UNIVERZITET U BEOGRADU
ELEKTROTEHNIČKI FAKULTET

Miroslava M. Ignjatović

HEURISTIKA ZA AUTOMATSKO SASTAVLJANJE PARALELNIH TESTOVA ZNANJA

doktorska disertacija

Beograd, 2022

UNIVERSITY OF BELGRADE
SCHOOL OF ELECTRICAL ENGINEERING

Miroslava M. Ignjatović

HEURISTIC FOR AUTOMATED PARALLEL KNOWLEDGE TESTS ASSEMBLY

Doctoral Dissertation

Belgrade, 2022

MENTOR:

dr Igor Tartalja, vanredni profesor
Univerzitet u Beogradu – Elektrotehnički fakultet

ČLANOVI KOMISIJE:

dr Dragan Bojić, redovni profesor
Univerzitet u Beogradu – Elektrotehnički fakultet

dr Tatjana Lutovac, redovni profesor
Univerzitet u Beogradu – Elektrotehnički fakultet

dr Dejan Simić, redovni profesor
Univerzitet u Beogradu – Fakultet organizacionih nauka

dr Jelica Protić, redovni profesor
Univerzitet u Beogradu – Elektrotehnički fakultet

dr Milo Tomašević, redovni profesor
Univerzitet u Beogradu – Elektrotehnički fakultet

Datum odbrane: _____ 2022. godine.

Naslov disertacije: Heuristika za automatsko sastavljanje paralelnih testova znanja.

Sažetak: Paralelni testovi znanja su testovi koji sadrže različite skupove stavki (skupove pitanja ili zadataka različitih vrsta) sa približno jednakim mernim osobinama, što ih čini međusobno zamenljivim. Oni imaju primenu kada više ispitanika rešava test, bez obzira da li to čine u isto ili u različito vreme. Rezultati dobijeni testiranjem moraju da budu poredivi.

Automatizam u procesu sastavljanja paralelnih testova znanja je praktično neophodan da bi se postigla objektivnost ispitivanja dobrim balansiranjem mernih osobina različitih primeraka testa i smanjila mogućnost grešaka ispitivača u merenju znanja. Što je testiranje masovnije, ušteda uloženog vremena ispitivača za pripremu paralelnih testova, primenom automatizacije sastavljanja, postaje značajnija.

Pošto problem automatskog sastavljanja paralelnih testova (ASPT) spada u NP-teške probleme kombinatorne optimizacije, korišćenje heuristika je očekivano i neophodno. Problem ASPT je izomorfan sa nekim postojećim problemima kombinatorne optimizacije, pa se heuristike za rešavanje tih problema mogu koristiti i za rešavanje problema ASPT. U disertaciji je dat opsežan pregled postojećih formulacija ovog problema, odnosno njihovih matematičkih modela, a zatim pregled heuristika koje se koriste za rešavanje različitih formulacija problema. Najpre su formulacije problema i heuristička rešenja nezavisno klasifikovani, a zatim se njihovom unakrsnom klasifikacijom ukazalo na otvoren prostor za istraživanje kroz nove kombinacije formulacija problema i heuristika za njihovo rešavanje.

Heuristike koje se koriste za rešavanje problema ASPT su uglavnom heuristike poboljšanja, čije se izvršenje prekida ili kada dođe do unapred zadatog vremena izvršavanja ili kada je dostignut željeni kvalitet rešenja. Konstruktivne heuristike se uglavnom koriste za kreiranje početnog rešenja za heuristike poboljšanja. Fokus ove disertacije je na predlogu novog efikasnog metoda polinomijalne kompleksnosti za kompletno rešavanje problema ASPT. Predloženi metod rešava jednu formulaciju *Problema pakovanja u korpe* prilagođavanjem konstruktivne heuristike koju su predložili Nawaz, Enscore i Ham, a koja do sada nije primenjivana za rešavanje problema ASPT. Među karakteristikama novog metoda ističu se predvidivo vreme njegovog izvršenja, efikasnost i jednostavnost implementacije. Predvidivo vreme i efikasno sastavljanje testova je od posebnog značaja u obrazovnim ustanovama u kojima se vrše česta testiranja relativno malog broja ispitanika, a gde su raspoloživi računarski resursi za ASPT ograničeni.

Predloženi metod ASPT je implementiran i upoređen sa efikasnim poredivim metodima koji su zasnovani na heuristikama poboljšanja, kakve su simulirano kaljenje (eng. *Simulated Annealing*, SA) i pretraga promenljivih okolina (eng. *Variable Neighborhood Sear*ch, VNS). U otvorenoj literaturi, VNS metod daje rezultate najboljeg kvaliteta za problem ASPT formulisan kao *Problem pakovanja u korpe*. Vreme izvršenja predloženog algoritma je značajno kraće u poređenju sa SA algoritmom, dok je kvalitet zanemarljivo lošiji. U poređenju sa VNS algoritmom, prosečno vreme izvršavanja je značajno kraće, dok za isto vreme izvršavanja (postignuto prekidanjem izvršavanja VNS algoritma), predloženi novi algoritam postiže bolji kvalitet rešenja za slučaj relativno malog broja paralelnih testova (20, 30 i 60) i banke stavki relativno male veličine (300 stavki), kakvi se često sreću kod testova znanja u obrazovnim institucijama.

Ključne reči: automatsko sastavljanje testova, paralelni testovi, kombinatorna optimizacija, pakovanje u korpe, heuristika, konstruktivna heuristika.

Naučna oblast: Elektrotehnika i računarstvo.

Uža naučna oblast: Softversko inženjerstvo.

Title of the doctoral dissertation: Heuristic for automated parallel knowledge tests assembly

Abstract: Parallel knowledge tests are tests which contain different sets of items (sets of questions or assignments of different types) with approximately equal measurement properties, that make them interchangeable. They are applicable when more examinees are solving the test, regardless of whether they do it at the same time or at different times. The results of the testing must be comparable.

Automation of the parallel test assembly process is practically necessary in order to achieve objectivity of the testing by well balancing the measurement properties of different test forms and to reduce the possibility of subjective errors by the examiner. Also, the more massive the testing, the more significant is saving up the invested examiner's time for preparation of parallel tests, by automation of assembly process.

Since the automated parallel test assembly problem (APTA) belongs to NP-hard combinatorial optimization problems, the use of heuristics is expected and necessary. The APTA problem is isomorphic with some existing combinatorial optimization problems, therefore heuristics for solving those problems can also be used for solving the APTA problem. In the dissertation an extensive review of the existing formulations of this problem is presented, its mathematical models, then an overview of the heuristics used for solving different problem formulations. Firstly, formulations of the problem and heuristic solutions are independently classified, then their cross-classification pointed to an open space for exploration through new combinations of formulations of the problem and heuristics for their solving.

The heuristics used to solve ASPT problems are mainly improvement heuristics, where the execution is interrupted either when a predetermined execution time occurs or when the desired quality of the solution is reached. Constructive heuristics are mainly used to create an initial solution for improvement heuristics. The focus of this dissertation is on the proposal of a new efficient method of polynomial complexity for the complete solving the APTA problem. The proposed method solves one formulation of the bin packing problem by adapting the constructive heuristic proposed by Nawaz, Enschore and Ham, which has not been used for solving the APTA problem so far. Among the characteristics of the new method, predictable execution time, efficiency and simplicity of implementation stand out. Predictable time and efficient tests assembly is of special importance in educational institutions where frequent testing of a relatively small number of examinees is performed, and where the available computer resources for ASPT are limited.

The proposed APTA method has been implemented and compared to the efficient comparable methods based on improvement heuristics, such as simulated annealing (SA) and variable neighbourhood search (VNS). In open literature, the VNS method gives the results of the best quality for the APTA problem formulated as the bin packing problem. The execution time of the proposed algorithm is significantly shorter compared to the SA algorithm, while the quality is negligibly worse. Compared to the VNS algorithm, the average execution time is significally shorter, while for the same execution time (achieved by interrupting the execution of the VNS algorithm), the proposed new algorithm achieves a better quality of result for the case of a relatively small number of parallel tests (20, 30 and 60) and relatively small item banks (300 items), which are often found in knowledge tests in education institutions.

Key words: automated test assembly, parallel tests, combinatorial optimization, bin packing, heuristic, constructive heuristic.

Scientific field: Electrical engineering and computer science.

Scientific subfield: Software engineering.

Sadržaj

Sadržaj ... iii
Spisak skraćenica ... iv
Spisak slika ... v
Spisak tabela ... vi
Spisak pseudokoda ... vii
1 Uvod ... 1
2 Postavka problema ... 5
2.1 Teorije testa ... 5
2.1.1 Klasična teorija testa ... 5
2.1.2 Teorija odgovora na stavku ... 8
2.2 Problem ASPT kao problem kombinatorne optimizacije ... 11
2.2.1 Problem kombinatorne optimizacije ... 12
2.2.2 Neki poznati problemi kombinatorne optimizacije koji se koriste i za ASPT ... 15
2.3 Problem ASPT rešavan u disertaciji ... 19
3 Pregled postojećih metoda ... 21
3.1 Pregled i klasifikacija formulacija problema ASPT ... 21
3.1.1 Ciljevi zasnovani na psihometriji ... 22
3.1.2 Uparivanje sa srodnim testovima ... 33
3.1.3 Maksimizacija broja paralelnih testova ... 35
3.1.4 Sumarni pregled formulacija problema ... 37
3.2 Pregled i klasifikacija heurističkih rešenja problema ASPT ... 38
3.2.1 Heuristike ... 40
3.2.2 Metaheuristike ... 44
3.2.3 Sumarni pregled postojećih heuristika za rešavanje problema ASPT ... 52
3.3 NEH heuristika ... 55
4 Predlog novog metoda ASPT ... 59
5 Rezultati ... 64
5.1 Postavka eksperimenta ... 64
5.2 Poređenje NEHTA algoritma sa SA algoritmom ... 65
5.3 Poređenje NEHTA algoritma sa VNS algoritmom ... 67
5.4 Paralelni NEHTA algoritam (NEHTA*) ... 71
6 Zaključak ... 73
Literatura ... 75

Spisak skraćenica

ASPT	Automatsko sastavljanje paralelnih testova
AST	Automatsko sastavljanje testa
BA	Pčelinji algoritam (eng. *Bees Algorithm*)
BPP	Problem pakovanja u korpe (eng. *Bin Packing Problem*)
BST	Velika test-senka (eng. *Big Shadow Test*)
CAT	Adaptivni testovi (eng. *Computerized Adaptive Tests*)
CBT	Testiranje uz pomoć računara (eng. *Computer Based Testing*)
CFT	Testovi sa fiksnom formom (eng. *Computerized Fixed Tests*)
CMT	Višestepeni testovi (eng. *Computerized Multistage Tests*)
CTT	Klasična teorija testa (eng. *Classical Test Theory*)
FSS	Protočno izvršavanje na nizu mašina (eng. *Flow-Shop Sequencing*)
GA	Genetski algoritam (eng. *Genetic Algorithm*)
ICF	Karakteristična funkcija stavke (eng. *Item Characteristic Function*)
IIF	Informaciona funkcija stavke (eng. *Item Information Function*)
ILP	Celobrojno linearno programiranje (eng. *Integer Linear Programming*)
IRT	Teorija odgovora na stavku (eng. Item Response Theory)
JSSP	Problem rasporeda poslova (eng. *Job-Shop Schedulig Problem*)
KNN	*k*-najbližih suseda (eng. *k-Nearest Neighbours*)
KP	Problem ranca (eng. *Knapsack Problem*)
MA	Maksimin model (eng. *Maximin Model*)
MCP	Problem maksimalne klike (eng. *Maximum Clique Problem*)
MI	Minimaks model (eng. *Minimax Model*)
MRSM	Metod nasumičnog uparivanja podtestova (eng. *Matched Random Subtests Method*)
MSPP	Problem maksimalnog pakovanja skupa (eng. *Maximum Set Packing Problem*)
NEH	Nawaz, Enscore i Ham
NFP	Problem mrežnog protoka (eng. *Network Flow Problem*)
NP	Nedeterministički u polinomijalnom vremenu (eng. *Nedeterministic Polynomial*)
SA	Simulirano kaljenje (eng. *Simulated Annealing*)
SEM	Standardna greška merenja (eng. *Standard Error of Measurement*)
SST	Simultano sastavljanje testova
PFSP	Protočno izvršavanje na nizu mašina (eng. *Permutation Flow-Shop Sequencing Problem*)
PPT	Testovi na papiru (eng. *Paper-and-Pencil Tests*)
PSO	Optimizacija rojem čestica (eng. *Particle Swarm Optimization*)
RST	Redno sastavljanje testova
TCF	Karakteristična funkcija stavke (eng. *Test Characteristic Function*)
TIF	Informaciona funkcija testa (eng. *Test Information Function*)
TS	Pretraga sa tabuima (eng. *Tabu Search*)
TTIF	Ciljna informaciona funkcija testa (eng. *Target Test Information Function*)
VNS	Pretraga promenljivih okolina (eng. *Variable Neighbourhood Search*)
WADM	Model ponderisanog apsolutnog odstupanja (eng. *Weighted Absolute Deviation Model*)
WDM	Model ponderisanog odstupanja (eng. *Weighted Deviation Model*)

Spisak slika

Slika 2.1 Karakteristična funkcija stavke 8
Slika 2.2 Informaciona funkcija stavke 10
Slika 2.3 Informaciona funkcija testa 10
Slika 2.4 Informaciona funkcija dva testa 11
Slika 2.5 NP složenost 14
Slika 3.1 Klasifikacija formulacija problema ASPT prema cilju 22
Slika 3.2 Klasifikacija pristupa za rešavanje problema ASPT 39
Slika 3.3 Primer pet poslova i tri mašine 56
Slika 3.4 Primer rada NEH heuristike za pet poslova i tri mašine 58
Slika 4.1 Primer sa pet setova i tri stavke po setu 60
Slika 4.2 Primer sa 12 skupova i 3 stavke po skupu 61
Slika 5.1 Poređenje vremena (a) i kvaliteta izvršavanja (b) NEHTA algoritma sa SA algoritmom 67
Slika 5.2 Poređenje kvaliteta NEHTA i VNS algoritma 69

Spisak tabela

Tabela 3.1 Značenje simbola 21
Tabela 3.2 Pregled formulacija problema sa ciljevima klasične teorije testova 37
Tabela 3.3 Pregled metoda za rešavanje problema ASPT u literaturi 53
Tabela 3.4 Unakrsna tabela formulacija i heurističkih metoda za rešavanje problema ASPT 54
Tabela 5.1 Poređenje NEHTA algoritma sa SA algoritmom 66
Tabela 5.2 Poređenje NEHTA algoritma sa VNS algoritmim 68
Tabela 5.3 Poređenje NEHTA algoritma sa VNS algoritmom kada se VNS algoritam ne prekida.. 70
Tabela 5.4 Poređenje NEHTA* algoritma sa NEHTA algoritmom 72

Spisak pseudokoda

Pseudokod 3.1 NEH algoritam 56
Pseudokod 4.1 NEHTA algoritam 62
Pseudokod 4.2 Opis procedure insertion (j, s, A) 62
Pseudokod 4.3 Opis funkcije calcMax(l, s, A) 63

1 Uvod

Paralelne test forme (paralelni testovi) su testovi sposobnosti koji se sastoje iz različitih stavki, ali poseduju jednake merne osobine. Testovima sposobnosti se smatraju, pored testova znanja kojima se bavi ova disertacija, i razne vrste testova ličnosti, kakvi se sreću u psihologiji. Korišćenje paralelnih testova u slučajevima testiranja većeg broja ispitanika je važno da bi se očuvala sigurnost sprovođenja testa, a time i verodostojnost rezultata testiranja, bez obzira da li se testiranje ispitanika sprovodi u isto ili u različito vreme. Jednake merne osobine paralelnih testova obezbeđuju da rezultati testa budu jednaki za ispitanike jednakih sposobnosti, bez obzira koja forma testa se rešava. Paralelni testovi se teško sastavljaju manuelno zato što takav postupak zahteva značajan napor i utrošeno vreme visoko obrazovanog kadra. Takođe, manuelno sastavljeni testovi su podložni subjektivnosti sastavljača testa, što negativno utiče na valjanost testa. Iz tog razloga, automatizacija procesa sastavljanja paralelnih testova je od velikog značaja, a u mnogim situacijama i neophodna.

U radu (Luecht & Sireci, 2011) dat je pregled više modela za testiranje uz pomoć računara (eng. *Computer Based Testing*, CBT). Glavna podela predloženih modela jeste na testove sa fiksnom formom (eng. *Computerized Fixed Tests*, CFT), adaptivne testove (eng. *Computerized Adaptive Tests*, CAT) i višestepene testove (eng. *Computerized Multistage Tests*, CMT). CFT je kategorija testova koja ima istu formu ili više test formi, ali su stavke na testu određene pre testiranja ispitanika i nepromenljive. Takvi testovi su analogni testovima na papiru (eng. *Paper-and-Pencil Tests*, PPT). CAT testovi se prilagođavaju ispitaniku na način da mu se nudi da odgovara na jednu po jednu stavku, dok se na osnovu rezultata odgovora na prethodne stavke bira sledeća stavka. MST testovi su slični CAT testovima, sa razlikom što se ispitaniku daje da odgovara na više nepromenljivih stavki u jednom trenutku, za razliku od CAT testova gde se odgovara serijalizovano, na jednu po jednu stavku. CAT testovi su manje pogodni za ocenjivanje znanja zbog nemogućnosti kontrole testa kao celine i zbog nemogućnosti da se ispitanik vrati na prethodna pitanja i izmeni odgovore. Ove mane CAT testova su ujedno i glavne prednosti CFT testova, zbog čega se CFT testovi najviše koriste za sertifikaciju i ocenjivanje znanja, dok su CAT testovi pogodniji za samotestiranje. Ova disertacija se bavi isključivo CFT kategorijom testova, kojoj pripadaju paralelni testovi.

Početni korak u automatizaciji procesa sastavljanja testa jeste kreiranje banke stavki. Banka stavki je skup stavki (skup pitanja, odnosno zadataka različitih vrsta) koje su smeštene na računar zajedno sa svojim atributima. Atributi stavke, kao i testa koji je podskup stavki iz banke stavki, mogu biti psihometrijski (statistički) i ne-psihometrijski. Psihometrijski atributi stavki i testa se najčešće određuju na način definisan izabranom psihometrijskom teorijom testa, kao što su klasična teorija testa (eng. *Classical Test Theory*, CTT) (Spearman, 1904) i teorija odgovora na stavku (eng. *Item Response Theory*, IRT) (Lord & Novick, 1968). Navedene dve teorije testa se najčešće koriste, dok postoje i druge kao što su teorija generalizabilnosti (Shavelson, Webb, & Rowley, 1989) i linearna regresija (Smits & Finkelman, 2014), kojima se ne bavi ova disertacija. Primeri psihometrijskih atributa su težina stavke, diskriminativnost stavke, težina testa i pouzdanost testa. Primeri ne-psihometrijskih atributa su: pripadnost stavke određenoj kategoriji (oblasti, tematskoj celini, eng. *topic*), tip stavke određen načinom odgovaranja na stavku (na primer, višestruki odgovori od kojih može biti tačan jedan ili više, spajanje pojmova, kratak tekstualni ili numerički odgovor), broj stavki na testu, vreme predviđeno za rešavanje stavke ili testa.

Osamdesetih godina dvadesetog veka su računari šire uvedeni u praksu testiranja. Stavke su smeštane u banke stavki, što je motivisalo i razvoj ranih algoritama za automatsko sastavljanje testa. Odabir stavki za test iz banke stavki, na osnovu njihovih atributa i specifikacije testa, korišćenjem

računara je poznato kao automatsko sastavljanje testa (AST) (Yen, 1983) i (van der Linden W. J., 1986).

Specifikacija testa predstavlja šemu (eng. *blueprint*) testa kojim se definišu željene vrednosti atributa testa (van der Linden W. J., 2005). Specifikacija testa se sastoji iz skupa ograničenja i funkcije cilja. Ograničenja osiguravaju da atributi stavki i/ili testa budu u okvirima predviđenih graničnih vrednosti, dok funkcija cilja predstavlja integralni atribut testa čija vrednost treba da bude maksimalna (ili minimalna, što se jednako tretira) za banku stavki koja je na raspolaganju. Najčešće funkcija cilja predstavlja meru preciznost testa, koja treba da bude maksimalna, odnosno greška merenja da bude minimalna, ali postoje i druge funkcije cilja koje će biti predstavljene u trećem poglavlju.

Automatsko sastavljanje paralelnih testova (ASPT) podrazumeva da se u postupku kreira više testova. Testovi se mogu kreirati jedan za drugim, i taj postupak se naziva redno (sekvencijalno) sastavljanje testova (RST), ili u isto vreme, kada se postupak naziva simultano sastavljanje testova (SST). Bez obzira da li je postupak sekvencijalni ili simultani, testovi dele istu banku stavki i specifikaciju testa. Sastavljanje paralelnih testova je značajno složeniji proces od sastavljanja jednog testa (jedinstvene forme za sve ispitanike). Složenost procesa ASPT, u poređenju sa procesom sastavljanja jedinstvenog testa (Boekkooi-Timminga, 1987) i (Chang & Shiu, 2012), dovela je do razvoja velikog broja različitih metoda za rešavanje ovog problema.

Problem ASPT se može proučavati po analogiji sa problemima kombinatorne optimizacije kod kojih je cilj da se pronađe optimalan podskup objekata, iz konačnog skupa objekata, koji zadovoljava određene uslove (Papadimitriou & Steiglitz, 1982). U slučaju problema ASPT, to implicira da se pronalazi optimalan podskup stavki, iz konačne banke stavki, takav da je zadovoljena specifikacija testa.

Optimalan podskup stavki predstavlja optimalno rešenje problema. Optimalno rešenje problema bi se moglo pronaći sastavljanjem svih mogućih kombinacija stavki koje zadovoljavaju skup ograničenja (dopustiva rešenja), zatim poređenjem njihovih vrednosti funkcije cilja, i na kraju izborom onog podskupa stavki koji ima najbolju vrednost funkcije cilja. Problem u ovakvom načinu traženja rešenja jeste u tome što broj mogućih kombinacija stavki eksponencijalno raste sa veličinom banke stavki. Ova činjenica utiče na vremensku kompleksnost algoritma, na način da rešenje nije moguće naći u realnom vremenu koristeći savremene računare, za realistične veličine banke stavki (Garey & Johnson, 1979) i (Hoos & Stutzle, 2005).

U početku razvoja metoda ASPT su se koristili softveri opšte namene za rešavanje optimizacionih problema (solveri). Međutim, kako se povećavala banka stavki i kako su se zahtevi u specifikaciji testa usložnjavali, korišćenje (meta)heuristika (Hussain, Salleh, Cheng, & Shi, 2019) je postao široko prihvaćeni pristup. Heuristike sužavaju prostore pretrage i daju suboptimalna rešenja uz značajno kraće vreme izvršavanja. Termin *metaheuristika* se prvi put pojavio u radu (Glover, 1986), međutim do sada ne postoji generalno prihvaćena definicija termina. U radu (Blum & Roli, 2003) su opisane neke karakteristike metaheuristika: metaheuristike su strategije za pronalaženje rešenja, opisi metaheuristika mogu biti na visokom nivou apstrakcije i metaheuristike nisu specifične za problem koji se rešava. U ovoj disertaciji će se ovaj termin odnositi na algoritamski okvir (eng. *framework*) rešenja koji služi za pravljenje konkretnih heuristika.

Heuristike koje se koriste za rešavanje problema kombinatorne optimizacije se mogu podeliti u dve klase: heuristike poboljšanja i konstruktivne heuristike. Kod heuristika poboljšanja se dopustivo rešenje, ili skup dopustivih rešenja, inicijalno generiše i iterativno poboljšava tražeći rešenje u definisanoj okolini (skup dopustivih rešenja) trenutnog rešenja. Kod ovih heuristika, da bi se izbegao lokalni minimum, koristi se koncept diverzifikacije, čime se okolina za pretragu rešenja menja. Heuristike poboljšanja, po svojoj prirodi, imaju eksponencijalnu složenost, pa je iz tog razloga neophodno uvesti kriterijum za prekidanje postupka. Moguće je ograničiti vreme izvršavanja algoritma, ukupan broj iteracija, broj iteracija bez poboljšanja vrednosti funkcije cilja ili

definisati željenu vrednost funkcije cilja po čijem dostizanju se može prekinuti izvršavanje algoritma (Gendreau & Potvin, 2010). Druga vrsta heuristika, konstruktivne heuristike, grade rešenje dodajući jedan po jedan element rešenja (objekat), sve dok se kompletno rešenje ne formira. One daju suboptimalno rešenje brže, ali je kvalitet rešenja često lošiji u poređenju sa heuristikama poboljšanja, pa su i manje zastupljene (Blum & Roli, 2003). Jedan od izuzetaka je Nawaz, Enscore i Ham (NEH) heuristika (Nawaz, Enscore Jr, & Ham, 1983), konstruktivna heuristika koja rešava problem određivanja redosleda poslova u protočnom izvršavanju na nizu mašina (eng. *flow-shop sequencing*, FSS), čije su prednosti nad mnogim heuristikama poboljšanja dokazane (Ruiz & Maroto, 2005), (Danilović & Ilić, 2015) i (Rossi, Nagano, & Neto, 2016).

Pojedini istraživači su pokazali izomorfizam između problema ASPT i već poznatih problema kombinatorne optimizacije, kao što su *Problem ranca* (eng. *Knapsack Problem*, KP) (Theunissen, 1985), *Problem pakovanja u korpe* (eng. *Bin Packing Problem*, BPP) (Garey, Graham, Johnson, & Yao, 1976), *Problem maksimalne klike* (eng. *Maximum Clique Problem*, MCP) (Ishii, Songmuang, & Ueno, 2014), *Problem maksimalnog pakovanja skupa* (eng. *Maximum Set Packing Problem*, MSPP) (Belov & Armstrong, 2006). Navedeni problemi su rešavani heuristikama koje se mogu prilagoditi i koristiti za rešavanje problema ASPT. Većina predloženih heuristika su heuristike poboljšanja. Primeri tih (meta)heuristika su pretraga promenljivih okolina (eng. *Variable Neighbourhood Search*, VNS) uvedena od (Mladenović & Hansen, 1997) i korišćena za ASPT u (Pereira & Vila, 2015), pretraga sa tabuima (eng. *Tabu Search*, TS) uvedena od (Glover, 1989) i korišćena za ASPT u (Hwang, Chu, Yin, & Lin, 2008), simulirano kaljenje (eng. *Simulated Annealing*, SA) uvedeno od (Kirkpatrick, Gelatt, & Vecchi, 1983) i korišćeno za ASPT u (Brusco, Köhn, & Steinley, 2013), genetski algoritam (eng. *Genetic Algorithm*, GA) uveden od (Holland, 1973) i korišćen za ASPT u (Sun, Chen, Tsai, & Cheng, 2008), optimizacija rojem čestica (eng. *Particle Swarm Optimization*, PSO) uvedena od (Eberhart & Kennedy, 1995) i korišćena za ASPT u (Yin, Chang, Hwang, Hwang, & Chan, 2006), (Ho, Yin, Hwang, Shyu, & Yean, 2009) i (Lin, Jiang, Gong, Zhan, & Zhang, 2019).

Predmet ove disertacije jeste pregled, analiza i klasifikacija postojećih formulacija problema ASPT i heurističkih rešenja tog problema, kao i predlog novog metoda za rešavanje izabrane formulacije problema ASPT.

Problem sastavljanja paralelnih testova, koji su poželjni u testiranju sposobnosti i/ili znanja, za sticanje licenci, postoji skoro četrdeset godina. Iako je problem već sazreo, mnogo različitih metoda se pojavilo u poslednjih dvadeset godina. Jedan od razloga za različitost pristupa jeste preklapanje različitih naučnih polja, kao što su psihometrija, primenjena matematika, operaciona istraživanja, veštačka inteligencija. Takva situacija je proizvela mešavinu koncepata i prirode doprinosa, koji potiču iz različitih oblasti, u predloženim rešenjima. Jedan od motiva za istraživanje iz ove disertacije jeste da se sistematizacijom ovih razlika i klasifikacijom istraživačkih pristupa pomogne istraživačima i praktičarima zainteresovanim za sprovođenje istraživanja i razvoja u domenu ASPT, da bolje pozicioniraju svoje mesto delovanja.

Motivacija istraživanja urađenih kroz ovu disertaciju je i da se zadovolje skupovi zahteva u uslovima koji uglavnom odgovaraju obrazovnim institucijama, od osnovne škole do visokoškolskih ustanova. Karakteristike tih zahteva su relativno male grupe ispitanika i, kao posledica, relativno mali broj paralelnih testova (od nekoliko desetina do nekoliko stotina) koji se sastavljaju iz banki stavki umerene veličine (od nekoliko stotina do nekoliko hiljada stavki), uz često testiranje ispitanika ograničenim računarskim resursima. Kao posledica, tražen je jednostavan i brz algoritam, čime se sprečava preopterećenje računarskih resursa koji se koriste za ASPT i administraciju testa.

Polazne hipoteze istraživanja su bile sledeće:

1. Iako se ASPT istražuje u različitim, gore navedenim, disciplinama, uz korišćenje raznorodne terminologije po disciplinama, moguće je na jedinstven način predstaviti različite formulacije problema i različita heuristička rešenja, koristeći ujednačenu terminologiju.

2. Moguće je odvojeno klasifikovati postojeće formulacije problema ASPT i heurističke metode za njihovo rešavanje.
3. Moguće je ukrštanjem klasifikacije formulacija problema ASPT i heurističkih metoda za njihovo rešavanje, te sagledanjem postojećih pristupa u takvom dvodimenzionalnom prostoru klasifikacije, ukazati na moguće pravce daljih istraživanja ASPT metoda.
4. Moguće je primeniti konstruktivnu heuristiku za potpuno i efikasno rešavanje problema ASPT, tako da se dobiju kvalitetni rezultati u procenjivom vremenu.
5. Rezultati primene konstruktivne heuristike su poredivi sa rezultatima postojećih metoda zasnovanih na heuristikama poboljšanja, a u određenim situacijama (posebno onim koji nalaze primenu u obrazovnim institucijama) daju bolji kvalitet ili postižu veću efikasnost.

Disertacija ima za primarni cilj predlog novog metoda zasnovanog na poznatoj konstruktivnoj heuristici, koja nije ranije primenjivana za rešavanje ni jedne formulacije problema sastavljanja paralelnih testova, a koji će biti dobro prilagođen ASPT u obrazovnim institucijama i davati, u datim uslovima, kvalitetnija i/ili efikasnija rešenja od poredivih postojećih metoda. Sekundarni cilj disertacije je odvojen prikaz, analiza i klasifikacija formulacija problema ASPT i heuristika koje su se koristile za njihovo rešavanje, kao i unakrsni prikaz formulacija problema i heurističkih rešenja, koji daje određene putokaze daljim istraživanjima.

Primarni cilj disertacije se ostvaruje kroz predlog novog metoda, kojim se postiže poboljšanje kvaliteta sastavljenih paralelnih testova i/ili ubrzanje procesa njihovog sastavljanja, čime se postiže ujednačena procena znanja i/ili ušteda resursa. Od posebnog značaja je moguća procena vremena izvršavanja predloženog algoritma koji je po strukturi konstruktivna heuristika, za razliku od heuristika poboljšanja, a koje primenjuje najveći broj postojećih metoda.

Sekundarni cilj disertacije se ostvaruje kroz poseban prikaz, analizu i klasifikaciju formulacija problema ASPT i heuristika koje se koriste za njegovo rešavanje. Korišćena je ujednačena terminologija pri formulaciji matematičkih modela i predložena je klasifikacija formulacija na osnovu funkcije cilja. Funkcije cilja koje su analizirane su maksimizacija kvaliteta testa merenog metrikama popularnih psihometrijskih teorija, maksimizacija broja test formi i minimizacija rastojanja od testa-klice (postojeći test koji ima sve tražene karakteristike, psihometrijske i nepsihometrijske). Sa druge strane, heuristička rešenja problema su analizirana i klasifikovana odvojeno u odnosu na formulacije problema. Rešenja su prikazana koristeći istu strukturu opisa i na ujednačen način. Ukrštanjem analiziranih formulacija i predloženih heuristika se ukazuje na otvoren prostor za istraživanje kroz nove kombinacije formulacija problema i heuristika za njihovo rešavanje.

Struktura disertacije je sledeća. U poglavlju 2 se definiše problem koji se rešava. Prvo se opisuju najvažniji koncepti teorija testiranja koje se koriste za određivanje atributa stavki i testa. Zatim se analizira problem ASPT kao problem kombinatorne optimizacije. Na kraju poglavlja se detaljno opisuje problem ASPT koji se rešava u disertaciji. Poglavlje 3 daje pregled postojećih metoda u literaturi koji su se bavili problemom ASPT. Prvo su analizirane i klasifikovane formulacije problema ASPT. Zatim su analizirana i klasifikovana postojeća heuristička rešenja. Na kraju ovog poglavlja se detaljno opisuje NEH heuristika na kojoj je zasnovan predložen novi metod. U poglavlju 4 se opisuje novi metod za rešavanje problema ASPT. Rezultati koji se postižu predloženim metodom su predstavljeni u poglavlju 5. Poglavlje 6 je zaključak u kojem se sumiraju rezultati sprovedenog istraživanja i daju smernice za nova istraživanja.

2 Postavka problema

U ovoj glavi se prvo ukratko opisuju teorije testa, čiji je odabir praktično uvek prvi korak u procesu sastavljanja testova. Primenjena teorija testa određuje način merenja psihometrijskih atributa stavki koje se smeštaju u banku stavki, kao i kvaliteta sastavljenog testa. Nakon toga se detaljno opisuje problem ASPT, koji se proučava u kontekstu problema kombinatorne optimizacije. Na kraju se detaljno razmatra problem rešavan u ovoj disertaciji.

2.1 Teorije testa

Teorije testa su matematički modeli koji služe za analizu rezultata testa sa ciljem procene njihovih karakteristika. Analizom se otkrivaju odnosi između ostvarenih rezultata testa i merene sposobnosti ispitanika. Najčešće se procenjuje pouzdanost testa, greška merenja, ili količina informacije koju rezultat testa daje o merenoj sposobnosti. Modeli su zasnovani na aksiomatici teorije verovatnoće. Svaka od ovih teorija uključuje i određene pretpostavke (Lord & Novick, 1968), koje će biti navedene u prikazu pojedine teorije testa u nastavku ovog odeljka. Dve najčešće korišćene teorije testa u psihometriji su klasična teorija testa i teorija odgovora na stavku.

2.1.1 Klasična teorija testa

U klasičnoj teoriji testa, težište je na rezultatu testa kao celine. Smatra se da je pretpostavka da svaki rezultat merenja sposobnosti sadrži i slučajnu grešku merenja, označila početak klasične teorije testa (Spearman, 1904). Na osnovu te pretpostavke se može reći da se izmeren rezultat testa (X) sastoji iz dve komponente: tačnog rezultata (T) i greške (E).

$$X = T + E \tag{1}$$

Tačan rezultat testa bi, u idealnom slučaju, mogao da se dobije kada bi osoba rešavala test beskonačan broj puta. Tada bi tačan rezultat testa bila srednja vrednost rezultata svih rešavanja testa. Pošto nije moguće da osoba rešava isti test više puta, a da na to ne utiče pamćenje prethodnih rešavanja i/ili neki spoljni faktori (uslovi testiranja), izmeren rezultat testa (nadalje rezultat testa) koji je potpuno tačan ($X = T$) je samo hipotetički moguć.

Pretpostavka je i da je slučajna greška normalne raspodele, zbog čega je očekivana vrednost greške (srednja vrednost greške prilikom teorijski beskonačnog broja rešavanja testa) jednaka nuli. Sledeća pretpostavka jeste da su greške nekorelisane između sebe, kao i da je slučajna greška nekorelisana sa tačnim rezultatom testa. Sve navedene pretpostavke su dokazane Teoremom 2.7.1 u (Lord & Novick, 1968).

Psihometričari su pokazali da teorija tačnog rezultata i greške, formula (1) i dodatne pretpostavke navedene u prethodnom pasusu, koja važi za osobu koja rešava isti test više puta, može da se primeni za slučaj rešavanja testa od strane više osoba (Allen & Yen, 1979). Ta činjenica je važna zato što omogućava da se karakteristike testa procene na osnovu testa koji jednokratno rešava grupa ispitanika.

Veoma važan atribut testa jeste koeficijent pouzdanosti njegovog rezultata. Koeficijent pouzdanosti rezultata testa (nadalje pouzdanost testa) se definiše kao kvadrat korelacionog koeficijenta između rezultata testa i tačnog rezultata testa (ρ_{XT}^2).

$$\rho_{XT}^2 = \frac{\sigma_{XT}^2}{\sigma_X^2 \sigma_T^2} \tag{2}$$

gde je σ_{XT} kovarijansa između rezultata testa i tačnog rezultata testa, σ_X^2 varijansa rezultata testa, σ_T^2 varijansa tačnog rezultata testa, gde su σ_X, odnosno σ_T, standardne devijacije (odstupanja) rezultata testa, odnosno tačnog rezultata testa, respektivno, za grupu ispitanika.

Pošto važi pretpostavka da tačan rezultat testa i greška nisu u korelaciji za bilo koju grupu ispitanika ($\sigma_{ET} = 0$), važi sledeća jednačina:

$$\sigma_{XT} = \sigma_{(T+E)T} = \sigma_T^2 + \sigma_{ET} = \sigma_T^2 \tag{3}$$

Prilikom izvođenja formule koristila se činjenica da ukoliko su dve promenljive nezavisne, onda je varijansa zbira te dve promenljive jednaka zbiru varijanse svake promenljive ponaosob. Koristeći pretpostavku da tačan rezultat i greška merenja nisu u korelaciji dobijamo i sledeće jednakosti:

$$X = T + E \rightarrow \sigma_X^2 = \sigma_T^2 + \sigma_E^2 \rightarrow \sigma_T^2 = \sigma_X^2 - \sigma_E^2 \tag{4}$$

Pouzdanost testa se, na osnovu formula (3) i (4), može prikazati na sledeći način:

$$\rho_{XT}^2 = \frac{\sigma_{XT}^2}{\sigma_X^2 \sigma_T^2} = \frac{\sigma_T^2}{\sigma_X^2} \tag{5}$$

Pošto je varijansa rezultata testa poznata, a varijansa greške se može proceniti, što će biti objašnjeno u nastavku, pouzdanost se može prikazati kao funkcija varijanse rezultata testa (σ_X^2) i varijanse greške (σ_E^2).

$$\rho_{XT}^2 = \frac{\sigma_T^2}{\sigma_X^2} = \frac{\sigma_X^2 - \sigma_E^2}{\sigma_X^2} = 1 - \frac{\sigma_E^2}{\sigma_X^2} \tag{6}$$

Pouzdanost testa se meri na skali od 0 do 1. Ukoliko je njegova vrednost 0, to znači da je varijansa rezultata testa jednaka varijansi greške, odnosno da je rezultat testa isključivo posledica greške merenja, dok vrednost 1 znači da je rezultat testa tačan, odnosno da greške nema. Standardna devijacija greške (σ_E) se naziva i standardnom greškom merenja (eng. *Standard Error of Measurement*, SEM). Pouzdanost se može proceniti i kao mera interne konzistencije, što će biti objašnjeno u nastavku odeljka.

