A probabilistic quality of service model for nondeterministic service compositions.

Kurdija, Adrian Satja

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Adrian Satja Kurdija

A PROBABILISTIC QUALITY OF SERVICE MODEL FOR NONDETERMINISTIC SERVICE COMPOSITIONS

DOCTORAL THESIS

Zagreb, 2020

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Adrian Satja Kurdija

A PROBABILISTIC QUALITY OF SERVICE MODEL FOR NONDETERMINISTIC SERVICE COMPOSITIONS

DOCTORAL THESIS

Supervisor: Associate professor Marin Šilić, PhD

Zagreb, 2020

Sveučilište u Zagrebu
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Adrian Satja Kurdija

VJEROJATNOSNI MODEL KVALITETE USLUGE ZA NEDETERMINISTIČKE KOMPOZICIJE USLUGA

DOKTORSKI RAD

Mentor: Izv. prof. dr. sc. Marin Šilić

Zagreb, 2020.

Doctoral thesis was made at the University of Zagreb,
Faculty of Electrical Engineering and Computing,
Department of Electronics, Microelectronics, Computer and Intelligent Systems

Supervisor: Associate professor Marin Šilić, PhD

Doctoral thesis contains: 101 pages.

Doctoral thesis number: _____

About the Supervisor

Marin Šilić was born in 1983 in Sarajevo, Bosnia and Herzegovina. He finished the elementary school and the gymnasium in Makarska. He received his M.Sc. degree in 2007, and his Ph.D. degree in 2013 at the Faculty of Electrical Engineering and Computing, University of Zagreb, under the supervision of Professor Siniša Srbljić. During his studies at the University of Zagreb, Marin was receiving a scholarship from the Croatian Ministry of Science. As an outstanding student he was enrolledin advanced study program with a special emphasis on the research work. He was awarded with the "Josip Lončar" award which is given to the top graduating students in computing at the University of Zagreb. In 2007, Marin joined the Faculty of Electrical Engineering and Computing at the University of Zagreb as a research assistant. He actively participated on the research project entitled “End-User Tool for Gadget Composition” sponsored by Croatian Ministry of Science and Google. In 2008, Marin was an software engineering intern in Google Inc., in New York, USA. He was working in the *Google Docs* team on the design and implementation of the *Google Spreadsheets List View*. Marin has presented his research results in the papers that are published in the respected journal and at the top software engineering venue. He is now an associate professor at the University of Zagreb, Faculty of Electrical Engineering and Computing. His research interests span distributed systems, service-oriented computing, software engineering, software reliability. He is a member of the IEEE.

O mentoru

Marin Šilić rođen je 1983. godine u Sarajevu u Bosni i Hercegovini. Osnovnu školu i opću gimnaziju završio je u Makarskoj. Diplomirao je 2007. godine, a doktorirao 2013. godine na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu pod mentorstvom prof. dr. sc. Siniše Srbljića. Tijekom studija primao je stipendiju Ministarstva Znanosti Republike Hrvatske namijenjenu posebno nadarenim studentima. Kao izvrstan student upisao je poseban program završetka studija s naglaskom naznanstveno-istraživački rad. Nakon završetka studija dobio je brončanu plaketu "Josip Lončar"koja se uručuje najboljim studentima računarstva na Sveučilištu u Zagrebu. Od 2007. godine zaposlen je kao znanstveni novak na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu. Aktivno je sudjelovao na istraživačkom projektu “End-User Tool for Gadget Composition” pod pokroviteljstvom Ministarstva znanosti Republike Hrvatske te kompanije Google. U 2008. godini boravio je na znanstvenom usavršavanju u kompaniji Google u uredu u New York-u, SAD. Tijekom usavršavanja radio je u *Google Docs* timu na oblikovanju i ostvarenju primjenskog sustava *Google Spreadsheets List View*. Rezultate svojih istraživanja Marin je opisao u člancima koji su objavljeni u uglednom časopisu i najjačem

skupu istraživača područja programskog inženjerstva. Radi kao izvanredni profesor na Fakutltetu elektrotehnike i računarstva Sveučilišta u Zagrebu. Znanstveni interesi mu pokrivaju distribuirane sustave, računarstvo zasnovano na uslugama, softversko inženjerstvo i pouzdanost softvera. Član je strukovne udruge IEEE.

To my parents, for all their inexhaustible love and help.

Acknowledgements

I would like to use this opportunity to thank the people from this Faculty who helped me in various ways during my PhD study and gave me professional support, personal support, and often both.

First and foremost, I express my sincere gratitude to my advisor, assistant professor Marin Šilić, who introduced me to the research area of this dissertation, led me with his knowledge and experience, showed me trust and gave huge help and support in all aspects and stages of my work and research. This dissertation would not have existed without him, and our fruitful collaboration will certainly continue. Thank you, Marin!

I am very grateful to my previous advisor, professor Siniša Srbljić, who gave me great support and many useful advices from his extensive professional and personal experience. Thank you, professor!

I cannot forget to mention all the help I received from assistant professors Goran Delač and Klemo Vladimir, my senior colleagues in work and research. I am very grateful for our collaboration in all these years.

Here I must mention professor Željko Ilić, with whom I wrote several papers in the area of wireless networks, and hope to write more. Five years ago he introduced me to my advisor, which eventually led to my application and work at this Faculty. Thank you, Željko!

Finally, to my colleagues in the lab, Petar Afrić and Lucija Šikić - thank you for wonderful ongoing collaboration, it is a great pleasure working with you!

Abstract

Cloud computing becomes the prevailing aspect of software engineering. In Service-Oriented Architecture (SOA), various atomic services respond to various client requests in a cloud. Selecting the actual service instance to process a particular request can be an issue when requirements for multiple Quality of Service (QoS) properties need to be satisfied for many users simultaneously. The problem becomes more complex if we take into account the compositeness of users' applications, which consist of many tasks with non-deterministic execution plan, where QoS properties are calculated over the whole composition. In this thesis, a fast heuristic model for multi-criteria service selection is proposed, designed for multi-user composite workflows with the goal of satisfying as many as possible of the given QoS requirements. To develop such a method, some essential tools were developed, such as probabilistic formulas for estimating QoS in non-deterministic service compositions. Another contribution is a real-time adaptive QoS prediction model, which can be used before service selection to provide its input – the QoS values for atomic services. The extensive experiments have demostrated the accuracy and efficiency of the proposed models compared to the existing ones.

Keywords: web services, quality of service, QoS, service composition, QoS prediction, service selection, collaborative filtering, transportation problem.

Prošireni sažetak

Vjerojatnosni model kvalitete usluge za nedeterminističke kompozicije usluga

Računarstvo u oblaku postaje prevladavajući aspekt softverskog inženjerstva. Sve manje računanja vrši se na lokalnim strojevima, a sve više u oblaku. Mnogi primjenski sustavi temelje se na web uslugama koje odgovaraju na brojne zahtjeve korisnika. *Arhitektura zasnovana na uslugama* arhitektonski je stil koji pretpostavlja velik broj usluga za višekratnu uporabu koje pružaju određene funkcionalnosti kroz svoja javno dostupna sučelja.

Primjenski sustav u oblaku može se smatrati kompozicijom usluga, a pojedine usluge koje sudjeluju u kompoziciji nazivaju se atomarnima. Kompozicija usluga može biti složena i njezin plan izvođenja nije nužno deterministički. Moguće su različite kompozicijske strukture: sekvencijalna izvedba, paralelno izvođenje, grananje i petlje. Na zadatak (funkcionalnost) potrebnu kao dio primjenskog sustava (apstraktnu uslugu) obično može odgovoriti mnogo mogućih kandidata – odgovarajuće instance usluge na poslužiteljima diljem svijeta, koje smatramo klasom usluga. Od mogućih kandidata treba izabrati odgovarajući prema realističnim kriterijima. Naime, svaka usluga (kao i cijeli primjenski sustav) ima specifična svojstva kvalitete usluge koja se uzimaju u obzir prilikom oblikovanja postupka odabira. Primjeri uključuju dostupnost, pouzdanost, vrijeme odziva, propusnost, reputaciju, trošak itd. Ova svojstva ovise o brojnim čimbenicima od kojih su neki vezani za korisnika, neki za uslugu, a neki za okruženje (mrežu).

U složenom primjenskom sustavu gdje se za korisnika poziva više usluga, svojstva kvalitete usluge izračunavaju se za čitavu kompoziciju. Korisnik može imati specifične zahtjeve koji definiraju minimalne/maksimalne zadovoljavajuće vrijednosti za određena svojstva kvalitete usluge. U sustavu s mnogo korisnika potreban je robustan model koji će globalno optimirati odabir usluga za pojedine korisnike i pojedine fukcionalnosti. Zbog nedeterminizma unutar plana izvođenja kompozicije usluga, treba uzeti u obzir vjerojatnosti grananja.

Postojećim pristupima ovom problemu nedostaje učinkovitosti i/ili općenitosti. Stoga je glavni cilj ove doktorske disertacije predložiti brzi heuristički model za višekriterijski odabir web usluga, oblikovan za višekorisničke kompozicije usluga s ciljem zadovoljavanja svih (ili što je više moguće) zadanih zahtjeva na kvalitetu usluge. U svrhu razvijanja takvog algoritma, razvijeni su neki pomoćni alati, poput vjerojatnosnih formula za procjenu očekivanog broja poziva svake usluge u nedeterminističkoj kompoziciji, te procjenu očekivanih ukupnih svojstava kvalitete usluge za zadani odabir konkretnih usluga. Vremenska učinkovitost metode odgovara polinomnoj vremenskoj složenosti, koja se ostvaruje svođenjem problema odabira usluga na kombinatorne probleme transporta za čije rješenje postoje efikasni algoritmi kao što su Vogelova aproksimacijska metoda (za početno rješenje) i transportna simpleks metoda za unapređenje rješenja. Općenitost modela odgovara istodobnom razmatranju većeg broja korisnika (ili primjenskih sustava) od kojih svaki ima (ne nužno jednak) plan izvođenja tj. kompoziciju

s više zahtjeva za pojedine apstraktne usluge (klase usluga), uz individualne zahtjeve na minimalne (ili maksimalne) vrijednosti pojedinih stvojstava kvalitete usluge. Planovi izvođenja dopuštaju sve standardne kompozicijske strukture (slijed, paralelno izvođenje, uvjetno grananje i petlje), uzimajući u obzir vjerojatnosti u slučaju grananja i petlji, koje se mogu dobiti npr. statistički na temelju bilježenja zapisa (logova) uporabe primjenskog sustava od pojedinog korisnika. Model je odgovoran za odabir odgovarajućih instanci usluga koje će se pozivati za sve zahtjeve, uzimajući u obzir njihova ograničenja propusnosti da se "popularne" usluge ne bi preopteretile. Model uzima u obzir i individualizirane vrijednosti svojstava kvalitete usluge koja ovise o parovima korisnik-usluga, što je realna pretpostavka koja nedostaje u mnogim prethodnim radovima gdje se zanemaruje utjecaj korisničkih čimbenika na kvalitetu usluge, tj. pretpostavlja se da kvaliteta ovisi samo o usluzi.

Predloženi algoritam odabira usluga ostvaruje se svođenjem problema na nekoliko neovisnih transportnih problema (engl. *transportation problem*) koji se iterativno redefiniraju i ponovno rješavaju dok se ne ispune korisnički zahtjevi za kvalitetu usluge. Unutar transportnog problema koristi se heuristički definirana cijena (nazvana *matching difficulty* ili "težina uparivanja") koja se temelji na korisničkim zahtjevima i procjenama ukupne (agregirane) kvalite usluge za pojedini odabir usluga. Za sam algoritam odabira usluga predložene su dvije varijacije: SS-TSM (*Service Selection using Transportation Simplex Method*) i SS-VAM (*Service Selection using Vogel Approximation Method*), koje se razlikuju se u pristupu rješavanju pojedinačnih transportnih problema. Ispostavlja se da SS-TSM pristup u većini slučajeva radi učinkovitije.

Učinkovitost, više nego preciznost, ostaje u središtu pažnje većine aktualnih napora u području računarstva u oblaku gdje broj korisnika i instanci usluga konstantno raste. Stoga je vrijeme izvođenja bila najvažnija mjera u našoj evaluaciji predloženih algoritama. Izveli smo opsežne eksperimente na posebnim i općenitijim slučajevima u kojima su predložene metode uspoređivane s optimalnim pristupom odabiru usluga (temeljenim na cjelobrojnom programiranju) i njegovim heuristički poboljšanjima kada su ona primjenjiva. Naši eksperimenti pokazuju da je predloženi SS-TSM model dominantan pristup u većini eksperimenata, uglavnom zbog značajnog smanjenje vremena izvođenja. Djelomični izuzetak je scenarij s jednom klasom usluga (gdje se aplikacija svodi na jedan poziv usluge), gdje predloženi (nešto manje precizan) SS-VAM pristup može biti brži u slučajevima velikih ograničenja za propusnosti usluga, dok alternativni pristup iz literature (zasnovan na svođenju problema na problem pridruživanja ili *assignment problem*) može biti brži kad su propusnosti niže. Zaključno, osim što je općenitija od postojećih pristupa, predložena metoda (konkretno, SS-TSM) pokazuje se i učinkovitijom od alternativnih (do približno 5 puta).

Budući da vrijednosti kvalitete usluge nisu unaprijed poznate za većinu parova korisnik-usluga, prije odabira usluga potrebno je provesti predviđanje nepoznatih vrijednosti kvalitete

usluge. Ono se za pojedinog korisnika zasniva na temelju poznatih vrijednosti za njemu slične korisnike i istu uslugu, ili za pojedinu uslugu na temelju poznatih vrijednosti za njoj slične usluge i istog korisnika. Odgovarajući model predviđanja treba biti dovoljno brz (primjenjiv u realnom vremenu) i prilagodljiv s obzirom na promjene u okruženju (npr. povećanje opterećenja mreže). Nakon toga, u fazi odabira usluge, koriste se predviđene vrijednosti kvalitete usluge. Stoga, kao još jedan doprinos u ovoj disertaciji, predstavljen je model predviđanja kvalitete usluge kao podrška u stvarnom vremenu za odabir atomarnih kandidata web usluga na osnovi njihovih svojstava kvalitete usluge. Predloženi pristup temelji se na suradničkom filtriranju (engl. *collaborative filtering*) te zadovoljava sljedeće osnovne zahtjeve: brzo i precizno predviđanje vrijednosti kvalitete usluge i prilagodljivost s obzirom na promjene u okolini. Model unaprijed računa sličnosti između korisnika i usluga s pomoću približnog množenja matrica radi smanjenja vremenske složenosti. Prilikom predviđanja kvalitete usluge za pojedini par korisnik-usluga, model uzima u obzir slične korisnike i usluge, ali povećava točnost predviđanja uključivanjem pouzdanosti poznatih zapisa koja ovisi o broju prethodnih poziva. Vremenska složenost dodatno se smanjuje spremanjem popisa sličnih korisnika i usluga koji se ažuriraju u stvarnom vremenu. Model se prilagođava promjenjivom okruženju: noviji zapisi dodaju se tako da imaju veći utjecaj (težinu) na predviđanje novih vrijednosti, čime model brzo uči o promjenama u sustavu. Da bismo procijenili uspješnost našeg pristupa, prikupili smo javno dostupne skupove podataka za kvalitetu web usluga, te smo oblikovali i proveli niz eksperimenata radi provjere pretpostavki i različitih aspekata predloženog modela. Opsežni rezultati evaluacije potvrđuju da je predloženi model brži od alternativnog, uz usporedivu točnost predviđanja. Osim toga, točnost/brzina približnog množenja matrica može se podešavati mijenjanjem veličine uzorka (točniji rezultat dobiva se većim brojem operacija). Ova značajka fleksibilnosti sačuvana je i u našem modelu: možemo proizvesti točnija predviđanja povećanjem veličine slučajnog uzorka, što zauzvrat rezultira lošijim vremenskim rezultatima modela. S druge strane, vrijeme izračuna možemo poboljšati smanjivanjem veličine slučajnog uzorka, što će rezultirati manje točnim predviđanjima. Na taj način možemo balansirati između dvaju suprotnih zahtjeva, točnosti i preciznosti, te se predloženi pristup sa svojim fleksibilnim dizajnom može primijeniti u različitim okruženjima.

Ova disertacija bavi se, dakle, dvama povezanim problemima: predstavlja i model predviđanja kvalitete usluge, kao i vjerojatnosni model odabira usluga za višekorisničke primjenske sustave s nedeterminističkom kompozicijom usluga. Modeli su pažljivo razvijeni s ciljem točnosti, primjenjivosti u stvarnom vremenu, a osobito efikasnosti. Napravljeni su opsežni eksperimenti kako bi se potvrdila valjanost predloženih modela.

Budući da se naš model predviđanja temelji na suradničkom filtriranju, kao sporedni doprinos predložili smo mjere globalne koreliranosti (engl. *Global Correlation Measure* ili GCM) za skup podataka za suradničko filtriranje. Naime, sustavi za preporučivanje zasnovani na surad-

ničkom filtriranju oslanjaju se na skupove podataka koji sadrže interese korisnika za različite stavke, a točnost različitih pristupa predviđanja ovisi o količini sličnosti (korelacije) između korisnika i stavki u skupu podataka. Kao heuristička procjena ovog aspekta kvalitete podataka, koji može poslužiti kao grubi pokazatelj sposobnosti predviđanja, definirali smo globalnu mjeru koreliranosti korisnika (GUCM) i globalnu mjeru koreliranosti stavki (GICM) skupa podataka koji sadrži poznate interese ili ocjene stavki od strane korisnika. Ove mjere mogu se koristiti za brzu procjenu je li skup podataka pogodan za suradničko filtriranje i možemo li očekivati visoku točnost predviđanja, tj. isplati li se razvijati prediktivni model. Predložene su mjere u rasponu od 0 do 1 i opisuju kvalitetu skupa podataka s obzirom na međusobne sličnosti korisnika i stavki: viša mjera ukazuje na veći broj sličnih parova i bolju mogućnost predviđanja. Eksperimentalno smo provjerili da predložene mjere koreliranosti (GUCM, GICM i njihov prosjek GCM) udovoljavaju željenim zahtjevima. Naime, one su u korelaciji s količinom sličnosti među korisnicima ili stavkama te brojem sličnih parova, kao i s točnošću standardnih modela predviđanja. Mjere su, očekivano, negativno korelirane s brojem prirodnih skupina sličnih korisnika ili stavki. Budući da se predložene mjere mogu učinkovito izračunati, u vremenu proporcionalnom broju poznatih ocjena, one mogu biti korisne pri odlučivanju o primjeni suradničkog filtriranja na određenom skupu podataka – rezultati ukazuju da vrijednost GCM-a viša od 0,14 sugerira visoku točnost predviđanja.

U nastavku ovog sažetka dajemo pregled poglavlja ove doktorske disertacije.

U prvom poglavlju (1 *"Introduction"*) predstavljen je kratak pregled razmatranog područja i problema koji se pojavljuju, uz motivaciju za pristupe njihovog rješavanja.

Drugo poglavlje (2 *"QoS in Service-Oriented Architecture"*) opisuje kontekst kvalitete usluge u arhitekturi zasnovanoj na uslugama i detaljnije objašnjava njezine probleme: predviđanje kvalitete usluge i odabir usluga. Navode se postojeći često korišteni pristupi njihovog rješavanja, kao i njihove prednosti i mane.

U trećem poglavlju (3 *"Real-time Adaptive QoS Prediction Using Approximate Matrix Multiplication"*) opisan je predloženi model predviđanja kvalitete usluge, kao podrška u stvarnom vremenu za odabir atomarnih usluga u kompoziciji na temelju njihovih predviđenih svojstava. Predloženi pristup zadovoljava sljedeće zahtjeve: brzo i točno predviđanje vrijednosti kvalitete usluge, te prilagodljivost s obzirom na promjene u okruženju. Model unaprijed računa sličnosti između korisnika i usluga s pomoću približnog množenja matrica radi smanjenja vremenske složenosti. Pri izračunavanju predviđanja za par korisnik-usluga, model razmatra slične korisnike i usluge, ali poboljšava točnost predviđanja uključivanjem broja promatranih zapisa. Vremenska složenost dodatno je smanjena održavanjem popisa sličnih korisnika i usluga koji se ažuriraju u stvarnom vremenu. Model se prilagođava promjenjivom okruženju: noviji zapisi imaju veću težinu, tj. veći utjecaj u formuli predviđanja. Budući da je model zasnovan na paradigmi suradničkog filtriranja, kao sporedni doprinos definira se i mjera kvalitete bilo kojeg

skupa podataka za suradničko filtriranje.

Četvrto poglavlje (4 *"Compositional QoS Model"*) navodi kompozicijske strukture u složenom primjenskom sustavu te definira vjerojatnosni model odgovoran za računanje očekivanog broja poziva usluge za svaku apstraktnu uslugu u nedeterminističkom planu izvođenja te procjenu očekivane vrijednosti pojedinog svojstva kvalitete usluge za određeni odabir atomarnih usluga. U ovom poglavlju definiramo i nov pristup procjeni korisnosti odabira pojedine atomarne usluge za pojedinog korisnika koji uzima u obzir i druge usluge u njegovoj kompoziciji, kao i njegove zahtjeve na kvalitetu usluge.

Alati opisani u četvrtom poglavlju upotrijebljeni su u općem modelu za odabir usluga, opisanom u petom poglavlju (5 *"Fast Multi-Criteria Service Selection for Multi-User Composite Applications"*). Ovaj je model efikasna heuristička metoda za višekriterijski odabir usluga, oblikovana za složene kompozicije usluga kod više korisnika, s ciljem zadovoljavanja što većeg broja njihovih zahtjeva na kvalitetu usluge. Predložena metoda svodi problem na nekoliko neovisnih problema transporta, iterativno poboljšavajući rješenje, te su predstavljene dvije njezine varijante.

Šesto poglavlje (6 *Evaluation*) opisuje iscrpne eksperimente u kojima se analiziraju različiti aspekti predloženih modela u usporedbi s postojećim pristupima. Rezultati podupiru tvrdnje o točnosti, učinkovitosti i skalabilnosti predloženih modela te su razvidne njihove prednosti u odnosu na druge modele iz literature.

O znanstvenim doprinosima i zaključcima raspravlja se u sedmom poglavlju (7 *"Conclusion"*).

Ključne riječi: web usluge, kvaliteta usluge, kompozicija usluga, predviđanje kvalitete usluge, odabir usluga, suradničko filtriranje, problem transporta.

Contents

Chapter 1

Introduction

Cloud computing becomes the prevailing aspect of software engineering. Less and less processing is done on local machines, while more and more processing is done in a cloud. Many applications are based on cloud services responding to numerous client requests. For example, an application such as Gmail uses various services (mail, chat/hangouts, calendar, tasks, translate). Service-Oriented Architecture (SOA) is an architectural style that assumes a variety of reusable services which provide certain functionalities through their publicly accessible interfaces.

As another example of a SOA application, consider a multi-tenant Service-Based System (SBS) which provides travel booking service for different travel agents (tenants). Such an applications considers various possibilities for trip scheduling, with access to providers of airline data, accommodation data, traffic/directions data, etc. Various tasks are performed in such an application: airline ticket search, car rental, hotel search, insurance quote, train ticket search, cruise ticket search, weather queries, etc. This application is run in a cloud with multiple services with different functionalities responding to the aforementioned queries.

Therefore, a cloud application can in effect be described as a *service composition*, and services which participate in the composition are called *atomic services*. More examples include intelligent energy systems, networking, financial/trading applications, as well as applications which rely on various news feeds, social network posts, etc. [1, 2, 3, 4, 5]. As these workflows can be very complex, the execution of atomic services within the application is not necessarily deterministic and various compositional structures are possible: sequential execution, parallel execution, branching, and loops [6].

A functionality needed as a part of an application (or simply *task*) can usually be served by many possible candidates – corresponding service instances on cloud servers all over the world [7]. Equivalent services corresponding to the same functional requirements are referred to as a *service class* or an *abstract service* [8].

When there are many possible candidates, one of them has to be chosen according to realistic

criteria. Namely, each service (as well as the whole application) has specific properties, which are considered when designing a selection procedure. The *functional* properties of a service correspond to its definition: what the service does. They ensure correct logical operating. The *non-functional* properties are often called *Quality of Service* (QoS) and include availability, reliability, response time, throughput, reputation, cost, etc. They have a significant impact on the perceived quality of the composite service [9]. Naturally, both functional and non-functional properties of a composite service are determined by the functional and non-functional properties of selected atomic services which participate in the composition. Service properties themselves depend on various user-specific, service-specific, and environment-specific parameters, such as locations of user and service, system properties, network properties, etc.

In a composite (multi-task) application where multiple services are invoked for a user, the QoS properties are calculated over the whole workflow. To define or measure "satisfaction" with the global QoS, a user may have specific *QoS requirements* which define minimal/maximal satisfactory values for certain QoS properties. These requirements can be part of a Service Level Agreement (SLA).

In a system with many users (tenants) with minimal requirements on various QoS properties such as response time, reliability, and price [10, 11], there is a need of a robust model which will globally optimize the service selection and provide an answer which service instances will be invoked by which users for which tasks. In order to provide a service selection with high QoS, or at least with minimal satisfactory QoS (according to the SLA requirements), the selection model must take the user-service QoS values into account.

However, a challenging fact is that the QoS values are not known for most user-service pairs, and there are several reasons for that. First, the past invocation data contains only the QoS values measured for the user-service pairs for which an actual invocation took place, which is a small subset of all possible user-service pairs. The number of such pairs is quadratic with respect to the number of users/services (e.g., a million for 1000 users and services), which is too large to perform measurements. Stress-testing such a network would also suffocate the network, which would give unrealistic QoS values.

For these reasons, when the actual QoS values are unknown, the *QoS prediction* takes place before the service selection. The prediction of an unknown QoS value for a user-service pair is done based on the known QoS values for similar user-service pairs. Afterwards, in the *service selection* phase, the QoS values are assumed to be known or already predicted with reasonable accuracy. Such a model can take various assumptions on the system properties. It can consider a single user at the time, or multiple users at the same time, and in both cases it can consider service compositions (multiple service classes) instead of a single service class. The output of this phase is a mapping: which service instance is invoked for which user at a particular point in the composition.

To better grasp the complexity of such a problem, consider a scenario in which travel agencies act as users (tenants) in a multi-tenant Service-Based System (SBS) [12] with multiple services responding to application requests such as accommodation search, flight search, insurance queries, popular event search, weather queries, etc. The travel agencies may use different execution plans in an application, with some tasks (e.g. accommodation search) depending on the other (e.g. flight search) in a complex compositional structure. Many services may provide the same functionality, but with different QoS properties (slow/fast response, more/less reliable results, free/paid with different prices, higher/lower reputation, etc.). The users share the execution engine of the SBS which performs the service selection for the observed users with the goal of respecting the individual QoS requirements for each user's service composition.

Approaching this problem, many considerations must be taken into account. Some service instances are infeasible to some users because of bad QoS parameters (resulting from e.g. location distance [13]); some are more reliable or have a better response time, but the price of invocation is higher; some of them might lack availability due to their heavy usage and/or inadequate throughput. Even for non-composite (single-task) applications, greedy algorithms will not work: selecting the closest, the fastest, the most reliable or the cheapest service instance for each request will generally overload the most "popular" service instances (breaking their throughput limit), and might also fail with respect to other criteria. In a composite case, especially if there are multiple potential execution paths for a composition (because of probabilistic branching), it is difficult to locally estimate whether any particular user-service choice leads to a good solution. For this reason, Mixed Integer Programming (MIP) techniques have been employed to solve the service selection problem [12], often with local reductions of the search space based on heuristics and/or greedy algorithms [14, 15, 16], since optimal MIP solution requires exponential time complexity.

In this disertation, attemps at both QoS prediction model [17] and service selection model [18] are presented. The models are carefully developed with the aims of accuracy, real-time applicability, and efficiency in mind. Extensive experiments are made to verify the validity of the proposed models.

The rest of the dissertation is organized as follows. Chapter 2 describes the context of QoS in Service-Oriented Architecture (SOA) and explains the tackled problems (QoS prediction and service selection) in more detail. Some common, recent or influential ("state-of-the-art") methods for dealing with these problems are presented, with their strengths and weaknesses outlined.

Chapter 3 describes the proposed QoS prediction model, published by the author in [17]. This model is a real-time support for selection of atomic service candidates based on their QoS properties while constructing composite applications. The proposed approach satisfies the following requirements: (i) fast and accurate prediction of QoS values, and (ii) adaptability

with respect to environment changes. The model precomputes the similarities between users and services using approximate matrix multiplication to reduce the time complexity. When calculating a prediction for a user-service pair, the model considers similar users and services, but enhances the prediction accuracy by incorporating the number of observed records. Time complexity is further reduced by storing the lists of similar users and services which are updated in real-time. The model adapts to the changing environment: newer records are set to have greater influence on the predictions.

Since this model is based on the collaborative filtering paradigm, as a side contribution we define a quality measure for a collaborative filtering dataset. This *Global Correlation Measure* (GCM) is published by the author in [19].

Chapter 4 considers compositional structures in a service composition (a composite application) and defines a probabilistic compositional QoS model responsible for computing an expected number of service invocations for each abstract service in a composition, as well as computing estimated QoS for any QoS property based on the actual service instances chosen for the composition. It also defines a novel paradigm for estimating selection utility for a particular task in a composition, which measures the cost of a given user-service match (choice) with respect to her/his compositional QoS requirements. This concept is named *matching difficulty* and is a prerequisite for the service selection algorithm. Both concepts were published by the author in [18].

Chapter 5 describes the proposed service selection model, also published in [18] with the early version published in [20]. This model is a fast heuristic method for multi-criteria service selection, designed for multi-user composite workflows with the goal of satisfying all, or as many as possible, of the given QoS requirements. The proposed method reduces the problem to several independent combinatorial transportation problems, using a global-aware utility cost based on expected compositional QoS, and iterative solution improvements.

Chapter 6 presents the exhaustive and detailed experiments which analyze different quality aspects of the proposed models in comparison with the existing approaches. The analysis of time complexity is given to support the efficiency and scalability claims for the proposed models. The evaluation results are summarized at the end of the chapter. Scientific contributions and conclusions are finally discussed in Chapter 7.

Chapter 2

QoS in Service-Oriented Architecture

Many of the modern software information systems are built upon principles defined in Service-Oriented Architecture (SOA) [21]. SOA is an architectural style that assumes a variety of atomic reusable services which provide certain functionality through their publicly accessible interfaces. More advanced functionality is achieved through atomic services composition into more complex composite services.

The process of atomic services selection is essential for the overall quality of a composite service [22], [23]. Both functional and non-functional properties of a composite service are determined by the functional and non-functional properties of selected atomic services. While functional properties assure correct logical operating, non-functional properties, often called *Quality of Service* (QoS), such as availability, reliability, response time, throughput, reputation, cost, etc., may have a significant impact on the perceived quality of the composite service [9]. Review and classification of QoS attributes of web services can be found in e.g. [24].

Application's quality of service (QoS) may depend on various user-specific, service-specific, and environment-specific parameters [25]. User-specific parameters include the user's location, network and device capabilities, and usage profiles. Service-specific parameters (for a service instance) include its location, computational complexity, and system resources (e.g. CPU, RAM, disk, and I/O operations). Environment-specific parameters include service provider load and network performance at the time of invocation. Regarding service-specific parameters, an important aspect of a service candidate is its *throughput* (other terms are *processing capacity* [26] and *service load* [27]) which dictates the maximal number of requests it can respond to in a given time frame. Finally, invocation *price* – which can correspond to an actual payment price, or a cost in terms of spent resources – can be taken into account. Price can be treated as additional QoS parameter, which can be service-specific and also user-specific.

One of the most important non-functional properties that significantly impacts the QoS of the entire composite application is reliability [28], [29]. Reliability requirements are in focus of current research in service-oriented systems [30], as well as heterogeneous embedded systems

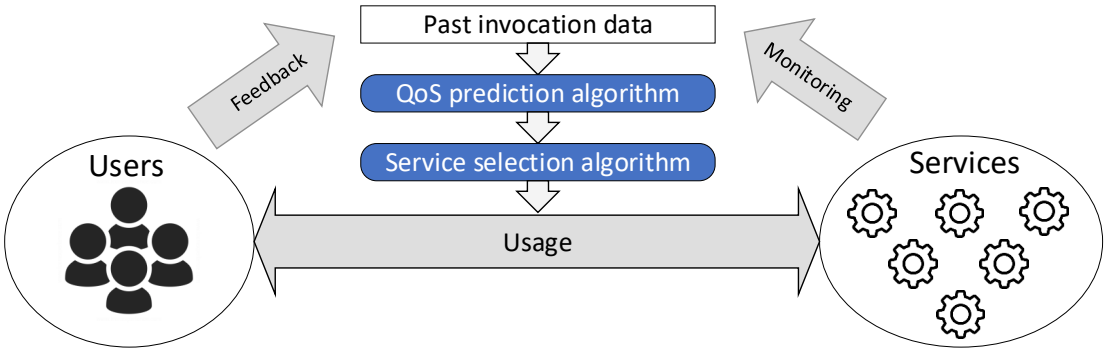


Figure 2.1: QoS-aware service recommendation

[31]. We define service reliability as the probability that a service invocation gets successfully completed, i.e., correct response is retrieved within the specified time constraints. The proposed reliability definition can be found in literature as *successful delivery rate* [32], *user-perceived reliability* [33] and *reliability on demand* [34]. We note that the adopted user-centric definition differs from the traditional system-centric definition that is used for "never-ending" systems [28]. The reliability-on-demand definition is more suitable for web services due to a nature of web service invocations which are discrete and relatively sparse events.

