Електротехнички факултет
Универзитет у Београду

Инжењерски оптимизациони алгоритми

др Драган Олћан, проф. (olcan@etf.rs)
др Јелена Динкић, доцент (jdinkic@etf.rs)
др Јована Петровић, асистент (jovanap@etf.rs)

Изборни предмет
Школска 2023/24. година

http://mtt.etf.rs/si/ioa.htm

Који су циљеви и исходи овог курса?

- **Упознавање са основним класама оптимизационих алгоритама** који се користе у инжењерству и ИТ струци
- **Оспособљавање за практичну примену** оптимизационих алгоритама при решавању инжењерских проблема
- Пројектовање се данас ради практично искључиво коришћењем рачунара
- Практично решавање инжењерских проблема по правилу захтева употребу оптимизације
- Познавање оптимизационих алгоритама је од изузетне важности [као и спознаја када и како их (не) користити]

Зашто курс о инжењерским оптимизационим алгоритмима?

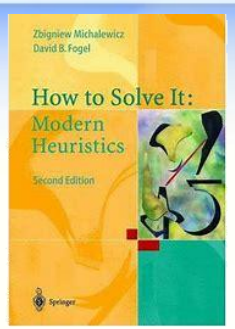
- Познавање модерних алгоритама за оптмизације је **неопходно** за све који се баве: **инжењерством, науком и бизнисом**
- Поред алгоритама, сагледаћемо **могућности за решавање** проблема данас
- Зашто једноставно не узмем рутину из неке од готових софтверских библиотека или постојећег софтвера?
  - Познавање и разумевање алгоритама је од изузетног значаја, чак и када се користе готова решења
  - Често је случај да постојећи софтвер није оптимално решење за наш проблем

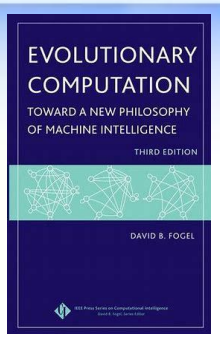
Шта је градиво курса?

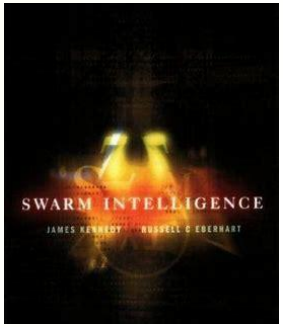
- **Увод.** Преглед појмова и представљање основне теорије решавања система нелинеарних једначина на које се своде оптимизациони алгоритми у инжењерству
- **Систематизација.** Поделе оптимизационих алгоритама
- **Оптимизациони алгоритми:**
  - систематско претраживање (енглески: systematic search)
  - случајно претраживање (енглески: random search)
  - градијентна метода (енглески: gradient method)
  - симплекс алгоритам (енглески: Nelder-Mead simplex)
  - Данцигов симплекс алгоритам (енглески: Dantzig simplex)
  - симулирано каљење (енглески: simulated annealing)
  - генетички алгоритам (енглески: genetic algorithm)
  - кретање јата (енглески: particle swarm optimization)
  - диференцијална еволуција (енглески: differential evolution)
- **Оптимизација са више критеријума.** Парето фронт и његово одређивање коришћењем оптимизационих алгоритама
- **Рад на рачунару.** Сагледавање особина и параметара оптимизационих алгоритама који су од значаја за практичну примену кроз програмирање и симулације на рачунару

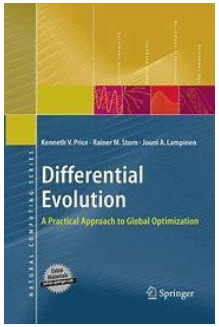
Литература

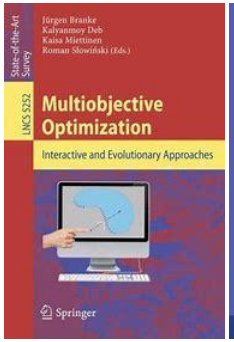
- Литература из оптимизационих алгоритама је изузетно **обимна**
- Употреба оптимизационих алгоритама је изузетно р а с п р о с т р а њ е н а
- Изабрани су они алгоритми (или класе алгоритама) који се данас најчешће употребљавају у инжењерској пракси
- Нагласак је на **алгоритмима широке намене** специјализовани алгоритми само као примери (нпр: quick-sort, Dijkstra's algorithm, Chess Master…)

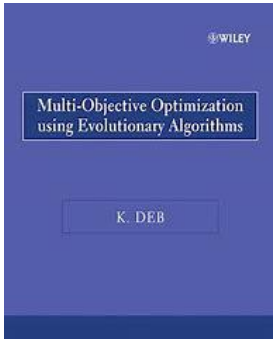








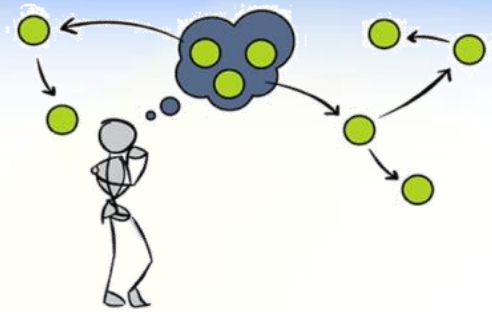




Како је организован курс?

- Предавања:
  30 часова предавања + 30 часова вежби +15 часова лабораторије
- Литература
  - Материјал са предавања (PDF за сваку седмицу, линк на сајту предмета)
  - Z. Michalewicz, D.B. Fogel, *How to Solve It: Modern Heuristics*, Springer, 2004
  - Xin-SheYang, *Engineering Optimization An Introduction with Metaheuristic Applications*, University of Cambridge, Department of Engineering, Cambridge, United Kingdom, Wiley 2010
  - D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Professional, 1989
- Оцењивање
  - **Предиспитне обавезе** (задаци на вежбама + бонуси) – одсеца се на 70 поена
    - **Задатак са вежби** се брани на вежбама у лабораторији
    - **Бонуси** до максимално 10 поена
  - **Испит** (задатак или задаци), највише 40 поена, одсеца се на 30 поена
  - **Коначна оцена** – укупан број поена се добија сабирањем поена добијених на основу предиспитних обавеза и испита. За полагање испита неопходно је освојити бар 51 поен. Оцене 6-10 су равномерно расподељене у опсегу од 51 до 100 поена
- Распоред (термини предавања и вежби):
  предавања петак 12h-14h и вежбе 14h-16h

Како се изводе предавања и вежбе?



- Предавања:
  - слајдови
  - табла & креда
  - рачунар за илустрације и показе
- Вежбе:
  - **Ви радите** на рачунару
  - програмирање: C/C++ и Python битно је да програм (про)ради!
  - коришћење постојећег софтвера

Предиспитне обавезе

- Задатак се задаје на крају предавања и важи до рока за предају
- Током вежби решавате задатак уз помоћ присутног наставника или сарадника
- Решење задатка се састоји од
  - ASCII (TXT) фајла са јасно записаним (нумеричким) одговорима на питања из задатка
  - Кодом за решење задатка (C/C++ за Visual Studio 2017/19 или Python 3.8)
- Решење се шаље као **један ZIP фајл** (< 10 MB) кроз портал на сајту
- Решења се достављају најкасније до рока који је објављен заједно са задатком (типично **среда** следеће седмице **у 12:00**, али може бити промењен због распореда)
- Освојени поени објављују се **четвртком до 21:00** на сајту предмета (структура поена за задатак биће изложена на одговарајућим вежбама)
- **Петком од 11:15 до 12:00** су лабораторијске вежбе где студенти који су позвани одговарају на питања у вези са својим решењем задатка (и евентуално решавамо примедбе на број освојених поена)

http://mtt.etf.rs/si/IOA/up.html

mtt.etf.bg.ac.rs/si/ioa.htm

To get future Google Chrome updates, you'll need Windows 10 or later. This computer is using Windows 7. Learn more

Katedra za opštu elektrotehniku
Elektrotehnički fakultet, Beograd

Home | Plan i program | Laboratorijske vežbe | Literatura | Stari ispitni rokovi | ETF Beograd

Plan i program

Predavanja

Laboratorijske vežbe

Literatura

Stari ispitni rokovi

Ostali predmeti Katedre

- Osnovi elektrotehnike
- Predmeti smera za MTT
- Elektromagnetika
- Teorija električnih kola

Inženjerski optimizacioni algoritmi

Cilj ovog predmeta je upoznavanje sa osnovnim klasama optimizacionih algoritama koji se koriste u inženjerstvu i IT struci. Naglasak je na osposobljavanju studenata za praktičnu primenu ovih algoritama za rešavanje optimizacionih problema. Osnovne klase algoritama koji se izučavaju u okviru predmeta su: potpuno pretraživanje, slučajno pretraživanje, gradijentni metod, Dantzig-ov simpleks algoritam, Nelder-Mead simpleks algoritam, simulirano kaljenje (engleski: simulated annealing), genetički algoritam (engleski: genetic algorithm), optimizacija jatom (engleski: i problemi sa više kriterijuma optimizacije, kao i odgovarajući algoritmi za odre inženjerske i IT prakse.

Karton predmeta.

Uputstvo za studente.

Fond časova: 2+2+1

Literatura:

- **Z. Michalewicz, D.B. Fogel**, *How to Solve It: Modern Heuristics*, Springer;
- **Xin-She Yang** *Engineering Optimization: An Introduction with Metaheuris*
- **D.E. Goldberg**, *Genetic Algorithms in Search, Optimization, and Machine*

Termin za nastavu, školska 2023/24. godina

Predavanja: petak 12-14h, sala 312
Vežbe: petak 14-16h, sala 70
Laboratorija: petak 11-12h, Kabinet 64b (laboratorijske ve počinju od druge sedmice semestra)

Link za slanje rešenja zadataka

Poeni osvojeni tokom tekuće školske godine.

Demo projekat Visual Studio 2019.

Stari ispitni rokovi:

- IOA, 17. februar 2023.
- IOA, 25. januar 2023.
- Kolokvijum 2, 9. decembar 2022.
- Kolokvijum 1, 4. novembar 2022.
- IOA, 12. februar 2022.
- IOA, 22. januar 2022.
- Kolokvijum 2, 10. decembar 2021.