Pošto se test sastoji od stavki, neophodno je definisati atribute stavki merene po CTT. Težina stavke (eng. *difficulty*, p_i) za dihotomno ocenjene stavke (0 poena za netačan odgovor, 1 za tačan odgovor) se definiše kao odnos ispitanika koji su tačno odgovorili na stavku (N_i) i ukupnog broja ispitanika (N):

$$p_i = \frac{N_i}{N} \tag{7}$$

Što je više ispitanika tačno odgovorilo na stavku to je stavka lakša, odnosno numerička vrednost težine je veća i obratno (što je paradoksalno, odnosno pokazuje da uobičajeni termin „težina stavke“ nije srećno izabran, jer se u stvari datim atributom izražava „lakoća“ rešavanja stavke). Ukoliko bi sve stavke u testu bile prelake (na primer $p = 0.9$) ili preteške (na primer $p = 0.1$), većina odgovora bi bila jednaka (tačno za $p = 0.9$ ili netačno za $p = 0.1$) što znači da bi rezultati testa bili približno jednaki. Da bi se maksimizovala varijansa rezultata testa (σ_X^2), a samim tim i pouzdanost testa, definisana formulom (6), preporuka je da težine stavke budu u opsegu [0.4, 0.6] (Krishnan, 2013).

Diskriminativnost stavke (eng. *discrimination index*, d_i) određuje u kojoj meri stavka može da razdvoji nivoe znanja. Jedan od načina procene diskriminativnosti stavke, prema CTT, jeste koeficijent korelacije između odgovora na stavku i rezultata testa kao celine:

$$d_i = \frac{\sigma_{iX}}{\sigma_i \sigma_X} \tag{8}$$

gde je σ_{iX} kovarijansa između odgovora na stavku i i rezultata testa, σ_i je standardna devijacija odgovora na stavku i, a σ_X je standardna devijacija rezultata testa. Velika vrednost diskriminativnosti stavke implicira da stavka dobro razdvaja osobe sa visokim i niskim rezultatom testa (van der Linden W. J., 2005).

Kronbahova alfa (α) je mera interne konzistencije testa, koja govori o tome koliko stavke na testu slično mere određenu sposobnost ispitanika (Cronbach, 1951). Ona se smatra i donjom granicom koeficijenta pouzdanosti (Teorema 4.4.3 u (Lord & Novick, 1968)). Za njeno izračunavanje je dovoljno da test jednokratno reši grupa ispitanika. Njena formula je:

$$\alpha = \frac{n}{n-1}\left[1 - \frac{\sum_{i=1}^{n} \sigma_i^2}{\sigma_X^2}\right] \tag{9}$$

gde je n ukupan broj stavki na testu, σ_i^2 je varijansa rezultata odgovora na stavku i, dok je σ_X^2 varijansa rezultata testa.

Pošto standardna devijacija rezultata testa može da se prikaže u funkciji parametara stavke (Formula 15.3.5, (Lord & Novick, 1968)) na sledeći način:

$$\sigma_X = \sum_{i=1}^{n} \sigma_i d_i \tag{10}$$

Koeficijent α sada može da se prikaže kao funkcija parametara stavke:

$$\alpha = \frac{n}{n-1}\left[1 - \frac{\sum_{i=1}^{n} \sigma_i^2}{(\sum_{i=1}^{n} \sigma_i d_i)^2}\right] \tag{11}$$

Kuder i Ričardson su uveli specijalni slučaj koeficijenta alfa kada su stavke na testu dihotomne i imaju vrednost jedan sa verovatnoćom p_i a vrednost nula sa verovatnoćom (1 - p_i) (Kuder & Richardson, 1937). Ta formula je poznata kao Kuder-Ričardsonova formula 20 (KR20):

$$\alpha_{KR20} = \frac{n}{n-1}\left[1 - \frac{\sum_{i=1}^{n} p_i(1-p_i)}{\sigma_X^2}\right] \tag{12}$$

Ukoliko stavke na testu imaju jednake težine (p) tada formula (12) postaje formula pod nazivom Kuder Ričardsonova formula 21 (KR21):

$$\alpha_{KR21} = \frac{n}{n-1}\left[1 - \frac{np(1-p)}{\sigma_X^2}\right] \tag{13}$$

Paralelne forme testa, po CTT teoriji, treba da imaju što približnije aritmetičke sredine i varijanse rezultata. Ako su ti zahtevi ispunjeni, može se dokazati da je korelacija između dve paralelne forme jednaka međusobno jednakim koeficijentima pouzdanosti tih formi (Lord & Novick, 1968), koji se prilikom sastavljanja testova najčešće maksimizuju.

2.1.2 Teorija odgovora na stavku

Počeci teorije odgovora na stavku se mogu naći u radovima (Lawley, 1943) i (Lord, 1952), između ostalih. Teorija odgovora na stavku podrazumeva da svaki ispitanik poseduje latentnu sposobnost (na primer, znanje, θ), kao i da može da odgovori tačno na stavku i sa određenom verovatnoćom P_i. Po održanom testu se za svaku stavku može dobiti grafik distribucije verovatnoće tačnog odgovora $P_i(\theta)$. Vrednost kojom se meri sposobnost (θ) predstavlja se na horizontalnoj osi (osi sposobnosti), dok se verovatnoća posedovanja odgovarajuće vrednosti sposobnosti ($P_i(\theta)$) predstavlja na vertikalnoj osi (osi verovatanoće).

Verovatnoća tačnog odgovora na stavku se često modeluje logističkom funkcijom sa tri parametra (postoje jedno-, dvo-, tro- i četvoro-parametarski modeli), koja se još naziva i karakterističnom funkcijom stavke (eng. *Item Characteristic Function,* ICF):

$$P_i(\theta) = c_i + (1 - c_i)\frac{1}{1 + e^{-a_i(\theta - b_i)}} \tag{14}$$

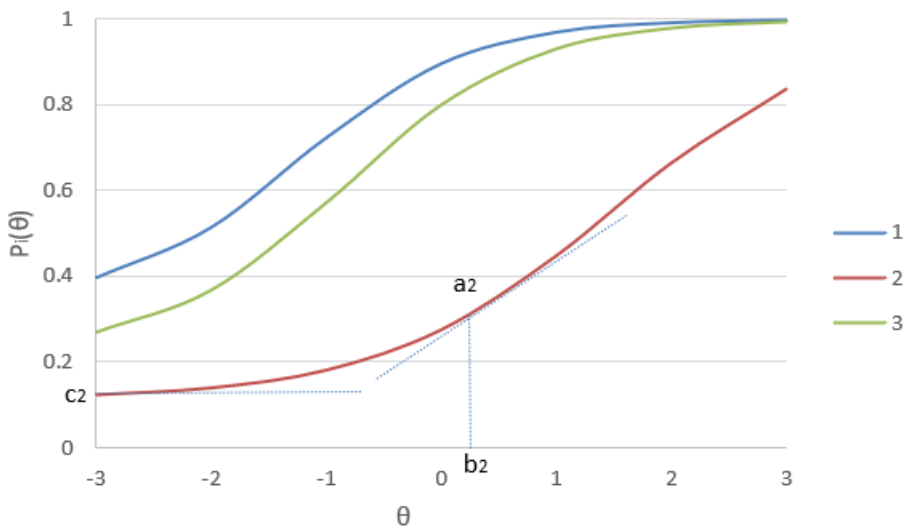
gde je c_i verovatnoća tačnog odgovora slučajnim pogađanjem, koja predstavlja minimalnu verovatnoću tačnog odgovora, b_i tačka na osi sposobnosti u kojoj je verovatnoća tačnog odgovora $(c_i + 1)/2$, dok je a_i nagib krive za vrednost sposobnosti b_i (Slika 2.1).

Model sa dva parametra se dobija iz formule (14), kada se uzme da je $c_i = 0$.

$$P_i(\theta) = \frac{1}{1 + e^{-a_i(\theta - b_i)}} \tag{15}$$

Četvrti parametar, koji označava maksimalnu verovatnoću tačnog odgovora ispitanika, retko se uključuje u model. Smatra se da je, u graničnom slučaju, moguće i da svi ispitanici daju tačan odgovor, te ovaj parametar ima vrednost 1.

Vrednost sposobnosti θ je u opsegu [-∞, ∞]. U praksi, od interesa je samo konačan broj diskretnih vrednosti θ_k, k = 1, 2,..., K koje se razmatraju, a odgovaraju ocenama sposobnosti. Pošto je parametar b_i vrednost na istoj skali, može da ima iste vrednosti kao i θ. Vrednosti parametra a_i, koji predstavlja tangens ugla koji zaklapa tangenta na krivu u prevojnoj tački, u kojoj je $\theta = b_i$, sa osom sposobnosti u pozitivnom smeru, su u opsegu [0, ∞], jer je ugao u opsegu [0, π/2].



Slika 2.1 - Karakteristična funkcija stavke

Atributi stavke, težina i diskriminativnost, se definišu pomoću parametara karakteristične funkcije stavke. Parametar b_i predstavlja težinu stavke na skali sposobnosti. Kod lakših stavki je kriva pomerena ulevo. Teže stavke zahtevaju veću vrednost sposobnosti θ, pa je kriva pomerena udesno. Diskriminativnost stavke je njen atribut koji određuje koliko dobro stavka razdvaja ispitanike sa sposobnostima većim od b_i i ispitanike sa sposobnostima manjim od b_i. Diskriminativnost stavke jednaka je vrednosti parametra a_i, odnosno nagibu krive u tački u kojoj je sposobnost jednaka b_i. Što je veći nagib krive to stavka bolje razdvaja ispitanike. Idealnu diskriminativnost bi imala stavka sa stepenastom krivom, koja do tačke $\theta = b_i$ ima vrednost c_i, a od te tačke vrednost 1.

Pošto se test sastoji iz stavki i za svaku stavku je verovatnoća tačnog odgovora definisana karakterističnom funkcijom stavke, karakteristična funkcija testa (eng. *Test Characteristic Function,* TCF) predstavlja zbir verovatnoća tačnog odgovora na svaku od stavki (Lawley, 1943). Ova vrednost predstavlja i očekivani rezultat testa ispitanika sa sposobnošću θ :

$$TCF(\theta) = \sum_{i=1}^{n} P_i(\theta) \tag{16}$$

Začetnik koncepta merenja informacije je matematičar R.A. Fisher, koji je definisao količinu informacije kao meru informacije koju merena slučajna promenljiva X nosi o nepoznatom parametru θ distribucije verovatnoće ($P(\theta)$) promenljive X (Fisher, 1922). U knjizi (Lord & Novick, 1968) uveden je koncept informacione funkcije stavke (eng. *Item Information Function*, IIF) kao meru informacije o sposobnosti θ (nepoznati parametar) koju nosi verovatnoća $P_i(\theta)$ odgovora na stavku i za ispitanika sa sposobnošću θ. Za dihotomne stavke, IIF je jednaka:

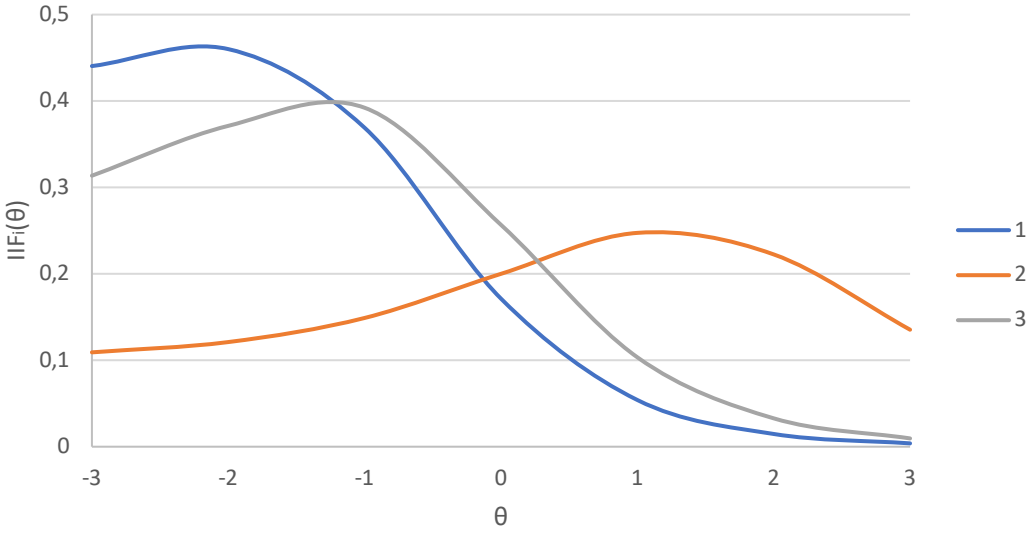
$$IIF_i(\theta) = \frac{(P_i'(\theta))^2}{P_i(\theta)(1 - P_i(\theta))} \tag{17}$$

Za tro-parametarski model karakteristične funkcije stavke (verovatnoće odgovora na stavku), ova vrednost je data sledećom formulom:

$$IIF_i(\theta) = a_i^2 \frac{1 - P_i(\theta)}{P_i(\theta)} \left(\frac{P_i(\theta) - c_i}{1 - c_i} \right)^2 \tag{18}$$

Kao što se zapaža u formuli (18), informaciona funkcija stavke direktno je proporcionalna kvadratu diskriminativnosti stavke. Što stavka bolje razgraničava ispitanike sa visokom od ispitanika sa niskom sposobnošću, to ona nosi veću informaciju u proceni sposobnosti.

Primeri informacionih funkcija tri različite stavke prikazuje Slika 2.2.

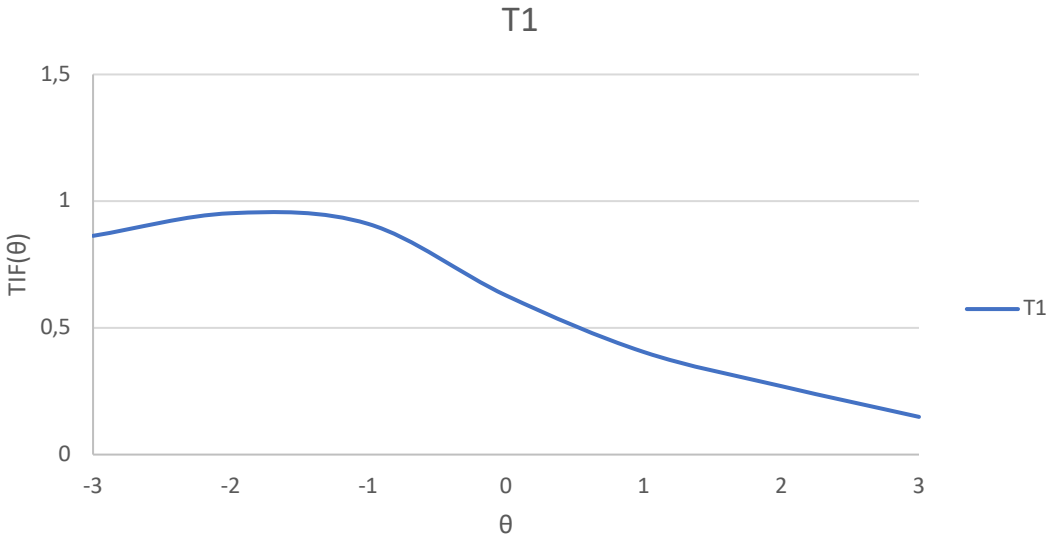


Slika 2.2 - Informaciona funkcija stavke

Informacija o sposobnosti ispitanika koju meri test u celini je suma individualnih informacionih funkcija stavki koje su odabrane za test i definisana je informacionom funkcijom testa (eng. *Test Information Function*, TIF) i (Lord & Novick, 1968).

$$TIF(\theta) = \sum_{i=1}^{n} IIF_i(\theta) \tag{19}$$

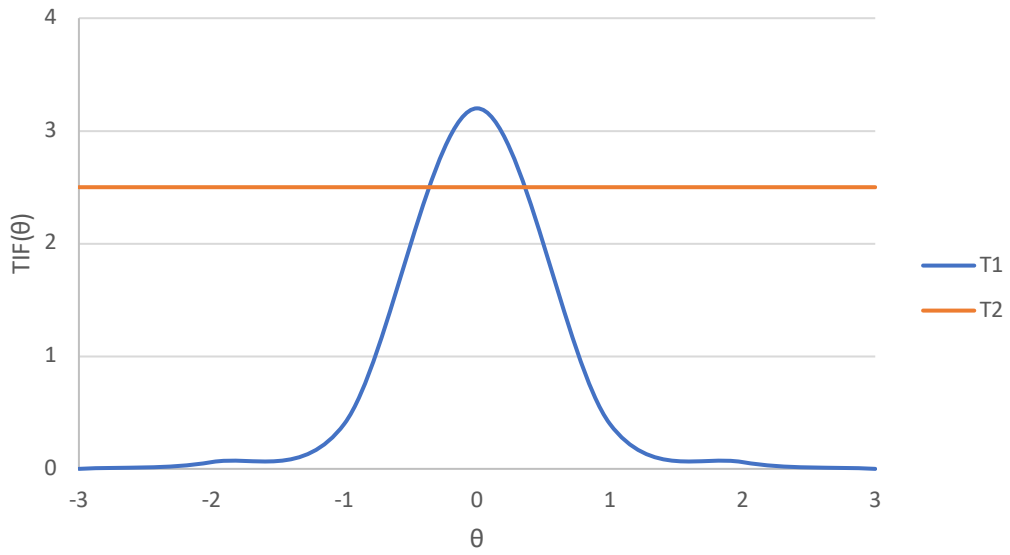
Slika 2.2 prikazuje primer informacionih funkcija (IIF) za tri stavke. Slika 2.3 prikazuje informacionu funkciju odgovarajućeg testa (TIF) koji sadrži samo te tri stavke.



Slika 2.3 - Informaciona funkcija testa

Cilj testiranja u IRT-u se najčešće definiše pomoću ciljne TIF (eng. *Target Test Information Function*, TTIF). Ukoliko se rade testovi gde je od interesa samo jedna vrednost sposobnosti θ, onda je TTIF funkcija uska zvonasta kriva koja ima maksimum u željenoj vrednosti sposobnosti

(Slika 2.4, T1), a ukoliko se sve vrednosti sposobnosti od jednakog značaja za testiranje, onda je vrednost TTIF funkcije jednaka za sve vrednosti sposobnosti θ (Slika 2.4, T2).



Slika 2.4 - Informaciona funkcija dva testa

Informaciona funkcija testa, po definiciji, je obrnuto srazmerna kvadratu standardne greške merenja (Lord, 1980). Da bi greška merenja bila minimalna, informaciona funkcija testa za datu sposobnost treba da bude maksimalna. Ukoliko se želi da test podjednako dobro meri sve vrednosti sposobnosti, TTIF ima istu vrednost za svako θ, te je cilj sastavljanja testa da ta vrednost bude maksimalna. Da bi sastavljeni testovi bili paralelni, njihove informacione funkcije treba da budu jednake, ili približno jednake.

Poredeći CTT i IRT, u radu (Hambleton & Swaminathan, 1984) daje se prednost IRT. U radu (Fan, 1998) je kroz izvođenje serije eksperimenata zaključeno da praksa ne podržava teoriju i da IRT nije superiorniji u odnosu na CTT u smislu procene osobina stavki i sposobnosti ispitanika. U radu (Macdonald & Paunonen, 2002) koristeći *Monte Carlo* tehniku, su potvrđeni nalazi iz rada Fana i zaključeno je da bez obzira na izbor teorije testa, zaključci o sposobnosti ispitanika će biti konzistenti i tačni. Prednost CTT teorije jeste u tome što je koncepcijski jednostavnija, dok je za sastavljača testa koji koristi IRT teoriju potrebno veće teorijsko znanje, neophodno za procenu TTIF (Lin C. J., 2008).

2.2 Problem ASPT kao problem kombinatorne optimizacije

Problem ASPT pripada klasi problema kombinatorne optimizacije, pa će na početku ovog odeljka biti objašnjeni osnovni pojmovi u teoriji kombinatorne optimizacije. Problem ASPT je izomorfan sa pojedinim problemima kombinatorne optimizacije i neke heuristike za rešavanje tih problema mogu da se koriste i za rešavanje problema ASPT. U nastavku odeljka će biti dati kratki opisi problema kombinatorne optimizacije koji su primenljivi u oblasti ASPT ili barem u oblasti AST. Za svaki od opisanih problema daje se najpre verbalni opis, zatim matematička formulacija i na kraju se uspostavlja analogija problema sastavljanja jednog ili više paralelnih testova sa datim opštim problemom, odnosno njegovom formulacijom.

2.2.1 Problem kombinatorne optimizacije

Problemom kombinatorne optimizacije se smatra problem odabira podskupa objekata iz konačnog skupa objekata na način da se dostigne minimum (ili maksimum) neke funkcije cilja. Problemi kombinatorne optimizacije se javljaju u mnogim oblastima primene informatike. Njima se posebno bavi naučna disciplina operacionih istraživanja. Primeri ovih problema su: *Problem bojenja grafa*, *Problem najkraćeg puta*, *Problem trgovačkog putnika*, *Problem optimalnog rasporeda radne snage* i *Problem pakovanja*.

U nastavku ovog odeljka će biti reči o pojmovima kao što su dopustivo i optimalno rešenje, funkcija cilja, matematička formulacija problema kombinatorne optimizacije, klase složenosti problema i heurističko rešenje problema.

2.2.1.1 Dopustiva rešenja, optimalno rešenje i funkcija cilja

Rešenja problema kombinatorne optimizacije koja zadovoljavaju logičke uslove se nazivaju *dopustiva rešenja*. *Prostor rešenja* je skup svih dopustivih rešenja. Kvalitet dopustivog rešenja se procenjuje pomoću funkcije cilja. U prostoru rešenja se traži *optimalno rešenje* koje ima minimalnu ili maksimalnu vrednost funkcije cilja. U opštem slučaju, problem kombinatorne optimizacije se može definisati na sledeći način (Cvetković, et al., 1996):

$$f(x^*) = min\{f(x)|x \in S\} \tag{20}$$

gde je S skup dopustivih rešenja, x je dopustivo rešenje, f je funkcija cilja, a x^* je rešenje iz skupa dopustivih rešenja takvo da njegova vrednost funkcije cilja ima minimalnu vrednost u skupu dopustivih rešenja. Drugim rečima, x^* je optimalno rešenje.

Pošto važi da je:

$$max\{f(x)|x \in S\} = -min\{-f(x)|x \in S\} \tag{21}$$

problem maksimizacije funkcije cilja može da se svede na problem minimizacije, pa se ova dva problema jednako tretiraju i rešavaju.

2.2.1.2 Formalni opis problema

Važan slučaj formulacije problema kombinatorne optimizacije je celobrojno linearno programiranje (eng. *Integer Linear Programming*, ILP). Za ILP važi da je $S \subseteq \mathbb{Z}^n$, gde je $\mathbb{Z}^n$ skup n-dimenzionalnih vektora celobrojnih promenljivih, koji predstavljaju sve moguće kombinacije celih brojeva klase n, kao i da se funkcija cilja i ograničenja mogu prikazati linearnim jednačinama:

$$max\, c^T x \tag{22}$$

$$Ax \leq b \tag{23}$$

$$\text{i} \;\; x \in \mathbb{Z}^n \;\; A \in \mathbb{R}^{m \cdot n} \;\; c \in \mathbb{R}^n \;\; b \in \mathbb{R}^m \tag{24}$$

gde je $x = (x_1, x_2, \dots, x_n)$ vektor celobrojnih promenljivih koje treba odrediti (promenljive odlučivanja), $c = (c_1, c_2, \dots, c_n)$ je vektor realnih koeficijenata funkcije cilja, $b = (b_1, b_2, \dots, b_m)$ je vektor realnih brojeva koji predstavljaju gornje granice ograničenja, $(\cdot)^T$ je transponovani vektor i A je matrica dimenzija $m \cdot n$ realnih koeficijenata koji učestvuju u ograničenjima. Rešenje ILP problema jeste vektor promenljivih x kojima su dodeljene celobrojne vrednosti takve da funkcija cilja ima maksimalnu vrednost i da su sva ograničenja zadovoljena.

Kada promenljive x mogu da imaju samo vrednosti nula i jedan, tada je ILP problem poznat pod nazivom 0-1 linearni problem (0-1 LP). Problem AST je rano prepoznat kao problem koji može biti formulisan kao 0-1 LP problem (Theunissen, 1985) i (Boekkooi-Timminga, 1987).

Ukoliko neke promenljive mogu da imaju celobrojne vrednosti, dok neke mogu da imaju realne vrednosti, problem je poznat kao mešoviti celobrojni linearni problem (eng. *Mixed Integer Linear Problem*, MILP). Njegov matematički model se može prikazati na sledeći način:

$$max(c^T x + d^T y) \tag{25}$$

$$Ax + By \leq b \tag{26}$$

$$\text{i } x \in \mathbb{Z}^{n_1} \quad y \in \mathbb{R}^{n_2} \quad A \in \mathbb{R}^{m \cdot n_1} \quad B \in \mathbb{R}^{m \cdot n_2} \quad b \in \mathbb{R}^m \quad c \in \mathbb{R}^{n_1} \quad d \in \mathbb{R}^{n_2} \tag{27}$$

gde je $x = (x_1, x_2, \dots, x_{n1})$ vektor celobrojnih promenljivih, $y = (y_1, y_2, \dots, y_{n2})$ je vektor realnih promenljivih, $c = (c_1, c_2, \dots, c_{n1})$ i $d = (d_1, d_2, \dots, d_{n2})$ su odgovarajući realni koeficijenti funkcije cilja, $b = (b_1, b_2, \dots, b_m)$ je realni vektor gornjih granica ograničenja, $(\cdot)^T$ je transponovani vektor, A je matrica koeficijenata dimenzija $m \cdot n_1$ i B je matrica koeficijenata dimenzija $m \cdot n_2$.

Ukoliko neke promenljive odlučivanja mogu da imaju samo vrednosti nula i jedan, dok neke mogu da imaju i realne vrednosti, problem je poznat kao MZOLP (eng. *Mixed Zero One Linear Programming*) problem.

Ako se formulacija 0-1 LP primeni na problem ASPT, x predstavlja vektor promenljivih koje se pridružuju stavkama iz banke stavki. Kada se testovi sastavljaju jedan za drugim (redno sastavljanje testova), promenljiva x_i je vrednost pridružena stavki i i ima vrednost nula ukoliko stavka nije dodeljena testu, ili jedan ukoliko jeste. U slučaju paralelnog sastavljanja testova, stavkama je pridružena promenljiva x_{it} čija je vrednost nula ukoliko stavka nije dodeljena nijednom testu, a jedan ukoliko je stavka i dodeljena testu t, pri čemu je $i = 1, 2, ..., I$; $t = 1, 2, ..., T$.

2.2.1.3 Klase složenosti problema

Složenost (eng. *complexity*) problema se definiše kao vremeska složenost najboljeg algoritma kojim se rešava problem. Postoji više klasa složenosti problema, o kojima će biti reči u ovom odeljku.

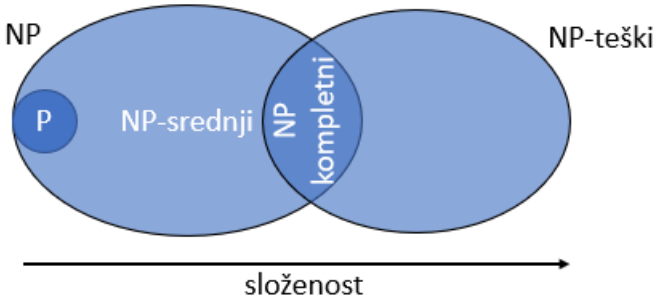
Kaže se da je problem u klasi složenosti *P* (*Polynomial*) ukoliko može da se reši determinističkom Turingovom mašinom (Turing, 1936) (hipotetička mašina) u polinomijalnom vremenu (vremenu koje polinomijalno zavisi od dimenzije problema). Kaže se i da se problem klase P može rešiti determinističkim algoritmom (algoritmom koji uvek daje tačno rešenje) u polinomijalnom vremenu.

Klasi NP (*Nondeterministic Polynomial*) pripadaju problemi koji mogu da se reše u polinomijalnom vremenu na nedeterminističkoj Turingovoj mašini. Nedeterministička mašina u rešavanju NP problema koristi nedeterminističke algoritme koji mogu da „pogađaju“ tačno rešenje. Rešenja NP problema mogu da se provere u polinomijalnom vremenu. Problemi klase P su podskup problema klase NP ($P \subseteq NP$), odnosno P problemi bi mogli da se smatraju NP-najlakšim problemima.

NP-kompletni (eng. *NP-complete*) problemi su najteži problemi u klasi NP problema. NP-kompletan problem ima osobinu da pripada klasi NP problema, kao i da postoji preslikavanje kojim se svaki NP problem može, u polinomijalnom vremenu, preslikati u njega. Drugim rečima, NP problemi su specijalizacije NP-kompletnih problema, odnosno NP-kompletni problemi su generalizacije NP problema. Ako postoji rešenje NP-kompletnog problema, ono se može primeniti i na svaki NP problem. Začetnik teorije NP-kompletnih problema je naučnik Tomas Kuk (Cook, 1971) koji je i uveo pojam polinomijalnog preslikavanja problema jednog u drugi. Naučnik Ričard Karp je posle toga izdvojio probleme poznate kao Karpovih 21 NP-kompletnih problema (Karp, 1972). ILP problemi su generalno NP-kompletni (Papadimitriou & Steiglitz, 1982). Problemi u klasi NP problema, a nisu NP-kompletni, nazivaju se ponekad NP-lakim (eng. *NP-easy*)

problemima. Problemi koji su u klasi NP problema, a nisu ni u klasi P problema ni u klasi NP-kompletnih problema, nazivaju se ponekad NP-srednjim (eng. *NP-intermediate*) problemima.

NP-teški (eng. *NP-hard*) problemi su najteži problemi za rešavanje. To su problemi koji su složeni bar toliko kao bilo koji problem u klasi NP, ali ne moraju da pripadaju klasi NP problema. Svojstvo problema da njegovo dopustivo rešenje ne može da se proveri u polinomijalnom vremenu je dovoljno da se problem svrsta u klasu NP-teških problema, ali nije potrebno. Postoje NP-teški problemi čije se rešenje može proveriti u polinomijalnom vremenu i NP-teški problemi kod kojih to nije moguće. Skupovi NP-teških i NP problema se delimično preklapaju (njihov presek nije prazan skup), a razlika skupova NP-teških i NP problema čini podskup NP-teških problema čije dopustivo rešenje ne može da se proveri u polinomijalnom vremenu. Problemi koji pripadaju i klasi NP problema i klasi NP-teških problema, su NP-kompletni problemi (Slika 2.5).



Slika 2.5 - NP složenost

2.2.1.4 O heuristikama

Nisu svi problemi kombinatorne optimizacije u klasi NP-teških problema (Hoos & Stutzle, 2005). Prilikom rešavanja problema, prirodno bi bilo očekivati da se pretraži prostor rešenja koji eksponencijalno raste sa dimenzijom problema, odnosno brojem objekata koji bi mogli da učestvuju u rešenju. Međutim, u slučaju nekih problema kombinatorne optimizacije, kao što je *Problem najkraćeg puta*, postoje efikasni algoritmi za rešavanje u polinomijalnom vremenu. *Problem najkraćeg puta* je problem u teoriji grafova, gde u grafu G grane imaju dodeljenu težinu i traži se put između dva čvora v_1 i v_2 takav da je zbirna težina grana na putu između čvora v_1 i čvora v_2 minimalna. Ovaj problem efikasno rešava algoritam poznat pod nazivom *Dajkstrin algoritam* (Dijkstra, 1959).

Međutim, mnogi problemi kombinatorne optimizacije jesu u kategoriji NP-teških problema i za njih ne postoji efikasan algoritam za rešavanje (vreme izvršavanja u najgorem slučaju eksponencijalno raste sa dimenzijom problema), tako da se pribegava traženju suboptimalnog rešenja koristeći heuristike. Generalno, heuristike mogu da se podele u dve klase na osnovu načina na koji se rešenje generiše (Blum & Roli, 2003):

a. Konstruktivne heuristike, u kojima se do rešenja dolazi dodavanjem komponenti u inicijalno prazno rešenje, na osnovu unapred definisanih kriterijuma;
b. Heuristike poboljšanja, u kojima se rešenja pronalaze pretragom u prostoru rešenja, gde se rešenja sistematično ispituju da bi se našlo suboptimalno rešenje.

Konstruktivne heuristike su brže od heuristika poboljšanja, ali se često dobijaju rešenja lošijeg kvaliteta nego kod heuristika poboljšanja. Ukoliko se prekine izvršenje konstruktivne heuristike, ne dobija se rešenje.

Heuristike poboljšanja uzastopno generišu i procenjuju rešenja u prostoru rešenja, sve dok nije zadovoljen uslov zaustavljanja algoritma. Uslov zaustavljanja može da bude proteklo vreme koje je

prevazišlo zadato maksimalno vremena za pretragu, spuštanje ispod praga greške pronađenog rešenja u odnosu na optimalno rešenje (ako postoji način da se meri takva greška) i drugi. Konstruktivne heuristike se često koriste za generisanje inicijalnog rešenja kod heuristika poboljšanja.

Razvijene su razne strategije koje efikasno pretražuju prostor rešenja i predstavljaju šablon za kreiranje konkretnih heuristika za određeni problem. Ove strategije su poznate kao metaheuristike.

U odeljku 3.2.2 će biti prikazane (meta)heuristike koje su se koristile za rešavanje ASPT problema i bile publikovane u otvorenoj literaturi.

2.2.2 Neki poznati problemi kombinatorne optimizacije koji se koriste i za ASPT

U ovom odeljku biće predstavljeni često sretani opšti probleme kombinatorne optimizacije, čije se formulacije koriste i pri formulisanju problema ASPT. To su: problemi pakovanja u koje spadaju problem ranca, problem pakovanja u korpe i problem pakovanja skupa, zatim problem maksimalnog protoka i problem maksimalne klike. Pri predstavljanju svakog od problema najpre će biti verbalno definisan problem, zatim će biti data njegova matematička formulacija, a na kraju će se povući analogija između datog opšteg problema i odgovarajućeg problema ASPT.

2.2.2.1 Problem pakovanja

U oblasti transporta i logistike postoji problem utovara tereta, gde je cilj da se objekti utovare u transportno sredstvo na optimalan način (ili uz najbolje iskorišćenje prostora ili uz postizanje najveće vrednosti utovarenih objekata). Ovaj problem pripada klasi *problema pakovanja*. Problemi pakovanja se nalaze u literaturi pod različitim imenima, kao što su problem ranca i problem pakovanja u korpe, ali je njihova logička struktura jednaka (Dyckhoff, 1990). Zajedničko im je što:

- postoji skup velikih objekata (kontejnera),
- postoji skup malih objekata (koji se pakuju u kontejnere),
- mali objekti se pakuju u velike objekte po definisanom šablonu.

Za formulaciju problema ASPT se najčešće koriste formulacije problema ranca, problema pakovanja u korpe i problema pakovanja skupa.

2.2.2.1.1 Problem ranca

Problem ranca (KP) je problem odabira objekata koji imaju određenu vrednost i težinu, za pakovanje u ranac. Jedan od mogućih načina pakovanja ranca jeste da se objekti pakuju tako da se minimizuje težina ranca (zbir težina objekata smeštenih u ranac), dok vrednost ranca (zbir vrednosti objekata smeštenih u ranac) ne sme da bude ispod predviđene vrednosti V. Ova definicija je jedna od varijanti KP problema. Ona se naziva jedno-dimenzionalnim problemom ranca (1D KP) zato što postoji samo jedno ograničenje nad atributom ranca (u ovom slučaju, to je vrednost). Ukoliko postoji više od jednog ograničenja nad atributima objekata i/ili ranca problem se naziva višedimenzionalnim problemom ranca. KP je NP-težak problem.

Problem ranca može da se formuliše kao 0-1 LP problem na sledeći način (Martello & Toth, 1990):

$$min \sum_{i=1}^{I} w_i x_i \tag{28}$$

$$\sum_{i=1}^{I} v_i x_i \geq V \tag{29}$$

$$x_i \in \{0,1\} \;\; i = 1, \dots I \tag{30}$$

gde x_i ima vrednost jedan ukoliko je objekat i dodeljen rancu, a nula ukoliko nije, w_i predstavlja težinu objekta i, v_i vrednost objekta i, I je ukupan broj raspoloživih objekata, a V je minimalna vrednost ranca. Težina i vrednost objekata, kao i minimalna težina ranca, su celobrojne vrednosti. Cilj je da se minimizuje težina ranca, pod uslovom da vrednost ranca ne bude manja od vrednosti V.

Proširena varijanta problema pakovanja ranca jeste problem pakovanja u više ranaca i njegova formulacija je sledeća:

$$min \sum_{i=1}^{I} \sum_{b=1}^{B} w_i x_{ib} \tag{31}$$

$$\sum_{i=1}^{I} v_i x_{ib} \geq V \quad b = 1, \dots, B \tag{32}$$

$$\sum_{b=1}^{B} x_{ib} \leq 1 \quad i = 1, \dots I \tag{33}$$

$$x_{ib} \in \{0,1\} \;\; i = 1, \dots I, b = 1, \dots, T \tag{34}$$

gde x_{ib} ima vrednost jedan ukoliko je objekat i dodeljen rancu b, a nula ukoliko nije, B je ukupan broj ranaca na raspolaganju. Cilj je da se minimizuje ukupna težina svih ranaca (31) sa ograničenjem da nijedan ranac nema vrednost manju od V (32), kao i da jedan objekat može da pripadne samo jednom rancu (33).

Kada se problem sastavljanja testa predstavi problemom ranca, tada stavke iz banke stavki predstavljaju objekte koji se smeštaju u rance, dok testovi koji se sastavljaju predstavljaju rance. U pojednostavljenoj varijanti problema, težina svakog objekta w_i jednaka je jedan, dok $IIF_i(\theta)$ stavke i za određenu sposobnost θ predstavlja vrednost svake stavke v_i. U tom slučaju je cilj da se minimizuje ukupan broj stavki u testu. Pri tome mora da bude zadovoljen niz uslova da vrednost TIF(θ), za svaku od vrednosti sposobnosti θ, ne bude manja od zadatih vrednosti, odnosno TTIF(θ). Problem ASPT koji se zasniva na formulaciji KP opisan je u odeljku 3.1.1.2.1 i odeljku 3.1.1.2.5.

2.2.2.1.2 Problem pakovanja u korpe

Problem pakovanja u korpe (BPP) je problem odabira objekata i smeštanja u korpe na način da se broj korpi, težina korpi, ili neke druge karakteristike korpi, optimizuju, vodeći računa o ograničenjima. Jednodimenzionalni problem pakovanja u korpe (1D BPP) je specijalizacija problema pakovanja u korpe, u kome je objekat koji se smešta u korpu opisan jednom veličinom (dimenzija). 1D BPP je NP-težak problem.

Ovaj problem može da se predstavi MZOLP ili ILP formulacijom na sledeći način (Martello & Toth, 1990):

$$min\, Z \tag{35}$$

$$\sum_{i=1}^{I} w_i x_{ib} \leq Z \quad b = 1, \dots, B \tag{36}$$

$$\sum_{b=1}^{B} x_{ib} = 1 \quad i = 1, \dots, I \tag{37}$$

$$x_{ib} \in \{0,1\} \quad i = 1, \dots, I \quad b = 1, \dots, B \tag{38}$$

gde x_{ib} ima vrednost jedan ukoliko je objekat i dodeljen korpi b, a nula ukoliko nije, w_i je realna ili celobrojna vrednost koja predstavlja težinu objekta i, I je broj raspoloživih objekata, B je broj korpi, Z je ili realna (kada je u pitanju MZOLP formulacija) ili celobrojna (kada je u pitanju ILP formulacija) nepoznata promenljiva koja predstavlja težinu najteže korpe. Cilj je da se minimizuje težina najteže korpe Z (35) i time izbalansira težina korpi. Ograničenje (36) zahteva da je težina svih objekata u nekoj korpi manja ili jednaka Z, a ograničenje (37) govori da objekat i može da bude samo u jednoj korpi. Dakle, Z je maksimum težine korpe koji treba da se minimizuje, pa se ovakav problem naziva i MINIMAX problemom.

