УНИВЕРЗИТЕТ У БЕОГРАДУ
МАТЕМАТИЧКИ ФАКУЛТЕТ

Лука Марковић

МЕЂУСОБНО СВОЂЕЊЕ НП-КОМПЛЕТНИХ ПРОБЛЕМА У СИСТЕМУ URSA

мастер рад

Београд, 2025.

Ментор:

др Предраг Јаничић, редован професор
Универзитет у Београду, Математички факултет

Чланови комисије:

др Филип Марић, редован професор
Универзитет у Београду, Математички факултет

др Весна Маринковић, редован професор
Универзитет у Београду, Математички факултет

Датум одбране: 15. јануар 2016.

Наслов мастер рада: Међусобно свођење НП-комплетних проблема у систему URSA

Резиме: Klasa np problema je jedna veliki skup problema u programiranju. Mnoge stvari se izucavaju na np kompletnim problemima, pa je tako tema ovog master rada nastala. Prvobitna ideja je rada je da se automatizuje proces svodjenja jednog problema na drugi, ali kako je to zahtevalo mnogo posla.

Rad se svodi na transforamciju nekih problema na SAT problem u jeziku URSA

prvo poglavlje govori sta je jezik ursa, postavlja teoriju np koplenosti. Drugo poglavlje se koncentirse na implementaciju samih algoritama prevodjenja. U zakljucku je obradjeno do cega smo dosli kao i moguca dalja istrazivanja

Кључне речи: НП-Комплетност, Теорија израчунљивости, језик УРСА, , Недетерминизам, Сертификати

Садржај

Глава 1

Увод

Motivacija za istraživanje Problem koji rešavaš Ciljevi rada Metodologija Organizacija rada

Глава 2

Разрада

2.1 НП-Комплетност

Постоје проблеми за које не постоји познато ефикасно решење. Иако се може тврдити да можда није уложено довољно напора да се такво решење пронађе, са великом сигурношћу се претпоставља да постоје проблеми за које ефикасно решење уопште не постоји.

Алгоритми који се могу решити у полиномијалном времену (O(p(n))) сматрају се ефикасним. Класа свих проблема који се могу решити ефикасним алгоритмом означава се са **P** (од енгл. polynomial time).

Посебан значај има класа **NP-комплетних проблема**. То су проблеми који нису у P, али припадају класи NP, и међусобно су строго еквивалентни: постоји ефикасан алгоритам за један NP-комплетан проблем ако и само ако постоји ефикасан алгоритам за све NP-комплетне проблеме. До данас такав алгоритам није пронађен, а верује се да ни не постоји. Управо због великог броја NP-комплетних проблема (постоје стотине, можда и хиљаде) ова област представља једну од најзанимљивијих у теорији сложености.

Редукција

Кључан концепт у проучавању NP-комплетности је **редукција** – свођење једног проблема на други. Да би се ова идеја формализовала, најчешће се посматрају проблеми одлучивања. Ако је могуће решити проблем одлучивања, обично се може решити и одговарајући изворни проблем.

Дефиниција. Нека су L_1 и L_2 два језика, подскупови скупова улаза U_1

и U_2. Кажемо да је L_1 *полиномијално сводљив* на L_2 ако постоји алгоритам полиномијалне временске сложености који сваки улаз $u_1 \in U_1$ трансформише у улаз $u_2 \in U_2$ тако да важи:

$$u_1 \in L_1 \iff u_2 \in L_2.$$

Ако постоји алгоритам за препознавање језика L_2, онда се он може комбиновати са алгоритмом редукције и на тај начин добити алгоритам за препознавање L_1.

Теорема. Ако је L_1 полиномијално сводљив на L_2 и ако за L_2 постоји алгоритам полиномијалне временске сложености, онда постоји алгоритам полиномијалне временске сложености за L_1.

Доказ (скица). Полиномијална сводљивост $L_1 \to L_2$ не подразумева сводљивост у обрнутом смеру. Разлог је што дефиниција сводљивости налаже да се произвољан улаз из L_1 може трансформисати у улаз за L_2, али не и обрнуто. У том смислу, ако је L_1 полиномијално сводљив на L_2, сматра се да је L_2 „тежи“ проблем.

Два језика L_1 и L_2 су полиномијално еквивалентна ако важи $L_1 \leq_p L_2$ и $L_2 \leq_p L_1$. Релација полиномијалне сводљивости је транзитивна.