Obaveštenja

Prezentacija predmeta

mtt.etf.rs/si/IOA/up.html

Not secure | mtt.etf.rs/si/IOA/up.html

Inženjerski optimizacioni algoritmi:

Indeks (godina/broj)_: /

Odaberite ZIP sa rešenjem koji ćete poslati:

Choose File No file chosen

Pošaljite ZIP sa rešenjem:

Upload ZIP fil

Историја курса и примена у индустрији

WIPL-D Optimizer

WIPL-D Optimizer is a powerful multi-alg
optimization tool that is being used by many suc
professionals around the world. The tool calculate
solution as well as multiple solutions for complex
criteria optimizations. Thanks to its simple and i
graphical interface, you can quickly solve the prob
hand. It enables a high level of design automatior
antenna, antenna system, scatterer, or a microwave

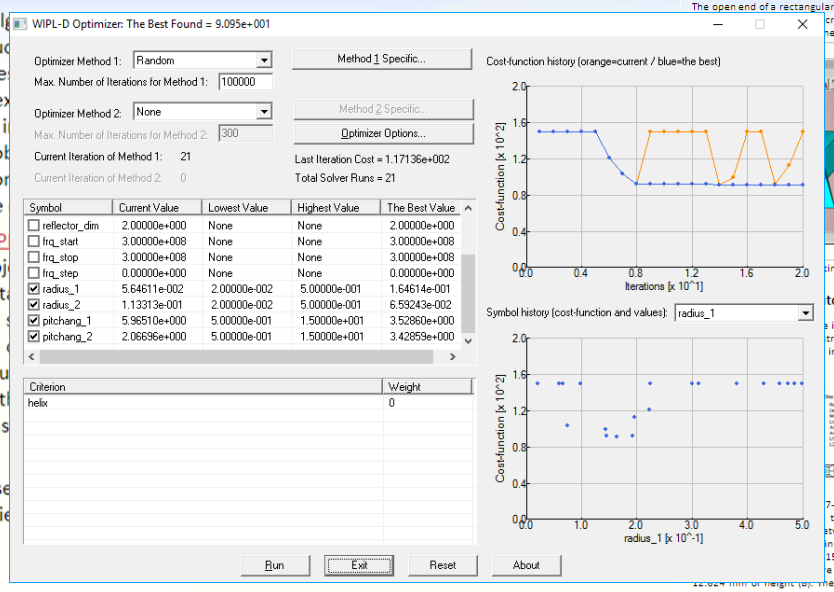
WIPL-D Optimizer is seamlessly integrated with WIP
and WIPL-D Microwave. When you create your proj
want to get optimum performance, you just st
Optimizer from within the design environment,
parameters to be optimized and the optimization
and let the tool do the rest. You can specify cost-fu
based on virtually all the EM simulation results t
calculated with WIPL-D 3D EM solver, as well as
simulation results from WIPL-D Microwave.

Various optimization algorithms are available. The se
covers all major methods proven to work efficie
practice. The available optimization algorithms are:

- Particle Swarm
- Genetic
- Simplex
- Random
- Systematic Search
- Simulated Annealing
- Gradient

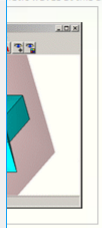
Some of the special options that distinguish WIPL-D Optimizer from other commercially available optimization tools:

- Hybrid optimization (involving two consecutive optimization algorithms)
- Pareto fronts (set of the best compromises in a multi-criteria optimization)
- Estimation of local minima (keeping several solutions that are in some proximity of the best found solution)
- Optimization repetitions (multiple optimizations from random starting points - decreases the probability of finding a local optimum which is not in fact the global optimum)



Simple Example: Optimized horn antenna

The open end of a rectangular waveguide is a source of electromagnetic waves, but it also represents a discontinuity creation of unwanted higher modes. A horn is added at the open end to reduce the hetic waves at this discontinuity and to diminish the higher modes.



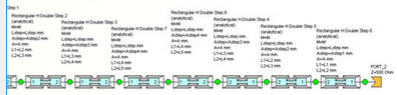
Let us assume that the length of the horn is specified. In that case, we can only change the surface of the open end of the horn in order to achieve greater gain. But, how can we achieve the best possible gain? On one hand, increase of the horn's aperture leads to decrease of the antenna's gain due to changes in the distribution of phase at the opening. On the other hand, by increasing the surface of the opening, the physical surface of the antenna is being increased and so is its effective surface. This leads to the augmentation of gain.

Let's say that we want to achieve the gain along the axis of the waveguide greater than 20 dB, and we wonder how large the surface of the opening needs to be. We than set the optimization criteria and the range for height and width of the horn and then we run the WIPL-D Optimizer.

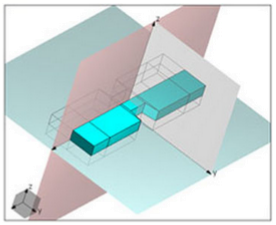
When the optimization procedure finishes, we get the optimum dimensions for the aperture of the horn. So, by setting only few imum dimensions of the horn in just a few seconds.

tomated Design of a Waveguide Filter

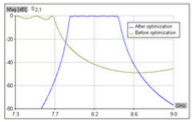
in which you can see the possibility of optimization-based design of a microwave circuit trate an effective procedure for designing waveguide filters consisting of series of coupled initial guess. The coupling is performed by using arbitrary waveguide discontinuities (e.g.,



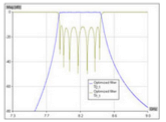
7-pole filter were -60 dB to 7.7 GHz and -0.3 dB tween 7.9 and 8.4 GHz. The n standard rectangular 153 IEC-R84 (WR-112). e 28.5 mm of width (A) and 12.624 mm of height (B). The dielectric is air and the metal is copper. The model is made using analytical models of rectangular H double step components in WIPL-D Microwave Pro environment.



The obtained filter mostly satisfies the desired characteristic. Agreement in stopband was excellent (60 dB were obtained) and insertion loss was almost (up to 0.38 dB with losses included) in the range requested. WIPL-D software enables fast and practical waveguide filter design. Even when there is only a theoretical image of solution, WIPL-D Microwave and WIPL-D Optimizer can be used to find the filter that satisfies the given criteria.



S21 parameter before and after optimization

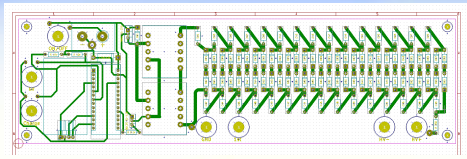
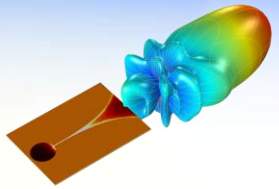
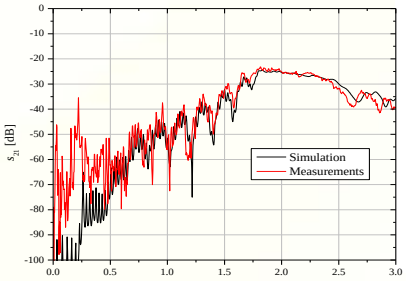


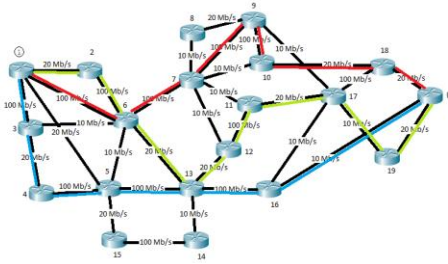
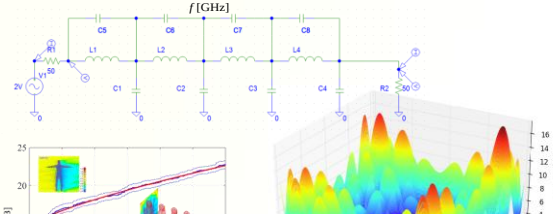
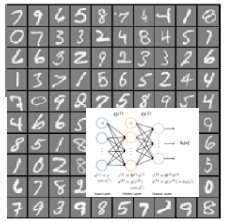
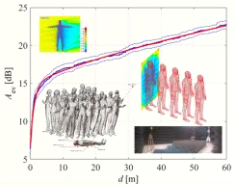
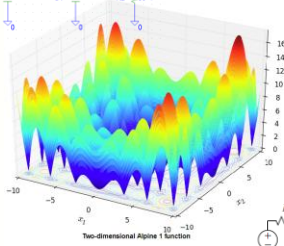
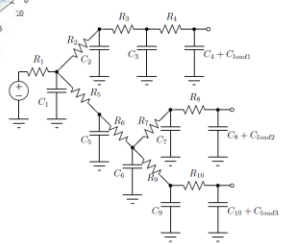
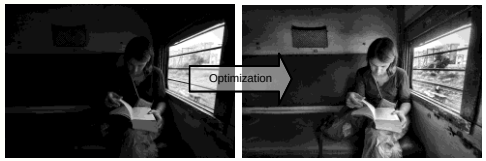
Final S11 and S21 - optimized filter

Copyright © 2013 WIPL-D d.o.o. All rights reserved.

