlišča, ki sicer ne vplivajo na obnašanje modela. Po drugi stani pa bi bilo potrebno za uporabo še kakšnih drugih pravil razvrščanja model razširiti, tako da bi upoštevali tudi kriterije, ki se ne nanašajo samo na čas izvajanja naloge. V nadaljevanju je mogoča tudi obravnava proizvodnih sistemov, kjer so dovoljene prekinitve. V ta namen bi bilo potrebno uporabiti časovne Petrijeve mreže, ki vpeljujejo časovne zakasnitve po principu časa trajanja omogočanja. Seveda pa je v nadaljevanju smiselna tudi uporaba barvnih Petrijevih mrež, s pomočjo katerih so zapisi modelov kompaktnejši.

Literatura

[1] Y. Abdeddaim, E. Asarin, O. Maler, Scheduling with timed automata, *Theoretical Computer Science*, Vol. 354, No. 2, str. 272–300, 2005.

[2] A.A. Al-Titinchi, K.M. Al-Aubidy, Modeling and Analysis of an On-Line FMS Scheduler Using Colored Petri Nets, *Int. Jour. of Computing & Information Sciences*, Vol. 2, No. 2, str. 74–83, 2004.

[3] R. Alur, D. L. Dill, A theory of timed automata, *Theoretical Computer Science*, Vol. 126, No. 2, str. 183–235, 1994.

[4] ANSI/ISA, *ANSI/ISA-88.01-1995 Batch Control Part 1: Models and Terminology (Formerly ANSI/ISA-S88.01-1995)*, ANSI/ISA, 1995, Spletni naslov: `http://www.isa.org/`.

[5] R.N. Anthony, *Planning and Control Systems: a framework for analysis*, Harvard University, Graduate School of Business Administration, Boston, 1965.

[6] F. Bause, P.S. Kritzinger, *Stochastic Petri Nets, An Introduction to the Theory*, Vieweg Verlag, Braunschweig/Wiesbaden (Germany), 2002.

[7] B. M. Beamon, J. M. Bermudo, A hybrid push/pull control algorithm for multi-stage, multi-line production systems, *Production Planning & Control*, Vol. 11, No. 4, str. 349–356, 2000.

[8] G. Behrmann, E. Brinksma, M. Hendriks, A. Mader, Production scheduling by reachability analysis - a case study, *IPDPS*, Denver, CA, USA, 2005.

[9] G. Behrmann, E. Brinksma, M. Hendriks, A. Mader, Scheduling lacquer production by reachability analysis - a case study, *Proceedings of the 16th IFAC World Congress*, Prague, Czech Republic, 2005.

[10] J. H. Blackstone, D. T. Phillips, G. L. Hogg, A state-of-the-art survey of dispatching rules for manufacturing job shop operations, *International journal of production research*, Vol. 20, No. 1, str. 27–45, 1982.

[11] J. Błażewicz, W. Domschke, E. Pesch, The job shop scheduling problem: Conventional and new solution techniques, *European Journal of Operational Research*, Vol. 93, No. 1, str. 1–33, 1996.

[12] J. Błażewicz, E. Pesch, M. Sterna, The disjunctive graph machine representation of the job shop scheduling problem, *European Journal of Operational Research*, Vol. 127, No. 3, str. 317–331, 2000.

[13] D. E. Blumenfeld, J. Li, An analytical formula for throughput of a production line with identical stations and random failures, *Mathematical Problems in Engineering*, Vol. , No. 3, str. 293–308, 2005.

[14] S. Bogdan, F.L. Lewis, Z. Kovačić, A. Gurel, M. Štajdohar, An implementation of the matrix-based supervisory controller of flexible manufacturing systems, *IEEE Transactions on Control Systems*, Vol. 10, No. 5, str. 709–716, 2002.

[15] M.C. Bonney, Z.M. Zhang, M.A. Head, C.C. Tien, R.J. Barson, Are push and pull systems really so different, *Int. Journal of Production Economics*, Vol. 59, No. 1–3, str. 53–64, 1999.