U formulaciji problema ASPT, stavke iz banke stavki predstavljaju objekte, dok testovi predstavljaju korpe. Težinski koeficijent stavke najčešće odgovara psihometrijskim atributima, kao što su težina rešavanja stavke ili diskriminativnost stavke ili kombinacija oba. Cilj je balansiranje težinskih koeficijenata stavki u testovima. Problem ASPT koji se zasniva na 1D BPP formulaciji opisan je u odeljku 3.1.1.1.2.

2.2.2.1.3 Problem pakovanja skupa

Problem pakovanja skupa (eng. *Set Packing Problem*, SPP) podrazumeva da postoji skup podskupova $S = \{S_1, S_2, \dots, S_m\}$ konačnog univerzalnog skupa elemenata $U = \{e_1, e_2, \dots, e_n\}$. Problem pakovanja skupa je problem pronalaženja skupa disjunktnih podskupova S_j ($j = 1,\dots,k \leq m$). Pronađeni skup disjunktnih podskupova se naziva *pakovanje*. Ovaj problem pripada klasi Karpovih 21 NP-kompletnih problema. Problem maksimalnog pakovanja skupa (MSPP) je optimizaciona verzija problema pakovanja skupa, gde se traži maksimalan broj disjunktnih skupova S_j. MSPP je NP-težak problem.

Problem maksimalnog pakovanja skupa može da se formuliše kao 0-1 LP na sledeći način (Wolsey & Nemhauser, 1999):

$$max \sum_{i=1}^{m} x_i \tag{39}$$

$$\sum_{i=1}^{m} y_{ji} x_i \leq 1 \quad j = 1, \dots, n \tag{40}$$

$$x_i, y_{ji} \in \{0,1\} \quad i = 1, \dots, m \quad j = 1, \dots, n \tag{41}$$

gde je x_i jednako jedan ukoliko je podskup S_i u paketu, a nula ukoliko nije, y_{ji} ima vrednost jedan ukoliko se element e_j nalazi u podskupu S_i, a nula ukoliko se ne nalazi. Cilj je da se maksimizuje ukupan broj podskupova u pakovanju (39). Ograničenje (40) obezbeđuje da su izabrani podskupovi disjunktni. Ne moraju svi elementi univerzalnog skupa da budu sadržani u podskupovima pakovanja. Varijanta problema gde su svi elementi univerzalnog skupa sadržani u podskupovima pakovanja se naziva problemom pokrivanja skupa (eng. *Set Covering Problem*).

Jedna od mogućih analogija sa problemom ASPT je opisana u radu (Belov & Armstrong, 2006). Pretpostavka je da se test sastoji iz podtestova. Svaki podtest može da ima preambulu koja prethodi stavkama podtesta na koje se daju odgovori. Jedna preambula može da prethodi jednoj ili više stavki. Banka sadrži stavke i podtestove koji grupišu stavke, ali stavka može da pripada i različitim podtestovima. Analogija problema ASPT sa problemom maksimalnog pakovanja skupa se može uspostaviti tako što podtestovi predstavljaju podskupove, a stavke predstavljaju elemente univerzalnog skupa. Cilj je da se izabere maksimalan broj podtestova koji nemaju zajedničke stavke. Problem ASPT koji se zasniva na MSPP formulaciji opisan je u odeljku 3.1.3.

2.2.2.2 Problem mrežnog protoka

Problem mrežnog protoka (eng. *Network Flow Problem*, NFP) je problem kombinatorne optimizacije sa raznim primenama u transportu, elektroenergetskim sistemama, računarskim mrežama i komunikacionim mrežama (Harris & Ross, 1955) i (Schrijver, 2002). Problem može da se predstavi usmerenim grafom, u kojem se razlikuju izvorišni čvorovi i odredišni čvorovi. Svaka grana ima pridruženu cenu. Varijanta NFP pod nazivom problem minimalne cene mrežnog protoka (eng. *Minimum-Cost Flow Problem*, MCFP) ima za cilj da se pronađe minimalna ukupna cena puteva koje prelaze svi raspoloživi objekti iz izvorišnih čvorova do svojih odgovarajućih odredišnih čvorova. Za ove probleme postoje efikasni algoritmi polinomijalne složenosti kao što je *Out-of-Kilter* algoritam (Fulkerson, 1961).

Ovaj problem može da se predstavi ILP formulacijom na sledeći način (Ahuja, Magnanti, & Orlin, 1993):

$$min \sum_{i=1}^{I} \sum_{j=1}^{J} c_{ij} x_{ij} \tag{42}$$

$$\sum_{j=1}^{J} x_{ij} \leq S_i \quad i = 1, \dots, I \tag{43}$$

$$\sum_{i=1}^{I} x_{ij} \geq D_j \quad j = 1, \dots, J \tag{44}$$

$$x_{ij} \in \{0,1, \dots\} \quad i = 1, \dots, I \quad j = 1, \dots, J \tag{45}$$

gde je x_{ij} broj objekata koji se transportovao iz izvorišnog čvora i ka odredišnom čvoru j, c_{ij} je cena transporta između čvora i i čvora j, S_i je maksimalan broj objekata koji su na raspolaganju u izvorišnom čvoru i, a koji se transportuju na J odredišnih čvorova. D_j je minimalan broj objekata koji stižu u čvor j iz I izvorišnih čvorova. Svaki izvorišni čvor je povezan po jednom granom sa svakim odredišnim čvorom tako da iz svakog izvorišnog čvora objekti mogu da se transportuju u svaki odredišni čvor. Cilj je da se minimizuje ukupna cena svih pređenih puteva (42). Ograničenje (43) obezbeđuje da se ne prekorači broj raspoloživih objekata iz svakog izvorišnog čvora, dok ograničenje (44) osigurava da se minimalan broj objekata isporuči u svaki odredišni čvor.

Kada se koristi u kontekstu formulacije problema ASPT, rešavanje problema minimalne cene mrežnog protoka se koristi kao prva faza rešavanja problema, gde se stavke grupišu u podskupove po nekom kriterijumu (na primer, sličan težinski koeficijent stavke koji najčešće odgovara psihometrijskim atributima kao što su težina stavke ili diskriminativnost stavke ili kombinacija oba), a posle raspoređuju u testove u drugoj fazi. Svaka stavka iz banke stavki predstavlja izvorišni čvor, dok podskup grupisanih stavki predstavlja odredišni čvor. Cena grane može da bude Euklidsko rastojanje vrednosti atributa izvorišnog čvora (stavke) od vrednosti odgovarajućeg

atributa koji je pridružen odredišnom čvoru. Problem ASPT koji se zasniva na NFP formulaciji opisan je u odeljku 3.1.1.1.3 i odeljku 3.1.2.1.

2.2.2.3 Problem maksimalne klike

Problem maksimalne klike (MCP), je kombinatorno optimizacioni problem u teoriji grafova. Graf G se opisuje parom $G = \{V, E\}$, gde V (eng. *Vertices*) predstavlja skup čvorova (temena), a E (eng. *Edges*) skup grana (linija) koje povezuju čvorove. Klika ($C \subseteq V$) predstavlja podskup čvorova u kojem su svi čvorovi međusobno direktno povezani granama, svaki sa svakim. Problem maksimalne klike jeste problem pronalaženja klike sa najvećim brojem čvorova. MCP je NP-težak problem.

Problem može da se predstavi 0-1 LP formulacijom na sledeći način (Bomze, Budinich, Pardalos, & Paleilo, 1999):

$$max \sum_{i=1}^{n} x_i \tag{46}$$

$$x_i + x_j \leq 1 \quad \forall (i,j) \notin E \tag{47}$$

$$x_i \in \{0,1\} \quad i = 1, \dots n \tag{48}$$

gde je x_i promenljiva koja ima vrednost jedan ukoliko je čvor i deo klike, a nula ukoliko nije. Cilj je naći kliku maksimalne veličine, merene brojem čvorova koji joj pripadaju (46). Ograničenje (47) osigurava da čvorovi i i j nisu deo klike ukoliko nisu povezani granom (grana je uređeni par čvorova (i, j)).

Analogija problema ASPT sa problemom maksimalne klike postoji kada je cilj kreiranja testova maksimizacija broja testova i to testova koji dele stavke. U tom slučaju test predstavlja čvor, a grana između dva čvora postoji ukoliko testovi dele stavke. Problem ASPT koji se zasniva na MCP formulaciji opisan je u odeljku 3.1.3.

2.3 Problem ASPT rešavan u disertaciji

U ovom odeljku će biti detaljno opisan problem koji se rešava u disertaciji i ukazano na glavne ciljeve istraživanja koji obuhvataju prikaz postojećih metoda za ASPT uz predlog njihove klasifikacije i razvoj novog metoda ASPT. Uz matematičku formulaciju rešavanog problema, biće naveden motiv, cilj i značaj istraživanja.

U rešavanju problema ASPT koriste se znanja iz raznih oblasti kao što su psihometrija, primenjena matematika, operaciona istraživanja, te se kao posledica toga u literaturi javlja puno različitih modela i pristupa rešavanju problema, koristeći neujednačene terminologije u kojima se isti pojmovi različito nazivaju. Jedan od načina da se ostvari jasan pregled dosadašnjeg naučnog rada objavljenog u literaturi koji se bavi problemom ASPT jeste klasifikacija pristupa uz korišćenje ujednačene terminologije u prikazima tih pristupa i korišćenje istih simbola za predstavljanje istih veličina u matematičkim izrazima. Za svaki pristup rešavanju problema ASPT je karakteristično da se problem najpre formalno opisuje (formuliše), a zatim rešava. U ovoj disertaciji se odvojeno analiziraju i klasifikuju formulacije problema ASPT i načini rešavanja problema ASPT pomoću heuristika. U prikazima različitih formulacija problema i njihovih rešenja koristi se ujednačena terminologija i oznake veličina.

U odeljku 3.1 biće prikazane i klasifikovane formulacije problema ASPT. Pregled i klasifikacija heuristika koje se koriste za rešavanje problema ASPT će biti prikazane u odeljku 3.2. Na kraju tog odeljka će se prikazati i ukrstiti formulacije sa postojećim heuristikama za rešavanje, čime će se ukazati na mogući prostor za istraživanje rešavanja nekih poznatih formulacija problema poznatim

(meta)heuristikama, koje se nisu do sada koristile za rešavanje datih formulacija. Pri tome treba imati u vidu da se ne može svaka od već primenjenih heuristika primeniti na svaku formulaciju problema.

Analizom postojećih pristupa u rešavanju problema ASPT došlo se do zaključka da postoji prostor za istraživanje koje bi dovelo do efikasnog (brzog) algoritma sa predvidivim vremenom izvršavanja, kojim se sastavljaju dobro balansirani paralelni testovi visokog kvaliteta. Efikasnost algoritma i njegovo predvidivo vreme izvršavanja je veoma važno za primenu u uslovima čestog sastavljanja testova ograničenim računarskm resursima. Osnovni motiv ovog istraživanja je bio razvoj upravo takvog algoritma za ASPT, koji bi bio primenljiv u navedenim uslovima, tipičnim za obrazovne institucije. Kao rezultat tog istraživanja, u ovoj disertaciji se predlaže jedan novi metod ASPT. Metod koristi formulaciju problema pakovanja u korpe (odeljak 2.2.2.1.2), koju rešava konstruktivnom heuristikom zasnovanom na NEH algoritmu (odeljak 3.3), pod imenom NEHTA (eng. NEH *Test Assembly*), koji će biti opisan u poglavlju 4.

Problem koji se rešava u ovoj disertaciji je formulisan u radu (Brusco, Köhn, & Steinley, 2013). Predložena formulacija problema ASPT je zasnovana na analogiji sa jednodimenzionalnim problemom pakovanja u korpe (1D BPP), gde se jednodimenzionalnost odnosi na jednu veličinu kojom se stavka opisuje (u ovom slučaju težinski koeficijent). Težinski koeficijent stavke može da se definiše u kontekstu klasične teorije testiranja (CTT) ili u nekom drugom statističkom kontekstu (Fan, 1998) i (van der Linden W. J., 2005).

Pretpostavka je da je broj testova koji se sastavlja fiksan, da težinski koeficijent svakog testa nije ograničen i da svaki test ima jednak broj stavki. Stavke su inicijalno grupisane u skupove s (s = 1,..., S) i svaka stavka pripada tačno jednom skupu. Stavke u skupu treba da imaju slične težinske koeficijente. Skupovi imaju jednak broj stavki i broj stavki u skupu je jednak broju testova koji se sastavljaju, T. Testovi se sastavljaju odabirom tačno jedne stavke iz svakog skupa za svaki test.

Cilj rešavanja problema je da sastavljeni paralelni testovi budu dobro balansirani po težini i diskriminativnosti stavki, kako bi na što približniji način merili sposobnost ispitanika. Drugim rečima, cilj je da se testovima dodele stavke tako da testovi imaju približno jednake težinske koeficijente. Cilj može da se postigne na način da se minimizuje maksimalni težinski koeficijent testa među paralelnim testovima.

Formulacija rešavanog problema je u literaturi poznata kao jednodimenzionalni minimaks problem pakovanja u korpe ograničene veličine (eng. *one-dimensional minimax bin-packing problem with bin size constraints*, 1D BPP MINIMAX_BSC) (Brusco, Köhn, & Steinley, 2013) i detaljno je opisana u odeljku 3.1.1.1.2. Opšti problem pakovanja u korpe je opisan u odeljku 2.2.2.1.2.

Među karakteristikama novog metoda ističu se predvidivo vreme njegovog izvršenja, efikasnost i jednostavnost implementacije. Predvidivo vreme i efikasno sastavljanje testova je od posebnog značaja u obrazovnim ustanovama u kojima se vrše česta testiranja relativno malog broja ispitanika, a gde su raspoloživi računarski resursi za ASPT ograničeni.

3 Pregled postojećih metoda

U ovom poglavlju će biti dat pregled objavljenih metoda ASPT u otvorenoj literaturi. Metodi će biti predstavljeni prvo po formulaciji, a posle po načinu rešavanja problema ASPT (Ignjatović, Bojić, & Tartalja, 2021).

Tabela 3.1 navodi značenje simbola koji će se na jedinstven način koristiti u pregledu postojećih metoda.

Tabela 3.1 Značenje simbola

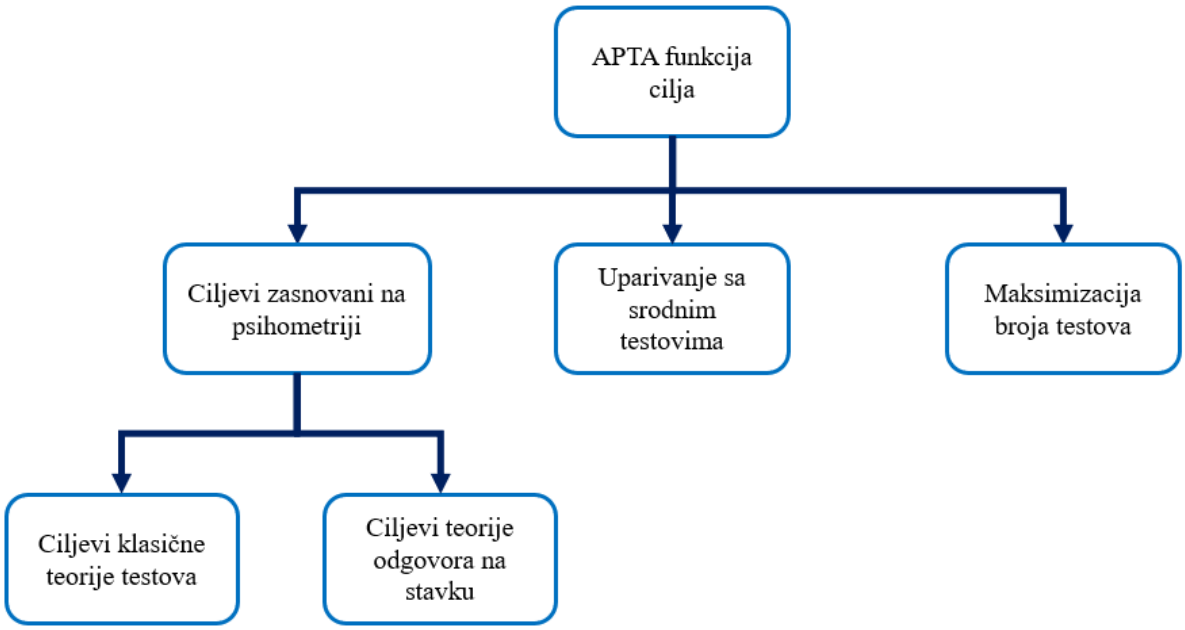
I	Ukupan broj stavki u banci stavki
i	Indeks stavke u banci stavki $i = 1, 2, \ldots, I$
x_i	Promenljiva odlučivanja koja označava prisustvo stavke i u testu
K	Ukupan broj diskretnih vrednosti sposobnosti za analizu
k	Indeks diskretnih vrednosti sposobnosti $k = 1, \ldots, K$
θ_k	Nivo sposobnosti k
$P_i(\theta_k)$	Verovatnoća tačnog odgovora na stavku i za sposobnost θ_k
$IIF_i(\theta_k)$	Informaciona funkcija stavke i za sposobnost θ_k
$TCF(\theta_k)$	Karakteristična funkcija testa za sposobnost θ_k
$TIF(\theta_k)$	Informaciona funkcija testa za sposobnost θ_k
$TTIF(\theta_k)$	Ciljna informaciona funkcija testa za sposobnost θ_k
n	Broj stavki u testu
T	Ukupan broj testova koji se sastavljaju
t	Indeks testa $t = 1, 2, \ldots, T$
x_{it}	Promenljiva odlučivanja koja označava prisustvo stavke i u testu t
p_i	Težina stavke, po CTT
d_i	Diskriminativnost stavke, po CTT
b_i	Težina stavke, po IRT
a_i	Diskriminativnost stavke, po IRT

3.1 Pregled i klasifikacija formulacija problema ASPT

Ovaj odeljak je posvećen pregledu metoda sa stanovišta formulacije problema ASPT. Formulacije koje se koriste u metodima će biti klasifikovane prema cilju, a zatim prikazane matematičkim formulama na ujednačen način koji može da se razlikuje od simbola koji su se koristili u originalnom radu u kojem je metod objavljen.

Formulacije koje se koriste u literature se mogu klasifikovati, na osnovu funkcije cilja, u tri kategorije (Slika 3.1):

1. Ciljevi zasnovani na psihometriji
2. Uparivanje sa srodnim testovima
3. Maksimizacija broja testova



Slika 3.1 - Klasifikacija formulacija problema ASPT prema cilju

Praktična ograničenja u modelima će uglavnom biti izostavljena u prikazu formulacija i mogu biti predstavljena uopštenom formom na sledeći način:

$$\sum_{i=1}^{I} v_{im}\, x_i \leq c_m \quad 1 \leq m \leq M \tag{49}$$

gde je v_{im} osobina m stavke i, a c_m je vrednost ograničenja (gornje granice) za osobinu m za ceo test. Neka od mogućih ograničenja su: broj stavki po oblastima, ograničenje vremena potrebnog za rešavanje testa, stepen asocijacije između stavke i oblasti, dozvoljen broj ponavljanja prethodno korišćenih stavki. Detaljan pregled mogućih ograničenja je dat u (van der Linden W. J., 2005).

3.1.1 Ciljevi zasnovani na psihometriji

U ovom odeljku su prikazane formulacije metoda sa ciljevima zasnovanim na psihometriji i to najčešće korišćenim teorijama testa, klasičnoj teoriji testa (CTT) i teoriji odgovora na stavku (IRT). Statistički atributi testa i stavke se razlikuju u jednoj i u drugoj teoriji, kao i definicija paralelizma testova, tako da se ciljevi razlikuju.

3.1.1.1 Ciljevi klasične teorije testova

Ako testovi imaju približno jednake rezultate testa, varijanse greške, kao i koeficijent α, kaže se da su paralelni po CTT (Lord & Novick, 1968). Slab paralelizam podrazumeva približno jednak srednji rezultat testa kao i približno jednaku vrednost standardne devijacije rezultata (Armstrong, Douglas, & Wang, 1994). Rezultat testa i koeficijent α zavise od težine i diskriminativnosti izabranih stavki, tako da formulacije uglavnom sadrže ove parametre za definisanje funkcije cilja kao i za ograničenja.

3.1.1.1.1 Balansiranje težine testova

Jin i koautori su u radu (Yin, Chang, Hwang, Hwang, & Chan, 2006) predložili formulaciju za simultano sastavljanje paralelnih testova sa ciljem da se minimizuje apsolutna razlika između prosečne težine svakog testa i ciljne težine (51). Formulacija je poznata pod nazivom sastavljanje serije ispitnih testova (eng. *serial test sheet composition*, STSC).

0-1 LP formulacija problema je sledeća:

$$\text{Promenljiva:} \quad x_{it} \quad i = 1,2,\ldots,I \quad t = 1,\ldots,T \tag{50}$$

$$\text{Funkcija cilja:} \quad Z_t = min\left\{\sum_{t=1}^{T}\left|\frac{\sum_{i=1}^{I} p_i\ x_{it}}{\sum_{i=1}^{I} x_{it}} - TDiff\right|\right\} \tag{51}$$

$$\text{Ograničenja:} \quad \sum_{i=1}^{I} x_{it} = n \quad t = 1,\ldots,T \tag{52}$$

$$x_{it} \in \{0,1\} \tag{53}$$

gde je *TDiff* ciljna težina.

Problem je rešavan heuristikom zasnovanom na metaheuristici optimizacije rojem čestica (PSO) (odeljak 3.2.2.6).

3.1.1.1.2 Balansiranje testova po diskriminativnosti i težini stavki

Formulacija prikazana u radu (Yin, Chang, Hwang, Hwang, & Chan, 2006) je proširena u radu (Hwang, Chu, Yin, & Lin, 2008) i koristi se takođe za simultano sastavljanje paralelnih testova. Funkcija cilja (51) je proširena tako da uzima u obzir i diskriminativnost. Predložena su dva cilja. Jedan je da se minimizuje maksimalna razlika prosečne diskriminativnosti bilo koja dva testa. Drugi je da se minimizuje razlika između prosečne težine testa i ciljne težine testa. Ova dva cilja su sadržana u jednoj funkciji cilja koristeći težinske koeficijente (54). Formulacija je poznata pod nazivom sastavljanje paralelnih ispitnih testova (eng. *parallel test sheet composition*, PTSC).

$$\text{Funkcija cilja:} \quad min\left\{\begin{array}{c}\omega_1 * \max\limits_{1\le t_1,t_2\le T}\left\{\left|\frac{\sum_{i=1}^{I} d_i x_{it_1}}{\sum_{i=1}^{I} x_{it_1}} - \frac{\sum_{i=1}^{I} d_i x_{it_2}}{\sum_{i=1}^{I} x_{it_2}}\right|\right\} + \\ \omega_2 * \frac{1}{T}\sum_{t=1}^{T}\left|\frac{\sum_{i=1}^{I} p_i\ x_{it}}{\sum_{i=1}^{I} x_{it}} - TDiff\right|\end{array}\right\} \tag{54}$$

Težinski koeficijenti ω_1 i ω_2 određuju važnost svakog cilja pojedinačno ($\omega_1 + \omega_2 = 1$).

Ovaj problem je rešavan heuristikom zasnovanom na pretrazi sa tabuima (odeljak 3.2.2.3).

Sastavljanje paralelnih testova tako da imaju balansirane prosečne težine i diskriminativnosti je proučavana i u radu (Ho, Yin, Hwang, Shyu, & Yean, 2009), koji se oslanja na formulaciju u radu (Hwang, Chu, Yin, & Lin, 2008). Za razliku prethodno opisana dva pristupa, koja su imala jednu funkciju cilja, u navedenom radu se koriste dve funkcije cilja:

$$\text{Funkcije cilja:} \quad min\left\{\max_{1\le t_1,t_2\le T}\left\{\left|\frac{\sum_{i=1}^{I} d_i x_{it_1}}{\sum_{i=1}^{I} x_{it_1}} - \frac{\sum_{i=1}^{I} d_i x_{it_2}}{\sum_{i=1}^{I} x_{it_2}}\right|\right\}\right\} \tag{55}$$

$$min\left\{\max_{1\le t\le T}\left\{\frac{\sum_{i=1}^{I} p_i\ x_{it}}{\sum_{i=1}^{I} x_{it}} - TDiff\right\}\right\} \tag{56}$$

Prva funkcija cilja (55) ima zadatak da minimizuje maksimalnu razliku prosečne diskriminativnosti između bilo koja dva testa, dok druga funkcija cilja (56) ima zadatak da maksimalna razlika između prosečne težine testa i ciljne težine bude minimalna. Ova formulacija je poznata pod nazivom sastavljanje paralelnih ispitnih testova sa više ciljeva (eng. *multi-objective parallel test sheet composition*, MOPTSC). Problem je rešavan heuristikom zasnovanom na metaheuristici optimizacije rojem čestica (PSO) (odeljak 3.2.2.6).

U radu Brusco et al. (Brusco, Köhn, & Steinley, 2013) je cilj postupka balansiranje težinskih koeficijenata testova. Težinski koeficijent kombinuje težinu i diskriminativnost stavke i računa se prema formuli (57) predloženoj u (van der Linden & Boekkooi-Timminga, 1988), a koja se koristi i u radovima (Brusco, Köhn, & Steinley, 2013) i (Pereira & Vila, 2015), kao i u ovoj disertaciji.

$$w_i = 0.5 \cdot p_i + 0.5 \cdot p_i \cdot (1 - p_i) \cdot d_i \tag{57}$$

gde je p_i težina stavke a d_i diskriminativnost stavke. U formuli stoji oznaka diskriminativnosti d_i koja označava ovu veličinu definisanu u CTT, međutim, ona bi, bez uticaja na metod NEHTA, mogla da se zameni i oznakom a_i koja označava istoimenu veličinu definisanu u IRT.

Formulacija MINIMAX_BSC problema može da se prikaže sledećim iskazima i (ne)jednakostima.

$$\text{Promenljive:} \quad Z, x_{ist} \quad i = 1, \dots, T \quad s = 1, \dots, S \quad t = 1, \dots, T \tag{58}$$

$$\text{Funkcija cilja:} \quad \min Z \tag{59}$$

$$\text{Ograničenja:} \quad Z \ge \sum_{i=1}^{T}\sum_{s=1}^{S} w_{is} x_{ist} \quad t = 1, \dots, T \tag{60}$$

$$\sum_{i=1}^{T} x_{ist} = 1 \quad s = 1, \dots, S \quad t = 1, \dots, T \tag{61}$$

$$\sum_{t=1}^{T} x_{ist} = 1 \quad i = 1, \dots, T \quad s = 1, \dots, S \tag{62}$$

$$x_{ist} \in \{0, 1\} \quad Z \ \ge 0 \quad Z \in \mathbb{R} \tag{63}$$

gde je x_{ist} binarna promenljiva, koja ima vrednost jedan ukoliko je stavka i iz skupa s dodeljena testu t, ili nula ukoliko nije, Z je realna promenljiva čija se minimalna vrednost traži. Minimizacija vrednosti Z, zajedno sa ograničenjem (60), koje obezbeđuje da je Z veće ili jednako težini najtežeg testa, osigurava da testovi imaju balansirane težine. w_{is} je težinski koeficijent stavke i iz skupa s definisana formulom (57). Ograničenje (61) obezbeđuje da je svakom od T testova pridružena tačno po jedna stavka iz jednog od S skupova (što sprečava da više od jedne stavke iz jednog skupa bude dodeljeno istom testu), dok ograničenje (62) osigurava da je svaka od T stavki iz jednog od od S

skupova dodeljena tačno jednom od T testova (što sprečava da ista stavka iz bilo kog skupa bude dodeljena više od jednom testu). Drugim rečima, ograničenja (61) i (62) obezbeđuju da u svakom testu bude po jedna stavka iz svakog skupa, odnosno da sve stavke iz jednog skupa budu raspoređene svaka u po jedan test. Ograničenje da je $Z \geq 0$ (63) se pojavljuje u originalnoj formulaciji (Brusco, Köhn, & Steinley, 2013), ali je već zadovoljeno na osnovu ograničenja (60) i činjenicom da je težinski koeficijent stavke, po svojoj prirodi, vrednost veća od nule.

U ovom radu je problem rešavan heuristikom zasnovanom na SA metaheuristici (odeljak 3.2.2.1).

Ista formulacija je korištena u radu (Pereira & Vila, 2015) i problem je rešavan heuristikom na osnovu VNS metaheuristike (odeljak 3.2.2.2).

3.1.1.1.3 Maksimizacija pouzdanosti testa

Radovi koji se razmatraju u ovom odeljku imaju cilj da maksimizuju CTT pouzdanost rezultata testa. Pouzdanost rezultata testa se definiše kao kvadrat korelacionog koeficijenta između rezultata testa i tačnog rezultata testa (2). Ova veličina ne može da se maksimizuje direktno, pa se maksimizuje donja granica pouzdanosti testa (Krombahova alfa) definisana formulom (11).

Maksimizacija koeficijenta α podrazimeva minimizaciju

$$\frac{\sum_{i=1}^{n} \sigma_i^2}{(\sum_{i=1}^{n} \sigma_i d_i)^2} \tag{64}$$

pod uslovom da je ograničen broj stavki u testovima (n je fiksno).

Na osnovu prethodno navedenog, problem može da se formuliše na sledeći način (Adema & van der Linden, 1989):

Promenljiva: $x_i \quad i = 1,2,\ldots,I$ (65)

Funkcija cilja: $$min \frac{\sum_{i=1}^{n} \sigma_i^2 x_i}{(\sum_{i=1}^{n} \sigma_i d_i\, x_i)^2} \tag{66}$$

Ograničenja: $$\sum_{i=1}^{I} x_i = n \tag{67}$$

$$x_i \in \{0,1\} \tag{68}$$

Pošto je funkcija cilja nelinearna funkcija, dok su funkcije u imeniocu i brojiocu kvadratna i linearna funkcija promenljivih, u prethodno pomenutom radu je predloženo da se jedna funkcija koristi kao funkcija cilja dok se druga koristi kao ograničenje i tako se dobija Model 1:

Promenljiva: $x_i \quad i = 1,2,\ldots,I$ (69)

Funkcija cilja: $$max \sum_{i=1}^{n} \sigma_i d_i\, x_i \tag{70}$$

Ograničenja: $$\sum_{i=1}^{n} \sigma_i^2 x_i \leq c \tag{71}$$

$$\sum_{i=1}^{I} x_i = n \tag{72}$$

$$x_i \in \{0,1\} \quad c > 0 \tag{73}$$

gde je c konstanta veća od nule. Vrednost konstante c (maksimalna suma varijansi odgovora na stavke) se određuje empirijski.

Pošto je maksimizacija diskriminativnosti stavke važnija od varijanse odgovora na stavku, Model 2 upravo to uzima u obzir:

$$\text{Promenljiva:} \quad x_i \quad i = 1,2,\ldots,I \tag{74}$$

$$\text{Funkcija cilja:} \quad max \sum_{i=1}^{n} d_i\, x_i \tag{75}$$

$$\text{Ograničenja:} \quad \sum_{i=1}^{I} x_i = n \tag{76}$$

$$x_i \in \{0,1\} \tag{77}$$

Prednost ovog modela je u tome što ne treba da se bira vrednost konstante c.

U radu (Armstrong, Douglas, & Wang, 1994) je korišćen Model 2. Problem je podeljen u dva potproblema. Prvi potproblem je rešavan kao problem maksimalnog mrežnog protoka (odeljak 2.2.2.2) sa ciljem da se kreira n podskupova sa T stavki, tako da stavke pripadaju istoj kategoriji, i u isto vreme da se maksimizuje ukupna suma diskriminativnost izabranih stavki. Izvorišni čvorovi su stavke iz banke stavki, a odredišni čvorovi su podskupovi stavki gde svaki podskup odgovara jednoj kategoriji. Težina grane je jednaka vrednosti diskriminativnosti stavke koja je u izvorišnom čvoru.

Prvi potproblem je rešavan poznatim algoritmom za rešavanje problema mrežnog protoka pod nazivom algoritam najkraćeg puta povećanja (Kennington & Wang, 1992). Drugi potproblem je dodela stavki iz odredišnih čvorova testovima. Ovaj problem je NP-težak i rešavan je heuristikom opisanom u radu (Armstrong, Jones, & Wu, 1992) (odeljak 3.2.1.2).

3.1.1.2 Ciljevi teorije odgovora na stavku

Rad (Samejima, 1977) definiše „slabo“ paralelne testove kao testove koji imaju identične samo informacione funkcije (TIF), dok rad (Lord, 1980) definiše „snažno“ paralelne testove kao testove koji imaju jednaku dužinu testa i istu karakterističnu funkciju testa (TCF). Ove definicije paralelizma se ne slede uvek. Za paralelne testove se obično očekuje da je TIF svakog od njih približno jednaka ciljnoj informacionoj funkciji testa (TTIF) ili da njihova TIF ima približno isti oblik kao i TTIF uz maksimizaciju vrednosti TIF.

3.1.1.2.1 Minimizacija broja stavki u testu

U radovima (Theunissen, 1985) i (Theunissen, 1986) se problem ASPT tretirao kao problem multidimenzionalnog KP. Cilj je da se minimizuje broj stavki u testu, dok TIF mora da bude veći od TTIF na istom nivou sposobnosti. Problem se formuliše kao 0-1 LP problem.

Promenljiva: $x_{it} \quad i = 1,2,\ldots,I \quad t = 1,\ldots,T$ (78)

Funkcija cilja: $$min \sum_{t=1}^{T} \sum_{i=1}^{I} x_{it} \tag{79}$$

Ograničenja: $$\sum_{i=1}^{I} IIF_i(\theta_k) x_{it} \geq TTIF(\theta_k) \quad k = 1,2,\ldots,K \quad t = 1,\ldots,T \tag{80}$$

$$x_{it} \in \{0,1\} \tag{81}$$

Kreatori testa obično žele da utiču na broj stavki u testu, tako da je ova formulacija suviše ograničavajuća.

3.1.1.2.2 Minimizacija razlike između TIF i TTIF

U radu (Boekkooi-Timminga, 1987) se daje jedna od prvih 0-1 LP formulacija problema ASPT, sa ciljem da se minimizuju razlike između TIF-ova simultano sastavljenih testova i TTIF-a na svim traženim nivoima sposobnosti (θ_k). Ovaj zahtev je ekvivalentan zahtevu da se minimizuje zbir TIF-ova testova na svim nivoima sposobnosti (83), pri ograničenju da svaki TIF testa treba da bude veći od vrednosti TTIF na istom nivou sposobnosti, formula (84). Ovaj model je poznat pod nazivom minimaks model (MI).

Metod se formalno definiše na sledeći način:

Promenljiva: $x_{it} \quad i = 1,2,\ldots,I \quad t = 1,\ldots,T$ (82)

Funkcija cilja: $$min \sum_{t=1}^{T} \sum_{k=1}^{K} \sum_{i=1}^{I} IIF_i(\theta_k) x_{it} \tag{83}$$

Ograničenja: $$\sum_{i=1}^{I} IIF_i(\theta_k) x_{it} \geq TTIF(\theta_k) \quad k = 1,2,\ldots,K \quad t = 1,\ldots,T \tag{84}$$

$$x_{it} \in \{0,1\} \tag{85}$$

Problem se rešava pomoću algoritma koji je zasnovan na metodu grananja i ograničavanja (B&B) (Land & Doig, 1960) (odeljak 3.2.1.1).

Adema je u svom radu (Adema, 1992) predložio MZOLP formulaciju za simultano sastavljanje testova. Promenljive koje se koriste u ovoj formulaciji sadrže razlike između TIF testa i TTIF za svaki nivo sposobnosti θ_k i definišu se na sledeći način:

$$u_k = max[0, TTIF(\theta_k) - TIF(\theta_k)] \tag{86}$$

$$v_k = max[0, TIF(\theta_k) - TTIF(\theta_k)] \tag{87}$$

$$w = \max_k (u_k + v_k) = max\left(\max_k u_k + \max_k v_k\right) \tag{88}$$

gde promenljiva u_k označava razliku između TIF i TTIF, kada je TTIF veći od TIF, a promenljiva u_k označava razliku za slučaj kada je TIF veći od TTIF. Promenljiva w sadrži najveću apsolutnu razliku između TIF i TTIF na svim nivoima sposobnosti θ_k. Cilj u prvoj fazi predloženog algoritma

jeste da se kreira jedan test veličine $T{\cdot}n$. U drugoj fazi rešavanja problema, sastavljen test sa $T{\cdot}n$ stavki deli se na testove sa n stavki.

Formulacija problema koji se rešava u prvoj fazi može da se prikaže na sledeći način:

Promenljive: $x_i, u_k, v_k, w \quad i = 0,1,..,I \quad k = 1,2,\ldots,K$ (89)

Funkcija cilja: $min\, w$ (90)

Ograničenja: $$\sum_{i=1}^{I} IIF_i(\theta_k)x_i + u_k - v_k = T \cdot TTIF(\theta_k) \quad k = 1,2,\ldots,K \tag{91}$$

$$\sum_{i=1}^{I} x_i = T \cdot n \tag{92}$$

$$x_i \in \{0,1\} \quad u_k, v_k \geq 0 \tag{93}$$

Cilj je da se minimizuje najveća apsolutna razliku između TIF i TTIF među svim nivoima sposobnosti θ_k (90). Ograničenje (91) pokazuje da je razlika u_k - v_k jednaka razlici TIF testa i TTIF.

Prva faza problema se rešava tako što se sastavlja jedan test veličine T·n (formule 89-93). U drugoj fazi se stavke iz testa kreiranog u prvoj fazi dodeljuju testovima. Za rešavanje druge faze problema se koristi heuristika namenjena za rešavanje problema rasporeda poslova (eng. *Job-Shop Schedulig Problem,* JSSP) koja je zasnovana na heuristici predstavljenoj u radu (Coffman Jr, Lueker, & Rinnooy Kan, 1988) (odeljak 3.2.1.2).

Van der Linden i Adema (van der Linden & Adema, 1998) su predložili model za redno sastavljanje paralelnih testova i ujedno rešavali problem neujednačenosti testova koji postoji pri rednom sastavljanju testova. Cilj je da se balansira kvalitet testova, tako da kvalitet kasnije sastavljenih testova ne opada u odnosu na ranije sastavljene testove. Za svaki test koji se sastavlja, sastavlja se i test-senka. Test-senka se kreira da bi se održao balans između kvaliteta trenutnog testa i kasnije sastavljenih testova. Sve stavke koje su se izabrale za test-senku se, posle postupka sastavljanja, vraćaju u banku stavki. Ovaj model je poznat i pod nazivom velika test-senka (eng. *Big Shadow Test*, BST).