Примери оптимизације из електротехнике и рачунарства

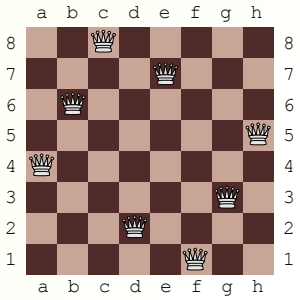
- Пројектовати електрично коло са одговарајућим особинама (преносне функције филтра, појачање појачавача, израда елемената кола у различитим технологијама, Q-фактор калема итд.)
- Пројектовање антене за задати дијаграм зрачења
- Проналажење оптималног рутирања у мрежама
- Проналажење оптималних кодова за пренос и запис према различитим критеријумима
- Оптимално покривање подручја радио сигналом
- Оптимално искоришћење фреквенцијског спектра
- Минимизација површине чипа и максимизације његових рачунарских перформанси
- Минимизација времена извршавања програма
- Проналажење функције која оптимално фитује задати скуп података
- Минимизација потрошње електричне енергије уређаја
- Максимизација капацитета батерије
- …

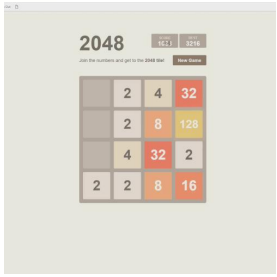




Други примери оптимизације

- Оптималан избор школе
- Оптималан избор изборног предмета на студијама
- Оптималан избор (будућег) радног места
- Оптимално коришћење свог и туђег времена
- Оптимална расподела економских ресурса
- Најбржа путања за задату полазну и крајњу тачку
- Избор оптималног потеза у игри
- Максимизација учинка радника
- Минимизација цене производње
- Оптималан избор тима за пројекат
- Оптимална расподела информација за постизање жељеног циља
- …





Како је оптимизација повезана са другим научним дисциплинама?

- Инжењери користе знања математике, логике, економије, искуство и интуицију да пронађу решење проблема
- Математика: решавање система нелинеарних једначина на континуалном или дискретном домену
- Pattern recognition / machine learning / data mining / knowledge discovery су уско повезани са оптимизацијом
- Оптимизација се врши алгоритмима који се данас по правилу реализују програмирањем
- Дарвин: еволуција ствара и чува особине које су боље за опстанак у условима у којима се одиграва природна селекција
- Економија је наука која се бави проучавањем како друштво управља ограниченим ресурсима

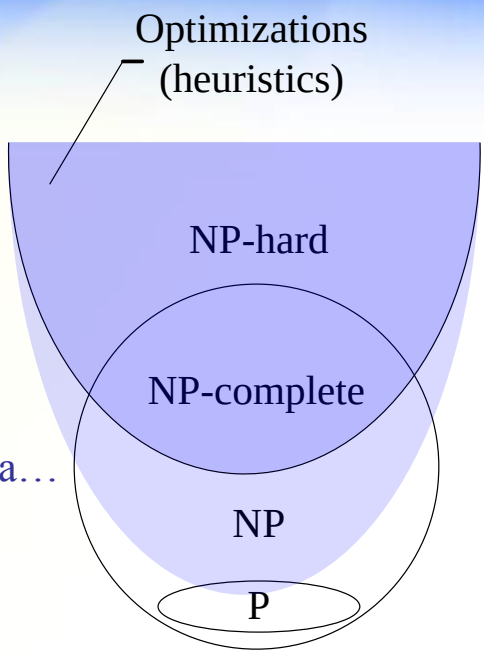
Шта је “проблем” или задатак?

- Проблем (задатак) постоји онда када желимо нешто да решимо (постоји несклад/разлика између текућег и жељеног стања)
  - Задатак на испиту: имам поставку, а желим решење
  - Електротехника: направи електрични уређај или систем задатих карактеристика
  - Рачунарство: направи софтвер задатих карактеристика
- Реални проблеми се увек решавају ван јасно дефинисане области (на факултету, по правилу се решавају у оквиру области)
  - Велики и први корак ка решавању је знати одакле почети
- Решење је постизање жељеног циља у оквиру ограничења
- У пракси најчешће мора да се решава више проблема истовремено, а ти проблеми често могу имати опречне захтеве
- Постоје разни проблеми – ми ћемо се бавити рачунским

Рачунски проблем (енг: computational problem)

- Проблем који се решава помоћу рачуна (и рачунара)
- Проблем = скуп свих вредности улазних података + одговарајући резултат за сваку вредност улазних података
  - Скуп улазних података може бити коначан или бесконачан
- Подела:
  - Проблеми одлучивања: имају бинаран резултат {да, не} или {1, 0} или {true, false}
  - Проблеми оптимизације: имају генералан нумерички резултат {реалан број, низ бита, скуп бројева, пермутација елемената скупа итд.}
- Проблеми оптимизације могу се трансформисати у проблеме одлучивања формулацијом питања
  - Оптимизација: пронаћи најкраћу путању између задатих тачака
  - Одлука: да ли постоји путања краћа од *x* [m] између задатих тачака?
  - Последица: **теорија рачунске сложености** (енг: computational complexity theory) **важи и за оптимизационе проблеме**!

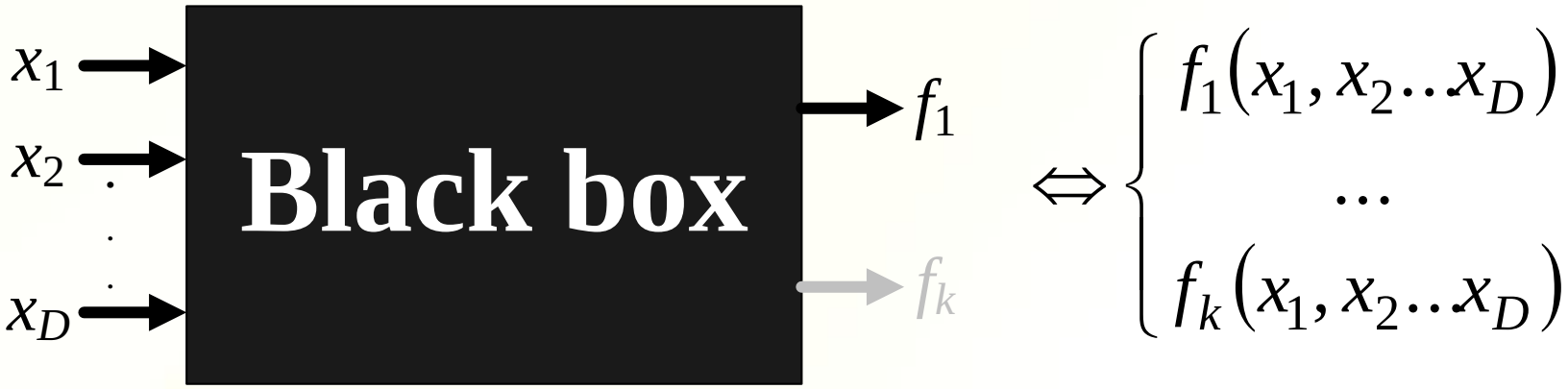
Сложеност алгоритама и поређење перформанси



- Тјурингова машина
- Детерминистички и недетерминистички приступи
- Временска сложеност и просторна сложеност:
  рачунарски ресурси потребни за решавање проблема
  = време (рачунске операције) + простор (меморијске локације)
- Претпоставимо да алгоритам обрађује N улазних елемената
- Сложеност алгоритма
  - линеарна, квадратна, логаритамска, полиномска, експоненцијална…
- Означавање
  - $O(N)$, $O(N^2)$, $O(\log N)$, $O(N^{\text{const.}})$, $O(2^N)$ итд.
- Теорија: алгоритми се пореде према сложености
- Питање од милион долара: P ≠ NP или P = NP?
- Пракса: $t = k_t \cdot O(f_t(N))$ и $m = k_m \cdot O(f_m(N))$
  да ли проблем може да се реши на задатом хардверу у задатом времену?
- За стохастичке оптимизационе алгоритме тражићемо и
  најбрже проналажње (средњег) најбољег решења

Black-box оптимизације (Модел црне кутије)

- У пракси решавање се своди на проблеме за које не постоји предзнање
  - све што знамо о проблему потребно је искористити за решавање
- За задате улазне податке (побуду) $\mathbf{x} = (x_1, x_2, \ldots x_D)$ добијају се резултати (одзиви) $\mathbf{f} = (f_1, f_2, \ldots f_k)$
- Одзиви нису познати унапред за све могуће побуде (ако су сви одзиви познати, најбољи одзив је решење проблема)
- Унутрашња организација непозната (због природе, ограничења…)



Резултати црне кутије:

“The Blind Men and the Elephant”

- John Godfrey Saxe (1816-1887) -

It was six men of Indostan
To learning much inclined,
Who went to see the Elephant
(Though all of them were blind),
That each by observation
Might satisfy his mind.

The *First* approached the Elephant,
And happening to fall
Against his broad and sturdy side,
At once began to bawl:
"God bless me! but the **Elephant**
Is very like a WALL!"

The *Second*, feeling of the tusk,
Cried, "Ho, what have we here,
So very round and smooth and sharp?
To me 'tis mighty clear
This wonder of an **Elephant**
Is very like a SPEAR!“

The *Third* approached the animal,
And happening to take
The squirming trunk within his hands,
Thus boldly up and spake:
"I see," quoth he, "the **Elephant**
Is very like a SNAKE!“

The *Fourth* reached out an eager hand,
And felt about the knee
"What most this wondrous beast is like
Is mighty plain," quoth he:
'"Tis clear enough the **Elephant**
Is very like a TREE!“

The *Fifth*, who chanced to touch the ear,
Said: "E'en the blindest man
Can tell what this resembles most;
Deny the fact who can,
This marvel of an **Elephant**
Is very like a FAN!“

The *Sixth* no sooner had begun
About the beast to grope,
Than seizing on the swinging tail
That fell within his scope,
"I see," quoth he, "the **Elephant**
Is very like a ROPE!“

And so these men of Indostan
Disputed loud and long,
Each in his own opinion
Exceeding stiff and strong,
Though each was partly in the right,
And all were in the wrong!