[16] F. D. J. Bowden, A brief survey and synthesis of the roles of time in Petri nets, *Mathematical and Computer Modelling*, Vol. 31, No. 10-12, str. 55–68, 2000.

[17] D. Cao, M. Chen, A mixed integer programming model for a two line CONWIP-based production and assembly system, *Int. Journal of Production Economics*, Vol. 95, No. 3, str. 317–326, 2005.

[18] J. Carlier, P. Chretienne, Timed Petri net schedules, str. 62–84, Springer-Verlag, Berlin, 1988.

[19] C. Chaouiya, Y. Dallery, Petri net models of pull control systems for assembly manufacturing systems, F. Dicesare, M. Silva, R. Valette, uredniki,

In Procs. of the 2nd Workshop on Manufacturing and Petri nets, ICATPN, Toulouse, France, 1997.

[20] M. A. Cruz-Chávez, F. Ramos-Quintana J. Frausto-Solís, The problem of using the calculation of the critical path to solver instances of the job shop scheduling problem, *International Journal of Computational Intelligence*, Vol. 1, No. 4, str. 334–337, 2004.

[21] C.S. Czerwinski, P.B. Luh, Scheduling products with bills of materials using an improved Lagrangean relaxation technique, *IEEE Transactions on Robotics and Automation*, Vol. 10, No. 2, str. 99–111, 1994.

[22] L. Davis, Job-Shop Scheduling with Genetic Algorithms, *Proc. Int. Conf. on Genetic Algorithms and their Applications*, str. 136–149, Hillsdale, NJ, ZDA, 1985.

[23] S.T. Enns, Finite capacity scheduling systems: performance issues and comparisons, *Computers and Industrial Engineering*, Vol. 30, No. 4, str. 727–739, 1996.

[24] C. Fayad, S. Petrovic, A fuzzy genetic algorithm for real-world job shop scheduling., *IEA/AIE*, str. 524–533, 2005.

[25] P. Freedman, Time, Petri nets, and robotics, *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 4, str. 417–433, 1991.

[26] H. L. Gantt, *Organizing for Work*, Harcourt, Brace and Howe, New York, 1919.

[27] F. Glover, M. Laguna, *Tabu search*, Kluwer Academic Publishers, London, 2001.

[28] D. Gradišar, V. Jovan, Povezava med programsksimi orodji Preactor SMC in GOSOFT 2000, Delovno poročilo, Institut Jožef Štefan, Ljubljana, 2002.

[29] D. Gradišar, G. Mušič, A matlab-based tool for timed Petri nets, *Proc. of VIII Triennial International SAUM Conference on Systems, Automatic Control and Measurements*, str. 267–272, Belgrade, Serbia, 2004.

[30] D. Gradišar, G. Mušič, Razvrščanje proizvodnih opravil z orodji za vodenje projektov, *Elektrotehniški vestnik*, Vol. 71, No. 3, str. 83–88, 2004.

[31] T. Gu, P. A. Bahri, A survey of Petri net applications in batch processes, *Computers in Industry*, Vol. 47, No. 1, str. 99–111, 2002.

[32] J. N. D. Gupta, J. C. Ho, Minimizing flowtime subject to optimal makespan on two identical parallel machines, *Pesquisa Operacional*, Vol. 20, No. 1, str. 5–17, 2000.

[33] J. N. D. Gupta, A. J. Ruiz-Torres, Minimizing makespan subject to minimum total flow-time on identical parallel machines, *European Journal of Operational Research*, Vol. 125, No. 2, str. 370–380, 2000.

[34] N.A.J. Hastings, C-H. Yeh, Bill of manufacture, *Production and inventory management journal*, Vol. 33, No. 4, str. 27–31, 1992.

[35] B. Hauptman, *Razporejanje proizvodnje v procesni industriji*, Doktorska disertacija, Fakulteta za Elektrotehniko, Univerza v Ljubljani, Ljubljana, 2004.