According to the adopted reliability definition, the reliability value can be computed from the past invocation sample as the number of successful invocations divided by the total number of performed invocations. However, real-world service-oriented systems may contain a very large number of users and services, and a single user accesses only a very limited and diverse subset of services. This means that the past invocation sample is very sparse and there is not enough collected data to compute the reliability value in a straightforward way for each user-service pair. One possible solution is to collect more data by performing additional service invocations. Still, there are obstacles which make the proposed solution impractical. For instance, service invocations may be charged, and performing many invocations can significantly impact service performance and thus make the collected data irrelevant [35]. Moreover, note that service-oriented systems are deployed on the Internet, which is a highly dynamic environment where the service invocation outcome is impacted by the variety of different parameters that determine service invocation context (e.g., network bandwidth, geographical location, service load, etc.). In such a dynamic environment, service providers register significant load variations during the day, which is the main cause why the user-perceived QoS values may change depending on the actual time of invocation [36], [37], [38], [39].

For such reasons, predicting the unknown values of QoS properties is often done from the past invocation sample using prediction models. As depicted in Fig. 2.1, service users perceive different QoS properties while accessing various services on the Internet depending on their actual invocation context. A partial past invocation sample can be gained by gathering users

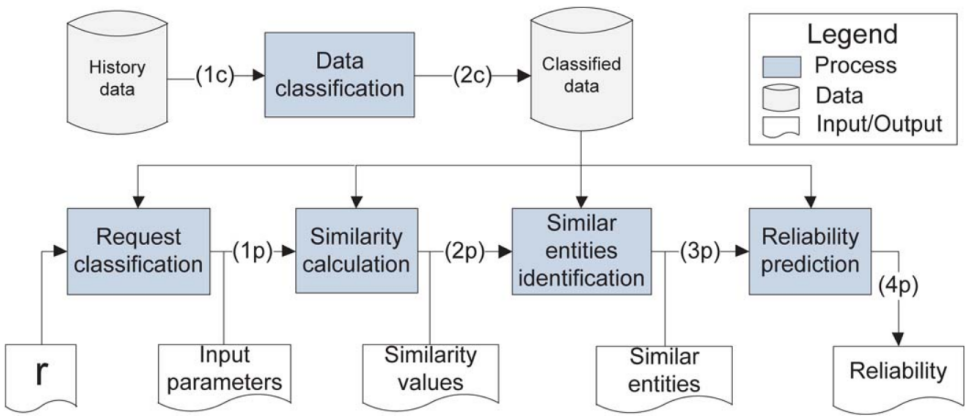


Figure 2.2: Example of a CF prediction model (LUCS) [52]

feedback while accessing services [40], and by collecting records about service invocations by monitoring service providers [41], [42]. In such circumstances where there is only a limited past invocation sample, for user-service pairs that lack sufficient data to compute the reliability value in a straightforward way, the best approach is to utilize prediction models. Finally, based on the output of a prediction model, the recommendation engine or a service composition framework selects the most appropriate service candidates for particular users with respect to predicted QoS properties.

2.1 QoS Prediction

The researchers have proposed a variety of prediction approaches that can be used to model the reliability of traditional system-centric software [43], [44], [45], [28], [29], [46], [47]. All these approaches treat software components as "white boxes" whose reliability is known or can be inferred through behavioral models. Such approaches are not suitable for modeling the reliability of web services because of two main reasons. First, web services are observed as "black boxes" that provide certain functionality and there is usually no insight in their internal structure. Second, services are deployed on the Internet and they get accessed remotely from different locations worldwide. Hence, a lot of external parameters have a significant impact on a particular service invocation. These parameters determine the service invocation context and they can be divided into several categories as already discussed (user-specific, service-specific, and environment-specific parameters).

We should also mention various approaches that model the reliability of composite services [48], [34], [49], [50], [51]. Still, those approaches treat atomic web services as components whose reliability values are already known.

The most successful approaches for predicting reliability of web services rely on users' previous experience aggregated in the past invocation sample. However, as already discussed,

QoS	User 1	User 2	User 3	User 4
Service 1	?	?	0.98	0.993
Service 2	?	0.9	?	?
Service 3	?	?	0.99	?
Service 4	0.97	?	0.95	?
Service 5	?	0.96	?	0.94
Service 6	0.992	?	?	?

Figure 2.3: Example of a user-item reliability matrix

it is not an easy task to acquire a comprehensive past invocation sample. Therefore, the best approach is to use prediction models to estimate the reliability for ongoing service invocations based on the limited past invocation data. The most effective prediction models for reliability of atomic web services are based on *collaborative filtering* (CF) [53] which, informally, calculates reliability for user/service based on the known values for users/services which are similar to them. An example model is sketched in Fig. 2.2.

When services are perceived as black boxes and no extra parameters are taken into account apart from the known QoS values for some user-service pairs, the QoS prediction problem can be handled by generic prediction algorithms initially designed for other recommendation problems, such as predicting user ratings for movies, books, or web shop items. A variety of prediction models have been proposed that utilize the collaborative filtering technique which is commonly applied in modern recommendation systems [53], [54], [55], [56], [57], [58]. There are three basic types of collaborative filtering: *memory-based*, *model-based*, and *hybrid* collaborative filtering. We briefly describe each type and mention its best representatives in the following sections.

2.1.1 Memory-Based Collaborative Filtering

The memory-based CF is borrowed from the area of recommender systems. The idea behind this technique is to extract and filter the information from multiple data sources, and then to leverage this information to estimate the missing data by using the available data from statistically most similar data sources.

This CF type uses *user-item* matrix for storing the data for reliability prediction. Each matrix cell contains a value r_{ui} (row u, column i) which represents the reliability value observed by user u while invoking service i. The user-item matrix is very sparse due to a fact that real-world service-oriented systems may contain a large number of users and services, and each user invokes only a small subset of services, which in turn results in a large number of empty cells in the matrix. Each empty cell represents a reliability value that might need to be predicted; see Fig. 2.3 for illustration.

To predict the missing values in the matrix, memory-based CF can be applied in two differ-

ent ways. The first approach, *UPCC* [59], computes the set of statistically most similar users (with respect to the current user) and estimates the missing value by combining the available values from these users. On the other hand, the *IPCC* approach [60] computes the set of statistically most similar services (with respect to the current service) and estimates the missing value based on the available values from these services. Both approaches use *Pearson Correlation Coefficient* (PCC) as a similarity measure. However, the researchers have shown that better prediction accuracy can be achieved by combining both approaches, i.e., by computing the final prediction as a linear combination of *UPCC* and *IPCC* which is incorporated in the *Hybrid* approach [40], [61].

To formalize the above ideas in a general context, let $r_{u,i}$ denote the rating (e.g. reliability or another QoS property) of user u on item i (web service, shopping item, movie...). Let $\bar{r}_u$ denote the average rating of user u and let $\bar{r}_i$ denote the average rating of item i. The similarity of users u, v is usually calculated as a Pearson Correlation Coefficient (PCC):

$$sim_{u,v} = \frac{\sum_{i \in I_{u,v}} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{u,v}} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{u,v}} (r_{v,i} - \bar{r}_v)^2}}, \tag{2.1}$$

where $I_{u,v}$ is the set of items rated by both users u and v. Analogously, the item similarity is calculated as

$$sim_{i,j} = \frac{\sum_{u \in U_{i,j}} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U_{i,j}} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{u \in U_{i,j}} (r_{u,j} - \bar{r}_j)^2}}, \tag{2.2}$$

where $U_{i,j}$ is the set of users who have rated both items i and j.

We now give prediction formulas for the standard CF approaches based on PCC, which are the above mentioned user-based (UPCC) and item-based (IPCC) predictions [59]. To predict the rating $r_{u,i}$, the user-based approach searches for positive similarities between the current user u and other users v who have rated item i. The prediction is then calculated using the weighted average of their ratings, with weights corresponding to similarities (a user with higher similarity to user u has greater weight in the average):

$$UPCC_{u,i} = \bar{r}_u + \frac{\sum_v sim_{u,v}(r_{v,i} - \bar{r}_v)}{\sum_v sim_{u,v}}. \tag{2.3}$$

The item-based prediction works analogously, taking into account the current user's ratings for items j with positive similarity to the current item i:

$$IPCC_{u,i} = \bar{r}_i + \frac{\sum_j sim_{i,j}(r_{u,j} - \bar{r}_j)}{\sum_j sim_{i,j}}. \tag{2.4}$$

The predictions can be combined by taking their linear combination:

$$HybridPrediction_{u,i} = \alpha\, UPCC_{u,i} + \beta\, IPCC_{u,i}, \tag{2.5}$$

where α and β can be tuned on the known data (a training set) with $\alpha + \beta = 1$.

Although memory-based CF approaches demonstrate promising prediction accuracy, they suffer from serious performance issues stemming from computation of similarities (and re-computation in case of a changing environment). Hence, our goal is to propose a fast prediction approach whose accuracy will be close to the accuracy of *Hybrid*, but whose time performance will be superior.

2.1.2 Model-Based Collaborative Filtering

The model-based CF approaches are often based on more sophisticated techniques such as machine learning or data mining algorithms. As such, those approaches can be computationally more complex and more difficult to implement. They attempt to learn the prediction model by identifying patterns using the training data in the offline phase, and then they utilize the prediction model to produce predictions for ongoing service invocations during the online phase.

A *matrix factorization* model [62] is based on *latent features*. Namely, it tries to describe each user and each item as a vector of a certain number of factors (e.g. 20) which are learned from the past interaction data. These factors can capture the more subtle user/item properties which are often difficult or even impossible to describe in words. It first appeared in 2006 as a winning model on a large *Netflix prize competition* dataset [63].

Formally, let K be the chosen number of latent features used in the model. For each item i we want to learn a K-dimensional vector q_i whose elements measure the extent in which the item possesses the corresponding latent features. At the same time, for each user u we want to learn a K-dimensional vector p_u whose elements measure the "interest" or affinity of the user to the corresponding latent features. Then the rating estimate r_{ui} will be the scalar product $p_u^T q_i$.

An input to the model (a training set) $\mathscr{R}$ contains the known ratings R_{ui} for some pairs of user u and item i. The model learns vectors p_u and q_u for all users and items such that $p_u^T q_i \approx R_{ui}$. If the model is trained carefully and without overfitting, the same near-equality should hold on the testing set, i.e., on ratings which are unknown to the model during the learning phase.

Why is the model called "matrix factorization"? If the system contains U users and I items, if the user-item ratings are assembled in matrix R with dimensions $U \times I$, if the latent vectors p_u are assembled as rows in matrix P with dimensions $U \times K$, and latent vectors q_i as columns in matrix Q with dimensions $K \times I$, then it will hold

$$R \approx PQ. \tag{2.6}$$

Therefore, we are approximating R as a product of two much smaller matrices P and Q. Because of sparsity of known ratings, the matrix R is only partially known (and its full size would not fit into memory). Since K is small, the matricess P and Q can fit into memory and be used in the online phase to calculate scalar products $p_u^T q_i$ in order to estimate an unknown rating R_{ui}.

The required matrices P and Q can be learned in multiple ways. Often a stochastic gradient descent is used, with the idea of starting with random values, calculating training predictions and moving the parameters p_u and q_i in direction of the gradient of the error function in order to reduce the error, i.e. to reduce the difference $R_{ui} - p_u^T q_i$ [62]. The error function is often quadratic and contains regularization:

$$L_{u,i} = (R_{ui} - p_u^T q_i)^2 + \lambda(||p_u||^2 + ||q_i||^2). \tag{2.7}$$

A more recent model *element-wise Alternating Least Squares* [64] learns P and Q in a different way. It treats P as fixed while analytically computing the optimal Q (with respect to the known ratings), then fixes Q while computing the optimal P, and so on, repeating the alternation until convergence. If carefully implemented, its time complexity is $O((U+I))K^2 + |\mathscr{R}|K)$ where $|\mathscr{R}|$ is the number of known user-item interactions, U the number of users, and I the number of items. The pseudocode is given below, where L ("loss") is a cumulative square error function. Many details are omitted for the case of simplicity, and an interested reader can find them in [64].

ALGORITHM 1: eALS method of matrix factorization

Input : ratings R, latent dimension K, regularization factor λ, item popularities c
Output: latent factor matrices P and Q
Randomly initialize P and Q;
while *no convergence* **do**
  Fix Q;
  for *all* $u = 1,\ldots,U$, $f = 1,\ldots,K$ **do**
    $p_{uf} :=$ solution of the equation $\frac{\partial L}{\partial p_{uf}} = 0$;
  end
  Fix P;
  for *all* $i = 1,\ldots,I$, $f = 1,\ldots,K$ **do**
    $q_{if} :=$ solution of the equation $\frac{\partial L}{\partial q_{if}} = 0$;
  end
end

In the context of QoS prediction, the researchers have recently proposed various prediction models that can be categorized and labeled as model-based CF. For instance, the researchers have proposed a QoS prediction approach for grid service composition based on Bayesian network [65]. Yu *et al.* have proposed a trace norm regularized matrix factorization based approach

[66]. Another approach based on neighborhood integrated matrix factorization is presented in [67]. A context-aware reliability prediction (CARP) approach uses invocation context-aware matrix factorization to alleviate the problem of data sparsity and produce more accurate predictions [68]. To address the *cold start* problem (prediction for users/services with no past invocation data), the researchers have introduced a novel location-aware matrix factorization prediction approach [69], which is also very similar to the research work presented in [70]. An interesting approach based on time series reliability prediction using artificial neural networks has been proposed recently [71]. An approach based on adaptive dynamic programming with fuzzy neural networks has been proposed in [72]. To enhance the availability of web services, the researchers have proposed an adaptive prediction framework based on fuzzy sensor web [73].

It should be noted that model-based CF prediction approaches usually require a fresh execution of offline learning phase in order to adapt to major changes in the environment, which is their main drawback in the context of web services reliability prediction.

2.1.3 Hybrid Collaborative Filtering

The hybrid CF approaches utilize some additional domain-specific data which describe the internals (e.g., structure, organization, or properties of the system) in order to produce more accurate predictions. These approaches can be very effective in alleviating disadvantages of memory-based CF such as data sparsity or cold start issues. Still, it appears to be quite challenging to obtain internal information in practice. Sometimes it requires further reorganization and restructuring of the existing system by introducing some new standards and conventions.

In such a way, the researchers have combined classical memory-based CF with some additional content-based features in their approach to produce more accurate services recommendations [74], [75]. Lee *et al.* [76] propose a novel approach that assumes existence of intelligent agents which constantly monitor the conditions in the environment/system. They utilize the information gained from agents to run matrix factorization considering dynamic invocation context for services reliability prediction. Deng *et al.* have proposed a social network based prediction approach which assumes that service users form a social network, which is rarely the case [77].

2.1.4 Shortcomings of Previous Approaches

Although the proposed approaches achieve remarkable performance, they demonstrate several serious disadvantages. The main disadvantage of the proposed approaches is related to their time performance. Having large number of users and services in real-world service oriented systems, those approaches simply do not produce predictions fast enough. Consider nowa-

days very popular products that offer *software as a service* (SaaS) for their clients [78], [79]. Those products offer various services located worldwide in the cloud and having different performance characteristics. A particular client may be interested in a service composition whose nonfunctional properties need to be optimized. With such requirements, a client needs instant information about the reliability of each atomic service so she can optimize her application regarding non-functional properties. To facilitate such requirements, the prediction model needs to produce relatively accurate predictions in real-time. Although very accurate in their predictions, the existing approaches have a limited support for real-time computation of predictions when the number of entities in the system is high.

The collabofative filtering based approaches achieve accurate predictions in domains where collected ratings are relatively stable for a longer period (i.e. movies or books ratings). On the other hand, web services are deployed on the Internet, which is a highly dynamic enviroment where actual data is prone to frequent changes. These frequent changes make the QoS prediction task even more difficult since the data collected in the past could be obsolete. In order to achieve accurate predictions, the model needs to automatically adapt to frequent changes in the environment. The approaches proposed in [25], [80], [81] handle situations in which fast and accurate predictions are needed. In that work, all three different groups of parameters that may impact the QoS are considered: user-specific, service-specific, and environment-specific parameters. However, all those approaches share the same disadvantage which is the inability to automatically adapt to eventual changes in the environment during the online prediction phase. In case the environment significantly changes, all these approaches require a fresh execution of the offline precompute phase.

2.1.5 Data Quality in CF Datasets

CF models are evaluated on various datasets, and results of a single model across different datasets are highly variable, since datasets typically exhibit different properties and yield different prediction accuracies. This introduces the need to measure the quality of a dataset which would be an indication of its prediction difficulty. This measure can be used as a reference point when proposing and evaluating a prediction model on a certain dataset. Developing a prediction model is expensive and time consuming, and quick heuristic indicators of potential prediction accuracy on a dataset can be very valuable; for example, when deciding which additional data to take into account.

For instance, an e-commerce provider can use this approach when designing the CF process. Specifically, the item vector can be structured in a way to take into account newly collected data, e.g. in case a new type of rating is introduced. A quality measure can then be used to evaluate how the new dataset would behave if collaborative filtering was to be extended to these new attributes. In effect, the measure would estimate the level of maturity that the available

collected data would give to the newly designed CF process. In other words, knowing the proposed measure of a dataset could help us predict whether high prediction accuracy can be expected.

Other potential uses of a data quality measure can be found for educational and scientific purposes. It is often the case when examining a new approach that the required specific dataset is not available. In such cases, the researchers often rely on synthetically generated data. The proposed measure could be used to assess if the generated dataset resembles a realistic CF dataset (is it suitable for making predictions) or its values behave as if they were random, and if so, to which degree. Consequently, the measure can be used to further fine-tune the artificial dataset generators.

Data quality has multiple aspects, discussed in e.g. [82] and [83]. *Natural noise*, based on user inconsistency, was discussed in [84], [85], and showed to constrain the prediction ability beyond some maximal ("magic barrier") value [86]. Another aspect is *missing data*, which is related to dataset *sparsity*, specifically discussed in [87]. However, sparsity is information about the amount of available data, not about its properties; for example, it does not capture whether there are many (or any) users with similar preferences. Marlin et al. [88] have analyzed *which* data is missing, showing and utilizing non-randomness of the missing data distribution.

However, to make reasonable predictions, all CF models explicitly or implicitly rely on the assumption of *similarity* between (some) users and/or similarity between (some) items. Prediction accuracy, and therefore the reliability of a recommender system, depends on the degree of correlation of datasets' values, which is opposed to randomness and noise. To the best of our knowledge, there are not many global correlation-based measures of a dataset quality, and existing ones include calculation of many user-user or item-item similarities, which is not more efficient then calculating all predictions. Namely, user-user and item-item correlations are usually calculated using *Pearson Correlation* (PCC) and less often using *vector cosine*, *Spearman rank* or *Kendall's* τ *correlation* [89]. However, these measures are defined on the level on individual users/items only. Computing all pairwise similarities to count the number of *similar pairs* (e.g., with $PCC > 0.5$) is infeasible for large datasets. Recently, [90] measured sparsity and *redundancy* for setting the sampling levels in a CF model. Redundancy was defined using the number of users, items, and ratings, as well as the average pairwise similarity of users having at least one jointly rated item. As it is the case for the number of similar pairs, this measure is computationally demanding as it potentially calculates $O(N^2)$ user-user similarities, each of which takes at least $O(\text{ratings per user})$ time, which gives a very impractical time complexity for datasets with e.g. 10^5 users.

Therefore, as a side contribution of this dissertation, in the next chapter we will define two novel heuristic measures of a CF dataset quality, designed to quickly estimate the amount of global user correlation and item correlation. Using comprehensive experiments on synthetic

and real-world data, we will show that the proposed measures satisfy several desirable properties such as correlating with the the accuracy of standard prediction models. We will derive several applicable principles from the results.

2.2 Service Selection

In a composite (multi-task) application where multiple services are invoked for a user, the user's QoS requirements are usually not observed for individual invocations but for the whole composition. Therefore, QoS properties are calculated over the whole workflow. These requirements can be part of a Service Level Agreement (SLA). The SLA usually considers multiple aspects of QoS (e.g. availability, reliability, and response time) and can be violated even when all aspects except one are satisfied. The issue becomes more complex when various compositional possibilities (Sequence, Parallel, Branch, Loop) have to be considered [6]. In a system with many users (a multi-tenant system) with minimal requirements on various QoS properties such as response time, reliability, and price [10, 11], there is need for a robust method which will globally optimize the service selection. Namely, the output of the method should be an answer to the following question: *which service instances will be invoked by which users for which tasks?*

The simplest idea that first comes to mind is a greedy approach: select a service with best QoS for each user. However, there are several problems with such an idea. First, QoS has many properties, so the meaning of "best QoS" is unclear: there might be (and often are) possible tradeoffs. For example, a service with higher reliability can be more costly; a service with the fastest response time can be unreliable. Another issue is that, in a composite case, we are not concerned only about the QoS of atomic services. What ultimately matters is QoS of the whole composition, not just of a single task. Furthermore, as mentioned in the Introduction, composition can be non-deterministic and probabilities and expected values might be taken into account. If there are multiple potential execution paths for a composition (because of probabilistic branching), it is difficult to locally estimate whether any particular user-service choice leads to a good global solution. But most importantly, a service has a throughput limit: the maximal number of requests (users) it can handle in a given time frame. Therefore, even for non-composite applications (containing only a single task), greedy algorithms will not work: selecting the "best" service instance for each request (by any criterion) will generally break the throughput limit of the most "popular" services by asigning them too many requests to handle. Therefore, more complex approaches need to be considered.

There has been a lot of work on QoS-based service selection, covering different aspects and problem variations. Most of them assume the QoS values are already known or predicted with reasonable accuracy, focusing on the constrained selection phase instead. The following

sections briefly introduce the most influential selection strategies.

2.2.1 Single-User Case

Multi-criteria service selection is difficult even when focusing on a *single user's* composite service pipeline. To see why, consider a very simple case where there are two tasks in the user's composition (A and B) with services a_1, a_2 handling A, and b_1, b_2 handling B. Assume that a_1 and b_1 have better reliability than a_2 and b_2, but on the other hand, a_2 and b_2 have a better response time than a_1 and b_1. Then any of the choices (a_1, b_1), (a_1, b_2), (a_2, b_1), (a_2, b_2) for service selection can be the best, depending on the actual QoS values of services and QoS requirements of the user. By increasing the number of tasks and available services, the number of possible selections grows exponentially.

To name a few approaches, [91] and [92] employ iterative multi-attribute combinatorial auction between services providers, while [93] seeks to improve this approach using an incentive mechanism. BigData-space service selection was tackled by [94], taking into account both qualitative and quantitative user QoS preference with the service trust. Recently, [95] studied the problem from a general Pareto-optimal angle in order to reduce search space in service composition. A polynomial time approximation for Pareto optimality was done by [96]. Similar work was done by [97], trying to overcome the limitations of Pareto optimality. Approach by [98] selects representative services adaptively: it divides the value range of each quality attribute into sub-ranges, considering the QoS values of a sub-range as local constraints, and selects the appropriate services from the divided QoS ranges so they can maximize the utility of a user task. Some recent papers such as [99, 100] took into account the possible correlation of services' QoS attributes in a composition, while the probabilistic QoS values are considered in [101]. Recently, [102] developed a mobility-aware approach.

To find the *optimal* solution for a single user's service selection with composite service pipeline, [103] reduced the problem to a Multiple Choice Knapsack Problem (MCKP) which is NP-hard, but solvable in pseudo-polynomial time. However, they considered only a single execution path in a composite pipeline. Taking into account a directed graph which contains all execution paths in a dynamic pipeline, the problem becomes a Multiple Choice Multiple Dimension Knapsack (MMKP), which is more complex and is usually solved by mixed integer programming (MIP) [104, 105]. Others tried to improve on this exponential solution by reducing the search space: papers such as [14, 106] decompose each global QoS constraint c_0 into a set of local constraints $c_1, \ldots, c_k$, one for each task in the composite pipeline. The most influential approach seems to be [107], which identifies representative "skyline" services to improve the efficiency along with determining local QoS quality levels.

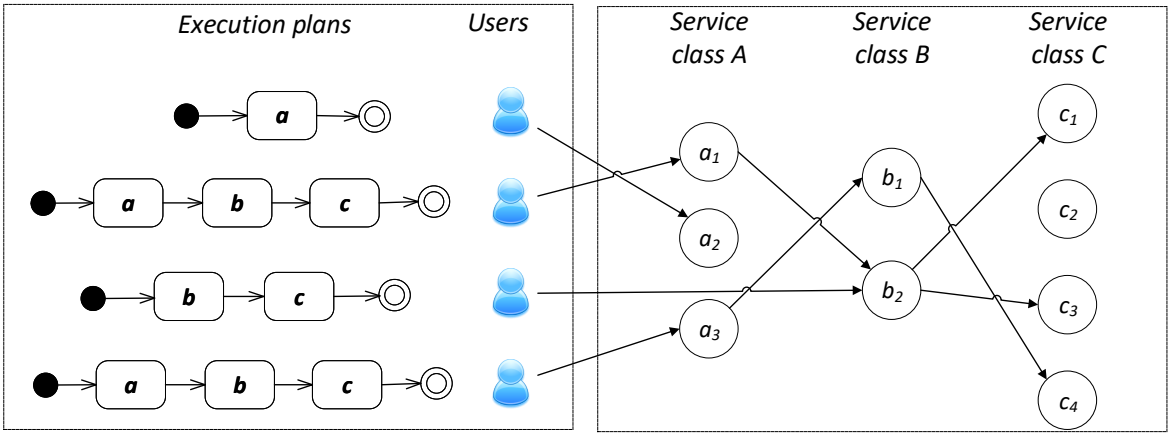


Figure 2.4: Multi-user service selection

2.2.2 Multi-User Case

In a multi-user scenario (Fig. 2.4), services' throughputs are taken into account, making the problem more complex than a sum of single-user problems, since the union of the solutions for single-user problems can overload some services. Again, the problem can be solved by mixed integer programming (MIP), which was done by [12, 15, 16]. To reduce the exponentially large MIP search space, [108] formed a reduced service set by dismissing the services dominated by another service, while [27] reduced the single-task problem size by clustering the services.

A polynomial, non-MIP approach on multi-user service selection was done by [26]. It considers a single-task problem and reduces it to an instance of the assignment problem. Namely, to account for throughput limits (THR_i = max. number of reqests to service i), this approach creates THR_i "virtual services" for each service i, each being able to handle a single request. For example, instead of a single service which can process 20 requests, we imagine 20 services which can process one request each. Using this approach, the problem is reduced to finding a one-to-one weighted matching between users and "virtual services" with minimal total cost. This becomes a combinatorial problem, known as the *assignment problem* (AP), illustrated in Fig. 2.5. In the mentioned paper, it is first tackled by a greedy algorithm (for efficiency pur-

	QoS	User 1	User 2	User 3	User 4
	Service 1				X
	Service 2				
Processing capacity (THR = 4)	*Service 3*	X			
	Service 3			X	
	Service 3				
	Service 3		X		

Figure 2.5: Service selection seen as assignment problem

poses), and then if the greedy algorithm does not find a satisfactory solution, the standard Hungarian (Kuhn-Munkres) algorithm for AP is applied [109]. The downside of this approach is that multiplying the number of entities in the algorithm (because of "virtual services") degrades the efficiency when throughputs are high.

The same idea is applied to a multi-task model by [106] using heuristic decomposition of global QoS constraints into single-task constraints. However, this model depends on the simplistic assumption that QoS values depend only on a service and not on a user. Similar drawback is present in recent MIP-based solutions for multi-user multi-task service selection [12, 15, 16]. In particular, the MIP search space in [16] is reduced using service clustering based on their user-independent QoS values.

2.2.3 Shortcomings of Previous Approaches

Instead of assuming that a service instance has the same QoS for all users, our work focuses on a more realistic scenario where each QoS value depends on a *user-service pair*. This assumption was present in our previous discussion of QoS prediction models and supported by the fact that QoS prediction models usually assume personalized QoS [110]. On the other hand, service selection models (named in the previous two sections) tend to simplify this, because personalized QoS increases the dimensionality of the problem input (QoS values form a matrix instead of an array) which might lead to higher computational complexities, depending on the chosen algorithm, or invalidate the approaches to reduce the search space by ranking or clustering the services. However, personalized QoS increases the accuracy of the obtained solution, since user-specific parameters (such as poor bandwidth) have a significant impact on the actual received QoS. This was the motivation for our choice of user-dependent (personalized) QoS.

Another limitation of previous works is their handling of service *compositions*, where each user's execution plan allows many potential actual execution paths (sequences of performed tasks) because of branching and loops. A compositional quality model proposed in [15] and subsequently used in [12, 16] calculates the aggregated QoS either too simplistically, by summing/multiplying QoS of all potential services (in case of *cost* and *availability*), or inefficiently, by taking max/min of all possible execution paths (in case of *response time* and *throughput*). The latter approach is also unsuitable for integer programming where the constraints are linear. In general, the QoS aggregation of such an approach is based on the worst-case execution path, which may have low probability, leading to poor accuracy in practice.

The work which will be presented in this dissertation models the compositional QoS in a more accurate and efficient way, focusing on *expected* QoS instead of a worst-case one. It overcomes the timing limitations of MIP by avoiding it completely, opting for a faster but potentially suboptimal solution instead. Summarized comparison with most related references discussed above is given in Table 2.1. This table shows some of the key aspects in which our

Table 2.1: Comparison of the most related papers (recent or influential)

	year	multi-user	multi-task (service composition)	user-dependent QoS	main approach	worst-case complexity
He et al. [12]	2015	✓	✓	×	mixed integer programming (MIP)	exponential
Alrifai et al. [107]	2010	×	✓	×	enhanced MIP	exponential
He et al. [15]	2012	✓	✓	×	greedy + MIP	exponential
Wang et al. [16]	2015	✓	✓	×	clustering + MIP	exponential
Wang and Cheng [27]	2016	✓	×	×	clustering + MIP	exponential
Wang et al. [26]	2014	✓	×	✓	assignment problem	polynomial
our work	2019	✓	✓	✓	transportation problem	polynomial

work tries to overcome the limitations of the representative previous works, also showing the difference in the main approach. The limitations shown in the table (and discussed above) are the exponentially large search space, the failure to consider multi-task service compositions (or realistic compositional QoS), or the assumption of QoS which does not depend on a user.

Chapter 3

Real-time Adaptive QoS Prediction Using Approximate Matrix Multiplication

The main motivation for the *real-time adaptive* QoS prediction model is a fact that much of recent work focuses on real-time applications in heterogeneous systems [111] and real-time workflows in various systems such as cyber-physical cloud systems [112]. Many of these systems depend on adaptive behavior [113]. Adaptability is especially crucial in highly dynamic, changing environments such as service-oriented systems, where predictions of QoS should be up to date. Therefore, this requirement is in focus of the present work. We propose a model that can be used as a real-time support for atomic service selection while constructing composite applications. The model was published in [17].

The proposed prediction model is based on collaborative filtering (CF). Before describing the model itself, in Sect. 3.1 we will present our related contribution: a quality measure for a CF dataset (published in [19]). Then, Sect. 3.2 describes the high-level overview of the proposed QoS prediction model, Sect. 3.3 introduces the basic notation used in the model description, while the later sections describe the three major phases of the proposed model: the *precomputation phase* (Sect. 3.4), the *prediction phase* (Sect. 3.5), and the *update phase* (Sect. 3.6).

3.1 Global Correlation Measures for a CF Dataset

As suggested in Sect. 2.1.5, we attempt to estimate the prediction ability of a CF dataset (such as a user-service QoS prediction dataset) based on user-user and item-item similarities, assuming that in a "high-quality" dataset there are groups of highly correlated users (we can predict the behavior of one based on the others) and groups of highly correlated items (we can predict the ratings of one based on the others). As these are, in general, two distinct properties, we are actually proposing two measures: one for the user correlation degree and one for the item cor-

relation degree. As a consequence, we can compare the two measures when deciding between the user-based and the item-based prediction model, as their prediction ability can differ. This is useful only if the measures can be computed efficiently, which is the case for the proposed measures.

Since high similarities among users and items imply better prediction ability (not only when using PCC-based CF approaches), we attempt to estimate the dataset quality using real numbers GUCM (Global User Correlation Measure) and GICM (Global Item Correlation Measure) with the following properties:

- GUCM, GICM$\in [0,1]$ with a low value corresponding to low similarities and a high value corresponding to high similarities.
- GUCM and GICM correlate with prediction accuracy, i.e., higher GUCM/GICM implies higher Top-N precision rates.
- GUCM and GICM are close to 0 on artificial random datasets (with random real-valued ratings).
- GUCM is close to 1 on a dataset where all users are like-minded. Analogously for GICM and items.
- GUCM negatively correlates with the number of natural clusters of similar users, i.e., with the variability of user types. Analogously for GICM and items.

The following subsection gives definitions of GUCM and GICM with the algorithm for their calculation. The properties described above are confirmed in the experimental results, Sect. 6.1.

3.1.1 Calculating GUCM and GICM

As the definitions and the calculation algorithms for GUCM and GICM are completely analogous (symmetrical), we will describe GUCM in detail; GICM is then easily defined and calculated by switching the roles of "users" and "items" (for example, by transposing the user-item matrix).

The basic idea is that if there are groups of like-minded users, this will reflect in their ratings of a single item: the ratings of this item will not be uniformly distributed, but will be divisible into groups of mutually close ratings. We will therefore focus on the ratings distribution at item level and estimate its divergence from the uniform distribution.