Класе сложености

Теорија NP-комплетности је део шире области која се назива *теорија сложености израчунавања.*

Формална дефиниција класе P гласи:

$P = \{L \mid \exists$ детерминистичка Тјурингова машина M која препознаје L у полиномијалном вре

Један од најпознатијих NP-комплетних проблема је **проблем трговачког путника (TSP)**. За њега није познат алгоритам полиномијалне сложености. Међутим, ако нам се да кандидат за решење (нпр. низ чворова), можемо у полиномијалном времену проверити да ли оно представља Хамилтонов циклус и израчунати његову дужину. Ово својство назива се **полиномијална проверљивост**.

Важно је истаћи да је полиномијална проверљивост (могућност брзе провере решења) различита од полиномијалне решивости (могућност брзог проналажења решења).

Класа **NP** се неформално дефинише помоћу недетерминистичког алгоритма, који се састоји из две фазе: фазе *погађања* и фазе *провере.*

- У фази погађања генерише се кандидатско решење S. - У фази провере, заједно са улазом u, алгоритам проверава да ли (u, S) представља исправно решење.

Формална дефиниција:

$NP = \{L \mid \exists$ недетерминистичка Тјурингова машина M са полиномијалним временом изврш

Проблем односа класа P и NP је познат као чувени проблем **P = NP?**

Дефиниција. Проблем H је **NP-тежак** ако је сваки проблем из NP полиномијално сводљив на H. **Дефиниција.** Проблем H је **NP-комплетан** ако важи: 1) $H \in NP$, 2) H је NP-тежак.

Проблем задовољивости (SAT)

Проблем задовољивости логичких формула (SAT) има посебан значај јер је управо он први проблем за који је доказано да је NP-комплетан (Кукова теорема).

Нека је S Булов израз у конјунктивној нормалној форми (КНФ). Тада је S конјункција више клаузи, при чему је свака клауза дисјункција литерала (променљивих или њихових негација):

$$S = (x \vee y \vee \neg z) \wedge (\neg x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z).$$

Булов израз је задовољив ако постоји такво додељивање вредности 0 и 1 променљивим да целокупан израз добије вредност 1. Задатак проблема SAT је да утврди да ли је задати израз задовољив. Није неопходно пронаћи конкретно додељивање вредности, већ само проверити постојање.

SAT припада класи NP јер се за произвољан избор вредности променљивих може у полиномијалном времену проверити да ли је израз тачан. Проблем је NP-тежак јер се извршавање сваког недетерминистичког алгоритма може описати Буловим изразом.

2.2 URSA језик за свођење проблема на CAT

УРСА (Uniform Reduction to SAT) представља специјализовани програмски језик и систем који омогућава јединствено свођење различитих комбинаторних проблема на CAT (Boolean satisfiability problem). Систем је развио проф.

Предраг Јаничић са Математичког факултета у Београду као део истраживања у области аутоматског резоновања и решавања ограничења.

Основна идеја УРСА система лежи у комбиновању императивног програмирања, сличног програмском језику Ц, са декларативним приступом решавања ограничења. Овакав хибридни приступ омогућава програмерима да користе познату императивну синтаксу за описивање проблема, док систем у позадини аутоматски генерише одговарајуће САТ формуле и позива напредне САТ солвере за проналажење решења.

Спецификација језика и архитектура система

УРСА језик подржава два основна типа података: нумеричке променљиве означене префиксом ‘n‘ и Boolean променљиве означене префиксом ‘b‘. Систем омогућава рад са низовима у једној и две димензије, што је посебно корисно за моделовање структура попут графова или матрица. Језик укључује стандардне Ц операторе за аритметичке операције (‘+‘, ‘-‘, ‘*‘), битовске операције `&`, `|`, `^`, `«`, `»`, као и уобичајене контролне структуре (‘if‘, ‘while‘, ‘for‘).

Кључне команде система укључују ‘assert()‘ за дефинисање ограничења која решење мора да задовољи, као и ‘minimize()‘ и ‘maximize()‘ за спецификовање оптимизационих циљева. Програмер пише код који описује како да се провери да ли је одређено решење валидно, а УРСА систем аутоматски трансформише овакву спецификацију у одговарајућу САТ формулу кроз процес симболичке егзекуције.

Разлози избора УРСА система

Избор УРСА система за имплементацију свођења НП проблема мотивисан је неколико кључних фактора. Универзалност система омогућава коришћење једног језика за моделовање различитих типова НП проблема, што значајно поједностављује процес имплементације. Природност императивне синтаксе чини систем приступачним програмерима који су већ упознати са језицима попут Ц-а, елиминишући потребу за учењем потпуно нових декларативних парадигми.