Problem sastavljanja paralelnih testova se sastoji iz serije rešavanja problema simultanog sastavljanja dva testa, koji je formulisan na sledeći način (MZOLP):

Promenljive: $x_i, z_i, y \quad i = 1,2,\ldots,I$ (94)

Funkcija cilja: $min\, y$ (95)

Ograničenja: $$\left|\sum_{i=1}^{I} IIF(\theta_k)x_i - TTIF_t(\theta_k)\right| \leq y \quad k = 1,\ldots,K \quad t = 1,..,T \tag{96}$$

$$\left|\sum_{i=1}^{I} IIF(\theta_k)z_i - TTIF_d(\theta_k)\right| \leq y \quad k = 1,\ldots,K \tag{97}$$

$$x_i \in \{0,1\} \quad z_i \in \{0,1\} \quad y \geq 0 \tag{98}$$

gde je y maksimalna apsolutna razlika TIF i TTIF na svim diskretnim vrednostima sposobnosti θ_k, z_i slučajna promenljiva koja označava prisustvo stavke i u test-senci, $TTIF_d$ označava $TIFF$ za test-senku.

Problem je rešavan postojećim softverom koji koristi B&B algoritam i princip prvog prihvatljivog celobrojnog rešenja (Timminga, van der Linden, & Schweizer, 1996) (odeljak 3.2.1.1).

U radu (Luecht, 1998) je predstavljena formulacija sa ciljem da se minimizuje apsolutno rastojanje ($dist_i$) od TTIF, nazvana model ponderisanog apsolutnog odstupanja (eng. *Weighted Absolute Deviation Model*, WADM). Ovo rastojanje figuriše u promenljivom težinskom koeficijentu (e_i), formula (103). U predloženom metodu, funkcija cilja se menja u svakom koraku i problem se rešava za svaku stavku koja se dodaje u test. Atributi prethodno odabranih stavki se sadrže u funkciji cilja (100) u svakom koraku.

Problem je formulisan na sledeći način. Za odabir stavke j, kada je izabrano j-1 stavki (j = 1,..., n), sledeći problem se rešava:

$$\text{Promenljiva:} \quad x_i \quad i = 1,2,\ldots,I \tag{99}$$

$$\text{Funkcija cilja:} \quad max \sum_{i=1}^{I} e_i x_i \tag{100}$$

$$\text{Ograničenja:} \quad \sum_{i=1}^{I} x_i = j \tag{101}$$

$$x_{i_1} = x_{i_2} = \ldots = x_{i_{j-1}} = 1 \tag{102}$$

gde je e_i promenljivi težinski koeficijent definisan na sledeći način (103):

$$e_i = 1 - \frac{\sum_{k=1}^{K} dist_i(\theta_k)}{\sum_{i \in S_{j-1}} \sum_{k=1}^{K} dist_i(\theta_k)} \quad i \in S_{j-1} \tag{103}$$

$$dist_i(\theta_k) = \left| \frac{TTIF(\theta_k) - \sum_{j=1}^{I} IIF_j(\theta_k) x_j}{n - j + 1} - IIF_i(\theta_k) \right| \quad i \in S_{j-1} \tag{104}$$

$$x_i \in \{0,1\} \tag{105}$$

gde je S_{j-1} skup indeksa za preostale stavke iz banke stavki, nakon što je izabrano j - 1 stavki. Za simultano sastavljanje testova, postoje odvojene funkcije cilja za svaki test i stavke se simultano sastavljaju.

Problem se u navedenom radu rešava koristeći pohlepni algoritam (eng. *greedy algorithm*) (odeljak 3.2.1.2).

Formula (104) prethodno prikazane formulacije, se koristi u radu (Songmuang & Ueno, 2011) za računanje verovatnoće da stavka bude izabrana u predloženom pčelinjem algoritmu (eng. *Bees algorithm*, BA), što će biti objašnjeno u odeljku 3.2.2.5. Cilj u tom radu je da se minimizuje prosečna razlika između TIF sastavljenih testova i TTIF, kao i da se dobije maksimalan broj testova iz banke stavki na raspolaganju.

U radu (Sun, Chen, Tsai, & Cheng, 2008) je predložen metod za simultano sastavljanje testova tako da se minimizuje ukupno kvadratno rastojanje između TIF svakog testa koji se sastavlja i TTIF, sumirano po svim diskretnim vrednostima sposobnosti θ_k (106).

Matematička formulacija je jednostavna:

$$\text{Funkcija cilja:} \quad min \sum_{k=1}^{K} \left(TIF_t(\theta_k) - TTIF(\theta_k)\right)^2 \quad t = 1,..,T \tag{106}$$

gde je $TIF_t(\theta_k)$ informaciona funkcija testa t koji se analizira.

Problem se rešava genetskim algoritmom (GA) (odeljak 3.2.2.4).

U radu (Chang & Shiu, 2012) se koristi isti cilj kao i u radu (Boekkooi-Timminga, 1987), a to je da se minimizuje razlika između TIF i TTIF, odnosno da se minimizuje zbir TIF-ova na svim nivoima sposobnosti θ_k, pri ograničenju da svaki TIF treba da bude veći od vrednosti TTIF na istom nivou sposobnosti. Predložena je ILP formulacija je za sastavljanje jednog testa.

Problem se matematički formuliše na sledeći način:

$$\text{Promenljive:} \quad y_i,\ c_{y_i h},\ s_{y_i v},\ q_{y_i m} \quad i = 1,2,\dots,n \quad h = 1,\dots,H \tag{107}$$

$$v = 1,2,\dots,V \quad m = 1,\dots,M$$

$$\text{Funkcija cilja:} \quad min \sum_{k=1}^{K} \sum_{i=1}^{I} IIF_{y_i}(\theta_k) \tag{108}$$

$$\text{Ograničenja:} \quad \sum_{i=1}^{I} IIF_{y_i}(\theta_k) \geq TTIF(\theta_k) \quad k = 1,2,\dots,K \tag{109}$$

$$\sum_{i=1}^{n} c_{y_i h} = C_h \quad h = 1,\dots,H \tag{110}$$

$$\sum_{i=1}^{n} s_{y_i v} = S_v \quad v = 1,2,\dots,V \tag{111}$$

$$\sum_{i=1}^{n} q_{y_i m} = Q_m \quad m = 1,2,\dots,M \tag{112}$$

$$\sum_{i=1}^{n} r_{y_i} \leq R^u \quad k = 1,2,\dots,K \tag{113}$$

$$y_i \in [1, I] \quad i = 1,2,\dots,n \quad c_{y_i h}, s_{y_i v}, q_{y_i m} \in \{0,1\} \tag{114}$$

U ovoj formulaciji su promenljive definisane na sledeći način: promenljiva y_i predstavlja redni broj stavke u banci stavki, $c_{y_i h}$ je binarna promenljiva koja ima vrednost jedan ukoliko stavka sa rednim brojem y_i pripada oblasti h, a vrednost nula ukoliko ne pripada, $s_{y_i v}$ je binarna promenljiva

koja ima vrednost jedan ukoliko stavka sa rednim brojem y_i meri veštinu v, a vrednost nula ukoliko je ne meri, $q_{y_i m}$ je binarna promenljiva koja ima vrednost jedan ukoliko stavka sa rednim brojem y_i pripada vrsti stavke m. C_h je ukupan broj stavki za h-tu oblast, S_v je ukupan broj stavki za v-tu veštinu, Q_m je ukupan broj stavki za m-tu vrstu stavki, r_{y_i} je broj reči stavke sa rednim brojem y_i i R^u je gornja granica za ukupan broj reči u testu.

Predstavljena formulacija problema je korišćena u radu (Chang & Shiu, 2012) za simultano kreiranje paralelnih testova koristeći CLONALG algoritam opisan u odeljku 3.2.2.7.

3.1.1.2.3 Maksimizacija TIFa dok se održava relativni oblik ciljnog TTIFa

U radu (van der Linden & Boekkooi-Timminga, 1989) autori su predložili korišćenje relativnog oblika TTIF uzimajući u obzir da sastavljači testa često ne umeju da zadaju TTIF u apsolutnim vrednostima za svaki nivo sposobnosti. Njima je mnogo jednostavnije da zadaju samo relativni oblik TIFF funkcije. Da bi se rešio ovaj problem, predložena je formulacija da se maksimizuje TIF za svaki nivo sposobnosti, dok se relativni oblik TIF održava u skladu sa zadatim oblikom TTIF.

Problem se matematički formuliše na sledeći način:

$$\text{Promenljive:} \quad y, x_i \quad i = 1,2,\ldots,I \tag{115}$$

$$\text{Funkcija cilja:} \quad max\, y \tag{116}$$

$$\text{Ograničenja:} \quad \sum_{i=1}^{I} IIF_i(\theta_k)x_i - r_k y \geq 0 \quad k = 1,2,\ldots,K \tag{117}$$

$$x_i \in \{0,1\} \quad y \geq 0 \tag{118}$$

Relativni oblik TTIF je opisan donjim granicama $r_k y$, gde je r_k relativna vrednost TTIF na nivou sposobnosti θ_k, a y je multiplikativna promenljiva čija se vrednost maksimizuje.

Ova formulacija je primenjena u radu (Boekkooi-Timminga, 1990) za redno i simultano sastavljanje paralelnih testova. Za redno sastavljanje testova je dodato još jedno ograničenje, koje određuje koliko sme da se TIF novog testa razlikuje od prethodno sastavljenih testova i predstavljeno je formulom:

$$\sum_{i=1}^{I} IIF_i(\theta_k)x_i - (1 - pct) \cdot TIF_t(\theta_k) \geq 0 \quad t = 1,\ldots,t^* \quad k = 1,\ldots,K \tag{119}$$

gde t označava indeks u skupu već sastavljenjih testova, t^* je indeks poslednjeg sastavljenog testa, i pct je dozvoljena relativna razlika u vrednosti TIF, koja ima vrednost u opsegu [0,1].

Predložena formulacija se lako prilagođava za simultano sastavljenje testova, tako što se promenljiva x_i zamenjuje sa promenljivom x_{it} i dodaje se ograničenje da se stavka može nalaziti u samo jednom testu.

Ova formulacija se koristila i u radu (Adema, Boekkooi-Timminga, & van der Linden, 1991) i rešavala heuristikom predloženom u (Adema, 1988). Ta heuristika se detaljno opisuje u odeljku 3.2.1.2.

3.1.1.2.4 Minimizacija odstupanja od ograničenja

Rad (Swanson & Stocking, 1993) se bavi problemom sastavljanja testa kada ne mogu sva ograničenja da budu zadovoljena. Ovaj problem može da se pojavi kada je veoma veliki broj ograničenja, ili banka stavki nema dovoljno stavki koje imaju potrebne vrednosti atributa da bi se zadovoljila sva ograničenja. Autori tretiraju ograničenja kao „poželjne osobine“. U njihovom

predlogu formulacije, odstupanjima atributa od graničnih vrednosti se dodeljuju težinski koeficijenti na osnovu prioriteta i takva ponderisana odstupanja se uključuju u definiciju funkcije cilja. Cilj je da se minimizuje suma odstupanja od ograničenja (121). Ovaj model je poznat pod nazivom model ponderisanog odstupanja (eng. *Weighted Deviation Model*, WDM).

Matematička formulacija problema je sledeća:

$$\text{Promenljiva:} \quad x_i \in \{0,1\} \quad i = 1,2,\ldots,I \tag{120}$$

$$\text{Funkcija cilja:} \quad min\left\{\sum_{m=1}^{M} w_m d_{L_m} + \sum_{m=1}^{M} w_m d_{U_m}\right\} \tag{121}$$

$$\text{Ograničenja:} \quad \sum_{i=1}^{I} v_{im} x_i + d_{L_m} - e_{L_m} = L_m \quad \mathrm{m} = 1,2,\ldots,M \tag{122}$$

$$\sum_{i=1}^{I} v_{im} x_i - d_{U_m} + e_{U_m} = U_m \quad \mathrm{m} = 1,2,\ldots,M \tag{123}$$

$$x_i \in \{0,1\} \quad d_{L_m}, d_{U_m}, e_{L_m}, e_{U_m} \geq 0 \tag{124}$$

gde je M ukupan broj atributa (i pridruženih ograničenja), w_m je težinski koeficijent koji se koristi da se prikaže značaj atributa stavke na funkciju cilja, v_{im} označava vrednost atributa m stavke i, d_{L_m} je pozitivno odstupanje od donje granice za atribut m kada je donja granica L_m podbačena (inače je 0) i e_{L_m} je pozitivno odstupanje od donje granice za atribut m kada je donja granica L_m premašena (inače je 0). Vrednosti e_{U_m} i d_{U_m} imaju odgovarajuću interpretaciju za gornju granicu U_m.

Ova formulacija se koristi za redno sastavljanje paralelnih testova u radu (Stocking, Swanson, & Pearlman, 1993). Usklađenost sa TTIF-om na istom nivou sposobnosti je predstavljeno ograničenjima (125) i (126).

$$\text{Ograničenja:} \quad \sum_{i=1}^{I} IIF_i(\theta_k) x_i + d_{L_k} - e_{L_k} = TTIF_L(\theta_k) \quad \mathrm{k} = 1,2,\ldots,K \tag{125}$$

$$\sum_{i=1}^{I} IIF_i(\theta_k) x_i - d_{U_k} + e_{U_k} = TTIF_U(\theta_k) \quad \mathrm{k} = 1,2,\ldots,K \tag{126}$$

gde su $TTIF_L$ i $TTIF_U$ donja i gornja granica TIFa za određenu sposobnost.

Za rešavanje problema su koristili heuristiku zasnovanu na pohlepnom algoritmu (odeljak 3.2.1.2).

3.1.1.2.5 Maksimizacija prosečne diskriminativnosti testa

U radu (Nguyen, Hui, & Fong, 2013) se predlaže formulacija APTA tretirajući problem kao multidimenzionalni KP problem (odeljak 2.2.2.1.1). Autori smatraju da je lakše koristiti diskriminativnost stavke nego IIF i zato su predložili formulaciju čiji je cilj da se maksimizuje srednja diskriminativnost testa.

Matematička formulacija problema je relativno jednostavna:

Promenljiva: $x_i \quad i = 1,2,\ldots,I$ (127)

Funkcija cilja: $max \frac{\sum_{i=1}^{I} a_i x_i}{n}$ (128)

Ograničenja: $$\sum_{i=1}^{I} x_i = n$$ (129)

$$x_i \in \{0,1\}$$ (130)

Za rešavanje problema je u navedenom radu (Nguyen, Hui, & Fong, 2013) predložena heuristika zasnovana na algoritmu grananja i ograničavanja (odeljak 3.2.1.1). Autori su utvrdili da se ova formulacija može proširiti za simultano sastavljanje paralelnih testova.

3.1.2 Uparivanje sa srodnim testovima

U ovoj kategoriji je cilj da se minimizuje Euklidsko rastojanje od test-klice, ili da se ono drži pod kontrolom. Paralelni testovi automatski nasleđuju psihometrijske i nepsihometrijske osobine od test-klice. Ovakvi pristupi su korisni zato što ponovo koriste već kreirane testove za koje je značajan napor uložen u njihovo sastavljanje i proveru kroz sprovođenje i merenje statistike rezultata.

3.1.2.1 Uparivanje sa zadatom test-klicom

U radu (van der Linden & Boekkooi-Timminga, 1988) je predložena automatizacija metoda *nasumičnog uparivanja podtestova* (eng. *Matched Random Subtests Method*, MRSM), formulisanog u radu (Gulliksen, 1950), za podelu testa u dve paralelne polovine. Rešavanje problema se obavlja u dve faze. U prvoj fazi se rešava problem uparivanja stavki iz dve polovine testa na osnovu njihove Euklidske razdaljine (131) sa ciljem da se minimizuje ukupna razdaljina i da se kao rezultat dobiju parovi stavki.

$$dist\ (i_1, i_2) = \sqrt{\left(p_{i_1} - p_{i_2}\right)^2 + \left(d_{i_1} - d_{i_2}\right)^2} \quad i_1 \neq i_2. \tag{131}$$

Problem se formuliše na sledeći način:

Promenljiva: $x_{i_1 i_2} \quad i_1 = 1,2,\ldots,n-1 \quad i_2 = i_1 + 1,\ldots,$ n (132)

Funkcija cilja: $$min \sum_{i_1=1}^{n-1} \sum_{i_2=i_1+1}^{n} dist\ (i_1, i_2) x_{i_1 i_2}$$ (133)

Ograničenja: $$\sum_{i_1=1}^{i_2-1} x_{i_1 i_2} + \sum_{i_1=i_2+1}^{n} x_{i_2 i_1} = 1 \quad i_1, i_2 = 1,2,\ldots,n$$ (134)

$$x_{i_1 i_2} \in \{0, 1\} \quad i_1 = 1,2,\ldots,n-1 \quad i_2 = i_1 + 1,\ldots,n$$ (135)

gde je $x_{i_1 i_2}$ ima vrednost jedan ukoliko su stavke i_1 i i_2 uparene, a vrednost nula ukoliko nisu. Ograničenje (134) obezbeđuje da se svaka stavka pojavi u tačno jednom paru. Cilj (133) je da se minimizuje suma rastojanja stavki u svim parovima stavki.

Nakon uparivanja stavki, paralelnim polovinama se dodeljuju stavke. Stavke mogu da se dodele polovinama ili nasumično ili na osnovu nekog drugog kriterijuma kao što je jednakost prosečnog rezultata testa ili variansi rezultata. Ukoliko se uparene stavke ne dodeljuju nasumično, onda drugi korak podrazumeva rešavanje još jednog kombinatorno optimizacionog problema.

Problem je rešavan softverom koji je implementirao algoritam opisan u radu (Land & Doig, 1960).

Rad (Armstrong, Jones, & Wu, 1992) rešava sličan problem. Postoji sastavljena test-klica i testovi se kreiraju tako da se mimimizuje Euklidska razdaljina testa od test-klice. Problem se rešava u dve faze. U prvoj fazi se rešava problem uparivanja $n \cdot T$ stavki iz banke stavki sa n stavki iz test-klice. Ova faza se rešava po analogiji rešavanja problema mrežnog protoka. Svaka stavka iz banke stavki predstavlja izvorišni čvor. Odredišni čvor je podskup stavki iz banke stavki koje odgovaraju stavki-klici (stavki iz test-klice). Svaki odredišni čvor će imati, na kraju, pridruženih T stavki iz banke stavki. Cena transporta iz izvorišnog u odredišni čvor jednaka je Euklidskoj razdaljini između stavke u izvorišnom čvoru i stavke-klice. Euklidska razdaljina može da se računa ili po CTT metrici (136) ili po IRT metrici (137) na sledeći način:

$$d_{ij}^{CTT} = \sqrt{w_1\left(p_i - p_j\right)^2 + w_2\left(d_i - d_j\right)^2} \quad w_1 + w_2 = 1 \tag{136}$$

$$d_{ij}^{IRT} = \sqrt{w_1\left(a_i - a_j\right)^2 + w_2\left(b_i - b_j\right)^2 + w_3\left(c_i - c_j\right)^2} \quad w_1 + w_2 + w_3 = 1 \tag{137}$$

gde su w_1, w_2, w_3, težinski koeficijenti koji označavaju važnost atributa stavke kome su pridruženi.

Rešenje problema definisanog u prvoj fazi može da se reši efikasnim algoritima za rešavanje problema mrežnog protoka (Ahuja, Magnanti, & Orlin, 1993). Drugi problem je raspodela stavki iz odredišnih čvorova u testove. Predložena je heuristika koja je zasnovana na heuristici predloženoj u radu (Ackerman, 1989).

3.1.2.2 Uparivanje sa skupom test-klica

U radu (Ignjatović, Bojić, Furlan, & Tartalja, 2015) je ispitivan potencijal rešavanja problema ASPT uparivanjem sastavljanog testa sa skupom test-klica. Metod je zasnovan na pronalaženju znanja o testovima implicitno sadržanom u banci već održanih testova (skup test-klica). Pristup koji se koristi jeste predviđanje zasnovano na algoritmu k-najbližih suseda (eng. *k-nearest neighbors*, KNN) sa ciljem da se pronađe test koji se na zadovoljavajući način uklapa u skup test-klica. Skup test-klica koji su ranije kreirani (ručno ili automatski), održani i izmereni, čine referentni skup testova. Metod ne traži optimalno već zadovoljavajuće rešenje, koje nije „gore“ od test-klica u referentnom skupu. Predloženi metod nije osetljiv na način kreiranja testova koji se nalaze u banci testova. Pored testova, u banci se nalaze i stavke opisane svojim atributima.

Osnovna ideja metoda jeste da se nađe skup stavki za sastavljani test koje zadovoljavaju ograničenja, i čije rastojanje od test-klica iz skupa se nalazi u granicama međusobnih rastojanja test-klica unutar referentnog skupa. Rastojanje između dva testa se računa na osnovu atributa testa. Svaki test se opisuje skupom atributa i matricom čiji elementi predstavljaju podtestove. Podtest, u ovom kontekstu, je skup stavki iste vrste (tipa pitanja) i iz iste kategorije (oblasti, teme). Test je opisan sledećim atributima: broj stavki, vrsta testa (kolokvijum, završni ispit), minimalna težina, maksimalna težina, srednja težina. Stavka ima sledeće atribute: vrsta stavke, težina i pripadnost kategoriji. Rastojanje između dva testa uzima u obzir, sa jednakim udelima, rastojanje mereno atributima testa i rastojanje mereno atributima stavki podtestova. Ono se računa pomoću formule (138).

$$d^{Total}(x^*, y^*) = \frac{1}{2} d^{test} + \frac{1}{2 \cdot v \cdot m} \sum_{j=1}^{v*m} d_j^{subtest} \tag{138}$$

gde je v broj vrsta stavki a m je ukupan broj kategorija. Euklidsko rastojanje se izračunava na osnovu formule (139).

$$d(x^*, y^*) = \sqrt{\sum_{i=1}^{n} w_i (x_i^* - y_i^*)^2} \tag{139}$$

gde su x* i y* vektori vrednosti normalizovanih atributa dva testa za koje se razdaljina izračunava, a w_i je težinski koeficijent atributa i ($\sum w_i = 1$), preko kojeg sastavljač testa može da upravlja značajem atributa u određivanju rastojanja. Paralelni testovi mogu da se sastavljaju redno bez opadanja u kvalitetu sledeće kreiranih testova.

U radu (Ignjatović, Bojić, Furlan, & Tartalja, 2015) nisu razmatrana moguća heuristička rešenja problema, već je samo procenjivan potencijal opisane formulacije metoda.

3.1.3 Maksimizacija broja paralelnih testova

Maksimizacije broja paralelnih testova se koristi sa ciljem dobrog iskorišćenja banke stavki. Dobro iskorišćenje banke stavki najčešće podrazumeva da testovi dele stavke, i tada se kaže da su testovi *preklapajući* (Songmuang & Ueno, 2011). Međutim, moguće je da se pronađe maksimalan broj testova tako da ne dele stavke, odnosno da budu *nepreklapajući* (Belov & Armstrong, 2006).

U radu (Belov & Armstrong, 2006) su proučavali problem maksimizacije broja testova po analogiji sa problemom maksimalnog pakovanja skupa (MSPP) (odeljak 2.2.2.1.3). Razmatran je model testa LSAT (eng. *Law School Admissions Test*) koji meri nekoliko sposobnosti: analitičko zaključivanje, logičko zaključivanje i razumevanje čitanja. Na osnovu tog modela se kreira test koji se sastoji iz podtestova, gde svaki podtest meri po jednu od prethodno navedenih sposobnosti. Podtest se sastoje iz preambule (uvodni materijal na koji se odnose stavke u podtestu) i sadrži jednu ili više stavki na koje se daje odgovor. Preambula može da bude uključena u više podtestova i tada bi ti podtestovi bili preklapajući. Testovi koji sadrže preklapajuće podtestove su takođe preklapajući. U jednom testu ne mogu da postoje preklapajući podtestovi, jer nema razloga da se ista preambula ponovi više puta u jednom testu.

Pretpostavka je da postoji banka stavki i preambula, ukupan broj sposobnosti koji se meri podtestovima (K), K specifikacija podtestova, maksimalan broj ponavljanja podtesta u testovima (U), kao i unapred definisani broj preklapajućih podtestova za svaku od sposobnosti (L). Inicijalno se za svaku sposobnost (k = 1,...,K) kreira L preklapajućih podtestova tako da zadovoljavaju specifikaciju podtesta za sposobnost k. Nakon toga se iz L preklapajućih podtestova, izdvaja maksimalni broj nepreklapajućih podtestova koji mere jednu sposobnost (rešava se prvi problem). Ovaj problem se formuliše kao MSPP. Kada se postupak ponovi K puta, od dobijenih podtestova se kreiraju preklapajući testovi tako da svaki test ima K podtestova, kao i da se svaki podtest ne ponovi više od U puta u testovima. Zatim se od dobijenih preklapajućih testova izdvaja maksimalan broj nepreklapajućih testova rešavajući MSPP (drugi problem).

U nastavku se formuliše prvi problem MSPP iz prethodno opisanog postupka.Pretpostavka je da je dobijeno L preklapajućih podtestova koji zajedno sadrže M preambula. Preklapajući podtest je predstavljen binarnom promenjivom p_j koja ima vrednost jedan ukoliko je podtest deo rešenja, ili vrednost nula ukoliko nije. x_{ij} je binarna promenljiva koja predstavlja preambulu, i ima vrednost jedan ukoliko podtest j sadrži preambulu i, ili nula u suprotnom.

Problem se formuliše na sledeći način:

$$\text{Promenljive:} \quad p_j, x_{ij} \quad j = 1,2,\dots,L \quad i = 1,\dots,M \tag{140}$$

$$\text{Funkcija cilja:} \quad max \sum_{j=1}^{L} p_j \tag{141}$$

$$\text{Ograničenja:} \quad \sum_{j=1}^{L} x_{ij} p_j \leq 1 \quad i = 1,\dots,M \tag{142}$$

$$x_{ij}, p_j \in \{0, 1\} \quad i = 1,2,\dots,M \quad j = 1,2,\dots,L \tag{143}$$

gde je cilj da se maksimizuje broj nepreklapajućih podtestova (141). Ograničenje (142) osigurava da je svaka preambula sadržana u najviše jednom podtestu. Opisana formulacija se primenjuje i za kreiranje maksimalnog broja nepreklapajućih testova gde testovi predstavljaju podtestove, a podtestovi predstavljaju preambule.

Podtestovi su sastavljeni koristeći *Monte Carlo* pristup, gde se stavke nasumično generišu. Ukoliko stavke u podtestu zadovoljavaju ograničenja specifikacije podtesta, onda je on zapamćen kao novi podtest (Belov & Armstrong, 2005).

Za reševanje MSP problema autori rada (Belov & Armstrong, 2006) su predložili heuristiku koja je zasnovana na B&B algoritmu i jednostavnoj heuristici sa elementima tabu pretrage (odeljak 3.2.1.1).

U radu (Ishii, Songmuang, & Ueno, 2014) su rešavali problem maksimizacije broja paralelnih testova koji mogu da se kreiraju na osnovu postojeće banke stavki i tako ona dobro iskoristi. Za povećanje iskorišćenja banke stavki, predloženo je da se ukloni ograničenje da stavke mogu pripadati samo jednom testu. Problem je predstavljen kao problem maksimalne klike (MCP) (odeljak 2.2.2.3), sa ciljem da se maksimizuje ukupan broj kreiranih paralelnih testova, pri čemu se dozvoljava da testovi dele stavke. U grafu kojim se problem modeluje, čvorovi predstavljaju testove, a po jedna grana postoji između svaka dva testa koji imaju zajedničke stavke.

Problem se formuliše na sledeći način:

$$\text{Promenljive:} \quad t_f \quad f = 1,2,\dots,T \tag{144}$$

$$\text{Funkcija cilja:} \quad \max|C| \tag{145}$$

$$\text{Ograničenja:} \quad C \subseteq V \quad \forall t_{f1}, t_{f2} \in C \quad (t_{f1}, t_{f2}) \in E \quad f_1 \neq f_2 \quad f_1, f_2 = 1,\dots,T \tag{146}$$

gde je V skup svih dopustivih testova (testova koji zadovoljavaju sva ograničenja osim ograničenja da ne dele stavke), E je skup parova (t_{f1}, t_{f2}) dopustivih testova koji imaju zajedničke stavke, T je broj dopustivih testova odabranih iz skupa V, i t_f je binarna promenljiva čija je vrednost jedan ukoliko je test sa indeksom f iz skupa V uključen u kliku C, ili nula u suprotnom.

Problem se rešava egzaktnim algoritmom koji je zasnovan na algoritmu opisanom u radu (Nakanishi & Tomita, 2008). Egzaktni algoritam nije primenljiv za slučaj velikih banki stavki, pa je predloženo korišćenje heuristika.

3.1.4 Sumarni pregled formulacija problema

Tabela 3.2 daje pregled formulacija koje su opisane u odeljku 3.1. Za svaku od kategorija formulacije problema biće naveden tip matematičkog modela, vrsta sastavljanja testa (redno ili simultano), kratak opis cilja optimizacije i izvor iz otvorene literature. Za naziv formulacije koristiće se sledeće skraćenice: 1D BPP – jednodimenzionalni problem pakovanja u korpe; MOPTSC – sastavljanje paralelnih ispitnih testova sa više funkcija cilja; PTSC – sastavljanje paralelnih ispitnih testova; STSC – sastavljanje serije ispitnih testova; NFP – problem mrežnog protoka; KP – problem pakovanja u ranac; MI – minimax model; WADM – model ponderisanog apsolutnog odstupanja; BST – velika test-senka; WDM – model ponderisanog odstupanja; JSSP – job shop scheduling problem; MA – maximin model; MCP – problem maksimalne klike; ML – mašinsko učenje; KNN – *k*-najbližih suseda i MSPP – problem maksimalnog pakovanja skupa. Za tip matematičkog modela koristiće se sledeće skraćenice: 0-1 LP – binarno linearno programiranje, ILP – celobrojno linearno programiranje i MZOLP – mešovito 0-1 linearno programiranje. Za vrstu sastavljanja testa koristiće se sledeće skraćenice: SST – simultano sastavljanje testova i RST – redno sastavljanje testova

Tabela 3.2 Pregled formulacija problema sa ciljevima klasične teorije testova

	Formulacija	Matematički model	Vrsta sastavljanja	Cilj	Izvor	Odeljak
Ciljevi klasične teorije testa	1D BPP	MZOLP	SST	Balansiranje testova po diskriminativnosti i težini stavki	(Pereira and Vila 2015)	3.1.1.1.2
	1D BPP	MZOLP	SST	Balansiranje testova po diskriminativnosti i težini stavki	(Brusco et al. 2013)	3.1.1.1.2
	MPOPTSC	0-1 LP	SST	Balansiranje testova po diskriminativnosti i težini stavki	(Ho et al. 2009)	3.1.1.1.2
	PTSC	0-1 LP	SST	Balansiranje testova po diskriminativnosti i težini stavki	(Hwang et al. 2008)	3.1.1.1.2
	STSC	0-1 LP	SST	Balansiranje težine testova	(Yin, et al. 2006)	3.1.1.1.1
	NFP	0-1 LP	SST	Maksimizacija pouzdanosti testa	(Armstrong et al. 1994)	3.1.1.1.3
Ciljevi teorije odgovora na stavku	KP	0-1 LP	SST	Maksimizacija prosečne diskriminativnosti testa	(Nguyen et al. 2013)	3.1.1.2.5
	MI	ILP	SST	Minimizacija razlike između TIF i TTIF	(Chang and Shiu 2012)	3.1.1.2.2
	WADM		SST	Minimizacija razlike između TIF i TTIF	(Songmuang and Ueno 2011)	3.1.1.2.2
	MI	0-1 LP	SST	Minimizacija razlike između TIF i TTIF	(Sun, et al. 2008)	3.1.1.2.2
	BST	MZOLP	RST	Minimizacija razlike između TIF i TTIF	(van der Linden and Adema 1998)	3.1.1.2.2
	WADM	0-1 LP	SST	Minimizacija razlike između TIF i TTIF	(Luecht 1998)	3.1.1.2.2
	WDM	MILP	RST	Minimizacija devijacije od ograničenja	(Stocking et al. 1993)	3.1.1.2.4
	JSSP	MZOLP	SST	Minimizacija razlike između TIF i TTIF	(Adema 1992)	3.1.1.2.2
	MA	MZOLP	SST	Maksimizacija TIFa dok se održava relativni oblik ciljnog TTIFa	(Adema et al. 1991)	3.1.1.2.3
	MA	MZOLP	RST, SST	Maksimizacija TIFa dok se održava relativni oblik ciljnog TTIFa	(Boekkooi-Timminga 1990)	3.1.1.2.3
	MI	0-1 LP	SST	Minimizacija razlike između TIF i TTIF	(Boekkooi-Timminga 1987)	3.1.1.2.2
	KP	0-1 LP	SST	Minimizacija broja stavki u testu	(Theunissen 1986)	3.1.1.2.1
Uparivanje sa srodnim testovima	ML		RST	Uparivanje sa skupom test-klica	(Ignjatovic et al. 2015)	3.1.2.2
	MRS		SST	Uparivanje sa zadatom test-klicom	(Chen 2015)	3.2.1.3
	MRS		SST	Uparivanje sa zadatom test-klicom	(Chen et al. 2012)	3.2.1.3
	NFP	0-1 LP	SST	Uparivanje sa zadatom test-klicom	(Armstrong et al. 1992)	3.1.2.1
	MRS	0-1 LP	SST	Uparivanje sa zadatom test-klicom	(van der Linden and Boekkooi-Timminga 1988)	3.1.2.1
Maksimizacija broja testova	MCP	0-1 LP	SST	Maksimizacija broja paralelnih testova	(Ishii et al. 2014)	3.1.3
	MSPP	0-1 LP	SST	Maksimizacija broja paralelnih testova	(Belov and Armstrong 2006)	3.1.3

Većina ciljeva zasnovanih na CTT se odnosi na sastavljanje paralelnih testova sa približno istim težinama i/ili diskriminativnostima stavki. Prednost korišćenja ciljeva zasnovanih na CTT je ujedno i prednost CTT teorije, a to je konceptualna jednostavnost metrike. Prema rezultatima istraživanja (Fan, 1998) i (Idowu, Eluwa, & Abang, 2011), kvalitet testova (procena sposobnosti ispitanika) dobijenih primenom CTT teorije približno je jednak kvalitetu testova dobijenih primenom IRT teorije.

Kada se razmatraju ciljevi zasnovani na IRT, minimizacija broja stavki u testovima se retko koristi kao cilj, jer kreatori testova vrlo često moraju da utiču na broj stavki u testu (Theunissen, 1985) i (Theunissen, 1986). Sa druge strane, da bi mogli da zadaju TTIF za različite vrednosti sposobnosti, i zatim minimizuju rastojanja od njega, kreatori testova moraju da budu upoznati sa metrikom za izračunavanje TIF (Theunissen, 1986), (Ackerman, 1989), (Adema, 1992), (Luecht, 1998), (van der Linden & Adema, 1998) i (Chen, Chang, & Wu, 2012). U tom slučaju, bolja opcija je korišćenje relativnog oblika TTIF (Boekkooi-Timminga, 1990) i (Adema, Boekkooi-Timminga, & van der Linden, 1991). Kada postoji problem neizvodljivosti u sastavljanju testa (sva ograničenja ne mogu uvek da budu ispunjena), ograničenja se mogu smatrati „poželjnim osobinama“ i kreirati funkcija cilja koja minimizuje rastojanja od ograničenja (Swanson & Stocking, 1993) i (Stocking, Swanson, & Pearlman, 1993).

Uparivanje testa sa srodnim testovima je koristan cilj kada kreatori testa već imaju test-klicu ili žele da je naprave da služi kao model za kreiranje drugih testova. Nedostatak ovog pristupa je što ne omogućava dalji uticaj na neke atribute testa kao što su broj zadataka u testu ili u kategoriji, težini testa (Luecht & Sireci, 2011). Navedeni cilj može da bude još korisniji kada kreatori testa već imaju referentni skup test-klica (Ignjatović, Bojić, Furlan, & Tartalja, 2015). Tada se navedeni nedostaci mogu umanjiti odgovarajućim izborom testova koji će ući u referentni skup.

Maksimizacija broja testova može da se koristi kada je cilj da se dobro iskoriste postojeće stavke (kojih je možda malo) da bi mogao da se kreira što veći broj testova. Takođe, rešenje ovog problema može da se iskoristi za grubu procenu kvaliteta banke stavki i daje mogućnost za analizu stavki koje nisu izabrane za testove (Belov & Armstrong, 2005) i (Belov & Armstrong, 2006).

3.2 Pregled i klasifikacija heurističkih rešenja problema ASPT

U ovom odeljku se daje pregled postojećih rešenja problema ASPT, klasifikovanih u dve kategorije: heuristička rešenja i metaheuristička rešenja. Formulacije problema ASPT čija rešenja se ovde prikazuju su opisane u odeljku 3.1.

Slika 3.2 prikazuje osnovnu klasifikaciju metoda rešavanja ASPT problema na heuristička rešenja i metaheuristička rešenja. Pod heurističkim rešenjima će se smatrati pristupi koji daju približna rešenja i koja su zavisna od konkretnog problema, dok metaheuristička rešenja predstavljaju opštije približne algoritme primenljive na veliki broj problema kombinatorne optimizacije (Talbi, 2009).

Već je ranije uvedena podela heuristika na konstruktivne heuristike i heuristike poboljšanja (2.2.1.4). Ova podela može da se primeni i na metaheuristike. Kada se ova podela primeni na rešavanje problema ASPT, kontruktivne heuristike kreiraju test tako što se dodaju stvake u test, jedna po jedna. Heuristike poboljšanja generišu kompletne testove i traže najbolje rešenje variranjem stavki u njima.