[36] L.E. Holloway, B.H. Krogh, A. Giua, A Survey of Petri Net Methods for Controlled Discrete Event Systems, *Discrete Event Dynamic Systems*, Vol. 7, str. 151–190, 1997.

[37] H. Huang, F.L. Lewis, O.C. Pastravanu, A. Gurel, Flowshop scheduling design in an FMS matrix framework, *Control Engineering Practice*, Vol. 3, No. 4, str. 561–568, 1995.

[38] M.V. Iordache, P.J. Antsaklis, Software tools for the supervisory control of Petri nets based on place invariants, Technical report, ISIS, University of Notre Dame, Notre Dame, IN, ZDA, 2002.

[39] A.S. Jain, J. Meeran, Job-shop scheduling using neural networks, *International Journal of Production Research*, Vol. 36, No. 5, str. 1249–1272, 1998.

[40] K. Jensen, *Coloured Petri nets. Basic concepts, analysis methods and practical use*, Springer-Verlag, Berlin, 1997.

[41] S. M. Johnson, Optimal two- and three-stage production schedules with setup times included, *Naval Research Logistics Quarterly*, Vol. 1, str. 61–68, 1954.

[42] A. Jones, L. Rabelo, *Survey of Job Shop Scheduling Techniques*, National Institute of Standards and Technology, Washington, 1998.

[43] C. V. Jones, The three-dimensional gantt chart, *Operations Research*, Vol. 36, No. 6, str. 891–903, 1988.

[44] V. Jovan, The specifics of production scheduling in process manufacturing, *Elektrotehniški vestnik*, Vol. 69, No. 5, str. 305–310, 2002.

[45] S.J. Kang, S.H. Jang, H.S. Hwang, K.B Woo, Colored timed Petri nets modeling and job scheduling using GA of semiconductor manufacturing, *IEICE Trans. on information and systems*, Vol. E82D, No. 11, str. 1483–1485, 1999.

[46] R. Karba, *Modeliranje procesov*, Založba FE in FRI, Ljubljana, 1999.

[47] T. Kern, *Metode in tehnike projektnega dela : izpis izbranih prosojnic*, Fakulteta za organizacijske vede, Kranj, 2003.

[48] T. Kis, D. Kiritsis, P. Xirouchakis, K.-P. Neuendorf, A Petri net model for integrated process and job shop production planning, *Journal of Intelligent Manufacturing*, Vol. 11, str. 191–207, 2000.

[49] G. Koole, *Optimization of Business Processes: An Introduction to Applied Stochastic Modeling*, 2005, Spletni naslov: `www.math.vu.nl/ koole/obp`.

[50] C. Y. Lee, M. Pinedo L. Lei, Current trends in deterministic scheduling, *Annals of Operations Research*, Vol. 70, No. 1, str. 1–41, 1997.

[51] D. Y. Lee, F. DiCesare, Integrated scheduling of flexible manufacturing systems employing automated guided vehicles, *IEEE Trans. on Industrial Electronics*, Vol. 41, No. 6, str. 602–610, 1994.

[52] D. Y. Lee, F. DiCesare, Scheduling flexible manufacturing systems using Petri nets and heuristic search, *IEEE Trans. on Robotics and Automation*, Vol. 10, No. 2, str. 123–132, 1994.

[53] L.C. Lee, A comparative study of the push and pull production systems, *Int. Journal of Operations and Production Management*, Vol. 9, No. 4, str. 5–18, 1989.

[54] F.L. Lewis, A. Gurel, S. Bogdan, A. Doganalp, O.C. Pastravanu, Analysis of deadlock and circular waits using a matrix model for flexible manufacturing systems, *Automatica*, Vol. 34, No. 9, str. 1083–1100, 1998.

[55] S.-C. Lin, E. D. Goodman, W. F. Punch, Investigating parallel genetic algorithms on job shop scheduling problems, *Evolutionar Programming VI, Proc. Sixth Internat. Conf., EP97*, str. 383–394, Indianapolis, ZDA, 1997.