Формализација

- Формално, сваки одзив је функција више променљивих (побуда)
- Одзив, или нека функција одзива, је мера колико је решење добро или лоше
- При оптимизацији (решавању проблема), за сваки скуп улазних података које желимо да разматрамо, потребно је израчунавати одзив(е)
- Подела модела (поставки) према броју одзива:
  - са једним одзивом (један критеријум оптимизације)
  - са више одзива (више критеријума оптимизације)
- Први део курса: оптимизације са једним критеријумом
- Последња трећина курса: вишекритеријумске оптимизације

Оптимизација и функције више променљивих

- Побуде ↔ оптимизационе променљиве
- Одзив ↔ оптимизациона функција (нумеричка мера квалитета решења)
  - Оптимизациона функција је функција са једном или више променљивих $f(x_1, x_2 .. x_D)$
  - Променљиве и резултат оптимизационе функције могу бити из скупа дискретних или континуалних бројева

Терминологија

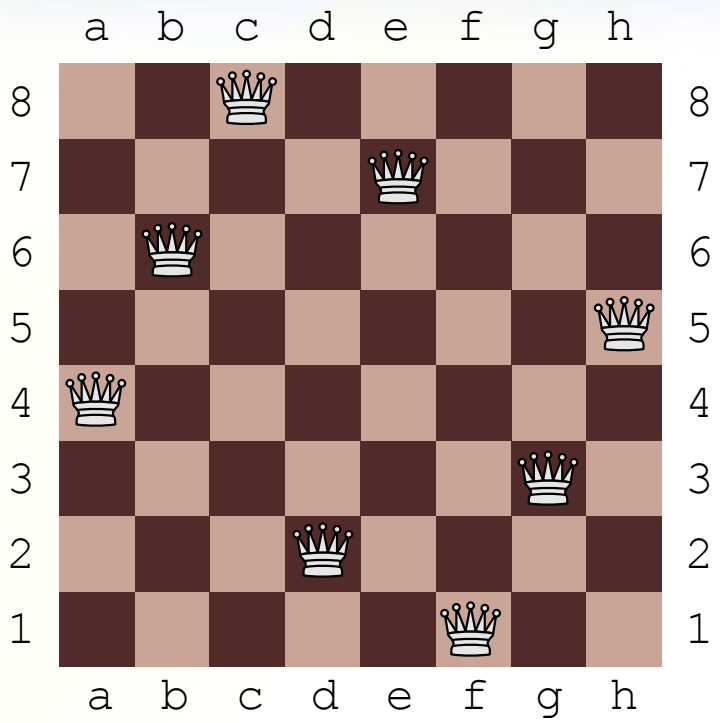
- **Оптимизациона функција** ≡
  функција грешке ≡ cost function ≡ evaluation function ≡
  нумеричка мера квалитета разматраног решења
- Једно израчунавање оптимизационе функције ≡ **итерација**
- **Оптимизационе променљиве** ≡ улазни подаци за проблем
- **Оптимизациони простор** (*S*) је
  скуп свих могућих вредности оптимизационих променљивих
  (енг: optimization space, search space, set of candidate solutions…)
- **Простор (смислених) решења** (*F*) је
  скуп свих решења које има смисла разматрати
  (енг: feasible solutions,…) $F \subseteq S$
- **Број димензија оптимизационог простора** (*D*)
  је број оптимизационих променљивих
- Свака тачка у оптимизационом простору
  представља један избор вредности улазних података

Колико је велики оптимизациони простор?

- Величина оптимизационог простора је од изузетне важности
- Избор оптимизационог алгоритма зависи од:
  - величине оптимизационог простора
  - домена оптимизационих променљивих
- Величина простора зависи од усвојеног записа!
  - Бијективни записи имају исту величину оптимизационог простора
  - Најбољи запис је онај за који важи $F = S$
- Оптимизациони простор може бити ограничен или неограничен (constrained vs. unconstrained)

Пример записа и величине опт. простора: *N*-краљица

- Проблем: поставити *N*-краљица на “шаховску таблу” димензија *N* x *N* тако да се краљице не нападају
- Нека су r_k и c_k позиције краљице k:
  $1 \leq k \leq N$ (редни број)
  $1 \leq r_k \leq 8$ (ред)
  $1 \leq c_k \leq 8$ (колона)
- Краљица:
  `f1->(`r_k`,`c_k`)=(1,6)`
- Колико је велики оптимизациони простор?



Величина оптимизационог простора зависи од записа!

- Запис #1: 64 могућа места (1,1), (1,2), …(8,8) на која можемо да ставимо 8 краљица
  x = $\{K_1, K_2, \ldots K_8\}$, $1 \leq K_n \leq 64$

$$|F| = \binom{64}{8} = 4\,426\,165\,368$$

- Запис #2: свака краљица у свом реду
  x = $\{K_1, K_2, \ldots K_8\}$, $1 \leq K_n \leq 8$
  К#1: (1, K_1),…К#8: (8, K_8)

$$|F| = 8^8 = 16\,777\,216$$

- Запис #3: краљице не могу бити у истом реду и у истој колони
  x = {3,6,2,4,1,5,7,8}
  К#1: (1,3), К#2: (2,6), …К#8: (8,8)

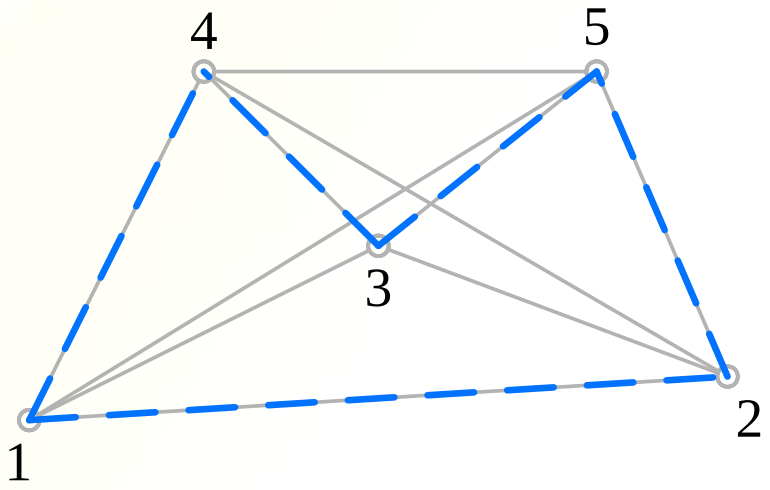
$$|F| = 8! = 40\,320$$

Основни типови оптимизационих проблема (TSP, SAT, NLP)

- Проблем трговачког путника (енглески: traveling salesman problem, TSP)
- Булова алгебра (дискретна стања) (енглески: Boolean satisfiability, SAT)
- Нелинеарни проблеми (енглески: Nonlinear programming, NLP)
- Домени основних типова проблема (дискретан или континуалан)

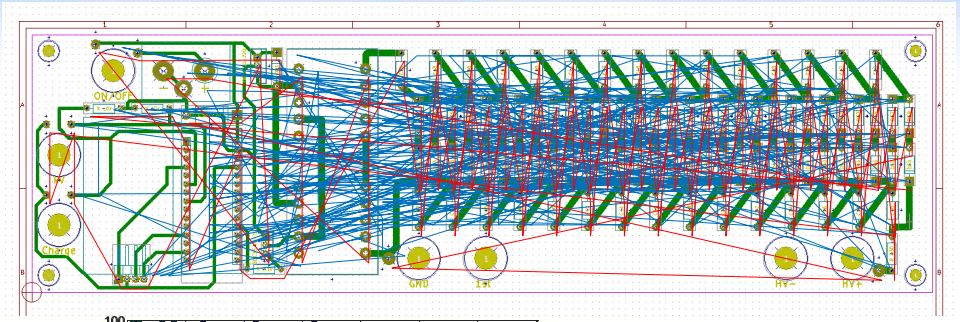
TSP: поставка

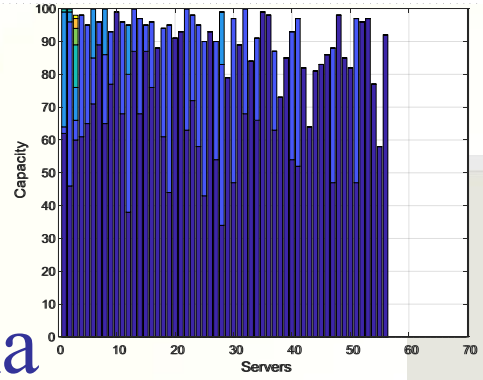
- За задати скуп D градова и позната растојања између сваког пара градова, пронаћи најкраћи пут који пролази кроз сваки град и завршава се у полазном граду
- Број могућих путања $D!$
  - $D = 15$, број путања $\approx 1{,}3 \cdot 10^{12}$
  - $D = 150$, број путања $\approx 57{,}1 \cdot 10^{261}$
- Приказана путања
  5-3-4-1-2(-5)
- Подела:
  - Симетричан: $dist(p,q) = dist(q,p)$
    - Асиметричан: $dist(p,q) \neq dist(q,p)$
  - Метрички ($d_{AB} \leq d_{AC} + d_{CB}$) и они који то нису
  - Еуклидски: симетричан + метрички

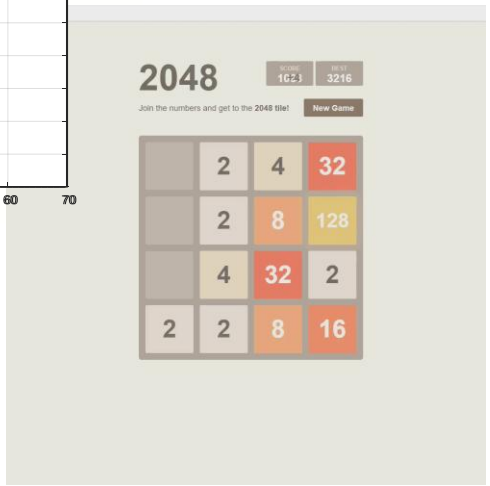


TSP: инжењерски проблеми

- Минимална путања алата за бушење на штампаним плочама
- Рутирање возила
- Путање у графовима
- Планирање и логистика
- …







TSP класа проблема и поделе

- Сваки проблем који може да се сведе на **проверу свих пермутација** називаћемо **проблем TSP класе**
- Енглески: TSP-encoded problems
- Поделе TSP:
  - Симетричан (енг: symmetric TSP, STSP)
  - Асиметричан (енг: asymmetric TSP, ATSP)
  - Најкраћи Хамилтонов пут (енг: shortest Hamiltoninan path, SHP)
    - Хамилтонова контура (енг: Hamiltonian cycle)
  - Обилазак са условима (енг: Sequential ordering problem, SOP)
  - Рутирање возила ограниченог капацитета (енг: Capacitated vehicle routing problem, CVRP)
  - …
- Посебне потклассе могу имати једноставнија решења
- Генерализација: job shop problem