Namely, looking at a single item i, we estimate the degree of users' *agreement* on the item rating as a real number in $[0,1]$ as follows. We first compute the number of *close pairs*, denoted by CP_i, which is the number of user pairs whose normalized ratings (with the average subtracted) differ by less than a certain threshold T:

$$CP_i = |\{\{u,v\} \subseteq U_i : |(r_{u,i} - \bar{r}_u) - (r_{v,i} - \bar{r}_v)| < T\}|, \tag{3.1}$$

where U_i is the set of all users who have rated item i. If their number is $|U_i| = K_i$, the *percentage of close pairs*, denoted by p_i, equals the number of close pairs divided by the number of all considered pairs:

$$p_i = \frac{CP_i}{K_i(K_i-1)/2}. \tag{3.2}$$

This represents the probability that a random pair of users have close ratings on item i. Since some close ratings naturally appear in random datasets as well, and since the threshold value T has a direct impact on p_i, we will "correct" p_i by subtracting the expected value of the same probability for the random distribution, i.e., the probability that two uniformly distributed random ratings from the same range differ by less than T. This probability equals

$$q = \frac{T(2D-T)}{D^2}, \tag{3.3}$$

where D is the given range, i.e., the difference between the maximum and the minimum value in the dataset: $D = r_{max} - r_{min}$. The proof of (3.3) is given in the Appendix.

We therefore define the *user agreement degree* on item i, denoted by UAD_i, as $p_i - q$, i.e., the difference between the percentage of close pairs and the expected percentage for the random dataset. In the unlikely case that $p_i - q < 0$, we set the value to zero:

$$UAD_i = \max\{p_i - q, 0\} \in [0,1]. \tag{3.4}$$

Finally, GUCM is defined as the weighted average of user agreements across items, with the weight w_i equal to K_i (the amount of data used in calculating UAD_i):

$$GUCM = \frac{\sum_{i=1}^{M} K_i \cdot UAD_i}{\sum_{i=1}^{M} K_i} \in [0,1], \tag{3.5}$$

where M is the total number of items. We propose to set the aforementioned "closeness" threshold T_{user} to be the standard deviation of the user averages $\bar{r}_u$, because a small standard deviation implies similar subjective scales and, therefore, smaller differences between close pairs. We have also tried other threshold values and found that the proposed choice gives meaningful results in practice.

In order to compute CP_i efficiently, we first sort the known ratings of item i as an increasing sequence $a_1, \ldots, a_{K_i}$. We want to find the number of pairs (k,l) such that $k < l$ and $a_l - a_k < T$. For a fixed k, we find the largest possible l still within the threshold, i.e., the largest a_l less than $a_k + T$, and then the number of pairs for this fixed k equals $l - k$ because all pairs $(k,k+1),(k,k+2),\ldots,(k,l)$ are close. When moving to the next k (i.e., increasing k by one), we update l, but it obviously increases or stays the same. This means that, after initial sorting in $O(K_i \log K_i)$, the time complexity of calculating CP_i is $O(K_i)$ as we only move the indices k

and l from the beginning to the end of the sorted ratings sequence.

The complete GUCM algorithm is given in Alg. 2 and computing GICM is entirely symmetrical (users $\longleftrightarrow$ items).

ALGORITHM 2: The algorithm for calculating GUCM

Data: User-item ratings $r_{u,i}$ ($u \in \{1,\ldots,N\}$, $i \in \{1,\ldots,M\}$)
$D := r_{max} - r_{min}$
for $u = 1$ *to N independently* **do**
  $\bar{r}_u := average\{r_{u,i} : u \text{ has rated } i\}$
end
$T := standardDeviation\{\bar{r}_u : u = 1,\ldots,N\}$
$q := T(2D - T)/D^2$
for $i = 1$ *to M independently* **do**
  $K_i := |\{u : u \text{ has rated } i\}|$
  $(a_1, a_2, \ldots, a_{K_i}) := sorted(\{r_{u,i} : u \text{ has rated } i\})$
  $CP_i := 0$
  $l := 1$
  for $k = 1$ *to* K_i **do**
    while $l < K_i$ *and* $a_{l+1} - a_k < T$ **do**
      $l := l + 1$
    end
    $CP_i := CP_i + l - k$
  end
  $p_i := CP_i/(K_i(K_i - 1)/2)$
  $UAD_i := \max\{p_i - q, 0\}$
end
Return $GUCM := (\sum_{i=1}^{M} K_i \cdot UAD_i)/(\sum_{i=1}^{M} K_i)$

Since calculations of UAD_i are mutually independent (as well as calculations of $\bar{r}_u$), they can be computed concurrently.

The properties of GCM are evaluated in the experiments (Chapter 6). In further study, the proposed measures could be refined to take into account semantic domain knowledge [114], fuzziness [115], and multirelational networks [116].

3.2 QoS Prediction Model Overview

We now describe our adaptive QoS prediction model for accurate predictions in real-time [17]. We assume a system with M users and N services. A high-level overview of the proposed model is depicted in Fig. 3.1. As can be seen in the figure, the model consists of the following phases:

- **The precompute phase (1) :** calculates approximate user-user and service-service similarities, as well as lists of similar users/services for each user/service. Its time complexity is between $O(M^2 \log M + N^2 \log N)$ and $O(M^2 N + N^2 M)$ and depends on the chosen accuracy of approximation.

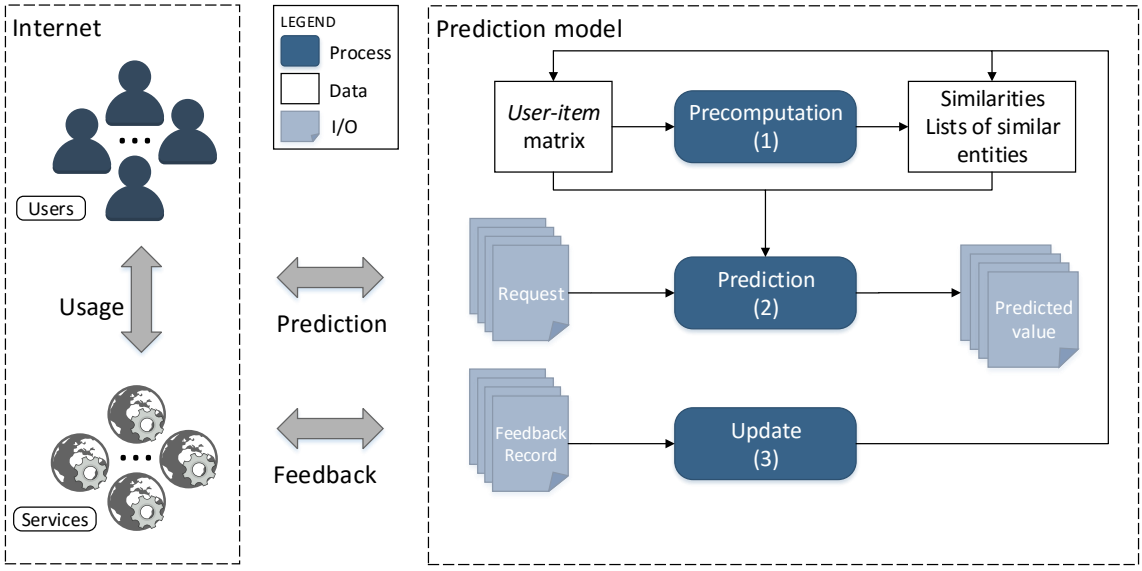


Figure 3.1: Prediction model high-level overview

- **The prediction phase (2) :** calculates the *prediction in point*, i.e., estimates the QoS value for user u on service i. It combines the user-based and service-based prediction, taking into account the K_1 users most similar to u and K_2 services most similar to i. Apart from the known QoS values, it also takes into account the support of each value, which is the number of observed invocations used to calculate this value. Another advantage over similar predictions is that it takes only $O((K_1+K_2)/d)$ time, where d is the density of the observed user-item matrix values.
- **The update phase (3) :** when a new record of QoS for user u on service i arrives, the values used for further predictions are updated accordingly, in $O(M+N)$ time. If the environment changes over time, more recent records have greater weight, which enhances the predictions. This makes the model adaptive.

3.3 Basic Notation

Throughout the rest of this chapter we will use the following notation.

- $u, v \in \{1,\ldots,M\}$ denote users, and $i, j \in \{1,\ldots,N\}$ denote services.
- $r_{u,i} \in [0, \text{inf}\rangle$ is the QoS value (e.g. reliability) of service i observed by user u.
- $n_{u,i} \in [0, \text{inf}\rangle$ is the number of observed invocations (records) of service i by user u in the past invocation sample, used in computing the known value $r_{u,i}$ (which is usually the average value of these records).
- $\bar{r}_u$ is the average QoS value of user u on different services observed by u.
- $\bar{r}_i$ is the average QoS value of service i for different users who observed it.

- $sim_{u,v} \in [-1,1]$ is the similarity between users u and v.
- $sim_{i,j} \in [-1,1]$ is the similarity between services i and j.
- $I_u \subseteq \{1,\ldots,N\}$ is the set of all services invoked by user u.
- $I_i \subseteq \{1,\ldots,M\}$ is the set of all users who invoked service i.
- $I_{u,v} \subseteq \{1,\ldots,N\}$ is the set of all services invoked by both users u and v.
- $I_{i,j} \subseteq \{1,\ldots,M\}$ is the set of all users who invoked both services i and j.

3.4 The Precomputing Phase

This section describes the precomputing phase. Since the prediction formula (given in Sect. 3.5) requires finding the most similar items to a given item, as well as their corresponding similarities and average QoS values ($\bar{r}$), we need to precompute them in order to have a fast prediction. The first part describes how precomputing similarities can be reduced to matrix multiplication, the second part discusses the matrix multiplication method, and the third part describes the rest of the precomputing phase.

3.4.1 Precomputing the Similarities

Let us focus on the user similarities first. According to the standard Pearson Correlation formula, the similarity of users u and v is calculated as

$$sim_{u,v} = \frac{\sum_{i \in I_{u,v}} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{u,v}} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{u,v}} (r_{v,i} - \bar{r}_v)^2}}. \tag{3.6}$$

Clearly, we can compute the similarity for a single pair in the $O(N)$ time complexity, which would give an $O(M^2N)$ time complexity for computing the similarities for all pairs of users. In order to do it faster, we break down the similarity formula into three parts:

$$sim_{u,v} = \frac{U_1[u,v]}{\sqrt{U_2[u,v]} \sqrt{U_3[u,v]}}, \tag{3.7}$$

where

$$U_1[u,v] = \sum_{i \in I_{u,v}} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v), \tag{3.8}$$

$$U_2[u,v] = \sum_{i \in I_{u,v}} (r_{u,i} - \bar{r}_u)^2, \tag{3.9}$$

$$U_3[u,v] = \sum_{i \in I_{u,v}} (r_{v,i} - \bar{r}_v)^2. \tag{3.10}$$

The basic idea is to see U_1, U_2, U_3 as $M \times M$ matrices which can be calculated by multiplying other matrices. For example, it looks like $U_1[u,v]$ is a product of the row-vector with values $r_{u,i} - \bar{r}_u$ and the column-vector with values $r_{v,i} - \bar{r}_v$, but the $i \in I_{u,v}$ condition makes things a little more complicated.

To actualize the idea, we create three auxiliary $M \times N$ matrices: B, U, and U', defined as follows:

$$B_{u,i} := \begin{cases} 1, & \text{if user } u \text{ has invoked service } i, \\ 0, & \text{otherwise,} \end{cases} \tag{3.11}$$

$$U_{u,i} := B_{u,i}(r_{u,i} - \bar{r}_u), \tag{3.12}$$

$$U'_{u,i} := B_{u,i}(r_{u,i} - \bar{r}_u)^2. \tag{3.13}$$

Now we can calculate U_1, U_2, and U_3 using the following theorem.

Theorem 1. *It holds that*

$$U_1 = UU^T, \tag{3.14}$$

$$U_2 = U'B^T, \tag{3.15}$$

$$U_3 = U_2^T. \tag{3.16}$$

Proof. From definition of B it follows that

$$i \in I_{u,v} \iff B_{u,i}B_{v,i} = 1. \tag{3.17}$$

Now we have

$$\begin{aligned} U_1[u,v] &= \sum_{i \in I_{u,v}} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v) \\ &= \sum_{i=1}^{N} B_{u,i}B_{v,i}(r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v) \\ &= \sum_{i=1}^{N} U_{u,i}U_{v,i} \\ &= \sum_{i=1}^{N} U_{u,i}U_{i,v}^T, \end{aligned}$$

which proves (3.14). Similarly,

$$\begin{aligned} U_2[u,v] &= \sum_{i \in I_{u,v}} (r_{u,i} - \bar{r}_u)^2 \\ &= \sum_{i=1}^{N} B_{u,i} B_{v,i} (r_{u,i} - \bar{r}_u)^2 \\ &= \sum_{i=1}^{N} U'_{u,i} B_{v,i} \\ &= \sum_{i=1}^{N} U'_{u,i} B^T_{i,v}, \end{aligned}$$

which proves (3.15). To prove (3.16), notice that $U_2[u,v] = U_3[v,u]$ follows directly from definitions (3.9) and (3.10). □

The theorem implies that all similarities $sim_{u,v}$ can be calculated at once using two matrix multiplications and other, faster operations. The situation with service similarities is analogous:

$$sim_{i,j} = \frac{\sum_{u \in I_{i,j}} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in I_{i,j}} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{u \in I_{i,j}} (r_{u,j} - \bar{r}_j)^2}} \tag{3.18}$$

$$= \frac{S_1[i,j]}{\sqrt{S_2[i,j]} \sqrt{S_3[i,j]}}. \tag{3.19}$$

The auxiliary matrices for computing S_1, S_2, S_3 are B (defined above), S and S', all of dimensions $M \times N$:

$$S_{u,i} := B_{u,i}(r_{u,i} - \bar{r}_i), \tag{3.20}$$

$$S'_{u,i} := B_{u,i}(r_{u,i} - \bar{r}_i)^2. \tag{3.21}$$

Then, S_1, S_2, and S_3 can be calculated using the following theorem, which is analogous to Theorem 1.

Theorem 2. *It holds that*

$$S_1 = S^T S, \tag{3.22}$$

$$S_2 = S'^T B, \tag{3.23}$$

$$S_3 = S_2^T. \tag{3.24}$$

Proof. Analogous to the proof of Theorem 1. □

Apart from being intermediary results in precomputing the similarities, matrices U_1, U_2, U_3, S_1, S_2, S_3 should be kept in the case of a real-time adaptive model, since the only way to quickly update a similarity is to update each of its three components (see Sect. 3.6.2).

3.4.2 Matrix Multiplication

We now discuss the method of matrix multiplication. If we use a standard matrix multiplication algorithm, the time complexity (assuming $M = N$ for simplicity) is $O(N^3)$, which is no better than the naive precomputing without matrix multiplication. But there are faster algorithms. For example, Strassen matrix multiplication takes $O(N^{2.807355})$ time.*

Apart from the exact matrix multiplication algorithms, there are much faster algoritms which compute an *approximation* of the product AB for given matrices A and B. As proposed by researchers in [117], in order to estimate $(AB)_{ij}$, instead of multiplying the whole i-th row of A with the whole j-th column of B, we randomly pick S elements from i-th row of A and the corresponding S elements from j-th column of B. The sum of their products, appropriately scaled, is an approximation of $(AB)_{ij}$. In our model we use uniform sampling, i.e., the elements of a row/column are picked with equal probabilities each time. The pseudocode is given in Algorithm 3 and describes the multiplication of an $a \times b$ matrix with a $b \times c$ matrix.

ALGORITHM 3: The approximate matrix multiplication (source: [117], Sect. 5.) with uniform sampling.

Data: matrices A (size $a \times b$) and B (size $b \times c$)
for $i = 1, \ldots, a$, *and* $j = 1, \ldots, c$ *independently* **do**
  for $t = 1$ *to* S *independently* **do**
    $k := randomElement\{1, \ldots, b\}$
    $X_t^{ij} := \frac{b}{S} A_{ik} B_{kj}$
  end
  Return $\sum_{t=1}^{S} X_t^{ij}$ as the approximation to $(AB)_{ij}$.
end

The time complexity clearly depends on the sample size S and equals $O(ac \cdot S)$. In our model, the multiplications are done for $(a,b,c) = (M,N,M)$ and $(a,b,c) = (N,M,N)$, which gives the time complexity of $O(M^2 \cdot S + N^2 \cdot S) = O(S(M^2 + N^2))$. By choosing the constant S, we can control the speed and the accuracy of the computation. Greater S gives a more accurate, but slower approximation. The experimental results (Sect. 6.2) show that the approximate matrix multiplication is useful for our purposes.

Finally, we could utilize parallel computing for the exact or the approximate matrix multiplication, which would lead to even lower time complexities, but is beyond the scope of this

*There are matrix multiplication algorithms with even better asymptotic time complexity (i.e., about $O(N^{2.37})$ for the Coppersmith–Winograd algorithm), but they only provide advantage for matrices too large for current practical purposes.

dissertation.

3.4.3 Precomputing Lists of Similar Items

After all similarities have been computed, for each user u we prepare a list of all other users v such that $sim_{u,v} > 0$, and we sort it from the most similar to the least similar user. We denote this list by $SimilarUsers[u]$. We analogously create the list $SimilarServices[i]$ for each service i. These lists will be used for fast prediction (see the following section).

The total time complexity of creating these lists is $O(M^2 \log M + N^2 \log N)$. Namely, for each of M users we are sorting a list of at most M elements, so each sorting takes $O(M \log M)$ time; analogously for the services.

We also need to precompute the user averages $\bar{r}_u$ and the service averages $\bar{r}_i$, using the standard formulas:

$$\bar{r}_u = \frac{\sum_{i \in I_u} r_{u,i}}{|I_u|}, \quad \bar{r}_i = \frac{\sum_{u \in I_i} r_{u,i}}{|I_i|}. \tag{3.25}$$

3.5 Prediction in Point

Suppose that we are to predict the QoS value for user-service pair (u,i). We first calculate the user-based prediction and the service-based prediction.

The standard formula for user-based prediction (UPCC) [59] is

$$UPCC_{u,i} = \bar{r}_u + \frac{\sum_{v \in I_i} sim_{u,v}(r_{v,i} - \bar{r}_v)}{\sum_{v \in I_i} |sim_{u,v}|} \tag{3.26}$$

Our model uses a slightly different approach which gave better experimental results (described in Sect. 6.2), namely:

- The summation goes over the set of K_1 users with greatest positive similarity to u which have observed service i. Let $best(u, K_1, i)$ denote this set. These users are found by iterating the precomputed list $SimilarUsers[u]$, which is sorted by similarity, until we find K_1 users that have observed service i, or reach the end of the list. Depending on the list length, we need to check between 0 and M users; the complexity analysis is given in Sect. 6.2.7.
- The value $r_{v,i}$ is not only multiplied (weighted) with $sim_{u,v}$, but also with the *support* of this value, which is the number of observed invocations $n_{v,i}$, since the accuracy of the value increases with the number of records. Namely, weight of the value $r_{v,i}$ (for a given u) is computed as

$$w[r_{v,i}] := sim_{u,v}\, n_{v,i}. \tag{3.27}$$

- Scaling of $r_{v,i}$ is not done by subtracting $\bar{r}_v$ and then adding the weighted sum to $\bar{r}_u$.

Instead, scaling is done by multiplying $r_{v,i}$ with the ratio $\bar{r}_u/\bar{r}_v$.
Therefore, in our model, the user-based prediction (UP) is calculated as

$$UP_{u,i} = \frac{\sum_{v \in best(u,K_1,i)} w[r_{v,i}]\, r_{v,i}\, \dfrac{\bar{r}_u}{\bar{r}_v}}{\sum_{v \in best(u,K_1,i)} w[r_{v,i}]}. \tag{3.28}$$

The service-based prediction (IP) is calculated analogously:

$$IP_{u,i} = \frac{\sum_{j \in best(i,K_2,u)} w[r_{u,j}]\, r_{u,j}\, \dfrac{\bar{r}_i}{\bar{r}_j}}{\sum_{j \in best(i,K_2,u)} w[r_{u,j}]}, \tag{3.29}$$

where

$$w[r_{u,j}] := sim_{i,j}\, n_{u,j}. \tag{3.30}$$

After $UP_{u,i}$ and $IP_{u,i}$ have been calculated, we combine them into final prediction:

$$P_{u,i} = \lambda_1 UP_{u,i} + \lambda_2 IP_{u,i}, \tag{3.31}$$

where $\lambda_1, \lambda_2 \in [0,1]$, $\lambda_1 + \lambda_2 = 1$ are arbitrarily chosen constants which control the influence of the user-based and service-based predictions. This idea is called Hybrid prediction [40].

In some special cases, when the available data is very sparse or unreliable, instead of constant weights λ_1, λ_2 of UP and IP, we can make the weights depend on the current support of the predictions $UP_{u,i}$ and $IP_{u,i}$. For example, if the data used for calulating $UP_{u,i}$ (this data is $r_{v,i}$ for all $v \in best(u,K_1,i)$) is more numerous and/or more reliable than the data for $IP_{u,i}$, then $UP_{u,i}$ should have a larger influence on the prediction ($\lambda_1 > \lambda_2$). Since (3.27) and (3.30) are our measure of support, we can set

$$\lambda_1 = \sum_{v \in best(u,K_1,i)} w[r_{v,i}], \quad \lambda_2 = \sum_{j \in best(i,K_2,u)} w[r_{u,j}], \tag{3.32}$$

with the appropriate scaling to ensure that $\lambda_1 + \lambda_2 = 1$. This formula gives better results in Stochastic experiment (see Sect. 6.2).

If there are no users similar to u (i.e., the list $SimilarUsers[u]$ is empty), we define $UP_{u,i} = \bar{r}_u$ as the average QoS value of this user, and analogously, $IP_{u,i} = \bar{r}_i$ in case of no services similar to i. In case that u is a new user which has not previously experienced any QoS values (a cold start problem), the value of $UP_{u,i}$ is not calculated and $IP_{u,i}$ is used as prediction, and vice versa in case of a new service.

3.6 Handling Updates

We now describe the mechanisms by which the model adapts to dynamic environment with many updates. Suppose that a new record for user u and service i has arrived with the value of val, meaning that the recently measured QoS value for the corresponding user-service pair equals val. The adaptive model should update all values used for futher predictions that this record affects. This section describes how various values are updated. The first part deals with updating $r_{u,i}$ and $n_{u,i}$. The second part deals with updating the averages $\bar{r}_u$ and $\bar{r}_i$, as well as the similarities that have changed. The third part deals with updating the *SimilarUsers* and *SimilarServices* lists that have changed.

3.6.1 Updating $r_{u,i}$ and $n_{u,i}$

If $n_{u,i}$ is the number of previous records for this user-service pair, and $r_{u,i}$ the previous average value of these records, then the previous total value equals $n_{u,i}r_{u,i}$. The new total value is $n_{u,i}r_{u,i} + val$, so the new average value is $r_{u,i} := (n_{u,i}r_{u,i} + val)/(n_{u,i} + 1)$. Also, $n_{u,i}$ should be increased by one.

However, in case of environment that changes over time, we want to make new records contribute more to the average, i.e., their weight w will not necessarily be equal to 1, and will increase with each new update. We therefore have the general formulas:

$$r_{u,i} := \frac{n_{u,i}r_{u,i} + w \cdot val}{n_{u,i} + w}, \tag{3.33}$$

$$n_{u,i} := n_{u,i} + w, \tag{3.34}$$

$$w := w + \Delta w. \tag{3.35}$$

Initially, $w = 1$. The weight increment constant Δw can be arbitrarily set depending on how fast the environment changes. If the environment is not subject to significant change, we set $\Delta w = 0$, i.e., the weight of each record will be equal to 1.

3.6.2 Updating the Averages and the Similarities

We proceed to update the averages $\bar{r}_u$ and $\bar{r}_i$. This can be done by simply calculating them again with time complexity of $O(M+N)$ using (3.25), since we have to update the similarities as well, which will also take $O(M+N)$ time.

Namely, we have to update $sim_{u,v}$ for all users v, and $sim_{i,j}$ for all services j, which is a total of $M+N$ values. To update a single similarity in $O(1)$ time, we have to keep the similarity

components (U_1, U_2, U_3 or S_1, S_2, S_3) introduced in the precomputing phase (Sect. 3.4). Each of these components is a sum, and we need to change a single summand, which is easily done in $O(1)$ time by subtracting the previous value of the summand and adding the new value of the summand. Then we update the similarity according to the formula (3.7) or (3.19).

3.6.3 Updating Lists of Similar Items

Upon updating a similarity $sim_{x,y}$, we might need to update the lists $SimilarUsers[x]$ and $SimilarServices[y]$. Namely, if the similarity has changed from negative to positive, a new item is added to the list. If the similarity has changed from positive to negative, an item should be removed from the list. This does not affect the overall time complexity since a similarity rarely changes its sign. When it does, its value is usually close to zero, so adding/removing is done around the end of a sorted list, which does not take much time.

However, even if the similarity stays positive, over time it can change enough to make a list of similar items not sorted anymore. This is not a big issue since prediction depends only on the set of the top K appropriate items in a list, regardless of their exact order; also, the items with high similarity to a certain item (those with greatest influence on the prediction) are likely to stay in the top K set. We should be concerned about this issue only when the environment or the available data is subject to a quite significant change.

In that case, there are several ways to keep the lists sorted (or roughly sorted). One way is to keep an auxiliary matrix to quickly tell the position of an item x in the list of an item y; then it is easy to move x a few places to the left or to the right in the list when $sim_{x,y}$ is changed. The second way is to sort the lists $SimilarUsers[u]$ and $SimilarServices[i]$ when the update for the pair (u,i) arrives, which slightly increases the time complexity of the update step: from $O(M+N)$ to $O(M\log M + N\log N)$. The third way is to sort all the lists periodically, when we estimate that cumulative similarity changes have been significant or when the model is idle.

3.6.4 Adding a New User/Service

The model should initially allocate arrays with enough memory for potential new users and services; in other words, M and N can be larger than the actual number of users and services at that point. With that assumption, adding a new user (or a new service) simply amounts to handling new updates to previously unused elements of similarity arrays. This is simulated in the "Changing Stochastic" and "Scalability" experiments (Sect. 6.2.2) where no QoS values are initially known to the model – in effect, no users and services are present. The "addition" of a new user u is equivalent to receiving the first update of her QoS value for any service. Analogously, the "addition" of a new service i is equivalent to receiving the first update of its QoS value for any user.

Chapter 4

Compositional QoS Model

In this chapter we introduce and define two important tools (prerequisites) for our service selection method which will be presented in Chapter 5.

First, we describe a realistic QoS model for service compositions where the actual execution path is non-deterministic, i.e., not known in advance. For a given execution plan consisting of standard compositional structures (Sequence, Parallel, Conditional Branch, and Loop), based on probabilities, it estimates the expected number of service invocations per task, as well as the expected QoS. The QoS model is defined is Sect. 4.1 along with the required formulas for recursive calculation of expected QoS values.

Next, in Sect. 4.2 we propose a novel paradigm for estimating selection utility for a particular task in a composition, which measures the cost of a given user-service match with respect to her/his compositional QoS requirements. This concept is named *matching difficulty*. In other words, we define a heuristic cost of assigning a particular service instance for a particular task to a particular user. This cost is composition-aware, meaning that it takes into account other tasks in the user's service composition, as well as the corresponding service candidates and their QoS values. This cost will then be minimized in the proposed service selection algorithm.

4.1 Probabilistic Compositional QoS

The main idea of the proposed compositional QoS model is to work with *expected* QoS since, in general, different execution paths have different probabilities. The model takes a composition (execution plan) as input and calculates the expected number of invocations for each task, as well as the expected aggregated QoS values if actual services selected for the tasks are given. The given execution plan can be a complex composition containing all standard compositional structures: Sequence, Branch, Parallel, and Loop (see Fig. 4.1 for illustration). The proposed model is recursive: each execution plan is either an atomic task or consists of several execution plans composed by a standard compositional structure.

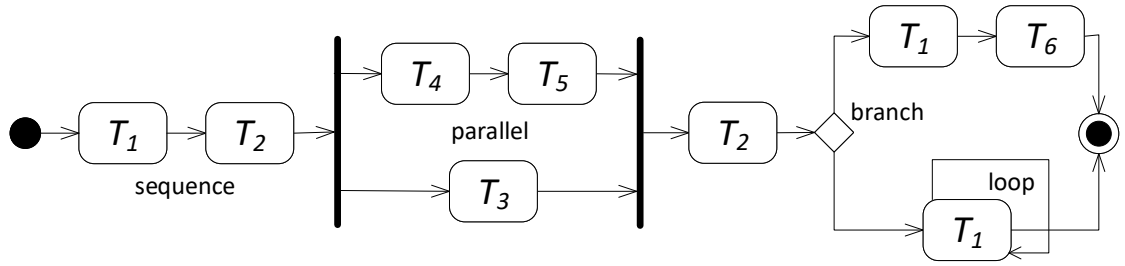


Figure 4.1: Example of a service composition with tasks $T_1, \ldots, T_6$

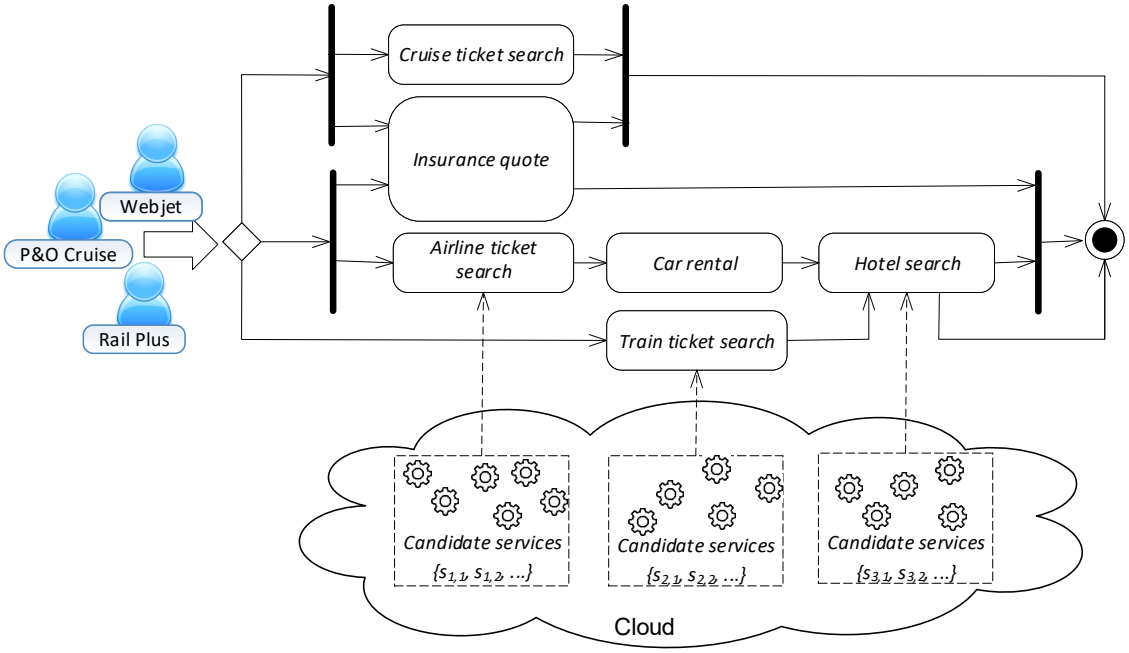


Figure 4.2: Composition example [12]: multi-tenant travel booking service-based system

Let us formalize the ideas from the previous paragraph by defining the expected number of invocations for each task in a composition and the expected compositional QoS. We assume that for each of the q considered QoS properties (such as response time, reliability, price...) there is an input matrix of size $|U| \times |I|$ describing the corresponding QoS values for each user-service pair. These values are either known or obtained by a QoS prediction model (such as the one presented in Chapter 3). Namely, let QoS_{ui}^{k} denotes the k-th QoS property for invocation of service $i \in I$ by user $u \in U$, where $k = 1, 2, \ldots, q$. (A complete list of notations used in this and the following chapter is given in Table 4.1.)

Definition 4.1.1 (Expected invocations)**.** *Let C ("composition") be an execution plan of user u, let $\{T_1, T_2, \ldots, T_n\}$ be the set of all tasks in the system, and let q be the number of observed QoS properties. Then $EI(C)$ denotes an n-dimensional vector such that $EI(C)_j$ is the expected*

Table 4.1: Notation list for Chapters 4 and 5

U	set of users
u	single user from U
I	set of services
n	number of tasks (service classes)
T_j	a functionality (task) performed by service class I_j ($j = 1,\ldots,n$)
I_j	set of functionally equivalent services (service class) corresponding to task T_j
i	single service from I or I_j
THR_i	throughput limit (max. throughput) for service i
q	number of considered QoS properties
QoS_{ui}^k	value of k-th QoS property for (u, i) pair
Req_u^k	requirement (min/max value) for obtained k-th QoS property for user u
x_{ui}	is service i selected for user u (0 or 1)
c_{ui}	utility cost of selecting service i for user u
C_u	execution plan (composition of tasks) for user u
$EI(C_u)_j$	expected number of invocations for task T_j in composition C_u
$EI(C_u)$	n-dimensional vector of values $EI(C_u)_j$ ($j = 1,\ldots,n$)
$QoS(C,s_1,...,s_n)_k$	expected value of k-th QoS property in composition C if services $s_1,\ldots,s_n$ are selected for tasks $T_1,\ldots,T_n$
$QoS(C_u)$	q-dimensional vector of expected $QoS(C_u,s_1,\ldots,s_n)_k$ values for $k = 1,\ldots,q$
p-choice$^k(u,I_j)$	service positioned at the p-th percentile when I_j is sorted by QoS_{ui}^k
$MD_j^k(i \to u)$	*matching difficulty* of selecting service $i \in I_j$ for user u with respect to k-th QoS property
U_j	set of users for which task T_j appears in their composition plan
TP_j	transportation problem for users in U_j and services in I_j
$c_{ui}(TP_j)$	utility cost of selecting service i for user u in TP_j
w_u^k	weight of k-th QoS property for user u
$\alpha(r)$	weight adjustment factor in r-th iteration of the algorithm
TP	transportation problem
VAM	Vogel Approximation Method for TP
TSM	Transportation Simplex Method for TP
SS-VAM	proposed service selection algorithm which uses VAM to solve TP
SS-TSM	proposed service selection algorithm which uses VAM+TSM to solve TP
AP	assignment problem (alternative approach)
MIP	mixed integer programming (alternative approach)

number of invocations of T_j in execution plan C.