Metaheuristike se klasifikuju i na osnovu broja rešenja koja se poboljšavaju u jednom trenutku i to na metaheuristike koje su zasnovane na jednom rešenju (eng. *Single-Solution Based Metaheuristics*, SSM) i metaheuristike koje su zasnovane na populaciji rešenja (eng. *Population Based Metaheuristics*, PBM). Metaheuristike koje su zasnovane na jednom rešenju (na primer,

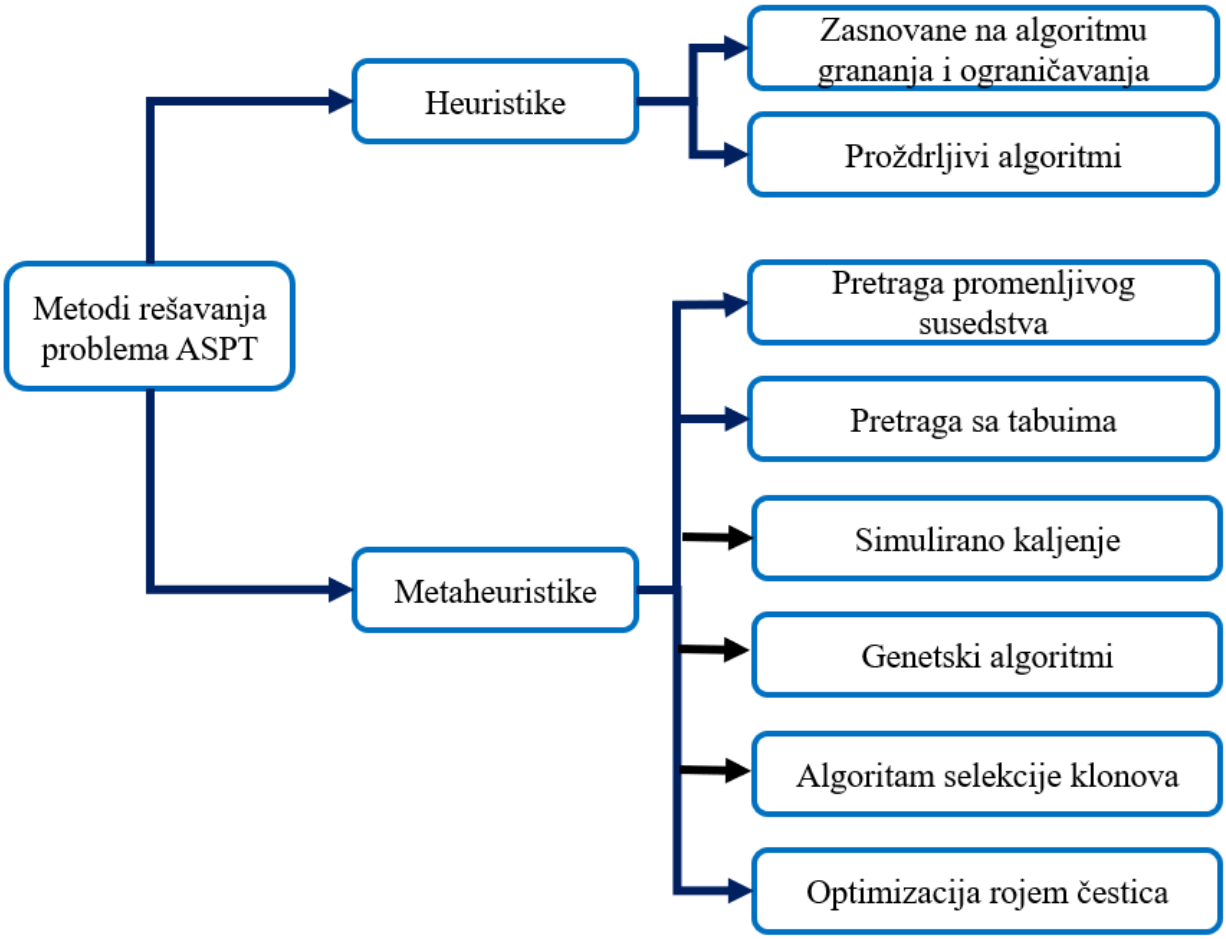
simulirano kaljenje, pretraga sa tabuima, pretraga promenljivih okolina) se kreću kroz prostor rešenja od trenutnog rešenja do sledećeg rešenja. Metaheuristike koje polaze od populacije rešenja (na primer, genetski algoritam, pčelinji algoritam, optimizacija rojem čestica, algoritam selekcije klonova) u svakom koraku generišu novu, poboljšanu populaciju rešenja.

Za različite metode rešavanja koje se u ovom odeljku prikazuju, izdvajaju se sledeće važne karakteristike:

a. *Inicijalna konfiguracija*. Način na koji se generišu inicijalna rešenja kao i inicijalne vrednosti određenih parametara heuristike (ukoliko postoje).

b. *Funkcija pomeraja*. Za (meta)heuristike poboljšanja termin pomeraja se odnosi na modifikaciju koja se primenjuje na postojeće rešenje da bi se dobilo novo kandidat-rešenje u blizini postojećeg, u prostoru rešenja, ili na metod kojim se dobija novo kandidat-rešenje udaljeno od postojećeg u prostoru rešenja. Za slučaj konstruktivnih heuristika, ovaj termin se odnosi na način odabira objekta koji se dodaje u nepotpuno rešenje.

c. *Uslov prekida*. Uslov kojim se prekida pretraga rešenja u (meta)heuristikama poboljšanja. Uslov prekida može da bude vremensko ograničenje, unapred definisan broj iteracija posle kojih se ne vidi poboljšanje u postojećem rešenju, ili kada se teorijski optimum pronađe.



Slika 3.2 - Klasifikacija pristupa za rešavanje problema ASPT

3.2.1 Heuristike

U ovom odeljku će biti prikazana postojeća rešenja, za rešavanje ASPT problema, koja su klasifikovana kao heuristička rešenja. Tu se mogu izdvojiti tri podgrupe metoda: heuristike zasnovane na algoritmu grananja i ograničavanja, pohlepni algoritmi i slučajno uzorkovanje.

3.2.1.1 Heuristike zasnovane na algoritmu grananja i ograničavanja

Algoritam grananja i ograničavanja (B&B), za optimalno rešavanje problema celobrojnog programiranja, predložen je u radu (Land & Doig, 1960). Algoritam obavlja pretragu stabla, ali efikasno tako što se odbacuju delovi prostora pretrage gde sigurno ne postoji rešenje. Algoritam koristi vraćanje unazad (eng. *backtracking*) sve dok se ceo prostor pretrage ne obiđe.

U ovom algoritmu postoje dva glavna principa:

1. Ukoliko se doda ograničenje u formulaciju problema maksimizacije, vrednost funkcije cilja ne može da se poveća, ali može da se smanji.
2. Ukoliko je trenutno kandidat-rešenje nedopustivo, dodavanjem ograničenja ne može da postane dopustivo.

U nastavku će biti opisan postupak za rešavanje 0-1 LP problema pomoću B&B algoritma. Pretraga počinje rešavanjem problema koje je dobijeno uklanjanjem ograničenja da promenljive mogu imati samo vrednosti nula i jedan (relaksirani problem, koji je ustvari LP problem), koristeći, na primer, simpleks algoritam (Dantzig, 1951). To rešenje predstavlja koren stabla. Ukoliko je dobijeno rešenje takvo da sve promenljive imaju dodeljene binarne vrednosti (0 ili 1), tada je to rešenje optimalno rešenje problema. Najčešće to neće biti slučaj, pa će neke od promenljivih imati dodeljene realne vrednosti u opsegu [0, 1]. Tada se postavlja da je maksimalna vrednost funkcije cilja $Z_{max} = -\infty$, i da rešenje nije pronađeno.

Iterativno se dodaju ograničenja na sledeći način. Izabranoj promenljivoj (x_i) iz LP problema u trenutnom čvoru stabla se najpre doda ograničenje $x_i = 0$, čime se dobija prvi novi LP potproblem i novi čvor-potomak u stablu, a zatim ograničenje da je vrednost $x_i = 1$, čime se dobija drugi novi LP potproblem i novi čvor-potomak u stablu.

Rešava se prvi LP problem (sa ograničenjem $x_i = 0$). Dobija se vrednost funkcije cilja Z_1. Rešava se drugi LP problem (sa ograničenjem $x_i = 1$) i dobija se vrednost funkcije cilja Z_2. Ukoliko nijedno rešenje nema dodeljene binarne vrednosti za sve promenljive, onda se porede vrednosti njihovih funkcija cilja. Ukoliko je $Z_1 > Z_2$ onda se pretraga nastavlja od čvora sa ograničenjem $x_i = 0$, a u suprotnom se pretraga nastavlja od čvora sa ograničenjem $x_i = 1$. U prvom slučaju više se ne pretražuje deo stabla koji počinje od čvora sa ograničenjem $x_i = 1$, a u drugom slučaju se ne pretražuje deo stabla od čvora sa ograničenjem $x_i = 0$, zato što se dodavanjem novih ograničenja vrednost funkcije cilja neće povećati.

Ukoliko su jedno ili oba rešenja LP problema takva da sve promenljive imaju binarne vrednosti (kandidat-rešenje), onda se proveravaju druga ograničenja u postavci problema. Ukoliko su ona zadovoljena, dobija se kandidat-rešenje koje je dopustivo. Vrednost funkcije cilja dopustivog rešenja (Z) se poredi sa Z_{max}. Ukoliko je Z veće od Z_{max}, tada je to rešenje novo trenutno rešenje i $Z_{max} = Z$. Dalje se pretraga vraća unazad na neobiđene čvorove. Od čvora koji ima nedopustivo rešenje se dalje ne nastavlja pretraga. Rešenje problema je poslednje zapamćeno trenutno rešenje, kada se obiđu svi čvorovi.

Pošto se sa brojem promenljivih stablo pretrage eksponencijalno povećava, ovaj metod nije efikasan kada je broj promenljivih veliki i zato se koriste heuristička rešenja.

U radu (Adema, 1988) je predložena heuristika zasnovana na B&B algoritmu za slučajeve kada je poznato da je (1) vrednost funkcija cilja optimalnog rešenja za 0-1 LP problem maksimizacije

blizu vrednosti funkcije cilja rešenja relaksiranog problema, odnosno rešenja LP problema (Z_{LP}) i (2) Z_{LP} nije blisko 0. Urađene su dve modifikacije B&B algoritma: (1) način inicijalizacije donje granice funkcije cilja i (2) uvedeno je novo pravilo za ograničavanje promenljivih.

Inicijalna konfiguracija. Nakon rešavanja inicijalnog LP problema, donja granica funkcije cilja se postavlja na vrednost $Z_{max} = k_1 Z_{LP}$ ($0 << k_1 < 1$), gde je k_1 unapred definisana konstanta bliska jedinci, a Z_{LP} vrednost funkcije cilja rešenja relaksiranog (LP) problema, koja je ujedno i gornja granica funkcije cilja 0-1 LP problema.

Funkcija pomeraja. Najpre se računa rešenje problema LP dobijenog uvođenjem novog ograničenja nad izabranom promenljivom x_j ($x_j = 0$ ili $x_j = 1$). Kada se to rešenje dobije, vrednosti promenljivih čije vrednosti nisu 0 ili 1 u rešenju LP problema (nebazične promenljive) se dodatno odrede tako da budu 0 ili 1. Za svaku nebazičnu promenljivu x_i se računa vrednost *smanjenja cene*. Vrednost smanjenja cene za x_i je vrednost v_i koja predstavlja iznos smanjenja vrednosti funkcije cilja (za problem maksimizacije) po jedinici povećanja vrednosti promenljive x_i (promena vrednosti promenljive sa 0 na 1), kada su ostale vrednosti promenljivih u rešenju nepromenjene. Vrednost smanjenja cene v_i određuje vrednost promenljive x_i. Pravila su sledeća: $x_i = 0$ ako je $Z_{LP} - k_2 Z_{LP} < v_i$ i $x_i = 1$ ako je $Z_{LP} - k_2 Z_{LP} < -v_i$, gde je k_2 pomoćna vrednost ($k_2 \geq k_1$). Na ovaj način se stablo pretrage smanjuje.

Uslov prekida. Algoritam prestaje sa radom kada se pronađe prvo dopustivo rešenje. Smatra se da je to rešenje dovoljno dobrog kvaliteta zato što se vrednost funkcije cilja rešenja nalazi u opsegu $[k_1 Z_{LP}, Z_{LP}]$.

Ova heuristika se koristila za rešavanje problema ASPT formulisanog u (Boekkooi-Timminga, 1990) (odeljak 3.1.1.2.2). Banka stavki je imala 300 stavki i kreirana su dva paralelna testa.

Mana ove heuristike je da mora da se proceni koliko je rešenje blizu optimalnog rešenja i u skladu sa time definiše koeficijent k_1 koji ako nije dobro definisan (vrednost funkcije cilja optimalnog rešenja je ispod vrednosti $k_1 Z_{LP}$) postupak može da se završi bez pronađenog rešenja. Takođe je veličina banke stavki sa kojom je rađeno umerene veličine, a sa povećanjem broja stavki bi se značajno povećalo vreme izvršavanja.

U radu (van der Linden & Boekkooi-Timminga, 1989) je predložena heuristika za rešavanje AST problema zasnovana na B&B algoritmu, gde se izvršavanje prekidalo kada se došlo do prvog dopustivog 0-1 LP rešenja. U radu (Adema, 1992), za formulaciju problema datu u odeljku 3.1.1.2.2, ova heuristika se koristila za sekvencijalno sastavljanje šest paralelnih testova iz banke stavki sa 600 stavki.

U radu (van der Linden & Adema, 1998) je predstavljena formulacija problema opisana u odeljku 3.1.1.2.2. Problem je rešavan komercijalnim softverom pod nazivom ConTEST čiji je algoritam zasnovan na B&B algoritmu čiji detalji nisu navedeni (Timminga, van der Linden, & Schweizer, 1996). Banka stavki koja se koristila je imala 753 stavki i sastavljala su se tri paralelna testa. Dobijeni testovi su bili visokog kvaliteta, ali je naglašeno da je to i posledica dobrog kvaliteta banke stavki.

U radu (Belov & Armstrong, 2006) se rešavao problem formulisan u odeljku 3.1.32.2.2.1.3. Dopustivi testovi su inicijalno sastavljeni i na njih se primenio B&B algoritam predložen u radu za pronalaženje maksimalnog broja testova. Navedeni rezultati pokazuju da se dobija dvostruko veći broj testova u odnosu na slučaj da su se testovi kreirali jedan po jedan.

Kao zaključak, heuristike zasnovane na B&B algoritmu su se pokazale korisne za banke stavki u opsegu (100-1000) stavki i nekoliko paralelnih testova (manje od 10). Problem koji se javlja jeste kada veličina banke stavki raste i kada se broj paralelnih testova povećava (Brusco, Köhn, & Steinley, 2013).

3.2.1.2 Heuristike zasnovane na pohlepnim algoritmima

Grupa algoritama koja traži lokalne optimume brzo napredujući ka njima, uz neizvesnost pronalaženja globalnog optimuma, naziva se pohlepnim algoritmima (Nemhauser, Wolsey, & Fisher, 1978). Pohlepni algoritam, u principu, traži sledeći element rešenja koji u tom koraku daje najbolje rešenje. Pošto u algoritmu nema vraćanja unazad, globalno rešenje može da se promaši i da se pretraga zaustavi na lokalnom optimumu. Ovaj problem se obično rešava tako što se uključuje diverzifikacija posle pronađenog lokalnog optimuma. Pohlepni algoritmi su brzi, ali rešenja su često samo lokalno optimalna.

U radu (Ackerman, 1989) je predložena pohlepna heuristika za simultano sastavljanje paralelnih testova sa ciljem da se upari TIF testa sa zadatim TTIF na nekoliko nivoa sposobnosti. Nakon što se testovi sastave, testovi sa najvećom razlikom TIF vrednosti od TTIF se međusobno ujednačuju tako što razmenjuju stavke.

Inicijalna konfiguracija. Prva stavka za svaki test se bira nasumično iz banke stavki. Ostale stavke iz banke stavki se zatim uređuju po opadajućim vrednostima njihove IIF, na svakom nivou sposobnosti, i smeštaju u liste (broj lista stavki odgovara broju nivoa sposobnosti, pri čemu su stavke deljene između pojedinih lista). TIF za svaki test se zatim izračuna, kao i razlika između izračunatog TIF i zadatog TTIF. Posle toga se testovi uređuju po opadajućim vrednostima maksimalne razlike između TIF testa i TTIF i smeštaju u jednu listu.

Funkcija pomeraja. Stavka sa najvećom vrednošću IIF, na nivou sposobnosti na kome prvi test iz liste testova ima najveću razliku TIF od zadatog TTIF, se bira za prvi test; sledeća stavka se bira za sledeći test tako da ima najveću vrednost IIF na nivou sposobnosti na kome taj test ima najveću razliku TIF i zadatog TTIF i postupak se ponavlja sve dok svi testovi ne dobiju po jednu novu stavku. TIF vrednosti za svaki test, kojem je dodata jedna nova stavka, se zatim ponovo izračunavaju, testovi se ponovo uređuju i postupak se ponavlja.

Uslov prekida. Kada se dostigne dužina testa (ukupan broj stavki), postupak se prekida, a zatim se testovi međusobno ujednačuju posebnom procedurom.

Banka stavki koja se koristila ima 600 stavki i kreirano je 6 testova.

Ova heuristika pokušava da ublaži problem nejednakosti kod testova, koristeći proceduru za ujednačenje. Problem koji se ovde javlja jeste da kada bi se više testova kreiralo (više od 10), nepoznato je kako bi to uticalo na vreme izvršavanja procedure za ujednačenje, zato što to nije razmatrano u radu. Druga mana pristupa je i u tome, što zbog prirode pohlepnog algoritma, stavke koje imaju najveću vrednost IIF se biraju, pa je moguće da se dostigne TTIF pre nego što je predviđen broj stavki odabran.

U radovima (Armstrong, Douglas, & Wang, 1994) i (Armstrong, Jones, & Wu, 1992) se koristila heuristika slična prethodno opisanoj Akermanovoj heuristici, za rešavanje problema koji je opisan u odeljku 3.1.2.1, a koji se deli na dva potproblema. Prvi potproblem je kreiranje podskupova stavki koje su uparene sa stavkama-klicama, tako što postoji po jedan podskup za svaku stavku-klicu. Drugi potproblem je dodela stavki, iz podskupova stavki koji su kreirani kao rezultat rešenja prvog problema, testovima. Heuristika se koristi za rešavanje drugog problema. Stavke se u podskupovima uređuju po opadajućim vrednostima udaljenosti od stavke-klice (136) (137). Takođe se testovi, u svakom koraku, uređuju po opadajućoj vrednosti udaljenosti od test-klice. Pohlepni algoritam bira stavku sa najmanjom udaljenošću i dodeljuje je testu sa trenutno najvećom udaljenošću od test-klice. Postupak se nastavlja na isti način sve dok se sve stavke ne dodele testovima.

U radu (Adema, 1992) je predložena heuristika za rešavanje problema opisanog u odeljku 3.1.1.2.2. I ovde se problem deli na dva potproblema. Prvi potproblem se rešava tako što se kreira test koji sadrži n·T stavki, koristeći heuristiku predloženu u radovima (van der Linden & Boekkooi-

Timminga, 1989) i (Adema, 1988). Drugi potproblem u kome se odabrane stavke dodeljuju testovima se rešava heuristikom opisanom u radu (Ackerman, 1989).

U radu (Swanson & Stocking, 1993) je predložena kombinovana konstruktivna heuristika sa heuristikom poboljšanja za rešavanje problema AST (u pitanju je sastavljanje samo jednog testa), za rešavanje WDM problema opisanog u odeljku 3.1.1.2.4. Heuristika ima dve faze: prva je početni odabir stavki za kompletan test, a druga faza je zamena odabranih stavki, sve dok je moguće poboljšanje. U prvoj fazi se testu dodaje jedna po jedna „najbolja“ stavka. Najbolja stavka za test je ona za koju je suma ponderisanih odstupanja vrednosti atributa od graničnih vrednosti minimalna (121). Na taj način se sastavlja test sve dok ne dostigne predviđenu dužinu. Kada se dostigne predviđena dužina, prelazi se na drugu fazu rešavanja, gde se odabrane stavke testa zamenjuju stavkama iz banke stavki sve dok ima poboljšanja (smanjenja) u sumi ponderisanih odstupanja. Ova heuristika može da se koristi za simultano sastavljanje paralelnih testova sa ciljem da se smanji ukupna suma odstupanja od graničnih vrednosti atributa za sve testove (Stocking, Swanson, & Pearlman, 1993).

U radu (Luecht, 1998) je predložena varijanta pohlepnog algoritma koji bira stavku po stavku tako što pronalazi trenutno najbolju stavku na osnovu modifikovane funkcije cilja. Za svaku stavku se funkcija cilja ažurira na osnovi atributa stavki koje su prethodno odabrane za test. Formulacija problema rešavanog u navedenom radu je objašnjena u odeljku 3.1.1.2.2.

U radu (Bojić, Bošnjaković, Protić, & Tartalja, 2016) su predložili metod zasnovan na modifikovanoj verziji planinarenja (eng. *hill-climbing*), gde se jedna modifikacija odnosi na razmatranje, u svakom koraku, skupa stanja sa skoro optimalnim vrednostima funkcije cilja, umesto samo jednog sa najboljom vrednošću. Druga modifikacija uključuje vremenski ograničeno vraćanje unazad sa nasumičnim ponovnim pokretanjem da bi se izbegli lokalni optimumi. Funkcija cilja se menja za vreme pretrage, na osnovu dubine stabla pretraživanja.

3.2.1.3 Heuristike zasnovane na slučajnom uzorkovanju

Ovaj pristup koristi pretraživanje prostora rešenja koristeći slučajno uzorkovanje. Slučajno uzorkovanje podrazumeva da se rešenja iz prostora rešenja biraju nasumično, a zatim se procenjuje njihova vrednost ciljne funkcije. Postupak se ponavlja sve dok se ne pronađe rešenje koje zadovoljava predviđen kvalitet rešenja. U slučaju sastavljanja testova, slučajno uzorkovanje omogućava da svaka dostupna kombinacija stavki (dostupni test) ima jednaku verovatnoću da bude odabrana (Belov, 2008).

U radu (Chen, Chang, & Wu, 2012) se koristi pristup predložen u (Gulliksen, 1950), da se testovi kreiraju tako što im se dodeljuju stavke iz banke stavki koje su uparene sa stavkama iz testa-klice (odeljak 3.1.2.1). U radu su predložene dve heuristike za sastavljanje testova uparivanjem stavke sa stavkom-klicom (odeljak 3.1.2.1). U prvoj heuristici se stavke iz banke stavki, kao i stavke-klice raspoređuju u ćelije na osnovu vrednosti atributa stavke. Dvodimenzionalni prostor je podeljen na $M{\cdot}N$ ćelija, gde prva dimenzija ($m = 1,...,M$) odgovara IRT težini (a), a druga dimenzija ($n = 1,...,N$) odgovara IRT diskriminativnosti (b). Stavke za testove se nasumično biraju iz ćelija koje odgovaraju atributima stavke-klice date ćelije tako da je distribucija stavki testa koji se sastavlja jednaka distribuciji stavki-klica. Druga heuristika se zasniva na podeli prostora u trodimenzionalne ćelije, na osnovu vrednosti IIF stavke za tri nivoa sposobnosti ($k = 1,2,3$). Iz praktičnih razloga, stavke su prvo podeljenje u podskupove na osnovu pripadnosti kategoriji i onda se stavke iz podskupa dele u ćelije. To znači da se kreiraju podtestovi za svaku kategoriju. Autori su definisali da je broj stavki koji treba da pripadne određenoj kategoriji nepromenljiv.

U radu (Chen, 2015) se ovaj pristup poboljšao na način da broj stavki koje treba da pripadaju određenoj kategoriji ne bude nepromenljiv, već da se nalazi u određenim granicama. Autori smatraju da oba pristupa daju rezultate slične rezultatima dobijenim pristupom sastavljanja testova

koristeći kombinatornu optimizaciju. Kriterujum za odabir jeste sličnost TIFa i TCFa kao i srednje kvadratne devijacije TIFa i TCFa između testa koji se sastavlja i test-klice.

3.2.2 Metaheuristike

3.2.2.1 Simulirano kaljenje

Simulirano kaljenje (SA) je metaheuristika poboljšanja, koja je inspirisana procesom kaljenja u metalurgiji (Khachaturyan, Semenovskaya, & Vainstein, 1979). Kaljenje u metalurgiji uključuje zagrevanje i kontrolisano postepeno hlađenje materijala da bi se kreirala čvrsta kristalna supstanca bez defekata. Cilj je da se pronađe uređenje atoma materijala takvo da se dobije minimalna ukupna energija materijala. Ukupna energija materijala određuje kvalitet rešenja. U procesu je često potrebno da se materijal ponovo zagreje, čime se privremeno smanji kvalitet rešenja, da bi se (novim hlađenjam) ispravile nepravilnosti. Raspored hlađenja služi da se smanji temperatura materijala i treba unapred da se definiše pomoću parametara *templength* i *cool*. Parametar *templength* predstavlja broj rešenja (uređenja atoma) koji se generiše pre nego što se temperatura promeni, dok parametar *cool* definiše iznos za koji se temperatura smanjuje. Ukoliko se energija materijala promenom temperature smanji, novo rešenje se prihvata, a ukoliko se energija poveća, rešenje se prihvata samo sa određenom verovatnoćom. Metaheuristika je opisana u radu (Kirkpatrick, Gelatt, & Vecchi, 1983).

Analogija sa problemom ASPT je sledeća: stavke predstavljaju atome, test predstavlja materijal koji se sastoji iz atoma, a funkcija cilja predstavlja ukupnu energiju materijala (konfiguracije atoma).

U radu (Brusco, Köhn, & Steinley, 2013) je predložena heuristika zasnovana na SA za rešavanje ASPT problema formulisanog kao 1D BPP (odeljak 3.1.1.1.2).

Inicijalna konfiguracija. Stavke su uparene i smeštene u S skupova, na osnovu sličnih težinskih koeficijenata stavke, w_i (57). Početno rešenje se sastavlja tako što se dodeljuje po jedna stavka nasumično izabrana iz svakog od S skupova jednom od T testova. Izračunava se početna vrednost funkcije cilja Z koja je jednaka težini najtežeg testa, gde je težina testa jednaka zbiru težinskih koeficijenata svih stavki testa. Početna vrednost parametra *temp* se računa kao maksimalna razlika težinskih koeficijenata između svih mogućih razmena stavki (videti niže), $temp = max\ |w'' - w'|$. Inicijalna vrednost za koeficijent hlađenja se unapred definiše ($0 < cool < 1$), kao i za vrednost trajanja temperature *templength*.

Funkcija pomeraja. Nasumično se izabere jedan skup s. Bira se stavka x' iz skupa s koja pripada testu sa najvećom težinom. Druga stavka x'' se nasumično bira iz istog skupa s ($x'' \neq x'$). Dve izabrane stavke (stavke x' i x'' sa težinskim koeficijentima w' i w'') se zamene u testovima kojima pripadaju. Razlika energije za ovu promenu, $\Delta(x'', x')$, se izračunava na sledeći način.

$$\Delta(x'', x') = \max\left\{w'' - w' + \sum_{i=1}^{I} w_i x_i'',\ \ w' - w'' + \sum_{i=1}^{I} w_i x_i''\right\} - Z \tag{147}$$

Ukoliko je $\Delta(x'', x') \leq 0$ prihvata se zamena stavki. Ukoliko je $\Delta(x'', x') > 0$, zamena pogoršava vrednost funkcije cilja, pa se ona dodatno analizira. Generiše se pseudoslučajan broj *rnd* u intervalu [0,1]. Ukoliko je $rnd < e^{-\Delta(x'',x')/temp}$ onda se zamena, odnosno pogoršanje, prihvata. Izračunava se vrednost funkcije cilja Z* za novo rešenje. Ukoliko je $\Delta(x'', x') \leq 0$ i $Z < Z^*$, novo rešenje postaje tekuće rešenje i $Z = Z^*$. Ukoliko je $\Delta(x'', x') > 0$ i $rnd < e^{-\Delta(x'',x')/temp}$ lošije rešenje postaje tekuće rešenje i $Z = Z^*$. Zatim se bira sledeći skup iz kojeg se biraju stavke za razmenu, a *templength* se dekrementira. Ukoliko je *templength* = 0, *temp* se smanjuje za iznos parametra *cool* i procedura se ponavlja.

Uslov prekida. Kraj kompletnog ciklusa koji nastupa kad *temp* dostigne vrednost nula. Rešenje algoritma je poslednje zapamćeno trenutno rešenje.

Problem koji je rešavan je imao do 6000 stavki i do 300 testova. Autori su prikazali da SA daje rešenja boljeg kvaliteta nego optimizacioni softverski paket CPLEX 12.1 (IBM, 2009), za slučajeve kada je broj testova veći ili jednak tri, u slučaju da su oba imali isto vremensko ograničenje. Kada je banka srednje (I = 300 ili I = 600) ili velike veličine (I = 3000 ili I = 6000), SA daje rešenja blizu optimalnog rešenja.

3.2.2.2 Pretraga promenljivih okolina

Pretraga promenljivih okolina (VNS) je metaheuristika koju su predložili u radu (Mladenović & Hansen, 1997) i predstavlja jednu generalizaciju heuristika pretraživanja. U njoj se inicijalno definiše određeni broj okolina za pretragu. Ideja je da se sistematski pretražuje okolina dok se ne dođe do lokalnog optimuma, a da se onda promeni okolina sa ciljem da se nađe globalni optimum. U VNS metaheuristici se definišu okoline, funkcija pomeraja i operator diverzifikacije (skoka u drugu okolinu) kao parametri heuristike.

Dopustiva rešenja u ASPT predstavljaju dopustivi testovi. Okoline su definisane kao konretne heuristike.

U radu (Pereira & Vila, 2015) je predložena heuristika za rešavanje problema formulisanog kao 1D BPP (odeljak 3.1.1.1.2), zasnovana na VNS metaheuristici. Definišu se tri okoline i uvodi procedura za pronalaženje optimalnog rešenja za slučaj dva skupa stavki (S = 2) (nadalje PS2).

Kada je broj skupova stavki S = 2, što implicira da testovi imaju po dve stavke, bez obzira na broj testova koji je određen brojem stavki po skupu, optimalno rešenje se dobija na sledeći način: stavke u jednom skupu se uređuju u nerastućem redosledu težina, dok se stavke u drugom skupu uređuju u neopadajućem redosledu težina. Optimalno rešenje se dobija tako što se i-ti test sastoji od stavki na poziciji i u svakom od ova dva uređena skupa. Kada bi se obavila kompletna pretraga, odnosno ispitivanje svih mogućih permutacija parova stavki iz dva skupa, ona bi imala eksponencijalnu složenost, zbog pretrage $n!$ slučajeva (svaki element iz prvog skupa može da bude uparen sa elementom iz drugog skupa). Predložena procedura ima složenost $O(T \cdot \log T)$ što odgovara složenosti algoritma za uređivanje.

Predložene su tri okoline koje se pretražuju, definisane na sledeći način. Okolina 1 se kreira uklanjanjem stavki koje potiču iz izabranog skupa s_j (j = 1,...,S) iz svih testova kojima su dodeljene. Ponovo se izračunavaju težine testova iz kojih su stavke uklonjene i stavke iz skupa s_j se ponovo dodeljuju testovima u skladu sa procedurom PS2 (prvi skup je skup težina stavki, dok je drugi skup skup težina testova bez stavki iz skupa s_j). Okolina 2 se kreira deljenjem svakog testa na dve particije, tako da jedna particija sadrži stavke iz s ($1 \leq s < S$) skupova, a druga particija sadrži stavke iz preostalih S - s skupova. Zatim, svaka particija se tretira kao jedna stavka iz jednog od dva nova kompozitna skupa, gde je težina te stavke jednaka zbiru težina stavki u datoj particiji. Konačno, particije koje se tretiraju kao stavke iz dva kompozitna skupa se optimalno dodeljuju testovima koristeći proceduru PS2. Okolina 3 se dobija kada se stavke iz dva po dva testa optimalno redistribuiraju, gde je jedan od dva testa onaj sa najvećim zbirom težina stavki. Prvo, testovi su raspoređeni u neopadajućem redosledu njihove težine. Zatim se svaki test uparuje sa najtežim testom tako što se rešava MINIMAX_BSC problem sa dva testa i ispituje se kvalitet dobijenog rešenja. Iako je problem sa dva testa NP-težak, što je dokazano teoremom 1 u pomenutom radu, moguće je redukovati ga u problem poznat kao problem zbira podskupa (eng. *subset sum problem*) i optimalno ga rešiti, što se dokazuje tvrdnjom 2 u istom radu.

Inicijalna konfiguracija. Pre primene heuristike poboljšanja, pohlepni konstruktivni heuristički algoritam daje početno rešenje. Stavke iz prvog, nasumično izabranog, skupa se raspoređuju u T testova redom. Stavke iz svakog sledećeg skupa se uređuju neopadajuće po težini, dok se testovi

uređuju nerastuće, po akumuliranoj težini. Stavke se dodeljuju testovima tako da se prva stavka dodeljuje prvom testu, druga stavka drugom testu i tako redom. Postupak se ponavlja sve dok se sve stavke iz svih skupova ne dodele testovima.

Funkcija pomeraja. Pretražuje se Okolina 1 dok se ne nađe lokalni optimum. Zatim se pretražuje Okolina 2 dok se ne nađe lokalni optimum. Ukoliko je dobijeno rešenje bolje od tekućeg, dobijeno rešenje postaje tekuće rešenje i pretraga se ponavlja od Okolina 1. Ukoliko rešenje nije poboljšano, pretražuje se Okolina 3. Ukoliko je pronađeno rešenje koje je bolje od tekućeg rešenja posle pretrage Okoline 3 ponavlja se pretraga počevši od Okoline 1. Ukoliko je posle pretrage Okoline 3 pronađeno rešenje lošije od trenutnog rešenja, primenjuje se operator diverzifikacije koji uklanja sve stavke is svih testova i postupak vraća na početak gde se svi testovi ponovo kreiraju pohlepnom heuristikom.

Uslov prekida. Kriterijum za prekid rada je istek predviđenog intervala vremena (eng. *wall time*) ili ukoliko se pronađe optimalno rešenje, odnosno optimalna vrednost funkcije cilja koju je u korišćenoj formulaciji problema moguće jednostavno izračunati. Optimalna vrednost funkcije cilja je jednaka prosečnoj težini testa (*172*) .

Autori su poredili rezultate VNS algoritma sa SA algoritmom predstavljenim u radu (Brusco, Köhn, & Steinley, 2013) gde su pokazali da VNS algoritam pokazuje bolje performanse od SA algoritma i dobija optimalna ili skoro optimalna rešenja za veličine banaka stavki do 60,000 i za sastavljanje do 3000 testova.

3.2.2.3 Pretraga sa tabuima

Pretraga sa tabuima (TS), predložena u (Glover, 1989), je metaheuristika koja rešava problem kombinatorne optimizacije tako što efikasno pretražuje prostor rešenja usmeravanjem pretrage. Algoritam počinje nasumično generisanim rešenjem. Pomeraji u prostoru se vrše na osnovu vrednosti funkcije cilja, tako što se bira najbolje rešenje među susednim rešenjima. Prethodni koraci u pretrazi se pamte u strukturi koja je nazvana *tabu* lista, po kojoj je metaheuristika i dobila naziv. U njoj se pamte promene koje su izvršene. Uz pomoć tabu liste se izbegava ponovno vraćanje na rešenja koja su već analizirana. U ovoj metaheuristici se definiše i kriterijum aspiracija koji služi da se poništi zabrana promene koja je zapamćena u tabu listi. Poništavanje zabrane promene je dozvoljeno ako promena vodi ka rešenju boljem od postojećeg rešenja.

U radu (Hwang, Chu, Yin, & Lin, 2008) je predstavljena heuristika za rešavanje problema ASPT, opisanog u odeljku 3.1.1.1.2. Heuristika je zasnovana na tabu metaheuristici. Komponente ovog algoritma su predstavljene u nastavku.

Inicijalna konfiguracija. Banka stavki sadrži I stavki $x_1, x_2,\ldots, x_I$ i listu od M koncepata $c_1,c_2,\ldots,c_M$. Nasumično se sastavlja rešenje $x = (x_{11},x_{12},\ldots,x_{1n}; x_{21},x_{22},\ldots,x_{2n};\ldots;x_{T1},x_{T2},\ldots,x_{Tn})$, gde x_{ij} predstavlja indeks stavke u banci stavki koja pripada testu i na poziciji j. Pretpostavlja se da svaki test ima n stavki, a da ima T testova.

Funkcija pomeraja. Sledeće rešenje za ocenu se generiše nasumičnim odabirom jedne stavke iz vektora x i zamenom drugom stavkom u skladu sa ograničenjima (da se ne naruši dostupnost rešenja). Broj dozvoljenih izmena je unapred definisan i fiksiran na 100. Tabu lista se ažurira informacijama o indeksu testa koji je promenjen i vrednosti promenljive koja je promenjena. Nije dozvoljeno menjati stavke koje su već menjane, a koje se nalaze u tabu listi. Ovaj tabu status se može poništiti ako je novo rešenje bolje od trenutnog rešenja.

Sva rešenja se procenjuju na osnovu izračunate vrednosti funkcije pogodnosti, kada je promena izvršena (151). Funkcija pogodnosti u sebi sadrži i funkcije kazne. Prva funkcija kazne se odnosi na odstupanje od ograničenja koje se odnosi na relevantnost stavke za određeni koncept:

$$R(x) = \sum_{t=1}^{T}\sum_{m=1}^{M} \max\left\{0, \left(h_m - \sum_{i=1}^{n} r_{im}x_{ti}\right)\right\} \tag{148}$$

gde je r_{im} stepen povezanosti stavke x_{ti} i koncepta c_m a h_m donja granica relevantnosti koncepta c_m za svaki test t. Druga i treća funkcija kazne se odnose na ograničenja koja nameću donju (l) i gornju granicu (u) očekivanog vremena za rešavanje testa:

$$L(x) = \sum_{t=1}^{T} \max\left\{0, \left(l - \sum_{i=1}^{n} q_i x_{ti}\right)\right\} \tag{149}$$

$$U(x) = \sum_{t=1}^{T} \max\left\{0, \left(\sum_{i=1}^{n} q_i x_{ti} - u\right)\right\} \tag{150}$$

gde je q_i očekivano vreme odgovora na stavku. Funkcija pogodnosti je definisana kao:

$$J(x) = 1 - \left(\omega_1 Z(x) + \omega_2 R(x) + \omega_3 L(x) + \omega_4 U(x)\right) \tag{151}$$

gde su ω_1, ω_2, ω_3, ω_4 koeficijenti normalizacije tako da funkcija pogodnosti $J(x)$ ima vrednosti u intervalu [0,1]. Što je veća vrednost, to je rešenje bolje. Tekuće rešenje je ono koje ima najbolju vrednost funkcije pogodnosti.

Uslov prekida. Pretraga se zaustavlja kada je broj izvršenih iteracija jednak veličini banke stavki.

Performanse predloženog algoritma su se poredile sa HC algoritmom koristeći istu funkciju pogodnosti i nekoliko banki stavki u rasponu od 500 do 50000 stavki. Zaključeno je da je kvalitet testova sastavljenih korišćenjem predloženog TS algoritma bolji od kvaliteta testova sastavljenih HC algoritmom, ali sa značajno produženim vremenom rada. Drugi faktor koji se ispitivao je broj generisanih rešenja za formiranje okoline. Kao što se i očekivalo, što više rešenja je generisano, to se postizao bolji kvalitet rešenja, ali po cenu velikog utroška procesorskog vremena.

3.2.2.4 Genetski algoritam

Genetski algoritam (GA), koji je uveo Džon H. Holand (Holland, 1973), je metaheuristika pretraživanja koja koristi ideju evolucije kroz prirodnu selekciju i reprodukciju (mutacijom i ukrštanjem) za rešavanje problema optimizacije. Tokom procesa pretraživanja, rešenja se poboljšavaju i približavaju optimalnim rešenjima. Skup rešenja se naziva populacija. Jedno rešenje se naziva hromozomom koji je sastavljen od gena. GA generiše poboljšanu populaciju iz tekuće populacije koristeći prirodnu selekciju najsposobnijih. Sledeća generacija se kreira korišćenjem operatora mutacije, selekcije i ukrštanja.

Transponovano na problem sastavljanja testa, skup paralelnih testova je populacija. Test je hromozom i sastoji se od I bitova (gena), gde je I broj stavki u banci. Svaka pozicija bita i je povezana sa promenljivom odlučivanja x_i koja predstavlja stavku iz banke stavki i može imati vrednost jedan ili nula, ako je stavka izabrana za test ili nije, respektivno.

U radu (Sun, Chen, Tsai, & Cheng, 2008) je primenjen GA za simultano sastavljanje paralelnih testova formulisanom u odeljku 3.1.1.2.2. Cilj je da se minimizira suma kvadratnih razlika između vrednosti TIF i TTIF, na svim nivoima sposobnosti.

Inicijalna konfiguracija. T testova je nasumično sastavljeno tako što su promenljivama odlučivanja nasumično dodeljene vrednosti jedan ili nula sa ograničenjem da u svakom testu ima tačno n promenljivih sa vrednošću jedan. Definišu se vrednosti za verovatnoću mutacije $0.0001 \leq p_m \leq 0.01$ i za verovatnoću ukrštanja $0.8 \leq p_c \leq 0.9$ testova. Definiše se i maksimalan broj generacija.