SAT: поставка

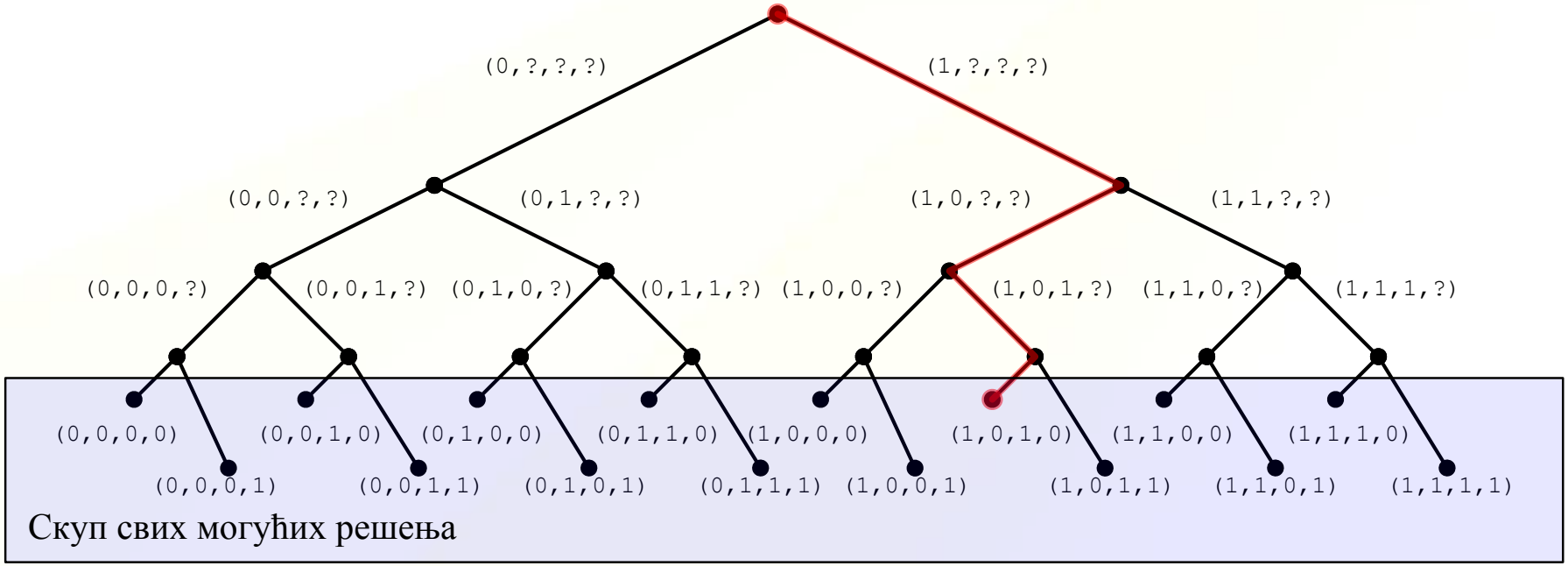
- За задати логички израз $F(x_1, x_2, \ldots x_D)$ пронаћи вредности променљивих
  $(x_1, x_2, \ldots x_D)$, $x_k \in \{\text{true (1), false (0)}\}$, $k = 1,2,\ldots D$
  тако да је $F(x_1, x_2, \ldots x_D) = \text{true}$

$$F(\mathbf{x}) = x_1 \wedge ((x_2 \wedge \bar{x}_1) \vee (\bar{x}_2 \wedge x_3 \wedge \bar{x}_4))$$

- Пример $\mathbf{x}_{\text{opt}} = (x_1, x_2, x_3, x_4) = (1,0,1,0)$
  $F(\mathbf{x}_{\text{opt}}) = \text{true}$
- Број различитих $\mathbf{x}$ је 2^D
  - $D = 15$, број могућих решења 32768
  - $D = 150$, број могућих решења $\approx 1{,}4 \cdot 10^{45}$
- Улазни подаци могу да се запишу као низ бита 100110...

SAT: приказ помоћу графова (бинарно стабло графа)

$$F(\mathbf{x}) = x_1 \wedge ((x_2 \wedge \overline{x}_1) \vee (\overline{x}_2 \wedge x_3 \wedge \overline{x}_4))$$



Скуп свих могућих решења

SAT класа проблема и поделе

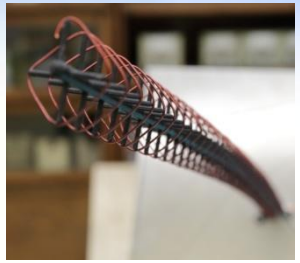
- Сваки проблем који се своди **на проверу свих могућих вредности секвенце бита** називаћемо проблем SAT класе
- Поделе (Karp's 21 NP-complete problems)
  - проблем ранца (knapsack)
  - подела посла (job-sequencing)
  - directed/undirected Hamiltonian cycle
  - …
- Примери: свака манипулација битима може да претвори у проблем SAT класе
  - Откључавање ZIP архиве
  - Пробијање RSA кодова
  - ...

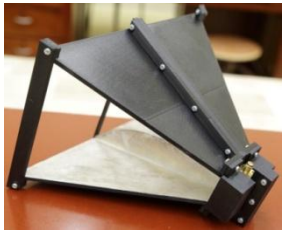
NLP проблеми

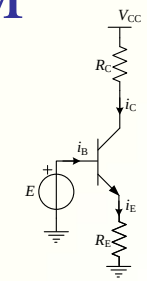
- Сваки оптимизациони проблем са **континуалним променљивима** спада у NLP класу проблема
- Број могућих решења је (теоријски) **бесконачан**!
- Посебни случајеви
  - диференцијабилне и недиференцијабилне *f*
  - линеарно програмирање
  - квадратно програмирање
  - конвексни/конкавни проблеми
  - са ограниченим или неограниченим доменом
  - ...

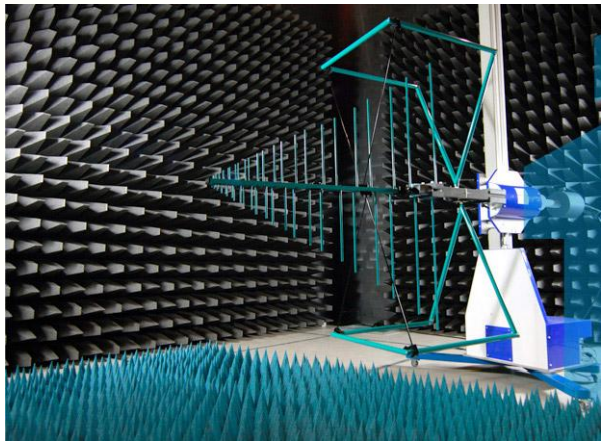
NLP инжењерски примери

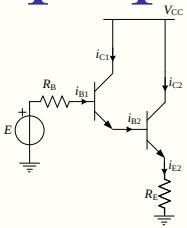
- Пројектовање ЕМ уређаја
- Решавање нелинеарних кола
- Фитовање резултата мерења
- Управљање нелинеарним системима
- Максимизација профита
- ...

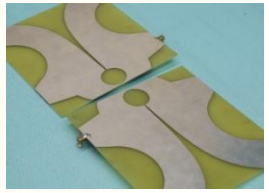




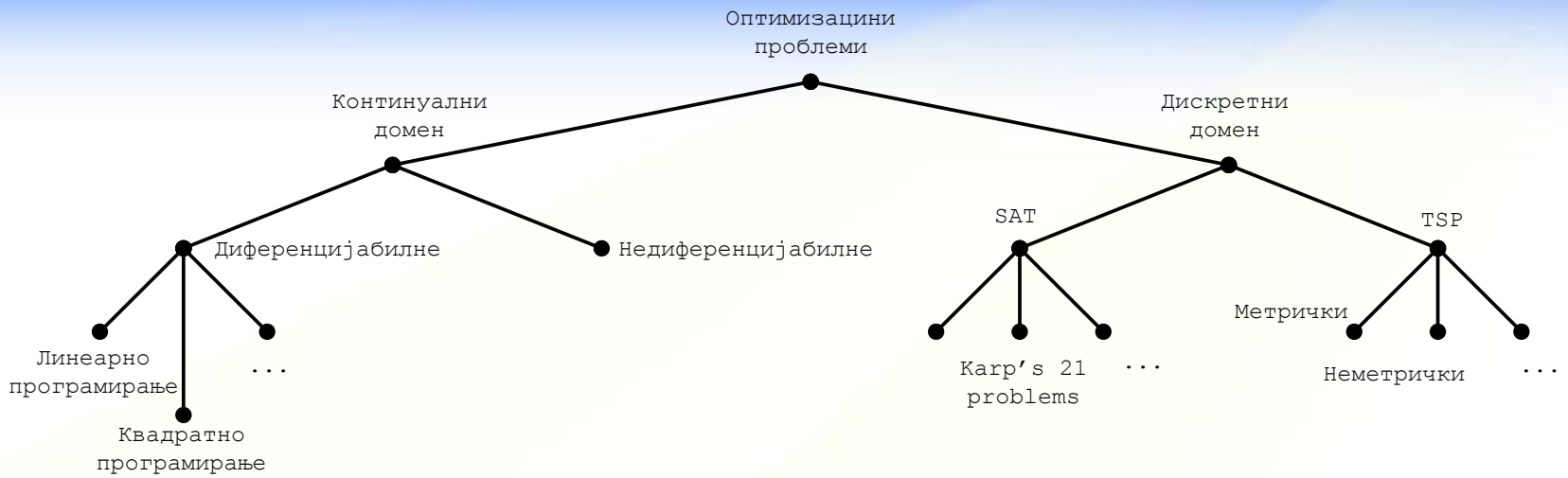








Класификација оптимизационих проблема



- Подела није јединствена
- Пермутације су специјалан случај низова бита: TSP ⊂ SAT ?
- Све што радимо на рачунару је са коначном тачношћу (64-бита): NLP ⊂ SAT ?
- Класификација је према **запису улазних података** (побуда) и **природи** оптимизационог проблема

Колико је спор мој брзи рачунар?