[56] T. Ljubič, *Planiranje in vodenje proizvodnje: Modeli-Metode-Podatki*, Založba Moderna organizacija v okviru FOV, Kranj, 2000.

[57] E. López-Mellado, Analysis of discrete event systems by simulation of timed Petri net models, *Mathematics and Computers in Simulation*, Vol. 61, No. 1, str. 53–59, 2002.

[58] T. Löscher, F. Breitenecker, D. Gradišar, G. Mušič, A matlab-based tool for timed Petri nets, *Zbornik 14. mednarodne Elektrotehniške in računalniške konference ERK 2005, zv. A*, str. 273–276, Portorož, 2005.

[59] T. Löscher, F. Breitenecker, D. Gradišar, G. Mušič, Timed Petri net simulation in matlab: A production cell case study, *Proc. 5th MATHMOD*, str. 51–56, Vienna, 2006.

[60] M.A. Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis, *Modelling with Generalised Stochastic Petri Nets*, John Wiley and Sons, England, 1995, Spletni naslov: `http://www.di.unito.it/ greatspn/GSPN-Wiley/`.

[61] M. Mascola, J-M. Bollon, A unified framework for describing the dynamics of pull control policies with batch production, *Proceedings of the 16th IFAC World Congress*, Prague, Czech Republic, 2005.

[62] M.H. Matcovschi, C. Mahuela, O. Pastravanu, Petri Net Toolbox for MATLAB, *Proceendings of the 11th IEEE Mediterranean Conf. on Control and Automation MED'03*, Rhodes, Greece, 2003.

[63] P.M. Merlin, A methodology for the design and implementation of communication protocols, *IEEE Trans. on Communications*, Vol. 24, No. 6, str. 614–621, 1976.

[64] Microsoft Inc., *User's Guide for Microsoft Project 2000*, Microsoft, ZDA, 2000.

[65] C.A. Méndez, J. Cerdá, I. E. Grossmann, I. Harjunkoski, M. Fahl, State-of-the-art review of optimization methods for short-term scheduling of batch processes, *Computers & Chemical Engineering*, Vol. 30, No. 6-7, str. 913–946, 2006.

[66] G. Mušič, Vodenje sistemov diskretnih dogodkov, *Elektrotehniški vestnik*, Vol. 66, No. 3, str. 176–186, 1999.

[67] G. Mušič, D. Gradišar, D. Matko, IEC 61131-3 Compliant Control Code Generation from Discrete Event Models, *Proceedings of the 13th Mediterranean Conference on Control and Automation*, str. 346–351, Limassol, Cyprus, 2005.

[68] G. Mušič, B. Zupančič, D. Matko, Petri net based modelling and supervisory control design in Matlab, *Eurocon 2003*, str. 362–366, Ljubljana, 2003.

[69] T. Murata, Petri nets: Properties, analysis and applications, *Proceedings of the IEEE*, Vol. 77, No. 4, str. 541–580, 1989.

[70] B. Murovec, *Preprečevanje neizvedljivosti urnikov pri metahevrističnem razvrščanju proizvodnih procesov*, Doktorska disertacija, Fakulteta za Elektrotehniko, Univerza v Ljubljani, Ljubljana, 2002.

[71] A. L. Nortcliffe, M. Thompson, K. J. Shaw, J. Love, P. J. Fleming, A framework for modelling in S88 constructs for scheduling purposes, *ISA Transactions*, Vol. 40, No. 3, str. 295–305, 2001.

[72] S. S. Panwalkar, W. Iskander, A survey of scheduling rules, *Operations Research*, Vol. 25, No. 1, str. 45–60, 1977.

[73] S. Phadnis, J. Brevick, Irani S., Development of a new heuristic for scheduling flow-shops with parallel machines by prioritizing bottleneck stages, *Journal of Integrated Design and Process Science*, Vol. 7, No. 1, str. 87–97, 2003.

[74] M. Pinedo, *Scheduling: Theory, Algorithms and Systems*, Prentice Hall, 1995.

