



Универзитет у Новом Саду
Факултет Техничких Наука
Нови Сад



Кандидат: мр Дарко Чапко

ОПТИМАЛНА ПОДЕЛА ВЕЛИКИХ МОДЕЛА
ПОДАТАКА У ОКВИРУ
НАДЗОРНО-УПРАВЉАЧКИХ
ЕЛЕКТРОЕНЕРГЕТСКИХ СИСТЕМА

докторска дисертација

Ментор: др Александар Ердeљан

Нови Сад, 2012.



УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:	
Идентификациони број, ИБР:	
Тип документације, ТД:	
Тип записа, ТЗ:	
Врста рада, ВР:	Докторска дисертација
Аутор, АУ:	Дарко Чапко
Ментор, МН:	проф. др Александар Ердељан
Наслов рада, НР:	Оптимална подела великих модела података у оквиру надзорно-управљачких електроенергетских система
Језик публикације, ЈП:	српски
Језик извода, ЈИ:	српски
Земља публиковања, ЗП:	Србија
Уже географско подручје, УГП:	
Година, ГО:	2012
Издавач, ИЗ:	
Место и адреса, МА:	
Физички опис рада, ФО: <small>(поглавља/страна)</small>	8 поглавља, 116 страна, 105 цитата, 13 табела, 24 слике
Научна област, НО:	Аутоматско управљање системима
Научна дисциплина, НД:	Дистрибуирано управљање
Предметна одредница/Кључне речи, ПО:	Оптимална подела великих модела података у надзорно-управљачким дистрибутивним системима
УДК	
Чува се, ЧУ:	Библиотека Факултета техничких наука, Универзитета у Новом Саду, Нови Сад
Важна напомена, ВН:	
Извод, ИЗ:	У раду је предложена процедура за поделу великих модела података које описују електроенергетске мреже. Ефикасност прорачуна аналитичких енергетских функција је од највеће важности за надзорно-управљачке електроенергетске система. Предуслов за ефикасне прорачуне је оптимална балансираност оптерећења процесора у оквиру мултипроцесорских система и оптимална подела података по процесорима. Предложени алгоритми за иницијалну поделу и динамичку прерасподелу су примењени на реалним моделима дистрибутивних мрежа.
Датум прихватања теме, ДП:	17. октобар 2011.
Датум одбране, ДО:	
Чланови комисије, КО:	Председник: проф. др Бранко Ковачевић
	Члан: проф. др Душан Петровачки
	Члан: проф. др Горан Швенда
	Члан: проф. др Зоран Јеличић
	Члан, проф. др Александар Ердељан
	Потпис



KEY WORDS DOCUMENTATION

Accession number, ANO :	
Identification number, INO :	
Document type, DT :	
Type of record, TR :	
Contents code, CC :	
Author, AU :	Darko Čapko
Mentor, MN :	prof. dr Aleksandar Erdeljan
Title, TI :	An Optimal Partitioning of Large Data Model in Supervisory Control and Data Acquisition in Power Energy Systems
Language of text, LT :	Serbian
Language of abstract, LA :	Serbian
Country of publication, CP :	Serbia
Locality of publication, LP :	
Publication year, PY :	2012
Publisher, PB :	
Publication place, PP :	
Physical description, PD : <small>(chapters/pages/ref./tables/pictures/graphs/appendix)</small>	8 chapters, 116 pages, 105 references, 13 tables, 24 pictures
Scientific field, SF :	Automation and Control Systems
Scientific discipline, SD :	Distributed control
Subject/Key words, S/KW :	
UC	
Holding data, HD :	Library of Faculty of Tehnical Sciences, University of Novi Sad, Novi Sad
Note, N :	
Abstract, AB :	Dissertation proposes a procedure for data model partitioning of power distribution network. Efficient calculation of the analytical functions is of the utmost importance for the distribution management system; the necessary preconditions for the efficient calculation are optimal load balancing of processors and data model partitioning among processors. The proposed algorithms for initial partitioning and dynamic repartitioning are applied to different real models of power distribution systems.
Accepted by the Scientific Board on, ASB :	October 27 th , 2011
Defended on, DE :	
Defended Board, DB :	
President:	prof dr. Branko Kovačević
Member:	prof dr. Dušan Petrovački
Member:	prof dr. Goran Švenda
Member:	prof dr. Zoran Jeličić
Member,	prof dr. Aleksandar Erdeljan
	Menthor's sign

Садржај

1. УВОД	1
2. НАДЗОРНО-УПРАВЉАЧКИ DMS СИСТЕМИ	5
2.1. <i>SMART GRID</i> СИСТЕМИ.....	5
2.2. ИСТОРИЈСКИ РАЗВОЈ НАДЗОРНО-УПРАВЉАЧКИХ СИСТЕМА У ДИСТРИБУТИВНИМ ЕЛЕКТРОЕНЕРГЕТСКИМ СИСТЕМИМА	7
2.3. РАЗВОЈ АРХИТЕКТУРЕ НАДЗОРНО-УПРАВЉАЧКИХ ДИСТРИБУТИВНИХ СИСТЕМА	9
2.4. ПОРЕЂЕЊЕ ДИСТРИБУИРАНОГ И ЦЕНТРАЛИЗОВАНОГ <i>DMS</i> СИСТЕМА.....	13
2.5. АНАЛИТИЧКЕ ЕЛЕКТРОЕНЕРГЕТСКЕ ФУНКЦИЈЕ.....	14
2.6. ОБЛАСТИ ПРОРАЧУНА.....	16
3. ПОДЕЛА МОДЕЛА ПОДАТАКА ДИСТРИБУТИВНЕ ЕЛЕКТРОЕНЕРГЕТСКЕ МРЕЖЕ	18
3.1. МОДЕЛ ПОДАТАКА У ДИСТРИБУТИВНИМ ЕЛЕКТРОЕНЕРГЕТСКИМ СИСТЕМИМА	18
3.2. МОДЕЛ ПОДАТАКА ЗАСНОВАН НА <i>СІМ</i> СТАНДАРДУ	19
3.3. ДЕФИНИСАЊЕ ПРОБЛЕМА ОПТИМАЛНЕ ПОДЕЛЕ ПОДАТАКА	21
3.3.1. <i>Дефинисање основних појмова</i>	21
3.4. ПРОЦЕС ПОДЕЛЕ МОДЕЛА ПОДАТАКА ДИСТРИБУТИВНЕ ЕЛЕКТРОЕНЕРГЕТСКЕ МРЕЖЕ	23
3.5. ДЕФИНИСАЊЕ ОПТИМИЗАЦИОНОГ КРИТЕРИЈУМА	25
3.5.1. <i>Иницијална подела података</i>	28
3.5.2. <i>Динамичка прерасподела модела података</i>	28
3.5.3. <i>Расподела ненапајаних делова мреже – острва</i>	30
4. АЛГОРИТМИ ЗА ПОДЕЛУ ГРАФОВА	32
4.1. АЛГОРИТМИ ЗАСНОВАНИ НА ПРЕБАЦИВАЊУ ЧВОРОВА.....	34
4.1.1. <i>KL алгоритам</i>	34
4.1.2. <i>FM алгоритам</i>	36
4.2. ПРИРОДОМ ИНСПИРИСАНИ АЛГОРИТМИ.....	37
4.2.1. <i>Генетски алгоритам</i>	37
4.2.2. <i>PSO алгоритам</i>	42
4.3. АЛГОРИТМИ ЗАСНОВАНИ НА ПОДЕЛИ ГРАФА У ВИШЕ НИВОА.....	45
4.3.1. <i>Вишефазни алгоритам за поделу графова</i>	45
4.3.2. <i>Алгоритам суперкорена</i>	48
4.4. ХИБРИДНИ АЛГОРИТМИ	54
4.4.1. <i>Еволутивни PSO (EPSO) алгоритам</i>	54
4.4.2. <i>Хибридни генетски алгоритам</i>	56
5. ДИНАМИЧКА ПРЕРАСПОДЕЛА ГРАФОВА	58
5.1. ИНИЦИЈАЛИЗАЦИЈА ДИНАМИЧКЕ ПРЕРАСПОДЕЛЕ.....	59
5.2. АРХИТЕКТУРА ДИСТРИБУИРАНОГ СИСТЕМА	61
5.3. АЛГОРИТМИ ЗА ДИНАМИЧКУ ПРЕРАСПОДЕЛУ	62
5.3.1. <i>WaveFront (WF) алгоритам</i>	63

5.3.2. Модификовани <i>WF</i> (<i>MWF</i>) алгоритам	67
5.3.3. <i>Cut-and-Paste</i> (<i>CP</i>) алгоритам за прерасподелу графова.....	68
5.3.4. <i>LMSR</i> алгоритам.....	69
5.3.5. <i>PNR</i> алгоритам.....	72
5.4. АЛГОРИТМИ ПРИЛАГОЂЕНИ ВИШЕПРОЦЕСОРСКИМ ХИЈЕРАРХИЈСКИМ АРХИТЕКТУРАМА	74
5.4.1. Хијерархијски <i>MWF</i> алгоритам за прерасподелу графова (<i>HMWF</i>)	75
5.4.2. Хијерархијски <i>CP</i> алгоритам за репартиционисање графова (<i>HCP</i>)	76
6. ЕКСПЕРИМЕНТАЛНИ РЕЗУЛТАТИ.....	78
6.1. ОПИС ТЕСТ МОДЕЛА ПОДАТАКА	78
6.2. Тестирања иницијалне поделе модела података	79
6.2.1. <i>PSO</i> алгоритам и његове модификације	80
6.2.2. Генетски алгоритам и његове модификације	84
6.3.2. Вишефазни алгоритам за поделу графова.....	91
6.3.3. Алгоритам суперкорена	93
6.3.4. Дискусија резултата алгоритама за иницијалну поделу модела података	93
6.3. Тестирања динамичке прерасподеле модела података	99
6.3.1. Тестирање <i>WF</i> и <i>LMSR</i> алгоритма	99
6.3.2. Тестирање <i>MWF</i> и <i>PNR</i> алгоритма.....	101
6.3.3. Поређење <i>WF</i> и <i>MWF</i> алгоритма.....	103
6.3.4. Тестирање алгоритама за динамичку прерасподелу на <i>NUMA</i> рачунару.....	103
6.3.5. Дискусија резултата алгоритама за динамичку прерасподелу модела података	105
7. ЗАКЉУЧАК.....	107
ЛИТЕРАТУРА	110

СПИСАК СКРАЋЕНИЦА

Скраћеница	Енглески термин	Српски термин
AMI	Advanced Metering Infrastructure	Напредна мерна инфраструктура
CIM	IEC 61970-301 Common Information Model	IEC-ов стандард за модел података електроенергетских система
CMS	Customer Management System	Систем за управљање подацима о потрошачима
CP	Cut-and-Paste algorithm	
CPU	Central Processing Unit	Процесор рачунара
DMS	Distribution Management System	Надзорно-управљачки систем за дистрибуцију електричне енергије
EES	Power Energy System	Електроенергетски систем
EMS	Energy Management System	Надзорно-управљачки систем за пренос електричне енергије
EPSO	Evolutionary PSO algorithm	Еволутивни <i>PSO</i> алгоритам
FDIR	Fault Detection, Isolation and Service Restoration	Детекција квара и рестаурација напајања– функција
FM	Fiduccia-Mattheyses algorithm	
GA	Genetic Algorithm	Генетски алгоритам
GIS	Geographic Information System	Географски информациони систем
HAN	Home Area Network	Аутоматизована кућна мрежа
HCP	Hierarchical Cut-and-Paste algorithm	
HMWF	Hierarchical Modified Wavefront algorithm	
HGA	Hybrid Genetic Algorithm	Хибридни генетски алгоритам
IEC	International Electrotechnical Commission	Међународна комисија за стандардизацију у области електротехнике
KL	Kernighan-Lin algorithm	
LF	Load Flow	Токови снага – функција

LMSR	Locally-Matched Multilevel Scratch-Remap algorithm	
MDM	Meter Data Management	Систем за руковање мерачима
MPI	Message Passing Interface	
NR	Network Reconfiguration	Реконфигурација мреже– функција
NUMA	Non-Uniform Memory Access	Мултипроцесорска архитектура са различитим нивоима меморијског приступа
OMS	Outage Management System	Систем за руковање испадима
PI	Performance Indices	Индекс перформанси– функција
PNR	Parallel Nested Repartitioning	Угњеждено паралелно репартиционисање
PSO	Particle Swarm Optimization	Оптимизација ројем честица - алгоритам
RP	Relay Protection	Релејна заштита– функција
RTU	Remote Terminal Units	Удаљена процесна јединица
SCA	Short Circuit Analysis	Анализа кратког споја– функција
SCADA	Supervisory Control and Data Acquisition	Систем за аквизицију података, надзор и управљање
SE	State Estimation	Естимација стања – функција
SG	Smart Grid	Паметне мреже
SM	Smart Metering	Систем за управљање паметним мерачима потрошње електричне енергије
SR	Super-Roots algorithm	Алгоритам суперкорена
TA	Topology Analyzer	Тополошка анализа – функција
VVC	Volt/Var control	Volt/Var управљање – функција
WF	Wavefront algorithm	
WCF	Windows Communication Foundation	

СПИСАК ТАБЕЛА

- Табела 6.1. *Тест модели података*
- Табела 6.2. *Резултати добијени PSO алгоритмом*
- Табела 6.3. *Упоредни резултати секвенцијалног (PSO) и дистрибуираног PSO (DPSO) алгоритма за 1000 итерација*
- Табела 6.4. *Упоредни резултати основног PSO и EPSO алгоритма*
- Табела 6.5. *Поређење генетских алгоритама са различитим бројем елитних јединки*
- Табела 6.6. *Упоредни резултати GA са различитим типовима укритања*
- Табела 6.7. *Резултати GA алгоритма*
- Табела 6.8. *Параметри са којима се извршавају GA, PGA, DGA и DGARP алгоритми*
- Табела 6.9. *Поређење GA, PGA, DGA и DGARP алгоритама*
- Табела 6.10. *Дистрибуирани GA са различитим периодима синхронизације*
- Табела 6.11. *Резултати HGA алгоритма са променљивим вероватноћама мутације и укритања*
- Табела 6.12. *Тестирање различитих варијанти вишефазног алгоритма*
- Табела 6.13. *Резултати SR, HGA и вишефазног алгоритма ($\epsilon=0.1$)*
-

СПИСАК СЛИКА

- Слика 2.1. *Еволуција надзорно-управљачких DMS система*
- Слика 3.1. *SIM модел конективности и топологије*
- Слика 3.2. *Један пример иницијалног модела података у дистрибутивним системима*
- Слика 3.3. *Укрупљени граф за прорачун токова снага*
- Слика 3.4. *Подела моделе дистрибутивне тест мреже на 3 партиције*
- Слика 4.1. *Просто укритање у једној тачки*
- Слика 4.2. *Укритање у једној тачки са нормализацијом*
- Слика 4.3. *Укритање са фаворизацијом најбоље партиције и нормализацијом*
- Слика 4.4. *Вишефазна подела графа на две партиције*
- Слика 4.5. *Вишефазни алгоритам суперкорена*
- Слика 4.6. *Илустрација топологије звезда и стохастичка звезда*
- Слика 5.1. *Промена графа услед затварања потенцијалне конекције*
- Слика 5.2. *Архитектура дистрибуираног система са 4 процесора*
- Слика 5.3. *Пример неповезаног графа са преопретерећеном изолованом партицијом π_2*
- Слика 5.4. *Пример примене LMSR алгоритма*
- Слика 5.5. *Матрица сличности Q и мапирање партиција*
- Слика 5.6. *Стање графа након мапирања*
- Слика 5.7. *Пример NUMA архитектуре*
- Слика 6.1. *Зависност време извршавања секвенцијалног и дистрибуираног (за различите T_{sync}) PSO алгоритма*
- Слика 6.2. *Резултати алгоритама за различите поделе модела bg54*
- Слика 6.3. *Резултати алгоритама за различите поделе модела bg63*
- Слика 6.4. *Резултати алгоритама за различите поделе модела pec106*
- Слика 6.5. *Резултати алгоритама за различите поделе модела it206*
- Слика 6.6. *Резултати алгоритама за различите поделе модела bg5x*
-

1. Увод

Надзорно-управљачки системи су постали интегрални део сваког модерног индустријског система, а међу њима се својом величином и сложеносту истичу системи за надзор и управљање електроенергетских система (*EES*). Њихова улога је да прикупе податке са одговарајуће опреме, добијене податке обраде и на основу резултата обраде генеришу одговарајуће управљачке акције, дају информације о алармима, изврше естимацију стања појединих величина и сл.

Велики значај за анализу и управљања *EES* представља процес моделирања система. У ту сврху су развијени софтверски модели података који описују све елементе (објекте) који се налазе у *EES*. Представљање електроенергетске мреже наведеним моделом података омогућује примену бројних оптимizacionих поступака приликом планирања и симулације система, као у управљање у реалном времену – у току рада *EES*. Вредности (стања) појединих величина се могу добити мерењем у реалном систему или естимацијом стања у оквиру формираног модела. Системи у којима се већи број података добија мерењем, могу се тачније моделовати (тј. врши се боља естимација стања), али се, друге стране, повећавају трошкови таквог система. С обзиром да су у оквиру *EES* за процес снабдевања електричном енергијом од произвођача до крајњег корисника надлежне преносна и дистрибутивна мрежа, значајно је анализирати сличности и разлике у моделима података ових мрежа. Преносне мреже карактерише знатно боља покривеност мерењима и знатно мања количина података. Покривеност мерењима у оквиру дистрибутивних мрежа је веома мала (1-10% од неопходног броја мерених величина за једнозначан прорачун режима мреже), док у савременим дистрибутивним *EES* може постојати више десетина милиона елемената. Због тога је код дистрибутивних *EES* знатно израженији проблем моделирања система и естимације стања, како са аспекта тачности тако и у погледу отежане обраде велике количине података и управљања системом.

Архитектура оваквог управљачког система може се генерално поделити на централизовану и децентрализовану. Класична, централизована архитектура подразумева да се сви подаци налазе у једном рачунару и обрађују од стране једног

или више процесора (ако се ради о мултипроцесорском рачунару). Децентрализована архитектура подразумева да су подаци распоређени у више рачунара, у оквиру дистрибуираног рачунарског система, и да се обрађују претежно од стране локалних процесора. Свака од наведених архитектура има својих предности и недостатака. Централизована архитектура подразумева јефтинији софтвер (једноставније софтверско решење) и скупљи хардвер (коришћење скупих мултипроцесорских суперрачунара), док је код децентрализоване архитектуре сложеније софтверско решење и јефтинији хардвер (више „јефтинијих“ рачунара у потпуности функционално замењују један „скупи“ суперрачунар).

С обзиром на све веће захтеве који се постављају пред овакве системе, повећавају се и количина података коју је потребно обрадити, као и типови података који се користе у обради. Обрада података подразумева бројне прорачуне (тзв. аналитичке енергетске функције) које је потребно извршити да би се ажурирала стања појединих величина, имао свеобухватнији увид у понашање система и, евентуално, генерисале одређене управљачке акције. Међутим, с обзиром на велику количину података и могућност учесталих промена одређених група мерених величина, велико оптерећење представља честа потреба за покретањем прорачуна (нпр. покретање естимације стања одређених група величина). Због тога је неопходно модел података изделити на групе (тзв. партиције) које су независне са аспекта прорачуна, и тако издељене податке паралелно обрађивати.

За поделу модела података са аспекта извршавања различитих енергетских функција [1], које се користе у анализи система, значајан утицај има тополошка структура мреже. Дистрибутивне мреже су радијалне структуре (сваки потрошач се напаја само једним путем са једног извора) са малим бројем контура (слабо упетљане) [1] и са сугестивном поделом одређених група радијалних стабала по партицијама. Са друге стране, преносне мреже нису радијалне структуре и пројектују се тако да обезбеђују редувантне путеве преноса електричне енергије, па стога садрже већи број петљи.

На основу наведених разлика између преносне и дистрибутивне мреже, проблеми поделе великих количина података су значајни у дистрибутивним мрежама, док се код преносних мрежа могу користити само у случајевима када је она велика и ако је отежана анализа преносне мреже (нпр. подела преносне мреже целе Европе, и сл.). Истраживања у оквиру овог рада су била усмерена на поделу

података дистрибутивних мрежа, као недовољно анализираном проблему модерних *EES*.

Податке је неопходно оптимално поделити пре покретања система (иницијално), распоредити их по процесорима (на којим ће се обрађивати), и по потреби вршити прерасподелу података између процесора у току рада система (динамички). Динамичка прерасподела података треба да буде таква да се миграција података изврши оптимално (са аспекта брзине), уз задовољење критеријума оптималности који се користи за иницијалну расподелу (добра повезаност између података унутар партиције, као и балансираност количине података по партицијама).

На основу анализе проблема поделе модела података, дефинисани су циљеви истраживања који су били усмерени у правцу истраживања могућности паралелизације аналитичких електроенергетских прорачуна. У ту сврху било је неопходно анализирати модел дистрибутивне електроенергетске мреже, на основу спроведених анализа формирати критеријум оптималности за поделу модела података и развити адекватне алгоритме који би се могли применити на моделима података реалних електроенергетских система. Да би се остварили наведени циљеви, формално су постављене три хипотезе:

1. Могуће дефинисати оптимизационе критеријуме за поделу модела података дистрибутивних *EES*,
2. Узимајући у обзир дефинисани оптимизациони критеријум, могуће је формирати алгоритам који ће извршити иницијалну поделу модела дистрибутивне електроенергетске мреже.
3. На основу анализе динамичких процеса у оквиру дистрибутивних *EES*, могуће је развити алгоритам адекватних перформанси за динамичку поделу модела података дистрибутивне електроенергетске мреже.

Дисертација пружа доказе постављених хипотезе на основу теоријских образложења и експеримената на реалним моделима података дистрибутивне електроенергетске мреже. Структура рада је организована из седам поглавља:

У наредном поглављу је описан надзорно-управљачки дистрибутивни *EES* (енгл. *Distribution Management Systems – DMS*). Наведен је историјски развој *DMS* система, као и преглед развоја архитектуре наведених система. Анализирани су трендови савремених надзорно-управљачких електроенергетских система и најзначајније компоненте у оквиру таквих система. Поред тога, дефинисане су

основне аналитичке енергетске функције (прорачуни) у циљу свеобухватнијег сагледавања могућности груписања елемената по областима које одговарају потребама за одређене прорачуне.

Процедура за поделу података дистрибутивне електроенергетске мреже је приказана у трећем поглављу. Поред тога, дефинисане су основне карактеристике *CIM* (енгл. *Common Information Model*) модела конективности, као и основни појмови и дефиниције потребне за формулисање проблема поделе електроенергетске мреже. У складу са дефинисаним појмовима, формиран је оптимизациони критеријум за иницијалну поделу модела (приликом покретања система) и динамичку прерасподелу модела. Утврђено је да се проблем поделе модела података у дистрибутивним *EES* може свести на проблем поделе графа.

Четврто поглавље садржи детаљан опис алгоритама за поделу графова са јасно дефинисаним и детаљно описаним корацима (описи сложенијих алгоритама су илустровани и псеудо кодом).

Пето поглавље садржи опис развијених алгоритама за динамичку прерасподелу модела података са наведеним примерима у којима су коришћени поједини алгоритми.

Резултати примене развијених алгоритама на реалним моделима електроенергетске мреже су презентовани у шестом поглављу. Добијени резултати алгоритама за иницијалну поделу су анализирани са аспекта критеријума оптималности, и извршене су упоредне анализе добијених резултата. Алгоритми за динамичку прерасподелу су тестирани у мултипроцесорском окружењу и извршено је детаљно поређење развијених алгоритама у оквиру тестова са „преливањем“ (пребацавањем) података између процесора. У седмом поглављу су дата закључна разматрања овог рада.

2. Надзорно-управљачки DMS системи

Надзорно-управљачки системи у дистрибуцији електричне енергије, познатији под називом *DMS* системи, у последњих 15-так година се намећу као значајан чинилац сваког електроенергетског система. Дистрибутивна мрежа је годинама била запостављана, јер се велика важност придавала искључиво преносној мрежи. Посебна пажња на дистрибутивне системе је кроз историју узрокована великим „испадима“ система након којих по правилу следиле значајније инвестиције у истраживања у области надзора и контроле дистрибутивне мреже. Последњих година се као незаобилазно решење за оптимално коришћење енергије намећу тзв. *паметне мреже* (енгл. *Smart Grid – SG*), у оквиру којих *DMS* системи играју веома важну улогу. Поред тога, тенденција коришћења обновљивих извора енергије (соларни панели, ветрењаче, и сл.) повећава значај дистрибутивних мрежа у оквиру којих се појављују наведени извори.

У циљу формирања потпуније слике о начину функционисање *DMS* система са аспеката који су значајни за поделу модела података, у оквиру овог поглавља ће се анализирати основне компоненте *SG* система, а након тога ће бити описани историјски развој *DMS* система, процес промене архитектуре *DMS* система, најзначајнији енергетски прорачуни и „области“ (скупови елемената) у оквиру модела података над којима се извршавају прорачуни.

2.1. *Smart Grid* системи

Основна идеја *SG* система је ефикасно коришћење енергије (у производњи, преносу, дистрибуцији и потрошњи), уз повећање поузданости, смањење губитака, унапређење процеса заштите животне средине као и висок степен аутоматизације на нивоу целокупног система. Поред тога, у последње две деценије је евидентан значајан раст потрошње електричне енергије [2]: што је довело до развоја свести о интензивнијем коришћењу обновљивих извора енергије (соларне, енергије ветре, мањих хидроелектрана, енергије отпада и биомаса и сл.). Према томе, *SG* су сложени системи са великим бројем софтверских компоненти (пакета) које се односе на следеће подсистеме [3]-[6]:

- Географски информациони систем (енгл. *Geographic Information System – GIS*) – обезбеђује просторне податке о објектима и олакшава идентификовање тражених објеката у реалном систему.
- Систем за руковање испадима (енгл. *Outage Management System – OMS*) – користи се процесу отклањања квара, а задужен је за управљање планираним или непланираним искључењима, која су проистекла утврђивањем квара на реалном систему (увидом у неисправан уређај и обавештавањем диспечера о насталој проблему) или из DMS система коришћењем аналитичких енергетских прорачуна.
- Систем за управљање паметним мерачима потрошње електричне енергије (енгл. *Smart Metering – SM*) – омогућује аутоматско прикупљање и обраду података о потрошњи електричне енергије, чиме је могуће ефикасније организовати поступак наплате [7]. Овај систем се састоји из више подсистема од којих су најзначајнији:
 - напредна мерна инфраструктура (енгл. *Advanced Metering Infrastructure – AMI*) – Овај подсистем садржи напредна електронска бројила (енгл. *Smart Meter*) са меморијом и мрежну инфраструктуру која омогућава преношење података меморисаних у бројилима,
 - систем за управљање мерним подацима (енгл. *Meter Data Management – MDM*) – прикупља податке о утрошеној потрошњи из AMI система, обрађује (нпр. припрема податке систему за наплату, валидира их, процењује потрошњу, генерише неопходне извештаје, и сл.) и управља подацима (прослеђујући их заинтересованим странама), и складишти податке.
- Систем за аквизицију података, надзор и управљање (енгл. *Supervisory Control and Data Acquisition – SCADA*) – задужен је за: прикупљање података (мерења и статуса расклопне опреме) са процесних јединица у систему, складиштење у базу података, приказ на графичком корисничком интерфејсу са кога је најчешће могуће управљање променама појединих величина и сл.
- Аутоматизована кућна мрежа (енгл. *Home Area Network – HAN*) – састоји се из „паметних“ уређаја преко којих је могуће управљати потрошњом (укључивањем/искључивањем) апарата у домаћинству, и аутоматизовати поједине активности у оквиру домаћинства (нпр. активирање алармног

уређаја, пуњење електричног аутомобила и сл.). У оквиру *SG* система се *HAN* интегрише са *SM* системом.

- Подсистем за управљање подацима о потрошачима – (*Customer Management System – CMS*) – првенствено је везан за систем наплате, укључивање и искључивање потрошача, пријаву кварова од стране потрошача, промену тарифног пакета и сл.
- Надзорно-управљачки систем за пренос електричне енергије (енгл. *Energy Management System – EMS*) – добро је покривен мерењима, одликује га велика поузданост, јер у случају испада, у неком делу преносне мреже, велики број корисника остаје без електричне енергија.
- Надзорно-управљачки систем за дистрибуцију електричне енергије (*DMS*) – најчешће му се поверава улога компоненте која повезује све остале подсистеме у оквиру *SG* система из разлога природне повезаности са:
 - *EMS* – преко високонапонских трансформаторских станица. Чвршћа спрега са *EMS* системом омогућује да се разменом информација и прорачуна брже установе потенцијално критични делови у оба система (*EMS* и *DMS*).
 - *SM*, *HAN*, *CMS* – везани су за део дистрибутивне мреже према крајњем потрошачу. Повезивање са *AMI* и *MDM* системом проширује могућности *DMS* система, отвара се могућност праћења потрошње на нивоу крајњих потрошача, даљинско управљање *HAN* системима и ефикаснију обраду података за *CMS* систем, и сл.
 - *OMS* – у циљу брже рестаурацију мреже и повећања поузданости система.
 - *GIS* – преузимање података из *GIS* система утиче на проширење модела података везаних за географске координате и резултате одређених геоинформационих прорачуна.

2.2. Историјски развој надзорно-управљачких система у дистрибутивним електроенергетским системима

Историјски посматрано, почеци управљања електроенергетским системима се могу везати за 20-те године прошлог века када компанија *ABB* обезбеђује свој први уређај за даљинско управљање електранама. Међутим, коришћење рачунара у управљању процесима (први *SCADA* системи) у оквиру преносних

електроенергетских мрежа се јавља 60-тих година прошлог века. Већина тадашњих *SCADA/EMS* система је радила искључиво на једном рачунару и били су намењени једном кориснику. Значајнија улагања у унапређење надзора и управљања у оквиру електроенергетских мрежа су урађена након великог испада система града Њујорка 1977. године, када је велики део мреже био без струје 2 дана. Ово искључење је изазвало нереде широких размера (пљачке, нереде, подметање пожара и сл.), па је довело до озбиљних потреба за аутоматским управљањем у електроенергетским системима. Управо у то време покренут је пројекат од стране *Oak Ridge National Laboratory* из државе Тенеси (САД) у кооперацији са електродистрибуцијама у циљу провере могућности управљања дистрибутивном мрежом [8]. Задатак овог експеримента је био да се провери изводљивост и практичност интегрисаних дистрибутивних система који би у себе укључивали: *SCADA* систем за дистрибуцију, управљање фидерима, аутоматску детекцију квара и рестаурацију мреже, систем за комуникацију, управљање оптерећењем и др. При томе су дефинисани командни контролни центри као места из којих би се путем рачунара управљало целокупном мрежом. Ово је само један из групе пројеката покренутих у то време који се могу сматрати почетком развоја дистрибутивних управљачких система. У то време су *SCADA* функције покривале само подстанице за које то није била велика инвестиција, а представљале су проширење функционалности *SCADA* које су покривале преносне мреже [9].

Термин „*Distribution Management System*“ је установљен средином 80-тих година, а једну од најчешће коришћених дефиниција је формулисао *Cassel* 1993. године [10]. Његова дефиниција гласи: *DMS систем је рачунарски управљачки систем за дистрибутивне контролне центре који поред традиционалних SCADA функције садржи и функције које анализирају тренутно стање дистрибутивне мреже и процењује њено будуће стање. У то време количина података који су покривали дистрибутивну мрежу се процењивала на максимално неколико десетина хиљада. Крајем 90-тих година се појављује већи број издвојених DMS система и од тада се они непрестано проширују и развијају. Количина података у дистрибутивним мрежама са више стотина хиљада долази и до неколико милиона елемената. Два велика „помрачења“ (испада) електроенергетског система у последњих 10-так година (1999. године 70% територије Бразила – око 90 милиона људи без струје; 2003. године у Североисточном делу САД и централној Канади – око 55 милиона људи) усмерила су велике инвестиције у правцу осавремењавања система за снабдевање*

електричном енергијом. Ово је довело до експанзије развоја савремених *SG* система, што у великој мери утиче и на развој *DMS* система. Са појавом оваквих система, стварају се и реалне потребе за обрадама података дистрибутивне мреже која садржи више десетина милиона елемената. Текући трендови у оквиру *SG* система се односе на коришћење обновљивих извора енергије, дистрибутивних генератора, електричних возила, кућне аутоматике, аутоматизованог система наплате, као и складиштење електричне енергије. С обзиром да увођење ових објеката директно утиче на управљање у оквиру дистрибутивне мреже, постављају се захтеви за постојањем свеобухватнијих модела података који ће бити у стању да одговоре постављеним циљевима.

2.3. Развој архитектуре надзорно-управљачких дистрибутивних система

У почетку су дистрибутивни управљачки системи представљали проширење *SCADA* система са елементима који су на прелазу између преносне и дистрибутивне мреже. Такав систем је пратио мали број величина и мали део дистрибутивне мреже око извора напајања (крајње тачке преносне мреже). Управљање оваквим системом су вршили оператери, најчешће на основу искуства. Потпуни надзор над дистрибутивним системом је постигнут развојем *DMS* система. Процес развоја *DMS* система са аспекта архитектуре система, могао би се поделити у неколико фаза [1], [5], [11]-[22]:

1) Библиотека аналитичких електроенергетских прорачуна (функција)

Ова апликација омогућује сложене прорачуне за дистрибутивне електроенергетске мреже (или делове мреже) који су дефинисане од стране корисника. Коришћење апликације има за циљ анализу дистрибутивног електроенергетског система, при чему су корисници уносили комплетан модел података дистрибутивне мреже (креирањем конфигурационог фајла, уносом из командне линије, и сл.) приликом анализе одређене мреже. Ова фаза *DMS* система је замишљена као прелазна фаза у којој би се тестирале аналитичке енергетске функције и установила могућност њихове примене на реалном систему. С обзиром да је такав начин уноса модела дистрибутивне мреже тешко проверљив и подложен грешкама, прешло се на развој корисничкој интерфејса који би олакшао употребу енергетских апликација.

2) Библиотека аналитичких електроенергетских функција интегрисана са корисничким интерфејсом

У овој генерацији система, библиотека аналитичких енергетских функција је интегрисана са богатим корисничким интерфејсом који омогућује ручно уношење мреже. Развијени графички кориснички интерфејс, поред уноса мреже, омогућује праћење резултата прорачуна, избор жељене аналитичке функције, промену неког дела мреже и постављање вредности појединих величина. Модел података дистрибутивне мреже се чува у бази података, и по потреби се може користити и едитовати. Систем (још увек) функционише као изолован, без икакве директне информације о реалним променама вредности појединих величина у систему (ове промене „ручно“ уноси корисник). Овај недостатак је исправљен у следећој генерацији *DMS* система.

3) Повезивање са *SCADA* системом

DMS систем из претходне генерације се (једносмерно) интегрише са *SCADA* системом, на тај начин што се са *SCADA*-е преносе вредности мерења, али се прорачуни одвијају ван реалног времена. Ова генерација система је коришћена за симулације рада система, и у процесу планирања развоја дистрибутивне мреже. С обзиром на немогућност примене аналитичких функција у реалном времену, јавила се потреба за проширењем функционалности постојећег *DMS* система.

4) Чвршћа интеграција са *SCADA* системом

Аналитички енергетски прорачуни се користи *online* и имају за циљ правовремено усмеравање и помоћ кориснику (оператеру) приликом дефинисања одређених управљачких акција. Оператеру се путем корисничког интерфејса приказује текуће естимирано стање мреже, процењује струја укључења/искључења прекидача, управља редоследом манипулације расклопном опремом (најчешће прекидачком опремом). Поред тога, омогућује се и интеграција са *OMS* подсистемом, који се користи се у току отклањања испада или извршавању планираних искључења. Ову фазу развоја *DMS* система одликује рад у блиско реалном времену са обogaћеним корисничким интерфејсом чиме се оператеру помаже у доношењу већине одлука о управљању. Изузетак су управљачке акције које се спроводе од

стране SCADA-е на основу унапред дефинисаних правила, нпр. распоређивање оптерећења (енгл. *Load shedding*) по унапред дефинисаним групама потрошача, укључивање/искључивање уличне расвете и сл.