- Колико времена могу да потрошим за оптимизацију?
- Колико пута могу да израчунам опт. функцију $f(\mathbf{x})$?
- Желим процену брзине: маскималан број позива (оптимизационе) функције
- Једноставан програм: прототип оптимизације

```c
#include <stdio.h>
#include <time.h>

double F(double a, double b)
{
    return a + b;
}

int main(void)
{
    double x,eval;
    double max = 7e9;

    time_t t1, t2;
    time(&t1);

    for (x=0; x < max; x=x+1.0 )
        eval = F(x,x);

    time(&t2);

    printf("%5.5e\n", (t2 - t1)/max);
    return 0;
}
```

Колико времена је потребно да решимо проблеме?

- Претпоставимо да је свака провера 1ns
- TSP 100 градова:
  $100! \cdot 1\text{ns} \approx 3 \cdot 10^{141}$ година!
- SAT 100 бита:
  $2^{100} \cdot 1\text{ns} \approx 4 \cdot 10^{13}$ година!
- NLP: **x** има ∞ могућности
  $\rightarrow$ потребно је теоријски бесконачно много времена!

Решавање реалних оптимизационих проблема

- Број могућих решења је изузетно велики те је потребно изузетно много рачунарских ресурса
- Постоји много **ограничења** тако да је тешко наћи било какво решење (а не оптимално)
  - Посебно ограничење: човек који решава проблем није адекватно припремљен
- Често је довољно наћи **задовољавајуће решење**
  - Трагање за најбољим (оптималним) решењем је можда занимљиво али најчешће неприхватљиво
- Тада се примењују **оптимизациони алгоритми и хеуристике**

Шта су хеуристике?

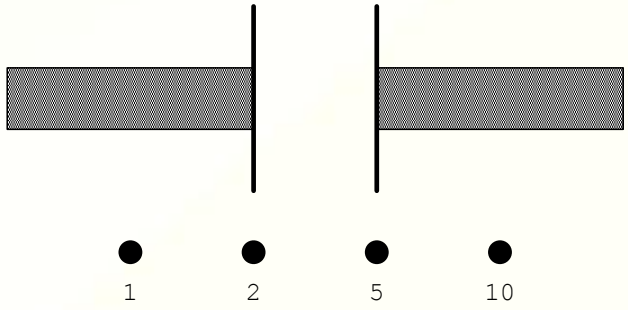
- ХЕУРИСТИКА (енг: heuristic)
  грчки корен речи “Εὑρίσκω“ – пронаћи или открити
- IEEE: **All engineering is heuristic**
- Технике решавања проблема засноване на искуству, учењу и откривању које доводе до решења (не мора нужно бити оптимално али је довољно добро)
- Када год је немогуће или непрактично потпуно претраживање простора, користе се хеуристике (интуиција, стереотипи, здрав разум, енг: rule-of-thumb, educated guess, …)
- **State-of-the-art** проблеми се **увек** решавају **хеуристички**

Шта је заједничко за све приступе решавању “проблема” (задатака)?

- Структура сваког алгоритма (приступа) за решавање “проблема” има три основна дела
  1. запис могућих решења,
  2. циљ који је потребно постићи и
  3. оптимизациону функцију (нумеричку меру појединачних решења)
- Запис (енг: representation)
  - TSP: једна пермутација, нпр. **x** = (2,1,3,6,4,5,7)
  - SAT: један низ бита, нпр. **x** = `1001100101`
  - NLP: један вектор реалних бројева, нпр. **x** = (1.5, –3.2, 4.76,17.2)
- Запис не мора да буде “природан”: TSP као бинарни низ, реални бројеви као низ бита, итд.
  - Генерално на запис се може применити било која трансформација (пресликавање) и добити други запис.
  - Бијективни записи проблема имају исту сложеност решавања!
- Циљ и оптимизациона функција НИСУ исто!
- Запис и опт. функцију формулише онај ко решава проблем

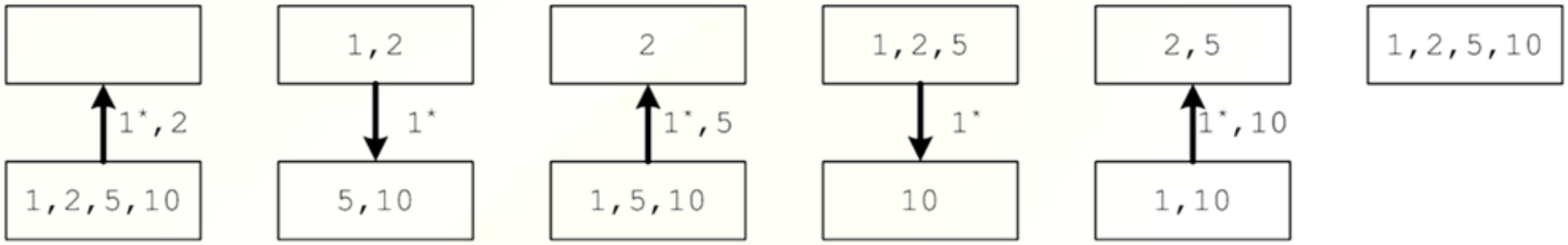
Пример:
4 човека и трошни мост

Четири особе морају да пређу са једне стране трошног моста на другу, током ноћи. Мост највише може да издржи две особе истовремено. За прелазак је неопходна лампа да би се осветлио сваки корак и избегле рупе (тј. при преласку у паровима иду брзином споријег). Постоји само једна лампа, коју није могуће пребацити са једне стране моста на другу (тј. увек неко мора да је носи). Први човек може да пређе мост за 1 min, други за 2 min, трећи за 5 min и четврти за 10 min. Пронаћи минимално време потребно да сви пређу на другу страну моста.



Пример:
Основне идеје и једно решење

- Домен проблема је дискретан, а број могућности је пребројив и коначан.
- Прелази се у паровима, а лампу враћа једна особа.
- Да би четири особе прешле, потребно је 3 преласка и 2 повратка.
- Лампу са друге стране враћа увек најбржи који је на тој страни.
- Интуитивно решење, најбржи носи (враћа) лампу и преводи једног по једног $\{1,2,1,1,5,1,1,10\}$, $T = 19\,\text{min}$.



Пример: Пребројавање решења и запис

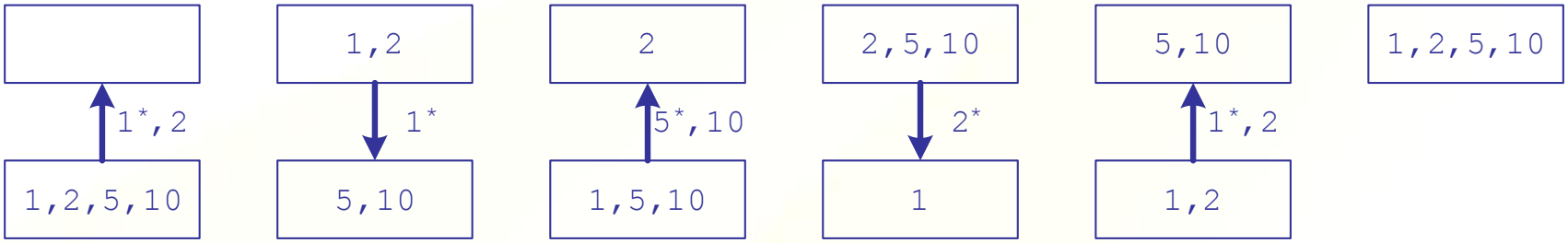
- При преласку ($\uparrow$) за избор првог пара постоји $\binom{4}{2}$ могућности, за избор другог пара $\binom{3}{2}$ могућности и за избор трећег пара $\binom{2}{2}=1$ могућност.
- При повратку ($\downarrow$) имамо само једну могућност.
- Укупан број могућих комбинација прелазака је $N=\binom{4}{2}\cdot 1\cdot\binom{3}{2}\cdot 1\cdot\binom{2}{2}=18$.
- Све комбинације можемо да запишемо у облику вектора $\{x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8\}$ где су елементи вектора брзине одговарајућих људи, тј. $x_k\in\{1,2,5,10\}$, $k=1,2,\ldots 8$. Први пар при преласку једнозначно одређују x_1,x_2, други пар x_4,x_5, трећи пар x_7,x_8, а повратке једнозначно одређују x_3 и x_6.
- Током прелазака потребно је водити евиденцију о томе ко је на којој страни (брзине једнозначно одређују људе). На почетку, на првој страни су сви $S_1=\{1,2,5,10\}$, а на другој нема људи $S_2=\{\}$. После преласка првог пара и повратка једне особе са лампом $|S_1|=3$ и $|S_2|=1$.
- Укупно време преласка је $T=\max(x_1,x_2)+x_3+\max(x_4,x_5)+x_6+\max(x_7,x_8)$.

Пример: све могућности

↑ #1		↓ #1	↑ #2		↓ #2	↑ #3		T
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	
1	2	1	1	5	1	1	10	19
1	2	1	1	10	1	1	5	19
1	**2**	**1**	**5**	**10**	**2**	**1**	**2**	**17**
1	5	1	1	2	1	1	10	19
1	5	1	1	10	1	1	2	19
1	5	1	2	10	2	1	2	20
1	10	1	1	2	1	1	5	19
1	10	1	1	5	1	1	2	19
1	10	1	2	5	2	1	2	20
2	5	2	1	2	1	1	10	20
2	5	2	1	10	1	1	2	19
2	5	2	2	10	2	1	2	21
2	10	2	1	2	1	1	5	20
2	10	2	1	5	1	1	2	20
2	10	2	2	5	2	1	2	21
5	10	5	1	2	1	1	5	23
5	10	5	1	5	1	1	2	23
5	10	5	2	5	2	1	2	24

Пример: решење

- Минимално време преласка је 17 min



Задатак за вежбе

- Купац је купио четири производа. Ако се цене изразе у доларима, онда су збир и производ свих цена (бројчано) исти и износе $7,77. Цене се заокружују на $0,01 (1 cent).
  (а) Написати програм који извршава потпуну претрагу по све четири цене и помоћу њега одредити цене. Израчунати максималан број позива оптимизационе функције.
  (б) Изразити једну цену преко осталих и написати програм који извршава потпуну претрагу по (преостале) три цене. Израчунати максималан број позива опт. функције и упоредити брзину програма у односу на програм из (а).
  (в) Који од ова два програма је бржи?