For example, if the number of tasks in the composition is $n = 3$, and if we expect two invocations of T_1, one invocation of T_2, and 0.5 invocations of T_3, then $EI(C) = [2, 1, 0.5]$.

Definition 4.1.2 (Expected QoS)**.** *Assume that services $s_1, s_2, \ldots, s_n$ are given such that s_j is selected for task T_j (for all $j = 1, 2, \ldots, n$). Then $QoS(C, s_1, \ldots, s_n)$ denotes the q-dimensional vector, written as $QoS(C)$ for simplicity, such that $QoS(C)_k$ is the expected value of k-th QoS property in execution plan C.*

As an example, let us assume three QoS properties ($q = 3$): reliability, response time, and price. The expected compositional values of these properties depend on a particular service selection. If the selection of service candidates $s_1, \ldots, s_n$ for composition C leads to expected reliability of 0.99, expected response time of $0.1s$, and expected price of \$5, then $QoS(C) = QoS(C, s_1, \ldots, s_n) = [0.99, 0.1, 5]$.

We now give the recursive formulas for calculating $EI(C)$ and $QoS(C)$ based on the compositional structure. For the sake of simplicity, we unify the formulas for different QoS properties. For example, since availability/reliability is the only QoS property where aggregation is calculated by multiplication (since this property represents success probability), we work with logarithms of actual values to convert multiplication into addition.

- **Atomic task.** If C consists of a single task T_i, then:

$$EI(C)_i = 1, \text{ and } EI(C)_j = 0 \text{ for tasks } j \neq i; \tag{4.1}$$

$$QoS(C)_k = QoS^k_{us_i} \text{ for QoS properties } k = 1, \ldots, q. \tag{4.2}$$

- **Sequence** (execution one-by-one). If C is a sequential composition of execution plans $C_1, C_2, \ldots, C_m$, then

$$EI(C) = \sum_{j=1}^{m} EI(C_j), \tag{4.3}$$

$$QoS(C) = \sum_{j=1}^{m} QoS(C_j). \tag{4.4}$$

- **Branch** (execution of one out of several paths). If C is a composition which branches into execution plans $C_1, C_2, \ldots, C_m$ with respective probabilities $p_1, p_2, \ldots, p_m$ with $p_1 + \cdots + p_m = 1$, then

$$EI(C) = \sum_{j=1}^{m} p_j EI(C_j), \tag{4.5}$$

$$QoS(C) = \sum_{j=1}^{m} p_j QoS(C_j). \tag{4.6}$$

- **Parallel** (concurrent execution). Let C be a parallel composition of execution plans C_1, C_2, ..., C_m. Then,

$$EI(C) = \sum_{j=1}^{m} EI(C_j). \tag{4.7}$$

The formula for $QoS(C)$ in this case depends on the specific QoS property. In the case of e.g. *price*,

$$QoS(C) = \sum_{j=1}^{m} QoS(C_j). \tag{4.8}$$

In the case of *availability* or *reliability*, the values for parallel plans are multiplied (because all of them must succeed). Since we work with logarithms to convert multiplication into addition (as already mentioned), the formula is again 4.8.
However, in the case of *response time*,

$$QoS(C) = \max_{j=1}^{m} QoS(C_j), \tag{4.9}$$

since the total response time will correspond to the longest plan in the parallel structure.

- **Loop** (repetitious execution). If C is a loop in which execution plan C' is repeated m times, where m is the expected number of repetitions in case that the precise number of repetitions is unknown, then

$$EI(C) = m \cdot EI(C'), \tag{4.10}$$

$$QoS(C) = m \cdot QoS(C'). \tag{4.11}$$

If C' is repeated until some success condition is met, and if the probability of success in a single repetition is $p(C')$, then the number of repetitions m is geometrically distributed. Namely, it is the number of Bernoulli trials needed to get one success, and its expected value is known to be $E[m] = \frac{1}{p(C')}$. (For example, if C' is successful in 50% cases, then two repetitions until success are expected on average.) In this case, therefore,

$$EI(C) = E[m] \cdot EI(C') = \frac{EI(C')}{p(C')}, \tag{4.12}$$

$$QoS(C) = E[m] \cdot QoS(C') = \frac{QoS(C')}{p(C')}. \tag{4.13}$$

The probabilities used in the above formulas depend on the functional properties of the application ("what it does") and can be estimated empirically depending on the application. For instance, in the travel-agent SBS example (similar to Fig. 4.2), branching into flight search or train search can depend on the condition "travelling abroad or not?". A simple log statistics can

estimate the probability of such a condition.

In the proposed service selection algorithm, the vector $EI(C_u)$ for each user u is calculated initially, while vectors $QoS(C_u)$ are calculated when testing specific service selections in later steps. The algorithm will be thoroughly described in Chapter 5.

4.2 Composition-Aware Utility Cost

After having derived the formulas for expected compositional QoS for a particular service selection, we are going to use them in order to define a good estimate to answer the following question:

How useful would it be to choose service candidate $i \in I$ (for the corresponding task T_j) in a service composition of user $u \in U$?

Such a cost will then guide the service selection algorithm, which will aim to optimize it. A single value of a utility cost is local, but it we expect the cost to lead to good performance, it should also be global-aware. This means that other tasks in the composition should be taken into account when estimating the cost, not only the QoS value of a connection $i \to u$ (or the QoS values of other service instances in task T_j only).

The idea of our utility cost is to reflect both the global QoS requirements of the user and the actual QoS values of all services, including the services corresponding to other tasks. For example, consider a situation where a relatively good QoS value of the observed service i might still be insufficient for user u because services in other tasks mostly have lower QoS values for this user. In this case, we need a better service: matching $i \to u$ should be expensive because it is still difficult to satisfy u's global QoS requirement considering other tasks. This will force the optimization algorithm to choose a service with even better QoS in task T_j. In another situation, if matching $i \to u$ is good enough even if the corresponding local QoS value is not high, the algorithm should be guided to make that choice based on the low utility cost.

Let us estimate how feasible it is (from 0 to 1) to match $i \to u$ for task T_j, considering the k-th QoS property. Informally:

- The matching difficulty of $i \to u$ is equal to 0 ("easy") if selecting $i \to u$ for task T_j along with the *worst* services for other tasks still satisfies Req_u^k, i.e. the k-th QoS requirement for user u.
- On the other hand, if selecting $i \to u$ for task T_j requires us to select only the *best* services for other tasks to satisfy the requirement, then the match $i \to u$ is 1 ("difficult").
- If selecting $i \to u$ for task T_j requires us to select at least the *average* services for other tasks to satisfy the requirement, then the match $i \to u$ is 0.5 ("medium").

To formalize the above intuitions, let us first formalize the notions of "worst", "best", and "average" service for a corresponding user, task and QoS property. For that purpose, we will perform

the QoS-based ranking of services in each task. Since the number of services differs from task to task, we use percentiles as normalized ranks, as in the following definition ("p" stands for "percentile").

Definition 4.2.1 (p-quality)**.** *For user u, QoS property k, and task T_j, we say that service $i \in I_j$ has p-quality if it is positioned at the p-th percentile in the list of services I_j sorted in ascending order by their k-th QoS property for user u (high percentile – better QoS). Such a service is denoted by p-choice$^k(u, I_j)$.*

For example, if service $i \in I_j$ is such that QoS^k_{ui} is better than (or equal to) 90% of all QoS^k_{us}, $s \in I_j$, then it has 90%-quality, i.e., $i = 90\%\text{-choice}^k(u, I_j)$. A service with the best QoS^k for user u in I_j has 100%-quality (it is $100\%\text{-choice}^k(u, I_j)$), and a service with the worst QoS has 0%-quality.

We are now ready to define the matching difficulty of $i \to u$ as an answer to the following question (informally):

If $i \to u$ is selected in task T_j, do we need low or high quality services in other tasks to satisfy Req^k_u?

Of course, if we need to choose high-quality services, there are less possibilities for selection and the difficulty is higher. This is formalized in the following definition.

Definition 4.2.2 (Matching difficulty)**.** *The matching difficulty $MD^k_j(i \to u)$, defined for user u and service i corresponding to task T_j, with respect to the k-th QoS property, is defined as the smallest number $p \in [0,1]$ for which the following statement is true:*

- *We can satisfy the k-th QoS requirement Req^k_u by the obtained $QoS(C_u)$ if we select the following services for user u: service i for task T_j, and services with p-quality for other tasks $T_{j'}$ $(j' \neq j)$.*

Formally,

$$\begin{aligned} MD^k_j(i \to u) = \min\{p \in [0,1] : \\ Req^k_u \leq QoS(C_u,\, i \cup \{p\text{-}choice^k(u, I_{j'}),\, j' \neq j\}). \end{aligned} \tag{4.14}$$

The precision of p is not essential: the cost is heuristic so it is reasonable to approximate MD with the error of up to 0.1. Therefore, to calculate MD, we suggest to iterate over $p = 0, 10\%, 20\%, \ldots, 100\%$ and select the first p in this sequence for which the corresponding p-quality selection satisfies $Req^k_u \leq QoS(C_u)$. Note that the global $QoS(C_u)$ are expected values, computed according to the proposed compositional QoS model in the previous section, Eq. 4.2-4.13.

Fig. 4.3 illustrates the concept of matching difficulty. To compute the utility of selection $i \to u$ in task T_j, we simulate selecting services in other tasks at different levels of quality. $MD^k_j(i \to u)$ is the lowest level for which the corresponding selection is enough to satisfy Req^k_u. For example, $MD^k_j(i \to u) = 0.2$ means that selecting the services ranked at 20-th percentiles for

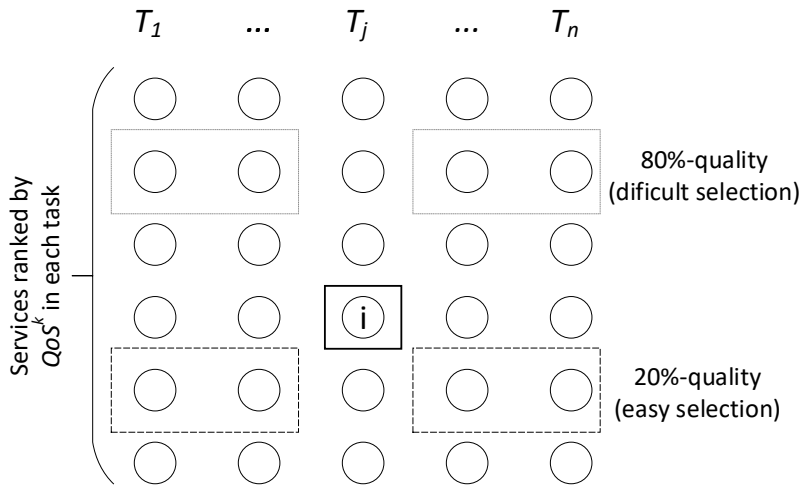


Figure 4.3: Illustration of the concept of matching difficulty $i \to u$ for a fixed user u, service i and QoS property k

all tasks other than T_j is just enough, and such a selection is easy to obtain (low difficulty, i.e., low heuristic cost). In a different case, $MD_j^k(i \to u) = 0.8$ means that selecting much "better" services is necessary, and such a selection is harder to obtain (high difficulty, i.e., high heuristic cost).

If implemented carefully, without calculating the same QoS sums multiple times for different tasks, computation of MD is not a dominating factor in the overall time complexity of the service selection algorithm. In other words, this computation does not significantly contribute to the overall execution time.

Since MD^k corresponds to a single k-th QoS property, the actual utility cost of a given match $i \to u$ should take into account all QoS properties, as in the following definition.

Definition 4.2.3 (Utility cost)**.** *The utility cost c_{ui} for matching the service instance i to user u (for task T_j) is the sum of matching difficulties $MD_j^k(i \to u)$ over all QoS properties $k = 1, 2, \ldots, q$:*

$$c_{ui} = \sum_{k=1}^{q} MD_j^k(i \to u). \tag{4.15}$$

Chapter 5

Fast Multi-Criteria Service Selection for Multi-User Composite Applications

In this chapter we describe an efficient approach to the most general service selection problem, published by the author in [18]. Namely, we have found that the general service selection problem under certain heuristic assumptions can be reduced to a combinatorial transportation problem, which has the advantages of a polynomial solution and allows for an iterative heuristic algorithm which can be applied instead of exact, but more time consuming mixed integer programming approaches.

Sect. 5.1 describes the solution for a single-task scenario, which is a special case of the general algorithm and also serves as a starting point to make the general algorithm easier to understand. Sect. 5.2 describes the proposed general algorithm.

5.1 Single-Task Model

For a single-task model (published by the author in [20]), we consider a set U of active users, each with an execution plan consisting of a single (atomic) task. We assume that all users demand the same task, with a set I of the available service instances (a service class). In case of different tasks for different users, we can separately solve the selection subproblems for each task, which can be parallelized.

Each service instance (or simply *service*) $i \in I$ has a throughput limit THR_i which is the maximal number of requests it can handle in the given time frame. For each considered QoS property (such as response time, reliability, price...) there is an input matrix of size $|U| \times |I|$ describing the corresponding QoS values for each (user, service) pair. Namely, QoS^k_{ui} denotes the k-th QoS property for invocation of service $i \in I$ by user $u \in U$.

As in the general model, at this point we do not explore how the QoS matrices are obtained. Their values can be derived either by QoS prediction (as in Chapter 3) or by estimating values

using service monitoring approaches, a variety of which have been described in the literature [118, 119]. Although sometimes effective, service monitoring methods can have limitations in practice as frequent service polling for purposes of reliability estimation can lead to degradation in service performance. One way to mitigate the issues present in service monitoring is to utilize predictive methods, either by fitting the collected data to predefined models or by using statistical methods [17, 61, 80, 81].

Each user's request has to be mapped to a service such that, for each (k-th) QoS property, the user's requirement for maximal/minimal QoS value Req_u^k is respected. In addition, throughput limits must be satisfied: for each $i \in I$ it must hold that THR_i is not less than the number of requests mapped to i. The goal is to find a solution respecting all (or as many as possible) QoS requirements.

Our approach constructs an instance of *transportation problem* (TP), a well known problem in operational research [120]. Let us first briefly define the (abstract) transportation problem in general terms, and then describe how our service selection problem can be reduced to it. In a transportation problem there are two kinds of entities, which can be represented as nodes on the left and the right side of a bipartite graph. On one side there are *suppliers* (e.g. factories), each having a given number of items to ship. On the other side there are *demanders* (e.g. shops), each having a given number of items to receive. Each supplier-demander connection (represented by an edge in the bipartite graph) has a *shipping cost*: the price of shipping a single item from the particular supplier to the particular demander. The goal is to find a *shipping distribution* (how many items to ship from a particular supplier to a particular demander) which

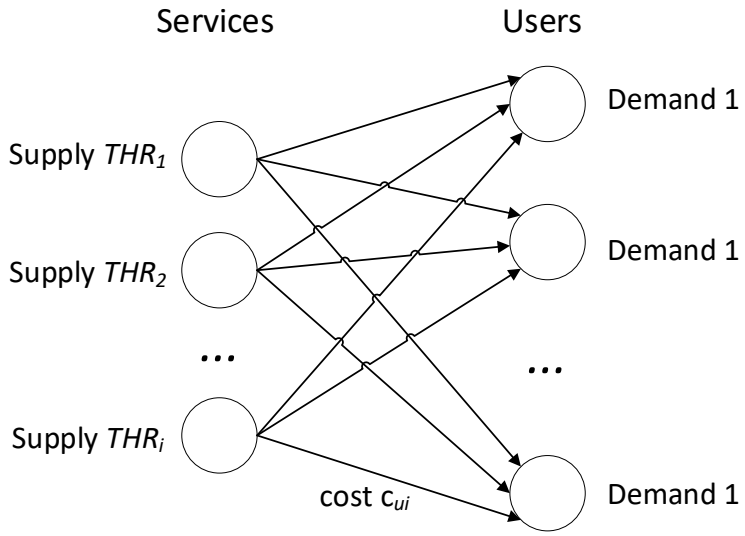


Figure 5.1: Reduction of single-task service selection to the transportation problem (THR_i is a throughput limit of service i)

will minimize the *total shipping cost*. That cost is equal to the sum of shipping costs for each supplier-demander connection multiplied by the chosen number of items shipped along that connection. The requirement to satisfy is that each demander receives as many items as its given demand and that each supplier ships no more items than its given supply.

The applied transportation problem in the case of a single-task service selection is depicted in Fig. 5.1. As can be seen in the figure, nodes on the right side of the bipartite graph represent users, each having a *demand* which in our case equals to the number of required requests – usually, one request per user is made in a single-task scenario. Nodes on the left side of the bipartite graph represent services, each having a *supply* which is in our case equal to its throughput* THR_i. Matching user u to service i has a *cost* c_{ui} of "shipping" a single request through the corresponding user-service edge, which in our case equals 0 ("free shipping") if the corresponding QoS^k_{ui} values satisfy all the QoS requirements of user u, and otherwise equals the number of unsatisfied requirements.

For example, if user u has three QoS requirements (for reliability, response time, and price), and if service i satisfies all of them, the shipping cost c_{ui} is zero. If service i satisfies the QoS requirements for response time and price, but not for reliability, then $c_{ui} = 1$ because one requirement is not satisfied. If none of the three QoS requirements would be satisfied by selecting $i \to u$, then $c_{ui} = 3$. Minimizing the total shipping cost in the service selection will, therefore, lead to minimization of the total number of unsatisfied QoS requirements (or, equivalently, maximization of the number of satisfied QoS requirements). Note that this definition is special to the single-task scenario only. In the general algorithm with service composition, utility cost defined in the previous chapter (based on *matching difficulty*) will be used as c_{ui}. It is not used here because of the nature of its definition, which is meaningless in the single-task case.

In our transportation problem, the goal is to find the shipping distribution (i.e., to choose edges) from the suppliers (services) to the demanders (users) which minimizes the total shipping cost while satisfying all demands and supply constraints. If the obtained total cost is 0, it means that the obtained user-service matching satisfies all of users' QoS requirements. This is formalized in the following definition.

Definition 5.1.1 (Single-task TP solution)**.** *Let $x_{ui} \in \{0,1\}$ denote whether an edge $i \to u$ is chosen for shipping; i.e., whether service i is selected for user u's request. The unknowns $\{x_{ui} : u \in U, i \in I\}$ form a solution of the corresponding transportation problem if the following constraints are satisfied:*

- *Request demands: for each user $u \in U$,*

$$\sum_{i \in I} x_{ui} = 1, \tag{5.1}$$

*Throughput is abbreviated as THR since we use the abbreviation TP for the transportation problem.

i.e., exactly one service is chosen for the request of user u.

- *Request supplies: for each service $i \in I$,*

$$THR_i \geq \sum_{u \in U} x_{ui}, \tag{5.2}$$

i.e., the total number of users invoking i does not exceed its throughput limit.

- *The total cost*

$$C_{total} = \sum_{u,i} x_{ui} c_{ui} \tag{5.3}$$

is minimized, where the edge cost c_{ui} is equal to the number of unsatisfied QoS requirements of user $u \in U$ in case that service $i \in I$ is chosen for her request.

The transportation problem in our case is *unbalanced* because not all supplies have to be used, i.e., service throughputs do not have to be exhausted. In other words, some services will not have to process as many requests as they are maximally able to. In the *balanced* TP, for which the standard TP algorithms are used, the difference between the total demand and the total supply is equal to zero, meaning that all supply must be spent, which makes the algorithms simpler. To balance the problem, we add an artificial "dummy user" with the demand equal to this difference ($\sum THR_i - \#requests$) and connect it to all services with zero cost, making them able to "spend" their unused throughput on the dummy user. In this model, each service *i* will process exactly THR_i requests, but only those which do not correspond to the dummy user will be actually realized. We thus have equality in the second constraint (for request supplies):

$$THR_i = \sum_{u} x_{ui}. \tag{5.4}$$

We solve the transportation problem in two steps, combining two standard and well-known algorithms for TP:

1. Finding an initial (heuristic) solution using Vogel Approximation Method (VAM) [120, 121].
2. Iteratively improving the solution until it is optimal, using the Transportation Simplex Method (TSM) which is a (polynomial) specialization of the simplex method designed for TP [122, 123].

This two-step approach we call SS-TSM (*Service Selection by Transportation Simplex Method*). If we stop after the first step, we get a faster, but possibly suboptimal solution, which we call SS-VAM (*Service Selection by Vogel Approximation Method*). The tradeoff between speed and optimality will be investigated in the experimental results (Chapter 6). We have experimented with other optimal solutions of TP, such as reducing it to a minimum cost flow problem, but the time performance was inferior to the proposed algorithm by an order of magnitude.

5.2 General Service Selection Algorithm

This section generalizes the approach from the previous section, describing the proposed algorithm for multi-user service selection in the multi-task case. Namely, we assume that each user runs an application which can, in the observed time frame (which is chosen based on the system properties), be described as a composition of tasks – a service composition. Such execution plans can incorporate various compositional structures including Sequence, Branch, Parallel, and Loop. This means that the actual execution path is not necessarily known in advance. In addition, users are not necessarily synchronized with each other – they can have different execution plans, sharing some or all of the tasks.

The procedure is performed by e.g. the execution engine in a multi-tenant Service-Based System (SBS). This corresponds to the "third multi-tenancy maturity level" from [12] where all tenants (users) share the execution engine of the SBS and a set of services which compose the SBS. The proposed algorithm is performed in three steps:

1. Expected user demands for each task are derived according to the proposed compositional QoS model (described in Sect. 4.1).
2. Heuristic user-service utility cost is derived according to the proposed concept of *matching difficulty* which is based on the global QoS requirements and QoS values (described in Sect. 4.2).
3. Local instances of transportation problem for each task (with user demands from step 1 and utility cost from step 2) are solved in several iterations until a global solution is found.

The high-level illustration of the algorithm is given in Fig. 5.2. The main phase of the algorithm (step 3) is solving a transportation problem TP_j for each task T_j that appears in any of the given service composition plans. A problem instance is similar to the single-task case, with the following setup:

- TP_j includes only the users for which T_j appears in their composition plan, and only the services corresponding to task T_j. Let U_j and I_j denote the corresponding set of users and the corresponding service class.
- The TP demand (number of requests) of user $u \in U_j$ now equals $EI(C_u)_j$ (expected number of invocations of T_j). The TP supply of a service instance $i \in I_j$ is equal to its THR_i limit as before.
- The TP_j utility cost c_{ui} is now equal to the *matching difficulty* between user u and service i according to the definition from Sect. 4.2:

$$c_{ui}(TP_j) = \sum_{k=1}^{q} MD_j^k(i \to u). \tag{5.5}$$

The iterative procedure starts by computing utility costs and solving the transportation prob-

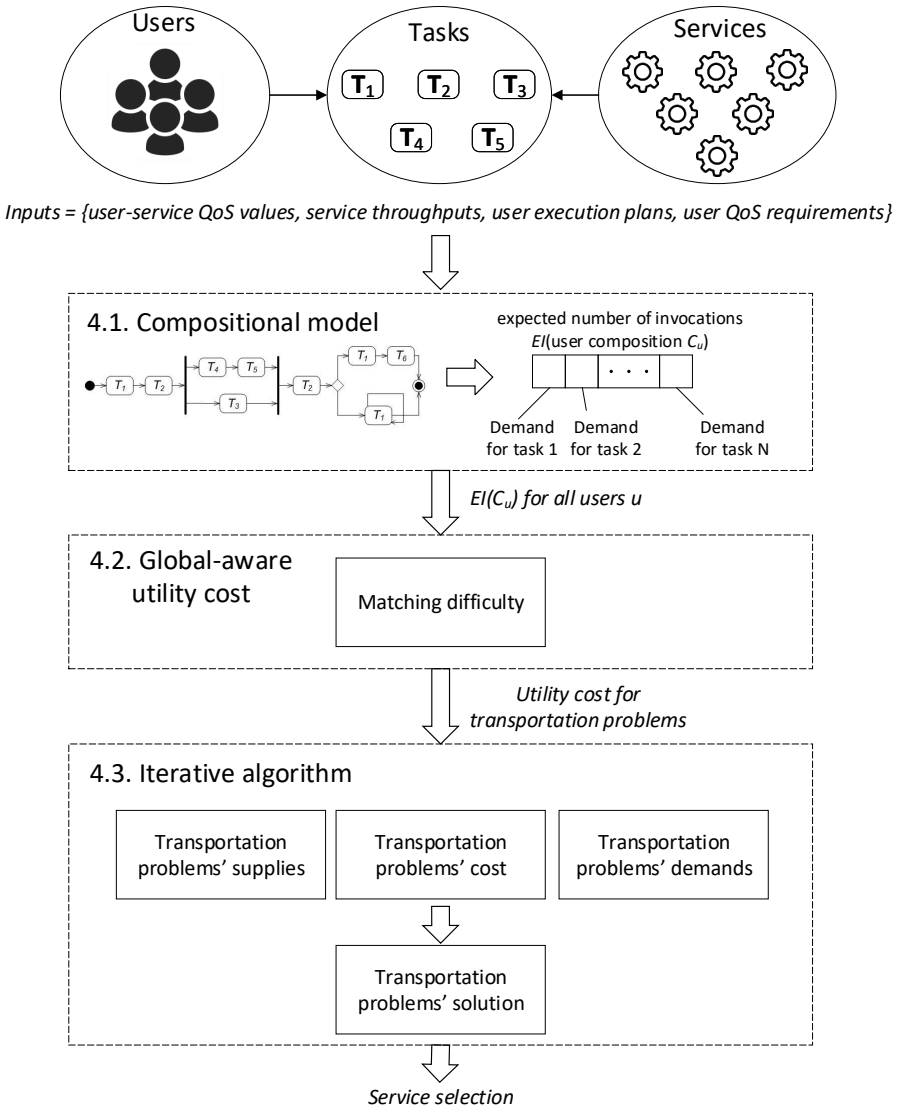


Figure 5.2: High-level algorithm illustration

lems TP_j for each task T_j, obtaining a global service selection (see Fig. 5.3). If the selection satisfies all users' QoS requirements, the algorithm is finished. Otherwise, we will tweak the utility cost to put more weight on the unsatisfied QoS requests and make another iteration of the algorithm. This is formalized in the following definition.

Definition 5.2.1 (Utility cost by iterations)**.** *Let w_u^k denote the weight (significance) of k-th QoS property for user u. Initially we set $w_u^k := 1$. After each iteration, if Req_u^k is not satisfied, w_u^k is multiplied by a factor of $\alpha(r) = 1 + e^{3-r}$, where r ("round") is the iteration number ($r = 0, 1, 2, \ldots$). For the subsequent iteration, the utility cost is redefined as follows:*

$$c_{ui}(TP_j) = \sum_{k=1}^{q} w_u^k MD_j^k(i \to u). \tag{5.6}$$

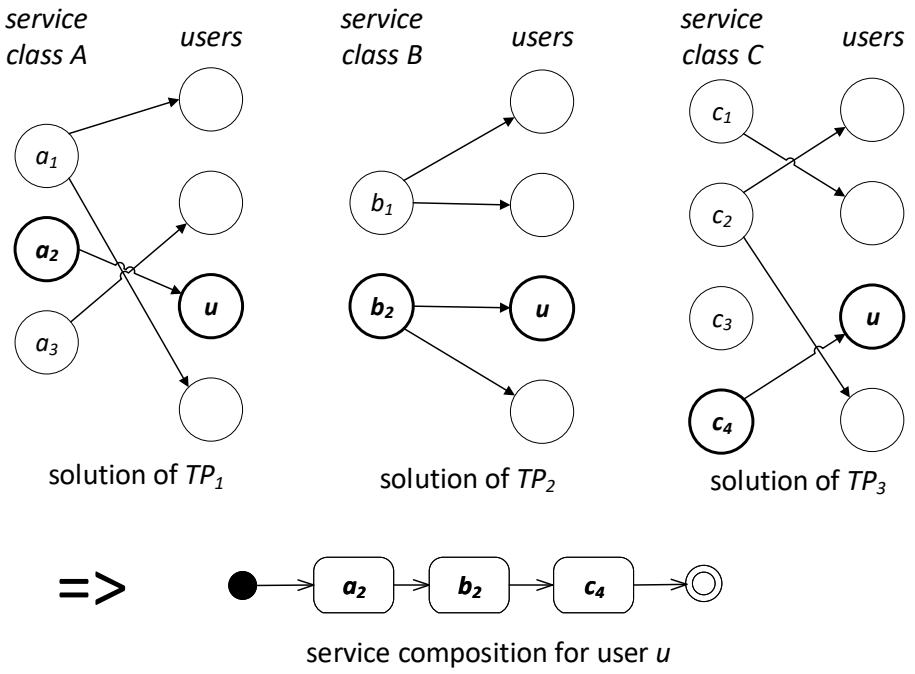


Figure 5.3: Illustration of the global selection: services a_2, b_2, c_4 are selected for tasks T_1, T_2, T_3 for user u.

Increasing weights for an unsatisfied requirement Req_u^k ensures that the corresponding QoS values QoS_{ui}^k have a significant influence on the utility cost in all service candidates for user u, so the algorithm will lean towards satisfying that requirement by minimizing the most significant cost summands for that user.

The reasoning behind the choice of a decreasing exponential function in $\alpha(r) = 1 + e^{3-r}$ is to ensure that the increments are initially large ($\alpha(0) \approx 21.1$), allowing for larger changes in early iterations, but they quickly drop in subsequent iterations. In case that some requirements remain unsatisfied, the algorithm converges after about 10 iterations as the factor $\alpha(r)$ becomes close to 1.0. We have found that the proposed choice of the exponent in e^{3-r} is a good balance between initial aggressive weight increments and low total number of iterations, both of which are important for efficiency.

In case of convergence without global success, the service selection from the best iteration in terms of the total number of satisfied QoS requirements (or any chosen measure) is selected as the "best effort" output of the algorithm.

The summary of the proposed algorithm is shown in Algorithm 4. In short, given the users' compositions plans, QoS requirements, and user-service QoS values, the algorithm first employs the compositional QoS model to calculate the expected number of invocations for each user and task. Then it estimates the matching difficulties and uses them to compute utility costs which are being minimized in iterative solutions of transportation problems. The solution of a

transportation problem in each task gives an appropriate service to select for each user.

ALGORITHM 4: General Service Selection Algorithm

Data: n tasks with $|I_1|,\ldots,|I_n|$ service instances
Data: users' composition plans $C_1,\ldots,C_{|U|}$
Data: user-service QoS matrices QoS_{ui}^k for each QoS property $k=1,\ldots,q$
Data: QoS requirements Req_u^k, $k=1,\ldots,q$ for each user $u=1,\ldots,|U|$

— Compute expected invocations $EI(C_u)$ for each user (Eq. 4.1-4.12)
— Compute matching difficulty $MD_j^k(i\to u)$ for each user u, QoS property k, task T_j, and service $i\in I_j$ (Eq. 4.14)
— Set $w_u^k := 1$ for each $u=1,\ldots,|U|$, $k=1,\ldots,q$

for $r=0,1,\ldots$ *until success or convergence* **do**
  for *each task* T_j *independently* **do**
    Calculate utility cost for TP_j (Eq. 5.6)
    Assign $EI(C_u)_j$, $u\in U_j$ as demands of TP_j
    Assign THR_i, $i\in I_j$ as supplies of TP_j
    $Selection_j := Solve(TP_j)$ by SS-TSM (or SS-VAM)
  end
  $Selection := \bigcup_{j=1}^{n} Selection_j$
  if *all* Req_u^k *are satisfied by Selection* **then**
    $success := true$
    $bestSelection := Selection$
  else
    $w_u^k := w_u^k \cdot (1+e^{3-r})$ for each unsatisfied Req_u^k
    if *#satisfied(bestSelection)* $<$ *#satisfied(Selection)* **then**
      $bestSelection := Selection$
    end
  end
end
return *bestSelection*

5.2.1 Complexity Analysis

As the maximal number of algorithm iterations is small, the overall time complexity is dominated by solving n transportation problems. Under careful implementation, the complexity of solving a single TP_j by Vogel Approximation Method (VAM) is $O(|U_j|\cdot|I_j|)$ [124], where U_j and I_j are sets of users and services corresponding to task T_j. Since the transportation problems for different tasks can be solved independently (concurrently) on several processors, the complexity of the proposed SS-VAM model is

$$O\left(\frac{1}{\#processors}\sum_{j=1}^{n}|U_j||I_j|\right). \tag{5.7}$$

In the proposed SS-TSM model, where the Transportation Simplex Method is used to optimize the initial VAM solution, the exact complexity depends on the actual cost values and the number of TSM iterations, which can be (almost) zero if VAM already finds the (almost) optimal

solution. In other words, SS-TSM does little improvement over SS-VAM. The experiments will show that this is often the case, i.e., the average complexity of SS-TSM is the same as for SS-VAM up to a constant factor. The theoretical worst-case complexity is proven to be polynomial even for the more general Network Simplex Method [125].

Chapter 6

Evaluation

In this chapter we present the extensive experiments covering the contributions of this dissertation. The chapter is organized as follows. First, in Sect. 6.1 we evaluate the proposed GCM measure for a collaborative filtering dataset (described in Sect. 3.1). Then, in Sect. 6.2 we evaluate the proposed QoS Prediction model (described in Chapter 3). Finally, in Sect. 6.3 we evaluate the proposed compositional service selection model from Chapters 4-5. The results presented here were also published by the author in [19], [17] and [18].