5) **Управљање у реалном времену са затвореном повратном спрегом**

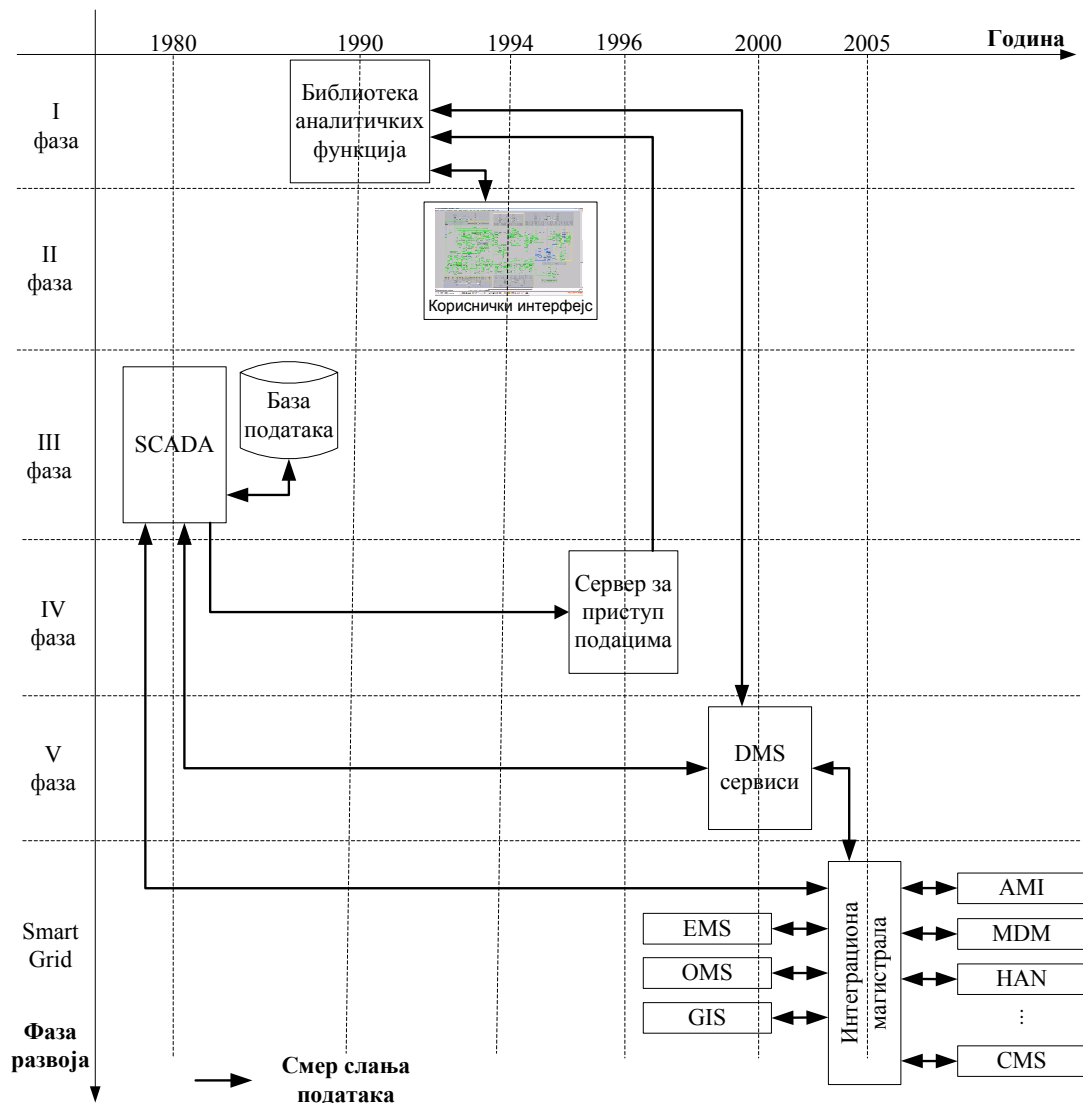
Системи ове генерацију представљају проширење система из претходне генерације, са значајним изменама у погледу управљања. У оквиру ових система, управљање са затвореном повратном спрегом се спроводи уз коришћење прорачуна везаним за оптималну регулацију напона, као и функција које се користе за детекцију квара у мрежи и рестаурацију напајања ненапојених делова мреже. Код ових система се централизован управљачки алгоритам спроводи аутоматски у командно контролном центру, и то по следећим правилима:

- у складу са управљачким уређајима дистрибуираним у мрежи;
- управљачки алгоритам се извршава на основу детаљног модела и бројних мерења;
- оператери прате рад у аутоматском режиму и могу да промене начин рада (критеријум по коме се врши управљање) и да укључе/искључе аутоматски режим рада.

У оквиру ових система, управљање се односи на: управљање напоном у дистрибутивној мрежи ради смањења потрошње, смањење потрошње управљањем паметним уређајима (веза *DMS – MDM – AMI – HAN*) у оквиру SG система, управљање дистрибутивним генераторима и др.

На слици 2.1. је приказана еволуција надзорно-управљачког *DMS* система са свим наведеним развојним фазама.

Имајући у виду проблеме са којима се сусреће *DMS* систем приликом извршавања енергетских прорачуна над великом количином података у реалном времену, истраживања спроведена у оквиру ове дисертације показала су да постоје реалне потребе за надоградњу система из последње генерације. Измене које предлажемо се односе на систем са дистрибуираним моделом података, у оквиру кога се неопходни енергетски прорачуни извршавају паралелно.



Слика 2.1. Еволуција надзорно-управљачких DMS система

б) Дистрибуирани систем – будућа архитектура

Дистрибуирани систем би подразумевао да су подаци расподељени по различитим процесорима (рачунарима). Сервиси који су постојали у претходној генерацији система, прилагођавају се за дистрибуирани рад и извршавају се паралелно на више рачунара. Сваки од сервиса је задужен за своје „парче“ података и сви аналитички прорачуни се извршавају над локалним подацима. Један од најважнијих задатака оваквог система је оптимална подела података по процесорима у циљу ефикаснијег рада система и повећања брзине добијања резултата аналитичких функција. Истраживања која су спроведена и описана у овом раду, заснована су на формирању правила и развоју алгоритама за оптималну поделу

великих количина података у циљу расподеле модела дистрибутивне мреже у оквиру дистрибуираног рачунарског система.

Овакав систем поред многих предности у односу на централизоване системе има и одређених недостатака. У циљу свеобухватније анализе разматраног проблема следи преглед упоредних карактеристика дистрибуираног и централизованог система.

2.4. Поређење дистрибуираног и централизованог DMS система

Величине данашњих дистрибутивних мрежа превазилазе могућности једног рачунара, како са аспекта меморијског заузећа тако и са аспекта сложености прорачуна (нпр. мрежа која има више десетина милиона елемената се у централизованим верзијама DMS система учитава комплетна у реал-тима режиму рада). За похрањивање великих мрежа у централизованим системима неопходно је поседовати моћне рачунаре чија је цена прилично висока. Поред тога, снажни рачунари су неопходни и због сложених прорачуна. Имајући у виду да су у дистрибуираним системима подаци распоређени на више рачунара (процесора), следи да је у дистрибуираним системима мање меморијско заузеће по сваком рачунару (процесору) у односу на централизоване податке.

Основна предност дистрибуиране обраде огледа се у чињеници да је снага система одређена збиром снага појединачних рачунара, док је код централизованог решења она ограничена снагом централног рачунара.

Могућност проширења дистрибутивне мреже ствара проблем перформанси код централизованог система, док је дистрибуирани систем скалабилан – практично нема ограничења на повећање мреже (додавањем рачунара се, без измене софтвера, повећава рачунарска снага система и могућност „увлачења“ већег броја података). Уколико се разматра аспект сигурности и поузданости система, отпорност на грешке у дистрибуираном систему је далеко већа, што доводи до закључка да су такви системи практично "неуништиви". Код централизованих система, након испада примарног (*master*) рачунара, а затим и резервног (*backup*) рачунара, систем престаје да ради!

У циљу лакшег одржавања, стабилнијег рада, мањег хабања, веће отпорности на грешке и дужег животног века неопходно је равномерно оптеретити ангазоване процесоре. Поред тога, све учесталији су захтеви корисника DMS софтвера за конзервативнијим коришћењем постојећих ресурса (меморије и процесорског

времена), чиме би се оставиле тзв. стратешке резерве (у пракси су чести захтеви за резервом од 50%). Уколико се овакве резерве остављају у централизованом систему, то доводи до додатног поскупљења рачунари чија је цена свакако висока, а њихови (скупи) ресурси остају неискоришћени. Поред тога, у системима са чврстим временским ограничењима, када постоји велики број задатака, систем се не би смео оптеретити преко 69% [23], иначе постаје нераспоредив. Сви наведени захтеви иду у прилог коришћењу дистрибуираног система.

Поред наведених предности дистрибуирани системи имају и неке недостатке у односу на централизоване. У недостатке би се могле сврстати следеће карактеристике:

1. Сложеније софтверско решење, али технички изводљиво:
 - Потребно је податке иницијално оптимално поделити на чворове (процесоре);
 - Подаци би требало да се оптимално „преливају“ између чворова;
 - Неопходно је омогућити размену података између чворова код прорачуна где се користе подаци целе мреже (нпр. приликом извршавања дистрибуираног алгоритма тополошке анализе).
2. Потребан је дистрибуиран алгоритам тополошке анализе мреже.

2.5. Аналитичке електроенергетске функције

У оквиру дистрибутивних надзорно-управљачких система се прикупљају подаци са терена преко *SCADA* система и врше прорачуни одговарајућих аналитичких електроенергетских функција. Резултати аналитичких функција омогућују анализу стабилности система, утврђивање и лоцирање кварова у систему, као и симулацију и оптимално подешавање система. На основу резултата појединих аналитичких функција се врше управљачке акције и мењају стања опреме посредством *SCADA*-е. Вредности величина и измене модела података дистрибутивне мреже се похрањују у бази података.

У основне аналитичке прорачуне спадају [1], [24]-[26]:

- *Тополошка анализа* (енгл. *Topology Analyzer – TA*) – утврђује топологију дистрибутивне мреже и повезаност проводне опреме. Заснована је на формирању графа из модела конективности који представља дистрибутивну мрежу и провери статуса расклопне опреме (прекидача, растављача, и др.).

- *Токови снага* (енгл. *Load Flow – LF*) – користе се за прорачуне стационарних стања, најчешће за радијалну, слабо повезану дистрибутивну мрежу. Могу се користити за прорачуне монофазних и трофазних стања токова снага. Резултати ове функције се користи у многим другим функцијама (као што су естимиција стања, *Volt/Var* управљање и др.) [27], [28].
- *Естимација стања* (енгл. *State Estimation – SE*) – израчунава се процена стања дистрибутивне мреже (процена оптерећења свих мрежних чворова, напоне и струје на свим сабирницама, деоницама и трансформаторима, губитака активне и реактивне снаге, итд.). Користи резултате *LF* прорачуна, а многе друге аналитичке функције су засноване на њеним резултатима (Детекција квара, Релејна заштита, Рестаурација мреже, итд.)
- *Индекс перформанси* (енгл. *Performance Indices – PI*) – одређује индексе перформанси дистрибутивне мреже (дела или целе мреже). Ова функција утврђује: прекорачења струја и напона, губитке активне снаге, стабилно распоређивање оптерећења између трансформатора (високи/средњи напон), распоређивање оптерећења између фидера, очекиване годишње вредности испоручене/неиспоручене електричне енергије, и др.[25]
- *Детекција квара и рестаурација напајања* (енгл. *Fault Detection, Isolation and Service Restoration– FDIR*) – утврђује квар у мрежи на основу даљинских мерења преко процесних јединица, лоцира и изолује тај део мреже, и обезбеђује напајање ненапајаним деловима мреже.
- *Реконфигурација мреже* (енгл. *Network Reconfiguration – NR*) – користи се за одређивање оптималне конфигурације мреже (која се карактерише отвореношћу расклопне опреме у нормалним режимима рада). При томе се постављају различити оптимизациони критеријуми, као нпр. минимизација губитака активне и реактивне снаге, максимална поузданост, најбоље балансирање оптерећења, најбоље напонске карактеристике, и сл. Ова функција се најчешће изводи на годишњој или сезонској основи, али је ово често грубо и недовољно за реализацију пуних ефеката *NR* функције [1], [25].
- *Анализа кратког споја* (енгл. *Short – Circuit Analysis – SCA*) – ова функција се користи за израчунавање струје кратког споја у симулационим режимима рада. Процењују се могући утицаји хипотетичких грешака на мрежу,

проверава се стање релејне заштите и препоручују нова подешавања за релеје или конфигурација мреже.

- *Релејна заштита* (енгл. *Relay Protection – RP*) – управља и проверава подешавање релеја у дистрибутивним фидерима под различитим условима и конфигурацијама мреже.
- *Volt/Var управљање* (енгл. *Volt/Var control – VVC*) – користи се за израчунавање оптималног напона напајања трансформатора - одређује се оптималан положај регулатора трансформатора [29].

Поред наведених постоји још велики број потенцијално значајних аналитичких функција, али се оне често укључују само на захтев корисника дистрибутивног надзорно-управљачког система.

2.6. Области прорачуна

Аналитичке електроенергетске функције се извршавају у различитим режимима рада система (*online*/симулација), а могу се извршавати на захтев корисника, периодично, или након неког догађаја у систему (као што су: промена стања расклопне опреме, промена вредности мерења, стања регулационе склопке и др.).

На основу анализе рада аналитичких функција утврђено је да се прорачуни врше са подацима у оквиру елемената који су напајани из истог извора енергије. Скуп елемената напајаних из истог извора се назива *корен* [30].

Према томе, прорачуни аналитичких функција се могу вршити паралелно по коренима чиме се знатно смањује време потребно за прорачуне наведених функција [31]-[37]. Ово убрзање се може повећати упошљавањем већег броја процесора (језгара), при чему чак и вишеструки прорачуни у оквиру једног корена могу бити подељени на више процесора. Тако нпр. функција *VVC* се изводи над свим (или одабраним) коренима у дистрибутивној мрежи, при чему је неопходно извршити вишеструке *LF* прорачуне у оквиру једног корена. Ти прорачуни су међусобно независни у току одређеног броја извршавања *LF* функције над једним кореном (нпр. првих 100 итерација *LF* прорачуна над неким кореном су међусобно потпуно независни), тако да се могу извршавати на независним процесорима. У складу са тим, укупан процес прорачуна се може убрзати онолико пута колико се процесора при том користи.

Уколико се ради о дистрибуираном систему, тада се паралелни прорачуни

извршавају на различитим чворовима над посебно груписаним деловима мреже које називамо *партиције*. На тај начин се смањује меморијско заузеће на појединачним чворовима. Идеја је поделити мрежу на делове – корене који се групишу у партиције. То је природна подела мреже (где се постиже независност њених делова) која омогућава да се прорачуни више корена спроводе паралелно (на засебним процесорима и/или рачунарима). Оваквим приступом је омогућено истовремено (за више корена одједном), независно покретање аналитичких функција (*LF*, *SE*, *VVC*, и др.).

Наведене функције се потпуно извршавају над једним кореном (нпр. *LF*, *SE*, *VVC*, и др.) или се извршавају над суседним коренима (нпр. *NR*). Због тога је потребно формирати партиције да се у њима налазе међусобно зависни (повезани) корени. Квантитативна мера повезаности корена може бити број отворене расклопне опреме између корена (тзв. граничних прекидача – енгл. *tie switches*). Тако на пример, ако између два корена R_x и R_y постоји веза преко 3 отворена гранична прекидача, а између корена R_x и R_z постоји веза преко једног отвореног граничног прекидача, сматра се да су корени R_x и R_y боље повезани од корена R_x и R_z .

3. Подела модела података дистрибутивне електроенергетске мреже

Тренд да се дистрибутивни електроенергетски системи повежу са другим системима у оквиру паметних мрежа утиче на проширење функционалности *DMS* система, али и проширење модела података дистрибутивне мреже. Ово проширење се односи не само на повећање количине података (који су по типу већ присутни у моделу), већ и на повећање грануларности (увођење нових типова података) [5], [38] – нпр. увођењем „паметних“ мерача у оквиру *AMI* подсистема, управљање коришћењем енергије у оквиру кућних система (*HAN* подсистем), и др. На тај начин се добија већа количина података са терена, што повећава оперативну ефикасност *DMS* система, и обезбеђује се више података за остале подсистеме у оквиру *SG* система. Интеграција са *GIS* системом подразумева потребу за проширењем модела са аспекта географских података, а појава нових (обновљивих) извора енергије у дистрибутивним мрежама [39] може у великој мери да промени тополошку структуру мреже и повећа комплексност већине аналитичких енергетских прорачуна. Поред тога, у оквиру *SG* система ће се поједине аналитичке функције (нпр. *TA*, *LF*, *NR*, *SCA* и *RP*) користити знатно чешће и биће проширене на део нисконапонске мреже до индивидуалних потрошача.

На основу свега наведеног може се закључити да се у оквиру модерних *SG* система количина података, која је у надлежности *DMS* система, знатно повећава, а аналитичке електроенергетске функције усложњавају и фреквентније извршавају. Интеграција и комуникација са осталим подсистемима у оквиру *SG* система додатно оптерећује *DMS* систем, па је због тога неопходно пронаћи решење које би растеретило систем у условима учесталог додатног оптерећења. Најприхватљивије решење је да се прорачуни паралелизују, а модел података подели на делове над којима би се прорачуни независно извршавали.

3.1. Модел података у дистрибутивним електроенергетским системима

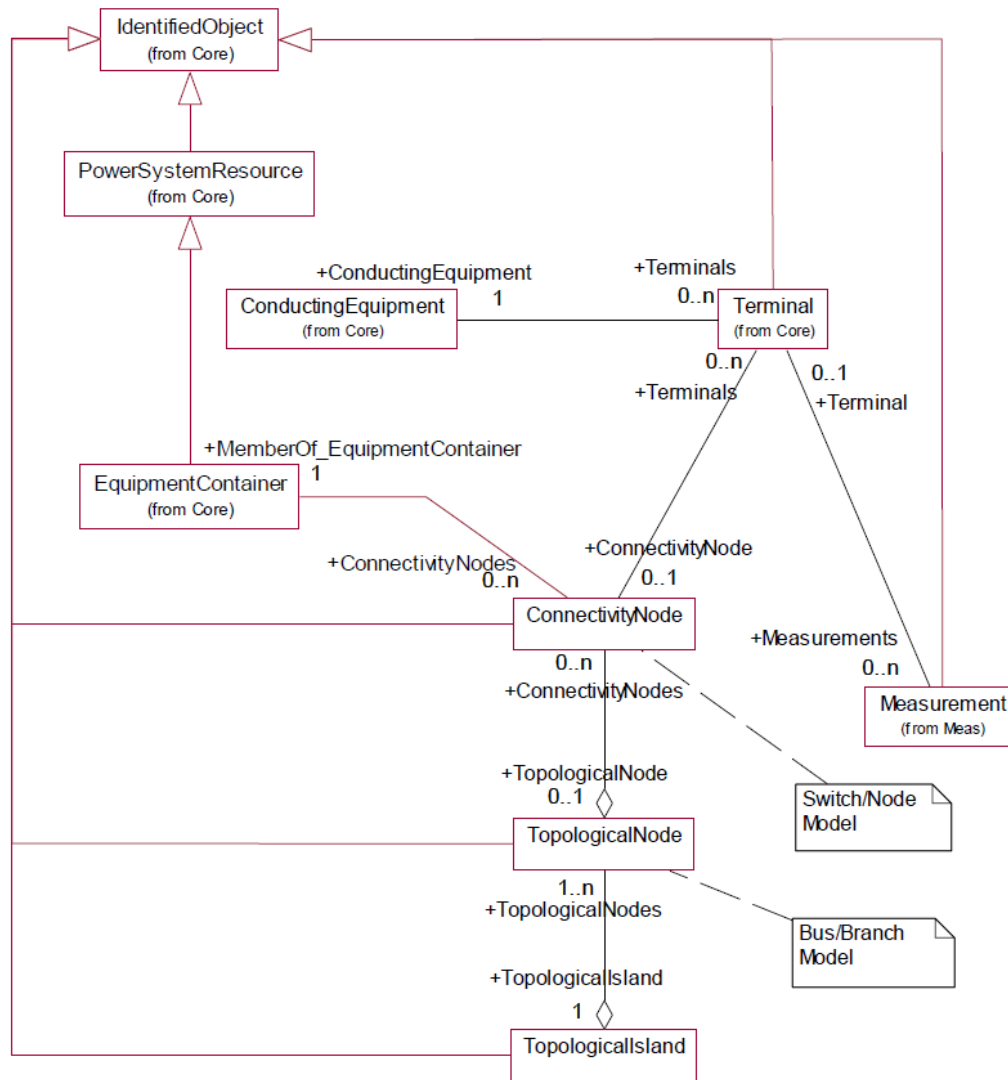
Тенденција за интеграцијом са осталим системима и оквиру *SG* решења доводи до потребе да се подаци размењују по правилима која су утврђена стандардима. Са

друге стране, модел података треба да подлеже тим правилима у циљу ефикаснијег рада система. У супротном би податке приликом размене требало пребацивати у стандардизован облик – што би довело до непотребне потрошње ресурса и времена. Због тога се у електроенергетским системима користе стандарди за дефинисање података који описују елементе електроенергетске мреже. Најраспрострањенији и општеприхваћени стандард за опис модела података је *CIM* стандард. Установљен је од стране *IEC* (енгл. *International Electrotechnical Commission*) комисије и дефинисан као део стандарда *IEC 61970-301* [40]. У овом стандарду су дефинисани основни ентитети који се појављују у електроенергетским системима (првенствено је намењен за преносне мреже), а касније је установљен стандард *IEC 61968-11* [41] који представља проширење модела са ентитетима који су специфични за дистрибутивне системе. Поред тога, дефинисани су и стандардни интерфејси за размену података – попут *GDA* (енгл. *Generic Data Access*) стандард *IEC 61970-403* [42]) који се ослања на *DAF* (енгл. *Data Access Facility*) спецификацију развијену од стране *OMG* групе [43], [44]. На тај начин је створена основа која прописује чување података и размену између интегралних компоненти у оквиру *SG* система. У оквиру овог рада се разматра подела великог модела података дистрибутивне мреже који је заснован на *CIM* стандарду [45].

3.2. Модел података заснован на *CIM* стандарду

CIM модел је апстрактни објектни модел који представља све најважније ентитете електроенергетске мреже (преносне и дистрибутивне) и релације између њих. Модел података дистрибутивне мреже има претежно радијалну [46] и слабо упетљану структуру, мада се врло често за веће градове (обично са америчког континента) дистрибутивна мрежа више упетљава (чиме се повећава покривеност снабдевања у случају „испада“ неког дела мреже, али се анализа система и управљање знатно компликују).

На слици 3.1. је приказан упрошћен *CIM* модел конективности (изграђености) и топологије мреже који је најзначајнији за анализу поделе модела података са аспекта прорачуна. Он креће од високонапонских трансформатирских станица (*Substation*) које су напајане из извора напајања (*EnergySource*) - места која добијају енергију из преносне мреже.



Слика 3.1. CIM модел конективности и топологије [40]

Глобално посматрано, модел се састоји из трансформаторских станица (*Substation* објекат наслеђује *PowerSystemResource*) које су међусобно повезане водовима (*ACLineSegment*) и напајају одређену групу потрошача (*EnergyConsumer*). Трансформаторске станице садрже опрему (*Equipment*) и чворове конективности (*ConnectivityNode*) који преко терминала (*Terminal*) повезују дату опрему. Најзначајнији типови опреме су трансформатори (*PowerTransformer*) и проводна опрема (*ConductingEquipment*). Типични елементи проводне опреме су: сабирница (*BusbarSection*), извор напајања (*EnergySource*), потрошач (*EnergyConsumer*), расклопна опрема (*Switch* – са основним подтипovima: прекидач - *Breaker*, растављач – *Disconnecter* и осигурач - *Fuse*), електрични вод (*ACLineSegment*) и др.

Сви објекти (елементи) наслеђују основни објекат *IdentifiedObject* који садржи основна обележја сваког ентитета: јединствени идентификатор, име и тип.

Два објекта (елемента типа опреме) су суседна ако референцирају исти чвор конективности (на неком свом крају имају исти чвор конективности). У суштини, модел конективности је чвор – грана оријентисан и због тога је погодан за представљање графом, при чему су гране и чворови инстанце типа *Equipment* и *ConnectivityNode*, респективно.

На основу модела конективности формира се тополошки модел код кога су сви суседни чворови конективности који су на истом потенцијалу представљени једним логичким – тополошким чвором. Тополошки модел зависи од статуса расклопне опреме – који директно утиче на формирање тзв. тополошких чворова (*TopologicalNode* – састоје се од проводне опреме и чворова који су на истом потенцијалу). Модел топологије по *CIM* стандарду се користи приликом извршавања прорачуна, груписања података који су значајни за кориснике система, и сл. *CIM* тополошки модел дефинише и тзв. *тополошка острва* (*TopologicalIsland*) – групу галвански повезаних елемената која могу бити напојена (одговарају раније дефинисаним коренима) или ненапојена (тзв. острва).

3.3. Дефинисање проблема оптималне поделе података

Задатак оптималне поделе података је да се сви подаци неопходни за један прорачун сместе у исту меморију чиме би се оптимално извршили прорачуни у систему. Уколико се ради о малој количини података, циљ је аутоматски задовољен складиштењем свих података у меморију једног рачунара. Проблем се усложњава уколико се ради о великој количини података (која се отежано и споро обрађује на једном рачунару). У циљу дефинисања оптимизационог проблема неопходно је дефинисати коришћене појмове.

3.3.1. Дефинисање основних појмова

На основу описаног модела података, потребно је дефинисати основне појмове у циљу формирања критеријумске функције за оптимизациони проблем поделе модела података. Ради једноставности, приликом дефинисања оптимизационог проблема ће се разматрати само једна аналитичка функција - ξ . Велики број дефиниција основних појмова је раније публикован у [30], [47], [48].

Претпоставка је да се скуп улазних величина U може поделити на два подскупа (две врсте промена улазних величина) $U = U_r \cup U_d$:

- U_r – скуп промена величина које утичу на релације између елемената у тој мери да узокују промену топологије мреже, мењају области прорачуна, па самим тим и иницирају покретање прорачуна (промена статуса расклопне опреме),
- U_d – скуп промена величина које утичу на покретање појединих прорачуна (промена мерења, промена стања регулационе склопке и сл.).

Дефиниција 1.: Под *елементом* α_i се дефинише опрема која поседује одређене атрибуте. Елемент је улаз у разматрану аналитичку функцију ξ (нпр. LF функцију).

Дефиниција 2.: Велики модел података Q је скуп елемената и релација између њих.

Дефиниција 3.: Ако су два елемента у релацији $N(\alpha_i, \alpha_j)$ тада се претпоставља да се они заједно користе у прорачунима за функцију ξ (разматрају се само релације које су значајне за функцију ξ). Тада су елементи повезани или *суседи*.

Два елемента (α_i и α_j) су повезана ако се сукцесивним кретањем по суседима може доћи од елемента α_i до елемента α_j , а да се при томе не пролази преко расклопне опреме која не проводи струју (нпр. отворених прекидача).

Дефиниција 4.: Релација суседности између елемената може бити привремена и зависити од њиховог стања (које се може споља мењати). Уколико релација између суседа зависи од улазне величине $u_{ij} \in U_r$ таква релација се назива *потенцијална конекција* $Pot(\alpha_i, \alpha_j, u_{ij})$. Другим речима, нпр. када се потенцијална конекција активира – два елемента постају суседи:

$$(\forall \alpha_i, \alpha_j \in Q) Pot(\alpha_i, \alpha_j, u_{ij} = active) \Rightarrow N(\alpha_i, \alpha_j) \quad (1)$$

Према томе, потенцијална конекција је релација која се карактерише стањем (активан и неактиван). Ако је стање активно онда су елементи повезани и они се морају заједно користити у прорачуну. У супротном, ако је стање неактивно, елементи нису повезани. Тако нпр. ако се разматра функција LF , тада се потенцијална конекција везује за прекидач (или растављач), затворени прекидач (растављач) представља активно стање, а отворени неактивно стање.

Дефиниција 5.: Скуп међусобно повезаних елемената се назива *област прорачуна* (или *регион*):

$$(\forall \alpha_i \in R_k) N(\alpha_i, \alpha_j) \Leftrightarrow \alpha_j \in R_k, \quad k \in \{1, 2, \dots, n\} \quad (2)$$

што је минималан скуп података неопходан за прорачун ξ . Област прорачуна за LF функцију је корен. Корен је напојени регион, а ненапојени регион се назива *острво*. Наведена дефиниција региона одговара дефиницији тополошког острва по *СІМ* стандарду. Острва се не разматрају директно приликом поделе модела података, већ се накнадно расподељују по процесорима.

Дефиниција 6.: Све области прорачуна чине *домен прорачуна* D :

$$D = R_1 \cup R_2 \cup \dots \cup R_n , \quad (3)$$

што значи да се функција $Y_i = \xi(R_i)$ примењује на сваки регион R_i и добијају се излазни резултати Y_i који се могу бити део излазног скупа резултата

$$Y = Y_1 \cup Y_2 \cup \dots \cup Y_n . \quad (4)$$

3.4. Процес поделе модела података дистрибутивне електроенергетске мреже

Процедура за поделу модела дистрибутивне мреже се састоји из следећих фаза:

1. Формирање иницијалног графа – графа конективности (на основу модела конективности између елемената проводне опреме и трансформатора)
2. Тополошка анализа и креирање графа за прорачуне (укрупњеног графа код кога чворови представљају корене и називају се *чворови прорачуна*)
3. Подела укрупњеног графа

1. Формирање иницијалног графа

Иницијални граф је формиран од проводне опреме и трансформатора, као елемената који учествују у прорачунима. Гране иницијалног графа представљају двокрајници (укључујући и двонамотајне трансформаторе), једнокрајници (чији се други крај дефинише као фиктивни чвор конективности), док се тронамотајни трансформатори представљају са две гране (основном и изведеном). Чворови (конективности) иницијалног графа представљају елементе типа *ConnectivityNode*.

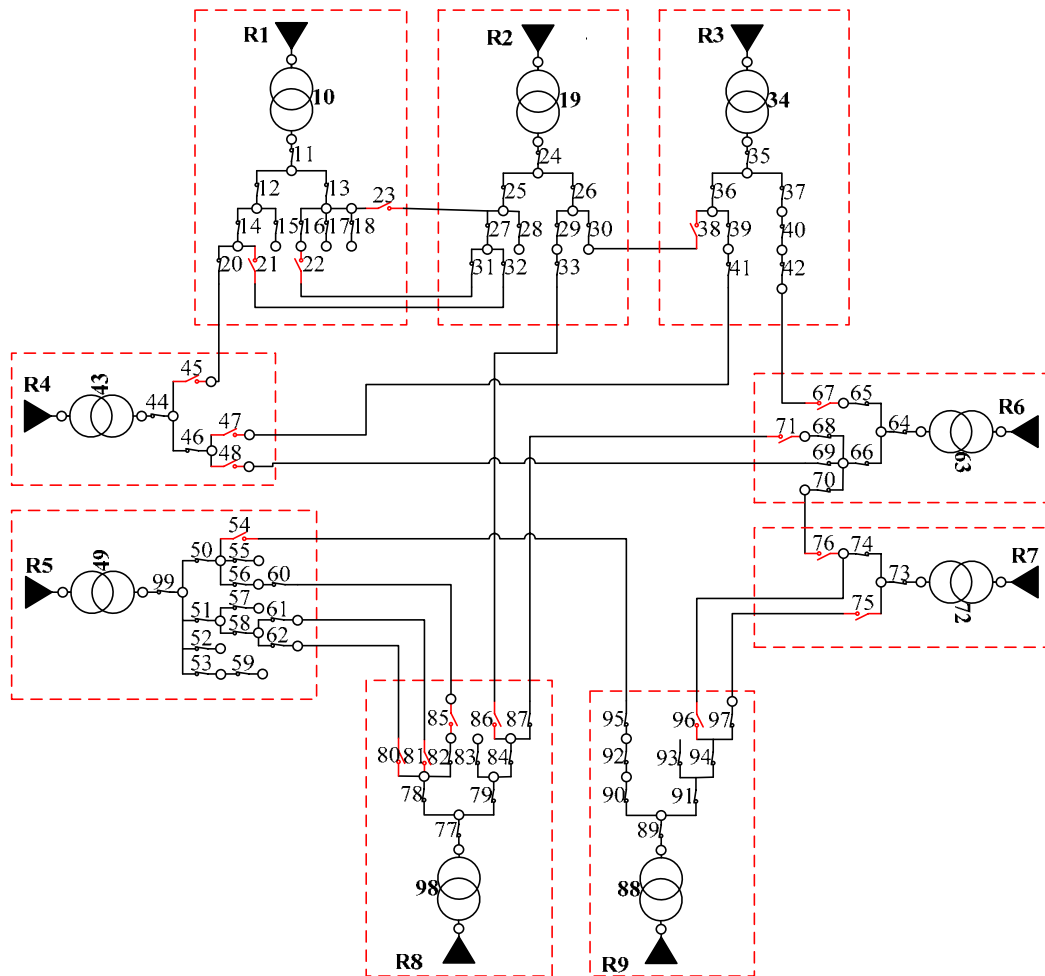
2. Тополошка анализа и креирање графа за прорачуне

Тополошка анализа зависи од стања улаза U_r (нпр. стања прекидача отворен/затворен са слике 3.1.). Циљ тополошке анализе модела је утврђивање основних група елемената (области прорачуна) неопходних за прорачуна. Резултат тополошке анализе је домен прорачуна D који се представља неусмереним тежинским графом $G = (V, E)$ где је V скуп чворова, а E скуп грана.

Чворови графа за прорачуне представљају напајане регионе (корене), док гране графа представљају везе између одговарајућих корена. За грану $e_{i,j} = (v_i, v_j)$ се дефинише тежина - $w(e_{p,q})$ као број потенцијалних конекција између две области:

$$w(e_{p,q}) = w(e(R_p, R_q)) = \sum_{\substack{\alpha_i \in R_p, \\ \alpha_j \in R_q}} Pot(\alpha_i, \alpha_j, u_{i,j} = inactive) \quad (5)$$

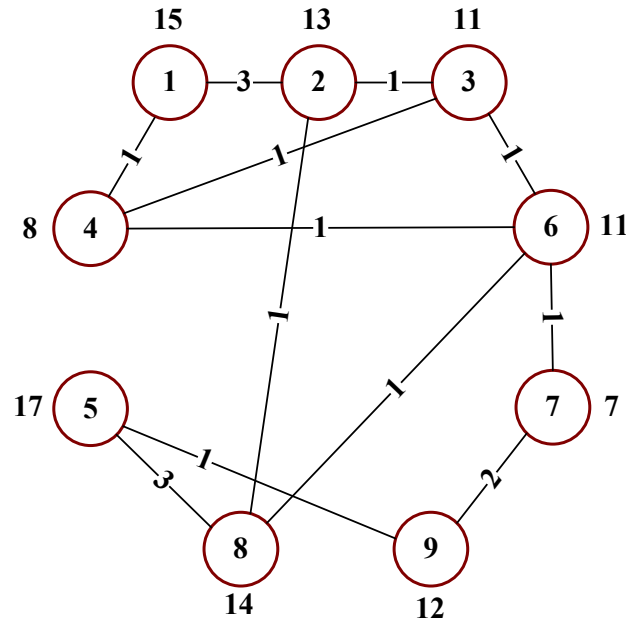
На пример између елемената из региона R_1 и R_2 постоје 3 отворена прекидача, па је због тога $w(e_{1,2})=3$.



Слика 3.2. Један пример иницијалног модела података у дистрибутивним системима

Тежина чвора прорачуна $w(v_i)$ зависи од комплексности прорачуна за одговарајућу регион R_i . Пример одређивања комплексности прорачуна за различите типове електричних елемената у електр енергетским системима је дат у [49]. Уколико се разматра LF функција може се приближно усвојити да је комплексност прорачуна за неки регион линеарно пропорционална укупном броју елемената у

региону, тј. тежина чвора v_i је једнака броју елемената у региону R_i . Тако нпр. за LF прорачуне, иницијални граф модела са слике 3.2. би се трансформисао у граф приказан на слици 3.3.