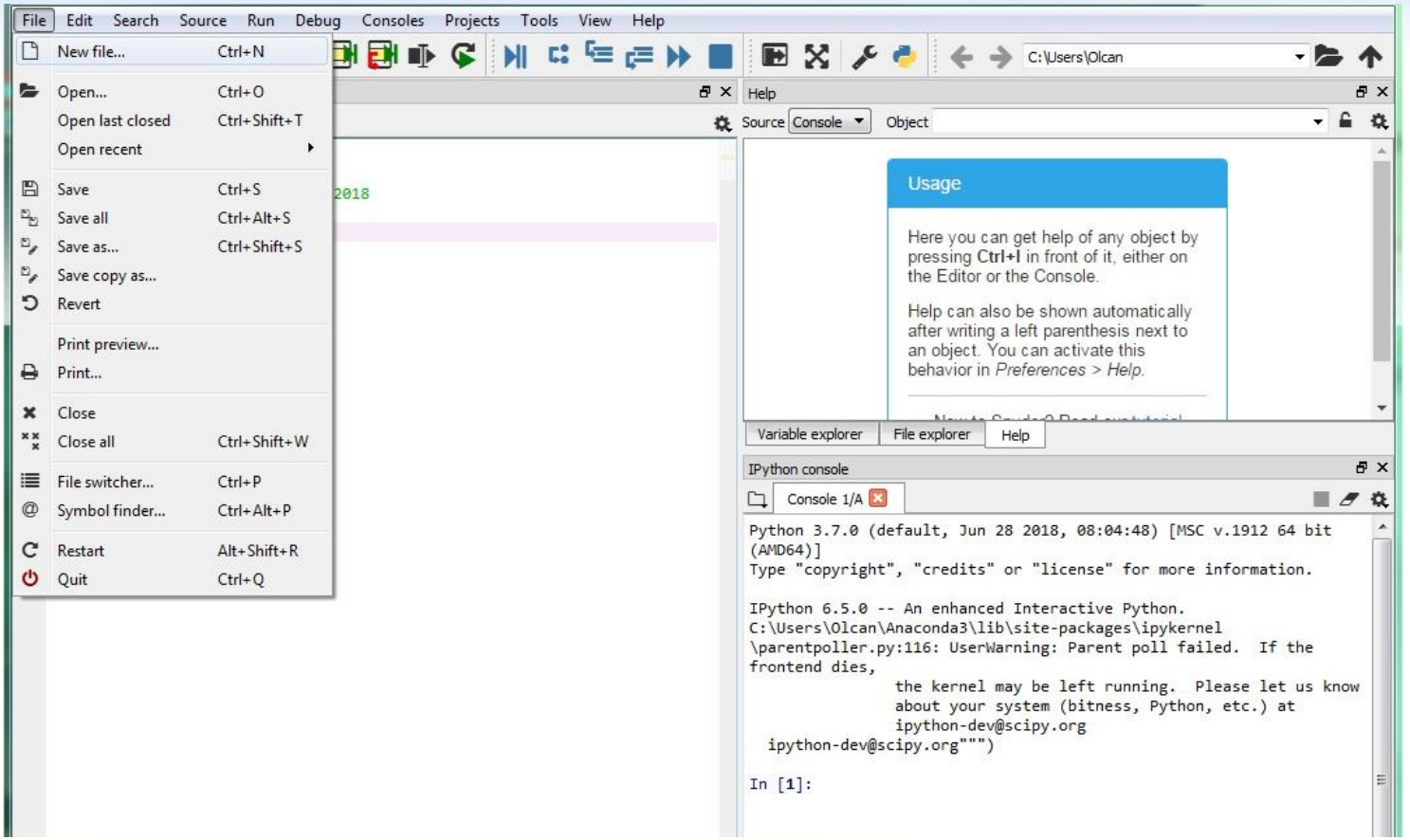
Задатак за вежбе

- Купац је купио четири производа. Ако се цене изразе у доларима, онда су збир и производ свих цена (бројчано) исти и износе $7,77. Цене се заокружују на $0,01 (1 cent).
  (а) Написати програм који извршава потпуну претрагу по све четири цене и помоћу њега одредити цене. Израчунати максималан број позива оптимизационе функције.
  (б) Изразити једну цену преко осталих и написати програм који извршава потпуну претрагу по (преостале) три цене. Израчунати максималан број позива опт. функције и упоредити брзину програма у односу на програм из (а).
  (в) Који од ова два програма је бржи?

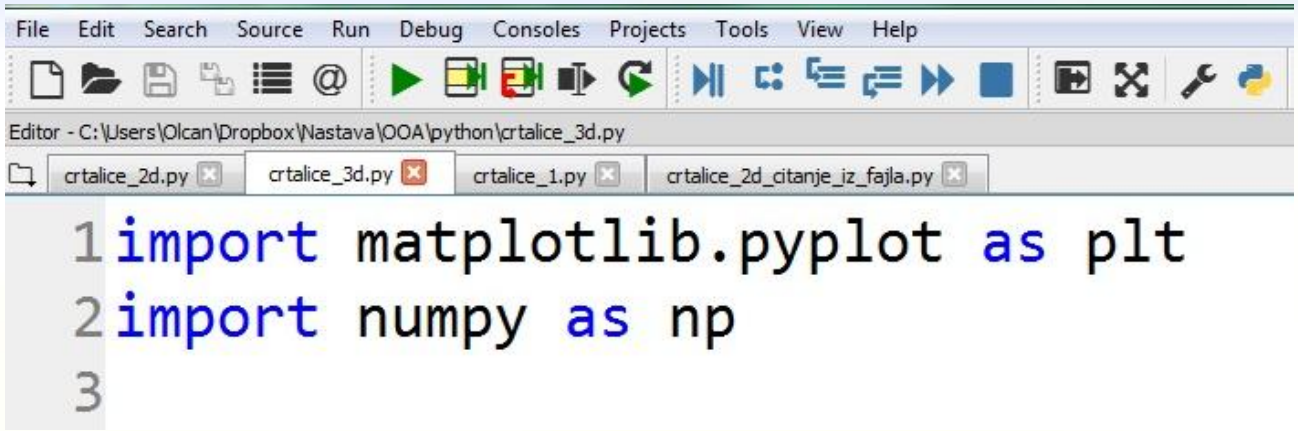
Вежбе на рачунару: Мотивација

- Упознавање са окружењима која ће бити коришћена на вежбама
  - **Python ≥ 3.8** (Spyder Anaconda 3/PyCharm): приказ и обрада резултата
  - **C/C++** (Visual Studio 2022): рачунарски захтевни делови програма
- Студенти се охрабрују да користе и своје рачунаре.

Python ≥3.8
(Spyder окружење са Anaconda 3)

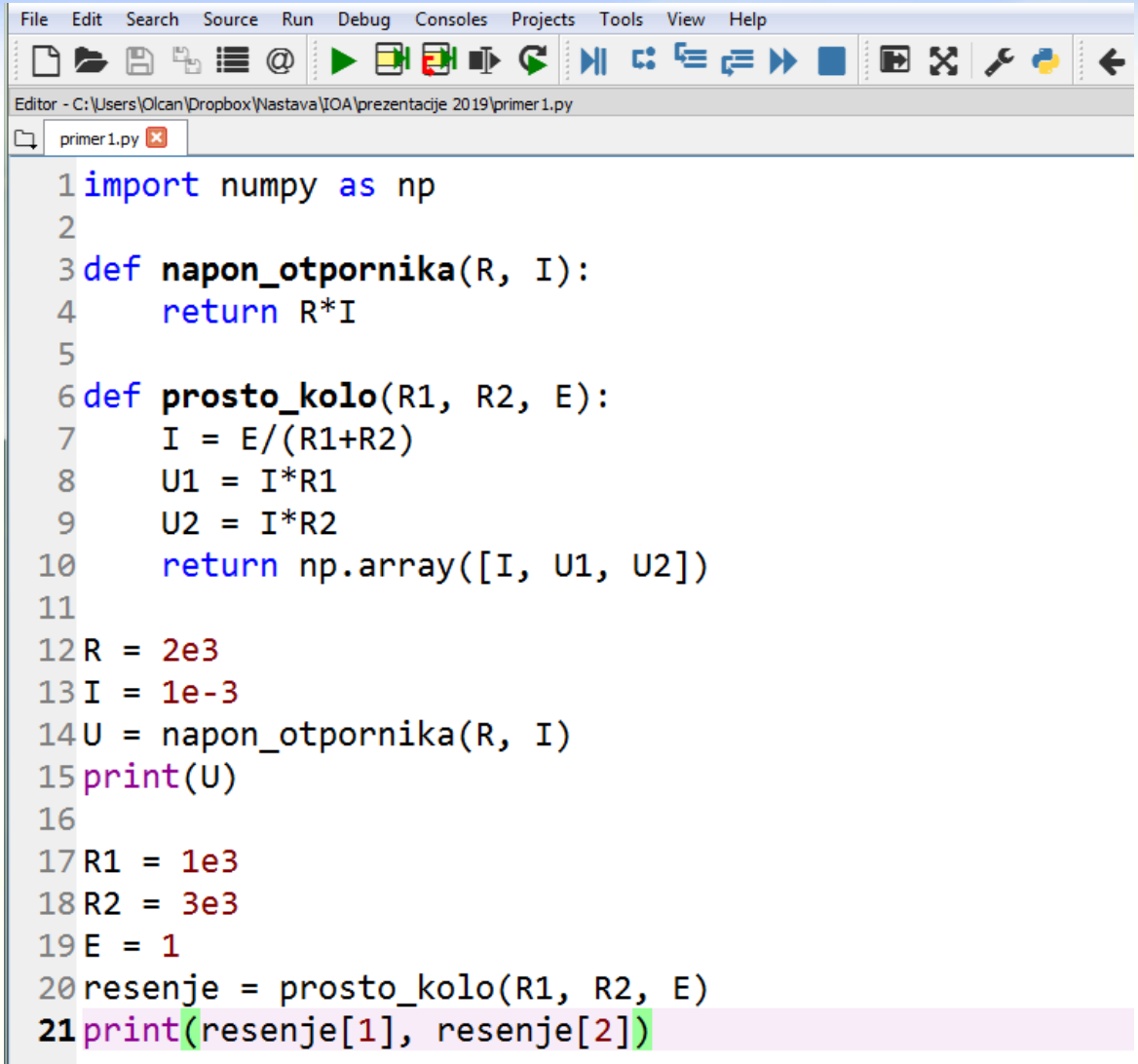


Библиотеке



- **plt**: figure(), plot(), plot_surface(), add_subplot(), xlabel(), set_zlabel(), plt.legend()
- **np**: arange(), meshgrid()
- Користан сајт: https://matplotlib.org/