6.1 Global Correlation Measures for a CF dataset

The goal of the experiments was to check whether the defined GUCM and GICM measures of correlation for a collaborative filtering dataset satisfy the required properties of such a measure, outlined in Sect. 3.1.

6.1.1 Results on Random Categorical Sets

In this experiment, we have randomly generated several categorical user-item matrices of size 5000×5000 with rating values in $\{0, 1, \ldots, K-1\}$, where K is the number of categories (possible rating values). The results are given in Fig. 6.1. We can see that a higher number of categories results in a lower measure value, which is a desired property because increasing the number of possible rating values increases the randomness ("entropy") in the dataset, making the users (or items) less likely to be similar. Also, as expected, the measure approaches the value of 0 for highly random datasets.

6.1.2 Results on Artificial Sets with Natural Clusters

To show that the measure is roughly proportional to the amount of agreement in the dataset, we have created several complete 5000×5000 user-item matrices and filled them with random

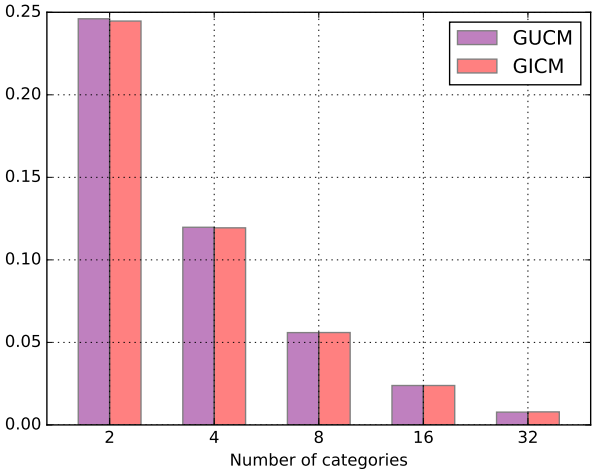


Figure 6.1: Random categorical sets, $N = M = 5000$

integer values from $[0, 10000]$ so that there are K *clusters* of equal users, where K was varied ($K = 1, 2, \ldots, 10, 20, \ldots, 90, 100$). The results are given in Fig. 6.2. As expected, $GUCM \approx 1$ when all users agree, and GUCM decreases as the number of natural clusters (user variability) increases; moreover, $GUCM \approx 1/K$. Since we only clustered the users, GICM is lower than GUCM as expected. High correlation between users implies some correlation between items, which explains why GICM is also negatively correlated with the number of user clusters.

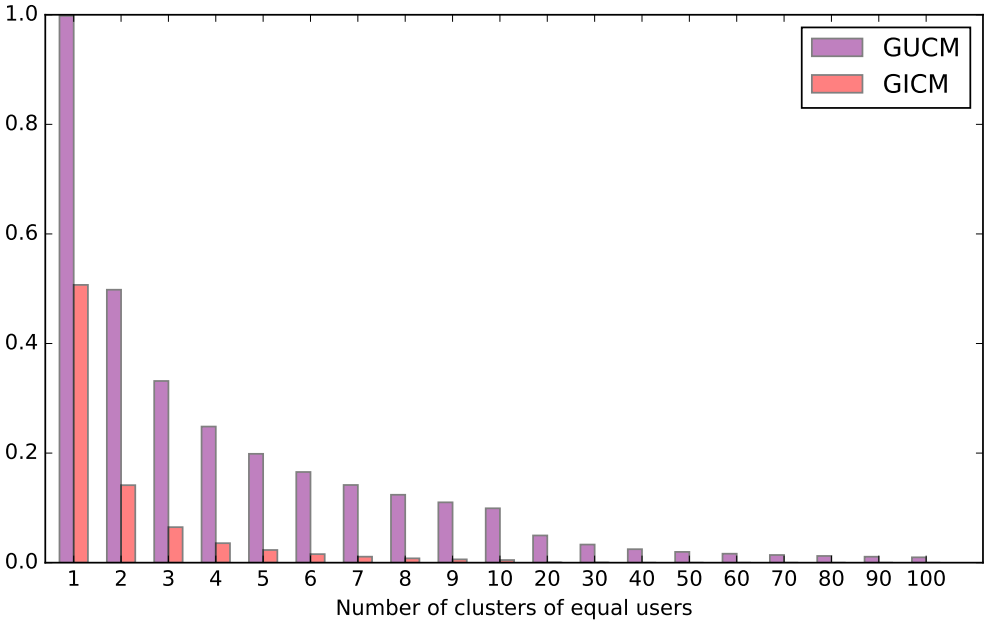


Figure 6.2: Artificial sets with natural user clusters, $N = M = 5000$

Table 6.1: Real-world datasets description

Dataset	*users*	*items*	*ratings*	*range*	*reference*
Amazon QoS[a]	50	49	2450	[0.0, 1.0]	[25]
FilmTrust[b]	1508	2071	35497	0.5, 1, ..., 4	[126]
Ciao[c]	17615	16121	72665	1, 2, 3, 4	[127]
Epinions[d]	40163	139738	664824	1, 2, 3, 4	[128]
Flixter[e]	147612	48794	8196077	0.5, 1, ..., 5	
MovieLens 100K[f]	943	1682	100000	1, 2, ..., 5	
MovieLens 1M[g]	6040	3706	1000209	1, 2, ..., 5	
Jester[h]	50691	150	1728830	[-10.0, 10.0]	[129]
Netflix[i]	50000	17770	10375459	1, 2, ..., 5	
Anime[j]	69600	9927	6337241	1, 2, ..., 10	
YahooMovies[k]	7642	11916	211231	1, 2, ..., 13	
BookCrossing[l]	105281	340541	1149763	0, 1, ..., 10	[130]
MovieTweetings[m]	51503	29564	656276	0, 1, ..., 10	[131]

[a] http://ccl.fer.hr/people/research-assistants/marin-silic/clus-evaluation-dataset
[b] http://www.librec.net/datasets/filmtrust.zip [c] http://www.librec.net/datasets/CiaoDVD.zip
[d] http://www.trustlet.org/downloaded_epinions.html
[e] http://www.cs.ubc.ca/ jamalim/datasets [f] http://grouplens.org/datasets/movielens/100k
[g] http://grouplens.org/datasets/movielens/1m [h] http://eigentaste.berkeley.edu/dataset
[i] http://www.netflixprize.com [j] https://www.kaggle.com/CooperUnion/anime-recommendations-database
[k] https://webscope.sandbox.yahoo.com [l] http://www2.informatik.uni-freiburg.de/∼cziegler/BX/
[m] https://github.com/sidooms/MovieTweetings

6.1.3 Results on Real-World Data

For each of 13 considered real-world datasets (see Table 6.1), we have randomly divided the data into training and testing sets (ratio 90:10) and the GUCM/GICM measures for each dataset were computed using the training set only. We have compared these measures with two other, computationally much more demanding measures. The first was the percentage of pairs with (PCC) similarity higher than 0.5, called *similar user pairs* and *similar item pairs*. The second was the average similarity of pairs which have at least one common rating, i.e., ratings different by at most 10% of the rating scale. We called this measure *user similarity degree* and *item similarity degree*. This measure is an ingredient of *Redundancy* measure from [90]. *

We have also implemented standard user-based (UPCC) and item-based (IPCC) collaborative filtering approaches based on similar users/items as described in Sect. 2.1.1, using the training set data to predict the values in the testing set, as well as the predictions based on

*The Redundancy measure itself, having other ingredients besides similarity and being made for a different purpose, did not correlate with the proposed measures.

Matrix Factorization (MF) algorithm using stochastic gradient descent [62], where the hyper-parameters were learned from the training set using cross-validation. The accuracy was defined using relevance of the items with highest (Top 10) ratings for a given user; namely, we used Normalized Discounted Cumulative Gain (nDCG) measure [132, 133].

Table 6.2: Numerical results for the real-world datasets

dataset	GUCM	GICM	sim. user pairs	sim. item pairs	user sim. deg.	item sim. deg.	UPCC nDCG	IPCC nDCG	MF nDCG
Netflix	0.078	0.109	0.155	0.285	0.161	0.469	0.902	0.904	0.889
MovieLens1M	0.068	0.121	0.201	0.168	0.215	0.199	0.909	0.868	0.910
MovieLens100K	0.068	0.124	0.179	0.211	0.171	0.276	0.914	0.920	0.905
BookCrossing	0.109	0.115	0.001	0.002	0.513	0.328	0.882	0.950	0.878
Jester	0.162	0.141	0.146	0.031	0.144	0.347	0.955	0.941	0.948
FilmTrust	0.142	0.213	0.147	0.028	0.113	0.246	0.958	0.979	0.955
Epinions	0.178	0.212	0.012	0.001	0.647	0.296	0.968	0.980	0.969
YahooMovies	0.159	0.235	0.307	0.029	0.435	0.300	0.971	0.971	0.968
Flixter	0.223	0.255	0.045	0.282	0.161	0.614	0.943	0.971	0.940
CiaoDVD	0.267	0.287	0.001	0.005	0.268	0.226	0.988	0.983	0.987
Anime	0.250	0.342	0.206	0.208	0.266	0.342	0.961	0.947	0.957
MovieTweetings	0.377	0.400	0.019	0.015	0.299	0.268	0.983	0.976	0.981
AmazonQoS	0.684	0.650	1.000	0.989	0.988	0.877	0.999	0.993	0.999

Table 6.3: Correlations between GUCM/GICM and other prediction ability indicators across 13 real-world datasets

perspective	*proposed metric*	*other metric*	*Pearson corr. coef.*	*p-value*
user-based	GUCM	similar user pairs	0.705	0.007
item-based	GICM	similar item pairs	0.679	0.011
user-based	GUCM	user similarity degree	0.689	0.009
item-based	GICM	item similarity degree	0.633	0.020
user-based	GUCM	UPCC Top-10 nDCG	0.728	0.005
item-based	GICM	IPCC Top-10 nDCG	0.622	0.023
general	(GUCM+GICM)/2	MF Top-10 nDCG	0.751	0.003

Numerical results are given in Table 6.2 and the correlations between the proposed measures and other prediction ability indicators (i.e., between columns of Table 6.2) are given in Table 6.3. For each row in Table 6.3, Pearson correlation coefficient (in range $[-1, 1]$) was calculated for two 13-element vectors, each containing the values of a given metric for all datasets. p-value is the probability that the obtained correlation coefficients were accidental. Relatively high coefficients ($0.62 - 0.75$) and low p-values confirm the relation between the proposed

measures and other prediction ability indicators, showing that higher GUCM and GICM tend to imply more similarities and greater prediction ability.

Table 6.4: Correlations between existing (less efficient) measures and prediction accuracies across 13 real-world datasets

perspective	*metric*	*prediction accuracy*	*Pearson corr. coef.*	*p-value*
user-based	similar user pairs	UPCC Top-10 nDCG	0.356	0.233
item-based	similar item pairs	IPCC Top-10 nDCG	0.088	0.775
general	(sim. user pairs + sim. item pairs)/2	MF Top-10 nDCG	0.285	0.345
user-based	user similarity degree	UPCC Top-10 nDCG	0.371	0.212
item-based	item similarity degree	IPCC Top-10 nDCG	0.299	0.320
general	(user sim. deg. + item sim. deg.)/2	MF Top-10 nDCG	0.360	0.227

For the sake of comparison, Table 6.4 presents the results for the alternative measures *similar user/item pairs* and *user/item similarity degree*. As we can see, the correlations between these measures and the prediction accuracies, although expectedly positive, are weaker and less significant than the same correlations for GUCM/GICM measures. This means that GUCM and GICM are better estimators of prediction accuracies, apart from being computationally much more efficient.

Relations between the proposed measures and the precision of prediction algorithms are illustrated in Fig. 6.3 (GUCM vs. user-based predictions), Fig. 6.4 (GICM vs. item-based predictions), and Fig. 6.5 (average of GUCM and GICM vs. MF-based predictions). In each figure, black columns represent the proposed measures (scale on the left y-axis), while red columns represent the Top-10 nDCG precision of the prediction algorithms (scale on the right y-axis). The figures show that usually, when GUCM and GICM are higher, Top-10 predictions are better. There are anomalies since the proposed measures are not more than heuristics, but

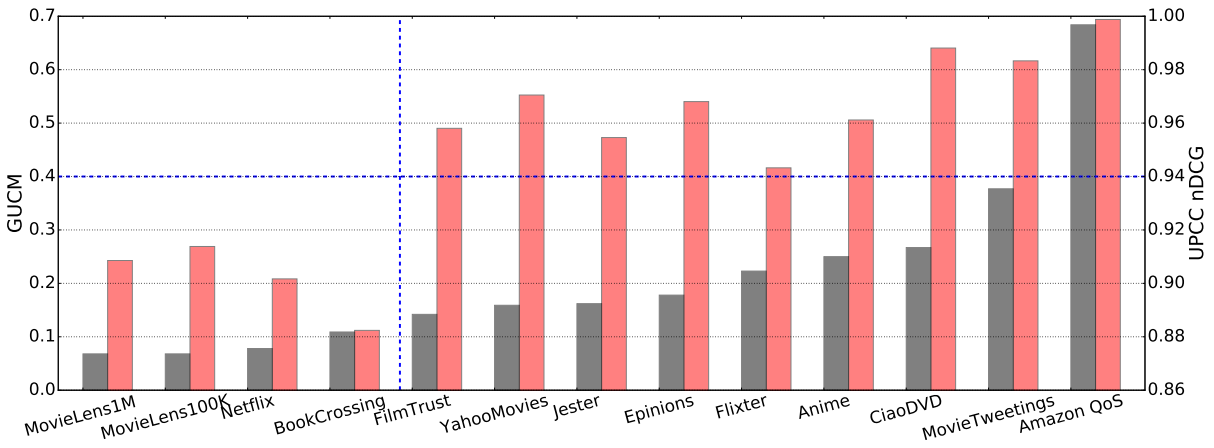


Figure 6.3: GUCM (black) and user-based Top-10 precison (red)

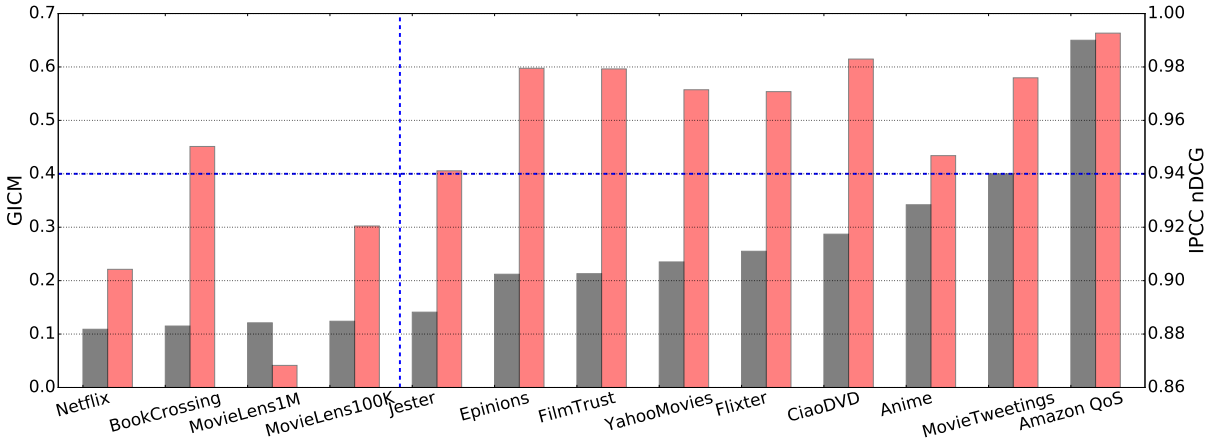


Figure 6.4: GICM (black) and item-based Top-10 precision (red)

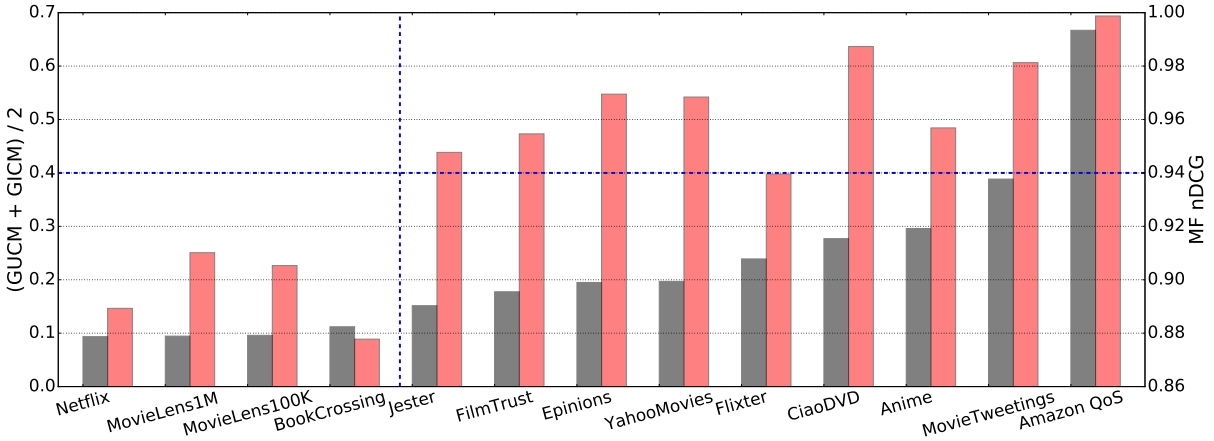


Figure 6.5: (GUCM+GICM)/2 (black) and MF-based Top-10 precision (red)

the following observation can be empirically derived from these results: a proposed measure lower than 0.13 usually implies the corresponding Top-10 precision lower than 94% (supported in 11/12 cases), while a proposed measure higher than 0.14 usually implies the corresponding Top-10 precision of at least 94% (supported in 27/27 cases). This is illustrated in the figures: the vertical line separates GCM measures lower and higher than 0.135, while the horizontal line separates precisions lower and higher than 94%.

6.2 QoS Prediction

In this section, we present the detailed evaluation results that analyze the prediction accuracy and time performance of the proposed real-time adaptive QoS prediction model. In our experiments, we have compared our model to the model called Hybrid [40], which is based on collaborative filtering and uses a similar formula:

$$P_{u,i} = \lambda_1 UPCC_{u,i} + \lambda_2 IPCC_{u,i}, \tag{6.1}$$

also considering only the most similar users/services, but calculating *UPCC* and *IPCC* in a standard way (as in Eq. (3.26)), while the proposed model uses a different formula (as in Eq. (3.28)).

To evaluate prediction accuracy, we used the standard error measures *Mean Absolute Error (MAE)* and *Root Mean Square Error* (*RMSE*):

$$MAE = \frac{\sum_j^K |p_j - \hat{p}_j|}{K}, \quad RMSE = \sqrt{\frac{\sum_j^K (p_j - \hat{p}_j)^2}{K}}, \tag{6.2}$$

where K is the cardinal number of the prediction set consisting of user-service prediction queries: p_j is the actual value, and $\hat{p}_j$ is the predicted value for a given query. Note that MAE and RMSE can range from 0 to ∞. It is a negatively-oriented score, which means that lower values are better. When the range of values was large (i.e. RT and TP datasets, see Sect.6.2.1), instead of MAE and RMSE we used *Mean Absolute Percentage Error (MAPE)* and *Root Mean Square Percentage Error* (*RMSPE*):

$$MAPE = \frac{\sum_j^K |(p_j - \hat{p}_j)/p_j|}{K}, \tag{6.3}$$

$$RMSPE = \sqrt{\frac{\sum_j^K ((p_j - \hat{p}_j)/p_j)^2}{K}}. \tag{6.4}$$

Sect. 6.2.1 describes the datasets used in the experiments. Sect. 6.2.2 describes types of experiments that were conducted in order to evaluate the proposed approach. Sect. 6.2.3 describes the parameters selection. Sect. 6.2.4 presents the results for AmazonQoS dataset. Sect. 6.2.5 presents the results for the Failure Probability (FP) dataset. Sect. 6.2.6 presents the results for the Extended RT and TP datasets. The time performance results, along with the time complexity discussion, are given in Sect. 6.2.7 and threats to validity are discussed in Sect. 6.2.8.

6.2.1 Datasets Overview

To evaluate the proposed model, we have used multiple datasets with different characteristics. In all experiments, some of the values in a dataset were given to the model as an input (training set); other values were unknown to the model (testing set). The model predicts the unknown values and the error is calculated by comparing predictions with actual values. Details are described in the next subsection.

The *AmazonQos* dataset[†] was obtained as described in [25]. This dataset contains reliability

[†]http://ccl.fer.hr/people/research-assistants/marin-silic/clus-evaluation-dataset/

values in range $[0,1]$ for 50 users and 49 services. The Failure Probability (FP), Response time (RT), and Troughput (TP) datasets are taken from *http://www.wsdream.net/dataset.html* [134]. The FP dataset is a user-item matrix (150 users, 100 services) consisting of values in range $[0,1]$, with 82.27% of the values being equal to 0.000. The RT and TP datasets are user-item matrices (339 users, 5825 services) consisting of positive values in range $[0,20]$ for RT and $[0,1000]$ for TP. To analyze the time performance, we have artificially extended the mentioned RT and TP datasets: each user vector in the user-item matrix was cloned 17 times and modified by +/-5%, giving the Extended RT and Extended TP datasets with 5763 users and 5825 services.

All experiments were conducted with various *densities* (5-50%). Density is the percentage of QoS values $r_{u,i}$ in the user-item matrix which are known to the model.

6.2.2 Types of Experiments

The following types of experiments have been conducted:

- *"Fixed Value Support" experiment.* We have predicted $P_{u,i}$ for all unknown values in the user-item matrix, using the known values $r_{u,i}$ which we assumed were equally reliable. The experiment compares the proposed model with Hybrid.
- *"Variable Value Support" experiment.* We have predicted $P_{u,i}$ for all unknown values in the user-item matrix, but the known values were modified: assuming that $r_{u,i}$ was obtained from 150 invocations, we have selected a random number (1-100) of invocations and calculated $r_{u,i}$ based only on these invocations. Thus, the modified value $r_{u,i}$ (which is used to predict the unknown values) is less accurate. The point of the modification is to utilize the information about the number of invocations $n_{u,i}$ as the support of the corresponding value $r_{u,i}$ (as in Eq. (3.27) and (3.30)), since a greater $n_{u,i}$ suggests a more accurate $r_{u,i}$. Utilizing this information is an important feature of our model (see details in Sect. 3.5). The experiment compares the proposed model with Hybrid.
- *"Stochastic" experiment.* The experiment starts with user-item matrix of a given density, using the variable support data (see the previous point). Then it generates a sequence of randomly shuffled events, containing 80000 updates (user-service invocations) and 400 predictions. The proposed model (labeled "Real-time" in figures) is compared to the same model which ignores the updates (labeled "Static") to see how well the model incorporates the updates to make predictions. Note that, in this experiment, the environment does not change, but the model's knowledge of QoS values changes (increases) as the user-service invocation records arrive.
- *"Changing Stochastic" experiment.* The experiment simulates the changing environment in which the QoS values change in each of the 7 time slots. The environment-specific parameters are stable in each time slot, i.e., the matrix of actual $r_{u,i}$ values (unknown to the model) is stable in each slot, but different for different slots. The corresponding

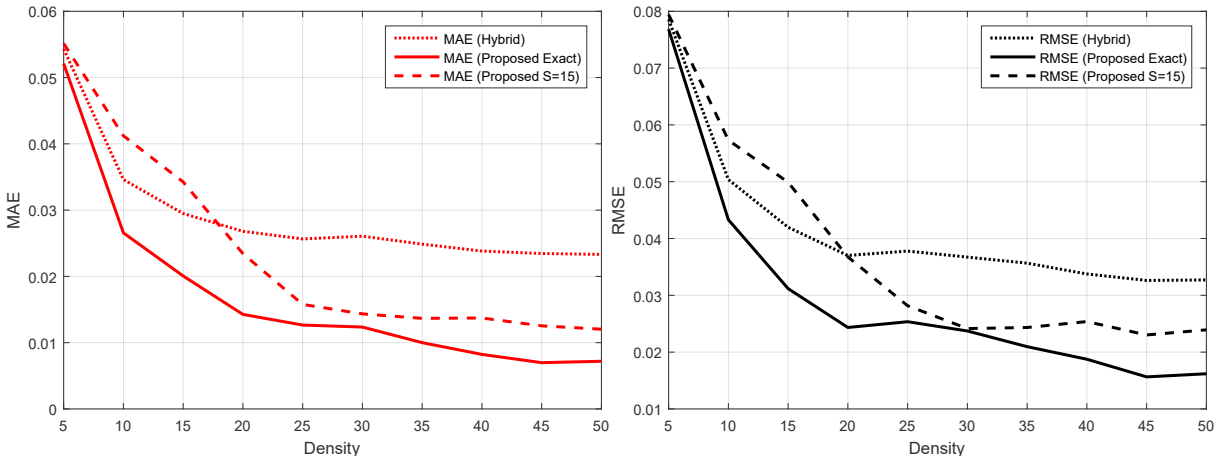
7 matrices were taken from different loads in the dataset from [25]. Initially, no values $r_{u,i}$ are known to the model. In each time slot, a sequence of randomly shuffled events is generated, containing 200000 updates (user-service invocations) and 1000 predictions based on the user-item matrix of the current time slot. To see how well it adapts to the changing environment, the proposed model (labeled "Proposed" in figures) with weight increment constant $\Delta w = 0.01$ (which makes the more recent records contribute more to the predictions) is compared to the same model with $\Delta w = 0$ (labeled "Control") where the weight is not incremented in time.

- *"Scalability" experiment.* The experiment simulates handling more and more updates and predictions. Using a constant matrix of $r_{u,i}$ values unknown to the model, a sequence of randomly shuffled events is generated, containing 100 updates and 100 predictions in the 1st time slot, 1000 updates and 1000 predictions in the 2nd time slot, 10^4 in the 3rd, 10^5 in the 4th, and 10^6 in the 5th time slot. We measure accuracy, as well as update and prediction time of the proposed model (labeled "Proposed") compared to the same model which ignores the updates (labeled "Control"). Note that the term *scalability* in this experiment is not related to the number of users and services, but to the number of incoming updates the system handles.

6.2.3 Model parameters

In the experiments, we investigated the impact of the approximate matrix multiplication versus the exact one. In figures depicting the results, the used value of S (sampling constant), which controls accuracy and time complexity of the approximation, is noted in the graph legend. If exact matrix multiplication is used instead of the approximate one, the word "Exact" is put instead of the S value. Special cases are the Changing Stochastic experiment and the Scalability experiment: since no values are known to the model initially, the precompute phase results with default (zero) values in all matrices so any matrix multiplication is redundant.

As for other parameters, such as K_1 (number of similar users in user-based prediction), K_2 (number of similar services in item-based prediction), and (λ_1, λ_2) which are weights for user-based and item-based predictions, in the experiment results presented below, for each of the tested models (except where noted) we used the parameter values which we empirically found to give the best results, among several tested possibilities, for the corresponding model and experiment on the ratings initally known to the model. In a practical system it is also expected that the administrator empirically finds the appropriate parameter values. These parameters are specified in the text as a triple (λ_1, K_1, K_2); note that $\lambda_2 = 1 - \lambda_1$.



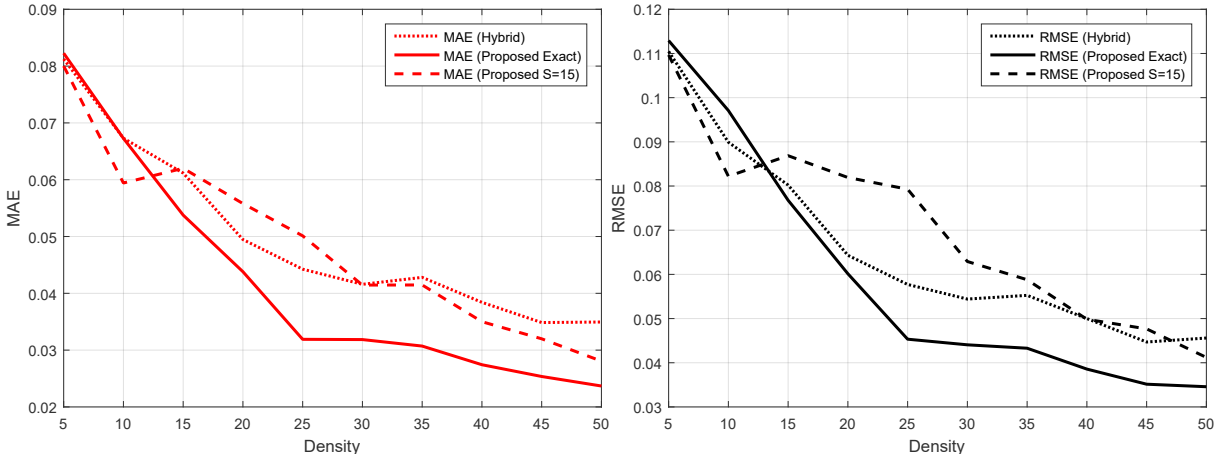
(a) (b)

Figure 6.6: Density impact on MAE and RMSE, AmazonQoS Fixed Value Support experiment

6.2.4 Results for the AmazonQoS Dataset

For this dataset (50 users, 49 services) we have performed all five types of experiments. Fig. 6.6 depicts the results for Fixed Value Support experiment for the proposed model and Hybrid. The parameters (λ_1, K_1, K_2) were $(0.1, 4, 12)$ for Hybrid and $(0.1, 4, 5)$ for the proposed model (Exact and $S = 15$). It can be seen that the exact approach outperforms the Hybrid approach for all data densities. For instance, for data density value of 10%, the Hybrid approach achieves the RMSE value of 0.050, while the exact approach achieves the RMSE value of 0.043. We obtain similar results if we consider MAE measure, as can be seen in Fig. 6.6a. The approximate matrix multiplication approach with the S value equal to 15 provides more accurate prediction for higher data densities (i.e., the RMSE value of 0.024 at density of 30%) when compared to the Hybrid approach (the RMSE value of 0.037 at density of 30%). On the other hand, for lower data densities, the approximate multiplication approach provides slightly less accurate predictions when compared to Hybrid approach.

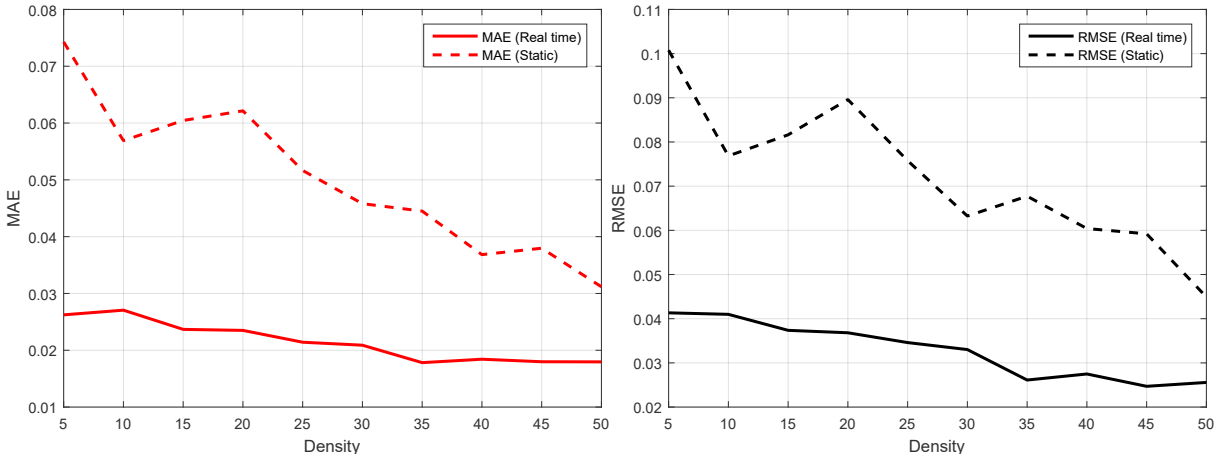
Fig. 6.7 depicts the results for Variable Value Support experiment for the proposed model (Exact and $S = 15$) and Hybrid. The parameters (λ_1, K_1, K_2) were $(0.2, 25, 25)$ for all models. Notice that K_1 and K_2 are larger than in the Fixed Value Support experiment. This is because the values used for prediction are less accurate so more of them need to be taken into account. As expected, all models are less successful than in the Fixed Value Support experiment. For density of 10%, the most accurate prediction is achieved by the proposed model with $S = 15$ (with RMSE of 0.082 and MAE of 0.059). For higher densities, the most accurate prediction is achieved by the proposed Exact model (i.e., the RMSE of 0.044 and MAE of 0.032 at density of 30%), while the model with $S = 15$ provides less accurate predictions (the RMSE of 0.063 and MAE of 0.041 at density of 30%), as well as the Hybrid approach, which gives results similar to $S = 15$ except for densities 15-30% where Hybrid is slightly better.