Слика 3.3. Укрупњени граф за прорачун токова снага

3. Подела укрупњеног графа – дефинисање критеријума оптималности

У мултипроцесорском окружењу, аналитичка функција ξ (LF и све функције које се извршавају над коренима) се могу извршавати паралелно над различитим регионима (коренима). Ако је број корена већи од броја процесора, корени се групишу у p партиција (резултат партиционисања је скуп партиција $\Pi = \{\pi_1, \pi_2, \dots, \pi_p\}$) и распоређују на p процесора. Ово значи да се на једном процесору секвенцијално извршавају прорачуни ξ функције која је њему додељена. Проблем поделе графа се своди на поделу неусмереног укрупњеног графа чији чворови (чворови прорачуна) и гране (све потенцијалне конекције између чворова прорачуна) имају одговарајуће тежине.

3.5. Дефинисање оптимизационог критеријума

У циљу дефинисања оптимизационог критеријума, за партицију π_k је потребно дефинисати тежина партиције W_{π_k} :

$$W_{\pi_k} = \sum_{R_j \in \pi_k} w_{R_j}, \quad (6)$$

као суму тежина свих припадајућих области, и функцију ϕ_k по формули:

$$\phi_k = \sum_{R_p, R_q \in \pi_k} Pot_{R_p, R_q}. \quad (7)$$

Ова функција је индикатор „добре повезаности“ између региона прорачуна унутар партиције. Тежина партиције представља меру комплексности прорачуна над подацима унутар партиције.

Задатак је груписати области у унапред дефинисани број партиција (p), тако да су тежине тих партиција приближно једнаке, али никада веће од *максимално дозвољене тежине партиције* M :

$$M = (1 + \varepsilon) \cdot \frac{1}{p} \sum_{k=1}^p W_{\pi_k}, \quad (8)$$

где је W_{π_k} тежина партиције, p је укупан број партиција, а $\varepsilon \in [0, (p-1)/p]$ је толеранција. Ако је $\varepsilon = 0$ тада све партиције треба да имају исту тежину (што представља случај идеалног балансирања), али решавање оптимизационог проблема подлеже строжијим ограничењима и оптимум се може пронаћи у веома ретким случајевима. Са повећањем толеранције повећава се и манипулативна моћ алгорита, па се и резултати оптимизације могу лакше достићи.

Оптимизациони критеријум се може дефинисати као максимална повезаност области унутар партиције:

$$F = \max \left(\sum_{k=1}^p \phi_k \right), \quad (9)$$

где је функција ϕ_k дефинисана у једначини (7), а тежине свих партиција су ограничене са:

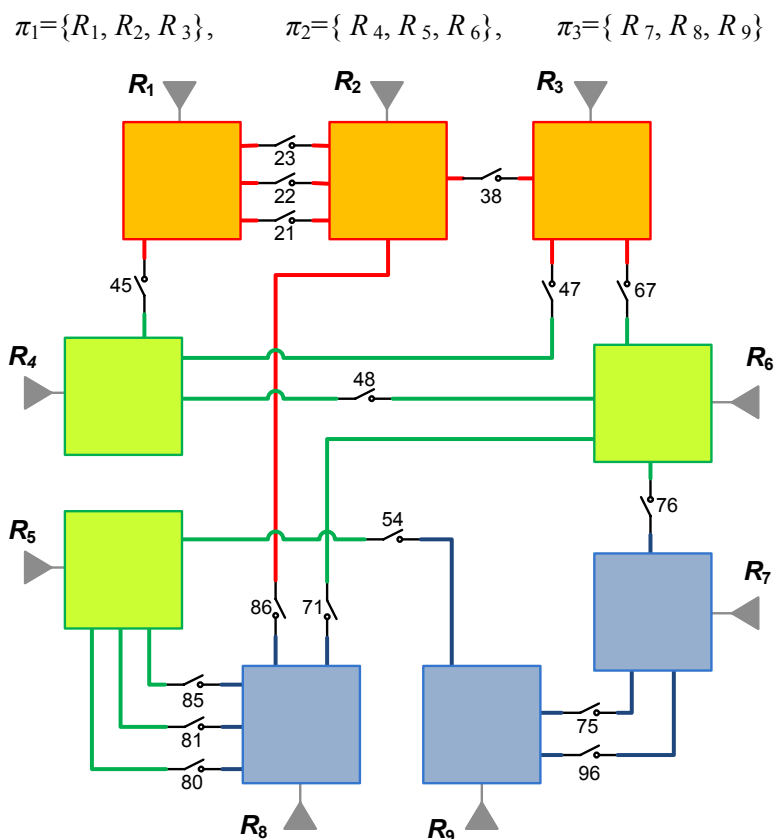
$$W_{\pi_k} \leq M, \forall k \in \{1, 2, \dots, p\}. \quad (10)$$

Треба нагласити да максимална повезаност унутар партиција у исто време значи минималан број веза између партиција.

Области прорачуна се могу мењати током времена у зависности од промене суседности узроковане улазним подацима. Када се активира потенцијална конекција између два елемента из различитих области (корена), те две области (корени) се повезују и граде једну област (корен). Уколико се нека потенцијална конекција деактивира – може доћи до цепања корена на два дела. Динамичке промене улазних $U_r = \{u\}$ узрокују активацију/деактивацију потенцијалних конекција, чиме се смањује/повећава број потенцијалних конекција између корена. На основу

наведених чињеница, може се рећи да је квантитативни показатељ вероватноће да се два региона (корена) повежу, сразмеран броју потенцијалних конекција између њихових елемената. Због тога се он користи у оптимизационом критеријуму за поделу корена између партиција.

На основу овако дефинисаних оптимизационих критеријума, примењују се алгоритми за поделу графа. Резултат извршавања алгоритма је распоређивање података груписаних у коренима по партицијама (процесорима). Уз претпоставку да је задатак поделити мрежу приказану на слици 3.2. на три партиције, резултат партиционисања би био (приказано на слици 3.4.):



Слика 3.4. Подела моделе дистрибутивне тест мреже на 3 партиције

С обзиром на чињеницу да у току рада система долази до промене модела података, подела података се врши:

1. Иницијално – пре почетка рада система, тј. пре покретања аналитичких прорачуна, и
2. Динамички – у току рада система.

3.5.1. Иницијална подела података

Иницијална подела података се врши пре покретања прорачуна, на основу стања модела података у том тренутку. У току рада система долази до промена модела (уносом, изменама или брисањем одређених елемената) које утиче на измену области прорачуна, па се врши њихова динамичка прерасподела.

Иницијална подела података се покреће у следећим случајевима:

- када подаци над којима се врше прорачуни нису претходно распоређени по партицијама, или
- уколико се укаже потреба за оптимизацијом рада система – ако су динамичке промене модела (и динамичке прерасподеле) довеле до недовољно добре поделе података.

Иницијална подела се врши на основу претходно дефинисаних критеријума (једначине (6) – (10)). Алгоритми за иницијалну поделу ће бити детаљније описани у поглављу 4.

3.5.2. Динамичка прерасподела модела података

За објашњење динамичке прерасподеле потребне су следеће дефиниције (које су формулисане у [50], [51]): Уколико нека партиција има тежину већу од максимално дозвољене партиције $W_{\pi_k} > M$, тада се каже да је граф *небалансиран*, а партиција π_k је *превелика партиција*. Издељени граф је *балансиран* ако у њему нема превеликих партиција.

Динамичка прерасподела се врши у току рада система (*on-line*) када се утврди да оптерећење процесора (у оквиру мултипроцесорског система) није добро распоређено – непосредно након спољашњих промена или периодично (уколико се промене често дешавају, па би честа прерасподела довела до инертности система). На тај начин се јавља небалансиран граф, који је узрокован променама у тежини региона или појавом новог региона. У таквим случајевима се покреће динамичка прерасподела графа. Задатак динамичке прерасподеле је добијање балансираних и оптимално повезаних партиција (једначине (9) – (10)) уз минимално пребацивање региона (корена) између партиција.

Небалансираност графа (процесора) доводи до прерасподеле графа у следећим случајевима:

1. **Прерасподела региона узрокована спољашњим утицајем.** Ово се дешава

када улазне величине активирају потенцијалне конекције између региона. Тада се два региона спајају у један (нови) регион чија је тежина једнака збиру тежина придружених региона, и све њихове гране се налазе у новом региону. Ако придружени региони припадају различитим партицијама и ако се њиховим спајањем ствара небалансиран граф – стартује се динамичка прерасподела графа.

У случајевима када се деактивира нека потенцијална конекција (нпр. отвори се неки прекидач), ово се дешава унутар неког региона и може узроковати поделу (цепање) региона на два дела. С обзиром да се оба дела региона остају у истој партицији, ова изолована појава не изазива небалансираност графа, а самим тим се не покреће ни динамичка прерасподела графа. Међутим, ово цепање региона ће бити узето у обзир приликом наредног покретања динамичке прерасподеле.

- 2. Прерасподела узрокована праћењем историјата оптерећености процесора.** Понекад се аналитичка функција ξ извршава непредвиђено, након промена различитих улазних величина, што неке регионе чини чешће коришћеним у прорачунима. Тада се праћењем (мерењем) оптерећености процесора могу поново одредити тежине региона, тј. тежина региона у таквим случајевима је пропорционална времену утрошеном за извршавање функције ξ над регионом.

Спољашње промене статуса расклопне опреме и проширење и измена модела (додавање, измена или брисање одређених елемената) могу довести до небалансираног оптерећења процесора и довести до промене графа. У *on-line* систему су статуси расклопне опреме даљински праћени (преко *SCADA* система или одговарајућег сервиса). Промене статуса расклопне опреме су релативно ретке (посебно оне који се налазе између два корена). Са друге стране, *LF* прорачуни се често извршавају, а иницирају се променама других величина (нпр. променом на мерним уређајима, промена положаја регулационе склопке, и др.). Због небалансираности графа узроковане променама стања потенцијалних конекција и великог броја прорачуна који се извршавају, неопходно је прерасподелити регионе у циљу оптималног коришћења ресурса, односно, повећања брзине извршавања прорачуна. Ово је повод за спровођење истраживања на пољу динамичке прерасподеле модела података дистрибутивне електроенергетске мреже. Алгоритми

за динамичку поделу ће детаљно бити описани у поглављу 5, заједно са специфичностима везаним за формулисање критеријума оптималности за динамичку прерасподелу графа.

3.5.3. Расподела ненапајаних делова мреже – острва

У систему могу постојати привремено ненапајани делови мреже, тзв. острва. Иако се ради о малим, обично алтернативним начинима напајања до других делова мреже, битно је распоредити их тако да се њиховим напајањем не ремети балансираност поделе модела.

Критеријуми за распоређивање острва у партиције су следећи:

- свако острво које је потенцијално повезано са само једним кореном (нпр. R_y) се распоређује у партицију у коју је распоређен корен R_y ;
- острва која су потенцијално повезана са више корена распоређују се у партиције на основу најбоље потенцијалне повезаности са суседним областима, с тим што се корени који су груписани у исте партиције третирају као једна област (приликом одређивања степена потенцијалне повезаности);
- уколико постоје острва која нису потенцијално повезана ни са једним кореном она ће бити распоређена на крају (појава оваквих острва представља невалидна ситуацију, тако да се не морају разматрати).

Ако се острво налази између два корена у оквиру исте партиције, онда и острво остаје у тој партицији. Уколико је острво на граници између два или више корена из различитих партиција, острво остаје у партицији у коме се налази корен унутар кога је настало острво.

Уколико се појаве петље између корена у оквиру исте партиције, корени остају у тој партицији и за наредни поступак поделе графа се третирају као један корен. Када се упетљани корени налазе у различитим партицијама, потребно их је поставити у исту партицију. У зависности од броја партиција у којима се налазе међусобно упетљани корени зависи и број могућих решења. Стога је прво потребно израчунати вредност критеријума оптималности за сваку захваћену партицију након преношења упетљаних корена у њега и затим извршити пренос корена у партицију за који је добијена најмања вредност.

Појављивањем упетљаних корена (из различитих партиција) често се нарушава оптималност подељеног графа (у том случају аутоматски се стартује процес прерасподеле).

Острво се пребацује у другу партицију у два случаја:

1. приликом покретања нове поделе или
2. приликом повезивања са неком од области.

Подела модела података се може покретати у следећим случајевима:

1. иницијално – при подизању система,
2. периодично,
3. на захтев,
4. *on-line* – приликом сваке промене статуса расклопне опреме (или промене која утиче на корене – нпр. убацивањем појединих делова мреже се могу проширити одређени корени),
5. аутоматски – када меморијска заузећа по партицијама пређу горњу границу толеранције (која је већа од максимално дозвољене M).

4. Алгоритми за поделу графова

Алгоритми за поделу графова имају примену у бројним системима где је у циљу оптимизације рада потребно извршити поделу на подсистеме. Примери овакве примене су:

- *VLSI* (енгл. *Very-Large-Scale Integration*) технологије – код ових система је неопходно извршити оптималну поделу транзисторских компоненти у групе (на нивоу система, штампаних плоча, чипова) тако да повезаност између компоненти у оквиру различитих група буде што мања;
- *Web* апликације – подела докумената у циљу оптималне класификације докумената, претпроцесирања за *PageRank* алгоритам, и др.;
- Велики модели података – захтевају скупе рачунаре за процесирање, па се дистрибуцијом података и обраде на више јефтинијих рачунара значајно смањује цена хардвера;
- Паралелни прорачуни над групама података.

Генерално посматрано, у сваком од поменутих система постоје међусобно спрегнути елементи које је потребно разврстати у групе тако да у свакој групи буду елементи који су повезани чврстим везама. У идеалном случају, када постоје дисјунктни подскупови елемената, проблем поделе је тривијалан, али практични проблеми су сложенији јер се типично намеће ограничење на величину групе и тада се код сваке могуће расподеле елемената појављују везе између група. Тиме се поступак поделе поставља као оптимизациони проблем где је потребно наћи одговарајућу поделу тако да везе међу групама буду минималне и уједно да су све групе приближно (или тачно) једнаке величине. Поступком минимизације веза између група добијају се групе (максимално) чврсто повезаних елемената.

Један од начина поделе система је примена алгоритама за поделу графова. Проблеми поделе графова спадају у групу *NP* (енгл. *Nondeterministic Polynomial Time*) комплетних проблема комбинаторне оптимизације [52].

За решавање проблема као што су подела графа, проблем трговачког путника, бојење графова и др., користе се алгоритми који могу да пронађу решење у простору допустивих решења за максимално време које је сразмерно експоненцијалној

функцији димензије проблема. Тада се каже да је временска комплексност алгоритма експоненцијална и користи се ознака са $O(x^n)$. Тако нпр. проблем поделе графа од n чворова на 4 партиције захтева време $O(4^n)$. Овакви проблеми се називају *NP* тешко решиви проблеми комбинаторне оптимизације. Са друге стране постоје проблеми за чије решавање се користе алгоритми решавање користе полиномско време у односу на улазну популацију тачака ($O(n^2)$, $O(n^3)$ и др.), па се такви проблеми називају *P* (*P* – *Polinomial Time*) проблеми. Посебну групу *NP* проблема представљају *NP* комплетни проблеми (постоји група оваквих проблема, међу које спада и проблем поделе графова) и они се дефинишу као најтежи проблеми комбинаторне оптимизације. Карактеристика ових проблема је да, уколико се пронађе алгоритам полиномске комплексности за било који проблем из групе *NP* комплетних проблема, сви остали проблеми се могу решити алгоритмом полиномске комплексности.

С обзиром на нерационалност захтева за експоненцијалним решавањем проблема, *NP* комплетни тешко решиви проблеми се решавају апроксимативним алгоритмима (чија је временска комплексност полиномска). Често се користе еволутивни алгоритми као што су генетски алгоритам [53], [54], алгоритам мравље колоније [55], *PSO* алгоритам [56], и др. Неки од њих ([57], [58]) дају врло добре резултате за графове мањих величина (до 100 чворова), док се поједине методе [59] посебно добро показују на графовима великих димензија (више милиона чворова).

Проблеми поделе графова се могу односити на различите типове графова, па се у складу са тим врши избор алгоритма за поделу. Графови могу бити дефинисани са геометријским информацијама у чворовима (или без њих), тежински – сваком чвору и/или грани се додељује одређена тежина, графови код којих гране могу да повезују више чворова (тзв. хиперграфови) и др. У оквиру овог рада ће се разматрати тежински графови код којих се тежине придружују и чворовима и гранама. Алгоритми који су разматрани у оквиру овог рада за поделу таквих графова се могу класификовати у следеће групе:

1. алгоритми засновани на пребацивању чворова из једне партиције у другу
2. еволутивни алгоритми
3. вишефазни (*multilevel*) алгоритми
4. хибридни алгоритми

4.1. Алгоритми засновани на пребацивању чворова

Алгоритми из ове групе се примењују у случајевима када већ постоји издељен граф, тј. након примене неког другог алгоритма. Понекад се они користе и самостално, с тим што се претходно граф издели на случајан начин уз задовољење основних критеријума за поделу (балансираност величине партиција и сл.).

Најчешће коришћени алгоритми за поделу графова засновани на пребацивању чворова су *KL* (*Kernighan-Lin*) алгоритам [60] и једна од његових варијанти *FM* (*Fiduccia-Mattheyses*) алгоритам [61].

4.1.1. *KL* алгоритам

KL алгоритам је један од најчешће коришћених алгоритама за поделу графа, а настао је као резултат истраживања која су спровели *Kernighan* и *Lin*, и описали у раду [60] из 1970. године. У њему је дат алгоритам за прерасподелу чворова између две партиције π_A и π_B . То је поступак у коме се чворови размењују између две партиције. Идеја је да се пронађу чворови чијом би се разменом смањио број пресечних грана између чворова који се налазе у различитим партицијама (напомена: нису разматрани тежински графови).

Полази се од скупа чворова $V = \{v_1, v_2, \dots, v_{2n}\}$ који су иницијално распоређени у партиције π_A и π_B (у сваку по n чворова). Задатак је пронаћи подскупове чворова $X \subset \pi_A$ и $Y \subset \pi_B$ чијом ће се разменом добити партиције π_A^* и π_B^*

$$\pi_A^* = \pi_A + Y - X \quad \pi_B^* = \pi_B + X - Y$$

такве да је збир пресечних грана између партиција π_A^* и π_B^* минималан, тј. да се не могу пронаћи чворови из партиција π_A^* и π_B^* чијом би се разменом смањио број пресечних грана, уз задовољење ограничења о величини партиција.

За неки чвор $v_a \in \pi_A$ дефинише се параметар *спољашње везе* S_a као

$$S_a = \sum_{v_y \in \pi_B} e_{ay} \quad (11)$$

и *унутрашње везе* U_a као

$$U_a = \sum_{v_x \in \pi_A} e_{ax} \quad (12)$$

где је e_{ij} грана између чворова v_i и v_j .

Након тога се за сваки чвор v_z дефинише параметар *допринос* $D_z = S_z - U_z$ као разлика између спољашњих и унутрашњих веза. Установљено је да ако се два чвора

v_a и v_b ($v_a \in \pi_A$ и $v_b \in \pi_B$) размењују, да се параметар *побољшање* g_{ab} (смањење пресечних грана) рачина по формули:

$$g_{ab} = D_a + D_b - e_{ab} \quad (13)$$

KL алгоритам је итеративни поступак код кога се у свакој итерацији бирају чворови v_a и v_b ($v_a \in \pi_A$ и $v_b \in \pi_B$) за које је израчунато максимално побољшање g_{ab} . Након тога се чворови размењују између партиција (у случају да је $g_{ab} > 0$) и поново рачунају побољшања g_{ab} за чворове који су суседни размењеним чворовима v_a и v_b (с обзиром да су се њима промениле *спољашње* и *унутрашње везе*).

Псеудо код *KL* алгоритма се може дефинисати као:

Input: partitioned graph $G=(V,E) \rightarrow \Pi=\pi_1 \cup \pi_2 \cup \dots \cup \pi_p$	
Output: improved partitions of graph $G \rightarrow \Pi'=\pi_1' \cup \pi_2' \cup \dots \cup \pi_p'$	
1	repeat
2	for each vertex v_i in partition π_A
3	$D_i = S_i - U_i$
4	end for
5	for each vertex v_j in partition π_B
6	$D_j = S_j - U_j$
7	end for
8	
9	select ($v_a \in \pi_A$ & $v_b \in \pi_B$) $g_{ab} = \max(g)$
10	update D_x and D_y (for neighbors of v_a and v_b)
11	recalculate g_{xp} and g_{yq}
12	until ($\max(g) \leq 0$)

Ефикасније извршавање *KL* алгоритма се постиже сортирањем доприноса D_i за обе партиције: $D_{a_1} \geq D_{a_2} \geq \dots \geq D_{a_n}$ за партицију π_A и $D_{b_1} \geq D_{b_2} \geq \dots \geq D_{b_n}$ за партицију π_B .

Затим се бира чвор из сваке партиције који даје највећи допринос (v_{a1} из партиције π_A и v_{b1} из партиције π_B), рачуна се побољшање $g_{a_1 b_1}$ и пореди са побољшањима прорачунатим за све комбинације укрштања неколико чворова са највећим доприносима из сваке партиције (линија 9 у псеудо коду). Експерименти показују да је довољно прорачунати g_{ij} које се добија укрштањем прва три чвора (са највећим D) из сваке партиције.

Карактеристика *KL* алгоритма је да квалитет његовог решења често зависи од почетних партиција, али у сваком случају даје веома добре резултате. За поделу на више партиција се користи *KL* алгоритам који се примењује итеративно на парове партиција (подскупове издељеног графа).

4.1.2. FM алгоритам

FM алгоритам [61] спада у групу миграционих хеуристика који се извршава на већ издељеном графу. Најчешће се (као и *KL* алгоритам) користи за побољшање решења добијеног неким другим алгоритмом. За разлику од *KL* алгоритма, код *FM* алгоритма се побољшање решења добија пребацивањем само једног чвора из једне партиције у другу. У раду [61] је предложен алгоритам за бисекцију графа (поделу на две партиције), али су касније развијене и варијанте *FM* алгоритма за поделу на више партиција [62].

Критеријум за пребацивање чвора v_i из партиције π_s у партицију π_j је максимална вредност параметра побољшања који се рачуна по формули:

$$g(i, J) = FS(i, J) - TE(i) \quad (14)$$

где је $FS(i, J)$ број грана које повезују чвор v_i са чворовима из партиције π_j , а $TE(i)$ је број грана између чвора v_i и осталих чворова из његове партиције (π_s).

У случају тежинског графа (код кога и гране и чворови имају одређене тежине), ове променљиве представљају збир тежина одговарајућих грана (уместо броја грана), док се тежине чворова користе за проверу балансираности партиција. *Fiduccia* и *Mattheyses* [61] предлажу метод за повећање брзине алгоритма коришћењем вишеструко спрегнутих листа (*bucket list*) што смањује време извршавања (чија је комплексност по итерацији сразмерна броју грана - $O(|E|)$). За поделу графа на више партиција, проверавају се сви могући преласци свих граничних чворова (који су суседни са чворовима из других партиција).

Псеудо код поједностављеног *FM* алгоритма је формулисан на следећи начин:

```

Input: partitioned graph  $G=(V, E) \rightarrow \Pi = \pi_1 \cup \pi_2 \cup \dots \cup \pi_p$ 
Output: improved partitions of graph  $G \rightarrow \Pi' = \pi_1' \cup \pi_2' \cup \dots \cup \pi_p'$ 

repeat
  for each border vertex  $v_i$  in graph  $G$ 
    for each partition  $\pi_j$  in  $\Pi$ 
       $g(i, j) \leftarrow FS(i, j) - TE(i)$ 
    end for
  end for
  find  $v_b \in \pi_s$  and  $\pi_d \leftarrow (\max(g) = g(b, D))$ 
  if  $(w(\pi_d \cup \{v_b\}) < M)$ 
     $\pi_d' = \pi_d \cup \{v_b\}$ 
     $\pi_s' = \pi_s \setminus \{v_b\}$ 
  end if
  recalculate  $g(i, j)$  (for neighbors of  $v_b$ )
until  $(\max(g) \leq 0)$ 

```

4.2. Природом инспирисани алгоритми

У ову групу алгоритама се могу сврстати сви оптимизациони алгоритми у чију процедуру је уграђена законитост понашања неког система у природи. Тако нпр. генетски алгоритми су преузели законитости које почивају на генетском укрштању хромозома човека, понашања група животиња су уграђени у велики број успешних оптимизационих алгоритама (нпр. понашање јата птица је коришћено при формирању *PSO* (енгл. *Particle Swarm Optimization*) алгоритма, понашање колоније мравца *ACO* (енгл. *Ant Colony Optimization*), понашање роја пчела *BCO* (енгл. *Bee Colony Optimization*), itd.).

С обзиром на дискретност оптимизационог проблема, за његово решавање се користе алгоритми: који су развијени за дискретне проблеме (алгоритам мравље колоније), који се равноправно могу користити и за дискретне и за континуалне проблеме (генетски алгоритам), или који су првенствено намењени континуалним проблемима, али се могу прилагодити и за решавање дискретних проблема (као нпр. *PSO*). Сви наведени алгоритми полазе од иницијалног скупа јединки, најчешће генерисаних на случајан начин и врше претрагу простора допустивих решења у циљу проналажења глобалног оптимума.

Представљање решења (јединке). Претпоставимо да је задати граф $G=(V,E)$ са скупом чворова (региона) $V=\{v_1, v_2, \dots, v_d\}$ потребно поделити на p партиција. Решење ће у том случају бити представљено вектором од d елемената чије вредности представљају индекс припадности одговарајућег чвора одређеној партицији (попримају вредности од 1 до p) [54], [57], [58].

Тако нпр. за граф од 8 чворова који се дели на 3 партиције, решење 12332131 се односи на следеће решење:

$$\pi_1=\{v_1, v_6, v_8\}; \quad \pi_2=\{v_2, v_5\}; \quad \pi_3=\{v_3, v_4, v_7\};$$

У даљем излагању ће бити описани генетски и PSO алгоритам, који су у различитим варијантама коришћени за проблем поделе графа заснованог на моделу података дистрибутивне електроенергетске мреже.

4.2.1. Генетски алгоритам

Генетски алгоритам (*GA*) спада у групу еволутивних алгоритама и заснован је на Дарвиновој теорији еволуције формираној у XIX веку [63]. Зачетником развоја генетског алгоритма сматра се немачки научник *Holland* који је 1975. године у својој

књизи „*Adaptation in Natural and Artificial Systems*“ поставио темеље генетског алгоритма [64]. Он је представио алгоритам као рачунарски процес који се базира на природном еволутивном процесу код кога је скуп могућих решења представљен јединкама (хромозомима) који чине једну популацију. Јединке су низови гена чије вредности представљају једно решење, које се израчунава на основу унапред задате критеријумске функције. Критеријумска функција (често се назива и *fitness* функција) служи за вредновање квалитета одређене јединке и користе се приликом избора јединки које ће бити пренесене у следећу генерацију. Идеја алгоритма је да свака следећа генерација задржи добре особине претходне генерације, и да се та добра својства пренесе и у наредним генерацијама.

Поступак иницијализације алгоритма се базира на постављању важних параметара који дефинишу понашање алгоритма (о којима ће бити речи касније) и креирању иницијалне популације. Иницијална популација се најчешће генерише на случајан начин. Број гена у јединки је једнак броју чворова графа који се дели, а и величина популације је често бар једнака том броју чиме се омогућује добра манипулативна способност алгоритма и претраживање по свим правцима у простору могућих решења.

GA је итеративни поступак који се састоји три оператора:

1. селекције
2. укрштања
3. мутације

Селекција је поступак избора јединки-родитеља из популације које ће учествовати у укрштању. Примењена су три типа селекције [65]: пропорционалне, турнирска и случајна. Турнирском селекцијом се из случајно изабраних група јединки бира најбоља. Најчешће су у питању две јединке од којих се бира боља, али их може бити и више. Уколико се користи ова селекција, уводи се параметар величина турнира n_t , којим се специфицира број јединки који учествују у такмичењу. Код пропорционалне (или рулет) селекције вероватноћа избора јединке је пропорционална релативном односу вредности критеријумске функције дате јединке и збира критеријумских функција свих јединки. Пропорционална селекција је једина коју у свом раду предлаже *Holland* [64]. Случајна селекција на случајан начин бира јединке које учествују у укрштању.

Укрштање (*crossover*) се врши између две јединке са циљем да се добре особине родитеља пренесу на потомке. Прво се у процесу селекције бирају

родитељи, а након тога се врши укрштање по одређеном правилу. Уколико су потомци погодни за популацију (дају добре вредности за критеријумску функцију) онда они замењују одређене („најлошије“) јединке из популације.

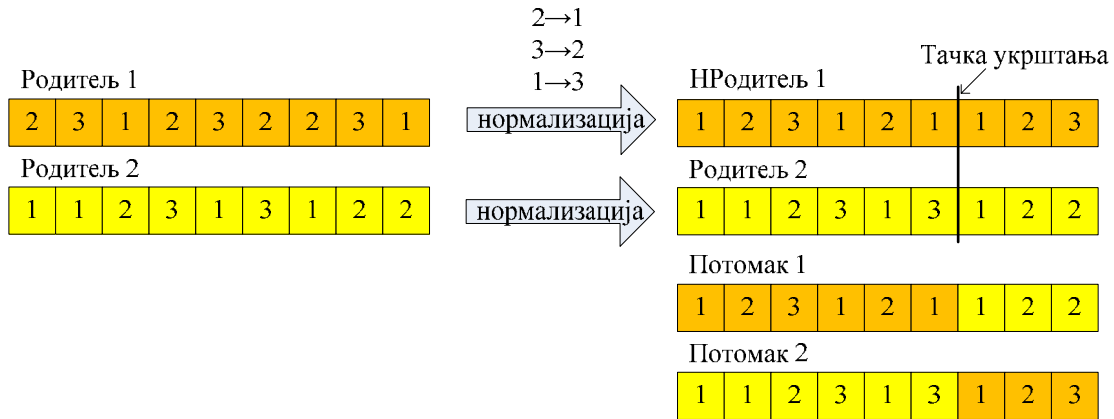
У раду је примењено више типова укршања: просто укрштање у једној тачки, укрштање у једној тачки са нормализацијом и укрштање са очувањем најбоље партиције (најбоља партиција се преноси на потомка) [57].

Просто укрштање у једној тачки подразумева да се изабере једна тачка (позиција) у јединки сваког родитеља [65]. На тај начин сваки родитељ се дели на два дела: 1. низ гена десно од тачке, и 2. низ гена лево од тачке. *Потмак1* настаје када се споје леви део првог родитеља и десни део другог родитеља. *Потмак2* се добија спајањем левог дела другог родитеља са десним делом првог родитеља. Пример простог укрштања у једној тачки је приказан на слици 4.1.



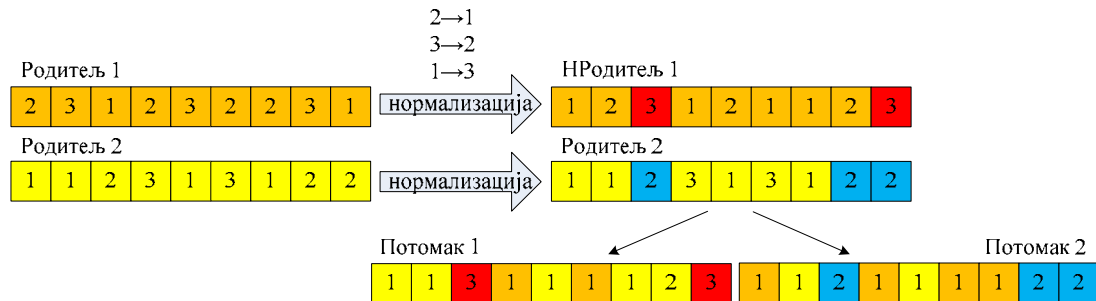
Слика 4.1. *Просто укрштање у једној тачки*

Нормализација подразумева јединствену репрезентацију јединке [66]. На пример, јединке 123121123 и 312313312 са аспекта поделе на партиције представљају исто решење (само су замењени индекси представе партиција у јединкама). Код ове варијанте укрштања оба родитеља се јединствено представљају тако што се индекси партицијама у оквиру јединке појављују у растућем редоследу. На тај начин се јединке 123121123 и 312313312 нормализацијом трансформишу у идентичне јединке са садржајем 123121123. Након нормализације родитеља, врши се основно укрштање по горе наведеној процедури. Пример укрштања у једној тачки са нормализацијом је приказан на слици 4.2.



Слика 4.2. Укрштање у једној тачки са нормализацијом

Укрштање са очувањем најбоље партиције представља покушај да се у генетски алгоритам унесе одређени детерминизам одређен природом проблема, који би довео до бољег резултата. У том случају се потомку 1 додељују сви гени најбоље партиције из родитеља 1 и сви остали гени родитеља 2. Са друге стране, потомку 2 се преносе гени из најбоље партиције родитеља 2 и сви остали гени родитеља 1. Додатна варијанта овог укрштања може бити и претходна нормализација родитеља. Пример укрштања са очувањем најбоље групе је приказан на слици 4.3. (у наведеном примеру најбоље партиције су: партиција 3 код 1. родитеља и партиција 2 код 2. родитеља)



Слика 4.3. Укрштање са фаворизацијом најбоље партиције и нормализацијом

Мутација се врши се када се на случајно изабраној јединки промени вредност гена (тј. једна случајно изабрани чвор промени партицију). Постоји могућност и да два чвора из различитих партиција пређу у партицију оног другог, тзв. *мутација заменом* (*swap*). Том приликом два случајно изабрана гена замене вредности.

У оквиру генетског алгоритма могуће је применити и тзв. *елитизам* –чување одређеног броја јединки са најбољим критеријумском функцијом за следећу генерацију. Те јединке могу да учествују у укрштању, мутацији и *swap* мутацији, али

њихова копија се сигурно неизмењена пребацује у следећу генерацију.

Преношење добрих особина *укрштањем* омогућава побољшање критеријумске функције у следећим генерацијама. Међутим може се десити да цела популација поприми особине најбоље јединке и да временом све постану идентичне. Та појава се региструје тако што након сортирања најбоља и најлошија јединка имају исту вредност критеријумске функције. Тада је алгоритам дошао у локални оптимум и не може даље да напредује. Да би се омогућило даље извршавање, потребно је уништити део популације, и поново генерисати нове јединке на случајан начин. При томе ће се најбоље јединке сачувати, да би се обезбедила конвергенција ка глобалном оптимуму. Генетски алгоритам се зауставља или када се извршио унапред задати број итерација (генерација) или када се не може постићи побољшање решења након одређеног броја генерација.

Значајни параметри за генетски алгоритам су:

- величина популације – укупан број јединки у свакој генерацији,
- вероватноћа укрштања – вероватноћа да ће одређена јединка учествовати у укрштању,
- вероватноћа мутације – вероватноћа да ће нека јединка бити мутирана,
- број итерација – максималан број итерација за које се извршава алгоритам,
- степен елитизма – број елитних јединки који се преносе у следећу генерацију,
- вероватноћа *swap* мутације – вероватноћа учешћа јединке у *swap* мутацији.

Паралелни генетски алгоритам

Идеја паралелног генетског алгоритма је покушај да се подели популација на делове који ће радити паралелно и независно, и након одређеног броја итерација размењивати добијене резултате [54], [67]. На тај начин би се могло убрзати време извршавања алгоритма, а и побољшати резултати спречавањем „заглављивања“ алгоритма у локалном оптимуму.

Паралелизација алгоритма се може реализовати на два начина, и то кроз:

1. Конкурентан рад – више програмских нити се истовремено извршавају на једном процесору.
2. Дистрибуирани алгоритам – делови популације би се распоредили на више процесора где се паралелно извршавају.

Поред тога, за претраживање простора могућих решења над различитим

деловима популације могу се користити различити параметри у генетском алгоритму.

Развијени паралелни генетски алгоритам подразумева паралелно извршавање генетских алгоритама над деловима популације и размену резултата између алгоритама. При томе је потребно дефинисати следеће параметре паралелног алгоритма: период синхронизације (T_{sync}) и стопу миграције (μ_s). Период синхронизације представља број итерација након којих алгоритми размењују јединке, а стопа миграције се односи на број јединки који се размењују. Са порастом периода синхронизације повећава се аутономност појединачних алгоритама, смањује се комуникација између алгоритама чиме се повећава брзина паралелног алгоритма. Са друге стране, за мале периоде комуникација између алгоритама је чешће, паралелни алгоритам се своди на секвенцијални и повећава се време паралелног алгоритма (због учестале комуникације). Стопа миграције утиче на аутономност појединачних алгоритама, када је стопа миграције већа – алгоритми имају већи међусобни утицај.