Funkcija pomeraja. Vrednosti funkcije podobnosti za sve testove u populaciji se izračunavaju pomoću formule (152).

$$fpod_t = \sum_{k=1}^{K} \left(TIF_t(\theta_k) - TTIF(\theta_k)\right)^2 \quad t = 1, \ldots, T \tag{152}$$

Podobnost testa predstavlja verovatnoću za reprodukciju p_r testa. Verovatnoća ukrštanja testa (p_c) određuje da li se testovi menjaju ukrštanjem promenljivih odlučivanja. Bira se određeni broj međusobno susednih (po rednom broju stavke u testu) stavki testa koje se razmenjuju između dva testa, sa verovatnoćom ukrštanja p_c. Zatim se biraju stavke, za svaki test posebno, sa verovatnoćom mutacije p_m, koje menjaju svoje vrednosti (sa 0 promena na 1 ili sa 1 promena na 0). Dobijeni testovi se stavljaju u novi skup testova (nova populacija). Iz skupa testova, T testova se bira na osnovu funkcije podobnosti (verovatnoće za reprodukciju). Što je manja vrednost funkcije podobnosti testa, veća je verovatnoća odabira testa.

Uslov prekida. Dostignut je maksimalan broj generacija.

Autori su predstavili rezultate sastavljanja 40 paralelnih testova iz banke od 1000 stavki i uporedili kvalitet sastavljenih testova sa kvalitetom testova kreiranih koristeći WDM model. Zaključak je da su se korišćenjem GA dobili kvalitetniji testovi, ali tokom znatno dužeg vremena.

3.2.2.5 Pčelinji algoritam

Pčelinji algoritam (BA) je algoritam za pretragu koji oponaša ponašanje medonosnih pčela prilikom traženja hrane (Pham, et al., 2006). Medonosne pčele imaju za cilj da pronađu bogat izvor hrane. Medonosne pčele žive u košnici gde čuvaju med koji su sakupile. Postoje tri vrste pčela u ovom procesu: izviđači, posmatrači i sakupljači. Pčele izviđači traže izvor hrane i kada ga pronađu, jave lokaciju drugim pčelama iz košnice tako što izvode takozvani „ples njihanjem“. Trajanje ovog plesa je proporcionalan količini hrane na izvoru. Posmatrači, kada saznaju za izvor hrane, postaju sakupljači i kreću po hranu.

U teoriji kombinatorne optimizacije, algoritam je sledeći. Definiše se određen broj pčela izviđača koje su nasumično raspoređene u prostoru pretrage. Kvalitet mesta gde se pčele nalaze (kandidat-rešenja) se procenjuje. Pčele kod kojih je kvalitet mesta (vrednost funkcije pogodnosti) najveći se biraju da bi se u okolini tog pronađenog mesta nastavila dalja pretraga. Dodeljuje se više pčela da pretražuje unapred određeni broj mesta. Ukoliko se među novim mestima nađe bolji kvalitet, još pčela se regrutuje da nastavi pretragu u novoj okolini tih mesta. Na kraju iteracije će postojati predstavnici svake grupe za pretragu i pčele izviđači koje pseudoslučajno izvršavaju nasumičnu pretragu.

Transponovano na APTA problem, test predstavlja mesto izvora hrane koji pronađe jedna pčela, dok je funkcija cilja jednaka kvalitetu mesta izvora hrane. Pčela predstavlja proces kreiranja jednog testa. Pčela izviđač je proces kojim se kreiraju početni testovi, dok je pčela posmatrač (sakupljač) proces kojim se poboljšavaju testovi.

U radu (Songmuang & Ueno, 2011) je predložena metoda zasnovana na BA metaheuristici za sastavljanje paralelnih testova sa ciljem da se minimizuje razlika između testova (odeljak 3.1.1.2.2).

Inicijalna konfiguracija. Kreira se N > T pčela izviđača koje kreiraju početne testove. Jedna pčela kreira jedan test.

Testovi se kreiraju na isti način. Prva stavka za test se bira na osnovu verovatnoće izbora stavke (p_i) (153). Za svaku preostalu stavku i se ažurira verovatnoća odabira stavke (p_i), kada je j stavki već izabrano:

$$p_i = \frac{\frac{d_i}{q_i}}{\sum_{i \in S_{n-j}} \frac{d_i}{q_i}}, \quad i \in S_{n-j}. \tag{153}$$

$$q_i = \sum_{k=1}^{K} \left| \frac{TTIF(\theta_k) - \sum_{i=1}^{I} IIF_i(\theta_k) x_i}{n - j + 1} - IIF_i(\theta_k) \right| \tag{154}$$

gde je q_i koeficijent izbora stavke (greška koja postoji ako je stavka i uključena u test) i d_i je promenljiva stavke i koja ima vrednost nula se ako dodavanjem stavke i ne zadovoljava nijedno ograničenje testa, ili jedan u suprotnom, S_{n-j} je skup preostalih stavki nakon odabira j stavki.

Na kraju ovog koraka je kreirano je N testova

Funkcija pomeraja. Određuje se maksimalan broj pčela posmatrača za poboljšanje svakog kreiranog testa, N_{max}. Za svaki kreiran test određuje se verovatnoća odabira testa (p_t) na sledeći način:

$$p_t = \frac{\frac{1}{e_t}}{\sum_{t=1}^{T} \frac{1}{e_t}} \tag{155}$$

$$e_t = \sum_{k=1}^{K} \left| \sum_{i=1}^{I} IIF_i(\theta_k) x_{it} - TTIF(\theta_k) \right| \tag{156}$$

Za svaki test se određuje broj pčela posmatrača (N_t) koje će da poboljšaju taj test.

$$N_t = N_{max} \cdot p_t \tag{157}$$

Nadalje se poboljšanje svakog testa obavlja tako što se menjaju pojedine stavke testa na osnovu novih formula za verovatnoću odabira stavke.

Uslov prekida. Kada se dostigne vreme predviđeno za izvršenje. Rešenje je T testova sa najmanjim vrednostima grešaka e_t (156).

Da bi se smanjilo vreme izvršavanja rada BA algoritma, realizovana je paralelizovana implementacija algoritma za izvršavanje na višeprocesorskom računaru, što je u slučaju BA algoritma moguće zato što se rešenja nezavisno poboljšavaju. Pokazano je da je vreme izvršavanja paralelizovanog BA algoritma je, značajno kraće od vremena izvršavanja GA algoritma i BST metoda.

3.2.2.6 Optimizacija rojem čestica

Optimizacija rojem čestica (PSO) simulira ponašanje jata ptica. Ptice se okupljaju, razilaze i ponovo grupišu kada traže hranu (Kennedy & Eberhard, 1995). Ptice (čestice) imaju dodeljenu poziciju i brzinu i teže da prate bolje pozicije uzimajući u obzir i pozicije koje je pronašla druga ptica u jatu, sa ciljem da se pronađe hrana. Ovo ponašanje vodi celo jato do hrane.

U teoriji kombinatorne optimizacije, određeni broj čestica leti u prostoru pretrage. Svaka čestica je kandidat-rešenje i predstavljena je vektorom u tom prostoru. Čestica ima pridruženu i brzinu kojom se kreće, koja predstavlja iznos moguće promene nad kandidat-rešenjem. Čestice međusobno komuniciraju. Svaka čestica ažurira svoj položaj sa ciljem da se približi optimalnoj vrednosti na osnovu dva elementa: najbolje pozicije koju je sama posetila ($pbest_i$) i najbolje pozicije koju je roj posetio ($gbest$). Za svaku česticu može da se izračuna koliko je udaljena od najbolje pozicije koju je roj posetio. U svakom koraku čestica menja svoju brzinu i položaj (Talbi, 2009).

Primenjeno na APTA problem, skup paralelnih testova predstavlja česticu. Njoj je pridružen dvodimenzioni vektor (matrica) položaja $X_j = (x_{11}, x_{12}, \ldots, x_{2I}, x_{22}, \ldots, x_{2I}, \ldots, x_{T1}, \ldots, x_{TI})$ $j = 1,\ldots,J$, gde je J ukupan broj čestica koje čine roj (kandidat-rešenje), dok x_{it} ima vrednost $x_{it} = 1$ ukoliko je *i-ta* stavka pridružena testu t, a $x_{it} = 0$ u suprotnom, gde je $i = 1,\ldots,I$, $t = 1,\ldots,T$. Svakoj čestici se

pridružuje i vektor brzine $V_j = (v_{11}, v_{12}, \ldots, v_{TI})$. Položaj čestice je definisan pomoću funkcije pogodnosti, dok je njena brzina definisana verovatnoćama da će stavka biti izabrana.

U radu (Yin, Chang, Hwang, Hwang, & Chan, 2006) je primenjena heuristika zasnovana na PSO metaheuristici za simultano sastavljanje testova sa ciljem da se minimizuje razlika između težine generisanih testova i ciljne težine (odeljak 3.1.1.1.1).

Inicijalna konfiguracija. Broj stavki za svaki test se nasumično određuje prema pravilu integriteta (158), da bi se zadovoljilo ograničenje ukupnog vremena trajanja testa:

$$\frac{l}{\max\limits_{i=1,\ldots,I}\{q_i\}} \leq \sum_{i=1}^{I} x_{it} \leq \frac{u}{\min\limits_{i=1,\ldots,I}\{q_i\}} \quad t = 1, \ldots, T \tag{158}$$

gde je q_i očekivano vreme za odgovor na stavku i, l i u su donja i gornja granica očekivanog vremena za odgovor na svaku stavku iz banke stavki. Broj čestica (J) se nasumično bira. Svakoj stavki u banci stavki je dodeljena verovatnoća izbora:

$$v_i = \frac{S - |p_i - TDiff|}{S} \quad i = 1, \ldots, I \tag{159}$$

gde je S konstanta koja se unapred dešiniše, a *TDiff* ciljna težina za svaki test. Stavke testova se biraju na osnovu njihove verovatnoće izbora (159). Brzina stavke (dela čestice) v_{ij} inicijalno je jednaka verovatnoći izbora stavke v_i. Inicijalno generisani testovi kandidat-rešenja, mogu da ne zadovoljavaju ograničenja. U toku postupka će kandidat-rešenja da se poboljšavaju ka dopustivim rešenjima.

Funkcija pomeraja. Svakom kandidat-rešenju je dodeljena funkcija pogodnosti koju treba minimizirati:

$$F_t(x) = Z_t + w_1\alpha + w_2\beta + w_3\gamma \quad t = 1, \ldots, T \tag{160}$$

gde su w_1, w_2, w_3 ponderi za kazne α, β, γ za narušavanje ograničenja i Z_t je funkcija cilja (161).

$$Z_t = \sum_{t=1}^{T} \left| \frac{\sum_{i=1}^{I} p_i \; x_{it}}{\sum_{i=1}^{I} x_{it}} - TDiff \right| \tag{161}$$

Vrednost funkcije pogodnosti se koristi da bi se odredilo najbolje kandidat-rešenje u roju (*gbest*). Ukoliko je $pbest_i < gbest$, tada se čestica ne menja i $gbest = pbest_i$. U suprotnom, stavke u česticama koje se razlikuju od stavki u čestici koja ima vrednost *gbest* se menjaju u skladu sa njihovim verovatnoćama. Brzina čestice se menja i izračunava na osnovu sledeće formule:

$$v_{it} = \frac{v_{it}^*}{2v_{max}} + 0.5 \quad i = 1, \ldots, I \quad t = 1, \ldots T \tag{162}$$

gde je v_{it}^* prethodna vrednost v_{it}, a v_{max} najveća vrednost verovatnoće stavke iz rešenja *gbest*.

Uslov prekida. Proces se zaustavlja kada se postigne unapred definisani maksimalni broj iteracija. Rešenje je *gbest*.

U radu (Ho, Yin, Hwang, Shyu, & Yean, 2009) je model proširen na višeciljnu funkciju. Cilj je da se minimizira maksimalna razlika u prosečnoj diskriminativnosti testova i minimizira maksimalna razlika između težine testa i ciljne težine (odeljak 3.1.1.1.1). U radu je poređen kvalitet generisanih rešenja dobijenih PSO algoritmom sa rešenjima dobijenim koristeći GA algoritam. Zaključili su da PSO algoritam proizvodi bolji kvalitet rešenja.

3.2.2.7 Algoritam selekcije klonova

CLONALG metaheuristika, koja je prvi put predložena u radu (de Castro & Von Zuben, 2002), je inspirisana odbrambenim sistemom organizma koji reaguje na patogene. Kada patogen uđe u organizam, u njemu se stvaraju antitela kao reakcija na antigen patogena.

U teoriji kombinatorne optimizacije, antitelo predstavlja kandidat-rešenje problema, i povezano je sa određenim antigenom koji predstavlja funkciju cilja. Afinitet antitela prema antigenu je određen nekom metrikom rastojanja (kao što je Euklidsko rastojanje) koja zavisi od konkretnog problema koji se rešava. Antitela se zatim kloniraju u količini koja je proporcionalna njihovom afinitetu. Klonovi zatim mutiraju, i odabira se određeni broj klonova koji imaju najveću vrednost afiniteta, za novu populaciju. Značajna karakteristika CLONALG algoritma jeste što pretraga rešenja može da se obavlja paralelno zato što se rešenja poboljšavaju nezavisno.

Transponovano na problem ASPT, antitelo predstavlja test, antigen predstavlja specifikaciju testa. Afinitet antitela prema antigenu je obrnuto srazmeran odstupanju testa od specifikacije testa.

U radu (Chang & Shiu, 2012) je predložen prilagođen CLONALG algoritam za simultano sastavljanje paralelnih testova sa ciljem da se minimizuje odstupanje TIF-a od TTIF-a, kao i druga odstupanja (odeljak 3.1.1.2.4)

Zahvaljujući karakteristici algoritma da se rešenja poboljšavaju nezavisno, model za redno sastavljanje testova se koristi za simultano sastavljanje paralelnih testova. Time se izbegava mana simultanog sastavljanja testova, a to je povećanje broja promenljivih, kao i mana rednog sastavljanja testova, gde se javlja nejednakost sastavljenih testova, takva da su ranije sastavljeni testovi boljeg kvaliteta nego testovi koji su kasnije sastavljeni.

Inicijalna konfiguracija. Nasumičnim odabirom stavki se redno sastavlja unapred odabrani broj testova ($N > T$). Stavke odabrane za svaki test se uklanjaju iz banke stavki pre sastavljanja sledećeg testa. Vrednost afiniteta se izračunava za svaki test pomoću funkcije (163).

$$F_t(x) = -(w_1\alpha + w_2\beta + w_3\gamma + w_4\chi + w_5\delta) \tag{163}$$

$$\sum_{k=1}^{K}\left|\sum_{i=1}^{I} I_i(\theta_k)x_i - TTIF(\theta_k)\right| = \alpha \tag{164}$$

$$\sum_{h=1}^{H}\left|\sum_{i=1}^{n} c_{x_i h} - C_h\right| = \beta \tag{165}$$

$$\sum_{v=1}^{V}\left|\sum_{i=1}^{n} s_{x_i v} - S_v\right| = \gamma \tag{166}$$

$$\sum_{m=1}^{M}\left|\sum_{i=1}^{n} q_{x_i m} - Q_m\right| = \chi \tag{167}$$

$$max\left\{0, \sum_{i=1}^{n} r_{x_i} - R^u\right\} = \delta \tag{168}$$

gde su w_1, w_2, w_3, w_4, w_5 ponderi funkcije kazne (α - odstupanje od TTIF-a (164), β - odstupanje od broja stavki u svim oblastima (165), γ - odstupanje od broja stavki u svim veštinama (166), χ - odstupanja od broja stavki u vrstama stavki (167), δ - pozitivno odstupanje ukupnog broja reči od gornje granice broja reči ili ima vrednost nula (168)). Funkcija afiniteta je jednaka negativnoj sumi ponderisanog odstupanja od ograničenja (163). Veća vrednost funkcije afiniteta znači bolje rešenje.

Funkcija pomeraja. *T* testova sa najvećim vrednostima funkcije afiniteta se biraju za kloniranje. Broj klonova za svaki test je jednak dužini testa (*n*). Nakon generisanja *n* klonova za svaki od *T* testova, klonovi mutiraju. Koristi se mutacija u jednoj tački (za prvi klon se mutira prva stavka, za drugi klon druga stavka i tako dalje). Stavke izabrane za kloniranje se zamenjuju sa pseudoslučajno izabranom stavkom iz banke stavki (stavke koje su sadržane u testu su smeštene u posebnu listu da se ne bi ponovo izabrale, zajedno sa svim stavkama odabranim za ostale testove). Nakon mutacije *n* klonova svakog testa, vrednost funkcije afiniteta se izračunava za svaki klon. Od svih klonova se bira klon sa najvećom vrednošću funkcije afiniteta. Ukoliko je ta vrednost veća od vrednosti funkcije afiniteta polaznog testa, onda se polazni test zamenjuje klonom i proces se ponavlja. U suprotnom, test ulazi u sledeđeću iteraciju.

Uslov prekida. Proces se završava kada se dostigne unapred definisan maksimalni broj iteracija.

Algoritam je testiran nad bankom sa 4000 stavki i brojem testova od 1 do 100. Vršeno je poređenje sa GA algoritmom, i rezultati pokazuju da se izvršavanjem CLONALG algoritma dobijaju rezultati sa manjim odstupanjima od optimalnog rešenja nego kod izvršavanja algoritma GA za približno jednako vreme izvršavanja oba algoritma

3.2.3 Sumarni pregled postojećih heuristika za rešavanje problema ASPT

U ovom odeljku će sumarno biti prikazani pristupi koji su prethodno detaljno opisani. Zatim će biti ukrštene formulacije problema ASPT sa njihovim (meta)heurističkim rešenjima, kako bi se ukazalo na postojeće i potencijalno nepostojeće metode u određenim ćelijama dvodimenzionalne matrice. Na kraju će se diskutovati mogući dalji pravci istraživanja primene (meta)heurisičkih metoda na formulacije problema ASPT.

Tabela 3.3 prikazuju sumarno (meta)heuristike opisane u ovom odeljku, a koje su predložene u literaturi za rešavanje problema ASPT. Kolona *Izvor* sadrži citate radova u kome je problem ASPT rešavan. U sledećoj koloni je obeleženo koja (meta)heuristika se koristila za rešavanje problema. Poslednja kolona sadrži informacije o veličini banke stavki (*#stavki*) kao i broj testova koji se sastavljao (*#testova*).

Tabela 3.4 prikazuje najsavremenije pristupe ASPT ukrštanjem poznatih formulacija problema sa primenjenim (meta)heurističkim rešenjima. Prazne ćelije ukazuju na moguće prostore za dalje istraživanje, primenom neke od postojećih (meta)heuristika na druge formulacije problema.

Poređenje performansi (meta)heurističkih pristupa rešavanju APTA prevazilazi okvire ove disertacije, prvenstveno iz razloga navedenih u radu (Hussain, Salleh, Cheng, & Shi, 2019). Primećeno je da se analize performansi (meta)heuristika uglavnom sprovode na osnovu najgore, najbolje, srednje i standardne devijacije vrednosti funkcije cilja, što nije dovoljno. Potrebno je uspostaviti opšte usaglašene kriterijume ocene performansi kako bi se uspostavio pouzdan zaključak o kvalitetu konkretne (meta)heuristike. Jedan od izazovnih pravaca u budućem istraživanju može biti upravo kreiranje zajedničkog skupa uslova i kriterijuma za merenje performansi i poređenje suštinski različitih (meta)heurističkih pristupa. Uspostavljanje standardnog skupa banaka stavki, ili standardnih algoritama za generisanje sintetičkih banaka stavki, mogao bi biti među prvim koracima u tom pravcu.

Poređenje kvaliteta i efikasnosti (meta)heurističkih pristupa rešavanja problema APTA bi trebalo da podrazumeva ne samo da se koriste iste banke stavki, već da se koristi ista formulacija problema. Jedan reprezentativan primer je rad (Pereira & Vila, 2015), gde su autori primenili VNS heuristiku da reše istu formulaciju problema koja je uvedena u radu (Brusco, Köhn, & Steinley, 2013) (gde je problem rešen SA metaheuristikom), a zatim je urađeno pošteno i adekvatno poređenje. U navedenom primeru je poređenje bilo moguće zato što je korišćena ista formulacija problema i isti algoritam za kreiranje banke stavki.

Kao što je predloženo u radu (Hussain, Salleh, Cheng, & Shi, 2019), dalje istraživanje bi moglo da bude u pravcu izgradnje softverskog okvira koji bi pomogao u poređenju različitih (meta)heurističkih metoda, bez njihovog razvoja od nule. Okvir bi trebalo da obezbedi mogućnost ujednačene analize performansi (meta)heuristika i njhovog poštenog poređenja.

Tabela 3.3 Pregled metoda za rešavanje problema ASPT u literaturi

Izvor	Heuristike			Metaheuristike							Veličina problema	
	B&B zasn.	Pohlepni algoritmi	Slučajno uzorkovanje	SA	TS	VNS	PSO	GA	BA	CLONALG	#stavki	#testova
(Ackerman 1989)		✓									600	6
(Adema 1988)	✓										100	2
(Adema et al. 1991)	✓										300	2
(Adema 1992)	✓										600	6
(Armstrong et al. 1992)		✓									510	6
(Armstrong et al. 1994)		✓									567	6
(Belov and Armstrong 2006)	✓										1,830	16
(Brusco et al. 2013)				✓							300-6,000	2-300
(Bojić, et al. 2016)		✓									600	5
(Chang and Shiu 2012)										✓	4,000	2-100
(Chen et al. 2012)			✓								540	5
(Chen 2015)			✓								540	5
(Nguyen et al. 2013)	✓										20,000-50,000	
(Ho, et al. 2009)							✓				250-50,000	2-4
(Hwang, et al. 2008)					✓						500-50,000	2-4
(Luecht 1998)		✓									3165	4
(Pereira and Vila 2015)						✓					300-60,000	2-12,000
(Songmuang and Ueno 2011)									✓		5,000-20,000	5-733
(Stocking at el. 1993)		✓									5,000	14
(Sun, et al. 2008)								✓			1,000	6
(van der Linden and Adema 1998)	✓										753	3
(van der Linden and Boekkooi-Timminga1989)	✓										600	6
(Yin, et al. 2006)							✓				5-10,000	10

Legenda: SA-simulirano kaljenje; TS-pretraga sa tabuima; VNS-pretraga promenljivih okolina; GA-genetski algoritam; BA-pčelinji algoritam; PSO-optimizacija rojem čestica; CLONALG-algoritam selekcije klonova;

Tabela 3.4 Unakrsna tabela formulacija i heurističkih metoda za rešavanje problema ASPT

		Ciljevi klasične teorije testa	Ciljevi teorije odgovora na stavku	Uparivanje sa srodnim testovima	Maksimizacija broja testova
Heuristike	Zasnovane na B&B		(Nguyen et al. 2013) (van der Linden and Adema 1998) (Adema 1992) (Adema et al. 1991) (van der Linden and Boekkooi-Timminga1989) (Adema 1988)		(Ishii et al. 2014) (Belov and Armstrong 2006)
	Pohlepni algoritmi		(Luecht 1998) (Swanson and Stocking 1993) (Ackerman 1989)	(Bojić, et al. 2016) (Armstrong et al. 1992)	
	Slučajno uzorkovanje			(Chen 2015) (Chen el al. 2012)	
Metaheuristike	SA		(Brusco et al. 2013)		
	TS	(Hwang, et al. 2008)			
	VNS	(Pereira and Vila 2015)			
	GA		(Sun, et al. 2008)		
	BA		(Songmuang and Ueno 2011)		
	PSO	(Ho, et al. 2009) (Yin, et al. 2006)			
	CLONALG		(Chang and Shiu 2012)		

Budući rad može da uključi i primenu novih formulacija kao što je predloženi pristup *ograničenja slučajnosti* (*eng. Chance-Constrained,* CC) (Proietti, Matteucci, Mignani, & Veldkamp, 2020). U citiranom radu se razmatra nepouzdanost u procenjenim atributima stavki. Umesto fiksne vrednosti za TIF, nepouzdanost je uključena u predloženoj funkciji cilja kao slučajna promenljiva. Heuristika koja se koristi za rešavanje je zasnovana na SA metaheuristici. SA je u ovoj disertaciji navedena kao jedna od metaheuristika koje se već koriste za rešavanje APTA problema (Brusco, Köhn, & Steinley, 2013), što je primer koji sugeriše da je ona pogodna za rešavanje APTA problema.

U pregledu literature nisu uključene formulacije APTA problema koje uključuju nekoliko funkcija cilja (više od jedne funkcije optimizacije). Razlog za ovakvu odluku leži u činjenici da se ovakve funkcije optimizacije pretežno koriste za sastavljanje kognitivnih testova, koji su van okvira ovog istraživanja. Na primer, u radu (Lin, Jiang, Gong, Zhan, & Zhang, 2019) se predlaže rešavanje problema sa više funkcija cilja zasnovano na metaheuristici optimizacije rojem čestica za sastavljanje testova paralelne kognitivne dijagnostike. Jedno od mogućih budućih istraživanja je da se APTA problem formuliše kao dvodimenzionalni problem pakovanja u korpe (dve funkcija cilja), gde se razmatraju dve dimenzije: težina i diskriminacija. Trenutno, većina pristupa kombinuje ova dva atributa stavke, koristeći težinske koeficijente, u jednu funkciju cilja.

Jedan od nedostataka (meta)heurističkih algoritama je potreba za izračunavanjem vrednosti funkcije cilja za svako rešenje, a to smanjuje performanse algoritma kada se veličina problema povećava (Dokeroglu, Sevinc, Kucukyilmaz, & Cosar, 2019). Ovaj problem može da se ublaži korišćenjem efikasne konstruktivne heuristike kao što je NEH (Nawaz, Enscore Jr, & Ham, 1983) i koja bi mogla da se primeni na 1DBPP formulaciju problema ASPT. U ovoj disertaciji se koristi ideja konstruktivne heuristike NEH pri rešavanju iste formulacije problema koju rešavaju na druge načine u radovima (Brusco, Köhn, & Steinley, 2013) i (Pereira & Vila, 2015). Heuristika NEH se ovde ne koristi za dobijanje početnog rešenja nad kojim se onda primenjuje neka heuristika

poboljšanja (kao u većini slučajeva gde se koriste konstruktivne heuristke), već kao mehanizam koji u potpunosti rešava problem ASPT, bez naknadnog izvršavanja neke heuristike poboljšanja. Upravo iz razloga oslanjanja na heuristiku NEH u predloženom rešenju ove disertacije, u narednom odeljku 3.3 biće detaljno opisana ova heuristika.

3.3 NEH heuristika

Metod NEHTA, predložen u ovoj disertaciji, rešava problem sastavljanja paralelnih testova, korišćenjem jedne ideje NEH heuristike (Nawaz, Enscore Jr, & Ham, 1983) za smanjenje ukupnog broja permutacija. To je razlog što je ova heuristika izdvojena i posebno opisana u ovom odeljku. Nakon definicije problema koji rešava NEH heuristika i opisa NEH algoritma, biće dat ilustrativni primer izvršenja algoritma. U sledećem poglavlju biće uspostavljena analogija između pojmova NEH heuristike i pojmova ASPT korićenih u metodu NEHTA.

NEH heuristika rešava permutacioni problem određivanja redosleda poslova u protočnom izvršavanju na nizu mašina (eng. *Permutation Flow-Shop Sequencing Problem*, PFSP), gde n poslova treba da se izvrši na m mašina, tako što se svaki posao delimično izvršava na svakoj od mašina, redom.

Svaki posao i ima predviđeno vreme izvršavanja na mašini j, $q_{i,j}$ ($i = 1,\ldots,n$, $j = 1,\ldots,m$), odnosno ukupno vreme izvršavanja svih delova posla i iznosi:

$$Q_i = \sum_{j=1}^{m} q_{i,j}, \qquad i = 1,\ldots,n \tag{169}$$

Cilj je da se definiše optimalni redosled poslova takav da je ukupno vreme od početka izvršavanja prvog posla na mašini 1 do trenutka kada će se izvršiti poslednji posao (nadalje stvarno vreme izvršavanja svih poslova), (eng. *makespan*) na mašini m minimalno.

Da bi se našlo optimalno rešenje, trebalo bi ispitati vremena izvršavanja svih poslova za sve moguće permutacije redosleda njihovog izvršavanja. U pitanju su permutacije n elemenata, te je broj takvih permutacija je $n!$. Zato ovaj problem pripada klasi NP-kompletnih problema (Garey, Johnson, & Sethi, 1976). NEH heuristika predlaže metod kojim se broj permutacija radikalno smanjuje i na taj način i vreme izvršavanja algoritma. Opis heuristike je u nastavku.

Poslovi se dodeljuju mašinama tako što se prvi posao dodeli prvoj mašini, pa kad završi izvršavanje odgovarajućeg svog dela na njoj, on se dodeljuje drugoj mašini, a drugi posao se dodeljuje oslobođenoj prvoj mašini i tako redom. U svakom trenutku postoje dve liste poslova: lista (već) raspoređenih i lista (još) neraspoređenih poslova. Inicijalno, neraspoređeni poslovi se uređuju prema nerastućim ukupnim vremenima izvršavanja svih delova posla. Lista raspoređenih poslova je inicijalno prazna. U prvom koraku, iz liste neraspoređenih poslova se prebacije prvi posao u listu raspoređenih. Raspoređivanje svakog narednog posla iz liste neraspoređenih poslova se svodi na pronalaženje njegove pozicije unutar liste raspoređenih poslova. Ova pozicija se pronalazi tako da relativne pozicije već raspoređenih poslova ostanu iste. Za svako moguće novo uređenje se izračunava stvarno vreme izvršavanja svih poslova i bira se uređenje takvo da je to vreme minimalno. Postupak se ponavlja sve dok se i poslednjem poslu iz liste neraspoređenih poslova ne pronađe optimalna pozicija u listi raspoređenih poslova. Ovakvim postupkom, dobija se da je ukupan broj računanja stvarnog vremena izvršavanja svih poslova jednak $n(n + 1)/2 - 1$, odnosno reč je o algoritmu kvadratne složenosti. Algoritam je opisan sledećim pseudokodom (Pseudokod 3.1), uz pretpostavku da su liste indeksirane počevši od vrednosti indeksa 1. Indeks j predstavlja mesto tekućeg posla za raspoređivanje u listi neraspoređenih poslova, indeks k predstavlja mesto u listi raspoređenih poslova na koje se privremeno umeće tekući posao, radi ispitivanja tekućeg

rasporeda, a indeks *poz* predstavlja mesto u listi raspoređenih poslova na koje se trajno umeće tekući posao.

Algoritam

```
 urediti poslove po nerastućem redosledu ukupnih vremena izvršavanja njihovih delova
 ukloniti prvi posao iz liste neraspoređenih poslova
 smestiti prvi posao u listu raspoređenih poslova
 j ←2
 repeat
     ukloniti posao sa pozicije j iz liste neraspoređenih poslova
      k ← 1
      inicijalizovati vreme najboljeg uređenja na maksimalno moguće vreme
      repeat
        poz ← 0
        umetnuti posao na poziciju k u listi raspoređenih poslova
        izračunati vreme izvršavanja uređenja
        if vreme izvršavanja uređenja  < vreme najboljeg uređenja then
           poz ← k
          vreme najboljeg uređenja ← vreme izvršavanja
        end if
        ukloniti posao sa pozicije k iz liste raspoređenih poslova
        k ← k + 1
      until k > j+1
      umetnuti posao na poziciju poz
      j ← j +1
 until j ≤ n
```

Pseudokod 3.1 – NEH algoritam

Rad NEH heuristike može da se opiše na primeru pet poslova (J1, J2, J3, J4, J5) koji se izvršavaju na tri mašine (M1, M2, M3), (Slika 3.3). Za svaki posao je, u ćelijama tabele, prikazano trajanje dela posla na svakoj mašini. Delovi posla treba da se izvršavaju redno na mašinama (prvi deo posla na mašini M1, pa onda drugi deo posla na mašini M2 i na kraju treći deo posla na mašini M3).

a

	M1	M2	M3	
J1	3	7	4	14
J2	6	2	3	11
J3	9	7	3	19
J4	8	6	2	16
J5	9	7	4	20

b

	M1	M2	M3	
J5	9	7	4	20
J3	9	7	3	19
J4	8	6	2	16
J1	3	7	4	14
J2	6	2	3	11

Slika 3.3 – Primer pet poslova i tri mašine

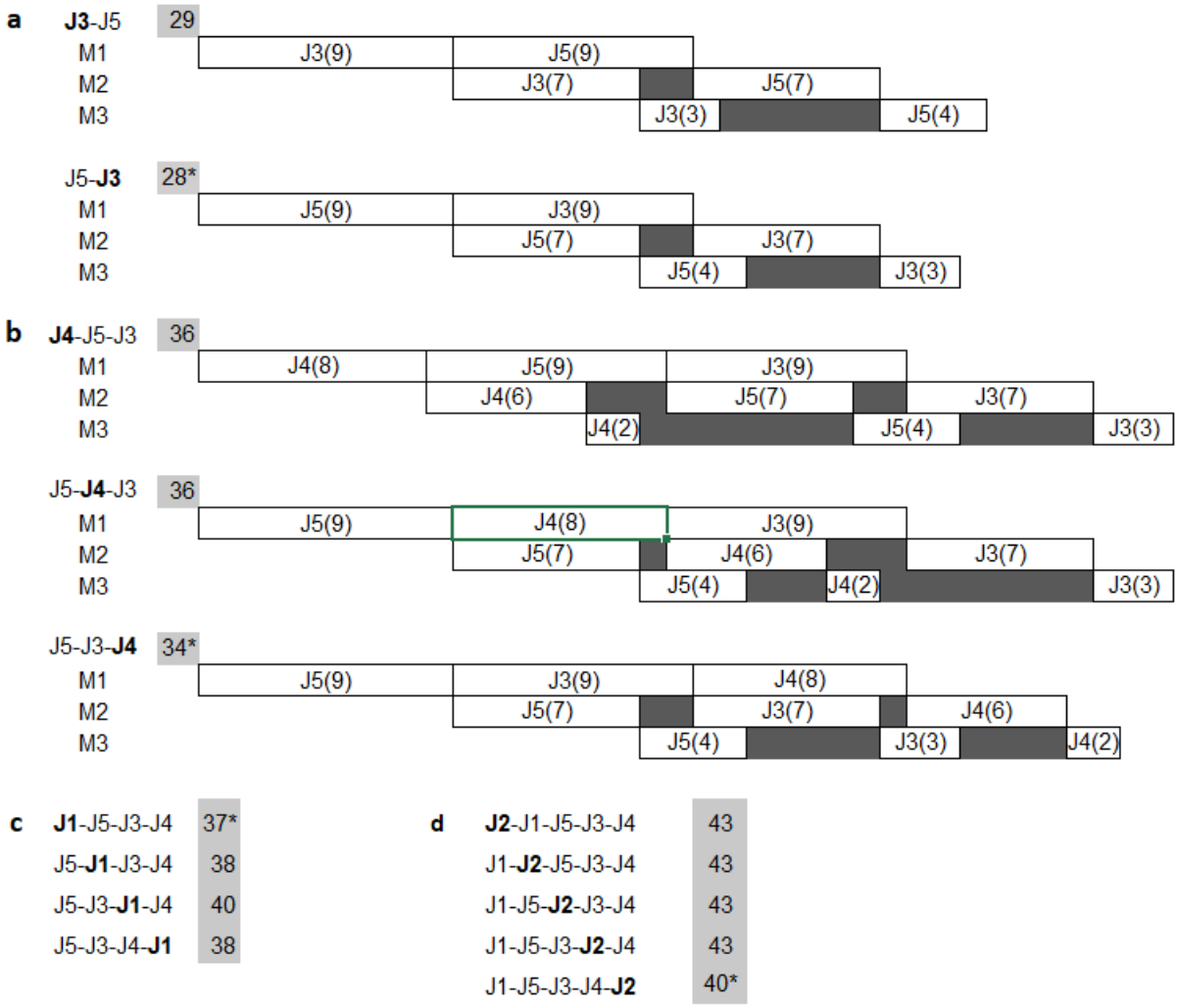
Za svaki posao se prvo izračunava ukupno vreme izvršavanja delova posla na svim mašinama (169). To vreme je prikazano u poslednjoj koloni označenoj sivom bojom (Slika 3.3 a). Zatim se poslovi uređuju po ukupnom vremenu izvršavanja delova posla nerastuće (Slika 3.3 b). Uređena lista neraspoređenih poslova je (J5, J3, J4, J1, J2).

U prvom koraku se iz liste neraspoređenih poslova uzima prvi posao, J5, i smešta u praznu listu raspoređenih poslova. Zatim se iz liste neraspoređenih poslova uzima prvi sledeći posao, J3, i ispituje se redosled izvršavanja poslova J3-J5 i J5-J3 (Slika 3.4 a). Pri redosledu J3-J5, deo posla J5(7) ne može da započne sa izvršavanjem na mašini M2 sve dok se ne završi izvršavanje dela posla J5(9) na mašini M1. Zato mašina M2 neki interval vremena ne radi, što je označeno crnim poljem (Slika 3.4 a).

Potom se izračunava stvarno vreme izvršavanja poslova od kako je prvi posao počeo da se izvršava, pa do završetka. Minimalno stvarno vreme izvršavanja poslova je za uređenje J5-J3 i iznosi 28. Posao J3 se smešta na poziciju 2 u listi raspoređenih poslova. Nadalje relativna pozicija poslova J5 i J3 ostaje ista, a to je da se posao J5 izvršava pre posla J3.

Bira se sledeći posao iz liste neraspoređenih poslova, J4, i traži se njegova pozicija u listi raspoređenih poslova, gde se već nalaze poslovi J5 i J3 uređeni na način J5-J3. Moguća uređenja su J4-J5-J3, J5-J4-J3 i J5-J3-J4. Za svako uređenje se izračunava stvarno vreme izvršavanja poslova (Slika 3.4 b). Uređenje J5-J3-J4 je uređenje sa minimalnim stvarnim vremenom izvršavanja poslova i iznosi 34. Posao J4 se smešta na poziciju 3 u listi raspoređenih poslova. Postupak se ponavlja na isti način dodavanjem posla J1 (Slika 3.4 c) i, na kraju, posla J2 (Slika 3.4 d). Konačno rešenje postupka je uređenje J1-J5-J3-J4-J2 sa ukupnim vremenom izvršavanja 40.

Koristeći ovaj metod, 14 permutacija je ispitivano, a ne 120 koliko bi ih bilo da su se sve moguće permutacije redosleda poslova ispitivale.



Slika 3.4 - Primer rada NEH heuristike za pet poslova i tri mašine

4 Predlog novog metoda ASPT

U radu (Ignjatović & Tartalja, 2022) se predstavlja nova konstruktivna heuristika polinomijalne složenosti, NEHTA, za kompetno rešavanje problema ASPT, koja se oslanja na način generisanja permutacija predložen u NEH algoritmu (Nawaz, Enscore Jr, & Ham, 1983), opisanom u odeljku 3.3, MINIMAX_BSC formulaciju problema (Brusco, Köhn, & Steinley, 2013) opisanu u odeljku 2.3 i dokazan postupak PS2 za sastavljanje proizvoljnog broja optimalnih testova iz samo 2 skupa stavki (Pereira & Vila, 2015), koji je opisan u odeljku 3.2.2.2. Postupak PS2 će biti primenjivan više puta, u svakom koraku NEHTA algoritma, za rešavanje problema ASPT sa proizvoljnim brojem skupova. Po formulaciji MINIMAX_BSC problema, pretpostavka je da svaki skup sadrži onoliko stavki koliko je testova, a da svaki test sadrži onoliko stavki koliko je skupova, tako da se svaka stavka iz jednog skupa raspoređuje u po jedan test, kao i da se po jedna stavka iz svakog skupa raspoređuje u jedan test.