(a) (b)

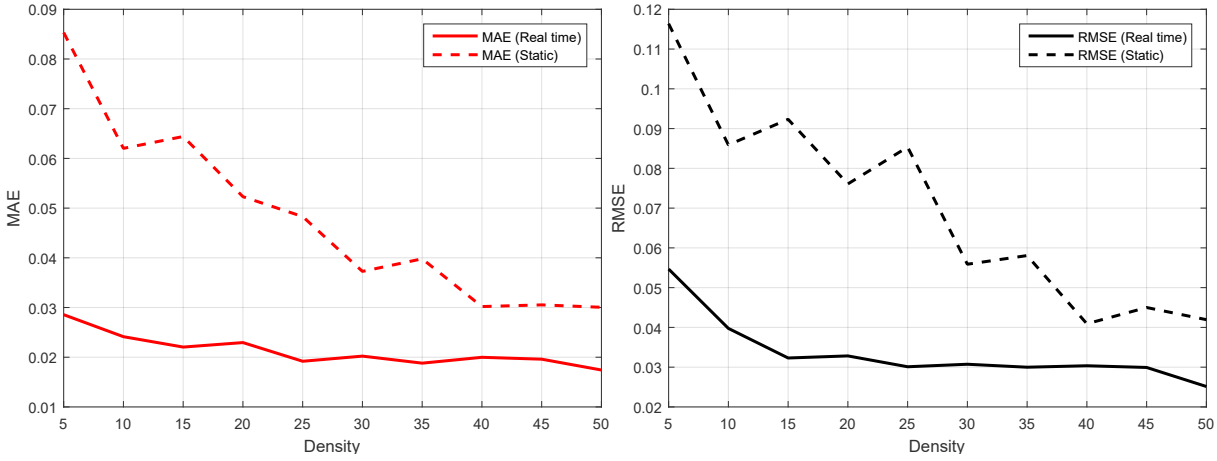
Figure 6.7: Density impact on MAE and RMSE, AmazonQoS Variable Value Support experiment

Figs. 6.8 and 6.9 depict the results for Stochastic experiment. The parameters (λ_1, K_1, K_2) were $(0.2, 25, 25)$ for the Static model, as in the Variable Value Support experiment. As for the Real-time model, λ_1 and λ_2 were set according to (3.32), and K_1, K_2 were maximal ($K_1 = M$, $K_2 = N$) to allow the model to take all updates into account. The results show that Real-time model is much more accurate than Static model: for higher densities and exact matrix multiplication (Fig. 6.8), the MAE and RMSE values are approximately 2 times lower in the Real-time model than in the Static model. If the approximate matrix multiplication is used (Fig. 6.9), the difference is still quite significant (at density of 30%, Real-time model gives RMSE of 0.031 and MAE of 0.020 while Static gives RMSE of 0.056 and MAE of 0.038). The difference is even larger for lower densities since Real-time model takes into account much more records



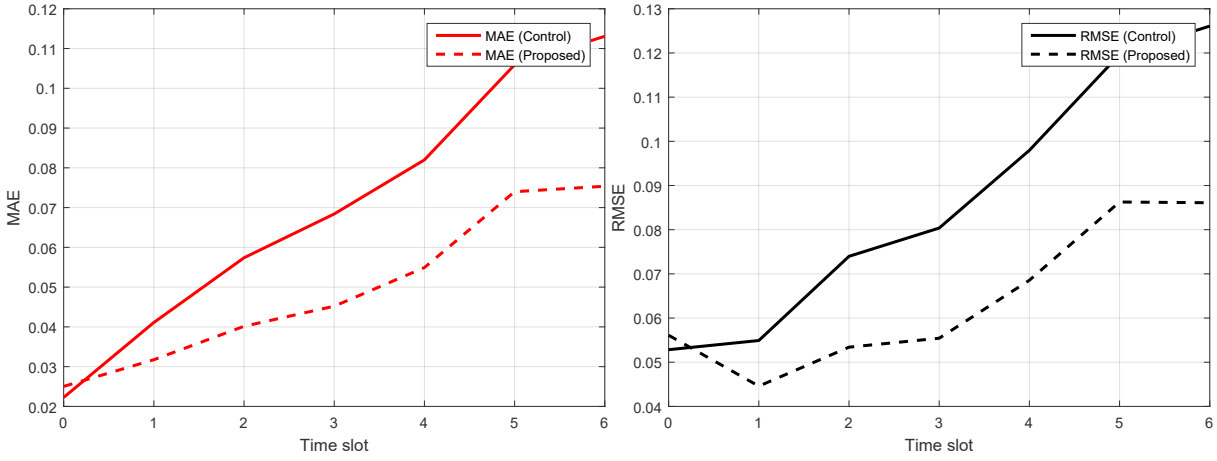
(a) (b)

Figure 6.8: Density impact on MAE and RMSE, AmazonQoS Stochastic experiment (exact matrix multiplication)



(a) (b)

Figure 6.9: Density impact on MAE and RMSE, AmazonQoS Stochastic experiment ($S = 15$)

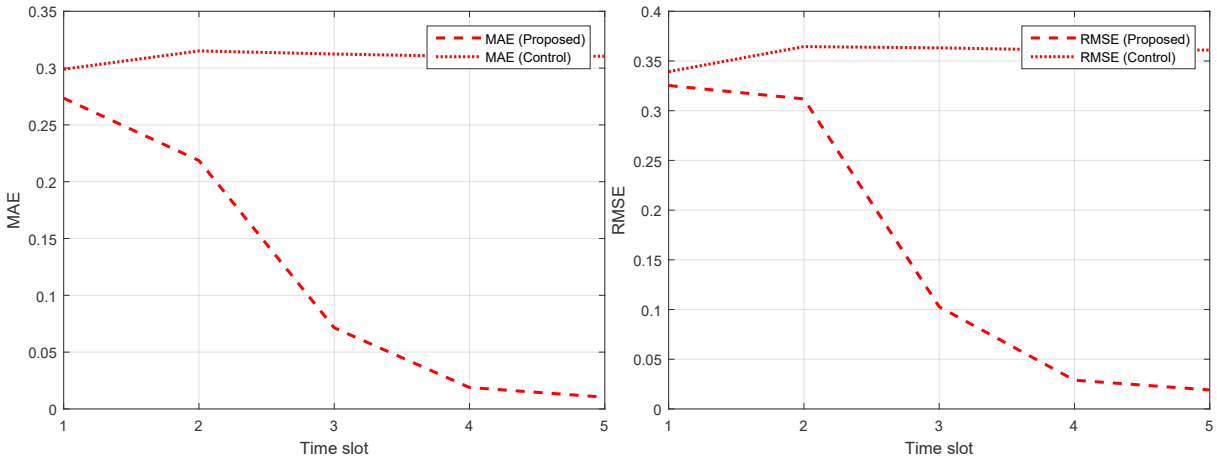


(a) (b)

Figure 6.10: AmazonQoS Changing Stochastic experiment

than Static model. The results show that the proposed model handles the updates well, i.e., successfully incorporates new records.

Fig. 6.10 depicts the results for Changing Stochastic experiment. For both models, λ_1 and λ_2 were set according to (3.32), and K_1, K_2 were again maximal ($K_1 = M$, $K_2 = N$) to allow the model to take all updates into account. The time slots (x-axis) were numbered 0-6. The results show that the proposed model adapts well to the changing environment by incrementing weights of the more recent records, since the control model (without the weight increment) is significantly less accurate (i.e., for slot 5, Control achieves RMSE of 0.120 and MAE of 0.106 while Proposed achieves RMSE of 0.086 and MAE of 0.074) and the differences increases over time. Notice that the Control model is better in the first time slot (i.e., for slot 0, Control achieves RMSE of 0.053 and MAE of 0.022 while Proposed achieves RMSE of 0.056 and MAE of 0.025): this is because the environment does not change within this slot.



(a) (b)

Figure 6.11: AmazonQoS Scalability experiment, accuracy

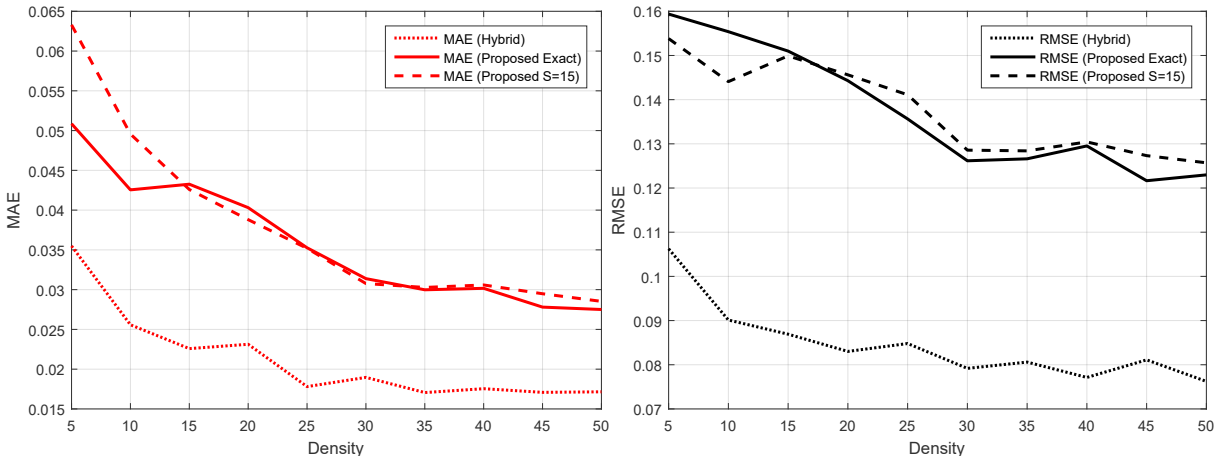
Fig. 6.11 depicts the results for Scalability experiment. Again, to allow the model to take all updates into account, λ_1 and λ_2 were set according to (3.32), and K_1, K_2 were maximal ($K_1 = M$, $K_2 = N$). The time slots (x-axis) were numbered 1-5. The results in Fig. 6.11 show that, as expected, the proposed model is more and more accurate when more and more updates are being received: i.e., in time slot 3 (10^4 updates and 10^4 predictions) MAE and RMSE are 0.071 and 0.103, respectively.

6.2.5 Results for the Failure Probability Dataset

For the FP dataset (150 users, 100 services) we have performed the Fixed Value Support experiment. The dataset is very specific: most values (82.27%) in the user-item matrix are zero.

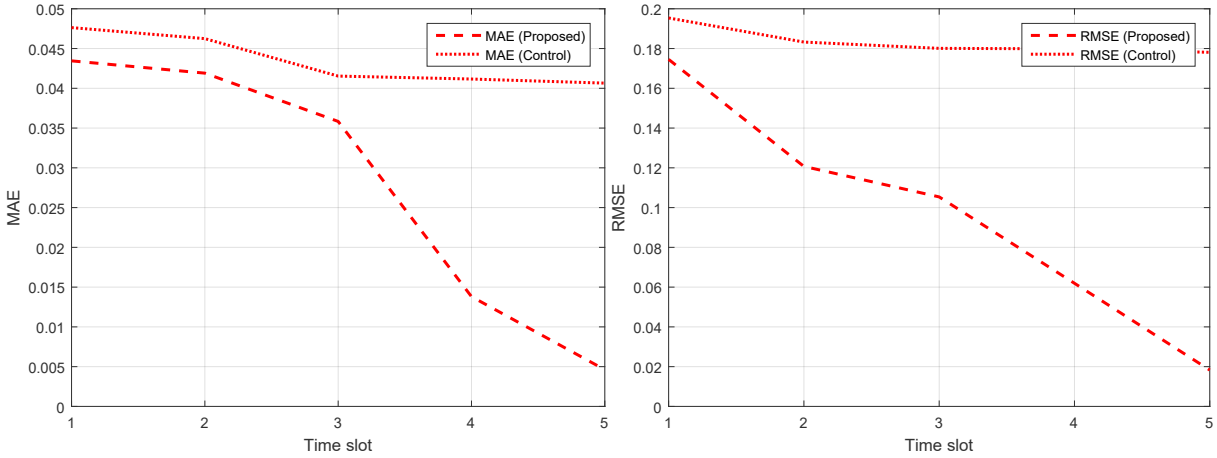
Fig. 6.12 depicts the results for Fixed Value Support experiment for the proposed model and Hybrid. The parameters (λ_1, K_1, K_2) were $(0.3, 14, 15)$ for Hybrid and $(0.8, 29, 1)$ for the proposed model (Exact and $S = 15$). The Exact and $S = 15$ model give similar results, with Exact being slighly more accurate, but both models are outperformed by Hybrid (i.e., at density of 20% Hybrid achieves RMSE of 0.083 and MAE of 0.023, while $S = 15$ achieves RMSE of 0.146 and MAE of 0.039). This means that the standard formula for *UPCC* and *IPCC* (see Eq. (3.26)) is more suited for this specific dataset than the formula used by the proposed model (see Eq. (3.28)).

Fig. 6.13 depicts the results for Scalability experiment. Again, to allow the model to take all updates into account, λ_1 and λ_2 were set according to (3.32), and K_1, K_2 were maximal ($K_1 = M$, $K_2 = N$). The time slots (x-axis) were numbered 1-5. The results in Fig. 6.13 show that, as expected, the proposed model is more and more accurate when more and more updates are being received: i.e., in time slot 3 (10^4 updates and 10^4 predictions) MAE and RMSE are 0.036 and 0.105, respectively.



(a) (b)

Figure 6.12: Density impact on MAE and RMSE, FP experiment



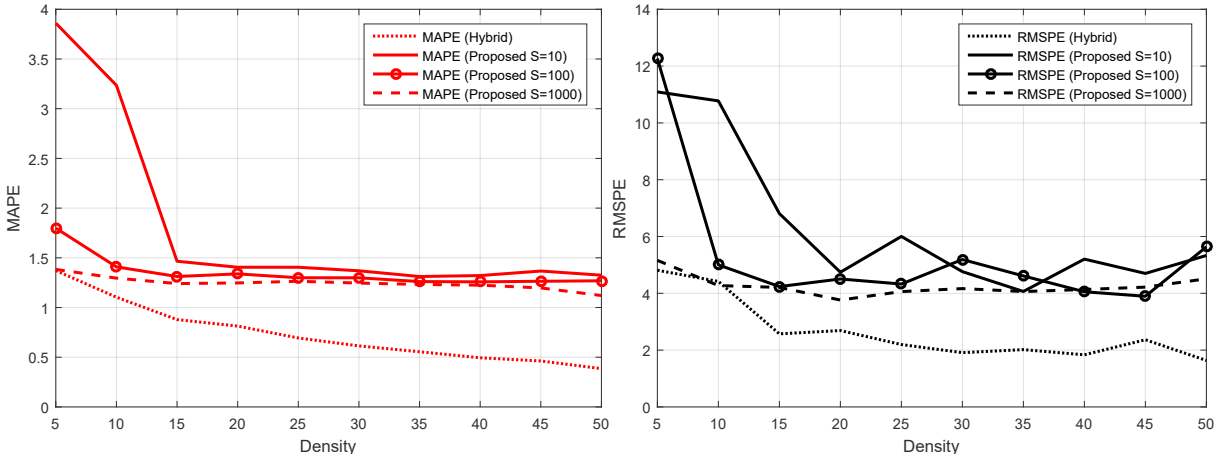
(a) (b)

Figure 6.13: FP Scalability experiment, accuracy

6.2.6 Results for Extended RT and TP Datasets

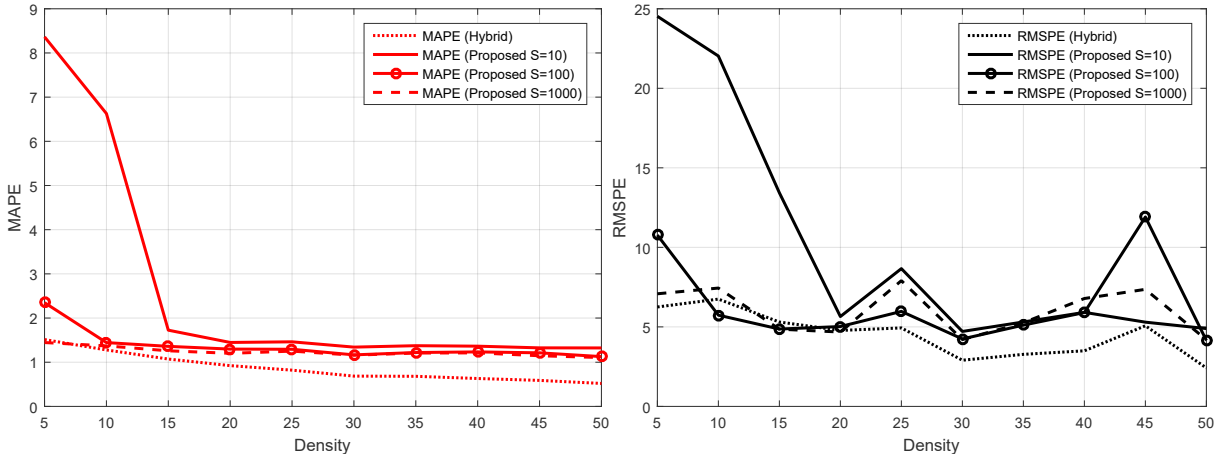
For the Extended RT and TP datasets (5763 users, 5825 services), obtained by artificially expanding the RT and TP datasets as described in Sect. 6.2.1, we have performed the Fixed Value Support experiment. However, unlike the previous Fixed Value Support experiments, we have not predicted all unknown values in the user-item matrix, but only 25000 randomly selected ones, since the size of the matrix implies large prediction times (see Sect. 6.2.7 for details). For the Hybrid and the proposed model, the parameters (λ_1, K_1, K_2) were $(0.9, 15, 70)$ for both datasets.

The results are depicted by Figs. 6.14 and 6.15. In the Extended RT experiment (Fig. 6.14), we see that Hybrid usually outperforms all proposed models: i.e., at density of 15%, Hybrid achieves (MAPE, RMSPE) of $(0.879, 2.573)$, while the proposed models $S = 10$, $S = 100$, and $S = 1000$ achieve $(1.465, 6.809)$, $(1.312, 4.238)$, and $(1.241, 4.208)$, respectively. In the



(a) (b)

Figure 6.14: Density impact on MAPE and RMSPE, Extended RT experiment



(a) (b)

Figure 6.15: Density impact on MAPE and RMSPE, Extended TP experiment

Extended TP experiment (Fig. 6.15), Hybrid generally outperforms the proposed models, which come close to Hybrid in RMSPE at densities $10-20\%$: i.e., at density of 15%, Hybrid achieves (MAPE, RMSPE) of $(1.072, 5.307)$, while the proposed models $S = 10$, $S = 100$, and $S = 1000$ achieve $(1.726, 13.431)$, $(1.359, 4.865)$, and $(1.259, 4.856)$, respectively. We see that in both experiments there are oscillations in RMSPE behavior, which is explained by high variance in the dataset values.

6.2.7 Time Complexity and Performance Results

Let us first summarize the time complexity results from the model description. As before, let M denote the number of users and N the number of services.

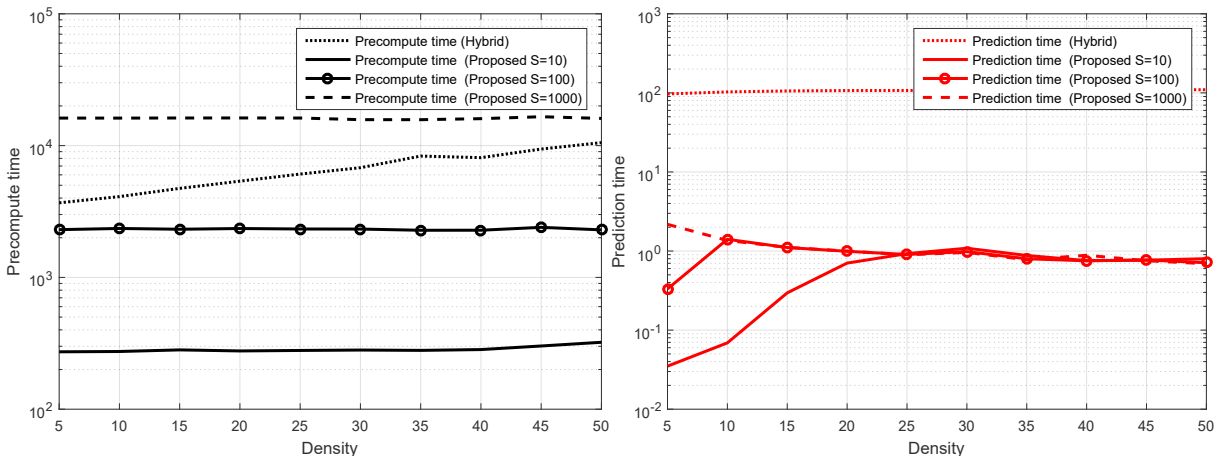
Time complexity of the precompute phase is $O(S(M^2+N^2)+M^2\log M+N^2\log N)$. The

first summand $S(M^2+N^2)$ corresponds to approximate matrix multiplication and depends on the chosen sampling constant S. The second summand $M^2 \log M + N^2 \log N$ corresponds to precomputing (sorting) lists of similar items. Time complexity of a single update equals $O(M+N)$.

We now analyze the time complexity of a single prediction. Recall that, in the prediction step for the pair (u,i), the list of users similar to user u (which is sorted by similarity) is iterated until we find K_1 users that have observed service i, or reach the end of the list. The actual number of iterations can be estimated using the density of the observed values in the user-item matrix, denoted by d. If an examined user has observed service i with the probability of d, the expected number of iterations to find K_1 observations is K_1/d. Analogously, when iterating the list of similar services, the expected number of iterations to find K_2 services observed by user u equals K_2/d. Thus, the average time complexity of a single prediction is $O((K_1+K_2)/d)$.

Duration of the precompute phase and the prediction phase were measured for the Extended RT and TP experiments, for both Hybrid and the proposed model with various values of S. However, it is important to note that even in Hybrid, we included the precompute step, i.e., we precomputed all user-service similarities based on the known values in user-item matrix, using the standard way without matrix multiplication. Otherwise, calculating predictions in Hybrid model would take too much time because of all the similarities being computed for each prediction. However, the difference in prediction times for Hybrid and the proposed model lies in the fact that Hybrid does not have lists of similar users/services precomputed; it has to sort the required similarities before finding the K_1 (or K_2) most similar users (or services), thus having the prediction complexity of $O(M \log M + N \log N)$.

Fig. 6.16 depicts the precompute and prediction times for the Extended RT experiment. (The graphs for the Extended TP experiment are almost the same since the matrix dimensions



(a) (b)

Figure 6.16: Precompute and prediction time, Extended RT experiment

are the same.) As expected, Hybrid's precompute time increases with the density, while the precompute time for the proposed model (which uses the approximate matrix multiplication) does not depend on the density. Also, as expected, higher constant S gives a higher multiplication time. Fig. 6.16a shows that Hybrid's (standard) similarities precomputation is slower than approximate matrix multiplication for $S = 100$, but faster than approximate matrix multiplication for $S = 1000$ (there are four matrix multiplications to perform, and multiplication steps use a random number generator which consumes time). The values for the proposed model are 280, 2300 and 16100 seconds (with slight variations) for $S = 10$, $S = 100$ and $S = 1000$, respectively.

As for the average prediction time, Fig. 6.16b shows that Hybrid is expectedly much slower than the proposed model ($O((K_1 + K_2)/d)$ vs. $O(M \log M + N \log N)$). At the first glance, the constant S for the approximate matrix multiplication should not affect the prediction time: why is the prediction time different for different S at low densities? To explain this, note that for $S = 10$ not many samples will be taken into account during similarity computation, which means that lists of similar entities will have few elements with strictly positive similarity to a given entity, and iteration of these lists during prediction will thus be shorter than in cases $S = 100$ or $S = 1000$ where more samples are taken into account in similarity computation. Apart from this differences at low densities, the graphs for $S = 10$, $S = 100$ and $S = 1000$ almost perfectly match (especially for $S = 100$ and $S = 1000$), as expected. For $S = 1000$, the values are $2.176s$ for $d = 0.05$ and $0.691s$ for $d = 0.50$. We see that the complexity bound $O((K_1 + K_2)/d)$ is only approximately accurate: the decrease in prediction time does not precisely match the decrease in density, which is also explained by shorter lists of similar entities at lower densities.

The results in Fig. 6.17 depict the total time taken for processing all updates and predictions in the corresponding time slot of the Scalability experiment. For the AmazonQoS dataset (50 users, 49 services) the model takes $0.505s$ for 10^4 updates and 10^4 predictions (time slot 3) and

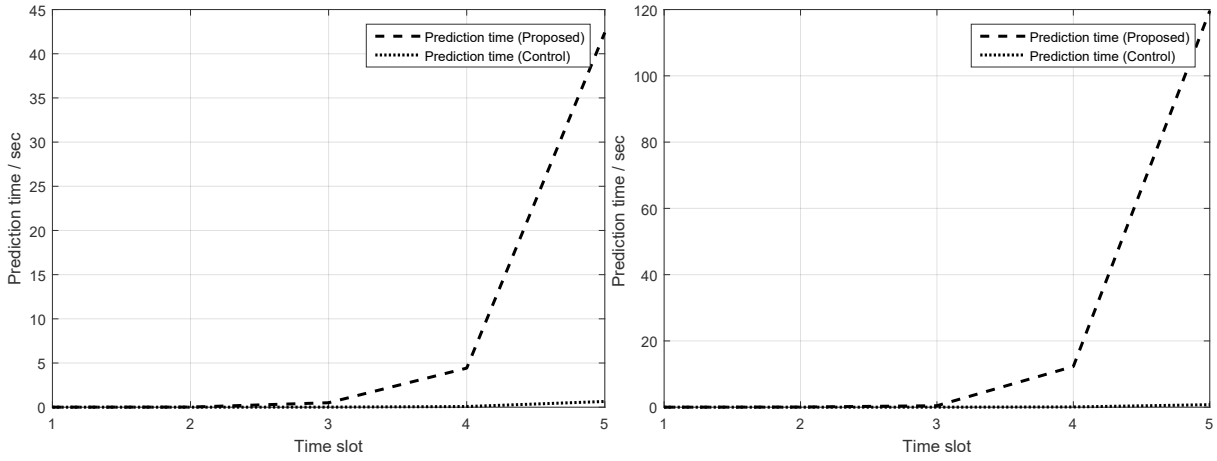


Figure 6.17: Scalability experiment, timing: a) AmazonQoS Dataset, b) FP dataset

4.422s for 10^5 updates and 10^5 predictions (time slot 4), which means that the model supports up to 10^5 events in real-time. Similarly, in the FP dataset (150 users, 100 services) the model takes 0.446s for 10^4 updates and 10^4 predictions (time slot 3) and 12.295s for 10^5 updates and 10^5 predictions (time slot 4).

6.2.8 Threats to Validity

Here we mention some possible threats to validity of the proposed model and the conducted experiments. The first threat is randomized approximate matrix multiplication, which gives slightly different results for different random seeds. The results regarding the expected accuracy of such multiplication are derived in [117].

Secondly, the data for Extended RT and TP experiments was created artificially, which means that the corresponding accuracy results are slightly less realistic than the accuracy results for other conducted experiments. However, we believe this dataset to reflect the realistic data since we used the actual RT and TP datasets to generate it, as described above. Furthermore, this does not affect the validity of time performance results on these datasets, because they mostly do not depend on actual numerical values.

6.3 Service Selection

In this section, we present the detailed evaluation results for the proposed compositional service selection model.

For experimental purposes we have used a mixture of artificial and actual data. In all experiments, three user-service QoS properties were considered: reliability, price, and response time. Also, throughput was considered as a processing capacity limit: maximal number of requests for a service instance.

Reliability values were generated artificially as

$$1 - 0.1^{\max\{\mathcal{N}(2,0.5),1\}} \in [0.9, 1\rangle, \tag{6.5}$$

where $\mathcal{N}(avg, std.dev.)$ is a normal distribution.

Price values were generated artificially as

$$\max\{0, U(-1,4)\} \in [0,4], \tag{6.6}$$

where $U(a,b)$ is a uniform distribution on $[a,b]$.

For the *response time* values we used the WS-DREAM dataset [118]. From the same dataset we used the throughput values to generate the service throughput limits: they range from 1 to

1000 with the average of 47.

Services were uniformly distributed among n tasks and there was a 50% probability that a specific user invokes a specific task; namely, $EI(C_u)_j \in \{0,1\}$. User QoS requirements were generated under three levels of difficulty: *easy* (satisfiable by taking 40%-quality services under definitions from Sect. 4.1), *medium* (satisfiable by taking 60%-quality services) and *hard* (satisfiable by taking 80%-quality services).

The execution times were measured under *Python 3* implementations[‡] on a 3.20 GHz processor with 4 cores, which means that 4 parallel processes were solving the TP tasks. Our own implementations of the proposed and all the competing algorithms were used, except in the case of mixed integer programming problems where we utilized the Pulp library [135].

The measured quantities were *accuracy* (percentage of satisfied QoS requirements), *execution time*, and *QoS improvement* which measured the average relative gain of the obtained QoS value vs. the corresponding QoS requirement over all users and QoS properties. More precisely,

$$\text{Avg. QoS improvement} = \frac{1}{Uq}\sum_{u=1}^{U}\sum_{k=1}^{q}\frac{QoS_u^k - Req_u^k}{|Req_u^k|}, \tag{6.7}$$

assuming that a higher QoS value means a higher improvement. If the QoS requirement is not satisfied ($QoS_u^k < Req_u^k$), the QoS improvement is negative. As almost all QoS requirements turn out to be satisfied ($QoS_u^k \geq Req_u^k$), the average QoS improvement is positive. For reliability, logarithms of actual values were used in all formulas.

In the experiments, we have varied the number of users, services, and tasks. Three main types of experiments were performed:

- *Single-Task Experiment*, where we tested the model proposed in Sect. 5.1 against the alternatives.
- *User-Independent-QoS experiment*, where data was generated with additional condition that it always holds that $QoS_{ui}^k = QoS_{vi}^k$ for different users u, v. The purpose of this assumption (non-personalized QoS) was to compare the proposed models with MIP-based models which depend on it.
- *General experiment* without special assumptions.

Each variation of each experiment was run 200 times with different random seeds and the results were averaged. For comparison with related work, we have chosen the recent models suitable for the corresponding service selection scenarios. The competing models were:

- *AP* ("assignment problem") [26] for the *Single-Task* scenario.
- *Clus2-MIP* and *Clus3-MIP* ("clustering + mixed integer programming") [16] for the *User-Independent-QoS* scenario where services are clustered into two or three clusters.
- *MIP* for the general mixed integer programming solution [12] in all experiments.

[‡]Implementation can be found on https://bitbucket.org/satja/rels.

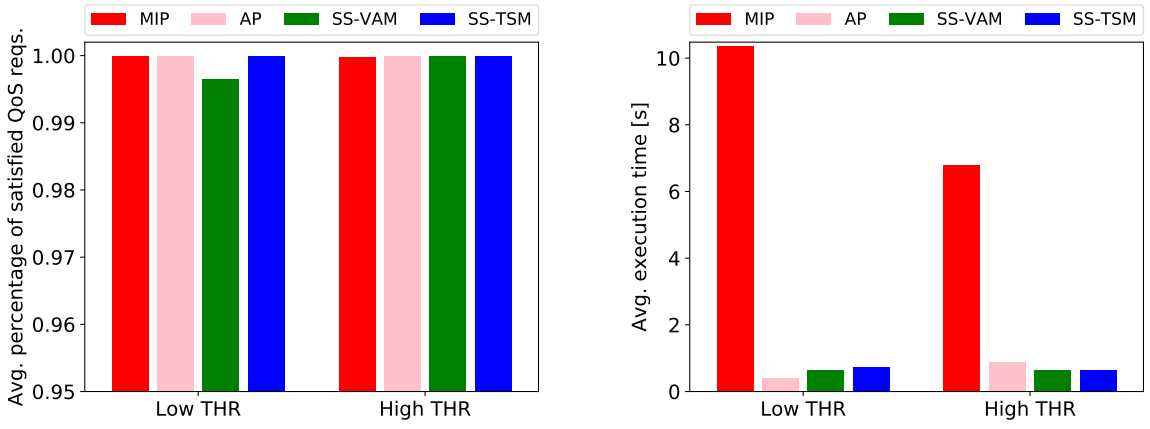
- *Greedy-MIP* as an attempt of improvement over MIP in *General* experiment, based on the idea from [15], modified and adapted to the general setting. Namely, aiming to reduce the search space of MIP, only 10 "best" services for each task are considered, and if the solution is not found, another 10 are added to consideration after each unsuccessful iteration of MIP. The priority of adding a service is defined by ranking the services by each QoS property for each user, and then adding up the ranking positions for each service.
- *SS-VAM* and *SS-TSM*, our proposed models.

6.3.1 Single-Task Experiment

Two basic variations of the Single-Task experiment both contained 500 users and 100 services with medium QoS difficulty, but differed in the service throughput limits, which were either *low THR* (1-50) in the first variation, or *high THR* (50-1000) in the second variation. This was done to differentiate between cases of higher and lower efficiency of the alternative AP approach which creates as many virtual instances as the total THR. QoS improvement was not calculated in this experiment due to the nature of problem reduction in this case, which takes the number of satisfied QoS requirements as cost (as explained in Sect. 5.1), instead of the actual QoS values or their utility.

The results in Fig. 6.18a confirm that all approaches are optimal except for SS-VAM which is slightly suboptimal, satisfying 99.7% requirements on average. Results in Fig. 6.18b show that, in terms of computational efficiency, the proposed approaches equally outperform AP in case of high THR (by 28.6%), and the opposite is true in case of low THR where SS-VAM is by 55.6% slower than AP. Due to its optimality, SS-TSM is slower than SS-VAM by 17.8% under low THR. All approaches outperform MIP by an order of magnitude.