Поред тога, разликују се два типа паралелног генетског алгоритма [68]:

- SIMD (енгл. *Single Instruction Multiple Data*) – паралелно извршавање истих генетских алгоритама (са истим параметрима),
- MIMD (енгл. *Multiple Instruction Multiple Data*) – паралелно извршавање генетских алгоритама са различитим параметрима.

4.2.2. PSO алгоритам

PSO представља стохастичку оптимизациону технику засновану на колективној интелигенцији популације. *Kennedy* и *Eberhart* су 1995. године, подстакнути социјалним понашањем јата птица у потрази за храном [69], у свом раду [56] развили *PSO* алгоритам који усваја заједничка својства понашања животиња које се крећу групама. Популација (тзв. рој) се састоји од низа јединки (тзв. честица) које се крећу кроз вишедимензиони простор и мењају своје позиције у зависности од сопственог искуства и искуства осталих јединки. Алгоритам је иницијално развијен за континуалне системе, а након тога су инведене варијанте алгоритма за дискретне системе: бинарни *PSO* [70] и целобројни *PSO* [71]. Код бинарног алгоритма свака јединка (честица) из популације може изети бинарне вредности (0 или 1). У општијем случају, приликом решавања проблема са целобројним решењима, до оптималне вредности се долази заокруживањем

достигнутог реалног оптимума на најближи цео број (или се у околини добијеног решења тражи најбоље целобројно решење).

С обзиром да *PSO* алгоритам није прилагођен проблемима поделе графова, приликом решавања оваквих проблема се могу користити и хибридни алгоритми (*PSO* алгоритам у комбинацији са неким другим алгоритмом). На тај начин се врши побољшање решења, и евентуално, убрзава поступак долажења до оптимума. Тако је нпр. у [72] је извршена подела графа на две партиције помоћу комбинације бинарног *PSO* алгоритма и вишефазног алгоритма специјализованог за поделу графова. Поред тога, у појединим варијантама хибридног *PSO* алгоритма – *EPSO* (еволутивни *PSO*) [73], *PSO* алгоритам се проширује са операторима других еволутивних техника (селекцијом, мутацијом, укрштањем) у циљу бржег доласка до оптималног решења.

Адаптацијом *PSO* алгоритма на проблем расподеле графова, честице које представљају неку расподелу, имају димензију једнаку броју чворова који се расподељују. Позиције честица се ажурирају у одређеном броју итерација у складу са дефинисаним критеријумом оптималности, а тражи се најбоља позиција честице.

Промена позиције честице i се врши померањем честице из претходне позиције по формули:

$$x_i = x_i + v_i \quad (15)$$

где су:

- x_i – позиција честице i ,
- v_i – брзина кретања честице i .

Ажурирање брзине кретања v_i за сваку димензију:

$$v_i = W \cdot v_i + C_1 \cdot r_1 \cdot (b_i - x_i) + C_2 \cdot r_2 \cdot (b_g - x_i) \quad (16)$$

где су:

- W - фактор инерције,
- C_1 - фактор индивидуалности,
- C_2 - социјални фактор,
- $(b_i - x_i)$ - индивидуална компонента (b_i најбоља позиција честице i),
- $(b_g - x_i)$ - социјална компонента (b_g најбоља позиција свих честица),
- r_1 и r_2 - случајни бројеви из интервала $[0,1]$.

За ажурирану позицију x_i честице i израчунава се вредност критеријума оптималности $F(x_i)$. Позиција x_i се памти као најбоља на нивоу јединке b_i и популације b_g , уколико је боља од дотадашње најбоље позиције. Велика брзина кретања честица резултује значајном променом позиције, па се уводи ограничење за

израчунату брзину са максимално дозвољеном брзином (v_M):

$$v_i = \begin{cases} -v_M, & v_i < -v_M \\ v_i, & -v_M \leq v_i \leq v_M \\ v_M, & v_i > v_M \end{cases} \quad (17)$$

Како би се обезбедила разноврснија брзине кретања за све димензије честице, врши се и додатна модификација већ одређене брзине v_i , као у формули:

$$v_i = v_i \cdot \Delta x_{max} \cdot r \quad (18)$$

где је $\Delta x_{max} \cdot r$ распон за модификацију брзине (Δx_{max} је максимално могућа промена позиције, а r је случајан број $r \in (0, 1]$).

PSO је, у основи, континуалан алгоритам, стога га је потребно прилагодити разматраном дискретном проблему и дефинисати простор у коме се честице могу позиционирати. Прво се врши модификација брзине заокруживањем на најближи цео број, како би се добила дискретну вредност:

$$v_i = \text{round}(v_i) \quad (19)$$

Позиције честица морају имати целобројне вредности из интервала $[0, p)$, па се на самом крају ради модификација позиција по модулу p :

$$x_i = |\text{mod}(x_i, p)| \quad (20)$$

где је $\text{mod}(a, b)$ остатак при дељењу a са b .

Након извршавања дефинисаног броја итерација *PSO* алгоритма добија се расподела чворова по партицијама која је садржана у најбољој позицији честице на нивоу популације.

Дистрибуирани PSO алгоритам

Дистрибуирани *PSO* алгоритам се заснива на паралелном претраживању при чему се популација (рој) састоји из више дистрибуираних подројева смештених на различитим процесорима.

На сваком процесору се извршава алгоритам (одређени број итерација) над локалним подројем, независно од осталих подројева. Након унапред дефинисаног броја итерација, размењују се резултати добијени над различитим подројевима. Адаптација дистрибуираног алгоритма се своди на увођење новог параметра у *PSO* алгоритам - T_{sync} који ће одређивати фреквенцију комуникације између подројева. Параметар T_{sync} представља број итерација после којег ће се ажурирати глобални оптимум (b_g). Када подројеви добију информацију о глобалном оптимуму мењају брзину као што је наведено у једначинама (16)-(19).

Променом периода синхронизације мења се и начин извршавања алгоритма. У случају малих вредности T_{sync} подрејеви су више међусобно зависни и ефекат паралелизма мање долази до изражаја. Поред тога, алгоритам је спорији због чешће комуникације између подрејева. Са друге стране за веће вредности периода синхронизације, подрејеви су независнији, али се то може одразити на квалитет решења.

4.3. Алгоритми засновани на подели графа у више нивоа

У ову групу спадају алгоритми који проблем поделе решавају у више фаза и на више нивоа. Најпознатија група ових алгоритама су вишефазни алгоритми за поделу графова, које су идејно развили научници *Barnard* и *Simon* 1993. године [74] у оквиру *NASA*-иног истраживачког пројекта. Наредних 10-так година су се научници бавили истраживањима вишефазних алгоритама [59], [75]–[80], а најзначајније резултате су постигли *Henderson* (на бази својих истраживања је развио познати софтверски алат *Chaco* [78], [81]) и *Karypis* (развио је можда најпознатији и један од најчешће коришћених - *METIS* алгоритам [59]). Карактеристика ових алгоритама је велика брзина и могућност примене на графове великих димензија, а да се при томе добијају веома добри резултати за критеријумску функцију.

4.3.1. Вишефазни алгоритам за поделу графова

У овом раду је анализиран и имплементиран вишефазни алгоритам из [59] на коме је заснован и *METIS* алгоритам. Вишефазна подела графа се састоји из три фазе (слика 4.4.):

- 1) укрупњавања,
- 2) иницијалне поделе и
- 3) финије прерасподеле.

Идеја је да се проблем поделе графа G сведе на поделу мањег изведеног графа, чиме се смањује простор претраживања и повећава ефикасност алгоритма поделе.

Фаза укрупњавања: Почетни граф $G=(V,E)$ који се састоји из скупа чворова $V=\{v_1, v_2, \dots, v_p\}$ и грана $E=\{e_1, e_2, \dots, e_q\}$, се трансформише у мање графове G_1, G_2, \dots, G_m за које важи да је $|V| > |V_1| > |V_2| > \dots > |V_m|$ [59].

Трансформација графа G_i се врши тако што се поједини чворови графа спајају у један чвор v_x . Тај чвор v_x представља чвор графа G_{i+1} (нижег хијерархијског нивоа),

а гране које су сједињени чворови градили са осталим (са њима несједињеним у v_x) чворовима се третирају као гране чвора v_x са осталим чворовима. У [59] се предлаже више различитих алгоритама за укрупњавање чворова, као што су:

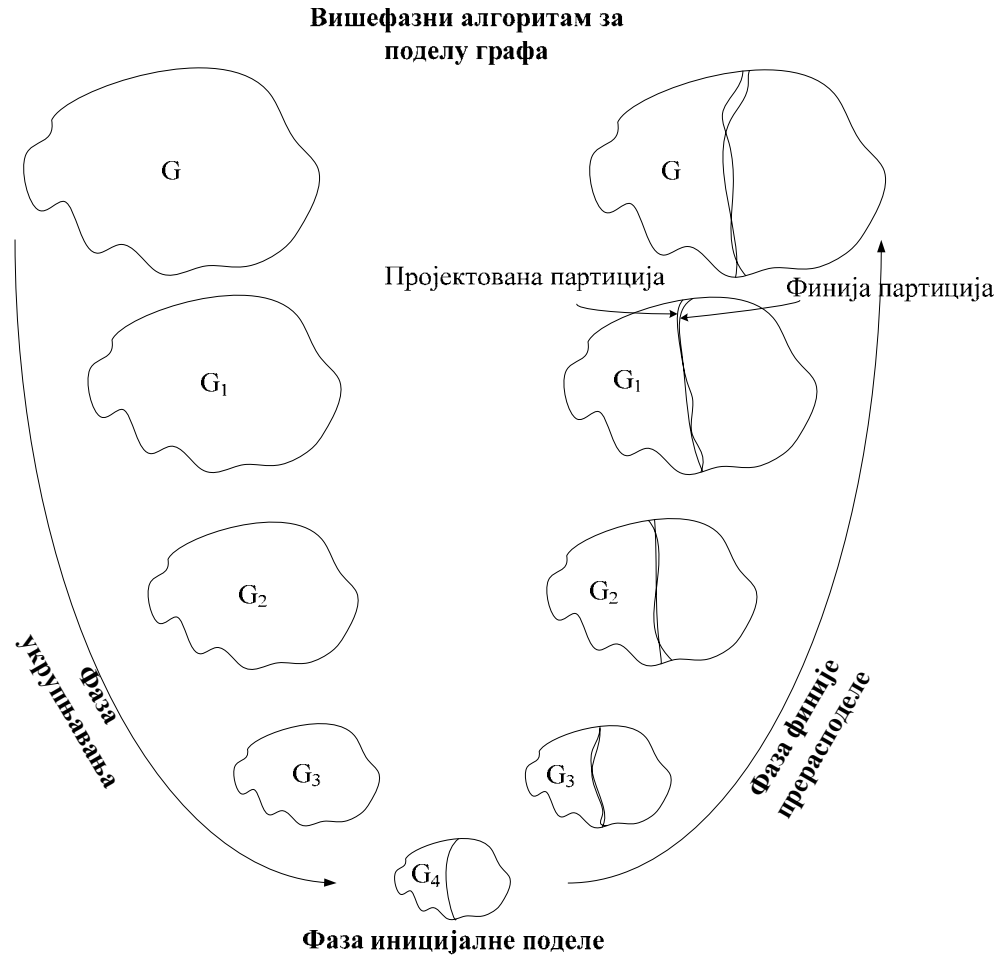
1. случајно сједињавање (*Random Matching - RM*) – сједињују се суседни неукрупњени чворови изабрани на случајан начин,
2. сједињавање по најтежим гранама (*Heavy Edge Matching - HEM*) – упарују се суседни чворови са максималним тежинама грана између њих,
3. сједињавање по „најлакшим“ гранама (*Light Edge Matching - LEM*) – спајају се чворови са најмањим тежинама грана између њих,
4. сједињавање по „најгушћим“ гранама (*Heavy Clique Matching - HCM*) – сједињују се чворови са максималним густинама грана [59].

На једном нивоу се примењује укрупњавање 2 или k чворова на све неупарене чворове на датом нивоу (које је могуће упарити). Број нивоа зависи од величине почетног графа, па се у случају великих графова (преко 100000 чворова) укрупњавање зауставља на графу од неколико десетина чворова. Уколико се ради о малим и слабо повезаним графовима ($|E| < |V|$) тада се алгоритмом укрупњавања често одрађује и друга фаза вишефазног алгорита – фаза иницијалне поделе графа.

Фаза иницијалне поделе: У овој фази се последњи укрупњени граф $G_m(V_m, E_m)$ дели на p партиција тако да величина партиције не прелази унапред задату дозвољену величину M уз задовољење критеријума оптималности F . Алгоритми коришћени за ову поделу су [59]:

1. *KL* алгоритам [60] – на случајан начин се изврши подела графа на партиције, па се након тога пребацују чворови из једне партиције у другу по дефинисаним критеријумима из *KL* алгоритма;
2. алгоритам нарастања чворова (енгл. *Graph growing partitioning algorithm - GGP*) – процес формирања партиција полази од случајно изабраног чвора, па се након тога скупљају сви чворови око њега, све док се не достигне жељена величина партиције, и тако за све партиције.
3. алгоритам „похлепног“ нарастања чворова (енгл. *Greedy graph growing partitioning algorithm - GGPP*) – за разлику од *GGP* алгоритма, приликом нарастања партиције, када се на случајан начин изабере један чвор за нову партицију, али се не узимају сви чворови који су повезани са њим, већ рачуна побољшање g_{ab} (према једначини (13)) за суседне чворове. Чвор за који се

добија највеће побољшање се убацује у партицију. Након тога се ажурира побољшање за све чворове који су суседни са пребаченим чвором. Поступак се понавља док се не добије жељена величина партиције, и тако за сваку партицију.



Слика 4.4. Вишефазна подела графа на две партиције

Фаза „финије” прерасподеле графа: Након извршене иницијалне поделе графа G_m приступа се побољшавању поделе тако што се подела графа G_m преводи у поделу графа из вишег нивоа G_{m-1} . Након тога се покушава побољшати подела пребацивањем чворова из скупа V_{m-1} (чворова графа са вишег нивоа) и на тај начин направити финију прерасподелу графа. Поступак се понавља (побољшањем поделе на следећем вишем нивоу графа) све док има побољшања поделе или док се не дође до највишег нивоа. За побољшање поделе, тј. пребацивање чворова између партиција користи се *KL* алгоритам, или његова варијанта - *FM* алгоритам.

4.3.2. Алгоритам суперкорена

Алгоритам суперкорена (енгл. *Super-Roots* - *SR*) је настао као резултат истраживања модела дистрибутивних електроенергетских мрежа [47]. Истраживања су спроведена над реалним моделима различитих дистрибутивних мрежа [30], [47], [48], а анализирани су и резултати из литературе [28], [82]. С обзиром да су основне карактеристике дистрибутивне мреже радијалност и слаба упетљаност, може се утврдити да су и области прорачуна за различите аналитичке електроенергетске функције слабо повезане. Уколико би се модел дистрибутивне мреже описао графом код кога би чворови одговарали областима прорачуна, а гране описивале повезаност између наведених области, тада би карактеристика таквог графа била мали број грана (испоставља се да је број грана мањи или приближно једнак броју чворова). Овакве графове карактеришу ретке матрице суседности, па се врло често природно групишу међусобно повезане области прорачуна које нису повезане са остатком мреже. Због тога се јавила идеја да се развије алгоритам који би решавао проблеме неповезаних графова.

За граф G се каже да је *неповезан* ако је могуће поделити његове чворове у два непразна скупа R и S , тако да ниједан чвор из скупа R није суседан са чвором у S [83]. Сви такви максимални графови се називају *повезане компоненте* [34], [83] или *суперчворови* (sn). У том случају, се неповезан граф може представити као скуп „изолованих“ суперчворова.

Суперчвор чија је тежина (збир тежина припадајућих чворова) већа од максимално дозвољене тежине за партицију M (једначина (8)) се назива *превелики суперчвор* и обележава се са sn_X . Са SN_X ће бити означен скуп превеликих суперчворова $SN_X = \{sn | w(sn) > M\}$ [47].

Суперчвор чија је тежина већа од 50% максимално дозвољене тежине [84] за партицију M назива се *велики суперчвор* (sn_L). Са SN_L је означен скуп великих суперчворова $SN_L = \{sn | w(sn) > 0.5 \cdot M\}$, ($SN_X \subset SN_L$) [47].

SR алгоритам је специјализован за проблеме партиционисања неповезаних графова. Овакви графови се могу представити скупом суперчворова (суперкорена). Уколико је потребно извршити поделу повезаног графа, добре особине SR алгоритма не могу доћи до изражаја и тада се SR алгоритам своди на $METIS$ алгоритам.

Алгоритам се састоји из три фазе [47]:

1. *Укрупњавања* – спајају се сви повезани чворови (корени) у један суперкорен

2. Подела превеликих суперкорена SN_X (нпр., употребом *METIS* алгоритма)
3. *Расподела*– подела бројева који представљају тежине неповезаних графова (коришћењем нпр. *Greedy* алгоритма за поделу бројева [85], [86]) и, у случају потребе, каснија подела *великих суперкорена* (SN_L).

Псеудо код *SR* алгоритма је следећи:

<pre> function parts=SR_Algorithm(G, p) Inputs: G=(V,E) - graph, V - set of vertices, E - set of edges, p - number of partitions Output: parts - solution of partitioning 1 SN ← Coarsening(G) 2 repeat 3 [SN_{XL}, SN_R] ← ClusterSupernodes(SN, M) 4 Θ ← Subdivision(SN_{XL}) 5 parts ← Partitioning(SNP, SN_R) 6 until all parts < M; M ← 0.5 M </pre>
--

Нотација:

Θ – скуп низова насталих за различите поделе превеликих суперкорена;

SN_R – низ суперкорена дозвољене величине.

Фаза *укрупњавања* се врши на сличан начин као код осталих вишефазних алгоритама [59], [78], али се код *SR* одједном сједињују сви међусобно повезани чворови (*Coarsening*). Укрупњени чворови се називају суперкорени и представљају скуп потенцијално повезаних региона.

Након фазе *укрупњавања*, врши се провера да ли је величина неког суперкорена већа од максимално дозвољене величине. Функција *ClusterSupernodes* разврстава суперкорене који су мањи од дозвољене величине партиције (M (SN_X) или $0.5M$ (SN_L)) у скуп SN_R , а оне веће у скуп SN_{XL} . Изглед псеудо кода функције *ClusterSupernodes* је следећи:

<pre> function [SN_{XL}, SN_R] = ClusterSupernodes(SN, X) Inputs: SN - set of super-nodes, X - maximal allowed weight of sn∈SN (M or 0.5M) Outputs: SN_{XL} (SN_R) - set of super-nodes which weights are (are not) larger than X, 1 SN_{XL} ← {}, SN_R ← {} 2 for each sn in SN 3 if w(sn)>X do 4 SN_{XL} ← SN_{XL} ∪ sn 5 else 6 SN_R ← SN_R ∪ sn 7 end if 8 end for </pre>
--

Уколико након фазе *укрупњавања* постоје превелики суперкорени, онда се врши њихова подела (фаза 2.), у супротном, прелази се на 3. фазу.

Подела превеликих суперкорена: превелике суперкорене је неходно изделити с обзиром да њихова величина превазилази дозвољену величину партиције. На основу дефинисане максималне дозвољене величине партиције M (8), минимални број делова ($nMin$) на које је потребно изделити суперкорен се одређује као:

$$nMin = \left\lceil \frac{W_{SN}}{M} \right\rceil, \quad (21)$$

али овај број може бити и већи, с обзиром да се мањи делови суперкорена “лакше” могу распоредити по партицијама у оквиру фазе 3. Из тог разлога оптималан број подподела није унапред познат и SR алгоритам врши више различитих подподела (линије 4-6 у функцији *Subdivision*): почиње са $nMin$ и повећава до укупног броја партиција – дефинисаног као максимални број подподела ($nMax=p$).

Након тога, потребно је применити неки алгоритам (нпр. METIS) за наведене варијанте партиционисања превеликих суперкорена (линија 5 у *Subdivision*). Псеудо код *Subdivision* фазе има следећи изглед:

```

function  $\Theta$  = Subdivision(SN)
Input:  SN - set of super-nodes that have to be divided
Outputs:  $\Theta$  - set of arrays of partitioning variants for extra
         large super-nodes
1  k←1
2  for each sn in SN do
3    j←1
4    for noParts←nMin to nMax
5       $\Theta[k][j] \leftarrow \Theta[k][j] \cup Metis(sn, noParts); j \leftarrow j+1$ 
6    end for
7    k←k+1
8  end for

```

Нотација:

$Metis(G, k)$ – METIS алгоритам за поделу графа G на k партиција.

Резултати партиционисања добијени поделом превеликих (или великих) суперкорена се називају *фрагменти суперкорена*. Функција F се израчунава за сваки sn_x и сваку варијанту партиционисања и формира се низ добијених решења (у опадајућем редоследу по вредностима за функцију F – линија 5 у функцији *Subdivision*). Ови низови су елементи скупа Θ (скуп низова са различитим подподелама за sn_x) и они се користе за утврђивање најбољег допустивог решења које се добија у следећој фази – фази расподеле.

На почетку фазе *расподеле* формира се скуп S од невеликих суперкорена SN_R и уситњених суперкорена. Ови фрагментисани суперкорени су изабрани као најбоље решење (највећа вредност функције F) из свих низова из скупа Θ . Задатак у овој

фази је расподела елемената из наведеног скупа S у задати број партиција. Из тог разлога се формира низ бројева који представљају величине свих (невеликих) суперкорена и уситњених суперкорена из датог скупа. Над оваквим низом се примењује неки од алгоритама за расподелу бројева [85], [86] (линија 4 у функцији *Partitioning*). Подељени бројеви указују на начин расподеле суперкорена, тј. њима припадајућих чворова. Уколико алгоритам за расподелу бројева није дао добар резултат, понавља се поступак распоређивања, при чему се узима следећа подела из вектора $\Theta[i]$ (линије 3-9 у *Partitioning*). Поред тога, ако постоји више подела (низова у Θ), избор следеће поделе треба да буде такав да она има најмање погоршање функције F (линија 4 у *next_division*) у односу на остале нивове (и потенцијалне кандидате за избор). Ако након свега није могуће добити балансиране партиције, алгоритам наставља са поделом великих суперкорена – SN_L (линија 6 у *SR_Algorithm: M* ← 0.5M).

Псеудо код фазе расподеле је следећи:

```
function part=Partitioning( $\Theta$ ,  $SN_R$ )
Inputs:  $\Theta$  - set of arrays of different subdivision of large super-nodes
         $SN_R$ - array of non large super-nodes
Output: part - solution of partitioning
1 di ← {1,1,...1} // division indexes
2 S ←  $SN_R \cup \{\Theta[di]\}$ 
3 do
4     parts = Number_Partitioning(S,n)
5     if all parts < M, return
6     di_new ← next_division( $\Theta$ , di)
7     S ← (S \ { $\Theta[di]$ }) ∪ { $\Theta[di\_new]$ }
8     di ← di_new
9 while di exist in divisions
```

Псеудо код поједностављене верзије *next_division* је следећи:

```
function di_next = next_division( $\Theta$ , di_old)
Inputs:  $\Theta$  - set of arrays of different subdivision
        di_old- division indexes for previous solution
Output: di next - division indexes for next solution
1 di = di_old
2 for each index in di
3     di[index] = di_old[index] + 1
4     if ( F( $\Theta[di]$ ) - F( $\Theta[di\_old]$ ) ) is min
5         return di
6 end for each
```

Пример: Подела модела дистрибутивне електроенергетске мреже коришћењем *SR* алгоритма

На слици 4.5. је приказан граф који представља пример дистрибутивне електроенергетске мреже. Њена структура је радијална и састоји се из 13 корена од којих су неки међусобно повезани, а неки усамљени. Гране графа представљају електричне елементе који су типа *Equipments*, а чворови графа представљају елементе типа *ConnectivityNode*.

Фаза I (Укрупњавање) – Граф чији чворови представљају корене се трансформише у граф суперкорена (слика 4.5.). Величине чворова (суперкорена) су једнаке збиру величина припадајућих корена.

При чему је:

$$SN_1 = \{R_1, R_2, R_3, R_4\}; \quad SN_2 = \{R_7, R_8\}; \quad SN_3 = \{R_9\}; \quad SN_4 = \{K_5, K_6\}; \\ SN_5 = \{R_{10}\}; \quad SN_6 = \{R_{11}\}; \quad SN_7 = \{R_{12}, R_{13}\};$$

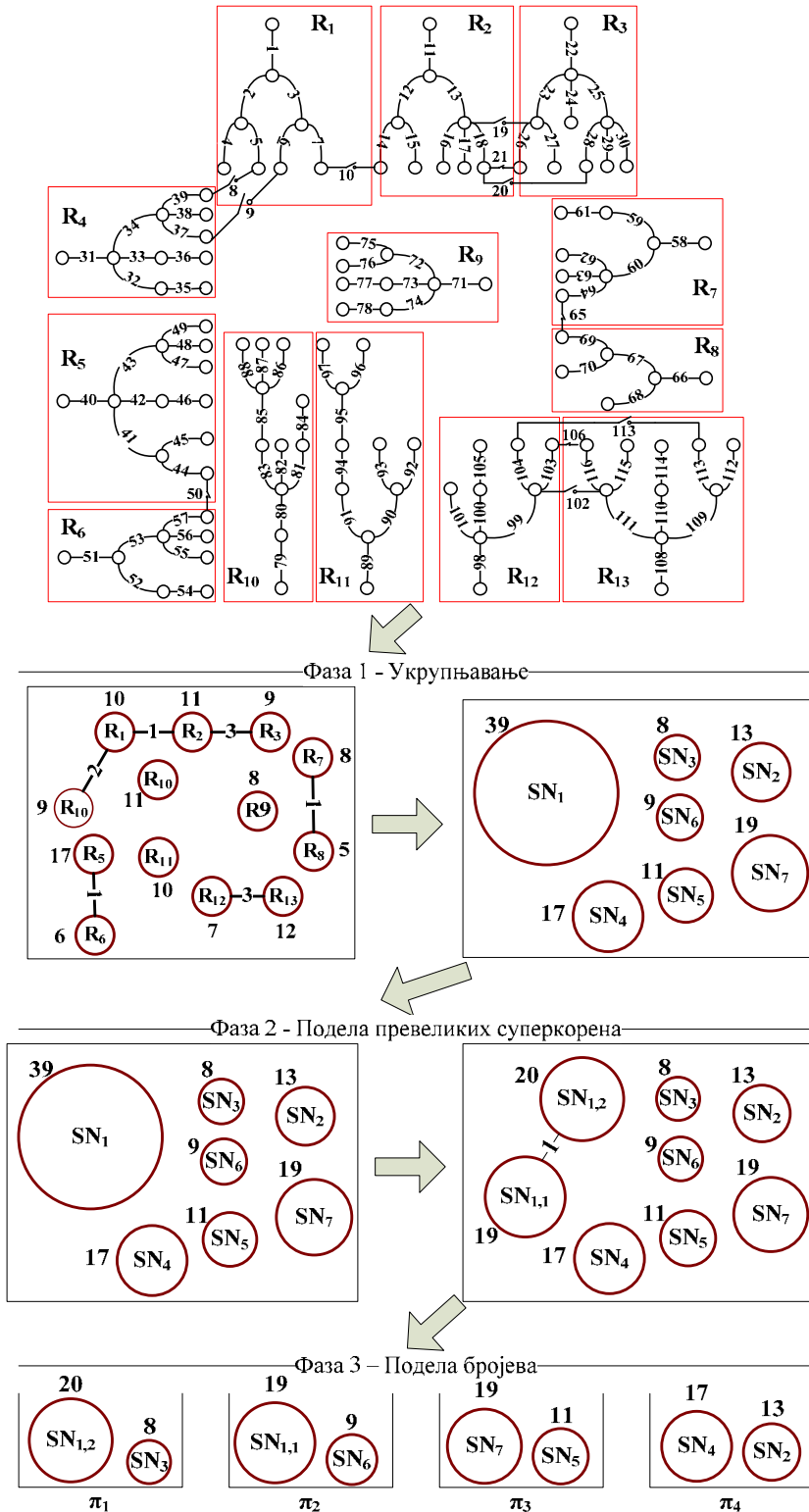
Као што се види са слике 4.5. на овај начин се добија граф чији су сви чворови неповезани (не постоје гране) и такав граф је улаз у фазу 2.

Фаза II (Подела превеликих суперкорена) – У овој фази се врши провера да ли је величина суперкорена већа од максимално дозвољене величине *M*. Уколико за сваки суперкорен важи да му величина није већа од *M* прелази се на 3. фазу. Такав је случај ако је потребно поделити модел са слике 4.5. (фаза 2) на два региона – сви суперкорени *SN* су мањи од *M* (без обзира на задату толеранцију, нпр. за толеранцију $\varepsilon=0.1, M=63.8$).

У супротном, потребно је поделити све суперкорене чија је величина већа од *M*. Поступак ће бити објашњен кроз наведени пример мреже и поделу на 4 региона са толеранцијом $\varepsilon=0.1$. Тада се добија да постоји превелики суперкорен *SN₁* чија је величина $w(SN_1)=w_{R1}+w_{R2}+w_{R3}+w_{R4}=10+11+9+9=39 > M \approx 32$ (9).

У том случају се применом неког од алгоритама за поделу графова (нпр. *METIS*) уз раније наведене критеријуме оптималности, могуће је поделити превелике суперкорене на потребан број делова. Према томе, решавање проблема у задатом примеру се своди на поделу суперкорена *SN₁* који се састоји из 4 корена. Претпоставимо да је *SN₁* подељен на два дела (слика 4.5.)

$$SN_1 = SN_{1,1} \cup SN_{1,2} \quad SN_{1,1} = \{R_1, R_4\} \quad SN_{1,2} = \{R_2, R_3\} .$$



Слика 4.5. Вишефазни алгоритам суперкорена

Фаза III (Расподела) – У овој фази се примењује један од алгоритама за поделу бројева, при чему бројеви представљају величине суперкорена. Након тестирања на

расположивим моделима је утврђено да се добри резултати добијају применом *Greedy* алгоритма (који је најједноставнији и најбржи).

Greedy алгоритам поделе на $p = 4$ партиције је заснован на следећим корацима:

1. Сортирају се бројеви у опадајућем редоследу – 20, 19, 19, 17, 13, 11, 9, 8
2. Првих p бројева се убацују у различитих p група ($p = 4$): $N_1 \leftarrow 20$, $N_2 \leftarrow 19$, $N_3 \leftarrow 19$, $N_4 \leftarrow 17$
3. Сваки следећи број се убацује у најмању групу и добија се: $N_1 = \{20, 8\}$, $N_2 = \{19, 9\}$, $N_3 = \{19, 11\}$, $N_4 = \{17, 13\}$.

На основу величина суперкорена добија се следећа подела суперкорена по партицијама:

$$\pi_1 = \{SN_{1,2}, SN_3\} = \{R_2, R_3, R_9\},$$

$$\pi_2 = \{SN_{1,1}, SN_6\} = \{R_1, R_4, R_{11}\},$$

$$\pi_3 = \{SN_7, SN_5\} = \{R_{12}, R_{13}, R_{10}\},$$

$$\pi_4 = \{SN_2, SN_4\} = \{R_5, R_6, R_7, R_8\},$$

при чему је $w(\pi_1) = 28$, $w(\pi_2) = 28$, $w(\pi_3) = 30$, $w(\pi_4) = 30$.

На крају се израчунава оптимизациона функција F (једначина (9)) као:

$$F = \phi_1 + \phi_2 + \phi_3 + \phi_4 \\ = (Pot_{R_2, R_3}) + (Pot_{R_1, R_4}) + (Pot_{R_{12}, R_{13}}) + (Pot_{R_5, R_6} + Pot_{R_7, R_8}) = 3 + 2 + 3 + (1 + 1) = 10$$

4.4. Хибридни алгоритми

Хибридизација алгоритама се врши у циљу побољшања ефикасности неког алгоритма који је коришћен за глобално претраживање. Побољшање алгоритма се огледа у бржем долажењу до задовољавајућег решења, као и добијању бољег решења [72], [87]-[90]. У даљем тексту ће бити приказана два хибридна алгоритма: *EPSO* (еволутивни *PSO* алгоритам) и *HGA* (хибридни генетски алгоритам).

4.4.1. Еволутивни *PSO* (*EPSO*) алгоритам

Еволутивни *PSO* алгоритам је подстакнут идејом да се креира алгоритам који би био комбинација *PSO* алгоритма и еволутивних алгоритама. Варијанта хибридизације алгоритма разматрана у овом раду је настала 2002. године као резултат истраживања које је спровео *Miranda* [73]. Специфичност алгоритма се огледа у томе што је у фази рекомбинације коришћено правило кретања честица дефинисано *PSO* алгоритмом. Разматрани параметри су груписани на објектне (позиција X) и стратегијске (тежине W).

EPSO алгоритам се извршава по следећим фазама [73]:

1. Репликација – свака честица се реплицира p пута;
2. Мутација – свака честица има мутиране стратегијске параметре;
3. Репродукција – мутирана честица ствара потомка на основу кретања честице заснованог на *PSO* алгоритму;
4. Евалуација – рачунање критеријумске функције за сваког потомка;
5. Селекција – на основу изабране процедуре (случајном, турнирском, пропорционалном и др.) најбоље честице опстају и преносе се на следећу генерацију.

Правило кретања честица у *EPSO* алгоритму је веома слично као код *PSO* алгоритма, тј. базира се на три компоненте: брзини, меморији и кооперацији. Разлика у односу на основни *PSO* алгоритам је мутација тежинских параметара. Применом мутације тежинских параметара смањује се могућност да честице достигну локални минимум и у њему остану.

Правило кретања *EPSO* алгоритма је формулисано на следећи начин:

$$\begin{aligned}x_i^{(k+1)} &= x_i^{(k)} + v_i^{(k+1)} \\v_i^{(k+1)} &= w_{i1}^* \cdot v_i^{(k)} + w_{i2}^* \cdot (b_i - x_i) + w_{i3}^* \cdot (b_g^* - x_i) \cdot P\end{aligned}\quad (22)$$

где су:

b_i - најбоље решење на нивоу честице i

b_g^* - мутирано најбоље решење на нивоу роја које се добија по формули:

$$b_g^* = b_g + \tau' \cdot N(0,1), \text{ при чему је}$$

b_g - најбоља честица ,

τ' – параметар који утичу на степен мутације,

$N(0,1)$ – нормална Гаусова расподела за случајну променљиву са нултом средњом вредношћу и варијансом једнаком 1

$x_i^{(k)}$ – позиција честице i у генерацији k

$v_i^{(k)} = x_i^{(k)} - x_i^{(k-1)}$ – брзина честице i у генерацији k

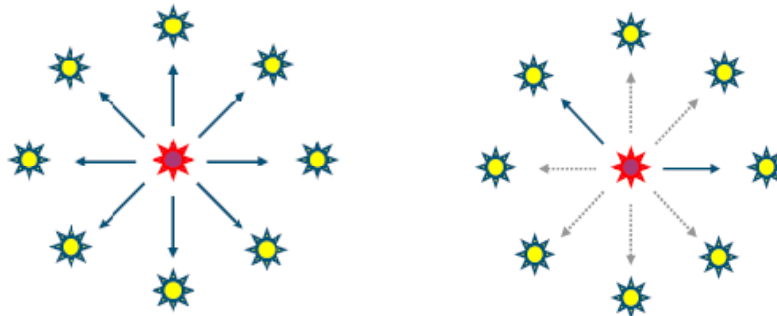
P - степен комуникације

$w_{i,1}^*$, $w_{i,2}^*$, $w_{i,3}^*$ –мутирани стратегијски параметри који се рачунају по формули:

$$w_i^* = w_i + \tau \cdot N(0,1) \quad (23)$$

τ – параметар који утичу на степен мутације.

На слици 4.6. су приказана два концепта комуникације између глобално најбоље честице и осталих честица. Стрелице представљене пуном линијом представљају слободан ток информација док испрекидана линија представља блокирање информација.