Ako se problem sastavljanja paralelnih testova, formulisan kao MINIMAX_BSC problem, preslika na permutacioni problem određivanja redosleda poslova u protočnom izvršavanju na nizu mašina, PFSP, skupovi grupisaih stavki sa disjunktnim opsezima težinskih koeficijenata (u daljem tekstu: težine) predstavljaju poslove (stavka predstavlja deo posla), težinski koeficijent stavke w_i predstavlja vreme izvršavanja dela posla na mašini i, a testovi predstavljaju mašine. Treba uočiti da se u algoritmu NEHTA koristi samo osnovna ideja NEH algoritma, ali da je algoritam drugačiji, prilagođen činjenici da se u NEHTA algoritmu traži da maksimalna težina testa bude minimalna, jer se na taj način balansiraju težine testova.

Najpre se formiraju dve uređene liste skupova. U listi A će se formirati konačno uređenje skupova. Na kraju postupka, stavke se biraju za testove na način da se i-ta stavka svakog skupa dodeljuje i-tom testu, po redosledu navođenja skupova u listi A. Lista A inicijalno sadrži samo skup sa najvećom ukupnom težinom stavki. Lista B inicijalno sadrži preostalih S-1 skupova uređenih po nerastućoj ukupnoj vrednosti težinskih koeficijenata stavki u skupovima (težina skupa). Stavke u skupovima liste B su uređene po nerastućoj težini.

U prvom koraku se bira prvi skup iz liste B i prebacuje u listu A u kojoj već postoji skup najveće težine. Redosled ta dva skupa u listi A je proizvoljan. To je različito u odnosu na NEH algoritam gde se već u koraku sa dva posla u listi raspoređenih poslova tražio njihov povoljniji redosled. Zatim se uzima sledeći skup iz liste B i pronalazi njegova pozicija unutar liste A. Primenjujući ideju NEH algoritma da se uređenje skupova u listi A ne menja dodavanjem novog skupa, novi skup može da se umetne pre prvog skupa, između prva dva skupa, ili posle drugog skupa. Svako od ova tri moguća uređenja skupova u listi A predstavlja jedno parcijalno kandidat-rešenje sa 3 skupa. Za svako kandidat-rešenje sa 3 skupa (X, Y, Z), se od prva dva skupa X i Y, pri čemu je prvi X uređen po nerastućim težinama, a drugi Y po neopadajućim težinama stavki, formira jedan skup W od parova, čiji elementi imaju zbirnu težinu stavki na odgovarajućim pozicijama u skupovima X i Y. Zatim se primenjuje postupak za dva skupa PS2, i to nad skupom W i trećim skupom datog kandidat-rešenja Z. Skup W se uređuje nerastuće po težinama elemenata, a skup Z neopadajuće po težinama stavki. Sabiraju se težine elemenata na odgovarajućim pozicijama skupova W i Z i formiraju parcijalni testovi od stavki iz elemenata skupa W i stavki skupa Z na odgovarajućim pozicijama u skupovima. Za svaki skup kandidat-rešenja se pronalazi maksimalna težina parcijalnih testova. Bira se kandidat-rešenje sa najmanjom maksimalnom težinom testa. Postupak se nastavlja tako što se za svaki sledeći skup iz liste B traži pozicija unutar već uređenih skupova u listi A, tako da se njihovo relativno uređenje ne menja. Od svih mogućih pozicija novog skupa u listi A, ciljna pozicija se bira tako da uređenje skupova u listi A ima minimalnu maksimalnu vrednost težine

parcijalno formiranih testova. Postupak se ponavlja sve dok svi skupovi iz liste *B* ne nađu na svojim pozicijama u listi *A*.

Algoritam je objašnjen na primeru (Slika 4.1) predstavljenom u (Pereira & Vila, 2015), iako ovaj primer ne sledi pretpostavku o bliskim težinama stavki u jednom skupu. Numerisan red (oznake S1,...,S5) predstavlja skup. U svakoj ćeliji se nalazi težina stavke odgovarajućeg skupa. Poslednja, siva kolona predstavlja sumu težina stavki odgovarajućeg skupa. Sivi red predstavlja ukupne težine testova gde se *i*-ti test sastoji iz *i*-tih stavki (p_i) svakog skupa. Izdvojena ćelija sa desne strane predstavlja maksimalnu težinu testa formiranog od skupova u listi *A*, za koju će se koristiti naziv maksimalna težina uređenja (*Zmax*).

Inicijalno postoji pet skupova (S1, S2, S3, S4, S5) sa po tri stavke (p1, p2, p3) (a). Skupovi i stavke u skupovima su uređeni po nerastućoj težini (b). Lista *A* sadrži skup S5, a lista *B* preostale skupove S2-S3-S4-S1.

a

	p_1	p_2	p_3	
S1	2	4	3	9
S2	8	6	4	18
S3	1	7	9	17
S4	4	5	8	17
S5	7	6	6	19

b

S5	7	6	6	19
S2	8	6	4	18
S3	9	7	1	17
S4	8	5	4	17
S1	4	3	2	9

c

S5	7	6	6	
S2	4	6	8	
	11	12	14	**14**

d

S5,S2	14	12	11	**14**
S3	1	7	9	
	15	19	20	**20**

e

S5	7	6	6	
S3	1	7	9	
	8	13	15	**15**

e1

S5,S3	15	13	8	**15**
S2	4	6	8	
	19	19	16	**19**

f

S3	9	7	1	
S5	6	6	7	
	15	13	8	**15**

f1

S3,S5	15	13	8	**15**
S2	4	6	8	
	19	19	16	**19**

g

S5	7	6	6
S4	4	5	8
S3	9	7	1
S1	2	3	4
S2	4	6	8
	26	27	27

h

	T1	**T2**	**T3**
S1	p_1	p_3	p_2
S2	p_3	p_2	p_1
S3	p_3	p_2	p_1
S4	p_1	p_2	p_3
S5	p_1	p_2	p_3

Slika 4.1 - Primer sa pet setova i tri stavke po setu

Skup S2 iz liste *B* se prebacuje u listu *A*, pa se skupovi S5 i S2 uređuju prema postupku PS2 (c). Maksimalna težina uređenja S5-S2 (maksimalna težina među testovima sa po 2 stavke), *Zmax* = 14, je jednaka maksimalnoj težini uređenja S2-S5, pa se proizvoljno bira jedno, na primer uređenje S5-S2, jer je ovo početno uređenje proizvoljno. Lista *A* sadrži skupove S5-S2, a lista *B* sadrži skupove S3-S4-S1.

U sledećoj iteraciji se bira sledeći skup iz liste *B* (skup S3) i traži se njegova pozicija u listi *A*. Pošto se uzajamno uređenje S5-S2 neće menjati, nova moguća uređenja su S5-S2-S3, S5-S3-S2 i S3-S5-S2. Cilj je izabrati uređenje sa minimalnom maksimalnom težinom parcijalnih testova. Skupu parova stavki formiranom od skupova S5 i S2 se kao drugi skup dodaje S3 i problem se rešava po opisanom postupku PS2. Maksimalna težina uređenja S5-S2-S3 (d) je *Zmax* = 20. Uređenja S5-S3-S2 (e, e1) i S3-S5-S2 (f, f1) daju jednaku maksimalnu težinu uređenja *Zmax* = 19 koja je manja od *Zmax* za uređenje S5-S2-S3, tako da ona postaje minimalna vrednost iteracije *Zminmax* = 19. Bira se jedno od dva uređenja liste *A* koja daju isti rezultat, na primer S3-S5-S2, i

postupak se nastavlja na isti način. Konačno uređenje je prikazano u tablici (g), a raspodela stavki p_i po testovima T_j u tablici (h).

a

	p1	p2	p3	
S1	0.179400	0.199168	0.221169	0.599737
S2	0.328048	0.309978	0.289920	0.927946
S3	0.359104	0.369233	0.404647	1.132983
S4	0.396198	0.377146	0.429644	1.202988
S5	0.430313	0.411808	0.453803	1.295923
S6	0.420420	0.404420	0.417188	1.242028
S7	0.370688	0.370392	0.382537	1.123617
S8	0.366000	0.372000	0.314976	1.052976
S9	0.267894	0.215853	0.286528	0.770275
S10	0.374798	0.368148	0.418125	1.161071
S11	0.332495	0.346469	0.316796	0.995759
S12	0.215712	0.200025	0.219965	0.635702
	4.041069	3.944640	4.155296	

ZLB= 4.047001

b

S5	0.453803	0.430313	0.411808	1.295923
S6	0.420420	0.417188	0.404420	1.242028
S4	0.429644	0.396198	0.377146	1.202988
S10	0.418125	0.374798	0.368148	1.161071
S3	0.404647	0.369233	0.359104	1.132983
S7	0.382537	0.370688	0.370392	1.123617
S8	0.372000	0.366000	0.314976	1.052976
S11	0.346469	0.332495	0.316796	0.995759
S2	0.328048	0.309978	0.289920	0.927946
S9	0.286528	0.267894	0.215853	0.770275
S12	0.219965	0.215712	0.200025	0.635702
S1	0.221169	0.199168	0.179400	0.599737

c

S12	p2	p3	p1
S9	p2	p3	p1
S11	p2	p3	p1
S8	p2	p1	p3
S10	p1	p2	p3
S2	p1	p2	p3
S1	p1	p2	p3
S4	p3	p1	p2
S7	p2	p1	p3
S3	p1	p3	p2
S5	p1	p2	p3
S6	p1	p3	p2

Z= 4.047427 pct:0.010551

5,6

5	0.453803	0.430313	0.411808
6	0.404420	0.417188	0.420420
	0.858223	0.847500	0.832228

4,5,6

4	0.429644	0.396198	0.377146
5	0.411808	0.430313	0.453803
	0.841452	0.826511	0.830949

4,5	0.841452	0.830949	0.826511	
6	0.404420	0.417188	0.420420	
	1.245872	1.248136	1.246931	**1.248136**

5,4,6

5	0.453803	0.430313	0.411808
4	0.377146	0.396198	0.429644
	0.830949	0.826511	0.841452

5,4	0.841452	0.830949	0.826511	
6	0.404420	0.417188	0.420420	
	1.245872	1.248136	1.246931	1.248136

5,6,4

5,6	0.858223	0.8475	0.832228	
4	0.377146	0.396198	0.429644	
	1.235369	1.243698	1.261872	1.261872

10,4,5,6

10	0.418125	0.374798	0.368148
4	0.377146	0.396198	0.429644
	0.795271	0.770996	0.797792

10,4	0.797792	0.795271	0.770996	
5	0.411808	0.430313	0.453803	
	1.209600	1.225584	1.224799	

10,4,5	1.2255835	1.2247985	1.209600	
6	0.377146	0.396198	0.429644	
	1.602730	1.620997	1.639244	**1.636723**

4,10,5,6

4,10	0.797792	0.795271	0.770996
5	0.411808	0.430313	0.453803
	1.209600	1.225584	1.224799

4,10,5	1.225584	1.224799	1.2096	
6	0.404420	0.417188	0.420420	
	1.630004	1.641986	1.630020	1.641986

4,5,10,6

4,5	0.841452	0.830949	0.826511
10	0.368148	0.374798	0.418125
	1.209600	1.205747	1.244636

4,5,10	1.244636	1.209600	1.205747	
6	0.404420	0.417188	0.420420	
	1.649056	1.626788	1.626167	1.649056

4,5,6,10

4,5,6	1.248136	1.246931	1.245872	
10	0.368148	0.374798	0.418125	
	1.616284	1.621729	1.663997	1.663997

Slika 4.2 – Primer sa 12 skupova i 3 stavke po skupu

Drugi primer rada algoritma prikazuje Slika 4.2, gde je korišćena banka stavki koja je korišćena i u radu (Brusco, Köhn, & Steinley, 2013). Lista u tablici (b) predstavlja početnu, uređenu, listu skupova *B*. Prikazan je postupak za uređenje samo prva četiri skupa. Konačno uređenje je prikazano u tablici (c), gde su u ćelijama tablice dati indeksi stavki. Težina najtežeg testa jednaka je Z = 4.047427, gde je relativno odstupanje od optimalnog rešenja Z_{LB} = 4.047001 jednako 0.01%.

Sledeći pseudokod (Pseudokod 4.1, Pseudokod 4.2, Pseudokod 4.3) formalno opisuje predloženi postupak.

Pseudokod 4.1 opisuje predloženi metod u celini. On sadrži pozive četiri procedure/funkcije: *initialization*(*S*, *A*, *B*), *remove*(*j*, *B*), *insertion*(*j*, *s*, *A*) i *formation*(*A*). U proceduri *initialization*(*S, A,*

B), *S* skupova stavki se uređuje u nerastućem redosledu težina skupova. Unutar svakog skupa stavki, stavke se uređuju u nerastućem redosledu njihovih težina.

U listu *A* se umeće prvi skup stavki *B*(1), koji se uklanja iz liste *B*, dok u listi *B* ostaje *S*-1 preostalih skupova stavki. U petlji `repeat-until` se u svakoj iteraciji najpre inkrementira indeks *j* koji predstavlja indeks izabranog skupa iz liste *B*, a zatim se funkcijom *remove*(*j*, *B*) uklanja *j*-ti skup *s* iz liste *B* i procedurom *insertion*(*j*, *s*, *A*) pronalazi optimalna pozicija skupa *s,* počevši od pozicije *j,* unutar liste *A*, u koju se umeće skup stavki *s*, tako da se relativno uređenje prethodno uređenih skupova ne menja. Procedura *insertion*(*j*, *s*, *A*) je opisana u algoritmu 2. Na kraju, funkcija *formation*(*A*) kreira testove tako što svakom *i*-tom testu ($i = 1,...,T$) dodaje *i*-tu stavku iz svakog skupa stavki koji pripada listi *A* (pri čemu se u tom trenutku svi skupovi stavki nalaze u listi *A*).

Algoritam 1. Opis NEHTA algoritma

```
initialization(S, A, B)
j ← 1
repeat
    j ← j + 1
    s ← remove(j, B)
    insertion (j, s, A)
until j = |S|
tests ← formation(A)
return tests
```

Pseudokod 4.1 – NEHTA algoritam

Pseudokod 4.2 opisuje rad procedure *insertion*(*j*, *s*, *A*), koja umeće skup *s* na optimalnu poziciju (počevši od pozicije *j*) u listu *A*. Prvo se računa inicijalna maksimalna težina uređenja *zMax* koristeći funkciju *calcMax*(*l, s, A*) opisanu u Algoritmu 3 (Pseudokod 4.3). U ovom trenutku, *j*-ti skup stavki (*s*), koji je uklonjen iz liste *B*, je inicijalno smešten na poslednju poziciju u listi *A* (pozicija $l = j$) i pozicija *l* je inicijalno optimalna pozicija ($l_{opt} = l$). U svakoj iteraciji petlje, sve dok je *l* veće od 1, pozicija *l* se dekrementira, isti skup *s* se privremeno smešta u listu *A* na poziciju *l* i nova moguća maksimalna težina uređenja *zMax* se izračunava koristeći *calcMax*(*l, s, A*), a pozicija l_{opt} za koju se postiže minimalna maksimalna težina se pamti. Na kraju, skup *s* se smešta na poziciju l_{opt} u listu *A* (*insert*(*s*, l_{opt}, *A*)).

Algoritam 2: *insertion*(*j*, *s*, *A*)

```
l ← j
l_opt ← l
zMax ← calcMax( l, s, A )
while l > 1
    l ← l - 1
    zMaxIt ← calcMax(l, s, A)
    if zMaxIt < zMax then
        l_opt = l
        zMax ← zMaxIt
    end if
end while
insert(s, l_opt, A)
```

Pseudokod 4.2 – Opis procedure insertion (j, s, A)

Pseudokod 4.3 Pseudokod 4.3 opisuje rad funkcije *calcMax*(*l*, *s, A*) koja računa težinu trenutnog uređenja skupova. To je maksimalna težina testa za trenutno uređenje skupova u listi *A*, kada je skup *s* na poziciji *l*. Na početku Algoritma 3, skup *s* se privremeno umeće u listu *A* na poziciju *l*. Elementi vektora *sum*, koji sadrži težine parcijalnih testova (testova formiranih od stavki iz skupova zaključno sa tekućim skupom *A*(*i*)), se inicijalizuju na vrednost 0. U spoljnoj petlji se prolazi kroz sve skupove liste *A*. U unutrašnjoj petlji, težina stavke sa indeksom *t* tekućeg skupa *A*(*i*) (*A*(*i*).*stavke*(*t*).*tezina*) se dodaje svakoj težini parcijalnog testa (*sum*(*t*)). Vektor težina stavki tekućeg skupa liste *A*(*i*) je uređen u nerastućem redosledu težina (u proceduri *initialization*), a vektor težina parcijalnih testova (*sum*) se uređuje po neopadajućem redosledu težina primenom funkcije *sortAsc*(*sum*). Kao posledica, dva vektora sa težinama su uređena u međusobno suprotnom redosledu da bi mogla da se primeni procedura PS2. Kada je spoljna petlja završila sa radom, skup privremeno smešten na poziciju *l*, se uklanja. Rezultat postupka je maksimalna težina testa (*sum*(*T*)) kome su dodate sve stavke skupova iz liste *A*.

Algoritam 3: *calcMax*(*l*, *s, A*)

```
A(l) ← s
i ← 1
for each t (1<= t <= T) do sum ← 0 end for
repeat
    for each t (1<= t <= T) do
    sum(t) = sum(t) + A(i).stavke(t).tezina
    end for
    sortAsc(sum)
    i ← i + 1
 until i = |A|+1
 remove(l, A)
 return sum(T)
```

Pseudokod 4.3 – Opis funkcije calcMax(l, s, A)

U Algoritmu 1 se procedura *insertion*, u `repeat-until` petlji, poziva *S* puta. Iz procedure *insertion*, u repeat-until petlji Algoritma 2, poziva se funkcija *calcMax* 1+2+3+...+*S* puta. U funkciji *calcMax()* (Algoritam 3) se u (spoljašnjoj) `repeat-until` petlji, koja ima 1+2+3+...+*S* iteracija, izvrši instrukcija u `for-each` petlji *T* puta i niz *sum* dužine *T* se uređuje (smatra se da je složenost algoritma za uređivanje skupa jednak $T \cdot \log T$). Dakle, složenost NEHTA algoritma je data izrazom (*170*).

$$(T+T\cdot\log T) + 2^2(T+T\cdot\log T) + 3^2(T+T\cdot\log T) + ... + S^2(T+T\cdot\log T) = S^{\cdot}T(1+\log T)(S+1)(2\cdot S+1)/6 \qquad (170)$$

Proizilazi da je red složenosti NEHTA algoritma dat u izrazu (*171*). Takav algoritam je u klasi složenosti P, odnosno izvrava se u polinomijalnom vremenu.

$$O(S^3\cdot T\cdot\log T) \qquad (171)$$

5 Rezultati

U ovom poglavlju su predstavljeni i analizirani rezultati eksperimenata koji su imali za cilj procenu kvaliteta i efikasnosti novopredloženog metoda NEHTA. Najpre je opisana postavka eksperimenta, a zatim je izvršeno poređenje za metodima zasnovanim na metaheuristikama simuliranog kaljenja (Brusco, Köhn, & Steinley, 2013) i pretrage promenljivih okolina (Pereira & Vila, 2015).

5.1 Postavka eksperimenta

U ovoj disertaciji se slede preporuke poređenja metaheuristika iz (Silberholz & Golden, 2010). Prva od njih je testiranje na istom skupu podataka. Da bi rezultati bili poredivi sa rezultatima dobijenim u radovima (Brusco, Köhn, & Steinley, 2013) i (Pereira & Vila, 2015), korišćene su iste postavke eksperimenata. Najpre, stavke su generisane na način opisan u (Brusco, Köhn, & Steinley, 2013), koji je primenjen i u radu (Pereira & Vila, 2015).

U eksperimentima prikazanim u ovoj disertaciji, kao i u prethodno spomenutim radovima, težina i diskriminativnost stavke se računaju na sledeći način. Vrednost težine p_1 za prvu stavku u skupu se generiše nasumično po uniformnoj raspodeli u intervalu [0.3, 0.8]. Vrednost diskriminativnosti d_1 za prvu stavku u skupu se, takođe, generiše nasumično po uniformnoj raspodeli u intervalu [0.25, 0.60]. Da bi bila zadovoljena pretpostavka sličnih težinskih koeficijenata stavki (w_i) u skupu, težina p_i preostalih stavki se generiše slučajno po uniformnoj raspodeli u intervalu [p_1 - 0.1, p_1 + 0.1], a diskriminativnost d_i u intervalu [d_1 - 0.1, d_1 + 0.1]. Težinski koeficijent stavke se izračunava prema formuli (57).

Realne vrednosti izračunatih težinskih koeficijenata stavki su pomnožene konstantom (10^6) i zaokružene na najbližu celobrojnu vrednost kao i u (Pereira & Vila, 2015), da bi rezultati bili prikazani sa tačnošću od 6 cifara.

Kao mera kvaliteta testa Q se definiše procentualno odstupanje od donje granice težine testa koja bi se ostvarila idealnim balansiranjem težina stavki po testovima. Donja granica težine testa (Z_{LB}) je jednaka prosečnoj težini testa u optimalnom rešenju, koja je data formulom (172).

$$Z_{LB} = \frac{W}{T} \tag{172}$$

gde je $W = \sum_{i=1}^{T} \sum_{s=1}^{S} w_{is}$ suma težina svih stavki u svim skupovima. T je ukupan broj stavki po skupu, odnosno broj testova, a S je broj skupova, odnosno ukupan broj stavki u testu.

Vreme izvršavanja algoritma je uvek važan ali i težak osnov za poređenje (Silberholz & Golden, 2010). Najbolje bi bilo kada bi mogao da se dobije izvorišni kod algoritma, da se izvrši na istom računaru i sa istim prevodiocem. Čak i da je to moguće, javlja se sledeći problem. Algoritmi mogu da budu programirani na različitim jezicima. Postoje pokušaji da se definišu multiplikativni faktori kojima bi se te razlike definisale (Bull, Smith, Pottage, & Freeman, 2001), ali se razlike javljaju i kod načina optimizacije različitih prevodilaca. Na primer, prevodilac *gcc* za jezik C ima preko 100 optimizacionih mogućnosti, tako da bi bilo potrebno da se poznaju i parametri prevođenja. Zato je ovaj način nedovoljno precizan.

Kada ne može da se dobije izvorišni kod, moguće je pokušati implementirati algoritam na istom jeziku na kome se implementirao drugi algoritam za poređenje. Taj način poređenja isto ima svoje

mane, zato što možda nisu svi detalji implementacije objašnjeni, pa je teško precizno implementirati originalni algoritam.

Treći način poređenja jeste da se koriste objavljeni rezultati, da se analizira hardver na kome se izvršavao algoritam i da se razlika u hardveru neutralizuje korišćenjem multiplikativnog faktora (PassMark® Software Pty Ltd).

U ovoj disertaciji su se koristili drugi i treći opisan način poređenja brzine rada dva algoritma.

Predloženi NEHTA algoritam je implementiran u jeziku C# i izvršavan na računaru sa 1.9GHz-a Intel Core i7 procesorom i 16 GB RAM-a, pod operativnim sistemom *Microsoft Windows*.

5.2 Poređenje NEHTA algoritma sa SA algoritmom

U ovom odeljku su opisani eksperimenti poređenja vremena izvršavanja i kvaliteta rešenja između metoda zasnovanog na metaheuristici SA i NEHTA metoda.

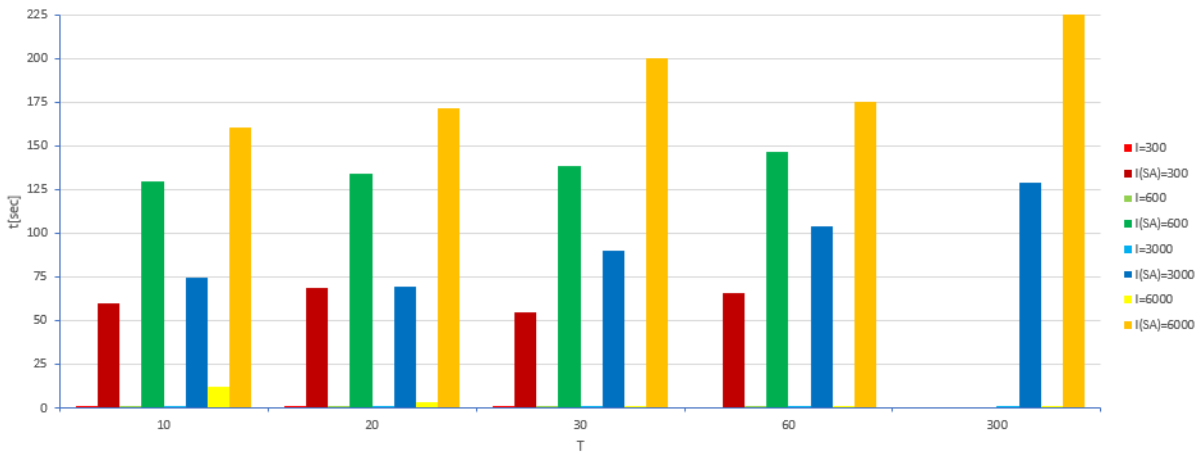
U Tabela 5.1 je dato poređenje mere kvaliteta (Q) i vremena izvršavanja (t) NEHTA algoritma sa odgovarajućim merama dobijenim primenom SA algoritma u radu (Brusco, Köhn, & Steinley, 2013). Originalno, SA algoritam je implementiran u jeziku Fortran 90, na računaru sa Pentium IV procesorom na 2.2GHz i sa 1 GB RAM-a. Poređenje vremena izvršavanja je izvršeno skaliranjem objavljenih vremena izvršavanja SA algoritma. Primenjen je multiplikativni faktor 0.25 na vreme izvršavanja SA algoritma. Multiplikativni faktor je procenjen na osnovu ocene performansi hardvera na kojima su se izvršavali NEHTA i SA algoritmi, objavljene u *CPU Single Thread Rating* (CPU Benchmarks). Poređenje je izvršeno na eksperimentalnim postavkama sa istim vrednostima parametara I i T iz rada (Brusco, Köhn, & Steinley, 2013). Iako se ovakvo poređenje uz skaliranje vremena izvršavanja ne može smatrati sasvim preciznim (Silberholz & Golden, 2010), uočljivo je značajno brže izvršavanje NEHTA algoritma na račun male degradacije kvaliteta.

Na osnovu eksperimentalnih rezultata, čiji je grafički prikaz dat na Slika 5.1, može se zaključiti da je vreme izvršavanja NEHTA algoritma višestruko manje u odnosu na vreme izvršavanja SA algoritma, za sve korišćene veličine banke stavki u rasponu od $I = 300$ do $I = 6000$ i za broj testova od $T = 10$ do $T = 300$. Jedini slučajevi za koje je SA algoritam brži u izvršavanju od NEHTA algoritma jeste za velike banke stavki $I = 6000$ i mali broj testova $T = 2$ ili $T = 3$, što je praktično bez značaja za testiranje u većini obrazovnih institucija. Kvalitet NEHTA algoritma, iako zaostaje za kvalitetom SA algoritma, u najgorem slučaju ($I = 300$, $S = 5$, $T = 60$) odstupa oko 0.55% od optimalnog rešenja. Za sve ostale kombinacije vrednosti (I, T), NEHTA algoritam daje rešenja koja odstupaju za manje od 0.17% od optimalnog rešenja. Primetiti da viši stubovi Q na grafiku (b) označavaju lošiji kvalitet.

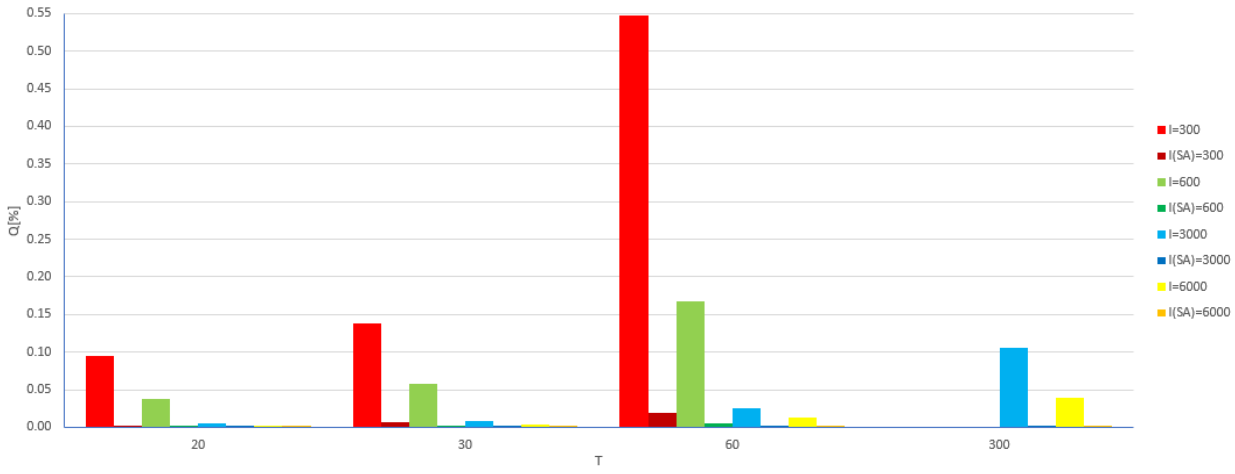
Sprovedena je i provera statističkog značaja rezultata eksperimenata primenom dvostranog *t-testa* za dva uzorka nad nizovima dobijenih vrednosti za zavisne promenljive t i Q, nad svim vrednostima nezavisnih promenljivih I i T. Analiza rezultata *t-testa* nad promenljivom t očekivano pokazuje da je razlika u vremenu izvršavanja NEHTA algoritma i SA algoritma statistički značajna (p-vrednost je $6.64 \cdot 10^{-8}$). Razlika u kvalitetu rešenja (Q), odnosno u relativnom odstupanju od optimalnog rešenja, dobijenom izvršavanjem jednog i drugog algoritma je takođe statistički značajna (p-vrednost je jednaka 0.035). Sa druge strane, relativno odstupanje od 0.55% od optimalnog rešenja, u najgorem slučaju, može da se smatra smanjenjem kvaliteta bez značaja za praktične primene.

Tabela 5.1 Poređenje NEHTA algoritma sa SA algoritmom

I	S	T	NEHTA t[sec]	NEHTA Q[%]	SA t[sec]	SA Q[%]
300	150	2	0.05	0.000000	65.00	0.000000
300	100	3	0.022	0.000013	61.50	0.000002
300	75	4	0.009	0.000182	54.00	0.000030
300	60	5	0.006	0.000261	56.25	0.000125
300	30	10	0.004	0.017315	60.25	0.000729
300	15	20	0.002	0.094268	68.75	0.002526
300	10	30	0.001	0.138768	55.00	0.006923
300	5	60	0.000	0.547832	66.00	0.019858
600	300	2	0.400	0.000000	111.50	0.000000
600	200	3	0.148	0.000014	123.50	0.000000
600	150	4	0.071	0.000033	143.25	0.000004
600	120	5	0.046	0.000231	115.25	0.000013
600	60	10	0.010	0.006292	129.50	0.000203
600	30	20	0.006	0.038215	134.25	0.000708
600	20	30	0.004	0.058304	138.25	0.001715
600	10	60	0.001	0.166758	146.75	0.004663
3000	1500	2	57.534	0.000000	67.50	0.000000
3000	1000	3	18.678	0.000000	77.00	0.000000
3000	750	4	8.912	0.000000	63.00	0.000001
3000	600	5	5.459	0.000018	76.50	0.000002
3000	300	10	1.282	0.000988	74.25	0.000014
3000	150	20	0.366	0.005406	69.75	0.000039
3000	100	30	0.173	0.009056	89.75	0.000051
3000	50	60	0.061	0.025533	103.75	0.000132
3000	10	300	0.006	0.105120	129.00	0.001377
6000	3000	2	488.000	0.000000	270.00	0.000000
6000	2000	3	182.469	0.000000	164.00	0.000000
6000	1500	4	84.621	0.000000	153.50	0.000000
6000	1200	5	48.342	0.000027	147.75	0.000001
6000	600	10	11.855	0.000537	160.75	0.000002
6000	300	20	3.042	0.001329	172.00	0.000008
6000	200	30	1.377	0.003446	200.25	0.000017
6000	100	60	0.399	0.013423	175.50	0.000033
6000	20	300	0.028	0.039086	225.00	0.000350



(a)



(b)

Slika 5.1 – Poređenje vremena (a) i kvaliteta izvršavanja (b) NEHTA algoritma sa SA algoritmom

5.3 Poređenje NEHTA algoritma sa VNS algoritmom

U ovom odeljku su opisani eksperimenti poređenja vremena izvršavanja i kvaliteta rešenja između metoda zasnovanog na metaheuristici VNS i NEHTA metoda.

Rezultati prikazani u radu (Pereira & Vila, 2015) pokazuju da VNS algoritam daje optimalne rezultate za većinu postavki I i T parametara, mereno izvršavanjem algoritma do isteka veoma dugačkog predviđenog intervala vremena od 600s. Važno je primetiti da iako je prosečno vreme da se dostigne optimalno rešenje po kombinaciji parametara I i T relativno kratko, realno vreme izvršavanja je veoma dugo (10 minuta). Pošto nije moguće znati vreme kada će najbolje rešenje biti dobijeno, vreme pretraživanja VNS algoritma pre nego što se prekine izvršavanje, u situacijama kada ne može da se pronađe optimalno rešenje, ne može da bude kratko.

Takođe, rezultati prikazani u radu (Pereira & Vila, 2015) pokazuju da je VNS algoritam sporiji od SA algoritma za eksperimentalne postavke gde je I do 10 puta veće od T. SA pokazuje bolje performanse od VNS algoritma za sve postavke sa velikim bankama stavki (I = 30000 i I = 60000). Za te postavke, SA algoritam je brži od VNS algoritma i do 20 puta.

Činjenica da VNS algoritam treba da se prekida posle određenog vremenskog intervala, dok se predloženi NEHTA algoritam može potpuno izvršiti u polinomijalnom vremenu, predstavljalo je motiv da se uporedi izvršavanje NEHTA i VNS algoritama prekidanjem VNS izvršenja u trenutku kada NEHTA algoritam pronađe rešenje (osim u slučajevima kada VNS algoritam pronađe optimalno rešenje pre isteka tog vremena), i da se tada uporede kvaliteti tako dobijenih rešenja. Vreme izvršavanja se kontrolisalo posle određenih blokova koda, pa je vreme izvršavanja VNS algoritma u nekim slučajevima, nešto malo duže od vremena izvršavanja NEHTA algoritma. VNS algoritam se izvršavao nad 10 eksperimentalnih postavki za svaku vrednost para parametara (I, T) prikazanu u Tabela 5.2, pa su u tabeli prikazane dobijene srednje vrednosti za Q i t. Postavke su bile iste i za VNS i za NEHTA algoritam.

Prikazani su rezultati implementirane VNS heuristike opisane u radu (Pereira & Vila, 2015), sa razlikom implementacije algoritma koji rešava slučaj sa dva testa. U citiranom radu je korišćena `expknap` implementacija opisana u (Pisinger, 1995). Umesto nje je korišćena implementacija opisana u (Neumenko). Nije primenjena *shaking* procedura opisana u sekciji 3.3. rada (Pereira & Vila, 2015) koja ne utiče na kvalitet rešenja kako je zaključeno u samom tom radu.

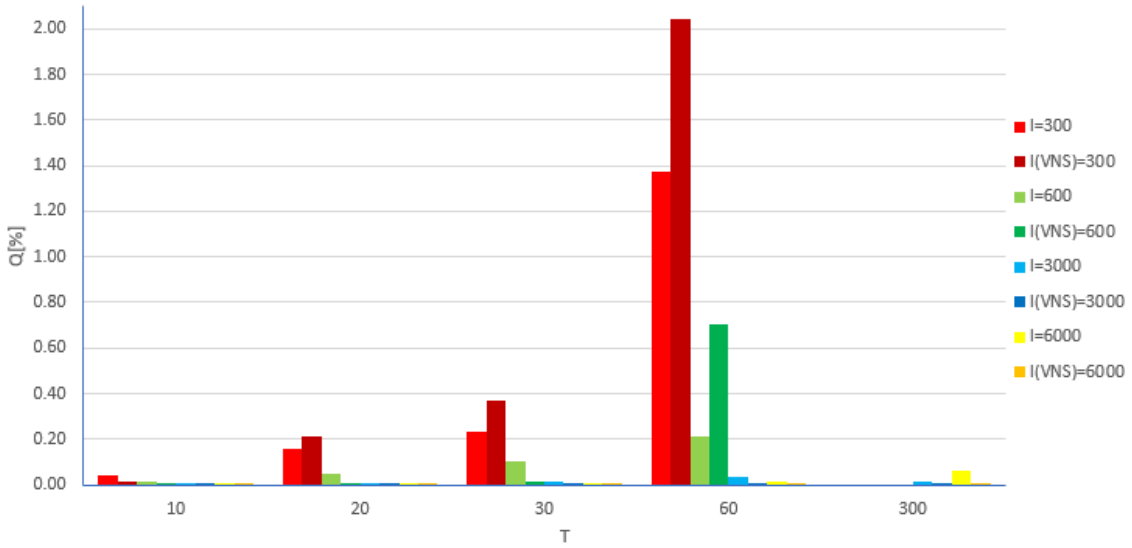
Tabela 5.2 prikazuje rezultate poređenja. VNS algoritam pronalazi optimalna rešenja u slučajevima malog broja testova (T = 2, 3, 4 i 5). Ovi slučajevi se mogu prepoznati u tabeli po značajno dužem vremenu izvršavanja NEHTA algoritma u poređenju sa VNS algoritmom. Međutim, ovako mali broj testova nema velikog značaja u obrazovnim institucijama.