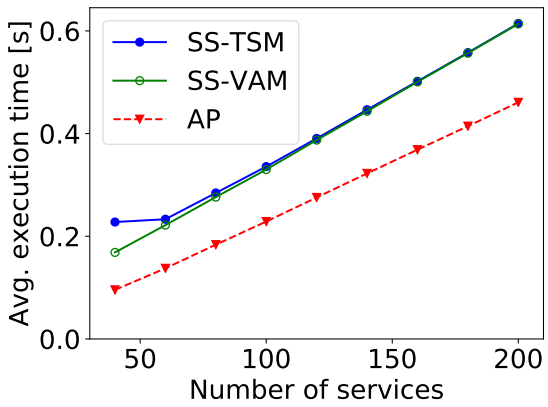
Fig. 6.19 shows timing results for variations of this experiment where number of services



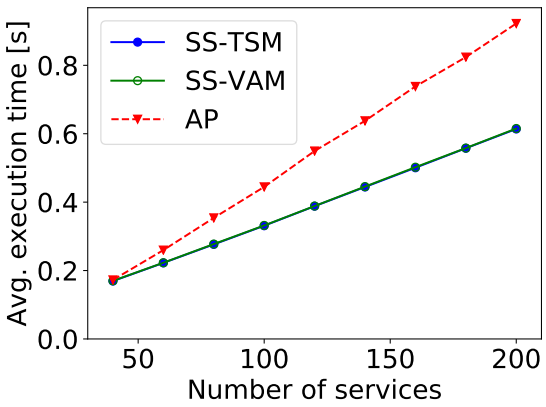
(a) Accuracy for low/high service throughputs

(b) Efficiency for low/high service throughputs

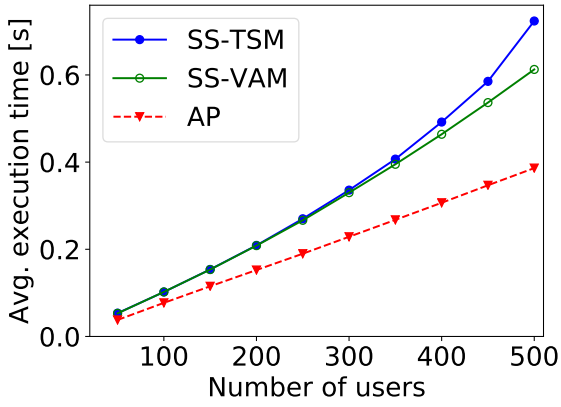
Figure 6.18: Single-Task Experiment: basic results



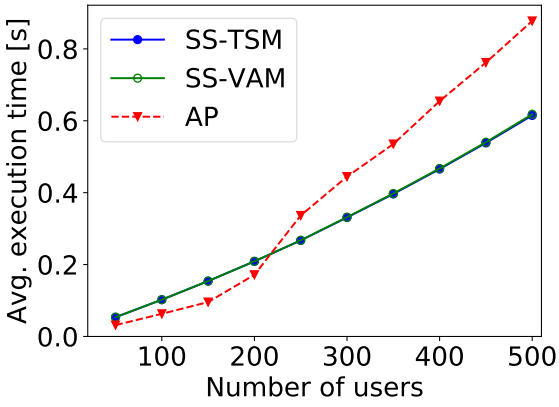
(a) Time vs. services, low THR



(b) Time vs. services, high THR



(c) Time vs. users, low THR



(d) Time vs. users, high THR

Figure 6.19: Single-Task Experiment: efficiency vs. number of services/users

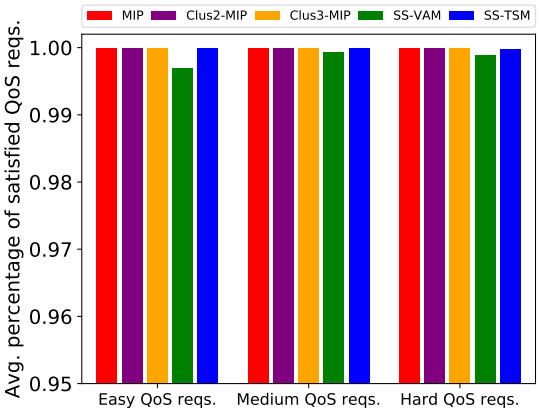
varies from 40-200 with 300 users (Figs. 6.19a-6.19b), or the number of users varies from 50-500 with 100 services (Figs. 6.19c-6.19d). All approaches show linear growth of execution time when the number of users/services grows linearly. In particular, SS-VAM shows a 3.63x increment in execution time for a 5x increment in the number of services, and a 12.4x increment in execution time for a 10x increment in the number of users. The proposed algorithms outperform AP when THR is high and problem dimensions are not small (at least 40 services for 300 users, and at least 200 users for 100 services). In Fig. 6.19c, SS-TSM shows a slightly non-linear growth for low THR and more than 400 users.

The difference in results between the cases of low and high THR stems from the fact that high THR significantly affects the execution time of AP because that model creates an additional amount of THR_i virtual services, while THR has no such influence on the proposed SS-VAM and SS-TSM models.

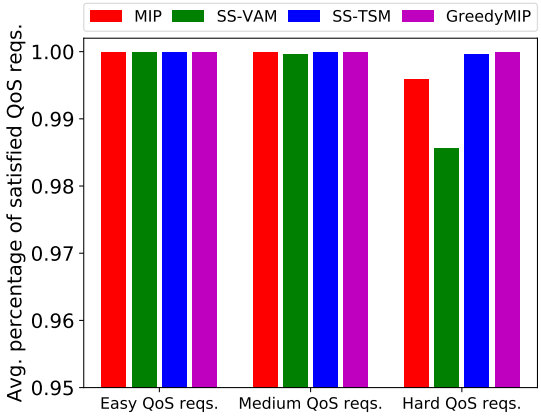
6.3.2 User-Independent-QoS Experiment

Basic variations of the User-Independent-QoS experiment contained 50 users, 4 tasks, and 200 services. The QoS values depended only on a service; in other words, for a fixed service $i \in I$ and a QoS property $k \in \{1,\ldots,q\}$, the value QoS_{ui}^{k} was equal for all users $u \in U$. As described earlier, the generated QoS requirements differed in difficulty (easy, medium, hard).

The results in Fig. 6.20a confirm that all approaches are optimal in terms of the QoS requirement satisfaction except for SS-VAM which is slightly suboptimal, satisfying 99.8% requirements on average. Results in Fig. 6.21a show that, in terms of time efficiency, the proposed approaches significantly outperform MIP, Clus2-MIP, and Clus3-MIP. On average, SS-TSM takes 14.2x less time than MIP, 22.7x less time than Clus2-MIP, and 22.5x less time than Clus3-MIP. Also, SS-TSM outperforms SS-VAM, which can be explained by noting that even though

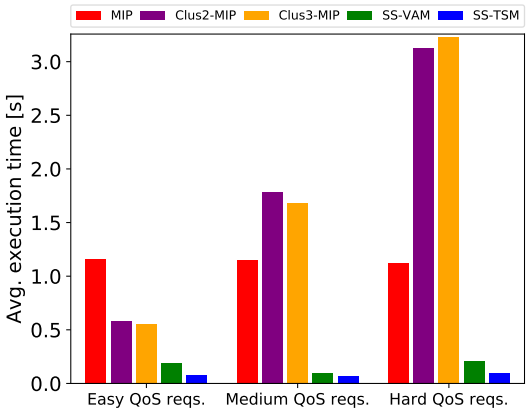


(a) User-Independent-QoS Experiment

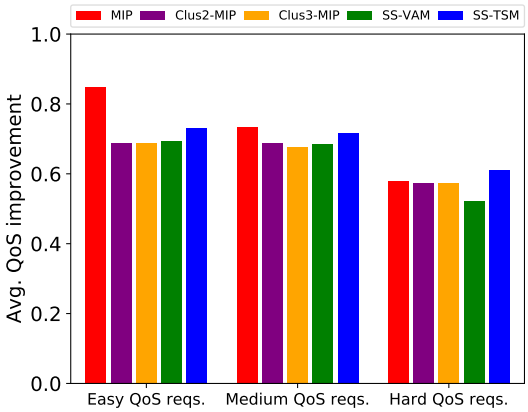


(b) General Experiment

Figure 6.20: Accuracy vs. difficulty: results



(a) Efficiency vs. difficulty



(b) QoS improvement vs. difficulty

Figure 6.21: User-Independent-QoS Experiment: results

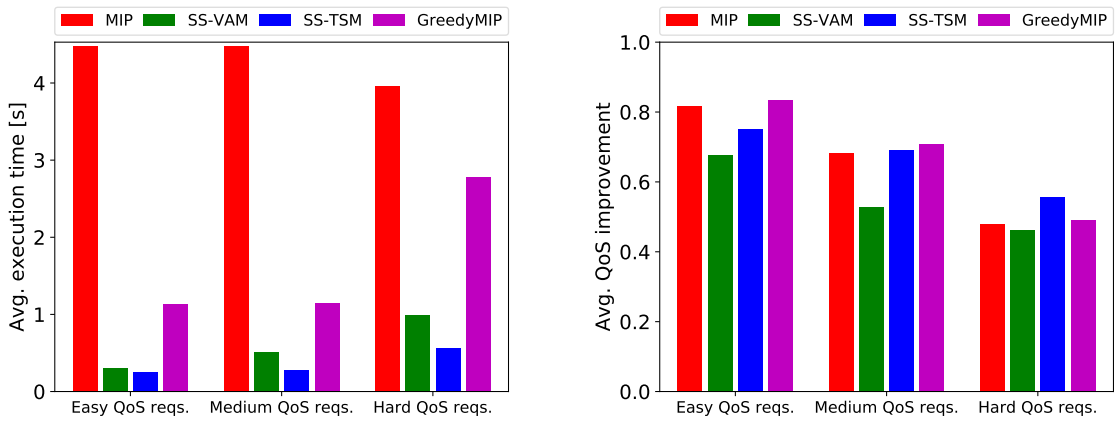
SS-VAM takes less time per iteration, SS-TSM takes less iterations, i.e., there is no need to reiterate the algorithm with adjusted utility cost if the solution by SS-TSM is already successful in terms of global QoS requirements. We note that, in this experiment, SS-TSM dominates SS-VAM in both accuracy and efficiency. Fig. 6.21b shows that the QoS improvement did not significantly differ across models. MIP gave most improvement, 3.5% higher than SS-TSM on average, while Clus2-MIP and Clus3-MIP gave less improvement than SS-TSM.

The difference in the effect of easy, medium, and hard QoS requirements can be seen in Fig. 6.21a. The execution time of MIP does not depend on the QoS requirement difficulty: as a generic approach, it performs equally fast for all difficulties. For easy QoS requirements, we see that MIP is indeed made faster by Clus2-MIP (by a factor of 2.0x) and Clus3-MIP (by a factor of 2.1x). For medium and hard QoS requirements, MIP is more efficient than variations of Clus-MIP because of the higher number of iterations needed by Clus-MIP to find a solution. It is important to note that the execution time of the proposed approach shows no significant dependence on the QoS requirement difficulty: SS-TSM is only 22% slower under hard (vs. easy) QoS requirements.

6.3.3 General Experiment

Basic variations of the General experiment contained 100 users, 8 tasks, and 300 services, differing in QoS difficulty (easy, medium, hard). The results of basic variations are shown in Fig. 6.20b and Fig. 6.22. As in the previous experiments, the proposed SS-TSM approach dominates the proposed SS-VAM approach in all measures (with the same explanation). Also, we see that Greedy-MIP approach dominates MIP in all measures, so the focus can be put to the comparison between SS-TSM and Greedy-MIP.

The results in Fig. 6.20b confirm success of the approaches in terms of the QoS requirement



(a) Efficiency vs. difficulty

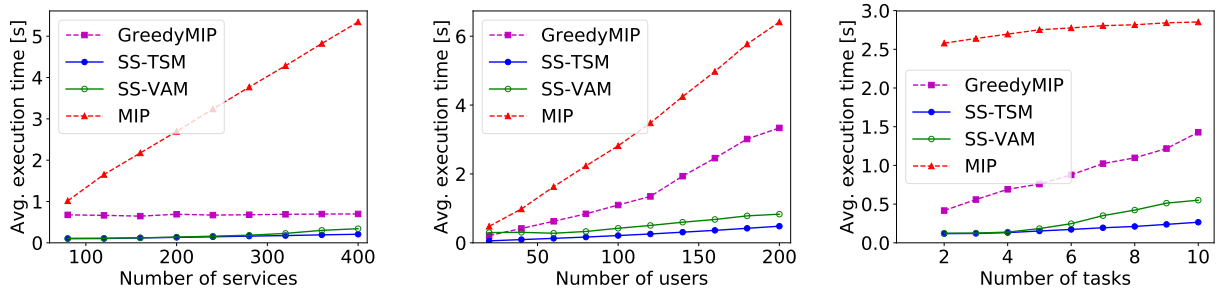
(b) QoS improvement vs. difficulty

Figure 6.22: General Experiment: basic results

satisfaction. Results in Fig. 6.22a show that, in terms of time efficiency, SS-TSM outperforms Greedy-MIP by a factor of 4.15–5.0x. Fig. 6.22b shows that QoS improvement did not significantly differ across models: on average, Greedy-MIP and SS-TSM gave similar improvements of $\approx 67\%$.

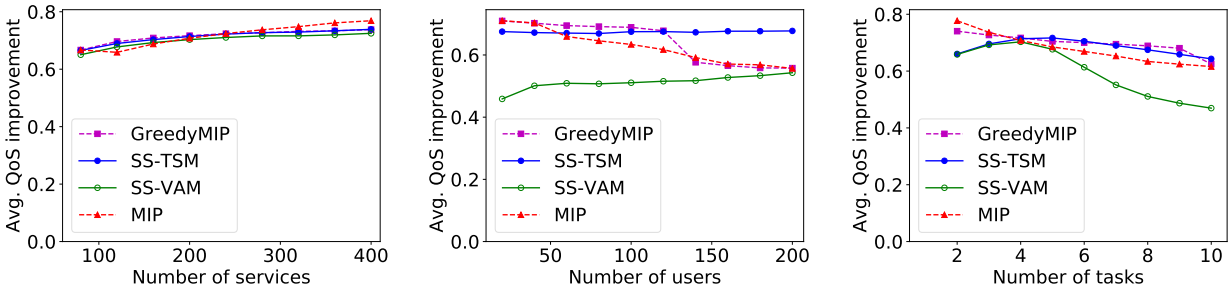
The difference in the effect of easy, medium, and hard QoS requirements can be seen in Fig. 6.22a. The proposed approach is more significantly affected by the requirement difficulty than in the User-Independent-QoS Experiment: SS-TSM is 2.22x slower under hard (vs. easy) QoS requirements. However, under hard requirements it is still 9.4x faster than MIP and 5.0x faster than Greedy-MIP. Fig. 6.22b shows that higher QoS requirement difficulty (hard vs. easy) affects the obtained QoS values in all approaches, reducing the QoS improvement for Greedy-MIP from 84% to 49%, and for SS-TSM from 75% to 56%, showing that reduction for SS-TSM is less significant. Compared to MIP and Greedy-MIP, the proposed SS-TSM gives a slightly lower QoS improvement under easy requirements, but higher under hard requirements.

More detailed variations of the General experiment are depicted in Figs. 6.23-6.24. Here, the number of services varied from 80 to 400, with 4 tasks and 100 users; the number of users varied from 20 to 200, with 8 tasks and 200 services; and the number of tasks varied from 2 to 10, with 100 users and 200 services. Fig. 6.23 shows that the execution time for the proposed



(a) Efficiency vs. services (b) Efficiency vs. users (c) Efficiency vs. tasks

Figure 6.23: General Experiment: time vs. number of services/user/tasks



(a) QoS improvement vs. services (b) QoS improvement vs. users (c) QoS improvement vs. tasks

Figure 6.24: General Experiment: QoS improvement vs. number of services/user/tasks

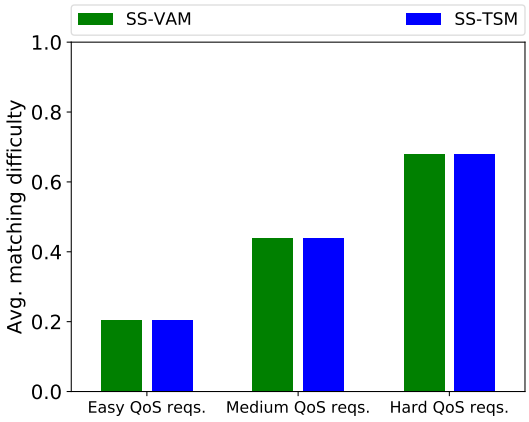
approach grows linearly, but slowly, with respect to the linear growth of services/users/tasks. In particular, for SS-TSM the 5x increment in the number of services induced the 2.25x increment in execution time; the 10x increment in the number of users induced the 8.0x increment in execution time; and the 5x increment in the number of tasks induced the 2.1x increment in execution time. The execution time for Greedy-MIP grows faster, with the respective increments in execution time equal to 15.9x (for 10x more users) and 3.4x (for 5x more tasks), with the exception of a negligible increment when the number of services is increased, owing to the fact that it greedily considers only the best of them. Finally, Fig. 6.24 shows that QoS improvement for SS-TSM does not significantly depend on the number of services, users, or tasks, while for the Greedy-MIP the improvement is slowly reduced by increasing the number of users or tasks.

6.3.4 Matching Difficulty and Results by Iteration

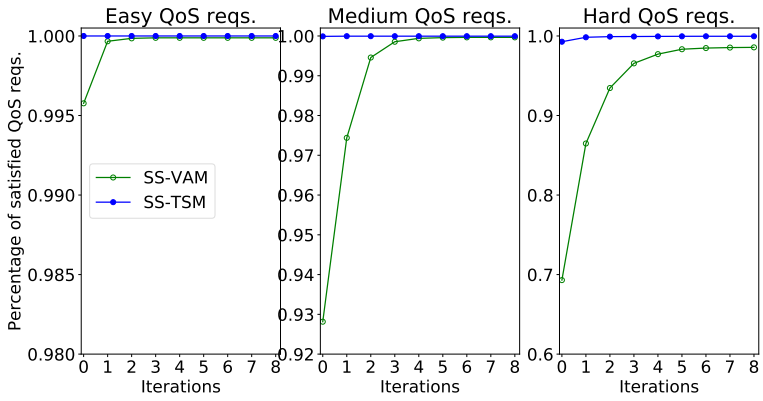
For the General experiment, which is the most representative for the scope of the proposed approach, we have measured additional output for the basic variation of the experiment, shown in Fig. 6.25. Namely, Fig. 6.25a shows how the average matching difficulty MD, which is the main ingredient of the proposed utility cost, (expectedly) grows with the increasing difficulty of the input QoS requirements. Figs. 6.25b and 6.25c show the behavior of the proposed methods across iterations, demonstrating that the algorithm quickly converges and that the actual number of iterations is usually lower than the maximal number, especially in the SS-TSM model where a single iteration is often sufficient to satisfy all QoS requirements. This explains the advantage of SS-TSM over SS-VAM in terms of total execution time.

6.3.5 Threats to Validity

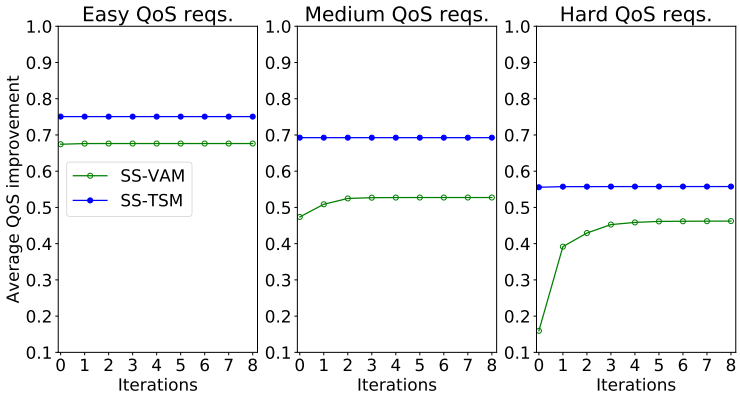
Regardless of our effort to make the test data realistic, the numerical results partly depend on the values generated artificially for testing purposes. They will certainly change in different environments, along with the actual implementation of the approach, which will make use of a more specific and more efficient hardware. Still, the experiments make their point, which was to give relations between the competing models in reasonably constructed service selection scenarios.



(a) *MD* vs. input QoS difficulty



(b) Accuracy vs. iterations



(c) QoS improvement vs. iterations

Figure 6.25: General Experiment: matching difficulty and results by iterations

Chapter 7

Conclusion

As cloud computing becomes the prevailing aspect of software engineering, paradigms such as Service-Based Systems (SBSs) or Software as a Service (SaaS) are coming into focus. They are based on principles defined in Service-Oriented Architecture (SOA) and assume a number of cloud services responding to numerous client requests. Selecting the actual service instance for request can be an issue, if requirements for multiple Quality of Service (QoS) properties need to be satisfied for many users simultaneously. The problem becomes more complex if we take into account the compositeness of users' applications, which consist of many tasks with non-deterministic execution plan, where QoS properties are calculated over the whole composition. The existing approaches for this problem lack either efficiency or generality.

Therefore, the main goal of this thesis was to propose a fast heuristic model for multi-criteria service selection, designed for multi-user non-deterministic composite workflows with the goal of satisfying all, or as many as possible, of the given QoS requirements. To develop such a method, some essential tools were developed, such as probabilistic formulas for estimating QoS in non-deterministic service compositions. Another contribution was the real-time adaptive QoS prediction model, which can be used before service selection to provide its QoS input.

Our service selection model was motivated by the need of creating a method that is, unlike previous works, both efficient (polynomial time complexity) and very general (multi-user, multi-task composition with non-deterministic branching, user-dependent QoS). Namely, the model considers multiple users, each having a multi-task execution plan with requirements of minimal (or maximal) performance values for several QoS parameters. The execution plans allow for all standard compositional structures (Sequence, Parallel, Conditional Branch, and Loop), taking into account probabilities in case of branching and loops. The model is responsible for selecting the service instances to be invoked for all tasks, considering their respective throughput limits, as well as QoS values which depend on the user-service pairs. The goal was to satisfy all (or as many as possible) of users' QoS requirements.

As a result, we have proposed a fast general service selection heuristic, aimed at satisfy-

ing multi-user multi-criteria QoS requirements. It can handle complex service compositions, differing across users, based on a novel probabilistic QoS compositional model. It takes into account personalized QoS values which depend on a service *and* a user, which is a realistic scenario that is missing in a significant part of related work. The proposed service selection method reduces the problem to several independent transportation problems with iterative solution improvements, using a global-aware utility cost which is based on expected compositional QoS and the novel concept of matching difficulty. Two variations were proposed: SS-TSM and SS-VAM, differing in their approach to solve individual transportation problems.

Efficiency, more than accuracy, stays in the focus of most current cloud computing efforts dealing with an increasing amount of users and service instances. Therefore, it was the most important measure in our model evaluation. We have performed extensive experiments covering both special and more general cases, where the proposed methods were compared with both optimal service selection approaches (based on mixed integer programming) and their heuristic improvements when they are applicable. Our experiments demonstrate that the proposed SS-TSM model is the dominating approach in most experiments, mainly because of a significant reduction in execution time. A partial exception is a single-task scenario, where the proposed (slightly less accurate) SS-VAM approach can be faster in cases of high service throughput limits, while the AP approach from literature (based on reduction to assignment problem) can be faster when throughputs are lower. To conclude, apart from being more general than the existing approaches, the proposed method (namely, SS-TSM) also turns out to be more efficient than the alternatives (up to 5x faster).

In order to perform an effective service selection, the QoS values must be either known or predicted with reasonable accuracy. Therefore, as another contribution in this dissertation, we presented a QoS prediction model as a real-time support for selection of atomic "black-box" service candidates based on their QoS properties while constructing composite applications. The proposed approach satisfies the following main requirements: fast and accurate prediction of QoS values, and adaptability with respect to environment changes. The model precomputes the similarities between users and services using approximate matrix multiplication to reduce the time complexity. When calculating a prediction for a user-service pair, the model considers similar users and services, but enhances the prediction accuracy by incorporating the number of observed records. Time complexity is further reduced by storing the lists of similar users and services which are updated in real-time. The model adapts to the changing environment: newer records are set to have greater influence on predictions.

We have quantitatively compared the proposed approach with the existing state-of-the-art memory-based collaborative filtering approach considering both prediction's *accuracy* and *time performance*. In order to evaluate the performance of our approach, we have collected publicly available web services QoS datasets. We have designed and conducted a series of ex-

periments that challenge different aspects of competing prediction approaches. The extensive evaluation results confirm that our approach supports the requirements set during the design process. The evaluation results show that our approach can produce predictions faster than standard UPCC/IPCC/Hybrid approaches while keeping the prediction accuracy very close or even better than the state-of-the-art. For instance, the proposed approach with random sample size in approximate multiplication set to $S = 15$, achieves 10.7% better RMSE value on average than the Hybrid approach while measuring on the *AmazonQoS* dataset. Furthermore, the experiments that simulate changing dynamic environment demonstrate runtime adaptability feature of our model. In such conditions, our approach obtains 25.6% better RMSE value that the competing approach. Finally, the conducted scalability experiments demonstrate that our approach can support in real-time up to 10^5 update and prediction requests.

The proposed approach is based on approximative matrix multiplication technique, which is flexible in a sense that its performance can be adjusted by parameterizing the random sample size S. This flexibility feature is also preserved in our approach. We can produce more accurate predictions by increasing the random sample size, which in turn results in worse time performance. On the other hand, we can enhance prediction time by decreasing the random sample size, which will result in less accurate predictions. Due to its flexibility, our approach can balance between two opposite requirements: *accuracy* and *efficiency*. To conclude, the proposed approach with its flexible design can be applied in different environments.

Since our prediction model is based on collaborative filtering, as a side contribution, we have proposed global correlation measures for a collaborative filtering dataset. Recommender systems based on collaborative filtering (CF) rely on datasets containing users' taste preferences for various items, and accuracy of various prediction approaches depends on the amount of similarity between users and items in a dataset. As a heuristic estimate of this data quality aspect, which could serve as an indicator of the prediction ability, we have defined the Global User Correlation Measure (GUCM) and the Global Item Correlation Measure (GICM) of a dataset containing known user-item ratings. The measures can be used to quickly estimate whether a dataset is suitable for collaborative filtering and whether we can expect high prediction accuracy of user-based or item-based CF approaches. The proposed measures range from 0 to 1 and describe quality of a dataset regarding the user-user and item-item similarities: a higher measure indicates more similar pairs and better prediction ability.

The proposed correlation measures (GUCM, GICM, and their average GCM) have been experimentally shown to satisfy the desired requirements. Namely, they do correlate with the amount of user-user and item-item similarities, as well as with the accuracy of standard prediction models. They approach a value of 1 on datasets with equal items and a value of 0 on datasets with random ratings, and negatively correlate with the number of natural clusters of similar users/items. Since the proposed measures can be computed efficiently, in time propor-

tional to the number of known ratings, they can be useful when deciding whether to apply CF on a given dataset – the results suggest that a GCM value higher than 0.14 tends to imply a high prediction accuracy. In case one needs to choose whether to include some additional or questionable records in a dataset, GCM can be used to estimate if the dataset quality would increase or decrease with a considered change. GCM can also be used to choose or differentiate between multiple datasets in various educational and research purposes, such as evaluating a new CF model.

Chapter 8

Appendix

8.1 Proof of Eq. (3.3)

Here we prove Eq. (3.3) which gives the probability that two random values from the uniform distribution on the interval of length D differ by less than T. If we rescale the interval to become $[0,1]$, the threshold difference should also be rescaled by factor $1/D$, i.e., we are looking for the probability that the difference between two random values in $[0,1]$ is less than T/D.

More precisely, let us assume that X and Y are independent uniformly distributed random variables from $[0,1]$. We are looking for the probability density function of their difference $X - Y \in [-1,1]$, which is given by the convolution of the corresponding separate density functions:

$$\begin{aligned} f_{X-Y}(x) &= (f_X * f_{-Y})(x) = \int f_X(y) f_{-Y}(x-y)\,dy \\ &= \int 1_{[0,1]}(y)\,1_{[-1,0]}(x-y)\,dy = \int_0^1 1_{[-1,0]}(x-y)\,dy = \int_{x-1}^{x} 1_{[-1,0]}(t)\,dt \\ &= \min\{x,0\} - \max\{x-1,-1\} = \min\{x,0\} - (\max\{x,0\} - 1) \\ &= 1 - |x|. \end{aligned}$$

Therefore, $\Pr[-T/D \le X - Y \le T/D]$ equals

$$\begin{aligned} & \textstyle\int_{-T/D}^{T/D} (1-|t|)\,dt \\ = \;& 2\textstyle\int_0^{T/D} (1-t)\,dt \\ = \;& 2\left(x - \frac{x^2}{2}\right)\Big|_0^{T/D} \\ = \;& 2\left(\frac{T}{D} - \frac{T^2}{2D^2}\right) \\ = \;& \frac{T(2D-T)}{D^2}, \end{aligned}$$

which proves Eq. (3.3).

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] Vrba, P., Mařík, V., Siano, P., Leitão, P., Zhabelova, G., Vyatkin, V., Strasser, T., "A review of agent and service-oriented concepts applied to intelligent energy systems", IEEE Transactions on Industrial Informatics, Vol. 10, No. 3, Aug 2014, str. 1890-1903.

[2] Papazoglou, M. P., Traverso, P., Dustdar, S., Leymann, F., "Service-oriented computing: State of the art and research challenges", Computer, Vol. 40, No. 11, Nov. 2007, str. 38–45, dostupno na: http://dx.doi.org/10.1109/MC.2007.400

[3] Duan, Q., Yan, Y., Vasilakos, A. V., "A survey on service-oriented network virtualization toward convergence of networking and cloud computing", IEEE Transactions on Network and Service Management, Vol. 9, No. 4, December 2012, str. 373-392.

[4] Hu, X., Chu, T. H. S., Leung, V. C. M., Ngai, E. C. H., Kruchten, P., Chan, H. C. B., "A survey on mobile social networks: Applications, platforms, system architectures, and future research directions", IEEE Communications Surveys Tutorials, Vol. 17, No. 3, thirdquarter 2015, str. 1557-1581.

[5] Chen, W., Paik, I., Hung, P. C. K., "Constructing a global social service network for better quality of web service discovery", IEEE Transactions on Services Computing, Vol. 8, No. 2, March 2015, str. 284-298.

[6] Cardoso, J., Sheth, A., Miller, J., Arnold, J., Kochut, K., "Quality of service for workflows and web service processes", Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 1, No. 3, 2004, str. 281 - 308, dostupno na: http://www.sciencedirect.com/science/article/pii/S157082680400006X

[7] Liu, Y., Ngu, A. H., Zeng, L. Z., "Qos computation and policing in dynamic web service selection", in Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers &Amp; Posters, ser. WWW Alt. '04. New York, NY, USA: ACM, 2004, str. 66–73, dostupno na: http://doi.acm.org/10.1145/1013367.1013379

[8] Cardellini, V., Casalicchio, E., Grassi, V., Presti, F. L., "Flow-based service selection for-

web service composition supporting multiple qos classes", in IEEE International Conference on Web Services (ICWS 2007), July 2007, str. 743-750.

[9] Avizienis, A., Laprie, J.-C., Randell, B., Landwehr, C., "Basic concepts and taxonomy of dependable and secure computing", Dependable and Secure Computing, IEEE Transactions on, Vol. 1, No. 1, Jan 2004, str. 11-33.

[10] Xu, Z., Martin, P., Powley, W., Zulkernine, F., "Reputation-enhanced qos-based web services discovery", in IEEE International Conference on Web Services (ICWS 2007), July 2007, str. 249-256.

[11] Tian, M., Gramm, A., Naumowicz, T., Ritter, H., Freie, J. S., "A concept for qos integration in web services", in Fourth International Conference on Web Information Systems Engineering Workshops, 2003. Proceedings., Dec 2003, str. 149-155.

[12] He, Q., Han, J., Chen, F., Wang, Y., Vasa, R., Yang, Y., Jin, H., "Qos-aware service selection for customisable multi-tenant service-based systems: Maturity and approaches", in 2015 IEEE 8th International Conference on Cloud Computing, June 2015, str. 237-244.

[13] Klein, A., Ishikawa, F., Honiden, S., "Towards network-aware service composition in the cloud", in Proceedings of the 21st International Conference on World Wide Web, ser. WWW '12. New York, NY, USA: ACM, 2012, str. 959–968, dostupno na: http://doi.acm.org/10.1145/2187836.2187965

[14] Alrifai, M., Risse, T., "Combining global optimization with local selection for efficient qos-aware service composition", in Proceedings of the 18th International Conference on World Wide Web, ser. WWW '09. New York, NY, USA: ACM, 2009, str. 881–890, dostupno na: http://doi.acm.org/10.1145/1526709.1526828

[15] He, Q., Han, J., Yang, Y., Grundy, J., Jin, H., "Qos-driven service selection for multi-tenant saas", in 2012 IEEE Fifth International Conference on Cloud Computing, June 2012, str. 566-573.

[16] Wang, Y., He, Q., Yang, Y., "Qos-aware service recommendation for multi-tenant saas on the cloud", in 2015 IEEE International Conference on Services Computing, June 2015, str. 178-185.

[17] Kurdija, A., Silic, M., Srbljic, S., "Real-time adaptive qos prediction using approximate matrix multiplication", Int. J. Web Grid Serv., Vol. 14, No. 2, Jan. 2018, str. 200–235, dostupno na: https://doi.org/10.1504/IJWGS.2018.090739

[18] Kurdija, A. S., Silic, M., Delac, G., Vladimir, K., “Fast multi-criteria service selection for multi-user composite applications”, IEEE Transactions on Services Computing, Early Access, 2019.

[19] Kurdija, A. S., Silic, M., Vladimir, K., Delac, G., “Efficient global correlation measures for a collaborative filtering dataset”, Knowledge-Based Systems, Vol. 147, 2018, str. 36 - 42, dostupno na: http://www.sciencedirect.com/science/article/pii/S0950705118300650

[20] Kurdija, A. S., Silic, M., Delac, G., Vladimir, K., Srbljic, S., “Efficient multi-user service selection based on the transportation problem”, in Web Services – ICWS 2018. Cham: Springer International Publishing, 2018, str. 507–515.

[21] Krafzig, D., Banke, K., Slama, D., Enterprise SOA: Service-Oriented Architecture Best Practices (The Coad Series). Prentice Hall PTR, 2004.