Слика 4.6. Илустрација топологије звезда и стохастичка звезда [91]

За разлику од *PSO* алгоритма код кога је комуникација између честица распоређена по тзв. звезда топологији у којој све честице у свакој итерацији имају информацију о глобално најбољој честици, *EPSO* алгоритам подржава комуникацију по принципу тзв. стохастичке звезде (слика 4.6.), која се заснива на степену комуникације P . Параметар P представља вероватноћу утицаја глобално најбоље честице на појединачну честицу. Смањењем размене информација између честица побољшава се локална претрага за сваку честицу и смањује могућност „заглављивања“ у локалном оптимуму. Вредност параметра комуникације P није фиксна и јединствена, већ се дефинише у зависности од проблема који се решава.

4.4.2. Хибридни генетски алгоритам

Хибридни генетски алгоритам је новоразвијени алгоритам који је настао у циљу побољшања решења генетског алгоритма за поделу графова. Он користи резултате *FM* алгоритма за побољшање локалног оптимума добијеног генетским алгоритмом. Иницијална популација се генерише на случајан начин (за сваки ген у свакој јединки број парције). Пошто *FM* алгоритам пребацује чвор из једне партиције у другу (код *GA* ово пребацавање се добија променом вредности одређеног гена), хибридизација генетског алгоритма је извршена у фази мутације. *HGA* користи класичну мутацију као и код генетског алгоритма, али се уводи и периодична *FM* мутација (сваких *freq* итерација).

Псеудо код *HGA* алгоритма се може приказати на следећи начин [48]:

```

Input: unpartitioned graph  $G=(V, E)$ 
Output: partitioned graph  $\Pi=\pi_1\cup\pi_2\cup\dots\cup\pi_p$ 
1  create initial population  $X=\{x_1, x_2, \dots, x_m\}$   $it \leftarrow 0$ 
2  repeat
3    select pairs  $(x_i, x_j)$  for crossover
4    for each pair  $(x_i, x_j)$ 
5       $offspring_{i,j} \leftarrow Crossover(x_i, x_j)$ 
6       $X = X \cup \{offspring_{i,j}\}$ 
7    end for
8     $X_{mut} \leftarrow$  select individuals for mutation
9    for each individual  $x_k \in X_{mut}$ 
10      $x_{nk} \leftarrow Mutation(x_k)$ 
11      $X = X \cup \{x_{nk}\}$ 
12    end for
13     $it \leftarrow it+1$ 
14    if  $it \% freq = 0$ 
15       $X_{FM} \leftarrow$  select individuals for FM
16      for each individual  $x_q \in P_{FM}$ 
17         $x_{nq} \leftarrow FM\_mutation(x_q)$ 
18         $X = X \cup \{x_{nq}\}$ 
19      end for
20    end if
21     $X \leftarrow$  choose the best  $m$  individuals from  $X$ 
22 until (no progress) or  $it > maxIt$ 

```

На почетку алгоритма се креира почетна популација на случајан начин (линија 1) и итеративним поступком се врши претрага простора могућих решења. Врши се избор јединки – родитеља за укрштање и укрштање према описаној процедури у генетском алгоритму (линије 3-7). Након тога се врши избор јединки за мутацију и класична мутација (линије 8-12), а уколико број текуће итерације одговара унапред дефинисаној фреквенцији – $freq$, врши се специјализована FM мутација (линије 14-20).

Алгоритам се завршава или када више нема побољшања решења (критеријумска функција нема побољшања након одређеног броја итерација) или након унапред задатог броја итерација $maxIt$ (линија 22).

5. Динамичка прерасподела графова

Под термином динамичка прерасподела графа подразумева прерасподелу чворова графа (задатака, података, и сл.) у току рада разматраног система. Најчешћа примена алгоритама за динамичку прерасподелу графа је за балансирање оптерећења процесора. Проблем динамичког балансирања оптерећења процесора у оквиру мултипроцесорским системима постао је актуелан 70-тих година прошлог века, а и данас се врше интензивна истраживања на овом пољу. Експанзија развоја мултипроцесорских рачунара утицала је на развој оперативних система који оптимизују оптерећење процесора и оптимално расподељују задатке који се у систему извршавају. Поред тога, специфичност функционисања појединих система (нпр. надзорно-управљачких система у електродистрибуцији) утицала је на додатна истраживања на алгоритмима за динамичко партиционисање.

Задатак алгорита за прерасподелу је успостављање балансираности партиција, при чему је потребно минимизирати следеће параметре [50], [51]:

- време извршавања алгорита T_a ,
- укупну количину података коју шаљу све партиције ($C_{migration}$),
- број грана (веза) између чворова из различитих партиција (C_{cut}).

Минимизацијом ових параметара смањује се време прерасподеле, тј. време од настанка небалансираног стања у систему до успостављања поновног балансираног стања система. Параметар $C_{migration}$ полази од претпоставке да ће се смањењем оптерећености мреже и укупне количине података који се пребацију из једне партиције у другу смањити време потребно за прерасподелу. Минимизација укупног броја грана (C_{cut}) између партиција смањује вероватноћу да дође до дисбаланса партиција под утицајем активације неке од наведених потенцијалних конекција.

Због тога се на основу израчунатих параметара често формира критеријум за вредновање квалитета пребацивања чворова из једне партиције у другу [92]:

$$F_{repart} = \alpha C_{cut} + \beta C_{migration} + \gamma C_{balance} \quad (24)$$

где су:

α , β и γ - који служе за фаворизовање неког од одговарајућих параметара у критеријумској функцији.

$C_{balance}$ – степен балансираности партиција који се израчунава као:

$$C_{balance} = \sum_{i=1}^p (W_{\pi_i} - \overline{W_{\pi}})^2 \quad (25)$$

при чему је $\overline{W_{\pi}} = \frac{\sum_{i=1}^p W_{\pi_i}}{p}$ - просечна тежина партиција.

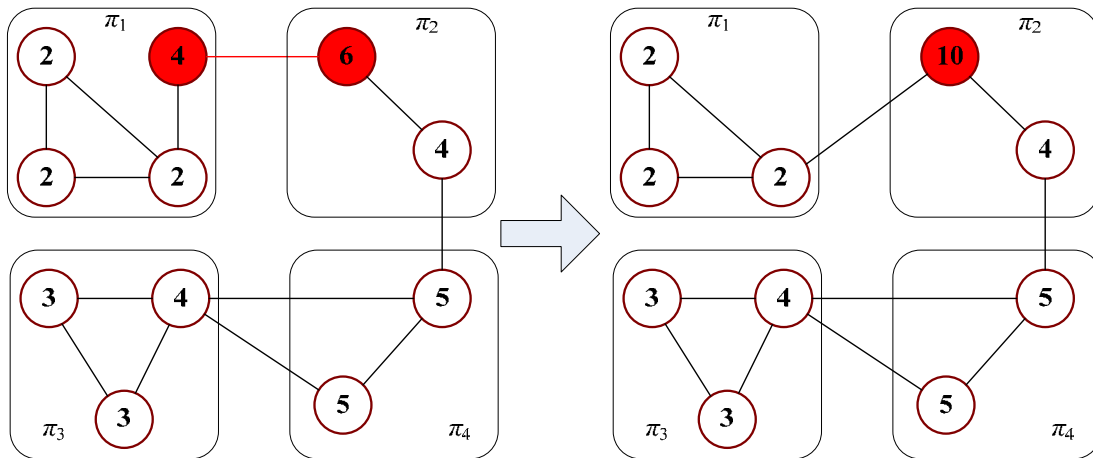
У групу најчешће коришћених техника динамичког балансирања оптерећења се могу сврстати у дифузиони алгоритми и алгоритми мапирања (*scratch-remap*). Оне минимизирају количину података која мигрира уз задовољење балансираности и минимизацију утицаја између процесора (партиција). Већина алгоритама је базирана на графовима, па се и проблем динамичког балансирања процесора своди на проблем динамичке прерасподеле графа.

Један од иницијатора динамичког балансирања базираног на дифузионим процесима је *Cybenko* [93]. Он је извео математички формализам за дифузионо преливање података у мултипроцесорским системима. Следећи његова истраживања развијени су други дифузиони алгоритми за динамичку прерасподелу. У [94] је први пут описан дифузиони алгоритам који користи Лангранжову матрицу за израчунавање тзв. Лагранжових множитеља преко којих се одређује прерасподела чворова. Дифузиони алгоритми су показали боље резултате од алгоритама базираних на мапирању, са аспекта количине података који мигрирају и C_{cut} - броја пресечних грана између партиција [50], [51]. У последње време се развијају и динамички алгоритми [95] који дају знатно боље резултате за C_{cut} и $C_{migration}$. Међутим ови алгоритми су знатно спорији (веће T_a) и немају примену у реалном времену код система са брзом динамиком. Они се могу користити у системима са споријом динамиком (нпр. надзорно-управљачким системима за дистрибуцију воде, термичким системима и сл.).

5.1. Иницијализација динамичке прерасподеле

Динамичка прерасподела графа се врши у току рада система над већ издељеним графом. Иницирана је променама графа које узокују небалансираност партиција у оквиру већ издељеног графа. Пример небалансираног издељеног графа је приказан на слици 5.1.b (ознаке у чворовима представљају тежине чворова). На

слици 5.1.a је приказан граф са балансираним партицијама (свака партиција је тежине 10), а након активирања потенцијалне конекције између чворова са тежинама 4 и 6 (црвена грана на слици) долази до њиховог спајања. Након тога, чвор мање тежине се пребацује у партицију π_2 и она постаје превелика ($W(\pi_2) = 14$). Када се утврди небалансираност графа – иницира се динамичка прерасподела графа.



а) балансиране партиције

б) небалансиране партиције

Слика 5.1. Промена графа услед затварања потенцијалне конекције

До промене графа долази под утицајем спољашњих измена модела, које могу бити:

1. додавање елемената у модел,
2. брисање елемената из модела, и
3. промена активности потенцијалних конекција (релација) између појединих елемената (нпр. отварање или затварање прекидача).

Разликују се следећи случајеви додавање елемената у модел:

- Додавање елемената одређеном корену – ова измена не утиче на везе између корена, већ само повећава тежину чвора који представља дати корен. Услед повећања тежине чвора може доћи до небалансираности, односно, партиција којој је додељен наведени чвор може бити превелика.
- Додавање елемента (или групе елемената) које узрокује повећање броја потенцијалних конекција између појединих корена. На тај начин се повећава тежина гране између два корена, као и тежине корена, што може да узрокује неоптималну вредност критеријумске функције и небалансираност партиција. Уколико су након ове измене модела све партиције остале балансиране – динамичка прерасподела неће бити покренута.

- Додавање елемената који узрокоју спајање два или више корена. Уколико се спојени корени налазе у различитим партицијама, може доћи до набалансираности партиција и покретања динамичке прерасподеле.

Брисање елемената може довести до смањења броја елемената у корену (смањења тежине чвора), до губитка одређених потенцијалних конекција између појединих корена (смањења тежине грана), или до укидања веза између корена (брисање гране). Према томе, брисање елемената као изолована активност не иницира динамичку прерасподелу модела.

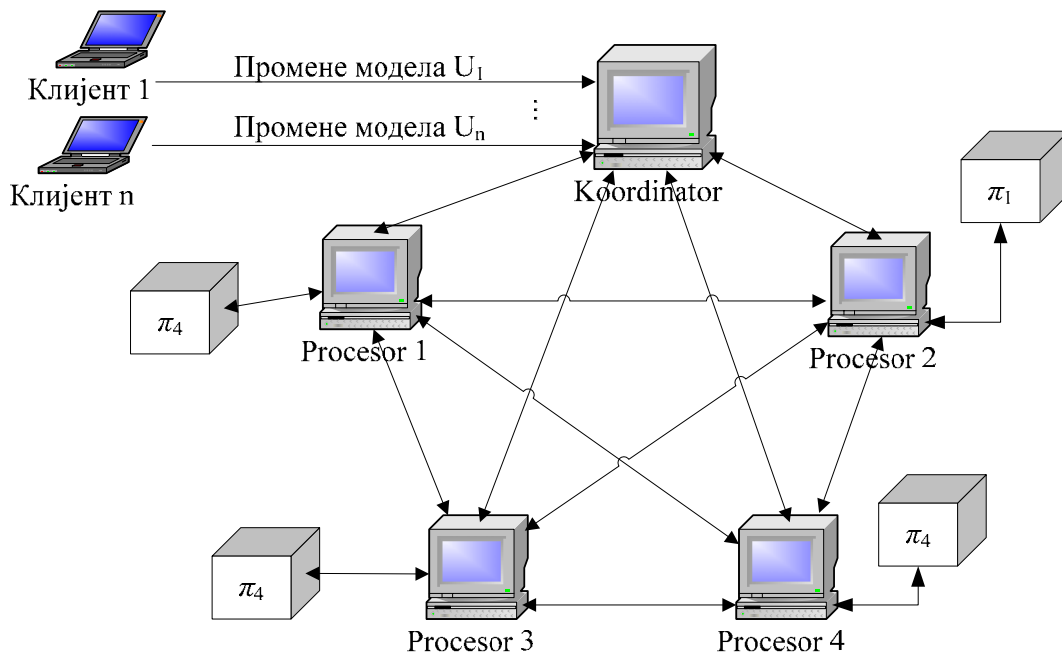
Промена активности потенцијалних конекција између појединих елемената може узроковати:

1. Спајање два корена – уколико је активирана потенцијална конекција између њих (затварање граничних прекидача). У том случају се два чвора графа спајају у један, и уколико су били у различитим партицијама може доћи до небалансираности система, и покретања динамичке прерасподеле.
2. Поделу корена – деактивацијом потенцијалне конекције (отварањем расклопне опреме) може доћи до стварања острва (ненапојеног дела мреже) и у таквим ситуацијама се не врши динамичка прерасподела. Уколико се, том приликом створено острво, припоји неком корену из друге партиције (затварањем расклопне опреме) – може доћи до покретања динамичке прерасподеле у случају небалансираног система.

5.2. Архитектура дистрибуираног система

Разматрана је архитектура дистрибуираног система у оквиру којег се алгоритми извршавају централизовано (што је приказано на слици 5.2.). Један рачунар у оквиру система (координатор) има информације о локацији свих елемената дистрибутивне мреже и њиховом стању (отворен или затворен елемент расклопне опреме). Координатор прима информације од клијената о промени стања расклопне опреме и прослеђује их рачунарима на којима се обрађују реални подаци о елементима мреже. У случају затварања граничних прекидача између два корена који се налазе у различитим партицијама долази до преласка мањег корена (оног са мањим бројем елемената) у партицију већег. Пошто координатор има информације о локацији елемената, он може да прорачуна да ли је партиција која је примила корен постала преоптерећена (превелика). Када установи да је нека од партиција постала

преоптерећена, координатор позива алгоритам за прерасподелу који одређује чворове (групе елемената) које је потребно мигрирати из једне партиције у другу како би се успоставило ново балансирано стање. Затим иницира процес прерасподеле и шаље информације о прерасподели рачунарима који обрађују податке. Током процеса прерасподеле подаци одређени за промену партиције нису у функцији (не могу се користити за прорачуне или мењати њихови атрибути) све до завршетка процеса прерасподеле. Сви процесори примају информације о прерасподели од координатора и започињу процес размене података.



Слика 5.2. Архитектура дистрибуираног система са 4 процесора

5.3. Алгоритми за динамичку прерасподелу

Заједничка карактеристика свих алгоритми за прерасподелу графа је да се састоје из следећих фаза:

1. избор донора - партиције из које се пребацује чвор у другу партицију,
2. избор партиције у коју се пребацује чвор,
3. избор чвора који се пребацује.

Анализирани су и развијени следећи алгоритми:

- *WF* (енгл. *WaveFront*) дифузиони алгоритам [51],
- модификовани *WF* (*MWF*) дифузиони алгоритам [96],
- *CP* (енгл. *Cut-and-Paste*) алгоритам [50],

- Угњежено паралелно репартиционисање (енгл. *Parallel Nested Repartitioning –PNR*) алгоритам [92], [97],
- Локално упарено мапирање у више нивоа (енгл. *Locally-Matched Multilevel Scratch-Remap – LMSR*) алгоритам [51].

С обзиром на различите архитектуре вишепроцесорских система, развијени су и алгоритми који су прилагођеним хијерархијским вишепроцесорским архитектурама. Анализом реалних дистрибутивних електроенергетских мрежа утврђено је да се ради о специфичним графовима (код којих је број грана приближно једнак броју чворова) [47], [48], па је значајно применити наведене алгоритме на графове који описују модел дистрибутивне мреже. На тај начин се добијају резултати који су релевантни за разматрани проблем поделе података у оквиру дистрибутивне мреже, а који се могу разликовати од тестова изведеним у [50], [51], [97].

5.3.1. *WaveFront (WF)* алгоритам

WF алгоритам [50], [51] спада у групу дифузионих алгоритама код којих се чворови приликом балансирања крећу у виду „таласа“ од преоптерећених партиција ка мало оптерећеним. То је итеративни поступак који се састоји из следећих фаза:

1. избор партиције – донора π_s , и
2. избор чвора који се пребације v_i .

Поступак се понавља све док постоје небалансиране партиције.

У свакој итерацији партиција - донор (π_s) пребације чвор другој партицији у циљу успостављања балансираног стања. Након тога, понавља се поступак избора донора и пребацивања неког чвора, све док се не успостави балансирано стање свих партиција. Приликом избора чвора који се пребације, предност имају чворови који се не налазе у својој матичној партицији (партиција у којој су се налазили на почетку алгоритма), чиме се смањује укупан број података који мењају партицију (минимизује се параметар $C_{migration}$) [98].

Поред раније наведених параметара који се узимају у обзир у динамичким алгоритмима поделе, *WF* алгоритам има задатак да минимизује параметар C_{maxV} (максималну количину података коју шаље/прима неки процесор). Параметар C_{maxV} је показатељ максималног времена које је потребно да нека партиција пошаље или прими податке.

Избор партиције донора

Прва фаза алгоритма је избор партиције донора. Бира се партиција код које је највећи однос величина *outflow* и *inflow* ($\pi_S = \max(\frac{outflow}{inflow})$). П)оступак добијања ове две величине се своди на решавање система линеарних једначина [94]:

$$L \cdot \lambda = b, \quad (26)$$

при чему је:

$$L = \begin{bmatrix} \text{deg}(1) & -1/0 & \dots & -1/0 \\ -1/0 & \text{deg}(2) & \dots & -1/0 \\ -1/0 & -1/0 & \ddots & -1/0 \\ -1/0 & -1/0 & \dots & \text{deg}(p) \end{bmatrix}_{p \times p}, \lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_p \end{bmatrix}_p, b = \begin{bmatrix} W_{\pi_1} - \bar{W}_{\pi} \\ W_{\pi_2} - \bar{W}_{\pi} \\ \vdots \\ W_{\pi_p} - \bar{W}_{\pi} \end{bmatrix}_p$$

где је

- L - тзв. Лапласова матрица димензије $p \times p$, при чему је:

$$L_{i,j} = \begin{cases} \text{deg}(i), & i = j \\ -1, & i \neq j, e(i,j) \in E \\ 0, & \text{остало} \end{cases} \quad (27)$$

- p – укупан број партиција,
- $\text{deg}(i)$ – укупан број веза који партиција π_i има са чворовима који се налазе у другим партицијама,
- λ – вектор решења, тзв. *дифузионо решење*.

За две партиције се каже да су суседне, ако постоји бар једна потенцијална конекција између њихових чворова (корена). Елементи вектора слободних чланова b се израчунавају као разлика између тренутне тежине партиције и просечне тежине партиција - \bar{W}_{π} :

$$b_i = W_{\pi_i} - \bar{W}_{\pi} \quad (28)$$

Вектор λ се добија *Gauss-Seidel* методом за нумеричко решавање система линеарних једначина. Затим се на основу дифузионог решења λ , за сваку партицију π_i ($i = 1, \dots, p$) израчунава вектор X_i . Елементи вектора X_i се добијају на следећи начин:

$$X_i(j) = \begin{cases} \lambda_i - \lambda_j, & i \neq j \\ 0, & i = j \end{cases} \quad j = 1, \dots, p \quad (29)$$

Параметар $outflow_{\pi_i}$ се добија као збир позитивних елемената вектора X_i :

$$outflow_{\pi_i} = \left\{ \sum_j X_i(j) \mid X_i(j) > 0 \right\} \quad (30)$$

Параметар $inflow_{\pi_i}$ се добија као збир апсолутних вредности негативних елемената вектора X_i :

$$inflow_{\pi_i} = \left\{ \sum_j |X_i(j)| \mid X_i(j) < 0 \right\} \quad (31)$$

Партиција за коју је добијен максималан однос $outflow/inflow$ се бира за донора.

Избор чвора који се пребацује

Други корак у WF алгоритму је избор чвора који шаље партиција донор. При томе се разматрају гранични чворови (који имају везе са чворовима из других партиција). Прва фаза при избору чвора је сортирање листе граничних чворова. Сортирање се врши у следећем редоследу:

1. чворови који имају бар једну везу са чвором који је у некој мало оптерећеној партицији (*underweight* - чија је тежина мања од $\bar{W}_{\pi}/(1+\epsilon)$),
2. чворови који имају бар једну везу са чвором који је у неком преоптерећеној партицији (*overweight* – чија је тежина већа од максимално дозвољене тежине партиције - M),
3. чворови који су суседни само са чворовима који се налазе у балансираним партицијама (да се не би нарушила балансираност партиција остварена у претходним итерацијама).

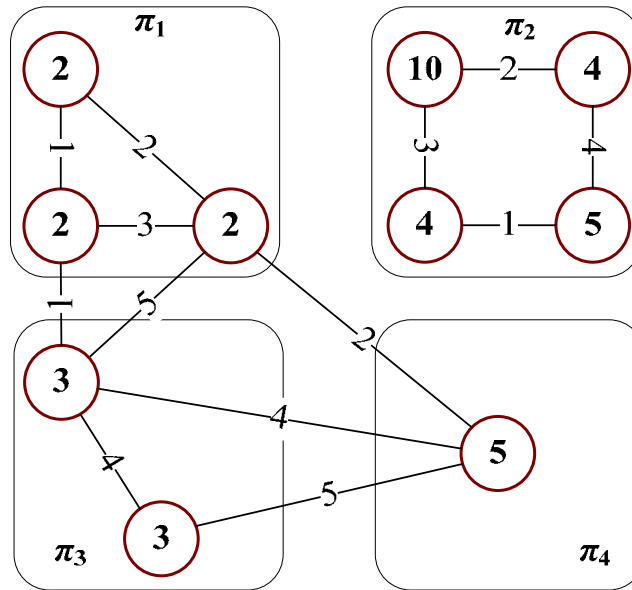
Приликом слања чворова предност увек имају чворови који се не налазе у својој матичној партицији, они се још називају и посећени чворови. Оваквом стратегијом слања се постиже да се такви чворови више пута пребацују из партиције у партицију, а да при томе не повећавају укупну количину података која се пребацују.

У случају да је чвор који је кандидат за слање суседан са више балансираних партиција, чвор се шаље се у ону партицију са чијим чворовима има више отворених веза. На тај начин се минимизује укупан број пресечних грана између партиција - C_{cut} .

С обзиром да је WF алгоритам првенствено развијен за динамичко партиционисање великих и добро повезаних графова, потребно је додатно дефинисати критеријуме који омогућују прерасподелу слабо повезаних графова. Наиме, код слабо повезаних графова постоје случајеви када су неки делови графа

изоловани, тј. када је граф неповезан. Овакви изоловани делови немају везу са окружењем, па је потребно проширити критеријум за пребацивање чворова.

Слика 5.3. приказује случај неповезаног графа код кога је партиција π_2 изолована и преоптерећена.



Слика 5.3. Пример неповезаног графа са преоптерећеном изолованом партицијом π_2

На основу описаног поступка добија се Лапласова матрица:

$$L = \begin{bmatrix} 8 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 15 & -1 \\ -1 & 0 & -1 & 11 \end{bmatrix}$$

Друга врста (и друга колона) матрице L имају све нулте елементе, па се на основу тога може закључити да је друга партиција изолована. Због тога, проширење WF алгоритма у оваквим ситуацијама представља додатна провера да ли је изолована партиција преоптерећена, и ако јесте она се проглашава за донора. У случају ако има више изолованих преоптерећених партиција, развијени алгоритам донором проглашава прву од њих (ону са најмањим индексом).

Псеудо код алгоритма би имао следећи изглед:

```

function WF_Algorithm( $G_p$ )
  Input\Output:  $G_p=(V,E)$  - partitioned imbalanced\balanced graph
  Calculate  $L, b, \lambda$ 
  for each partition  $\pi_i$  in  $\Pi$  do
    Calculate  $X_i$ 
     $outflow_{\pi_i} \leftarrow CalculateOutflow(\pi_i)$ 
     $inflow_{\pi_i} \leftarrow CalculateInflow(\pi_i)$ 
  end for
  while ( $G_p$  is imbalanced) or (no balancing progress) do
    if isolated imbalanced partition exists
       $\pi_S \leftarrow$  isolated imbalanced partition
    else
       $\pi_S \leftarrow BestRatioPartition(inflow / outflow)$ 
    end if
     $R \leftarrow SelectRegion(\pi_S)$ 
     $\pi_D \leftarrow GetBestNeighbour$ 
     $\pi_S \leftarrow \pi_S \setminus \{R\}, \pi_D \leftarrow \pi_D \cup \{R\}$ 
    Calculate  $L, b, \lambda$ 
    for each  $\pi_x$  in  $\{\pi_S, \pi_D, N(\pi_S), N(\pi_D)\}$  do
      Calculate  $X_i$ 
       $outflow_{\pi_x} \leftarrow CalculateOutflow(\pi_x)$ 
       $inflow_{\pi_x} \leftarrow CalculateInflow(\pi_x)$ 
    end for
  end while

```

5.3.2. Модификовани *WF* (*MWF*) алгоритам

MWF алгоритам [96] је настао по угледу на *Wavefront* алгоритам [51]. Псеудо код *MWF* алгоритма је:

```

function MWF_Algorithm( $G_p$ )
  Input\Output:  $G_p=(V,E)$  - partitioned imbalanced\balanced graph
  for each partition  $\pi_i$  in  $\Pi$  do
     $outflow_{\pi_i} \leftarrow CalculateOutflow(\pi_i)$ 
     $inflow_{\pi_i} \leftarrow CalculateInflow(\pi_i)$ 
  end for
  while ( $G_p$  is imbalanced) or (no balancing progress) do
     $\pi_S \leftarrow BestRatioPartition(inflow / outflow)$ 
     $R \leftarrow SelectRegion(\pi_S)$ 
     $\pi_D \leftarrow GetBestNeighbour$ 
     $\pi_S \leftarrow \pi_S \setminus \{R\}, \pi_D \leftarrow \pi_D \cup \{R\}$ 
    for each  $\pi_x$  in  $\{\pi_S, \pi_D, N(\pi_S), N(\pi_D)\}$  do
       $outflow_{\pi_x} \leftarrow CalculateOutflow(\pi_x)$ 
       $inflow_{\pi_x} \leftarrow CalculateInflow(\pi_x)$ 
    end for
  end while
  while  $G_p$  is imbalanced do
     $R \leftarrow GetBestMoving, R \in \pi_S, \pi_D \leftarrow \min(W_\pi)$ 
     $\pi_S \leftarrow \pi_S \setminus \{R\}, \pi_D \leftarrow \pi_D \cup \{R\}$ 
  end while

```

MWF (као и WF алгоритам) је итеративни процес заснован на прорачуну два низа: $outflow$ и $inflow$. Оно што овај алгоритам разликује од WF алгоритма је начин одређивања параметара $outflow$ и $inflow$. Параметар $outflow_{\pi_i}$ представља збир тежина чворова које би партиција π_i требала да пошаље осталим партицијама. Она се израчунава ($CalculateOutflow$) као разлика између тежине партиције W_{π_i} (из једначине (6)) и M (из једначине (8)):

$$outflow_{\pi_i} = W_{\pi_i} - M \quad (32)$$

Параметар $inflow_{\pi_i}$ представља збир величина чворова које партиција π_i треба да прими од осталих партиција. Он се израчунава ($CalculateInflow$) као збир $inflow$ параметара израчунатих за партиције које су суседне са партицијом π_i ($\pi_j \in N(\pi_i)$):

$$inflow_{\pi_i} = \sum_{\pi_j \in N(\pi_i)} (W_{\pi_j} - M) \quad (33)$$

У свакој итерацији се бира чвор R ($SelectRegion$) из партиције (π_S) која има најбољи однос $outflow/inflow$ ($BestRatioPartition$) и он се пребацује у другу партицију (π_D) у циљу побољшања балансираности и функције F ($GetBestNeighbour$) [96]. У овом алгоритму се избор чвора (региона) за пребацивање из једне партиције у другу ($SelectRegion$) врши на основу критеријумске функције F_{repart} (дефинисане једначином (24)).

Ако балансираност није постигнута, најслабије повезан чвор (регион - R) у свим преоптерећеним партицијама (који се проналазе преко функције $GetBestMoving$) се пребацује у најмању партицију. На тај начин је омогућено пребацивање чворова и у случајевима неповезаних графова.

5.3.3. *Cut-and-Paste (CP)* алгоритам за прерасподелу графова

У оквиру овог алгоритма [50] сваки гранични чвор (регион) (BR – скуп граничних региона) се посећује на случајан начин. Уколико је регион у некој преоптерећеној партицији и ако је суседни са неким чвором из партиције која је мало оптерећена, тада се дати чвор пребацује у мало оптерећену партицију (али само у случају да није нарушена балансираност циљне партиције). Уколико разматрани регион има више суседа из мало оптерећених партиција, тада ће он бити пребачен у партицију која ће највише допринети побољшању оптимизационе функције F ($GetBestNeighbour$). Након што је сваки гранични чвор посећен тачно једном, поступак се понавља док све партиције нису избалансиране или када није могуће

побољшати балансираност. У случају да је на крају систем остао неизбалансиран, врши се пребацивање најслабије повезаног региона (*GetBestMoving*) из преоптерећене партиције у најмању партицију.

Псеудо код *CP* алгоритма [50] је формулисан на следећи начин:

```
function CP_Algorithm( $G_p$ )
Input\Output:  $G_p=(V,E)$  - partitioned imbalanced\balanced graph
repeat
  for each non visited  $R_b$  in BR do
     $R_b \leftarrow$  visited
    if ( ( $R_b \in \pi_x$ )  $\wedge$  ( $W_{\pi_x} > M$ ) )
       $\pi_k \leftarrow$  GetBestNeighbour
       $\pi_x \leftarrow \pi_x \setminus \{R_b\}$ ,  $\pi_k \leftarrow \pi_k \cup \{R_b\}$ 
      BR  $\leftarrow$  BR  $\cup$  { $R_y \in \pi_x \mid (R_y, R_b) \in E$ }
    end if
  end for
until ( $G_p$  is balanced) or (no balancing progress)
while  $G_p$  is imbalanced do
   $R \leftarrow$  GetBestMoving,  $R \in \pi_s$ ,  $\pi_D \leftarrow \min(W_{\pi})$ 
   $\pi_s \leftarrow \pi_s \setminus \{R\}$ ,  $\pi_D \leftarrow \pi_D \cup \{R\}$ 
end while
```

5.3.4. *LMSR* алгоритам

LMSR је алгоритам код кога се прерасподела чворова одређује на основу поређења унапред дефинисаног балансираног стања ($\Pi^b = \{\pi_1^b, \pi_2^b, \dots, \pi_p^b\}$) и тренутног - небалансираног стања издељеног графа ($\Pi^n = \{\pi_1^n, \pi_2^n, \dots, \pi_p^n\}$). Балансирано стање графа представља унапред дефинисани циљ, тако да је резултат расподеле код овог алгоритма унапред познат. То значи да су унапред познати број пресечних грана C_{cut} и степен балансираности $C_{balance}$ из критеријумске функције F_{repart} . Параметри који нису унапред познати су укупна количина података која се пребације - $C_{migration}$ и брзина извршавања алгоритма T_a .

LMSR алгоритам се састоји из следећих корака:

1. дефинисање улаза: циљни - балансиран граф - Π^b и тренутни – небалансиран граф - Π^n ,
2. формирање матрице сличности Q ,
3. селектовање максималних елемената - мапирање партиција.

Дефинисање циљног издељеног графа се врши неким од алгоритма за иницијалну расподелу пре покретања система.

Формирање матрице сличности. Поступак одређивања прерасподеле чворова се заснива на тзв. *матрици сличности* Q . Матрица сличности има следеће карактеристике:

1. димензије матрице су $p \times p$, где је p број партиција;
2. врсте матрице представљају партиције балансираног стања, а колоне матрице представљају новонастале партиције (небалансираног графа);
3. елемент матрице Q_{ij} представља збир тежина чворова који се налазе и у партицији π_i^b и у партицији π_j^n ,

Мапирање партиција. Након формирања матрице сличности потребно је изабрати по један елемент из сваке врсте и колоне тако да збир изабраних елемената буде максималан.

Резултат мапирања партиција је упаривање партиција балансираног и небалансираног графа, чиме се утврђује који се чворови већ налазе у жељеним партицијама, а који треба да мигрирају. Идеја је да се партиције које са највећим бројем истих података у оба издељена графа (балансирани и небалансирани) третирају као иста партиција. На тај начин се минимизује укупна количина података коју је потребно пребацити $C_{migration}$, као и највећа количина података коју шаље/прима неки регион C_{maxV} .

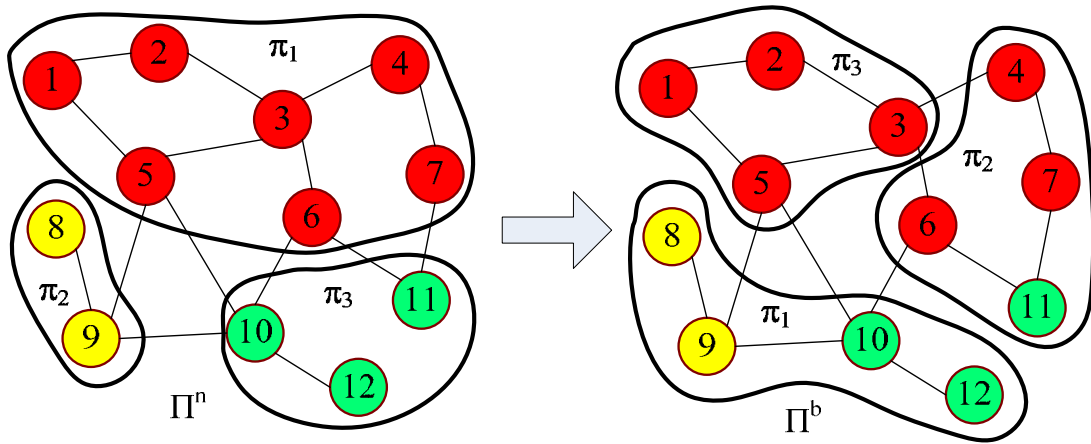
Уколико је изабран елемент Q_{ij} то значи да се мапирају партиције π_i^n и π_j^b , тј. оне се односе на исту циљну партицију π_j (на почетку прерасподеле је $\pi_j = \pi_i^n$, а након прерасподеле $\pi_j \cong \pi_j^b$). С обзиром да је почетно стање графа – неизбалансирана подела Π^n , сви чворови који се налазе у циљној партицији π_j^b , а не налазе се у партицији π_i^n - морају да се пребаце у партицију π_j . Са друге стране, ако се чвор налази у партицији π_i^n , а не налази се у партицији π_j^b биће пребачен из партиције π_j у неку другу партицију.

Поступак мапирања може бити прилично сложен, поготову у случајевима када се врши прерасподела на већи број партиција, тј. када је матрица Q већих димензија. Тада је за одређивање оптималног мапирања користи неки од оптимизационих алгоритама. Нпр. један од погодних алгоритама је и *Greedy* алгоритам код кога се сви елементи матрице сличности сортирају у један низ у опадајућем редоследу. Затим се итеративно пролази кроз елементе низа и проверава да ли се елемент

налази у резервисаној врсти и колони. Уколико његова врста и колона нису резервисани он се проглашава изабраним елементом и резервишу се његова врста и колона.