[75] A. Polič, K. Jezernik, Closed-loop matrix based model, *IEEE Transactions on Industrial Informatics*, Vol. 1, No. 1, str. 39–46, 2005.

[76] B. Potočnik, A. Bemporad, F.D. Torrisi, G. Mušič, B. Zupančič, Hybrid modelling and optimal control of a multi product bach plant, *Control Engineering Practice*, Vol. 12, No. 9, str. 1127–1137, 2004.

[77] Preactor Inc., *Preactor User Guide*, Preactor, Chippenham, UK, 2000.

[78] J.-M. Proth, X. Xie, *Petri Nets: A Tool for Design and Management of Manufacturing Systems*, John Wiley & Sons, Inc., 1996.

[79] C. Ramchandani, *Analysis of Asynchronous Concurrent Systems by Petri Nets*, Doktorska disertacija, Laboratory of Computer Science, MIT, Cambridge, Massachusetts, 1974.

[80] M. Rant, *Operativna priprava proizvodnje*, Moderna organizacija, Kranj, 1988.

[81] A. Ratzer, L. Wells, H. Lassen, M. Laursen, J. Qvortrup, M. Stissing, M. Westergaard, S. Christensen, K. Jensen, CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets, *Proceedings of the 24th ICATPN 2003*, str. 450–462, Netherlands, 2003.

[82] L. Recalde, M. Silva, J. Ezpeleta, E. Teruel, Petri nets and manufacturing systems: An examples–driven tour, G. Rozenberg J. Desel, W. Reisig, urednik, *Lectures on Concurrency and Petri Nets. Advances in Petri Nets*, Volume 3098, str. 742–788, Springer–Verlag, 2004.

[83] B. Roy, B. Sussmann, Les problemes d'ordonnancement avec contraintes disjonctives, *SEMA*, Number 9, Paris, France, 1964.

[84] I. Sabuncuoglu, B. Gurgun, A neural network model for scheduling problems, *European Journal of Operational Research*, Vol. 93, No. 2, str. 288–299, 1995.

[85] M. Silva, E. Teruel, Petri nets for the design and operation of manufacturing systems, *European Journal of Control*, Vol. 3, No. 3, str. 182–199, 1997.

[86] E. Silver, D. Pyke, R. Peterson, *Inventory Management and Production Planning and Scheduling*, John Wiley & Sons, Inc., 1998.

[87] M.L. Spearman, D.L. Woodruff, W.J. Hopp, CONWIP - A pull alternative to kanban, *Int. Journal of Production Research*, Vol. 28, No. 5, str. 879–894, 1990.

[88] M.L. Spearman, M.A. Zazanis, Push and pull production systems - issues and comparisons, *Operations research*, Vol. 40, No. 3, str. 521–532, 1992.

[89] T-H. Sun, C-W. Cheng, L-C. Fu, A Petri net based approach to modeling and scheduling for an FMS and a case study, *IEEE Trans. Industrial Electronics*, Vol. 41, No. 6, str. 593–601, 1994.

[90] D. Tacconi, F. Lewis, A new matrix model for discrete event systems: Application to simulation, *IEEE Transactions on Control Systems*, Vol. 17, No. 5, str. 62–71, 1997.

[91] O.W. Thompson, Production modeling: A full understanding of the business of manufacturing, *Proc. of the American Production and Inventory Control Society*, str. 448–451, Homewood, Il, ZDA, 1987.

[92] Z.J. Tian, C.T. Ng, T.C.E. Cheng, On the single machine total tardiness problem, *European Journal of Operational Research*, Vol. 165, No. 3, str. 843–846, 2005.

[93] W. M. P. van der Aalst, *Petri net based scheduling*, Number 95, Eindhoven University of Technology Computing Science Reports/23, 1995.

[94] W. M. P. van der Aalst, The application of Petri nets to workflow management, *Journal of Circuits, Systems and Computers*, Vol. 8, No. 1, str. 21–66, 1998.