Tabela 5.2 Poređenje NEHTA algoritma sa VNS algoritmim

I	S	T	NEHTA t[sec]	NEHTA Q[%]	VNS t[sec]	VNS Q[%]
300	150	2	0.103	0.000000	0.021	0.000000
300	100	3	0.082	0.000088	0.021	0.000000
300	75	4	0.050	0.001368	0.026	0.000000
300	60	5	0.035	0.002950	0.035	0.000000
300	30	10	0.014	0.043181	0.014	0.014343
300	15	20	0.003	0.156627	0.003	0.209935
300	10	30	0.001	0.230359	0.001	0.372119
300	5	60	0.002	1.372933	0.002	2.043660
600	300	2	1.411	0.000000	0.050	0.000000
600	200	3	0.630	0.000018	0.198	0.000000
600	150	4	0.367	0.000247	0.125	0.000000
600	120	5	0.254	0.001225	0.043	0.000000
600	60	10	0.107	0.012865	0.107	0.000025
600	30	20	0.019	0.050568	0.020	0.002954
600	20	30	0.009	0.100600	0.009	0.013292
600	10	60	0.004	0.214805	0.004	0.701125
3000	1500	2	57.076	0.000000	1.449	0.000000
3000	1000	3	19.174	0.000000	2.282	0.000000
3000	750	4	9.294	0.000014	2.054	0.000000
3000	600	5	5.574	0.000092	3.707	0.000000
3000	300	10	1.353	0.002252	1.368	0.000045
3000	150	20	1.042	0.007909	1.042	0.000068
3000	100	30	0.587	0.014842	0.588	0.000071
3000	50	60	0.290	0.035061	0.290	0.000486
3000	10	300	2.570	0.014551	2.570	0.000025
6000	3000	2	522.046	0.000000	12.005	0.000000
6000	2000	3	184.515	0.000003	8.547	0.000000
6000	1500	4	91.828	0.000009	8.313	0.000000
6000	1200	5	50.990	0.000071	9.636	0.000000
6000	600	10	10.677	0.000872	10.682	0.000002
6000	300	20	6.972	0.003503	7.007	0.000035
6000	200	30	3.343	0.006042	3.369	0.000047
6000	100	60	1.285	0.012820	1.288	0.000046
6000	20	300	0.167	0.064864	0.282	0.001773
30000	10	3000	0.247	0.128193	0.249	0.042192
30000	5	6000	0.201	1.446316	0.203	0.480882
60000	10	6000	0.542	0.101193	0.545	0.031430
60000	5	12000	0.481	2.299516	0.497	0.184439

Poređenje kvaliteta NEHTA i VNS algoritma (koji se izvršava u istom vremenskom intervalu u kom se izvršava i NEHTA algoritam) pokazuje da oba metoda daju vrlo slične kvalitete rešenja.

Razlika u kvalitetu rešenja, gde je NEHTA algoritam pokazao bolji kvalitet nego VNS algoritam (nižu vrednost Q) za isto vreme izvršavanja je vidljivo (Slika 5.2) za slučajeve $I = 300$ i $T = 20$, 30, 60 kao i za $I = 600$ i $T = 60$. U drugim slučajevima koji su od interesa ($T \geq 10$), iako je kvalitet NEHTA rešenja malo lošiji od kvaliteta VNS rešenja, vrednost Q je ispod 0.2% osim za slučajeve kada je veoma mali broj stavki u testu ($S = 5$).



Slika 5.2- Poređenje kvaliteta NEHTA i VNS algoritma

Provera statističkog značaja rezultata eksperimenta je sprovedena primenom jednostranog *t-test*-a za dva uzorka nad promenljivom Q. Dobijena p-vrednost, za $I = 300$ koja iznosi 0.058, je na granici koja potvrđuje statistički značaj rezultata eksperimenta. Na osnovu toga se može tvrditi da sa verovatnoćom od 94.2% važi zaključak o razlici u kvalitetu rešenja NEHTA i VNS algoritma izveden na osnovu rezultata eksperimenta. Granična vrednost za *p-vrednost* se može objasniti stohastičkom prirodom pretrage Okoline 1 i Okoline 2 u VNS algoritmu. Druge p-vrednosti, za $I = 600$, 3000, 6000 su iznad vrednosti 0.07, što ukazuje da je razlika u kvalitetu bez značaja.

Takođe je izvršen eksperiment gde su se NEHTA i VNS algoritam izvršavali nad istim eksperimentalnim postavkama, ali VNS algoritam je pušten da se izvršava do „graničnog“ vremena od 10 min (Tabela 5.3), ako ranije ne stigne do optimalnog rešenja. Algoritmi su se izvršavali nad istih 10 postavki za svaki par vrednosti parametara (I, T) u tabeli i prosečno vreme izvršavanja se izračunavalo. Prosečno vreme izvršavanja NEHTA algoritma za sve postavke je bilo 25.6 s, dok je za VNS to vreme bilo 192s.

Tabela 5.3 Poređenje NEHTA algoritma sa VNS algoritmom kada se VNS algoritam ne prekida

I	S	T	NEHTA t[sec]	NEHTA Q[%]	VNS t[sec]	VNS Q[%]
300	150	2	0.103	0.000000	0.021	0.000000
300	100	3	0.082	0.000088	0.021	0.000000
300	75	4	0.050	0.001368	0.026	0.000000
300	60	5	0.035	0.002950	0.035	0.000000
300	30	10	0.014	0.043181	0.288	0.000000
300	15	20	0.003	0.156627	480.065	0.000441
300	10	30	0.001	0.230359	600.000	0.013116
300	5	60	0.002	1.3729328	600.000	0.052859
600	300	2	1.411	0.000000	0.050	0.000000
600	200	3	0.630	0.000018	0.198	0.000000
600	150	4	0.367	0.000247	0.125	0.000000
600	120	5	0.254	0.001225	0.043	0.000000
600	60	10	0.107	0.012865	0.130	0.000000
600	30	20	0.019	0.050568	120.770	0.000070
600	20	30	0.009	0.100600	481.931	0.000092
600	10	60	0.004	0.214805	600.000	0.010822
3000	1500	2	57.076	0.000000	1.449	0.000000
3000	1000	3	19.174	0.000000	2.282	0.000000
3000	750	4	9.294	0.000014	2.054	0.000000
3000	600	5	5.574	0.000092	3.707	0.000000
3000	300	10	1.353	0.002252	10.828	0.000000
3000	150	20	1.042	0.007909	138.289	0.000003
3000	100	30	0.587	0.014842	61.160	0.000000
3000	50	60	0.290	0.035061	243.868	0.000019
3000	10	300	2.570	0.014551	600.002	0.000004
6000	3000	2	522.046	0.000000	12.005	0.000000
6000	2000	3	184.515	0.000003	8.547	0.000000
6000	1500	4	91.828	0.000009	8.313	0.000000
6000	1200	5	50.990	0.0000715	9.636	0.000000
6000	600	10	10.677	0.000872	15.928	0.000000
6000	300	20	6.972	0.003503	86.405	0.000000
6000	200	30	3.343	0.006042	200.750	0.000000
6000	100	60	1.285	0.012820	6.199	0.000000
6000	20	300	0.167	0.064864	600.007	0.000291
30000	10	3000	0.247	0.128193	600.003	0.002204
30000	5	6000	0.2012	1.4463162	600.005	0.002028
60000	10	6000	0.5425	0.1011935	600.006	0.000747
60000	5	12000	0.48125	2.299516	600.009	0.001659

5.4 Paralelni NEHTA algoritam (NEHTA*)

Ostvarena ušteda u vremenu izvršavanja u odnosu na SA i VNS algoritam daje mogućnost da se proširi prostor pretraživanja. U NEH heuristici poslovi sa najdužim vremenom izvršavanja imaju prioritet u odnosu na poslove sa kraćim vremenom izvršavanja. Po analogiji, u NEHTA slučaju kreće se od skupova sa najvećom težinom. Ta činjenica može da se iskoristi na sledeći način. Za neki, relativno mali, prethodno definisan broj skupova velike težine mogu da se ispitaju sve permutacije tih skupova i oni međusobno urede. Preostali skupovi bi se uključivali u listu raspoređenih skupova po prethodno opisanoj NEHTA heuristici. Takva modifikacija algoritma će biti nazvana NEHTA*. Algoritam NEHTA* može i da se paralelizuje, te da se paralelno izvrše koraci kojim se ispituju pojedine permutacije rasporeda relativno malog broja skupa najveće težine i na kraju izabere najbolji.

U Tabela 5.4 su prikazani rezultati poređenja NEHTA i NEHTA* algoritama. U ovom eksperimentu NEHTA* algoritam samo za prvih 6 skupova ispituje sve permutacije. Prikazani rezultati su dobijeni sekvencijalnim izvršavanjem, sa ciljem da se samo ukaže na moguće podizanje kvaliteta rešenja na ovaj način.

Može se uočiti da je za slučajeve od interesa ($T > 5$) kvalitet NEHTA* algoritma nešto bolji od kvaliteta NEHTA algoritma. Međutim dobijeno poboljšanje sa 6 početnih skupova ukazuje na mogućnost da se generisanjem svih permutacija nešto većeg broja skupova najveće težine može dobiti značajnije poboljšanje kvaliteta. Takvo istraživanje je van okvira ove diseracije, ali predstavlja jedan od mogućih pravaca daljeg istraživanja.

Tabela 5.4 Poređenje NEHTA algoritma sa NEHTA algoritmom*

I	S	T	NEHTA t[sec]	NEHTA Q[%]	NEHTA* t[sec]	NEHTA* Q[%]
300	150	2	0.050	0.000000	0.197	0.000000
300	100	3	0.022	0.000013	9.797	0.000000
300	75	4	0.009	0.000182	6.665	0.000016
300	60	5	0.006	0.000261	3.741	0.000079
300	30	10	0.004	0.017315	0.938	0.007360
300	15	20	0.002	0.094268	0.250	0.081976
300	10	30	0.001	0.138768	0.102	0.110788
300	5	60	0.000	0.592088	0.001	0.552185
600	300	2	0.400	0.000000	0.395	0.000000
600	200	3	0.148	0.000014	13.821	0.000000
600	150	4	0.071	0.000033	8.319	0.000010
600	120	5	0.046	0.000231	4.972	0.000016
600	60	10	0.010	0.006292	1.174	0.005043
600	30	20	0.006	0.038215	0.343	0.022282
600	20	30	0.004	0.058304	0.168	0.045774
600	10	60	0.001	0.166758	0.045	0.134388
3000	1500	2	56.944	0.000000	57.046	0.000000
3000	1000	3	18.678	0.000000	85.551	0.000000
3000	750	4	8.912	0.000000	9.181	0.000000
3000	600	5	5.459	0.000018	151.301	0.000001
3000	300	10	1.282	0.000988	34.353	0.000669
3000	150	20	0.366	0.005406	8.700	0.004189
3000	100	30	0.173	0.009056	3.889	0.007416
3000	50	60	0.061	0.025533	1.144	0.023397
3000	10	300	0.006	0.105120	0.060	0.093861
6000	3000	2	488.000	0.000000	n.a.	n.a.
6000	2000	3	182.469	0.000000	169.564	0.000000
6000	1500	4	88.765	0.000000	89.809	0.000000
6000	1200	5	50.983	0.000034	318.301	0.000001
6000	600	10	10.287	0.000523	74.141	0.000155
6000	300	20	3.058	0.002564	19.180	0.001386
6000	200	30	1.346	0.005400	7.993	0.003803
6000	100	60	0.486	0.011510	2.309	0.009269
6000	20	300	0.022	0.087480	0.110	0.066257

6 Zaključak

U okviru ove disertacije je predložen i analiziran metod NEHTA, koji potpuno rešava problem ASPT pomoću isključivo konstruktivne heuristike. Problem je formulisan kao jednodimenzionalno pakovanje objekata u korpe. Metod koristi ideju NEH konstruktivne heuristike za smanjenje broja permutacija, kao i proceduru PS2, koju su predložili Pereira i Vilá u radu koji rešava istu formulaciju problema, ali pomoću VNS heuristike. Procedurom PS2 se dobija optimalno rešenje za slučaj samo dva skupa stavki iz kojih se one biraju za proizvoljan broj paralelnih testova. Osnovni rezultat istraživanja sprovedenog u okvirima ove disertacije jeste uspešna adaptacija efikasne konstruktivne heuristike za potpuno rešavanje problema ASPT. Konstruktivne heuristike se uglavnom koriste za dobijanje samo početnih rešenja za heuristike poboljšanja. Kod konstruktivne heuristike vreme izvršavanja se može predvideti, čime se daje značajna prednost u odnosu na metode koji se zasnivaju na heuristikama poboljšanja. Prikazani eksperimentalni rezultati ukazuju na mogućnost da predloženi metod zameni heuristike poboljšanja u slučajevima kada se zahteva da se za što kraće vreme dobiju prihvatljivi rezultati.

Urađeno je detaljno poređenje vremena izvršavanja i kvaliteta rešenja NEHTA algoritma sa heuristikama poboljšanja (SA i VNS), kojima se rešava problem formulisan na isti način, sa istim skupom ograničenja i istom ciljnom funkcijom. To su i jedini autoru poznati algoritmi iz otvorene literature koji rešavaju istu formulaciju problema, korišćenu u ovoj disertaciji. Koristeći realistične eksperimentalne postavke, utvrđeno je da je vreme izvršavanja NEHTA algoritma je nekoliko redova veličine manje, u poređenju sa SA algoritmom, za kvalitet koji je lošiji za manje od 0.6%. Kod poređenja sa VNS algoritmom, pokazano je da za isto vreme izvršavanja, NEHTA algoritam daje bolji kvalitet rešenja (dva puta manju relativno odstupanje od idealnog rešenja Q) za slučajeve umerenog broja paralelnih testova (T = 20, 30 i 60) i relativno malog skupa raspoloživih stavki (I = 300). Ti slučajevi mogu biti od interesa u obrazovnim institucijama, od osnovnog do visokog obrazovanja. Ukoliko vreme izvršavanja VNS algoritma ne bi bilo ograničeno na vreme izvršavanja NEHTA algoritma, već ograničeno na 10 minuta (kao što je ograničeno u originalnom radu), VNS algoritam bi pokazao bolji kvalitet, ali uz neuporedivo veće angažovanje računarskih resursa. Kao zaključak, opisani NEHTA algoritam, koji je jednostavniji za implementaciju od poznatih algoritama koji rešavaju istu formulaciju problema, može da se koristi uspešnije od tih algoritama u slučajevima kada je značajno imati kratko, unapred procenjivo, vreme izvršavanja algoritma, dok se dobija zanemarljivo umanjen kvalitet rešenja za umeren broj paralelnih testova i banke sa relativno malim brojem stavki. To je čest slučaj u mnogim obrazovnim institucijama koje pate od nedostatka dovoljno moćnih računarskih resursa.

Kratko vreme izvršavanja NEHTA algoritma daje mogućnost da se proširi prostor pretrage čime bi se poboljšao kvalitet rešenja. Takođe, NEHTA algoritam može da se koristi kao početno rešenje za razne heuristike poboljšanja, čime bi se njihove performanse (brzina ili kvalitet) mogle poboljšati kvalitetnim početnim rešenjem. Dobro početno rešenje može da poboljša kvalitet konačnog rešenja, zadržavajući isto vreme izvršavanja, ili je moguće naći rešenje istog kvaliteta za kraće vreme. Planovi za dalja istraživanja su jednim delom vezani za ove mogućnosti unapređenja i primene NEHTA algoritma.

Značajan deo disertacije se odnosi na pregled i sistematizaciju literature o metodima za ASPT. Dat je strukturirani pregled odabranih pristupa u oblasti sastavljanja paralelnih testova u poslednjih nekoliko desetina godina, sa fokusom na pristupe koji koriste heuristička rešenja. U tom periodu su se pojavile različite formulacije problema ASPT i heuristika za njihovo rešavanja. Formulacije su prikazane na jedinstven način matematičkim formulama i klasifikovane na osnovu cilja optimizacije. Zatim su se heuristike koje su se koristile za rešavanje ASPT problema analizirale i

klasifikovale. Prikazane su njihove osnovne karakteristike. Kao i formulacije, tako su i heuristike formulisane koristeći jedinstvenu terminologiju.

Na osnovu odvojenih klasifikacija formulacija i heuristika se došlo do sledećih zaključaka. Postoje različite formulacije problema APTA koje imaju različite optimizacione ciljeve i različita ograničenja, kojima bitno utiču na način rešavanja problema. Heuristike koje su opisane u disertaciji mogu da nađu veoma dobra rešenja, često bliska optimalnim, za određene formulacije problema. Pregled literature ukazuje na mogućnost uspešnog rešavanja problema koristeći heuristike umesto uopštenih solvera za rešavanje problema linearnog programiranja. Algoritam zasnovan na VNS heuristici predstavljen u radu Pereira i Vilá može da posluži kao primer.

Unakrsna klasifikacija formulacija problema ASPT i heurističkih rešenja, predstavljena u ovoj disertaciji, može da inspiriše istraživače da eksperimentišu različitim ukrštanjima između opisanih formulacija problema i heuristika za rešavanje, ili da primene heuristiku koja se do sad nije koristila za neku od postojećih formulacija problema ASPT, kao što je upravo urađeno u ovoj disertaciji, primenom algoritma NEH na formulaciju jednodimenzionalnog pakovanja objekata u korpe.

Jedan od problema koji je tokom istraživanja jasno uočen, jeste različitost eksperimentalnih postavki pri evaluaciji različitih pristupa, korišćenih u odgovarajućim radovima. Komplikovano je uporediti performanse različitih heuristika bez zajedničkih referentnih banki stavki i drugih elemenata eksperimentalnih postavki. To upravo navodi na još jednu ideju za dalja istraživanja, vezanu za razvoj skupa referentnih banki i ekserimentalnih postavki, koje bi omogućile fer poređenje efikasnosti i kvaliteta raznovrsnih heuristika za rešavanje problema ASPT.

Posebna avenija daljih istraživanja vezana je za paralelizaciju algoritma NEHTA*. Ona se račva u dva pravca: jedan je vezan za konvencionalni paralelizam MIMD (eng. *Multiple Instruction Multiple Data*) tipa, a drugi za paralelizam SIMD (eng. *Single Instruction Multiple Data*) tipa, kakav se sreće kod savremenih grafičkih procesora.

Literatura

1. Ackerman, T. A. (1989). An alternative methodology for creating parallel test forms using IRT information function. *Proc. Annu. Meeting of the National Council on Measurement in Education.*
2. Adema, J. J. (1988). *A note on solving large-scale zero-one programming problems.* University of Twente, Department of Education.
3. Adema, J. J. (1992). Methods and models for construction of weakly parallel tests. *Appl. Psychol. Meas., 16*(1), 53-63.
4. Adema, J. J., & van der Linden, W. J. (1989). Algorithms for computerized test construction using classical item parameters. *J. Educ. Behav. Stat., 14*(3), 279-290.
5. Adema, J. J., Boekkooi-Timminga, E., & van der Linden, W. J. (1991). Achievement test construction using 0-1 linear programming. *Eur. J. Oper. Res., 55*(1), 103-111.
6. Ahuja, R. K., Magnanti, T., & Orlin, J. (1993). *Network Flows: Theory, Alorirhms, and Applications.* Pearson Education.
7. Allen, M. J., & Yen, W. M. (1979). *Introduction to measurement theory.* Long Grove: Waveland Press, Inc.
8. Armstrong, R. D., Douglas, J. H., & Wang, Z. (1994). Automated parallel test construction using classical test theory. *J. Educ. Behav. Stat., 19*(1), 73-90.
9. Armstrong, R. D., Jones, D. H., & Wu, I. (1992). An automated test development of parallel tests from a seed test. *Psychometrika, 57*(2), 271-288.
10. Belov, D. I. (2008). Uniform test assembly. *Psychometrika, 73*(1), 21-38.
11. Belov, D. I., & Armstrong, R. D. (2005). Monte Carlo test assembly for item pool analysis and extension. *Appl. Psych. Meas., 29*(4), 239-261.
12. Belov, D. I., & Armstrong, R. D. (2006). A constraint programming approach to extract the maximum number of non-overlaping test forms. *Comput. Optim. Appl., 33*(2), 319-332.
13. Blum, C., & Roli, A. (2003, September). Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Comp. Surv., 35*(3), 268-308.
14. Boekkooi-Timminga, E. (1987). Simultaneous test construction by zero-one programming. *Methodika, 1*, 101-112.
15. Boekkooi-Timminga, E. (1990). The construction of parallel tests from IRT-based item banks. *J. Educ. Behav. Stat., 15*(2), 129-145.
16. Bojić, D. M., Bošnjaković, A. M., Protić, J. Ž., & Tartalja, I. I. (2016). A Modified Hill-Climbing Algorithm for Knowledge Test Assembly Based on Classified Criteria. *Int. J. Soft. Eng. Know., 26*(6), 953-980.
17. Bomze, I. M., Budinich, M., Pardalos, P. M., & Paleilo, M. (1999). The maximum clique problem. In *Handbook of combinatorial optimization* (pp. 1-74). Boston: Springer.
18. Brusco, M. J., Köhn, H. F., & Steinley, D. (2013, July). Exact and approximate methods for a one-dimensional minimax bin-packing problem. *Ann. Oper. Res., 206*(1), 611-626.

19. Bull, M., Smith, L., Pottage, L., & Freeman, R. (2001). Benchmarking Java against C and Fortran for scientific applications. *ACM 2001 Java Grande/ISCOPE conference* (pp. 97-105). New York: ACM.

20. Chang, T., & Shiu, Y. (2012). Simultaneously construct IRT-based parallel tests based on an adapted CLONALG algorithm. *Appl. Intell., 36*, 979-994.

21. Chen, P. (2015). A sampling and classification item selection approach with content-balancing. *Behav. Res. Meth., 47*(1), 98-106.

22. Chen, P. H., Chang, H. H., & Wu, H. (2012). Item selection for the development of parallel forms from and IRT-based seed test using a sampling and classification approach. *Educ. Psychol. Meas., 72*, 933-953.

23. Coffman Jr, E. G., Lueker, G. S., & Rinnooy Kan, A. H. (1988). Asymptotic methods in the probabilistic analysis of sequencing nad packing heuristics. *Manage. Sci., 34*(3), 266-290.

24. Cook, R. (1971). The complexity of theorem-proving procedures. *Proc. 3rd Ann. ACM Symp. on Theory of Computing* (pp. 151-158). New York: Association for Computing Machinery.

25. Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika, 16*(3), 297-334.

26. Cvetković, D., Čangalović, M., Dugošija, Đ., Kovačević-Vujčić, V., Simić, S., & Vuleta, J. (1996). *Kombinatorna optimizacija: Matematička teorija i algoritmi.* Društvo operacionih istraživača Jugoslavije.

27. Danilović, M., & Ilić, O. (2015). A generalized constructive algorithm using insertion-based heuristics. *Comput. Oper. Res., 66*, 29-43.

28. Dantzig, G. B. (1951). Maximization of a Linear Function of Variables Subject to Linear Inequalities. In T. C. Koopmans, *Activity Analysis of Production and Allocation* (pp. 339-347). New York: John Wiley & Sons.

29. de Castro, L. N., & Von Zuben, F. J. (2002). Learning and optimiyzation using the colanal selection principle. *IEEE Trans. Evol. Comput., 6*, 239-251.

30. Dijkstra, E. W. (1959). A note on two problems in connection with graphs. *Numer. Math., 1*(1), 269-271.

31. Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., & Cosar, A. (2019). A survey on new generation metaheuristic algorithms. *Comp. Ind. Eng., 137*, 106040.

32. Dyckhoff, H. (1990). A typology of cutting and packing problems. *Eur. J. Oper. Res., 44*(2), 145-159.

33. Eberhart, R., & Kennedy, J. (1995). Particle swarm optimization. *Proceedings of the IEEE international conference on neural networks. 4*, pp. 1942-1948. Citeseer.

34. Fan, X. (1998). Item response theory and classical test theory: An empirical comparison of their item/person statistics. *Educ. Psychol. Meas., 58*(3), 357-381.

35. Fisher, R. A. (1922). On the mathematical foundations of theoretical statistics. *Proceedings of the Royal Society.*

36. Fulkerson, D. R. (1961). An Out-of-Kilter method for Minimal-Cost Flow Problems. *J. Soc. Ind. Appl. Math., 9*(1), 18-27.

37. Garey, M. R., Graham, R. L., Johnson, D. S., & Yao, A.-C. (1976). Resource constrained scheduling as generalized bin packing. *J. Comb. Theory, 21*(3), 257-298.

38. Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Math. Oper. Res., 1*(2), 117-129.

39. Garey, M., & Johnson, S. D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco, United States: W. H. Freeman and Company.

40. Gendreau, M., & Potvin, J.-Y. (2010). *Handbook of metaheuristics.* New York: Springer.

41. Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res., 13*(5), 533-549.

42. Glover, F. (1989). Tabu search - part I. *ORSA J. Comp., 1*(3), 190-206.

43. Gulliksen, H. (1950). *Theory of Mental Tests.* New York: Wiley.

44. Hambleton, R. K., & Swaminathan, H. (1984). *Item Response Theory: Principles and Applications.* Boston: Springer.

45. Harris, T. E., & Ross, F. S. (1955). *Fundamentals of a Method for Evaluating Rail Net Capacities.* Rand Corp Santa Monica.

46. Ho, T. F., Yin, P. Y., Hwang, G. J., Shyu, S. J., & Yean, Y. N. (2009). Multi-objective parallel test-sheet composition using enhanced particle swarm optimization. *Educ. Technol. Soc., 12*, 193-206.

47. Holland, J. H. (1973). Genetic algorithms and the optimal allocation of trials. *SIAM J. Comput., 2*, 88-105.

48. Hoos, H. H., & Stutzle, T. (2005). *Stochastic Local Search.* Morgan Kaufmann.

49. Hussain, K., Salleh, M. N., Cheng, S., & Shi, Y. (2019). Metaheuristic research: a comprehensive survey. *Artif. Intell. Rev., 52*(4), 2191-2233.

50. Hwang, G. J., Chu, H. C., Yin, P. Y., & Lin, J. Y. (2008). An innovative parallel test sheet composition approach to meet multiple assessment criteria for national tests. *Comput. Educ., 51*(3), 1058-1072.

51. IBM. (2009). *IBM ILOG CPLEX12.1.* Retrieved from ftp://public.dhe.ibm.com/software/websphere/ilog/docs/optimization/cplex/ps_usrmancplex.pdf.

52. Idowu, E. O., Eluwa, A. N., & Abang, B. K. (2011). Evaluation of Mathematics Achievement Test: A Comparison Between Classical Test Theory (CTT) and Item Response Theory (IRT). *J. Educ. Soc. Res., 1*(4).

53. Ignjatović, M. M., & Tartalja, I. I. (2022). A Constructive Heuristic for Automated Parallel Tests Assembly. *Int. J. Softw. Eng. Know., 32*(3), 1-21.

54. Ignjatović, M. M., Bojić, D. M., & Tartalja, I. I. (2021). A Survey on Problem Formulations and (Meta) Heuristic-Based Solutions in Automated Assembly of Parallel Test Forms. *Int. J. Softw. Eng. Know., 31*(8), 1171-1212.

55. Ignjatović, M., Bojić, D., Furlan, B., & Tartalja, I. (2015). Potential of knowledge discovery in automated test assembly. *Telfor J., 7*, 108-112.

56. Ishii, T., Songmuang, P., & Ueno, M. (2014). Maximum clique algorithm and its approximation for uniform test form assembly. *IEEE Trans. Learn. Technol., 7*(1), 83-95.

57. Karp, R. M. (1972). Reducibility among combinatorial problems. *Complexity of computer computations* (pp. 85-103). Boston: Springer.

58. Kennedy, J., & Eberhard, C. (1995). Particle swarm optimization. *Proc. ICNN'95:Int. Conf. Neural Networks* (pp. 1942-1948). IEEE.

59. Kennington, J., & Wang, Z. (1992). Shortest augmenting path algorithm for the semi-assignment problems. *Oper. Res., 40*(1), 178-187.

60. Khachaturyan, A., Semenovskaya, S., & Vainstein, B. (1979). Statistical-thermodynamic approach to determination of structure amplitude phases. *Sov. Phys. Crystallogr., 24*(5), 519-524.

61. Kirkpatrick, S., Gelatt, D. C., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*(4598), 671-680.

62. Krishnan, V. (2013). *The early child development instrument (EDI): An item analysis using classical test theory (CTT) on Alberta's data.* Faculty of Extension, University of Alberta, Edmonton.

63. Kuder, F. G., & Richardson, M. W. (1937). The theory of the estimation of test reliability. *Psychometrika, 2.3*, 151-160.

64. Land, A. H., & Doig, A. G. (1960). An automatic method for solving discrete programming problems. *Econometrika, 28*, 497-520.

65. Lawley, D. N. (1943). On problems connected with item selection and test construction. *Proceedings of the Royal Society of Edinburgh Section A: Mathematics*, *61*, pp. 273-287.

66. Lin, C. J. (2008). Comparison between classical test theory and item response theory in automated assembly of parallel test forms. *JTLA, 6*(8). Retrieved from https://ejournals.bc.edu/index.php/jtla/article/download/1638/1473

67. Lin, Y., Jiang, Y., Gong, Y., Zhan, Y., & Zhang, Y. (2019). A discrete multiobjective particle swarm optimizer for automated assembly of parallel cognitive diagnosis tests. *IEEE T. Cybernetics., 49*, 2792-2805.

68. Lord, F. M. (1952). *Psychometric Monograph No. 7.* Psychometric Society.

69. Lord, F. M. (1980). *Applications of item response theory to practical testing problems.* New York, USA: Routledge.

70. Lord, F. M., & Novick, M. R. (1968). *Statistical theories of mental test scores.* Addison-Wesley.

71. Luecht, R. M. (1998). Computer-assisted test assembly using optimization heuristic. *Appl. Psychol. Meas., 22*(3), 224-236.

72. Luecht, R. M., & Sireci, S. G. (2011). *A Review of Models for Computer-Based Testing.* College Board.

73. Macdonald, P., & Paunonen, S. (2002). A Monte Carlo comparison of item and person statistics based on item response theory versus classical test theory. *Edu. Psych. Meas., 62*(6), 921-943.

74. Martello, S., & Toth, P. (1990). *Knapsack Problems: Algorithms and Computer Implementations.* John Wiley & Sons.

75. Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Comput. Oper. Res., 24*(11), 1097-1100.

76. Nakanishi, H., & Tomita, E. (2008). *An o (20.19171n)-time and polynomial-space algorithm for finding a maximum clique.* IPSJ SIG Tech. Rep.

77. Nawaz, M., Enscore Jr, E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 91-95.

78. Nemhauser, G. L., Wolsey, L. A., & Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions - I. *Math. Program., 14*(1), 265-294.

79. Neumenko, F. (n.d.). *Partition Problem Solution*. Retrieved 5 2020, from https://www.codeproject.com/Articles/1265125/Fast-and-Practically-perfect-Partition-Problem-Sol

80. Nguyen, M. M., Hui, S. C., & Fong, A. C. (2013). Large-scale multiobjective static test generation for web-based testing with integer programming. *IEEE T. Learn. Technol., 6*, 46-59.

81. Papadimitriou, C. H., & Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity.* New York: Dover Publications.

82. PassMark® Software Pty Ltd. (n.d.). *CPU Benchmarks*. Retrieved 5 2020, from https://www.cpubenchmark.net/compare/Intel-i7-8665U-vs-Intel-Pentium-4-2.20GHz/3434vs1065

83. Pereira, J., & Vila, M. (2015). Variable neighborhood search heuristics for a test assembly design problem. *Expert. Syst. Appl., 42*(10), 4805-4817.

84. Pham, D. T., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., & Zaidi, M. (2006). The Bees algorithm - A novel tool for complex optimization. *Proc. 2nd I*PROMS Virtual Int. Conf.: Intelligent Production Machines and Systems* (pp. 454-459). Elsevier Science.

85. Pisinger, D. (1995). An expanding-core algorithm for the exact 0-1 knapsack problem. *Eur. J. Oper. Res., 87*(1), 175-187.

86. Proietti, G. S., Matteucci, M., Mignani, S., & Veldkamp, B. P. (2020). Retrieved from http://amsacta.unibo.it/6401/1/paper_ccATA.pdf

87. Rossi, F. L., Nagano, M. S., & Neto, R. F. (2016). Evaluation of high performance constructive heuristics for the flow shop with makespan minimization. *Int. J. Adv. Manuf. Technol., 87*(1), 125-136.

88. Ruiz, R., & Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *Eur. J. Oper. Res., 165*(2), 479-494.

89. Samejima, F. (1977). Weakly parallel tests in latent trait theory with some critisisms of classical test theory. *Psychometrika, 42*, 193-198.

90. Schrijver, A. (2002). On the history of the transportation and maximum flow problems. *Math. Program., 91*(3), 437-445.

91. Shavelson, R. J., Webb, N. M., & Rowley, G. L. (1989). Generalizability theory. *Am. Psychol., 44*(16), 922-932.

92. Silberholz, J., & Golden, B. (2010). Comparison of Metaheuristics. In M. Gendreau, & J.-Y. Potvin, *Handbook of metaheuristics* (pp. 625-640). Boston: Springer.

93. Smits, N., & Finkelman, M. D. (2014). Variable length testing using the ordinal regression model. *Stat. Med., 33*(3), 488-499.

94. Songmuang, P., & Ueno, M. (2011). Bees algorithm for construction of multiple test forms in e-testing. *IEEE Trans. Learn. Technol., 4*(3), 209-221.

95. Spearman, C. (1904). The proof and measurement of association between two things. *Am. J. Psychol., 15*(1), 72-101.

96. Stocking, M. L., Swanson, L., & Pearlman, M. (1993). Application of an automated item selection method to real data. *Appl. Psychol. Meas., 17*(2), 167-176.

97. Sun, K. T., Chen, Y. J., Tsai, S. Y., & Cheng, C. F. (2008). Creating IRT-based parallel test forms using the genetic algorithm method. *Appl. Meas. Educ., 21*(2), 141-161.

98. Swanson, L., & Stocking, M. L. (1993). A model and heuristic for solving very large item selection problems. *Appl. Psychol. Meas., 17*(2), 151-166.

99. Talbi, E.-G. (2009). *Metaheuristics: from design to implementation* (Vol. 74). John Wiley & Sons.

100. Theunissen, T. (1985). Binary programming and test design. *Psychometrika, 50*(4), 411-420.

101. Theunissen, T. (1986). Some applications of optimization algorithms in test design and adaptive testing. *Appl. Psychol. Meas., 10*(4), 381-389.

102. Timminga, E., van der Linden, W., & Schweizer, D. A. (1996). ConTEST (Computer program and manual). Groningen, The Netherlands.

103. Turing, A. M. (1936). On computable numbers, with an application to the Entcheidungsproblem. *J. Math., 58*, 345-363.

104. van der Linden, W. J. (1986). The changing conception of measurement in education and psychology. *Appl. Psych. Meas., 10*(4), 325-332.

105. van der Linden, W. J. (2005). *Linear models for optimal test design (statistics for social and behavioral sciences).* Springer.

106. van der Linden, W. J., & Adema, J. J. (1998). Simultaneous assembly of multiple test forms. *J. Educ. Meas., 35*(3), 185-198.

107. van der Linden, W. J., & Boekkooi-Timminga, E. (1988). A zero-one programming approach to Guiliksen's matched random subtests method. *Appl. Psychol. Meas., 12*(2), 201-209.

108. van der Linden, W. J., & Boekkooi-Timminga, E. (1989). A maxmin model for IRT-based test design with practical constraints. *Psychometrika, 54*(2), 237-247.

109. Wolsey, L. A., & Nemhauser, G. L. (1999). *Integer and combinatorial optimization.* John Wiley & Sons.

110. Yen, W. M. (1983). Use of three-parameter logistic model in the development of standardized achievement test. In R. K. Hambleton, *Applications of item response theory* (pp. 123-141).

111. Yin, P. Y., Chang, K. C., Hwang, G. Y., Hwang, G. H., & Chan, Y. (2006). A particle swarm optimization approach to composing serial test sheets for multiple assessment criteria. *J. Edu. Tech. Soc., 9*(3), 3-15.

Biografija

Kandidat Miroslava Ignjatović (devojačko Mitrović) je rođena 20.01.1974. god u Nišu. Osnovnu i srednju školu je završila u Beogradu. Diplomirala je 2000. god na Elektrotehničkom fakultetu u Beogradu na odseku za Računarsku tehniku i informatiku.

Od 2001. do 2009. god. je živela i radila u Holandiji, gde je bila zaposlena kao softverski inženjer u nekoliko firmi, od kojih je najpoznatija Air France (KLM). U IT odeljenju firme Air France je bila stalno zaposlena i radila je na razvoju portala koji koriste piloti, zatim na održavanju Frequent Flyer programa, kao i na razvoju drugih projekata. U okviru rada se usavršavala i sertifikovala za Java tehnologiju.

2009-te god. se vraća u Srbiju gde počinje sa nastavnim radom u Visokoj školi informacionih i komunikacionih tehnologija, u Beogradu, kao saradnik na predmetima Projektovanje softvera, Administriranje baza podataka, Baze podataka, Programiranje i Objektno orjentisano programiranje. Učestvovala je kao rukovodilac na TEMPUS projektu: "Vizuelnost i Matematika kroz vizuelnu umetnost, nauku i razne aktivnosti", 530394-TEMPUS-1-2012-1-HU-TEMPUS-JPHES koji je trajao od 2012 do 2015 god.

Doktorske akademske studije je upisala prvi put u decembru 2009-te god. i nakon položenih svih ispita i pauze, ponovo u oktobru 2016. god., na modulu Softversko inženjerstvo, pod mentorstvom prof. dr Igora Tartalje. Tokom doktorskih studija je objavila dva rada u međunarodnim časopisima sa impakt faktorom, dva rada u međunarodnim časopisima bez impakt faktora, jedan rad na međunarodnoj konferenciji i više radova na domaćim konferencijama.

образац изјаве о ауторству

Изјава о ауторству

Име и презиме аутора Мирослава Игњатовић

Број индекса 5042/2016

Изјављујем

да је докторска дисертација под насловом

Хеуристика за аутоматско састављање паралелних тестова знања

- резултат сопственог истраживачког рада;
- да дисертација у целини ни у деловима није била предложена за стицање друге дипломе према студијским програмима других високошколских установа;
- да су резултати коректно наведени и
- да нисам кршио/ла ауторска права и користио/ла интелектуалну својину других лица.

Потпис аутора

У Београду, 28.03.2022.

М. Игњатовић

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора Мирослава Игњатовић

Број индекса 5042/2016

Студијски програм Електротехника и рачунарство

Наслов рада Хеуристика за аутоматско састављање паралелних тестова знања

Ментор проф. др Игор Тарталја

Изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла ради похрањена у **Дигиталном репозиторијуму Универзитета у Београду.**

Дозвољавам да се објаве моји лични подаци везани за добијање академског назива доктора наука, као што су име и презиме, година и место рођења и датум одбране рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

Потпис аутора

У Београду, 28.03.2022

M. Ignjatović

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић" да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

Хеуристика за аутоматско састављање паралелних тестова знања

која је моје ауторско дело.

Дисертацију са свим прилозима предао/ла сам у електронском формату погодном за трајно архивирање.

Моју докторску дисертацију похрањену у Дигиталном репозиторијуму Универзитета у Београду и доступну у отвореном приступу могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons) за коју сам се одлучио/ла.

1. Ауторство (CC BY)

2. Ауторство – некомерцијално (CC BY-NC)

3. Ауторство – некомерцијално – без прерада (CC BY-NC-ND)

4. Ауторство – некомерцијално – делити под истим условима (CC BY-NC-SA)

5. Ауторство – без прерада (CC BY-ND)

6. Ауторство – делити под истим условима (CC BY-SA)

(Молимо да заокружите само једну од шест понуђених лиценци.
Кратак опис лиценци је саставни део ове изјаве).

Потпис аутора

У Београду, 28.03.2022

M. Kovatović

1. **Ауторство**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце, чак и у комерцијалне сврхе. Ово је најслободнија од свих лиценци.

2. **Ауторство – некомерцијално**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела.

3. **Ауторство – некомерцијално – без прерада**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела. У односу на све остале лиценце, овом лиценцом се ограничава највећи обим права коришћења дела.

4. **Ауторство – некомерцијално – делити под истим условима**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца не дозвољава комерцијалну употребу дела и прерада.

5. **Ауторство – без прерада**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца дозвољава комерцијалну употребу дела.

6. **Ауторство – делити под истим условима**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца дозвољава комерцијалну употребу дела и прерада. Слична је софтверским лиценцама, односно лиценцама отвореног кода.