Пример: Примена *LMSR* алгоритма за прерасподелу графа на 3 партиције

Нека су дефинисани улази: тренутни – небалансиран граф – ($\Pi^n = \{\pi_1^n, \pi_2^n, \pi_3^n\}$) слика 5.4.a) и балансиран граф $\Pi^b = \{\pi_1^b, \pi_2^b, \pi_3^b\}$) (слика 5.4.b). Ради поједностављења сви чворови имају јединичне тежине, као и гране (тежине грана се уопште не разматрају у алгоритму!). С обзиром да је тренутно стање графа небалансирано, покреће се *LMSR* алгоритам за прерасподелу чворова.



а) стање графа пре примене *LMSR* алгоритма

б) балансирано стање графа

Слика 5.4. Пример примене *LMSR* алгоритма

Након дефинисања улаза формира се матрица сличности $Q_{3 \times 3}$ као на слици 5.5. Нпр. елемент $Q_{11} = 0$, због тога што партиције π_1^b и π_1^n немају заједничких чворова; елемент $Q_{12} = 3$, због тога што партиције π_1^b и π_2^n имају три заједничка чвора, итд.

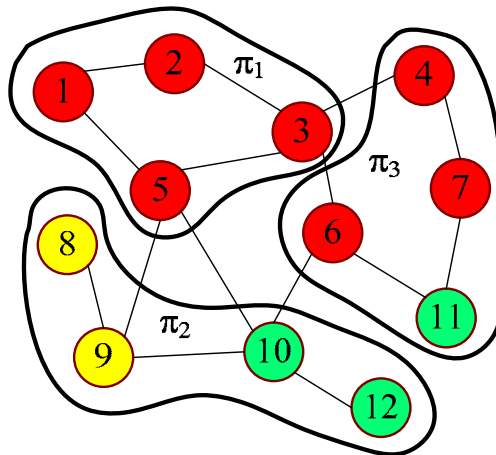
Када се дефинишу сви елементи матрице Q потребно је изабрати елементе матрице (из сваке врсте и колоне само по један), тако да је њихов збир максималан. За наведени пример изабрани елементи би били (подвучени елементи у матрици): Q_{13} , Q_{21} и Q_{32} . Избором ових елемената дефинише се мапирање партиција (као на слици 5.5.):

$$\begin{aligned} Q_{13} &\Rightarrow \pi_1^n \leftrightarrow \pi_3^b \\ Q_{21} &\Rightarrow \pi_2^n \leftrightarrow \pi_1^b \\ Q_{32} &\Rightarrow \pi_3^n \leftrightarrow \pi_2^b \end{aligned}$$

		Балансиране партиције			Пресликавање	
		1	2	3	Постојеће партиције	Балансиране партиције
Постојеће партиције	1	0	3	<u>4</u>	1	→ 3
	2	<u>2</u>	0	0	2	→ 1
	3	2	<u>1</u>	0	3	→ 2

Слика 5.5. Матрица сличности Q и мапирање партиција

На основу овако мапираних партиција одређују се чворови који се пребацују из једне партиције у другу (чворови 4, 6 и 7 се пребацују из партиције π_1 у партицију π_3 , а чворови 10 и 12 из π_3 у π_2). Према томе, новоформирана подела има изглед као на слици 5.6., и она је добијена из небалансираног графа (са слике 5.4.а) тако што је 5 чворова променило партиције.



Слика 5.6. Стање графа након мапирања

5.3.5. PNR алгоритам

PNR алгоритам [92] поред процедуре за одређивање чворова које је потребно пребацити из једне партиције у другу, дефинише комплетан поступак прерасподеле у мултипроцесорском окружењу.

Нека су дефинисани процесори P_1, P_2, \dots, P_p којима су додељени подаци смештени у одговарајућим партицијама $\Pi = \{\pi_1, \pi_2, \dots, \pi_p\}$. Сваки процесор израчунава тежину своје партиције и шаље је процесору координатору P_c који поседује информације о свим чворовима и партицијама. Уколико дође до

небалансираности партиција, координатор итеративно одређује чворове које је потребно пребацити у друге партиције.

На почетку се израчунају вредности критеријумске функције F_{repart} за све могуће преласке сваког граничног чвора. Чвор за који се добије минимум функције F_{repart} се бира за пребацивање у текућој итерацији. У случају да два више чворова има исту вредност критеријумске функције, бира се онај чвор који се не налазу у свом матичном региону. Након пребацивања датог чвора из партиције π_i у партицију π_j поново се прорачунавају тежине измењених партиција, критеријумске функције за све суседе пребаченог чвора, и критеријумске функције за преласке осталих чворова у партиције - π_i и π_j (или из њих).

У следећој фази алгоритма процесор координатор P_c шаље свим процесорима нови распоред чворова у њиховим партицијама. Након добијених инструкција сваки од процесора шаље потребне податке (чворове) са своје партиције у друге партиције.

Специфичан проблем који се јавља приликом имплементације овог алгоритма је проблем изолованости преоптерећених партиција. Овај проблем се јавља у следећим ситуацијама:

Изолованост преоптерећене партиције која се јавља у случају неповезаног графа – за такве партиције не постоје гранични чворови (њени чворови нису повезани са остатком графа). У таквим ситуацијама се разматрају сви чворови изолованих партиција и пребацују они са минималном критеријумском функцијом F_{repart} . Поред тога, поставља се ограничење да се једном пребачени чворови из оваквих партиција не могу вратити у своју матичну партицију (чиме се спречава појава бесконачне петље).

Окруженост преоптерећене партиције само преоптерећеним партицијама. Овај случај се своди на ситуацију да постоји група преоптерећених партиција изолованих од остатка графа. У том случају се такође примењује правило да се пребачени чворови не могу вратити у матичну партицију.

На основу свега наведеног, псеудо код *PNR* алгоритма је следећи:

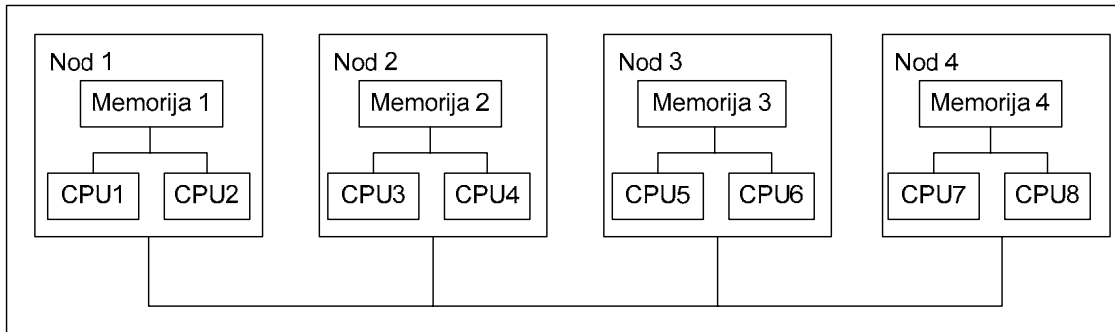
<pre>function PNR_Algorithm(G_p) Input\Output: $G_p=(V,E)$ - partitioned imbalanced\balanced graph while (G_p is imbalanced) or (no balancing progress) do $I \leftarrow$ find isolated overbalanced partitions if (I is empty) for each node v_i in BR do</pre>

```
        calculate  $F_{repart}$ 
    end for each
     $\pi_S, \pi_D, v_i \leftarrow$  get partitions and vertex for  $\min(F_{repart})$ 
else
    for each node  $v_i$  in  $I$  do
        calculate  $F_{repart}$ 
    end for each
     $\pi_S, \pi_D, v_i \leftarrow$  get partitions and non visited vertex for
                                                 $\min(F_{repart})$ 
end if
 $\pi_S \leftarrow \pi_S \setminus \{v_i\}, \pi_D \leftarrow \pi_D \cup \{v_i\}$ 
recalculate  $W_{\pi_S}$  and  $W_{\pi_D}$ 
end while
 $P_c$  sends commands to local processors (donors):  $P_1, P_2, \dots, P_p$  send
vertices to specific processor
```

5.4. Алгоритми прилагођени вишепроцесорским хијерархијским архитектурама

Разматране су *NUMA* (енгл. *Non-Uniform Memory Access*) архитектуре код којих су меморије хијерархијски организоване, брзина приступа меморијама зависи од нивоа меморије и места процесора у наведеној архитектури. Пример *NUMA* архитектуре је приказан на слици 5.7. *NUMA* је мултипроцесорска архитектура организована по нодовима, а сваком ноду припада група процесора и део меморије [99]. Брзина приступа меморији и хијерархија меморијских модула зависи од конкретне *NUMA* архитектуре. Процесори много брже приступају својој локалној меморији (или меморији на њиховом ноду), у односу на брзину приступа меморији на другом ноду. Постоји могућност да се и нодови групишу, па да се на тај начин разликује приступ меморији на ноду из локалној групи од приступа меморији из других група нодова. На тај начин се формира још један хијерархијски ниво меморијског приступа.

На слици 5.7. је приказана архитектура са два ниво приступа меморији. На сваком ноду се налазе по два процесора који могу приступити меморијама на другом ноду, али је тај приступ знатно спорији од приступа локалним меморијама.



Слика 5.7. Пример *NUMA* архитектуре

У циљу развоја алгоритама који је прилагођен *NUMA* архитектури извршено је тестирање брзине приступа меморијама за *NUMA* рачунар са 2 нода и 4 процесорска језгра по ноду (*CPU* AMD Opteron 2.4GHz, 8GB RAM по језгру). За разлику од *NUMA* архитектуре приказане на слици 5.7. у оквиру тестираног система је на сваком *NUMA* ноду процесор имао своју приватну меморију. Према томе, у разматраном тест систему су постојала 3 нивоа меморијског приступа:

- 1. ниво – приступ подацима који се налазе у локалној меморији процесора,
- 2. ниво – приступ подацима који се налазе у меморији процесора из истог нода,
- 3. ниво - приступ подацима који се налазе у меморији процесора из другог нода.

Извршена су мерења брзина израчунавања различитих прорачуна над подацима који се налазе у меморијама из различитих нодова. Добијени резултати указују да је приступ подацима који се налазе у локалним меморијама процесора 2 пута бржи од приступа подацима у меморијама процесора који се налазе на истом *NUMA* ноду, а 4 пута бржи од приступа подацима који се налазе у меморијама процесора из других *NUMA* нодова.

С обзиром на различите брзине приступа меморијским модулима на различитим нивоима, потребно је другачије третирати и податке који се налазе на различитим хијерархијским нивоима меморије. Из тог разлога су развијене варијанте хијерархијских алгоритама, као предлог начина коришћења динамичке прерасподеле података у случају коришћења система базираног на *NUMA* архитектурама.

5.4.1. Хијерархијски *MWF* (*HMWF*) алгоритам за прерасподелу графова

Овај алгоритам представља варијанту *MWF* алгоритма која је прилагођена хијерархијским вишепроцесорским архитектурама. *HMWF* алгоритам је модификован

тако што се израчунавање *inflow* и *outflow* вектора врши у два хијерархијска нивоа (што је прилагођено *NUMA* системима). На првом нивоу *MWF* алгоритам је примењен за балансирање збира тежина партиција на нивоу *NUMA* нодова, док је на другом нивоу извршено балансирање партиција у оквиру појединих *NUMA* нодова.

Псеудо код *HMWF* алгоритам је следећи:

```
function HMWF_Algorithm(Gp)
  numaGraph ← get graph with NUMA partitions
  newNUMAGraph ← MWFAlgorithm(numaGraph)

  for each numa node in newNUMAGraph do
    subGraph ← get graph of the NUMA node
    MWFAlgorithm(subGraph)
  end for
```

Покретање алгоритма је иницирано спољашњим променама стања потенцијалних конекција између чворова из различитих партиција (нодова). Уколико се утврди неизбалансираност партиција, прва фаза алгоритма је балансирање између *NUMA* нодова (балансирање графа са *NUMA* нодовима као чворовима - *numaGraph*). Због тога се уводе два низа *outflowNode* и *inflowNode*. Параметар *outflowNode[i]* представља збир тежина чворова (региона) које *NUMA* нод *i* треба да пошаље другим нодовима, а *inflowNode[i]* представља збир тежина региона које *NUMA* нод *i* треба да прими од других нодова. Прво се израчунавају параметри *outflowNode* и *inflowNode*, а затим се они користе при одлучивању који регион је потребно пребацити из једног *NUMA* нода у други. Када се *NUMA* нодови избалансирају, *MWF* алгоритам се примењује за преуређивање партиција на сваком ноду (*subGraph* – граф партиција на *NUMA* ноду).

5.4.2. Хијерархијски *CP* (*HCP*) алгоритам за репартиционисање графова

Развијени *HCP* алгоритам модификује *CP* алгоритам у процесу пребацивања чвора (региона) у неизбалансиране партиције. Основна идеја овог алгоритма је слична као и код *HMWF* алгоритма – репартиционисање у два нивоа. На првом нивоу се *CP* алгоритам примењује само на регионе које се налазе на различитим *NUMA* нодовима, док се на другом нивоу *CP* алгоритам примењује на регионе који су налазе у одређеном *NUMA* ноду.

Псеудо код *HCP* алгоритам је следећи:

```
function HCP_Algorithm(Gp)
  numaGraph←get graph with NUMA nodes as partitions
  newNUMAGraph←CPAlgorithm(numaGraph)

  for each numa node in newNUMAGraph do
    subGraph←get graph of the NUMA node
    CP_Algorithm(subGraph)
  end for
```

Алгоритам се стартује приликом промене стања одговарајуће потенцијалне везе. На првом нивоу се разматрају графови чије партиције одговарају *NUMA* нодовима, а гране су са тежинама које представљају збир тежина свих грана између одговарајућих нодова (*numaGraph*). Након тога, примењује се *CP* алгоритам за репартиционисање *numaGraph* графа. У другој фази алгоритма се врши независно репартиционисање свих графова који одговарају појединачним *NUMA* нодовима (*subGraph*). У том случају, гране која спаја чворове између различитих *NUMA* нодова се не узимају у обзир (избацивањем тих грана се стварају независни *subGraph* графови). За репартиционисање *subGraph* графова се такође користи *CP* алгоритам.

6. Експериментални резултати

6.1. Опис тест модела података

Тестирање је вршено на моделима података који описују реалне дистрибутивне електроенергетске мреже¹. У табели 6.1. приказане су карактеристике разматраних дистрибутивних мрежа.

Табела 6.1. Тест модели података

Назив графа	Укупан број елемената	Величина иницијалног графа		Величина укрупњеног графа	
		Број чворова	Број грана	V	E
<i>bg54</i>	1126254	295225	299131	54	44
<i>it206</i>	1787939	431102	434486	206	286
<i>pec106</i>	2284322	762411	766755	106	52
<i>bg63</i>	1196078	300503	304637	63	52
<i>bg5x</i>	5980390	1502505	1523180	315	260

Наведени модели представљају податке следећих дистрибутивних мрежа:

- модели *bg54* и *bg63* представљају различите варијанте дистрибутивне мреже града Београда;
- *pec106* је део модела дистрибутивне мреже Северне Каролине (Сједињене Америчке Државе);
- *it206* је модел мреже града Милана (Италија);
- *bg5x* је модел мреже града Београда мултиплициран 5 пута.

Параметар *укупан број елемената* (из табеле 1.) се односи на све типове елемената у моделу. Са аспекта анализе поделе података најзначајнији су елементи типа проводне опреме и трансформатори. Процедура поделе модела података подразумева превођење модела података у иницијални граф (као што је описано у поглављу 3). Величине тако добијених иницијалних графова су дате у табели преко:

- броја грана (колоне 4) – гране представљају елементе типа проводне опреме и трансформаторе (*ConductingEquipment* и *Transformer*), и

¹ Модели података су обезбеђени од фирме *Telvent DMS* из Новог Сада чија је основна делатност развој дистрибутивних менаџмент система.

- броја чворова (колона 3) – чворове представљају чворови конективности (*ConnectivityNode*) који повезују наведену опрему.

Анализом иницијалних графова могуће је утврдити да се ради о слабо повезаним графовима код којих је број грана приближно једнак броју чворова. Међутим, оно што је значајније са аспекта примене алгоритама за поделу графова је укрупњени граф. Код укрупњеног графа чворови (прорачуна) су области прорачуна (тј. корени), а тежине грана представљају број потенцијалних конекција између елемената из различитих корена. Број чворова прорачуна укрупњеног графа је означен са $|V|$, а број грана са $|E|$.

Из табеле 6.1. се може закључити да је код већине укрупњених графова број грана мањи од броја чворова прорачуна (код графова *bg54*, *bg63*, *pec106* и *bg5x*) и они спадају у слабије повезане. Ово је посебно изражено за модел *pec106* код кога је број грана више од 2x мањи од броја чворова прорачуна(52:106). Само у случају модела *it206* је већи број грана од броја чворова прорачуна (286:206). Генерално, дистрибутивна мрежа је најчешће слабо упетљана радијална мрежа, која се карактерише већом упетљаношћу у градским деловима (поготово већим градовима), а у осталим деловима радијалност постаје доминанта карактеристика мреже. На основу анализираних карактеристика укрупњеног графа, може се закључити да је нешто већа упетљаност у оквиру мреже која је представљена графом *it206* у односу на остале тест графове.

6.2. Тестирања иницијалне поделе модела података

Тестирање алгоритама за иницијалну поделу је вршено на моделима података који су описани у табели 6.1.

Тестиране су три групе алгоритама:

1. *PSO* и његове модификације – дистрибуирани (*DPSO*) и еволутивни (*EPSO*) алгоритам,
2. *GA* и његове модификације – дистрибуирани (*DGA*) и хибридни (*HGA*) алгоритам,
3. Алгоритми који се извршавају у више нивоа – вишефазни алгоритам за поделу графова и алгоритам супекорена.

Сви алгоритми су имплементирани у програмском језику *C#* и тестирани са различитим параметрима везаним за проблем поделе и конфигурацију алгоритма. Циљ експеримената је да се за различите моделе података тестирају поделе на

различит број партиција (p) са променама дозвољеног одступања (ε) тежине партиција од просечне тежине партиције.

Имајући у виду реалне потребе у оквиру надзорно-управљачких дистрибутивних система, усвојено је да број партиција може попримити вредности: $p = \{2, 3, 4, 5, 6, 8\}$..

Дозвољена толеранција на одступање од просечне тежине партиције у појединим тестовима постављана на $\varepsilon = \{0.1, 0.2, 0.3\}$. С обзиром да су прелиминарни тестови показали да промена толеранције не утиче значајније на однос добијених резултата за имплементиране алгоритме, накнадно је усвојено да толеранција буде постављена само на 0.1.

Иницијална подела модела података се примењује на већ изграђеном моделу дистрибутивне мреже и може се извршити пре покретања система (*off-line*). Из тог разлога у оквиру анализе алгоритама за иницијалну поделу није анализирано време извршавања алгоритма, осим у случајевима поређења паралелних (дистрибуираних) верзија алгоритама са секвенцијалним (основним) верзијама алгоритама.

6.2.1. PSO алгоритам и његове модификације

PSO алгоритам

Код PSO алгоритма су извршена тестирања са следећим параметрима:

- број јединки (честица) – зависи од величине графа – идеја је да минимални број честица одговара броју чворова графа (тј. није мањи од броја чворова). За примену алгоритма на наведене моделе је коришћен следећи број честица:
- *bg54* и *bg63* - 100 честица,
- *pec106* – 110 честица,
- *it106* – 220 честица и
- *bg5x* – 330 честица.
- фактор инерције W је постављен на 0.9 и линеарно се смањује до 0.4 за унапред задати број итерација (препоруча из [100]),
- брзина честице v је ограничена укупним бројем партиција $v \leq |V_{\max}| \leq p$.
- $C_1 = C_2 = 2$.
- толеранција $\varepsilon \in \{0.1, 0.2, 0.3\}$
- максималан број итерација извршавања алгоритма - $maxIt = 5000$
- број понављања сваког теста – 100 пута.

У табели 6.2. су приказани резултати који представљају средње вредности за 100 мерења.

Табела 6.2. Резултати добијени PSO алгоритмом

<i>p</i>	<i>Model</i>	0.1	0.2	0.3
2	<i>bg54</i>	392	395	397
	<i>bg63</i>	475	473	484
	<i>pec106</i>	149	155	158
	<i>it206</i>	853	841	849
	<i>bg5x</i>	2379	2334	2342
3	<i>bg54</i>	385	381	383
	<i>bg63</i>	455	462	460
	<i>pec106</i>	146	149	152
	<i>it206</i>	746	760	712
	<i>bg5x</i>	2190	2191	2119
4	<i>bg54</i>	363	370	366
	<i>bg63</i>	444	449	446
	<i>pec106</i>	138	140	142
	<i>it206</i>	674	737	642
	<i>bg5x</i>	2062	2104	2069
5	<i>bg54</i>	336	359	358
	<i>bg63</i>	419	442	428
	<i>pec106</i>	129	135	136
	<i>it206</i>	603	652	633
	<i>bg5x</i>	1949	2077	1980
6	<i>bg54</i>	310	353	358
	<i>bg63</i>	360	421	428
	<i>pec106</i>	120	128	135
	<i>it206</i>	530	588	633
	<i>bg5x</i>	1933	1942	1980
8	<i>bg54</i>	255	270	298
	<i>bg63</i>	237	337	371
	<i>pec106</i>	119	128	130
	<i>it206</i>	421	504	435
	<i>bg5x</i>	1908	1912	1960

Дистрибуирани PSO алгоритам

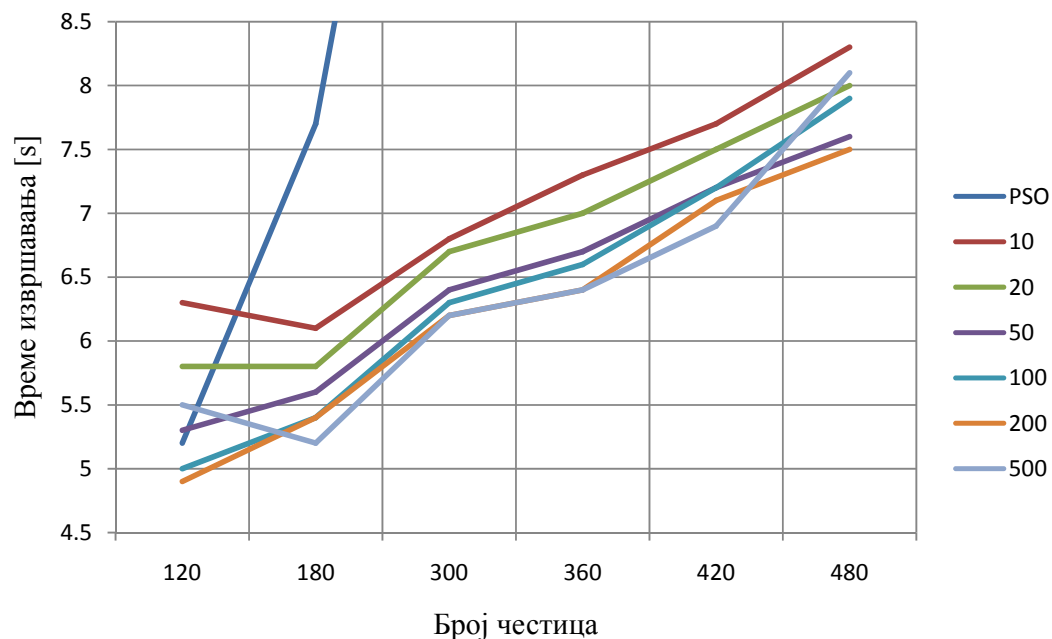
Дистрибуирани PSO алгоритам је имплементиран у C# програмском језику, а кластеризација ројева је изведена под Windows HPC Server 2008 оперативним

системом, при чему је за комуникацију између ројева коришћен *MPI* (енгл. *Message Passing Interface*) комуникациони протокол [101], [102].

Извршено је поређење секвенцијалног и дистрибуираног *PSO* алгоритма над моделом *bg63* са следећим подешавањима: $W = [0.9, 0.4]$, $C_1 = 2$, $C_2 = 2$, димензија честице је 63, $maxIt=1000$, $\epsilon=0.1$, број партиција $p=6$. Тестирање дистрибуираног алгоритма је вршено са 6 подројева који се налазе на 6 процесора. Анализирано је време извршавања алгоритма T_a и вредност критеријумске функције F за тестове у којима је број честица дистрибуираног и секвенцијалног алгоритма исти (с тим што су код дистрибуираног алгоритма ројеви подељени на 6 подројева). У табели 6.3. су приказани резултати тестова за секвенцијални и дистрибуирани алгоритам.

Резултати приказани у табели 6.3. представљају средње вредности добијене за 100 понављања тестова. Тестови су вршени за различите величине популације (120 – 480 честица) и за различите периоде синхронизације за дистрибуирани алгоритам $T_{sync} \in \{10, 20, 50, 100, 200, 500\}$.

Зависност времена извршавања секвенцијалног и дистрибуираног алгоритма (за различите периоде синхронизације) је приказана на слици 6.1.



Слика 6.1. Зависност време извршавања секвенцијалног и дистрибуираног (за различите T_{sync}) *PSO* алгоритма

На основу приказаних резултата могу се извести следећи закључци:

1. са порастом периода синхронизације (T_{sync}) смањује се време извршавања дистрибуираног алгоритма,

2. време извршавања дистрибуираног алгоритма је знатно мање (2x и више) тек када у популацији има 300 и више јединки,
3. када је популација мања неопходна је чешћа комуникација (мање T_{sync}) између подреја да би се достигли исти резултати као код секвенцијалног алгоритма,
4. уколико број честица у подреју није мањи од величине честице, дистрибуирани алгоритам даје боље резултате за критеријумску функцију и знатно је краће време његовог извршавања,
5. време извршавања дистрибуираног алгоритма са популацијом 6x80 честица је упоредиво са временом извршавања секвенцијалног алгоритма над популацијом од 180 честица, а добијају се доста бољи резултати (372/361),
6. оправданост *DPSO* алгоритма се посебно показује у случају потребе за већом популацијом. Уколико би величине јединке биле веће од 300 (тј. ако би се делили графови већи од 300 чворова), тада би коришћење паралелног дистрибуираног *PSO* алгоритма било потпуно оправдано и у значајној мери би утицало на квалитет решења.

Табела 6.3. Упоредни резултати секвенцијалног (*PSO*) и дистрибуираног *PSO* (*DPSO*) алгоритма за 1000 итерација

Величина популације		Параметри	<i>PSO</i>	<i>DPSO</i>					
				Период синхронизације (T_{sync})					
<i>PSO</i>	<i>DPSO</i>			10	20	50	100	200	500
120	6x20	<i>F</i>	358	357	355	353	358	358	355
		T_a [s]	5.2	6.3	5.8	5.3	5	4.9	5.5
180	6x30	<i>F</i>	361	361	358	362	360	357	356
		T_a [s]	7.7	6.1	5.8	5.6	5.4	5.4	5.2
300	6x50	<i>F</i>	366	365	366	363.5	365	366	363
		T_a [s]	13.3	6.8	6.7	6.4	6.3	6.2	6.2
360	6x60	<i>F</i>	368	368	364	367	366	365	367
		T_a [s]	15.1	7.3	7	6.7	6.6	6.4	6.4
420	6x70	<i>F</i>	370	371	367	365	369.5	366	366
		T_a [s]	17.5	7.7	7.5	7.2	7.2	7.1	6.9
480	6x80	<i>F</i>	371	370.5	369	371	366	372	371
		T_a [s]	20.9	8.3	8	7.6	7.9	7.5	8.1

Еволутивни PSO алгоритам

Тестирања развијеног *EPSO* алгоритма су извршена са следећим параметрима:

Број партиција p	2, 3, 4, 5 и 6
Толеранција	0.1 (10%), 0.2, 0.3
Број итерација	5000
Степен мутације τ	0.2
Степен мутације τ' за b_g^*	0.1
Степен репликације r	3
Фактор комуникације P	1
Број честица	100

У табели 6.4. су приказани резултати добијени применом *EPSO* алгоритма. Приказане су средње вредности критеријумске функције након 40 понављања сваког теста.

Поређењем резултата добијених помоћу ова два алгоритма може се закључити да еволутивни *PSO* алгоритам даје значајно боље резултате у свим случајевима поделе. Његова доминација је најупечатљивија при поделама већих графова, као и при поделама на већи број партиција. Тако нпр. приликом поделе на 3-6 партиција на моделима *bg5x* и *it206* применом *EPSO* алгоритма добијају се боље вредности 20-50% у односу на обичан *PSO* алгоритам. Код мањих модела - *bg54*, *bg63* и *rec106* ова разлика иде до 10% у корист *EPSO* алгоритма.

6.2.2. Генетски алгоритам и његове модификације

Генетски алгоритам

Прелиминарна тестирања генетског алгоритма су имала за циљ да упореде резултате добијеним применом различитих типова селекције, укрштања и степен елитизма.

У радовима [30], [48], [57], [103] је презентован део добијених резултата. Тестирања су показала да пропорционална и случајна селекција дају подједнако добре резултате и да су нешто бољи од турнирске селекције. С обзиром да се пропорционална селекција показала нешто бољом – она је коришћена у свим тестовима. Поред тога, најбољи резултате је постигао генетски алгоритам са вероватноћом мутације 50%.

Табела 6.4. Упоредни резултати основног PSO и EPSO алгоритма

	Алгоритам	PSO	EPSO	PSO	EPSO	PSO	EPSO
p	Model	$\varepsilon = 0.1$		$\varepsilon = 0.2$		$\varepsilon = 0.3$	
2	<i>bg54</i>	392	398	395	399	397	400
	<i>bg63</i>	475	482	473	484	484	484
	<i>pec106</i>	149	161	153	161	157	162
	<i>it206</i>	853	914	841	935	849	936
	<i>bg5x</i>	2379	2442	2334	2447	2342	2449
3	<i>bg54</i>	385	385	381	389	383	391
	<i>bg63</i>	455	471	462	475	460	479
	<i>pec106</i>	146	155	148	155	150	155
	<i>it206</i>	746	894	760	895	712	899
	<i>bg5x</i>	2190	2389	2191	2392	2119	2395
4	<i>bg54</i>	363	373	370	379	366	385
	<i>bg63</i>	444	456	449	466	446	468
	<i>pec106</i>	138	148	139	152	141	153
	<i>it206</i>	674	852	737	871	642	880
	<i>bg5x</i>	2062	2351	2104	2356	2069	2360
5	<i>bg54</i>	336	354	359	375	358	379
	<i>bg63</i>	419	447	442	456	428	456
	<i>pec106</i>	129	142	131	149	138	151
	<i>it206</i>	603	829	652	839	633	853
	<i>bg5x</i>	1949	2336	2077	2345	1980	2348
6	<i>bg54</i>	310	352	353	360	358	362
	<i>bg63</i>	360	422	421	447	428	450
	<i>pec106</i>	120	132	124	145	129	145
	<i>it206</i>	530	820	588	826	633	830
	<i>bg5x</i>	1933	2275	1942	2282	1980	2285

Тестирање степена елитизма. За процену степена елитизма коришћен је тест над моделом *bg54* и следеће параметри генетског алгоритма:

- пропорционална селекција,
- укрштање у једној тачки (*T1*) са вероватноћом 50%,
- вероватноћа мутације -50%,
- степен swap мутације -50%.
- *maxIt* = 1000 итерација
- број елитних јединки: 1, 3 и 10.

Резултати тестирања поделе мреже са толеранцијом $\varepsilon=0.1$ су приказани у табели 6.5.

Табела 6.5. Поређење генетских алгоритама са различитим бројем елитних јединки

p	Број елитних јединки		
	1	3	10
2	396	396	396
3	376	375	375
4	361	360	361
5	345	340	344
6	332	332	329

На основу добијених резултата може се утврдити да је најпогодније преносити 1 елитну јединку, те ће у будућим тестовима увек бити преношена у следећу генерацију једна елитна јединка.

Тестирање типа укрштања. С обзиром на различите варијанте свих фаза генетског алгоритама за поделу графа, потребно је проценити који од облика укрштања би дао најбоље резултате. Због тога су извршени тестови за укрштање у једној тачки (*TI*), укрштање са очувањем најбоље групе (*OHI*), и укрштање у једној тачки са нормализацијом јединки (*TIH*). Тестови су извршени на моделу *bg54* за следеће параметре алгорита:

- пропорционална селекција,
- укрштање са вероватноћом 50%,
- вероватноћа мутације -50%,
- вероватноћа *swar* мутације -50%
- број елитних јединки -1.

Добијени резултати (средње вредности) за 100 понављања за поделе мреже са толеранцијом $\epsilon=0.1$ су приказани у табели 6.6.

На основу резултата приказаних у табели 6.6, може се закључити да укрштање са нормализацијом јединке (*TIH*) даје најбоље вредности критеријумске функције у свим тестовима, па ће оно бити коришћено за тестирање генетског алгорита.

Табела 6.6. Упоредни резултати *GA* са различитим типовима укрштања

<i>p</i>	Тип укрштања		
	<i>T1</i>	<i>ОНГ</i>	<i>T1H</i>
2	396	395	397
3	376	377	377
4	361	361	364
5	345	340	345
6	332	332	332

Тестирање генетског алгоритма. Након одређивања утврђивања погодне варијанте за је вршено са следећим параметрима:

- пропорционална селекција,
- укрштање са вероватноћом 50%,
- вероватноћа мутације -50%,
- вероватноћа *swar* мутације -50%
- број елитних јединки -1.

Добијени резултати показују да основни генетски алгоритам постиже знатно боље резултате од основног *PSO* алгоритма (табела 6.2).

Паралелни и дистрибуирани GA алгоритам

Тестирање паралелног и дистрибуираних генетског алгоритма је вршено на оперативном систему *Windows HPC Server 2008*.

Идеја паралелног генетског алгоритма је да истовремено постоји више популација које ће независно напредовати, а које би након одређеног броја итерација размењивале своје најбоље јединке. Паралелизам се ослања на *MPI* механизам за размену података између процеса [102]. Такође, паралелни алгоритам је могуће покренути и конкурентно на једном рачунару. Процеси се разликују по свом јединственом редном броју који се назива статус (*rank*). Процес са статусом нула је главни и задужен је за прикупљање и расподелу најбољих јединки. Он прима најбоље јединке од свих процеса, убацује их у низ и сортира. Потом најбољу јединку из низа шаље свим процесима и они је уврштавају у своју популацију. Број јединки који се размењује и фреквенција размене између популација могу да варирају. Још

једна предност паралелног генетског алгоритма је могућност да се сваки процес покрене са различитом комбинацијом параметара. Тиме је омогућено да различите популације на различит начин еволуирају.

Табела 6.7. *Резултати GA алгоритма*

<i>p</i>	<i>Model</i>	ϵ		
		0.1	0.2	0.3
2	<i>bg54</i>	397	398	399
	<i>bg63</i>	479	482	484
	<i>pec106</i>	157	159	160
	<i>it206</i>	936	921	917
	<i>bg5x</i>	2393	2403	2404
3	<i>bg54</i>	378	387	390
	<i>bg63</i>	466	476	476
	<i>pec106</i>	152	153	154
	<i>it206</i>	889	863	875
	<i>bg5x</i>	2327	2311	2305
4	<i>bg54</i>	361	371	382
	<i>bg63</i>	461	459	466
	<i>pec106</i>	149	149	149
	<i>it206</i>	859	829	858
	<i>bg5x</i>	2270	2311	2291
5	<i>bg54</i>	353	371	377
	<i>bg63</i>	437	459	460
	<i>pec106</i>	135	139	143
	<i>it206</i>	845	799	853
	<i>bg5x</i>	2262	2276	2259
6	<i>bg54</i>	335	354	360
	<i>bg63</i>	444	448	461
	<i>pec106</i>	135	138	142
	<i>it206</i>	850	764	843
	<i>bg5x</i>	2246	2229	2264
8	<i>bg54</i>	285	310	310
	<i>bg63</i>	363	379	436
	<i>pec106</i>	132	135	140
	<i>it206</i>	480	816	554
	<i>bg5x</i>	2225	2212	2212

Паралелни генетски алгоритам (*PGA*) је тестиран покретањем 6 програмских нити паралелно на једном рачунару, док је дистрибуирани генетски алгоритам

(*DGA*) тестиран у мрежи рачунара, при чему је покретано 6 процеса на 6 процесора. Да би се упоредили резултати секвенцијалног (*GA*), паралелног (*PGA*) и дистрибуираног (*DGA*) алгоритма коришћени су параметри из табеле 6.8. Идеја да дистрибуирани генетски алгоритми (*DGA*) раде са различитим параметрима подстакла је тест дистрибуираног алгоритма (*DGARP*) код кога се од 6 дистрибуираних генетских алгоритама – прва 3 алгоритама (на 3 процесора) извршавају са истим параметрима, а преостала 3 алгоритама се извршавају са специфичним параметрима као што је наведено у табели 6.8.