[95] W. M. P. van der Aalst, K. van Hee, *Workflow Management Models, Methods, and Systems*, The MIT Press, 2002.

[96] K. Venkatesh, M.-C. Zhou, M. Kaighobadi, R.J. Caudill, A Petri net approach to modeling and analysis of flexible factory automated systems under push and pull paradigms, *Int. Journal of Production Research*, Vol. 34, No. 3, str. 595–620, 1996.

[97] T.E. Vollmann, W.L. Berry, D.C. Whybark, F.R. Jacobs, *Manufacturing Planning and Control for Supply Chain Management*, McGraw-Hill, New York, 2005.

[98] J. Wijngaard, P. Zijlstra, MRP application the batch process industry, *Production Planning & Control*, Vol. 3, No. 3, str. 264–270, 1992.

[99] J. Wolter, S. Chakrabarty, Methods of kowledge representation for assembly plannig, *Proc. of NSF Design and Manufacturing Systems Conf.*, str. 463–468, Atlanta, Ga, ZDA, 1992.

[100] S. Yang, D. Wang, A new adaptive neural network and heuristics hybrid approach for job-shop scheduling, *Computers and Operations Research*, Vol. 28, No. 10, str. 955–971, 2001.

[101] C.-H. Yeh, Schedule based production, *International Journal of Production Economics*, Vol. 51, No. 3, str. 235–242, 1997.

[102] H. Yu, A. Reyes, S. Cang, S. Lloyd, Combined Petri net modelling and ai-based heuristic hybrid search for flexible manufacturing systems-part i: Petri net modelling and heuristic search, *Computers and Industrial Engineering*, Vol. 44, No. 4, str. 527–543, 2003.

[103] H. Yu, A. Reyes, S. Cang, S. Lloyd, Combined Petri net modelling and ai-based heuristic hybrid search for flexible manufacturing systems-part ii: heuristic hybrid search, *Computers and Industrial Engineering*, Vol. 44, No. 4, str. 545–566, 2003.

[104] W. Zhang, M. Chen, A mathematical programming model for production planning using CONWIP, *Int. Journal of Production Research*, Vol. 39, No. 12, str. 2723–2734, 2001.

[105] M-C. Zhou, A.D. Robbi, Applications of Petri net methodology to manufacturing systems, S. Joshi, G. Smith, uredniki, *Computer Control of Manufacturing Systems*, str. 307–330, Chapman & Hall, 1994.

[106] M. C. Zhou, K. Venkatesh, *Modeling, Simulation and Control of Flexible Manufacturing Systems - A Petri Net Approach*, World Scientific Publishing Co, 1999.

[107] M.C. Zhou, H.-S. Chiu, H.H. Xiong, Petri net scheduling of a flexible manufacturing system using branch and bound method, *Proc. of 1995 IEEE Int. Conf. on Industrial Electronics*, str. 211–216, Orlando, FL, 1995.

[108] A. Zimmermann, D. Rodriguez, M. Silva, Modelling and optimization of manufacturing systems: Petri nets and simulated annealing, *European Control Conference 99*, 1999.

[109] W. M. Zuberek, Timed Petri nets, definitions, properties, and applications, *Microelectronics and Reliability*, Vol. 31, No. 4, str. 627–644, 1991.

[110] W. M. Zuberek, W. Kubiak, Timed Petri nets in modeling and analysis of simple schedules for manufacturing cells, *Computers and Mathematics with Applications*, Vol. 37, No. 11-12, str. 191–206, 1999.

[111] R. Zurawski, MengChu Zhou, Petri nets and industrial applications - a tutorial, *IEEE Trans. on Industrial Electronics*, Vol. 41, No. 6, str. 567–583, 1994.

Izjava

Izjavljam, da sem doktorsko disertacijo izdelal samostojno pod vodstvom mentorja izred. prof. dr. Gašperja Mušiča. Izkazano pomoč drugih sodelavcev sem v celoti navedel v zahvali.

Dejan Gradišar