Аутоматизација процеса генерисања САТ формула представља значајну предност, јер ослобађа програмера од потребе за ручним креирањем сложених логичких формула. Ефикасност система произилази из коришћења напредних

САТ солвера који представљају тренутни врх технологије у области аутоматског резоновања. Посебно значајне су битовске операције које су уграђене у језик и које су погодне за прецизно моделовање комбинаторних проблема.

Предности у формалном изражавању проблема

УРСА систем пружа значајне предности у формалном изражавању и моделовању проблема. Једноставност моделовања омогућава да се проблеми описују као тестови валидности решења, што директно одговара математичким дефиницијама проблема. Флексибилност система дозвољава коришћење истог језика за проблеме из различитих домена, од теорије графова до криптоанализе.

Прецизност формалног описа постиже се јасним разликовањем између спецификације проблема и процеса решавања. Програмер се фокусира на дефинисање услова које решење мора да задовољи, док се техничке имплементационе детаље препуштају систему. Верификабилност кода омогућава да се написана спецификација директно пореди са математичком дефиницијом проблема, што олакшава проверу исправности имплементације.

Практичне предности система

Са практичне стране, УРСА омогућава компактно представљање сложених проблема кроз релативно кратке кодове који јасно описују логику проблема. Аутоматска оптимизација кроз напредне САТ солвере обезбеђује ефикасно решавање без потребе за ручним подешавањем алгоритма. Могућност спецификовања оптимизационих циљева кроз ‘minimize‘ и ‘maximize‘ команде проширује примену система ван стандардних decision проблема.

Једноставност модификације кода за различите варијанте истог проблема омогућава брзо експериментисање и итеративно побољшавање модела. Ова карактеристика је посебно важна у контексту истраживања свођења проблема, где је потребно имплементирати и тестирати различите варијанте трансформација између проблема.

Свођење алгоритама ћемо радити у језику урса. То је језик који је настао на Математичком факултету за сврху свођења проблема на проблем САТ(можда је и САТ солвер, прочитати изучити). (Линк ка раду и сајту)

2.3 Свођење НП-комплетних проблема

Да би се показало да је неки проблем НП-Комплетан, морамо да покажемо да припада и класи НП и класи НП-тешких проблема.

Проблем припада НП-класи ако може да се у полиномијалном времену провери тачност, односно за дати сертификат, можемо у полиномијалном времену проверимо да ли је то решење проблема.

Да бисмо доказали да неки проблем припада класи НП-тешких проблема, потребно је да неки познати НП-тежак проблем сведемо на овај проблем у полиномијалном времену.

Узајамно свођење проблема САТ и Клике

Проблем клике: клика је подграф графа такав да сви чворови тог подграфа су међусобно повезани, што значи да је тај подграф заправо комплетан граф. Проблем проналаска максималне клике је проблем налажења таквог подграфа са највећим могућим бројем чворова. Наш проблем одлучивања ће бити проналазак клике величине К.

Да би се показало да клика припада класи НП проблема задајемо сертификат: скуп од С чворова који се састоје у клики и С је подграф графа G. Да би смо проверили да ли је С величине К треба нам време О(1). Затим треба још да проверимо да ли су сви чворови међусобно повезани за шта нам треба јос О(к*к) времена. Тако да провера да ли к чворова из С су комплетни захтева време од О(н*н) где је н број чворова графа.

Да бисмо показали да је клика НП-тежак проблем, свешћемо проблем САТ на њу. То радимо тако што кренемо од неке инстанце сат проблема, и кажемо да ако је САТ формула задовољива, онда ће и у добијеном графу постојати клика бар величине броју клаузула

Свођење проблема Клике на проблем Покривача грана

Узајамно свођење проблема Покривача грана и Dominantnog skupa

Глава 3

Закључак

sesti panagram
sedmi panagram

Биографија аутора

Вук Стефановић Караџић (*Тршић, 26. октобар/6. новембар 1787. — Беч, 7. фебруар 1864.*) био је српски филолог, реформатор српског језика, сакупљач народних умотворина и писац првог речника српског језика. Вук је најзначајнија личност српске књижевности прве половине XIX века. Стекао је и неколико почасних доктората. Учествовао је у Првом српском устанку као писар и чиновник у Неготинској крајини, а након слома устанка преселио се у Беч, 1813. године. Ту је упознао Јернеја Копитара, цензора словенских књига, на чији је подстицај кренуо у прикупљање српских народних песама, реформу ћирилице и борбу за увођење народног језика у српску књижевност. Вуковим реформама у српски језик је уведен фонетски правопис, а српски језик је потиснуо славеносрпски језик који је у то време био језик образованих људи. Тако се као најважније године Вукове реформе истичу 1818., 1836., 1839., 1847. и 1852.