Табела 6.8. Параметри са којима се извршавају *GA*, *PGA*, *DGA* и *DGARP* алгоритми

Алгоритам	<i>GA</i> , <i>PGA</i> , <i>DGA</i> , <i>DGARP</i> (1., 2., 3.)	<i>DGARP</i> 4.	<i>DGARP</i> 5.	<i>DGARP</i> 6.
Селекција	пропорционална	проп.	проп.	проп.
Вероватноћа укрштања	50%	80%	20%	20%
Вероватноћа мутације	50%	20%	80%	20%
Вероватноћа <i>swap</i> мутације	50%	20%	20%	80%
Број елитних јединки	1	1	1	1

Поред тога, код паралелног и дистрибуираног алгоритма се на сваких 10 итерација размењују најбоље јединке ($T_{sync} = 10$) и при томе се размењује по једна елитна јединка ($\mu_s = 1$). У табели 6.9. су приказани резултати примењених алгоритама за толеранцију на балансираност $\varepsilon=0.1$ и $maxIt = 1000$ итерација, а величина популације је 100 јединки код *GA*, а 6x100 јединки код *PGA*, *DGA* и *DGARP*.

Табела 6.9. Поређење *GA*, *PGA*, *DGA* и *DGARP* алгоритама

<i>p</i>	Алгоритам	<i>GA</i>	<i>PGA</i>	<i>DGA</i>	<i>DGARP</i>
2	<i>F</i>	396	397	397	398
	T_a [s]	15.54	69.31	21.14	20.16
3	<i>F</i>	375	379	381	380
	T_a [s]	13.21	62.48	19.22	18.29
4	<i>F</i>	360	367	367	368
	T_a [s]	11.97	58.44	18.16	17.2
5	<i>F</i>	343	350	349	352
	T_a [s]	11.54	57.01	17.81	16.87
6	<i>F</i>	324	336	333	342
	T_a [s]	11.18	56.07	17.68	16.64

На основу добијених резултата може се закључити да дистрибуирани генетски алгоритам са различитим параметрима за појединачне *GA* алгоритме (*DGARP*) даје најбоље резултате.

Тестирање периода синхронизације. С обзиром на могућност промене периода синхронизације, извршени су и тестови *DGA* алгоритма за различите $T_{sync}=\{10, 50, 100\}$ и 1000 итерација, и добијени резултати су приказани у табели 6.10.

Табела 6.10. *Дистрибуирани GA са различитим периодима синхронизације*

p	T_{sync}	10	50	100
2	F	397	397	397
	T_a [s]	21.14	20.68	20.52
3	F	381	382	382
	T_a [s]	19.22	18.72	18.72
4	F	367	369	370
	T_a [s]	18.16	17.68	17.60
5	F	349	351	353
	T_a [s]	17.81	17.25	17.31
6	F	333	335	341
	T_a [s]	17.68	17.07	17.13

Анализом резултата из табеле 6.10. може се закључити да се најбољи резултати добијају када је период синхронизације постављен на 100 итерација (када је социјални фактор слабије изражен). Поред тога, период синхронизације не утиче значајније на време извршавања алгоритма.

Хибридни генетски алгоритам

У прелиминарним тестовима задатак је био проценити комбинацију параметара за коју се добијају најбољи резултати, и то: степен мутације и степен укрштања. Утврђено је да се најбољи резултати добијају када 50% јединки учествује сваких 100 итерација у *FM* мутацији. Укупан број јединки у генерацији зависи од величине графа:

1. *bg54* и *bg63* – 100 јединки,
2. *it206* – 220 јединки и
3. *bg5x* – 330 јединки.

Сви тестови су урађени за толеранцију на балансираност $\varepsilon=0.1$, вероватноћа мутације је попримала вредности из скупа $\{0.1, 0.3, 0.5\}$, а вероватноћа укрштања из

скупа {30%, 50%}. Након 5000 генерација, примењује се *FM* мутација над 10 јединки. У табели 6.11. су приказани добијени резултати.

Табела 6.11. *Резултати HGA алгоритма са променљивим вероватноћама мутације и укрштања*

Модел	мутација	0.1		0.3		0.5	
	укрштање	30	50	30	50	30	50
<i>bg54</i>	2	398	398	398	398	398	398
	4	380	380	380	380	380	380
	6	359	360	359	359	357	357
<i>bg63</i>	2	483	483	483	483	483	483
	4	467	467	467	467	466	466
	6	441	437	440	436	434	435
<i>it206</i>	2	940	940	939	939	939	939
	4	878	881	875	871	877	872
<i>bg5x</i>	2	2451	2449	2450	2443	2449	2446
	4	2384	2374	2374	2372	2374	2373

На основу добијених резултата се може закључити да се најбољи резултати добијају када 10% јединки учествује у мутацији, и ако 30% јединки учествује у укрштању. Резултати тестирања *HGA* алгоритма са наведеним оптималним параметрима за мутацију и укрштање су приказани у табели 6.13. на основу којих се може закључити да *HGA* постиже знатно боље резултате од основног *GA* алгоритма.

6.3.2. Вишефазни алгоритам за поделу графова

Прелиминарни тестови вишефазног алгоритма за поделу графова имали су за циљ процену која варијанта алгоритма је погодна за примену над моделима дистрибутивних електроенергетских мрежа. У ту сврху тестиране су све варијанте предложених алгоритама у свим фазама вишефазног алгоритма, и то:

1. фаза – укрупњавање - у овој фази коришћени су *RM*, *HEM*, *LEM* и *HCM* алгоритми
2. фаза – иницијална подела – примењени су *KL*, *GGP* и *GGGP* алгоритми
3. фаза – финија прерасподела – *FM* алгоритам

Извршени су тестови над графовима *bg54*, *bg63*, *pec106*, *it206* и *bg5x*, и поделе на 2, 4, 6 и 8 партиција са толеранцијом $\epsilon=0.1$.

Анализом добијених резултата (који су приказани у табели 6.12.) се може утврдити да су најбоље вредности за критеријумску функцију добијене при иницијалној подели *GGGP* алгоритмом и *HEM* алгоритмом коришћеним за укрупњавање (нешто бољи резултати од *HCM* алгоритма). Доминација *HEM* алгоритма посебно долази до изражаја код графа *it206*, а с обзиром да је то граф са нешто вишим степеном повезаности између чворова прорачуна (корена), процена је да ће ова варијанта алгоритма посебно доћи до изражаја ако се дели мрежа која се одликује већим степеном повезаности између корена [104].

Табела 6.12. Тестирање различитих варијанти вишефазног алгоритма

Алг.		<i>KL</i>				<i>GGP</i>				<i>GGGP</i>			
Модел	<i>p</i>	<i>RM</i>	<i>HEM</i>	<i>LEM</i>	<i>HCM</i>	<i>RM</i>	<i>HEM</i>	<i>LEM</i>	<i>HCM</i>	<i>RM</i>	<i>HEM</i>	<i>LEM</i>	<i>HCM</i>
<i>bg54</i>	2	389	378	379	392	395	396	393	396	391	395	392	396
	4	352	335	309	348	352	357	343	369	376	383	364	383
	6	297	297	265	328	322	322	301	347	350	365	350	371
	8	256	296	248	291	298	303	273	332	335	364	334	365
<i>bg63</i>	2	481	481	475	481	480	479	475	482	478	480	476	481
	4	438	438	418	459	445	431	406	460	454	462	449	466
	6	395	405	371	416	407	386	360	439	429	442	411	457
	8	367	364	333	383	381	375	326	413	418	438	407	451
<i>pec106</i>	2	161	160	160	158	160	159	159	159	161	161	161	163
	4	153	153	153	153	141	141	138	138	153	155	152	155
	6	141	142	141	141	129	121	123	120	145	148	146	149
	8	122	130	125	129	120	116	114	117	140	144	141	142
<i>it206</i>	2	918	921	894	907	919	928	917	921	929	933	931	935
	4	850	869	828	844	826	870	805	839	884	898	874	904
	6	801	840	783	816	772	827	740	794	830	869	823	870
	8	748	805	732	797	749	805	711	776	790	839	761	851
<i>bg5x</i>	2	2463	2462	2460	2460	2430	2440	2419	2456	2461	2464	2452	2464
	4	2429	2437	2400	2444	2307	2363	2291	2412	2447	2454	2438	2453
	6	2439	2426	2432	2457	2209	2237	2158	2363	2401	2441	2418	2443
	8	2240	2349	2209	2369	2152	2183	2110	2344	2307	2395	2294	2387

С обзиром да је наведени вишефазни алгоритам формиран по угледу на *METIS* алгоритам, експерименти на свим моделима су урађени *METIS* алгоритмом и добијени резултати су приказани у табели 6.13.

6.3.3. Алгоритам суперкорена

Новоразвијени алгоритам суперкорена је тестиран под истим условима као и вишефазни (*METIS*) алгоритам. Као што је раније објашњено, алгоритам се састоји из 3 фазе:

1. укрупњавања чворова прорачуна у суперкорене (сједињавања свих повезаних чворова),
2. поделе превеликих суперкорена, и
3. распоређивање бројева које одговарају тежинама изведених суперкорена.

Тестирање је извршено над свим моделима из табеле 6.1., а у фази поделе превеликих суперкорена је коришћен *METIS* алгоритам за добијање фрагмената превеликих суперкорена. У 3. фази је коришћен *Greedy* алгоритам за прерасподелу бројева. Резултати изведених експеримената са алгоритмом суперкорена су представљени у табели 6.13. На основу анализе добијених резултата може се утврдити веома добро понашање алгоритма суперкорена у свим веријантама подела модела. У великом броју случајева алгоритам суперкорена даје боље резултате од свих осталих алгоритама.

Детаљнији упоредни преглед резултата свих алгоритама за иницијалну поделу биће обухваћен дискусијом резултата по појединим тест моделима података.

6.3.4. Дискусија резултата алгоритама за иницијалну поделу модела података

У циљу свеобухватнијег поређења резултата имплементираних алгоритама, графички ће бити представљени упоредни резултати свих алгоритама разврстани по моделу података тестиране мреже.

Подела модела података дистрибутивне мреже bg54

На слици 6.2. су приказане су нормализоване вредности критеријумске функције F (у односу на најбоље остварени резултат за одговарајућу поделу добијене критеријумске функције) за модел мреже *bg54*.

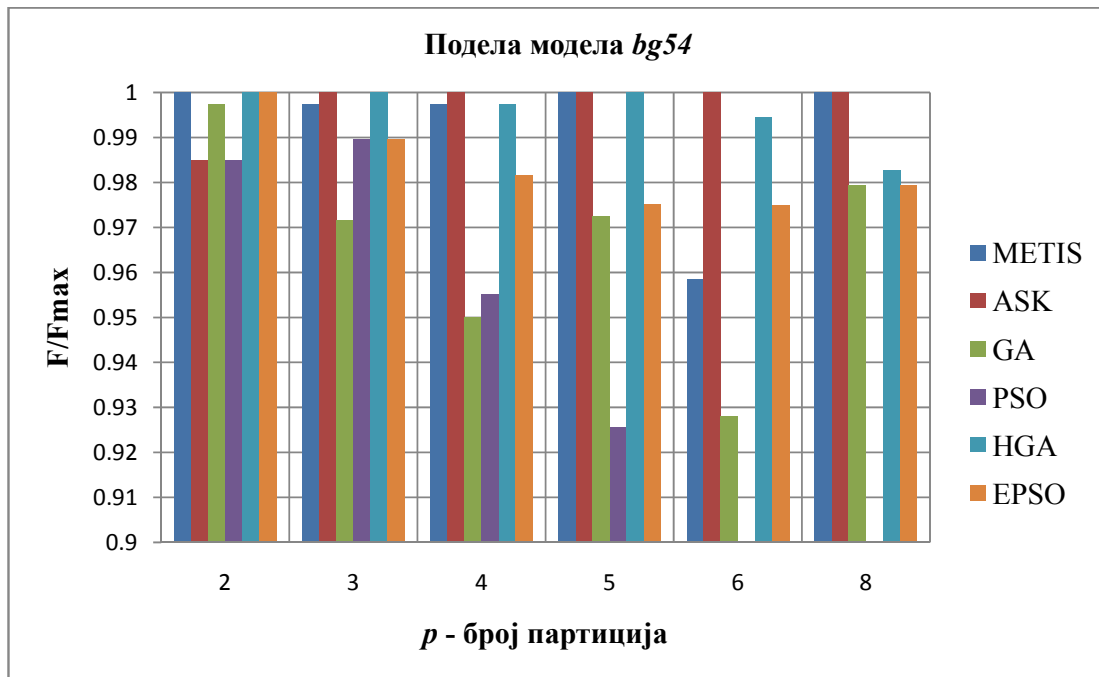
Резултати тестирања на моделу *bg54* показују да је најбоље резултате остварио *SR* алгоритам у свим случајевима поделе (осим за поделу на две партиције). *METIS* алгоритам се показао бољим од *SR* алгоритма само у случају поделе на две партиције, док је у случају поделе на 6 партиција постигао лошије резултате од *SR* алгоритма.

Табела 6.13. Резултати *SR*, *HGA* и вишефазног алгоритма ($\epsilon=0.1$)

p	Модел	<i>SR</i>	<i>METIS</i>	<i>HGA</i>
2	<i>bg54</i>	392	398	398
	<i>bg63</i>	483	483	483
	<i>pec106</i>	163	163	163
	<i>it206</i>	939	937	940
	<i>bg5x</i>	2465	2465	2447
3	<i>bg54</i>	389	388	389
	<i>bg63</i>	475	474	474
	<i>pec106</i>	161	161	161
	<i>it206</i>	929	907	912
	<i>bg5x</i>	2465	2465	2411
4	<i>bg54</i>	380	379	379
	<i>bg63</i>	469	465	468
	<i>pec106</i>	156	156	153
	<i>it206</i>	869	883	881
	<i>bg5x</i>	2400	2452	2373
5	<i>bg54</i>	363	363	363
	<i>bg63</i>	464	461	461
	<i>pec106</i>	152	152	148
	<i>it206</i>	884	888	841
	<i>bg5x</i>	2465	2465	2351
6	<i>bg54</i>	361	346	359
	<i>bg63</i>	449	444	431
	<i>pec106</i>	152	152	150
	<i>it206</i>	842	870	821
	<i>bg5x</i>	2465	2463	2232
8	<i>bg54</i>	291	291	265
	<i>bg63</i>	372	370	369
	<i>pec106</i>	148	148	145
	<i>it206</i>	802	800	757
	<i>bg5x</i>	2424	2385	2280

Поред тога, веома добре резултате је остварио *HGA* алгоритам: F_{max} за поделу на 2, 3 и 5 партиција – врло блиске оптималној вредности (разлика је мања од 1%), док је за

поделу на 8 партиција остварио слабији резултат за око 2%. *EPSO* алгоритам је постигао стабилне резултате са одступањем мањим од 3%, а у случају поделе на 2 партиције је достигао F_{max} . *GA* и *PSO* постижу знатно лошије резултате, при чему је *GA* за све поделе бољи од *PSO* алгоритма.

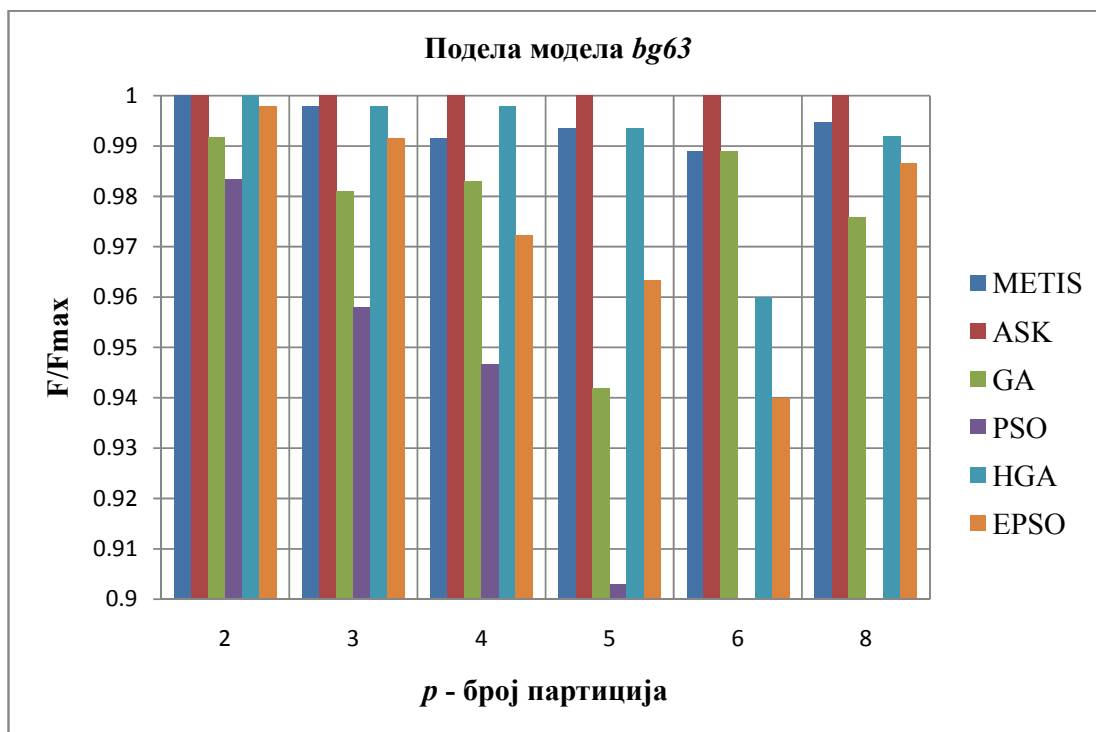


Слика 6.2. Резултати алгоритама за различите поделе модела *bg54*

Подела модела података дистрибутивне мреже *bg63*

Упоредни резултати тестирања алгоритама за иницијалну поделу на моделу *bg63* су приказани на слици 6.3.

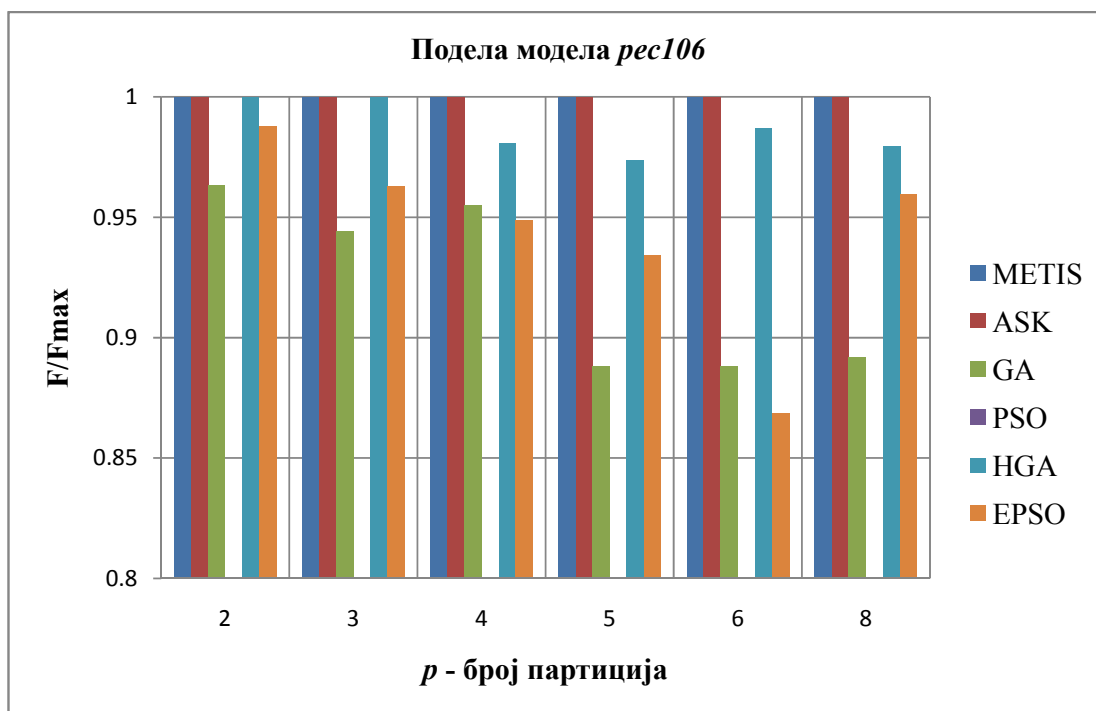
На основу приказаних резултата може се закључити да *SR* алгоритам постиже најбоље резултате за све поделе. *METIS* алгоритам је достигао најбољу вредност критеријумске функције - F_{max} за поделу на две партиције, док је за остале поделе одступање његових резултата од оптималне вредности до 1%. Поред тога, по резултатима се још истиче *HGA* који је за поделу на две партиције остварио оптималну вредност критеријумске функције, за поделу на 3, 4, 5 и 8 партиција је достигао вредности функције F са одступањем до 1% од F_{max} , а нешто лошији резултат са одступањем 4% је остварио за поделу на 6 партиција. Резултати добијени алгоритмима *EPSO* и *GA* су у границама до 3%, сем за случај поделе на 5 партиција (*GA*) и 6 партиција (*EPSO*) када је одступање 6% од оптималне вредности критеријумске функције. *PSO* алгоритмом се добијају најлошији резултати.



Слика 6.3. Резултати алгоритама за различите поделе модела *bg63*

*Подела модела података дистрибутивне мреже *res106**

Упоредни резултати тестирања алгоритама за иницијалну поделу на моделу *res106* су приказани на слици 6.4.



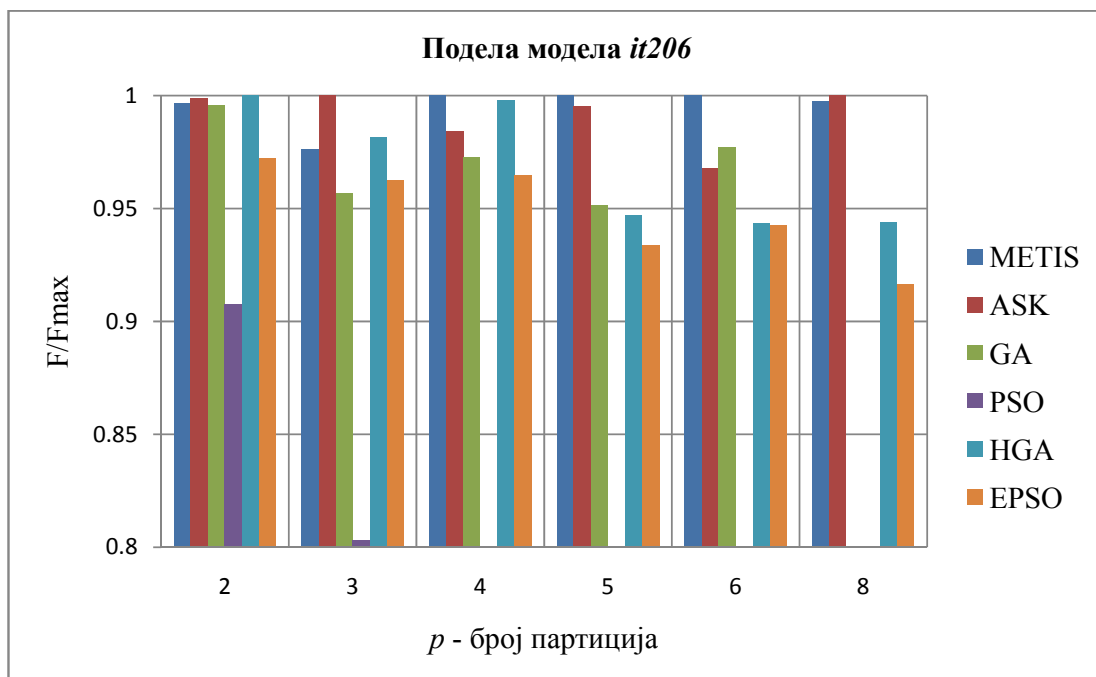
Слика 6.4. Резултати алгоритама за различите поделе модела *res106*

Алгоритми *SR* и *METIS* остварују најбоље резултате за све врсте подела мреже *res106*. *HGA* алгоритам достиже оптималне вредности критеријумске функције за поделе на 2 и 3 партиције, а у осталим случајевима његови резултати су у границама до 3% од F_{max} . *EPSO* алгоритам добија нешто боље резултате од *GA* и они су у са одступањем 1-7% сем у случају поделе на 6 партиција где је одступање 13% од F_{max} . *PSO* алгоритам остварује најлошије резултате.

Подела модела података дистрибутивне мреже *it206*

Упоредни резултати тестирања алгоритама за иницијалну поделу на моделу *it206* су приказани на слици 6.5.

На основу приказаних резултата се може закључити да *METIS* алгоритам остварује најбоље резултате у случајевима поделе на 4, 5 и 6 партиција док је у осталим случајевима одступање његове функције F од F_{max} у границама 1-3%. *SR* алгоритам остварује најбоље резултате за поделе на 3 и 8 партиција док су у осталим случајевима одступања његових резултата у границама од 0.5-3% од оствареног оптимума. Анализом резултата поделе може се закључити *METIS* и *SR* показују подједнако добре резултате за разматране тестове.



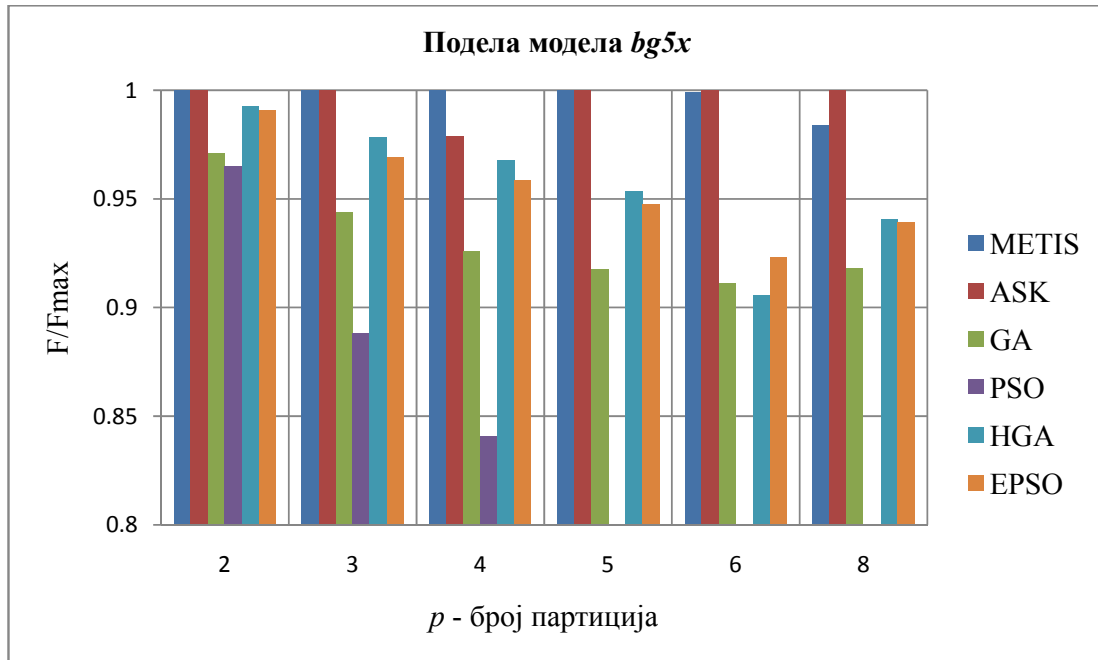
Слика 6.5. Резултати алгоритама за различите поделе модела *it206*

Поред тога, *HGA* постиже најбоље резултате за поделу на 2 партиције, у случају поделе на 3 и 4 партиције његови резултати одступају 2% и 0.5% од

оствареног оптимума, а за поделе на 5, 6 и 8 партиција одступање је око 6%. *GA* (1-5% одступање од F_{max}) се показао нешто бољим од *EPSO* алгоритма (3-8%), осим у случају поделе на 8 партиција. *PSO* остварује најлошије резултате.

Подела модела података дистрибутивне мреже *bg5x*

Упоредни резултати тестирања алгоритама за иницијалну поделу на моделу *bg5x* су приказани на слици 6.6.



Слика 6.6. Резултати алгоритама за различите поделе модела *bg5x*

Резултати показују да су најбоље резултате остварили *METIS* и *SR* алгоритам. Нешто лошије резултате *SR* алгоритам је постигао за поделу на 3 партиције, док је *METIS* био лошији за поделе на 6 и 8 партиција. Остали алгоритми су постигли следеће резултате: *HGA* алгоритам (2-9% одступања), *EPSO* (2-8% одступања), *GA* (3-9% одступања), а знатно лошије резултате је остварио *PSO* алгоритам.

6.3. Тестирања динамичке прерасподеле модела података

Динамичка прерасподела модела података је тестирана у више етапа:

- упоредно тестирање *WF* и *LMSR* алгорита на моделима *bg54*, *bg63* и *it206*,
- тестирање и поређење *MWF* и *PNR* алгорита на моделу *bg63*,
- поређење *WF* алгорита и његове модификације *MWF* алгорита,
- упоредно тестирање хијерархијских алгоритама *HMWF* и *HCP* са основним (неадаптираним) верзијама наведених алгоритама (*MWF* и *CP*).

Поред анализе резултата добијених од стране различитих алгоритама, извршени су и експерименти у оквиру дистрибуираних мултипроцесорских система. У овој групи експеримената су подаци распоређивани по меморијама различитих процесора. При томе су мерени време трајања алгорита и време потребно за пребацивање података са једног процесора (рачунара) на други у циљу балансираности система.

6.3.1. Тестирање *WF* и *LMSR* алгорита

Алгоритми су имплементирани у програмском језику *C#* и за комуникацију користе *WCF* протокол [105]. Експерименти су изведени на хомогеној мрежи рачунара (*Intel Core2Duo* процесор на 2 GHz и 2 GB RAM меморије и *Windows XP* оперативним системом). Тестирање је извршено над моделима *bg54*, *bg63* и *it206*, са толеранцијом на балансираност $\varepsilon=0.1$ (10%) и у мрежи од $p \in \{4, 6, 8\}$ процесора.

Сценарио тестирања је био следећи:

1. тестирани модел је издељен (иницијалном поделом) на p партиција и на основу ове поделе су подаци подељени на p процесора;
2. на случајан начин (отварањем и затварањем прекидача) се од издељеног балансираног графа добијају небалансирани графови;
3. покреће се одговарајући алгоритам за динамичку прерасподелу небалансираног графа (*WF* и *LMSR* алгоритам су покретани за исте графове у одвојеним тестовима);
4. на основу резултата алгорита, процесор координатор шаље задатке осталим процесорима у мрежи (захтеве за слањем одређених група података);
5. након пријема задатака од стране процесора координатора, сваки процесор прослеђује своје податке другим процесорима.

Свака врста теста је поновљена 100 пута генерисањем промене статуса прекидача одабраних на случајан начин, а добијени резултати (средње вредности) су приказани у табели 6.13.

Табела 6.13. Резултати извршавања *WF* и *LMSR* алгоритма

p	Модел	<i>bg54</i>		<i>bg63</i>		<i>it206</i>	
	Алгоритам	<i>WF</i>	<i>LMSR</i>	<i>WF</i>	<i>LMSR</i>	<i>WF</i>	<i>LMSR</i>
4	$\epsilon_{max}[\%]$	7.3	7.49	5.45	9.57	7.38	7.89
	C_{cut}	146	177	98	199	631	719
	$C_{migration}$	19970	31741	14105	24639	5807	41151
	$T_a[ms]$	9.04	0.28	6.7	0.27	5.58	0.41
	$T_{migration}[ms]$	256	413	223	340	120	442
6	$\epsilon_{max}[\%]$	8.06	9.36	6.9	7.3	7.3	9.84
	C_{cut}	149	251	92	230	624	688
	$C_{migration}$	30632	49491	18327	45063	6408	47006
	$T_a[ms]$	30.8	0.49	24.5	0.3	13.2	0.59
	$T_{migration}[ms]$	420	534	257	488	151	502
8	$\epsilon_{max}[\%]$	7.61	9.93	9.43	9.8	8.55	9.32
	C_{cut}	208	237	108	219	706	773
	$C_{migration}$	40682	46252	20654	51412	9509	34937
	$T_a[ms]$	41.6	0.8	27.6	0.51	16.2	0.76
	$T_{migration}[ms]$	495	523	306	577	217	396

На основу приказаних резултата може се утврдити да *WF* алгоритам постиже боље резултате од *LMSR* алгоритма по свим параметрима за све тестове. *WF* алгоритмом је остварена боља балансираност процесора (ϵ_{max}) као и мања тежина пресечних грана (C_{cut}). Овај параметар се односи на број потенцијалних конекција између података у различитим партицијама, што указује на мању вероватноћу да приликом промене статуса расклопне опреме дође до небалансираности система. Поред тога, применом *WF* алгоритма се добијају знатно боље (мање) вредности за количине података које је потребно преbacити ($C_{migration}$) са једног процесора на други, што директно утиче на оптерећење комуникационих путева. Са друге стране, *LMSR* је знатно бржи, али с обзиром на време трајања пребацивања података ($T_{migration} \gg T_a$) може се закључити да је *WF* алгоритам погоднији за употребу у дистрибутивним управљачким системима у реалном времену од *LMSR* алгоритма.

6.3.2. Тестирање *MWF* и *PNR* алгорита

У овој фази тестирања динамичке прерасподеле модела извршено је поређење резултата добијених применом *MWF* и *PNR* алгорита на моделу *bgb3*, и то за:

- различите степене небалансираности партиција;
- симулацију примене алгорита у дистрибуираном систему – када је небалансираност система узрокована променом статуса расклопне опреме.

Тестирање алгорита за различите степене небалансираности партиција. Сви тестови су вршени на рачунарима са *Intel Core2Duo* процесором на 2 GHz и 2 GB RAM меморије. Експерименти су вршени за различите степене небалансираности партиција и то када је небалансираност појединих партиција:

- већа од 50% просечне тежине партиција,
- мања од 50% просечне тежине партиција,
- без ограничења.

Тестови су поновљени 500 пута и просечне вредности су приказане у табели 6.14. У овим тестовима су коришћени параметри $\alpha = 1$, $\beta = 1.2 \cdot 10^{-4}$ и $\gamma = 6 \cdot 10^{-6}$ (из једначине (24)). У току експерименета мерено је време извршавања алгорита и сви параметри који фигуришу у критеријумској функцији (једначина (24)). Поред тога, у табели 6.14. су приказане вредности параметра n_V који представља број корена (чворова прорачуна) који су променили партицију.

Табела 6.14. Резултати тестирања *MWF* и *PNR* алгорита за различите степене небалансираности

stepen nebalansiranosti	$> 0.5\bar{W}_\pi$		$< 0.5\bar{W}_\pi$		произвољно	
	<i>MWF</i>	<i>PNR</i>	<i>MWF</i>	<i>PNR</i>	<i>MWF</i>	<i>PNR</i>
$\varepsilon_{max}[\%]$	6.77	6.97	6.80	7.08	6.79	7.04
n_V	7	10	4	7	5	8
$C_{migration}$	38908	41007	20355	23348	28305	30889
C_{cut}	205	197	236	222	223	212
$T_d[ms]$	15.04	39.30	8.36	27.09	11.22	32.30

На основу добијених резултата се може закључити да *PNR* алгорита постиже боље резултате са аспекта броја пресечних грана између партиција C_{cut} . Са друге стране, *MWF* алгорита је бржи, постиже незнатно бољу балансираност партиција, мањи број чворова (n_V) се пребацује и мања количина података мигрира у току прерасподеле ($C_{migration}$).