[22] Yu, T., Zhang, Y., Lin, K.-J., “Efficient algorithms for web services selection with end-to-end qos constraints”, ACM Trans. Web, Vol. 1, No. 1, May 2007, dostupno na: http://doi.acm.org/10.1145/1232722.1232728

[23] Wang, S., Huang, L., Sun, L., Hsu, C.-H., Yang, F., “Efficient and reliable service selection for heterogeneous distributed software systems”, Future Generation Computer Systems, Vol. 74, 2017, str. 158-167, dostupno na: http://www.sciencedirect.com/science/article/pii/S0167739X15003970

[24] Kaewbanjong, K., Intakosum, S., “Qos attributes of web services: A systematic review and classification”, Journal of Advanced Management Science, 01 2015, str. 194-202.

[25] Silic, M., Delac, G., Krka, I., Srbljic, S., “Scalable and accurate prediction of availability of atomic web services”, Services Computing, IEEE Transactions on, Vol. 7, No. 2, April 2014, str. 252-264.

[26] Wang, S., Hsu, C.-H., Liang, Z., Sun, Q., Yang, F., “Multi-user web service selection based on multi-qos prediction”, Information Systems Frontiers, Vol. 16, No. 1, 2014, str. 143–152, dostupno na: http://dx.doi.org/10.1007/s10796-013-9455-4

[27] Wang, H., Cheng, Y., “Interval number based service selection for multi-users’ requirements”, in 2016 IEEE International Conference on Web Services (ICWS), June 2016, str. 712-715.

[28] Lyu, M. R., Handbook of software reliability engineering. Hightstown, NJ, USA: McGraw-Hill, Inc., 1996.

[29] Lyu, M. R., “Software reliability engineering: A roadmap”, in 2007 Future of Software Engineering, ser. FOSE ’07. Washington, DC, USA: IEEE Computer Society, 2007, str. 153–170, dostupno na: http://dx.doi.org/10.1109/FOSE.2007.24

[30] Xie, G., Zeng, G., Chen, Y., Bai, Y., Zhou, Z., Li, R., Li, K., “Minimizing redundancy to satisfy reliability requirement for a parallel application on heterogeneous service-oriented systems”, IEEE Transactions on Services Computing, Vol. PP, No. 99, 2017, str. 1-1.

[31] Xie, G., Chen, Y., Liu, Y., Wei, Y., Li, R., Li, K., “Resource consumption cost minimization of reliable parallel applications on heterogeneous embedded systems”, IEEE Transactions on Industrial Informatics, Vol. PP, No. 99, 2017, str. 1-1.

[32] Zeng, L., Benatallah, B., Ngu, A., Dumas, M., Kalagnanam, J., Chang, H., “Qos-aware middleware for web services composition”, Software Engineering, IEEE Transactions on, Vol. 30, No. 5, May 2004, str. 311-327.

[33] Wang, D., Trivedi, K., “Modeling user-perceived service availability”, in Service Availability, ser. Lecture Notes in Computer Science, Malek, M., Nett, E., Suri, N., (ur.). Springer Berlin Heidelberg, 2005, Vol. 3694, str. 107-122, dostupno na: http://dx.doi.org/10.1007/11560333_10

[34] Cortellessa, V., Grassi, V., “Reliability modeling and analysis of service-oriented architectures”, in Test and Analysis of Web Services, Baresi, L., Nitto, E., (ur.). Springer Berlin Heidelberg, 2007, str. 339-362, dostupno na: http://dx.doi.org/10.1007/978-3-540-72912-9_12

[35] Cheung, L., Golubchik, L., Sha, F., “A study of web services performance prediction: A client’s perspective”, in Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on, July 2011, str. 75-84.

[36] Wang, Y., Lively, W. M., Simmons, D. B., “Web software traffic characteristics and failure prediction model selection”, J. Comp. Methods in Sci. and Eng., Vol. 9, No. 1,2S1, Apr. 2009, str. 23–33, dostupno na: http://dl.acm.org/citation.cfm?id=1608790.1608793

[37] Baryshnikov, Y., Coffman, E., Pierre, G., Rubenstein, D., Squillante, M., Yimwadsana, T., “Predictability of web-server traffic congestion”, in Proceedings of the 10th International Workshop on Web Content Caching and Distribution, ser. WCW ’05. Washington, DC, USA: IEEE Computer Society, 2005, str. 97–103, dostupno na: http://dx.doi.org/10.1109/WCW.2005.17

[38] Andreolini, M., Casolari, S., “Load prediction models in web-based systems”, in Proceedings of the 1st International Conference on Performance Evaluation Methodolgies and Tools, ser. valuetools ’06. New York, NY, USA: ACM, 2006, dostupno na: http://doi.acm.org/10.1145/1190095.1190129

[39] Lee, Y.-T., Chen, K.-T., “Is server consolidation beneficial to mmorpg? a case study of world of warcraft”, in Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing, ser. CLOUD ’10. Washington, DC, USA: IEEE Computer Society, 2010, str. 435–442, dostupno na: http://dx.doi.org/10.1109/CLOUD.2010.57

[40] Zheng, Z., Lyu, M. R., “Collaborative reliability prediction of service-oriented systems”, Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering, ICSE ’10, Vol. 1, 2010, str. 35-44, new York, NY, USA.

[41] Baresi, L., Guinea, S., “Event-based multi-level service monitoring”, in ICWS, 2013, str. 83-90.

[42] Zhang, P., Zhuang, Y., Leung, H., Song, W., Zhou, Y., “A novel qos monitoring approach sensitive to environmental factors”, in Web Services (ICWS), 2015 IEEE International Conference on, June 2015, str. 145-152.

[43] Abdel-Ghaly, A., Chan, P., Littlewood, B., “Evaluation of competing software reliability predictions”, Software Engineering, IEEE Transactions on, Vol. SE-12, No. 9, Sept 1986, str. 950-967.

[44] Musa, J. D., Iannino, A., Okumoto, K., Software reliability: measurement, prediction, application (professional ed.). New York, NY, USA: McGraw-Hill, Inc., 1990.

[45] Friedman, M., Tran, P., “Reliability techniques for combined hardware/software systems”, in Reliability and Maintainability Symposium, 1992. Proceedings., Annual, Jan 1992, str. 290-293.

[46] Cheung, L., Roshandel, R., Medvidovic, N., Golubchik, L., “Early prediction of software component reliability”, in Software Engineering, 2008. ICSE ’08. ACM/IEEE 30th International Conference on, May 2008, str. 111-120.

[47] Cheung, L., Krka, I., Golubchik, L., Medvidovic, N., “Architecture-level reliability prediction of concurrent systems”, in Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering, ser. ICPE ’12. New York, NY, USA: ACM, 2012, str. 121–132, dostupno na: http://doi.acm.org/10.1145/2188286.2188305

[48] Tsai, W. T., Zhang, D., Chen, Y., Huang, H., Paul, R., Liao, N., "A software reliability model for web services", in The 8th IASTED International Conference on Software Engineering and Applications, 2004, str. 144–149.

[49] Hwang, S.-Y., Lim, E.-P., Lee, C.-H., Chen, C.-H., "Dynamic web service selection for reliable web service composition", Services Computing, IEEE Transactions on, Vol. 1, No. 2, April 2008, str. 104-116.

[50] Delac, G., Silic, M., Vladimir, K., "Reliability sensitivity analysis for yahoo! pipes mashups", in Information Communication Technology Electronics Microelectronics, International Convention on, 2013, str. 851-856.

[51] Delac, G., Silic, M., Srbljic, S., "A reliability improvement method for soa-based applications", Dependable and Secure Computing, IEEE Transactions on, Vol. 12, No. 2, March 2015, str. 136-149.

[52] Silic, M., Delac, G., Krka, I., Srbljic, S., "Scalable and accurate prediction of availability of atomic web services", Services Computing, IEEE Transactions on, Vol. 7, No. 2, April 2014, str. 252-264.

[53] Su, X., Khoshgoftaar, T. M., "A survey of collaborative filtering techniques", Adv. in Artif. Intell., Vol. 2009, Jan. 2009, str. 4:2–4:2, dostupno na: http://dx.doi.org/10.1155/2009/421425

[54] Burke, R., "Hybrid recommender systems: Survey and experiments", User Modeling and User-Adapted Interaction, Vol. 12, No. 4, Nov. 2002, str. 331–370, dostupno na: http://dx.doi.org/10.1023/A:1021240730564

[55] Ma, H., King, I., Lyu, M. R., "Effective missing data prediction for collaborative filtering", in Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ser. SIGIR '07. New York, NY, USA: ACM, 2007, str. 39–46, dostupno na: http://doi.acm.org/10.1145/1277741.1277751

[56] Das, A. S., Datar, M., Garg, A., Rajaram, S., "Google news personalization: Scalable online collaborative filtering", in Proceedings of the 16th International Conference on World Wide Web, ser. WWW '07. New York, NY, USA: ACM, 2007, str. 271–280, dostupno na: http://doi.acm.org/10.1145/1242572.1242610

[57] Wei, C., Hsu, W., Lee, M. L., "A unified framework for recommendations based on quaternary semantic analysis", in Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, ser.

SIGIR '11. New York, NY, USA: ACM, 2011, str. 1023–1032, dostupno na: http://doi.acm.org/10.1145/2009916.2010052

[58] Guan, H., Li, H., Guo, M., "Semi-sparse algorithm based on multi-layer optimization for recommendation system", in Proceedings of the 2012 International Workshop on Programming Models and Applications for Multicores and Manycores, ser. PMAM '12. New York, NY, USA: ACM, 2012, str. 148–155, dostupno na: http://doi.acm.org/10.1145/2141702.2141719

[59] Breese, J. S., Heckerman, D., Kadie, C., "Empirical analysis of predictive algorithms for collaborative filtering", in Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, ser. UAI'98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, str. 43–52, dostupno na: http://dl.acm.org/citation.cfm?id=2074094.2074100

[60] Sarwar, B., Karypis, G., Konstan, J., Riedl, J., "Item-based collaborative filtering recommendation algorithms", in Proceedings of the 10th International Conference on World Wide Web, ser. WWW '01. New York, NY, USA: ACM, 2001, str. 285–295, dostupno na: http://doi.acm.org/10.1145/371920.372071

[61] Zheng, Z., Ma, H., Lyu, M. R., King, I., "Qos-aware web service recommendation by collaborative filtering", IEEE Trans. Serv. Comput., Vol. 4, 2011, str. 140-152.

[62] Takács, G., Pilászy, I., Németh, B., Tikk, D., "Scalable collaborative filtering approaches for large recommender systems", J. Mach. Learn. Res., Vol. 10, Jun. 2009, str. 623–656, dostupno na: http://dl.acm.org/citation.cfm?id=1577069.1577091

[63] Bennett, J., Lanning, S., Netflix, N., "The netflix prize", in In KDD Cup and Workshop in conjunction with KDD, 2007.

[64] He, X., Zhang, H., Kan, M.-Y., Chua, T.-S., "Fast matrix factorization for online recommendation with implicit feedback", in Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, ser. SIGIR '16. New York, NY, USA: ACM, 2016, str. 549–558, dostupno na: http://doi.acm.org/10.1145/2911451.2911489

[65] Pourhaji Kazem, A. A., Pedram, H., Abolhassani, H., "Bnqm", Expert Syst. Appl., Vol. 42, No. 20, Nov. 2015, str. 6828–6843, dostupno na: http://dx.doi.org/10.1016/j.eswa.2015.04.045

[66] Yu, Q., Zheng, Z., Wang, H., "Trace norm regularized matrix factorization for service recommendation", in International Conference on Web Services, ser. ICWS

'13. Washington, DC, USA: IEEE Computer Society, 2013, str. 34–41, dostupno na: http://dx.doi.org/10.1109/ICWS.2013.15

[67] Zheng, Z., Ma, H., Lyu, M., King, I., "Collaborative web service qos prediction via neighborhood integrated matrix factorization", Services Computing, IEEE Transactions on, Vol. 6, No. 3, July 2013, str. 289-299.

[68] Zhu, J., Zheng, Z., Lyu, M. R., "Context-Aware Reliability Prediction of Black-Box Services", ArXiv e-prints, Feb. 2015.

[69] Lee, K., Park, J., Baik, J., "Location-based web service qos prediction via preference propagation for improving cold start problem", in Web Services (ICWS), 2015 IEEE International Conference on, June 2015, str. 177-184.

[70] He, P., Zhu, J., Zheng, Z., Xu, J., Lyu, M., "Location-based hierarchical matrix factorization for web service recommendation", in Web Services (ICWS), 2014 IEEE International Conference on, June 2014, str. 297-304.

[71] Tan, W., Li, L., Sun, Y., "A novel performance prediction framework for web service workflow applications", in Human Centered Computing, ser. Lecture Notes in Computer Science, Zu, Q., Hu, B., Gu, N., Seng, S., (ur.). Springer International Publishing, 2015, Vol. 8944, str. 55-68, dostupno na: http://dx.doi.org/10.1007/978-3-319-15554-8_5

[72] Luo, X., Lv, Y., Li, R., Chen, Y., "Web service qos prediction based on adaptive dynamic programming using fuzzy neural networks for cloud services", IEEE Access, Vol. 3, 2015, str. 2260-2269.

[73] Ramalingam, S., Mohandas, L., "A fuzzy based sensor web for adaptive prediction framework to enhance the availability of web service", Int. J. Distrib. Sen. Netw., Vol. 2016, Jan. 2016, str. 2:2–2:2, dostupno na: https://doi.org/10.1155/2016/4972061

[74] Yao, L., Sheng, Q., Segev, A., Yu, J., "Recommending web services via combining collaborative filtering with content-based features", in Web Services (ICWS), 2013 IEEE 20th International Conference on, June 2013, str. 42-49.

[75] Xu, Y., Yin, J., Deng, S., Xiong, N. N., Huang, J., "Context-aware qos prediction for web service recommendation and selection", Expert Systems with Applications, Vol. 53, 2016, str. 75 - 86, dostupno na: http://www.sciencedirect.com/science/article/pii/S0957417416000208

[76] Lee, K., Choi, O., Baik, J., "Service reliability prediction method for service-oriented multi-agent system", in Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference on, Dec 2014, str. 653-660.

[77] Deng, S., Huang, L., Xu, G., “Social network-based service recommendation with trust enhancement”, Expert Systems with Applications, Vol. 41, No. 18, 2014, str. 8075 - 8084, dostupno na: http://www.sciencedirect.com/science/article/pii/S0957417414004102

[78] Zheng, Z., Wu, X., Zhang, Y., Lyu, M., Wang, J., “Qos ranking prediction for cloud services”, Parallel and Distributed Systems, IEEE Transactions on, Vol. 24, No. 6, June 2013, str. 1213-1222.

[79] Wu, L., Garg, S., Buyya, R., “Sla-based resource allocation for software as a service provider (saas) in cloud computing environments”, in Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on, May 2011, str. 195-204.

[80] Silic, M., Delac, G., Srbljic, S., “Prediction of atomic web services reliability based on k-means clustering”, in Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, ser. ESEC/FSE 2013. New York, NY, USA: ACM, 2013, str. 70–80, dostupno na: http://doi.acm.org/10.1145/2491411.2491424

[81] Silic, M., Delac, G., Srbljic, S., “Prediction of atomic web services reliability for qos-aware recommendation”, Services Computing, IEEE Transactions on, Vol. 8, No. 3, May 2015, str. 425-438.

[82] Wang, R. Y., Strong, D. M., “Beyond accuracy: What data quality means to data consumers”, J. Manage. Inf. Syst., Vol. 12, No. 4, Mar. 1996, str. 5–33, dostupno na: http://dx.doi.org/10.1080/07421222.1996.11518099

[83] Pipino, L. L., Lee, Y. W., Wang, R. Y., “Data quality assessment”, Commun. ACM, Vol. 45, No. 4, Apr. 2002, str. 211–218, dostupno na: http://doi.acm.org/10.1145/505248.506010

[84] Amatriain, X., Pujol, J. M., Tintarev, N., Oliver, N., “Rate it again: Increasing recommendation accuracy by user re-rating”, in Proceedings of the Third ACM Conference on Recommender Systems, ser. RecSys ’09. New York, NY, USA: ACM, 2009, str. 173–180, dostupno na: http://doi.acm.org/10.1145/1639714.1639744

[85] Amatriain, X., Pujol, J. M., Oliver, N., I Like It... I Like It Not: Evaluating User Ratings Noise in Recommender Systems. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, str. 247–258, dostupno na: http://dx.doi.org/10.1007/978-3-642-02247-0_24

[86] Said, A., Jain, B. J., Narr, S., Plumbaum, T., Users and Noise: The Magic Barrier of Recommender Systems. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, str. 237–248, dostupno na: http://dx.doi.org/10.1007/978-3-642-31454-4_20

[87] Grčar, M., Mladenič, D., Fortuna, B., Grobelnik, M., Data Sparsity Issues in the Collaborative Filtering Framework. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, str. 58–76, dostupno na: http://dx.doi.org/10.1007/11891321_4

[88] Marlin, B. M., Zemel, R. S., Roweis, S., Slaney, M., "Collaborative filtering and the missing at random assumption", in Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence, ser. UAI'07. Arlington, Virginia, United States: AUAI Press, 2007, str. 267–275, dostupno na: http://dl.acm.org/citation.cfm?id=3020488.3020521

[89] Su, X., Khoshgoftaar, T. M., "A survey of collaborative filtering techniques", Adv. in Artif. Intell., Vol. 2009, Jan. 2009, str. 4:2–4:2, dostupno na: http://dx.doi.org/10.1155/2009/421425

[90] Sar Shalom, O., Berkovsky, S., Ronen, R., Ziklik, E., Amihood, A., "Data quality matters in recommender systems", in Proceedings of the 9th ACM Conference on Recommender Systems, ser. RecSys '15. New York, NY, USA: ACM, 2015, str. 257–260, dostupno na: http://doi.acm.org/10.1145/2792838.2799670

[91] He, Q., Yan, J., Jin, H., Yang, Y., "Quality-aware service selection for service-based systems based on iterative multi-attribute combinatorial auction", IEEE Transactions on Software Engineering, Vol. 40, No. 2, Feb 2014, str. 192-215.

[92] Zhong, Y., Li, X., He, Q., "Iterative auction based service selection for multi-tenant service-based systems", in Proceedings of the Australasian Computer Science Week Multiconference, ser. ACSW '17. New York, NY, USA: ACM, 2017, str. 44:1–44:4, dostupno na: http://doi.acm.org/10.1145/3014812.3014858

[93] Wang, P., Du, X., "Qos-aware service selection using an incentive mechanism", IEEE Transactions on Services Computing, Vol. PP, No. 99, 2017, str. 1-1.

[94] h. wang, Yu, C., Wang, L., Yu, Q., "Effective bigdata-space service selection over trust and heterogeneous qos preferences", IEEE Transactions on Services Computing, Vol. PP, No. 99, 2015, str. 1-1.

[95] Chen, Y., Huang, J., Lin, C., Hu, J., "A partial selection methodology for efficient qos-aware service composition", IEEE Transactions on Services Computing, Vol. 8, No. 3, May 2015, str. 384-397.

[96] Trummer, I., Faltings, B., Binder, W., "Multi-objective quality-driven service selection – a fully polynomial time approximation scheme", IEEE Transactions on Software Engineering, Vol. 40, No. 2, Feb 2014, str. 167-191.

[97] Nacer, A. A., Bessai, K., Youcef, S., Godart, C., “A multi-criteria based approach for web service selection using quality of service (qos)”, in 2015 IEEE International Conference on Services Computing, June 2015, str. 570-577.

[98] Cho, J. H., Ko, H. G., Ko, I. Y., “Adaptive service selection according to the service density in multiple qos aspects”, IEEE Transactions on Services Computing, Vol. 9, No. 6, Nov 2016, str. 883-894.

[99] Deng, S., Wu, H., Hu, D., Zhao, J. L., “Service selection for composition with qos correlations”, IEEE Transactions on Services Computing, Vol. 9, No. 2, March 2016, str. 291-303.

[100] Wang, Y., He, Q., Ye, D., Yang, Y., “Service selection based on correlated qos requirements”, in 2017 IEEE International Conference on Services Computing (SCC), June 2017, str. 241-248.

[101] Hwang, S. Y., Hsu, C. C., Lee, C. H., “Service selection for web services with probabilistic qos”, IEEE Transactions on Services Computing, Vol. 8, No. 3, May 2015, str. 467-480.

[102] Deng, S., Huang, L., Hu, D., Zhao, J. L., Wu, Z., “Mobility-enabled service selection for composite services”, IEEE Transactions on Services Computing, Vol. 9, No. 3, May 2016, str. 394-407.

[103] Yu, T., Lin, K.-J., “Service selection algorithms for web services with end-to-end qos constraints”, in Proceedings. IEEE International Conference on e-Commerce Technology, 2004. CEC 2004., July 2004, str. 129-136.

[104] Zeng, L., Benatallah, B., Ngu, A. H. H., Dumas, M., Kalagnanam, J., Chang, H., “Qos-aware middleware for web services composition”, IEEE Transactions on Software Engineering, Vol. 30, No. 5, May 2004, str. 311-327.

[105] Ardagna, D., Pernici, B., Global and Local QoS Guarantee in Web Service Selection. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, str. 32–46, dostupno na: http://dx.doi.org/10.1007/11678564_4

[106] Jin, H., Zou, H., Yang, F., Lin, R., Zhao, X., “A hybrid service selection approach for multi-user requests”, in 2012 IEEE 14th International Conference on High Performance Computing and Communication 2012 IEEE 9th International Conference on Embedded Software and Systems, June 2012, str. 1142-1149.

[107] Alrifai, M., Skoutas, D., Risse, T., “Selecting skyline services for qos-based web service composition”, in Proceedings of the 19th International Conference on World Wide Web, ser. WWW ’10. New York, NY, USA: ACM, 2010, str. 11–20, dostupno na: http://doi.acm.org/10.1145/1772690.1772693

[108] Kang, G., Liu, J., Tang, M., Liu, X., Fletcher, K. K., “Web service selection for resolving conflicting service requests”, in 2011 IEEE International Conference on Web Services, July 2011, str. 387-394.

[109] Kuhn, H. W., “The hungarian method for the assignment problem”, Naval Research Logistics Quarterly, Vol. 2, No. 1-2, 1955, str. 83-97, dostupno na: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109

[110] Shao, L., Zhang, J., Wei, Y., Zhao, J., Xie, B., Mei, H., “Personalized qos prediction forweb services via collaborative filtering”, in IEEE International Conference on Web Services (ICWS 2007), July 2007, str. 439-446.

[111] Xie, G., Jiang, J., Liu, Y., Li, R., Li, K., “Minimizing energy consumption of real-time parallel applications using downward and upward approaches on heterogeneous systems”, IEEE Transactions on Industrial Informatics, Vol. 13, No. 3, June 2017, str. 1068-1078.

[112] Xie, G., Zeng, G., Jiang, J., Fan, C., Li, R., Li, K., “Energy management for multiple real-time workflows on cyber–physical cloud systems”, Future Generation Computer Systems, 2017, dostupno na: http://www.sciencedirect.com/science/article/pii/S0167739X1731066X

[113] Xie, G., Zeng, G., Li, Z., Li, R., Li, K., “Adaptive dynamic scheduling on multi-functional mixed-criticality automotive cyber-physical systems”, IEEE Transactions on Vehicular Technology, Vol. PP, No. 99, 2017, str. 1-1.

[114] Shambour, Q., Lu, J., “A trust-semantic fusion-based recommendation approach for e-business applications”, Decision Support Systems, Vol. 54, No. 1, 2012, str. 768 - 780, dostupno na: http://www.sciencedirect.com/science/article/pii/S0167923612002400

[115] Zhang, Z., Lin, H., Liu, K., Wu, D., Zhang, G., Lu, J., “A hybrid fuzzy-based personalized recommender system for telecom products/services”, Information Sciences, Vol. 235, 2013, str. 117 - 129, data-based Control, Decision, Scheduling and Fault Diagnostics, dostupno na: http://www.sciencedirect.com/science/article/pii/S0020025513000820

[116] Mao, M., Lu, J., Zhang, G., Zhang, J., "Multirelational social recommendations via multigraph ranking", IEEE Transactions on Cybernetics, Vol. 47, No. 12, Dec 2017, str. 4049-4061.

[117] Drineas, P., "Fast monte-carlo algorithms for approximate matrix multiplication", Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on, 2011, str. 452-459, las Vegas, Nevada, USA.

[118] Zheng, Z., Lyu, M. R., "Ws-dream: A distributed reliability assessment mechanism for web services", in 2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN), June 2008, str. 392-397.

[119] Ardagna, C. A., Jhawar, R., Piuri, V., "Dependability certification of services: a model-based approach", Computing, 2013, str. 1-28.

[120] Taha, H. A., Operations Research: An Introduction (8th Edition). Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.

[121] Reinfeld, N., Vogel, W., Mathematical programming. Prentice-Hall, 1958, dostupno na: https://books.google.hr/books?id=Hj9ZAAAAMAAJ

[122] Dantzig, G., Linear programming and extensions, ser. Rand Corporation Research Study. Princeton, NJ: Princeton Univ. Press, 1963, dostupno na: http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+180926950&sourceid=fbw_bibsonomy

[123] Winston, W., Goldberg, J., Operations Research: Applications and Algorithms. Thomson Brooks/Cole, 2004, dostupno na: https://books.google.hr/books?id=tg5DAQAAIAAJ

[124] Chaudhuri, A., De, K., Chatterjee, D., Mitra, P., "Trapezoidal fuzzy numbers for the transportation problem", ArXiv, Vol. abs/1307.1893, 2013.

[125] Kelly, D. J., O'Neill, G. M., "The minimum cost flow problem and the network simplex solution method", Master's thesis, University College, Dublin, Ireland, 1991.

[126] Guo, G., Zhang, J., Yorke-Smith, N., "A novel bayesian similarity measure for recommender systems", in Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI), 2013, str. 2619-2625.

[127] Guo, G., Zhang, J., Thalmann, D., Yorke-Smith, N., "Etaf: An extended trust antecedents framework for trust prediction", in Proceedings of the 2014 International Conference on Advances in Social Networks Analysis and Mining (ASONAM), 2014, str. 540-547.

[128] Massa, P., Avesani, P., “Trust-aware recommender systems”, in Proceedings of the 2007 ACM conference on Recommender systems, 2007, str. 17-24.

[129] Goldberg, K., Roeder, T., Gupta, D., Perkins, C., “Eigentaste: A constant time collaborative filtering algorithm”, Information Retrieval, Vol. 4, No. 2, 2001, str. 133–151, dostupno na: http://dx.doi.org/10.1023/A:1011419012209

[130] Ziegler, C.-N., McNee, S. M., Konstan, J. A., Lausen, G., “Improving recommendation lists through topic diversification”, in Proceedings of the 14th International Conference on World Wide Web, ser. WWW ’05. New York, NY, USA: ACM, 2005, str. 22–32, dostupno na: http://doi.acm.org/10.1145/1060745.1060754

[131] Dooms, S., De Pessemier, T., Martens, L., “Movietweetings: a movie rating dataset collected from twitter”, in Workshop on Crowdsourcing and Human Computation for Recommender Systems, held in conjunction with the 7th ACM Conference on Recommender Systems, 2013, str. 2.

[132] Järvelin, K., Kekäläinen, J., “Cumulated gain-based evaluation of ir techniques”, ACM Trans. Inf. Syst., Vol. 20, No. 4, Oct. 2002, str. 422–446, dostupno na: http://doi.acm.org/10.1145/582415.582418

[133] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G., “Learning to rank using gradient descent”, in Proceedings of the 22Nd International Conference on Machine Learning, ser. ICML ’05. New York, NY, USA: ACM, 2005, str. 89–96, dostupno na: http://doi.acm.org/10.1145/1102351.1102363

[134] Zhang, Y., Zheng, Z., Lyu, M. R., “Wsexpress: A qos-aware search engine for web services”, Proceedings of the 8th International Conference on Web Services (ICWS2010), 2010, str. 83-90, miami, Florida, USA.

[135] Mitchell, S., Consulting, S. M., Dunning, I., “Pulp: A linear programming toolkit for python”, https://pythonhosted.org/PuLP/, dostupno na: https://pythonhosted.org/PuLP/ 2011.

Biography

Adrian Satja Kurdija was born in Sarajevo in 1991 and has lived in Zagreb since early childhood, where he finished elementary and high school. He was successful in national and international high school competitions in mathematics and informatics. In 2015 he graduated in computer science and mathematics at the Faculty of Science of the University of Zagreb. He was a software engineering intern in *imo.im* (2012) and *Google* (2013) for three months each. Since 2014 he has been a member of the National Committee for informatics competitions, for which he creates tasks and helps to prepare contestants. As a result of this work, he published the handbook *Algoritmi u Pythonu* in 2019 as a co-author. From 2016 he has been employed at the Faculty of Electrical Engineering and Computing of the University of Zagreb, first as a research assistant on the project *Recommender System for Service-Oriented Architecture* led by Professor Siniša Srbljić, and from 2018 as a teaching assistant at the Department of Electronics, Microelectronics, Computer and Intelligent Systems. During his doctoral studies at the same faculty, he studied QoS prediction using collaborative filtering and algorithms for web service selection. Results of this research with his advisor, Assistant Professor Marin Šilić, and other colleagues, have been published in several prominent computer science journals. They have also been presented in invited lectures at the University of California Irvine and the University of Southern California in 2019.

List of Publications

Journal Papers

1. Kurdija, A. S., Šilić, M., Delač, G., Vladimir, K., "Fast Multi-Criteria Service Selection for Multi-User Composite Applications", IEEE Transactions on Services Computing, *Early Acces*
2. Kurdija, A. S., Šilić, M., Vladimir, K., Delač, G., "Efficient Global Correlation Measures for a Collaborative Filtering Dataset", Knowledge-Based Systems, Vol. 147, No. 1, May 2018, pp. 36-42.
3. Kurdija, A. S., Šilić, M., Srbljić, S., "Real-time adaptive QoS prediction using approximate matrix multiplication", International Journal of Web and Grid Services, Vol. 14, No.

2, January 2018, pp. 200-235.

4. Švedek, V., Kurdija, A. S., Ilić, Ž., "Wireless channel estimation in OFDM systems based on collaborative filtering techniques", Journal of Electrical Engineering, Vol. 70, No. 3, July 2019, pp. 244-252.
5. Čaklović, L., Kurdija, A. S., "A universal voting system based on the Potential Method", European Journal of Operational Research, Vol. 259, No. 2, June 2017, pp. 677-688
6. Kurdija, A. S., Smiljanić, M., Ilić, Ž., "A chunk and power allocation algorithm for proportional fairness in OFDMA relay networks", Wireless Networks, Vol. 22, No. 8, November 2016, pp. 2741-2751.
7. Kurdija, A. S., Ilić, Ž., Šišul, G., "Unequal-Slot Based Data Rate Maximization Algorithm for Multiuser OFDMA Systems", IEEE Communications Letters, Vol. 16, No. 5, May 2012, pp. 682-684.

Conference Papers

1. Kurdija, A. S., Šilić, M., Delač, G., Vladimir, K., Srbljić, S., "Efficient Multi-user Service Selection Based on the Transportation Problem", Proceedings of the International Conference on Web Services ICWS 2018, June 2018, pp. 507-515.
2. Afrić, P., Šikić, L., Kurdija, A. S., Delač, G., Šilić, M., "REPD: Source Code Defect Prediction As Anomaly Detection", 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), July 2019.
3. Afrić, P., Kurdija, A. S., Šikić, L., Šilić, M., Delač, G., Vladimir, K., Srbljić, S., "GRASP Method for Vehicle Routing with Delivery Place Selection", Proceedings of the International Conference on AI and Mobile Services AIMS 2019, June 2019., pp. 72-83.
4. Afrić, P., Kurdija, A. S., Šikić, L., Šilić, M., Delač, G., Vladimir, K., Srbljić, S., "Population-Based Variable Neighborhood Descent for Discrete Optimization", Proceedings of the International Conference on AI and Mobile Services AIMS 2019, June 2019., pp. 1-12.

Books

1. Dmitrović, N.; Kurdija, A. S., "Algoritmi u Pythonu" (in Croatian), Školska knjiga, Zagreb, August 2019.

Životopis

Adrian Satja Kurdija rođen je 1991. godine u Sarajevu, a u Zagrebu živi od ranog djetinjstva. Ondje pohađa osnovnu i srednju školu, tijekom koje osvaja priznanja na domaćim i stranim natjecanjima iz matematike i informatike. Godine 2015. završava diplomski studij Računarstvo i matematika na Prirodoslovno-matematičkom fakultetu Sveučilišta u Zagrebu. Sudjelovao je na tromjesečnim stručnim praksama u softverskim tvrtkama *imo.im* (2012.) i *Google* (2013.). Od 2014. član je Državnog povjerenstva za informatička natjecanja za koja izrađuje zadatke i pomaže u pripremi natjecatelja. Kao rezultat toga rada 2019. godine objavljuje priručnik *Algoritmi u Pythonu* kao koautor. Od 2016. zaposlen je na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu, najprije kao doktorand kod prof. dr. sc. Siniše Srbljića na projektu *Sustav preporuke u arhitekturi zasnovanoj na uslugama*, a od 2018. godine kao asistent na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave. Tijekom doktorskog studija na istom fakultetu istraživao je predviđanje kvalitete usluge suradničkim filtriranjem te algoritme za odabir web usluga. Rezultati tih istraživanja s mentorom Marinom Šilićem i drugim kolegama pretočeni su u nekoliko znanstvenih radova koji su objavljeni u uglednim časopisima u području računarstva i predstavljeni na pozvanim predavanjima na sveučilištima University of California Irvine i University of Southern California 2019. godine.