Симулација промене статуса расклопне опреме. Друга група тестова је везана за симулацију рада алгоритама у дистрибуираном систему. Размена информација и података између процесора је извршена преко *WCF* комуникационог протокола. Реални подаци дистрибутивне мреже *bg63* (објекти проводне опреме) су распоређени дистрибуирано на 4 рачунара и извршена је симулација динамичких промена у систему (отварања и затварања расклопне опреме). При томе су приликом промена статуса расклопне опреме мењани и статуси расклопне опреме унутар партиција – која не утичи на дисбалансирање система, па ни на покретање алгоритама. Међутим, промене статуса ове расклопне опреме утичу на промену графа и на касније акције везане за миграцију података у случају појаве небалансираног стања система.

У току тестирања је извршена промена статуса расклопне опреме 480 пута. С обзиром да је са аспекта покретања алгорита најзначајнија промена (затварање) граничних прекидача између две партиције – овакви прекидачи су форсирано затварани сваку четврту промену (изабран је гранични прекидач на случајан начин), док су остале три промене везане за отварање/затварање расклопне опреме унутар партиција (што утиче на промену чворова и њихових тежина). Према томе, алгоритми су позивани 120 (480/4) пута и резултати су приказани у табели 6.15. Приказани резултати немају за циљ поређење два алгорита (то је урађено претходним тестовима) пошто нису вршени под истим условима, већ да се након ових тестова створа слика о количини података који мигрирају између рачунара и времену потребном за преношење података.

Табела 6.15. *Резултати тестирања MWF и PNR алгоритама приликом симулације рада система*

Алгоритам	MWF	PNR
$\varepsilon_{max}[\%]$	6.63	8.37
n_V	4	4
$C_{migration}$	12495	10267
C_{cut} пре прерасподеле	117	121
C_{cut} након прерасподеле	116	116
$T_a[ms]$	10.83	26.16
$T_{migration}[ms]$	542	371

Параметар $T_{migration}$ представља време потребно за пребацивање података са једног рачунара на други. На основу резултата приказаних у табели може се утврдити да су времена потребна за миграцију података знатно већа од времена потребног за извршавање алгоритма ($T_{migration} \gg T_a$), па је због тога доминантнији критеријум количина података коју је потребно мигрирати, тј. у критеријумској функцији (једначина (24)) није ни укључено време извршавања алгоритма.

6.3.3. Поређење *WF* и *MWF* алгоритма

У циљу провере оправданости развоја *MWF* алгоритма, као измењене верзије *WF* алгоритма, извршени су упоредни тестови ова два алгоритма. Добијени резултати на 100 случајно генерисаних небалансираних варијанти модела су приказани у табели 6.16.

Табела 6.16. Упоредни тестови *WF* и *MWF* алгоритама

Модел		<i>bg54</i>		<i>bg63</i>		<i>it206</i>	
р	Алгоритам	<i>WF</i>	<i>MWF</i>	<i>WF</i>	<i>MWF</i>	<i>WF</i>	<i>MWF</i>
4	$\varepsilon_{max}[\%]$	7.02	6.31	7.06	6.45	8.22	7.98
	C_{cut}	268	291	333	362	701	724
	$C_{migration}$	33208	27747	28898	25737	20046	19322
	$T_a[ms]$	20.6	18	17.8	16.4	16.9	14.4

На основу добијених резултата може се закључити да сваки од алгоритама има својих предности: *WF* у прогледу броја пресечних грана (C_{cut}), а *MWF* са аспекта количине података које је потребно пребацивати из једне партиције у другу ($C_{migration}$). Коришћење *MWF* алгоритма (у односу на *WF*) се препоручује у дистрибутивним управљачким системима који раде у реалном времену због мањег времена неопходног за миграцију података и брзине извршавања алгоритма.

6.3.4. Тестирање алгоритама за динамичку прераспделу на *NUMA* рачунару

Експерименти из ове групе су обављени на *NUMA* платформи са 2 нода са по 4 процесорска језгра по ноду (*CPU AMD Opteron 2.4GHz, 8GB RAM* по језгру).

Алгоритми *MWF*, *CP*, *HMWF* и *HCP* су примењени за динамичку прераспделу и тестирани на моделима *it206* и *bgx5*. Тестови су извршени на издељеном графу који је дисбалансиран на случајан начин. У свим тестовима небалансираност је узрокована променом статуса граничних прекидача (тј. потенцијалних конекција) између чворова прорачуна који се налазе у различитим партицијама. У том случају се два чвора спајају, што може довести до небаласираног графа. Извршене су две

групе тестова: у тесту I активирани су потенцијалне конекције између чворова који се налазе у различитим *NUMA* нодовима, док се у тесту II активирала конекција само између чворова који се налазе на истом *NUMA* ноду. Тестови су поновљени 100 пута и просечне вредности значајних величина су наведене у табели 6.17. Сви тестови су вршени са толеранцију на балансираност постављену на $\varepsilon = 0.1$.

Табела 6.17. Резултати динамичке прерасподеле на *NUMA* рачунару

Тест	Модел	Алгоритам	C_{cut}	$n_{\Delta\pi}$	n_V	$C_{migration}$	T_a [ms]
I	it206	<i>MWF</i>	422	56	24	86722	30
		<i>CP</i>	317	78	27	57731	18
		<i>HMWF</i>	397	60	26	28990	73
		<i>HCP</i>	529	96	22	28965	18
	bgx5	<i>MWF</i>	503	14	9	81774	31
		<i>CP</i>	363	13	6	61016	21
		<i>HMWF</i>	472	20	14	62802	64
		<i>HCP</i>	399	22	12	68591	20
II	it206	<i>MWF</i>	417	55	30	107863	32
		<i>CP</i>	317	78	16	36325	17
		<i>HMWF</i>	301	52	0	0	42
		<i>HCP</i>	522	96	0	0	13
	bgx5	<i>MWF</i>	503	14	6	58175	45
		<i>CP</i>	363	13	6	43726	20
		<i>HMWF</i>	451	19	0	0	48
		<i>HCP</i>	398	19	0	0	11

Добијени резултати показују да *CP* алгоритам постиже најбоље резултате са аспекта броја пресечних грана у скоро свим тестовима. Међу осталим алгоритмима издваја се *HMWF* алгоритам који постиже нешто боље резултате за C_{cut} од *MWF* и *HCP* алгоритама.

Вредност параметра $C_{migration}$ представља количину података који се преносе између два *NUMA* нода. Поређењем добијених резултата са аспекта количине пренесених података закључује се да *HMWF* и *HCP* алгоритми постижу најбоље резултате (*HMWF* је незнатно бољи), док је *MWF* најлошији.

Поред тога, приказани су број чворова (корена) који мењају партицију - $n_{\Delta\pi}$ (међу њима се налазе и они који су у добром *NUMA* ноду, па нису мигрирали – само

су додељени другом процесору на истом ноду), као и n_V – број чворова који су мигрирали са једног нода на други. На основу добијених резултата евидентно је да *HCP* размењује највише чворова између партиција, али истовремено број чворова који мигрирају у други *NUMA* нод је веома мали (што је узроковано хијерархијском конфигурацијом алгоритма). На основу времена извршавања се може закључити да је *HCP* најбржи (нешто је бржи од *CP* алгоритма), док је *HMWF* најспорији.

Узимајући у обзир све добијене резултате, може се закључити да *HMWF* алгоритам постиже боље резултате од других алгоритама са аспекта количине података који се преносе са једног нода на други и веома добре резултате за број пресечних грана - C_{cut} , али је спорији од осталих алгоритама. Са друге стране, *HCP* алгоритам је најбржи и остварује веома добре резултате са аспекта миграције података, али са аспекта броја пресечних грана не показује стабилне (поуздано добре) резултате као *HMWF*. Међутим, имајући у виду да се ради о потенцијалним конекцијама између различитих партиција, тј. о вероватноћи да ће доћи до потенцијалног дисбаланса система – овај критеријум се у одређеним случајевима може разматрати са мањим приоритетом.

6.3.5. Дискусија резултата алгоритама за динамичку прераспodelу модела података

Упоредна анализа резултата динамичких алгоритама (приказаних у табелама 6.13.-6.16.) доводи до закључка да се *WF* знатно боље понаша од *LMSR* алгоритма по свим критеријумима: брзине динамичке прераспodelе модела и критеријумске функције.

Поређење *MWF* и *PNR* алгоритма је показало да се за случајно изабране небалансиране графове (табела 6.14.) *PNR* алгоритам постиже боље резултате за пресечне гране C_{cut} , али је лошији са аспекта количине података која треба да се пренесе са једног процесора на други.

Симулацијом промене статуса расклопне опреме се утврдило да постоји могућност процене утицаја броја пресечних грана на понашање система у дужем временском периоду (табела 6.15.). Ако се промена статуса потенцијалних веза (отварање/затварање граничних прекидача) између корена из различитих партиција дешава често, онда се на основу добијених резултата (табела 6.15.) може закључити да се 30% мање времена времена потроши на миграцију података уз коришћење *PNR* алгоритма. Уколико су промене потенцијалних веза између партиција релативно

ретке (процена је да процентуално то треба да буде 15% у односу на промене статуса остале расклопне опреме), тада се препоручује коришћење *MWF* алгоритма за динамичку прерасподелу модела података.

Слични закључци се добијају поређењем *MWF* и *WF* алгоритма (табела 6.16.): *WF* алгоритам се препоручује у случајевима када је динамика промене граничних прекидача између партиција релативно честа (25% од укупног броја промена статуса потенцијалних веза између корена), у супротном се препоручује коришћење *MWF* алгоритма.

Анализа резултата за динамичку прерасподелу модела података дистрибутивне мреже на *NUMA* рачунарима показује да је оправдана примена хијерархијских алгоритама (*HMWF* и *HCP*).

7. Закључак

Традиционални надзорно-управљачки електроенергетски системи су засновани на централизованој архитектури са аспекта складиштења и обраде података. Да би овакви системи могли да одговоре захтевима за обрадом велике количине података у реалном времену, потребно је да поседују скупе „суперрачунаре“ са великим бројем процесора. Међутим, имајући у виду стално проширење захтева који се постављају пред надзорно-управљачке дистрибутивне електроенергетске системе, може се очекивати да се количина података коју је потребно обрадити драстично повећава. Услед тога, ни моћни рачунари нису гаранција за квалитетну обраду велике количине података у условима тенденције сталног проширења модела података.

Развој *Smart Grid* система, у оквиру којих се дистрибутивни надзорно-управљачки системи интегришу са *GIS, EMS, OMS, MDM, AMI, HAN*, и др., као и преузимање и обрада великих количина података из ових система, додатно оптерећује нормално функционисање дистрибутивних електроенергетских система. Из тог разлога се јавила потреба за дистрибуираним системом и поделом модела на делове (партиције) који би се одвојено процесирали на различитим процесорима (рачунарима). Уколико би проширење модела података оптеретило одређени дистрибуирани рачунарски систем, једноставним додавањем процесора (рачунара) би се могао растеретити систем.

На основу реалних потреба и анализе начина извршавања најчешће коришћених аналитичких енергетских функција, осмишљена је процедура за поделу модела података електроенергетске мреже. Ова процедура се састоји из три фазе: тополошке анализе дистрибутивне мреже, формирање тежинског графа као резултата тополошке анализе и примена алгоритма за поделу графа на основу дефинисаног критеријума оптималности. Тополошка анализа се промењује на моделу конективности базираном на *СІМ* стандарду и у оквиру ње се проналазе групе елемената који се напајају из истог извора. Ове групе елемената се називају корени (или области прорачуна) и представљају се чворовима графа – тзв. чворовима прорачуна, док гране новоформираног графа настају на основу веза

између новоформираних чворова прорачуна. Дефинисан је критеријум оптималности за поделу модела на основу детаљне анализе понашања аналитичких енергетских функција који се спроводе у дистрибутивним електроенергетским системима.

Развијени су различити алгоритми за иницијалну поделу графова који представљају модел дистрибутивне електроенергетске мреже. Осмишљена су два нова – оригинална алгоритма за иницијалну поделу модела: хибридни генетски алгоритам и алгоритам суперкорена. Алгоритам суперкорена (*SR*) представља оригинални вишефазни алгоритам специјализован за поделу модела података који се своди на неповезан граф. Хибридни генетски алгоритам представља спој генетског алгоритма и једног од најчешће коришћених алгоритама за кориговање поделе већ издељеног графа – *FM* алгоритма. *FM* алгоритам је примењен у фази мутације генетског алгоритма, чиме је добијен хибридни генетски алгоритам са веома dobrim карактеристикама у погледу добијених резултата. Поред тога, општеприхваћени алгоритми (*PSO* и генетски алгоритам) су прилагођени конкретном проблему поделе графова, при чему су тестиране и дистрибуиране верзије наведених алгоритама. Сви развијени алгоритми су тестирани на реалним моделима података дистрибутивних електроенергетских мрежа. Добијени резултати су поређени са резултатима које постиже један од најчешће коришћених алгоритама за поделу графова – *METIS* алгоритам. Нови алгоритми су показали изузетно добре резултате, чиме је потврђена оправданост њихове примене за иницијалну поделу података. На основу добијених резултата може се закључити да је *SR* алгоритам је остварио најбоље резултате у великој већини тестова, па се стога *SR* алгоритам препоручује за иницијалну поделу великих модела података дистрибутивне електроенергетске мреже.

У дисертацији је обухваћен и проблем динамичке прерасподеле модела података дистрибутивне мреже. Осмишљени су нови алгоритми за динамичку прерасподелу модела података (*MWF* алгоритам) и за динамичку прерасподелу модела у оквиру мултипроцесорских хијерархијских организованих архитектура (*HMWF* и *HCP* алгоритам). Резултати новоформираних алгоритама су поређени са резултатима познатих алгоритама за динамичку прерасподелу (*WF*, *PNR*, *LMSR* и *CP*). Анализирани су различити случајеви динамике промене граничних прекидача и у случајевима када је динамика промене граничних прекидача релативно ретка (15% у односу на промену остале расклопне опреме), тада се препоручује коришћење новоразвијеног *MWF* алгоритма у односу на оригинални *WF* алгоритам (на основу

кога је *MWF* настао). Гранични прекидачи (потенцијалне конекције) између корена (из различитих партиција) у нормалном режиму рада система су по дефиницији отворени. Према томе, промена њиховог стања се дешава веома ретко - само у случају квара неког дела мреже или промени нормалног режима рада (тзв. нормалног уклопног стања, што се у неким дистрибутивним системима дешава са променама годишњих доба), па је услов за ретким променама граничних прекидача аутоматски задовољен. Наведеној констатацији иде у прилог чињеница да је тополошка структура дистрибутивних мрежа радијална и слабо упетљана, па је стога и број потенцијалних веза између различитих области прорачуна занемарљив у односу на укупну количину расклопне опреме.

На основу експеримената којима је симулирана промена статуса расклопне опреме и примена алгоритама за динамичку прераспodelу модела података, утврђено је да постојећи – *PNR* алгоритам постиже најбоље резултате у условима који одговарају реалним потребама дистрибутивних надзорно-управљачких система.

Правци даљих истраживања на проблему поделе модела података дистрибутивне мреже би се могли свести на следеће задатке:

- проширити критеријум за одређивање комплексности прорачуна аналитичких електроенергетских функција на анализу петљи и процену комплексности функција које се извршавају над више корена (региона),
- детаљније анализирати и оптимизовати рад новоразвијених алгоритама за иницијалну поделу (алгоритма суперкорена и хибридног генетског алгоритма) у случајевима мрежа које се сведе на неповезане графове,
- анализирати алгоритме за динамичку прераспodelу модела података у условима када се прорачуни извршавају дистрибуирано у реалном времену,
- применити развијену теорију и алгоритме на друге критичне инфраструктурне системе као што су системи за пренос и дистрибуцију воде, нафте, гаса и др.

Литература

- [1] Strezoski, V., Popović, D. S., Bekut, D., Katić, N., Švenda, G., Gorečan, Z., Dujić, J.: Osnovne energetske funkcije za analizu, upravljanje i planiranje pogona srednjenaponskih distributivnih mreža, IV skup Trendovi Razvoja: „Nove tehnologije u elektrodistribuciji“, Kopaonik, 1998.
- [2] Sionsansi, F.P.: Smart Grid: Integrating Renewable, Distributed & Efficient Energy, Academic Press, Elsevier, 2011.
- [3] Bose A.: Smart Transmission Grid Applications and Their Supporting Infrastructure, IEEE Transactions on Smart Grid, Vol. 1, No. 1, pp. 11-19., 2010.
- [4] Santacana, E., Rackliffe, G., Tang, L., Feng, X.: Getting Smart, IEEE Power and Energy Magazine, Vol.8, No.2, pp.41-48, 2010.
- [5] Jiyuan, F., Borlase, S.: The Evolution of Distribution, IEEE Power & Energy Magazine, pp. 63-68, march/april 2010.
- [6] Farhangi, H.: The Path of Smart Grid, IEEE Power & Energy Journal, Vol 8, No 1., pp. 18-28, 2010.
- [7] Vukmirović S., Erdeljan A., Lendak I., Čapko D. A novel software architecture for smart metering systems, Journal of Scientific & Industrial Research, Vol 69, pp. 937-941, 2010.
- [8] McConnell, B.W., Reddoch, T.W., Purucker, S.L., Monteen L.D.: Distribution Energy Control Center Experiment, IEEE Transactions on Power Apparatus and Systems, Vol. PAS-102, No. 6, pp. 1582-1587, 1983.
- [9] Gonen, T.: Electric Power Distribution System Engineering, McGraw-Hill, Inc., 1986.
- [10] Cassel, W. R.: Distribution Management Systems: Functions and Payback, IEEE Transactions on Power Systems, Vol. 8, No. 3., pp. 796-801, 1993.
- [11] Čapko, D., Erdeljan, A.: Vruća rezerva OPC servera veličina, 47. Konferencija ETRAN 2003., Herceg Novi, 2003.
- [12] Čapko, D., Erdeljan, A.: Udvojeni OPC serveri veličina u nadzorno upravljačkim sistemima”, Industrijska energetika, Lepenski Vir, 2004.
- [13] Čapko, D., Erdeljan, A.: Povezivanje klijentskih aplikacija sa redundantnim serverima za pristup velicinama”, 49. Konferencija ETRAN 2005., Budva, 2005.
- [14] Čapko D., Erdeljan A., Popović D., Lendak I.: The Solution of Redundancy Problem in Distributed Control Systems with Hot Backup of Data Access Server, 8th International Symposium Interdisciplinary Regional Research (Hungary, Romania, Yugoslavia) ISIRR 2005, Segedin, 2005.
- [15] Erdeljan, A.: Distribuiran sistem za automatski nadzor i upravljanje zasnovan na savremenim tehnikama programiranja, doktorska disertacija, Novi Sad, Februar 2000.

- [16] Erdeljan, A., Trninić, N., Čapko, D.: An OPC Data Access Server Designed for Large Number of Items, 6th International Symposium Interdisciplinary Regional Research (Hungary, Romania, Yugoslavia) ISIRR 2002, Novi Sad, 2002.
- [17] Erdeljan A., Popović D., Lendak I., Čapko D.: Client Side Architecture of a High Speed Data Access Server in Distribution Management Systems, 8th International Symposium Interdisciplinary Regional Research (Hungary, Romania, Yugoslavia) ISIRR 2005, Segedin, 2005.
- [18] Popovic, I., Vrtunski, V., Popovic, M.: Formal Verification of Distributed Transaction Management in a SOA Based Control System. ECBS 2011, pp. 206-215, 2011.
- [19] Popović, D.S., Stanković, P.B: Mogućnosti integracije Energetskih aplikacija i SCADA sistema, VI skup Trendovi Razvoja: „Nove tehnologije u elektrodistribuciji“, Kopaonik, 2000.
- [20] Stokić, V. Stričević, L., Galović, A., Nikolić, M., Trninić, N., Čapko, D., Poljački, J., Hajduković, M.: Database and Communication Aspects of Fault Tolerance in a Distributed Management System, 8th International Symposium Interdisciplinary Regional Research (Hungary, Romania, Yugoslavia) ISIRR 2007, Novi Sad, 2007.
- [21] Nedić, N., Erdeljan, A., Lendak, I., Čapko D.: Pristup modelu podataka elektroenergetskog sistema putem Web servisa“, 53. Konferencija ETRAN 2009, Vrnjačka Banja, 2009.
- [22] Northcote-Green, J., Wilson, R.: Control and Automation of Power Distribution Systems, CRC Press, 2007.
- [23] Lin, C. L., Layland, J. W.: Scheduling Algorithms for Multiprogramming in Hard-Real-Time Environment, Journal of the ACM, Vol. 20., No 1., pp. 46-61, 1973.
- [24] Momoh, J.A.: Electric Power Distribution, Automation, Protection, and Control, CRC Press, 2007.
- [25] Popović, D.S.: Power applications: A cherry on the top of the DMS cake, DA/DSM DistribuTECH Europe track 3, Vienna, Austria, 2000.
- [26] Popović, D., Bekut, D., Treskanica, V.: Specijalizovani DMS algoritmi, DMS group, Novi Sad, 2004.
- [27] Allaoua, B., Laoufi, A.: Optimal Power Flow Solution Using Ant Manners for Electrical Network, Journal of Advances in Electrical and Computer Engineering, Romania, Vol.9, No.2, pp. 34-40, 2009.
- [28] Shirmohammadi, D., Hong, H.W., Semlyen, A., Luo, G.X.: A Compensation-Based Power Flow Method For Weakly Meshed Distribution And Transmission Networks, IEEE Transactions on Power Systems, Vol. 3, No. 2, pp. 753-762, 1988.
- [29] Strezoski, V., Katic, N., Janjic, D.: Voltage control integrated in distribution management system, Electric Power Systems Research, No. 60, pp. 85-97, 2001.
- [30] Čapko, D., Erdeljan, A., Popovic, M., Svenda, G.: An Optimal Relationship-Based Partitioning of Large Datasets, 14th East-European Conference on Advances in Databases and Information Systems, Novi Sad, Lecture Notes in Computer Sciences, Vol.6295, pp.547-550., 2011.
- [31] Popovic, M., Basicovic, I., Vrtunski, V.: A Task Tree Executor: New Runtime for Parallelized Legacy Software, 16th Annual IEEE International Conference and

- Workshop on the Engineering of Computer Based Systems, ECBS 2009, San Francisco, USA, 2009.
- [32] Igumenov, A., Petkus, T.: Analysis of Parallel Calculations in Computer Network, Information Technology and Control, Vol.37, No.1, pp. 57-62, 2008.
- [33] IEEE Committee Report, "Parallel Processing in Power Systems Computations", IEEE Transactions on Power Systems, Vol.7, No.2, pp. 629-638, 1992.
- [34] Grama, A., Gupta, A., Karypis, G., Kumar, V.: Introduction to Parallel Computing, Second Edition, Addison Wesley, 2003.
- [35] Falcao, D.M.: High Performance Computing in Power System Applications, Selected papers from the Second International Conference on Vector and Parallel Processing, Systems and Application, Porto, Portugal, pp.1-23., 1996.
- [36] Kleinberg, M., Miu, K., Nwankpa, C.: A Study of Distribution Power Flow Analysis Using Physically Distributed Processors, IEEE Power Systems Conference and Exposition, Atlanta, pp 518-521, 2006.
- [37] Xue, W., Qi, S.: Multilevel Task Partition Algorithm for Parallel Simulation of Power System Dynamics, Proceedings of the 7th international conference on Computational Science, Part I: ICCS 2007, Beijing, China, 2007.
- [38] Heydt, G.: The next generation of power distribution systems, IEEE Transactions on Smart. Grid, Vol. 1, No. 3, pp. 225-235, 2010.
- [39] Strezoski, V., Popović, D. S., Bekut, D., Švenda, G.: DMS - Basis for Increasing of the Penetration of Green Distributed Generation in Distribution Networks, V International Renewable Energy Forum CLEAN ENERGY TECHNOLOGIES – 2011, Novi Sad, 2011.
- [40] IEC 61970 Energy management system application program interface (EMS-API) – Part 301: Common Information Model (CIM) Base, IEC, Edition 2.0, 2007.
- [41] IEC/TR 61968-11 Application integration at electric utilities - System interfaces for distribution management - Part 11: Common information model (CIM) extensions for distribution, IEC, Edition 1.0, 2010.
- [42] IEC 61970 Energy management system application program interface (EMS-API) – Part 403: Generic data access, IEC, 2008.
- [43] OMG Specification: Utility Management Systems (UMS) Data Access Facility DAF, Version 2.0.1, 2005.
- [44] Erdeljan, A., Čapko, D.: Analiza primene DAF i GDA specifikacija za pristup objektnom modelu UMS-a, 51. Konferencija ETRAN 2007., Herceg Novi, jun 2007.
- [45] Vukmirović, S., Erdeljan, A., Lendak, I., Čapko, D.: Extension of the Common Information Model with Virtual Meter, Electronics and Electrical Engineering, Kaunas: Technologija, Vol.107, No.1.,pp. 59-64, 2011.
- [46] Strezoski, V.: Osnovni pojmovi relevantni za tehnički sistem upravljanja distributivnih mreža, V skup Trendovi Razvoja: „Nove tehnologije u elektrodistribuciji“, Kopaonik, 1999.

- [47] Čapko, D., Erdeljan, A., Popovic, M., Svenda, G.: An Optimal Initial Partitioning of Large Data Model in Utility Management Systems, *Advances in Electrical and Computer Engineering*, Vol. 11, No. 4, pp. 41-46, 2011.
- [48] Čapko, D., Erdeljan, A., Vukmirović, S., Lendak, I.: A Hybrid Genetic Algorithm for Partitioning of Data Model in Distribution Management Systems, *Information technology and control*, Vol. 40, No. 4, pp. 316-322, 2011.
- [49] Zhang, P., Marti, J.R., Dommel, H.W.: Network Partitioning for Real-Time Power System Simulation, *International Conference on Power System Transients*, Montreal, Canada, 2005.
- [50] Schloegel, K., Karypis, G., Kumar, V.: Multilevel diffusion schemes for repartitioning of adaptive meshes. *Journal of Parallel and Distributed Computing*, Vol. 47, No. 2, pp. 109-124, 1997.
- [51] Schloegel, K., Karypis, G., Kumar, V.: Wavefront diffusion and LMSR: Algorithms for dynamic repartitioning of adaptive meshes, *Technical Report TR 98-034*, Dept. of Computer Science and Engineering, University of Minnesota, 1998.
- [52] Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, San Francisco, 1979.
- [53] Bui, T.N., Moon, B.R.: Genetic Algorithm and Graph Partitioning, *IEEE Trans. Computers*, Vol. 45, No. 7, pp. 841-855, 1996.
- [54] Talbi, E.G., Bessiere, P.: A Parallel Genetic Algorithm for the Graph Partitioning Problem, In: *Proc. of the International Conference on Supercomputing*, Cologne, 1991.
- [55] Craus, M., Bulancea, C.: Using Agent Technology Combined with ACO Metaheuristics on Load Balancing Algorithms, *Journal of Advances in Electrical and Computer Engineering*, Vol 4, No. 2, pp. 55-59, 2004.
- [56] Kennedy, J., Eberhart, R.: Particle swarm optimization, In: *Proceedings of the 1995 IEEE International Conference on Neural Networks (ICNN)*, IEEE Service Center, Piscataway, New Jersey, Vol.4, pp.1942–1948, 1995.
- [57] Čapko, D., Erdeljan, A., Vukmirovic, S., Lendak, I.: Using Genetic Algorithm for Power Distribution Network Partitioning, *2nd Regional Conference, Industrial Energy and Environmental Protection in South Eastern European Countries*, Zlatibor, Serbia, 2010.
- [58] Čapko, D., Erdeljan, A., Lendak, I.: PSO algorithm for Graph Partitioning, *17th Telecommunication Forum 2009*, Belgrade, 2009.
- [59] Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal of Scientific Computing*, Vol.20, No.1, pp. 359-392, 1998.
- [60] Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, Vol. 49, No. 2, pp. 291-307, 1970.
- [61] Fiduccia, C. M., Mattheyses, R. M.: A linear time heuristic for improving network partitions. In: *Proc. 19th IEEE Design Automation Conference*, pp. 175-181, 1982.
- [62] Sanchis, L.: Multiple-Way Network Partitioning, *Technical Report – TR 181*, University of Rochester, Computer Science Department, 1986.

- [63] Darwin, Č.: Postanak vrsta, Nolit, Beograd, 1985.
- [64] Holland, J.H.: Adaptation in Natural and Artificial Systems, The University of Michigan Press, 1975.
- [65] Sivanandam, S.N., Deepa, S.N.: Introduction to Genetic Algorithms, Springer Verlag, 2007.
- [66] Choi, S.S., Moon, B.R.: Normalization in genetic algorithms, In Proceedings of the Genetic and Evolutionary Computation Conference, pp. 862-873, 2003.
- [67] Cantu-Paz, E.: A Survey of Parallel Genetic Algorithms, Technical Report, Illinois Genetic Algorithms Laboratory, Department of General Engineering, Illinois, 1997.
- [68] Tomassini, M.: Parallel and Distributed Evolutionary Algorithms: A Review, Chapter 7 in Evolutionary Algorithms in Engineering and Science, Wiley, 1999.
- [69] Wilson, E.O.: Sociobiology: The new synthesis. Belknap Press, Cambridge, 1975.
- [70] Kennedy, J., Eberhart, R.: A discrete binary version of the particle swarm algorithm, In: Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics, IEEE Service Center, Piscataway, New Jersey, Vol. 5, pp. 4104–4108, 1997.
- [71] Laskari, E., Parsopoulos, K., Vrahatis, M.: Particle swarm optimization for integer programming, In: Proceedings of the IEEE Congress on Evolutionary Computation, Honolulu, Hawaii USA, Vol. 2, pp. 1582–1587, 2002.
- [72] Sun, L., Leng, M. Yu, S.: A New Multi-level Algorithm Based on Particle Swarm Optimization for Bisecting Graph, In: Proceedings of the 3rd international conference on Advanced Data Mining and Applications, Harbin, China, pp. 69-80, 2007.
- [73] Miranda, V. and Fonseca, N.: EPSO - Evolutionary Particle Swarm Optimization, a New Algorithm with Applications in Power Systems, Proceedings of IEEE/PES Transmission and Distribution Conference and Exhibition 2002: Asia Pacific, Vol.2, pp.745–750, 2002.
- [74] Barnard, S.T., Simon, H.D.: A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems, in Proc. 6th SIAM Conf. Parallel Processing for Scientific Computing, SIAM, pp. 711-718., 1993.
- [75] Abou-Rjeili, A., Karypis G.: Multilevel Algorithms for Partitioning Power-Low Graphs, IEEE International Parallel & Distributed Processing Symposium (IPDPS), 2006.
- [76] Walshaw, C., Cross, M.: JOSTLE: Parallel Multilevel Graph-Partitioning Software – An Overview. In F. Magoules, editor, Mesh Partitioning Techniques and Domain Decomposition Techniques, Civil-Comp Ltd, pp. 27-58, 2007.
- [77] Pellegrini, F., Roman, J.: SCOTCH: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs, HPCN-Europe, Springer Lecture Notes in Computer Sciences, Vol.1067, pp. 493-498,1996.
- [78] Henderson, B., Leland, R.: A Multilevel Algorithm for Partitioning Graphs, Proceedings of ACM/IEEE conference on Supercomputing, San Diego, 1995.

- [79] Meyerhenke, H., Monien, B., Schamberger, S.: Accelerating shape optimizing load balancing for parallel FEM simulations by algebraic multigrid, In Proc.20th Intl. Parallel and Distributed Processing Symposium (IPDPS'06), IEEE, 2006.
- [80] Meyerhenke, H., Monien, B., Sauerwald, T.: A new diffusion-based multilevel algorithm for computing graph partitions of very high quality, In: Proceedings of the 22nd International Parallel and Distributed Processing Symposium (IPDPS'08), IEEE Computer Society, pp. 1-13, 2008.
- [81] Henderson, B., Kolda, T.G.: Graph partitioning models for parallel computing, *Parallel Computing*, Vol. 26, No. 12, pp. 1519-1534, 2000.
- [82] Grigoras G., Cartina G., Bobric E.C.: Strategies for Power/Energy Saving in Distribution Networks, *Journal of Advances in Electrical and Computer Engineering*, Romania, Vol 9., No.1, pp. 61 –64, 2010.
- [83] Godsil, C., Royle, G.: *Algebraic Graph Theory*, Springer Verlag, 2001.
- [84] Aggarwal, G., Motwani, R., Zhu, A.: The Load Rebalancing Problem, in Proc. ACM SPAA, 2003.
- [85] Korf, R.E.: A Complete Anytime Algorithm for Number Partitioning, *Artificial Intelligence*, Vol.106, No.2, pp. 181-203, 1998.
- [86] Korf, R.E.: Multi-Way Number Partitioning, *International Joint Conference on Artificial Intelligence*, Pasadena, California, US, 2009.
- [87] Misevičius. A.: An Extension of Hybrid Genetic Algorithm for the Quadratic Assignment Problem, *Information Technology and Control*, Vol.4, No.33, pp. 53-60, 2004.
- [88] Misevičius, A., Rubliauskas., D.: Performance of Hybrid Genetic Algorithm for The Grey Pattern Problem, *Information Technology and Control*, Vol.34, No.1, pp. 15-24, 2005.
- [89] El-Mihoub, T.A., Hopgood, A.A., Nolle, L., Battersby A.: Hybrid Genetic Algorithm: A Review, *Engineering Letters*, Vol.13, No.2, pp. 124–137 , 2006.
- [90] Kang, S., Moon, B.: A Hybrid Genetic Algorithm for Multiway Graph Partitioning, In Proceedings of the Genetic and Evolutionary Computation Conference, 2000.
- [91] Miranda, V., Keko, H., Jaramillo, A.: *EPSO: Evolutionary Particle Swarms*, *Advances in Evolutionary Computing for System Design*, Springer, pp.139-167., 2007.
- [92] Castanos, J.G.: "The Dynamic Adaptation of Parallel Mesh-Based Computation". Department of Computer Science Brown University Providence, Rhode Island 02912, May 1996.
- [93] Cybenko, G.: Dynamic load balancing for distributed memory multiprocessors, *Journal of Parallel and Distributed Computing*., Vol. 7, No. 2, pp. 279-301, 1989.
- [94] Hu, Y.F., Blake, R.J.: An optimal dynamic load balancing algorithm, Technical Report DL-P-95-011, Daresbury Laboratory, Warrington, UK, 1995.
- [95] Meyerhenke, H.: Dynamic Load Balancing for Parallel Numerical Simulations based on Repartitioning with Disturbed Diffusion, *15th International Conference on Parallel and Distributed Systems*, 2009.

- [96] Čapko, D., Erdeljan, A., Popovic, M., Svenda, G.: Dynamic Repartitioning of Large Data Model in Distribution Management Systems, Electronics and electrical engineering, Kaunas: Technologija, Vol. 121, No. 5, (Accepted paper), 2011.
- [97] Castanos, J.G., Savage, J.E.: Repartitioning Unstructured Adaptive Meshes, Department of Computer Science, Brown University, 2000.
- [98] Čapko, D., Erdeljan, A., Lendak, I., Ilić, S.: Primena difuzionog algoritma za dinamičku preraspodelu modela distributivne elektroenergetske mreže, 56. Konferencija ETRAN, Teslić, 2011.
- [99] Correa, M., Chanin, R., Sales, A., Scheer, R., Zorzo, A.F.: Multilevel Load Balancing in NUMA Computers, Technical Report TR 049, PUCRS, Porto Alegre, 2005.
- [100] Eberhart, R., Shi, Y.: Particle Swarm Optimization: Developments, Applications and Resources, In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2001), Seoul, Korea, 2001.
- [101] Gropp, W., Lusk, E., Skjellum, A.: Using MPI, 2nd Edition: Portable Parallel Programming with the Message Passing Interface, Cambridge, MIT Press, 1999.
- [102] MPI Forum: MPI: A Message-Passing Interface Standard, Version 2.2, High Performance Computing Center Stuttgart, 2009.
- [103] Mihajlović, D.: Podela grafa distribuiranim genetskim algoritmom, Diplomski rad, Fakultet tehničkih nauka, 2010,
- [104] Čapko D., Erdeljan A., Vukmirovic S., Lendak I.: Multilevel algorithm for data model partitioning in power distribution networks, 16th International Symposium POWER ELECTRONICS Ee2011, Novi Sad, Serbia, 2011.
- [105] Stojanović, Đ., Vukmirović, S., Čapko, D.: Analiza performansi Windows Communication Foundation-a u UMS sistemima, 52. Konferencija ETRAN 2008, Palić, 2008.