Дефинисање функција



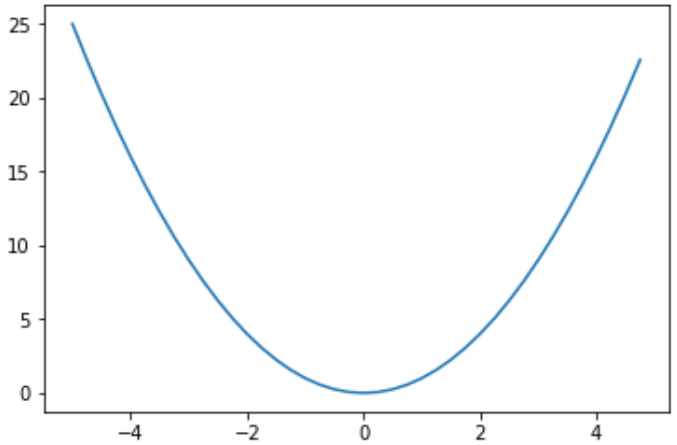
1D функција: *plot*

File Edit Search Source Run Debug Consoles Projects Tools View Help

Editor - C:\Users\Olcan\Dropbox\Nastava\OOA\python\crtalice_2d.py

crtalice_2d.py | crtalice_3d.py

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 fig = plt.figure()
5 x = np.arange(-5, 5, 0.25)
6 y = x**2
7 plt.plot(x,y)
```



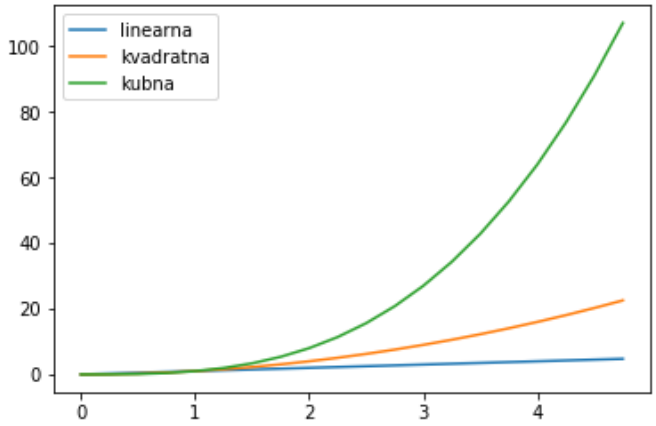
Легенда

File Edit Search Source Run Debug Consoles Projects Tools View Help

Editor - C:\Users\Olcan\Dropbox\Nastava\OOA\python\crtalice_2d.py

crtalice_2d.py* | crtalice_3d.py | crtalice_1.py | crtalice_2d_citanje_iz_fajla.py

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 fig = plt.figure()
5 x = np.arange(0, 5, 0.25)
6
7 plt.plot(x, x, label = 'linearna')
8 plt.plot(x, x**2, label = 'kvadratna')
9 plt.plot(x, x**3, label = 'kubna')
10 plt.legend()
```



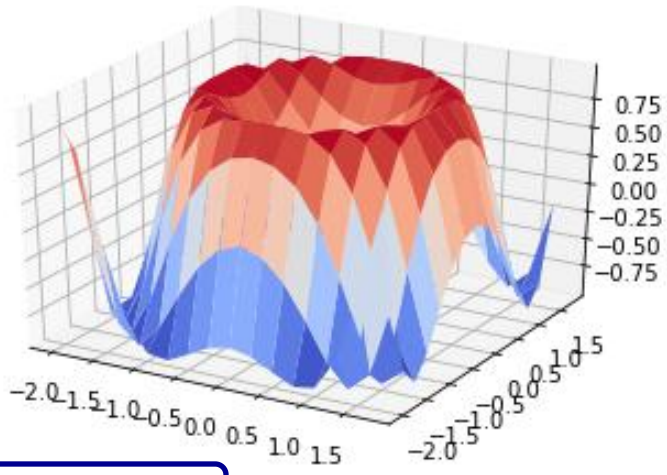
2D функција: *plot_surface*

File Edit Search Source Run Debug Consoles Projects Tools View Help

Editor - C:\Users\Olcan\Dropbox\Nastava\OOA\python\crtalice_3d.py

crtanje_s_par.py | crtalice_3d.py | crtalice_2d.py

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  fig=plt.figure()
5  ax=fig.add_subplot(projection='3d')
6
7  x=np.arange(-2,2,0.25)
8  y=np.arange(-2,2,0.25)
9  x,y=np.meshgrid(x,y)
10
11 z=np.sin(x**2+y**2)
12
13 h=ax.plot_surface(x,y,z,cmap=plt.cm.coolwarm)
```



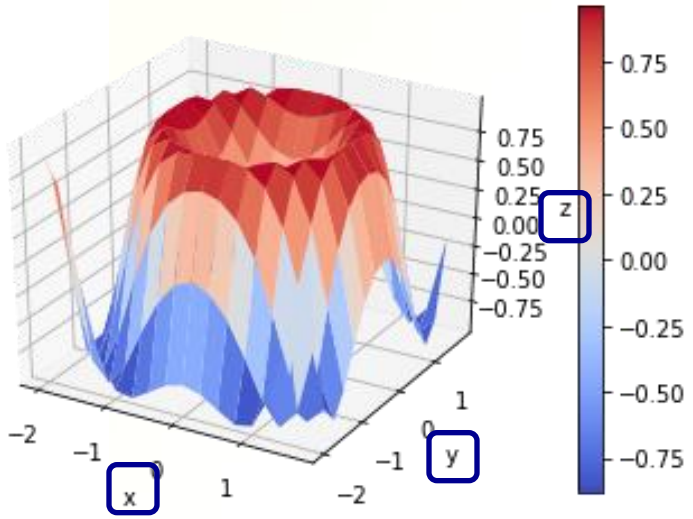
Означавање оса

File Edit Search Source Run Debug Consoles Projects Tools View Help

Editor - C:\Users\Olcan\Dropbox\Nastava\OOA\python\crtalice_3d.py

crtanje_s_par.py | crtalice_3d.py | crtalice_2d.py

```
1   import matplotlib.pyplot as plt
2   import numpy as np
3
4   fig=plt.figure()
5   ax=fig.add_subplot(projection='3d')
6
7   x=np.arange(-2,2,0.25)
8   y=np.arange(-2,2,0.25)
9   x,y=np.meshgrid(x,y)
10
11  z=np.sin(x**2+y**2)
12
13  h=ax.plot_surface(x,y,z,cmap=plt.cm.coolwarm)
14
15  ax.set_xlabel('x')
16  ax.set_ylabel('y')
17  ax.set_zlabel('z')
18
19  fig.colorbar(h)
```



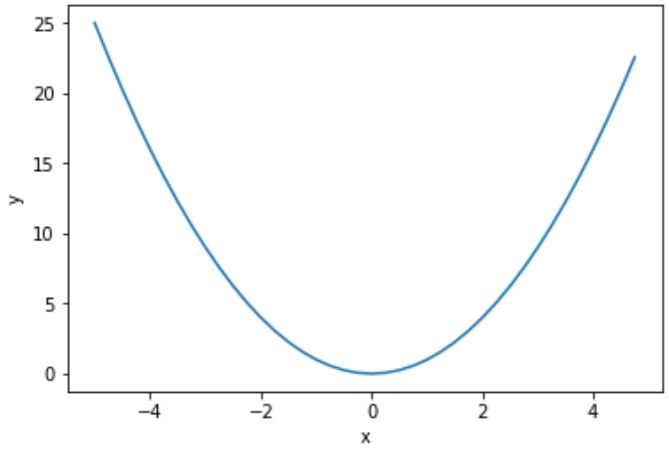
Учитавање података из датотеке

File Edit Search Source Run Debug Consoles Projects Tools View Help

Editor - C:\Users\Olcan\Dropbox\Nastava\OOA\python\crtalice_2d_citanje_iz_fajla.py

crtanje_s_par.py | crtalice_3d.py | crtalice_2d.py | crtalice_2d_citanje_iz_fajla.py

```
1 import matplotlib.pyplot as plt
2
3 file = open('funkcija.txt','r')
4 x = []
5 y = []
6 for line in file:
7     s = line.split()
8     if (len(s)!=0):
9         x.append(float(s[0]))
10        y.append(float(s[1]))
11 file.close()
12
13 plt.figure()
14 plt.plot(x,y)
15 plt.xlabel('x')
16 plt.ylabel('y')
17
```



funkcija.txt - No...

File Edit Format View Help

-5.0	25.0
-4.75	22.5625
-4.5	20.25
-4.25	18.0625
-4.0	16.0
-3.75	14.0625
-3.5	12.25
-3.25	10.5625
-3.0	9.0
-2.75	7.5625
-2.5	6.25
-2.25	5.0625
-2.0	4.0
-1.75	3.0625
-1.5	2.25
-1.25	1.5625
-1.0	1.0
-0.75	0.5625
-0.5	0.25
-0.25	0.0625
0.0	0.0
0.25	0.0625
0.5	0.25
0.75	0.5625
1.0	1.0
1.25	1.5625
1.5	2.25
1.75	3.0625
2.0	4.0
2.25	5.0625
2.5	6.25
2.75	7.5625

Ln 1,

Пример минимизације оптимизационе функције

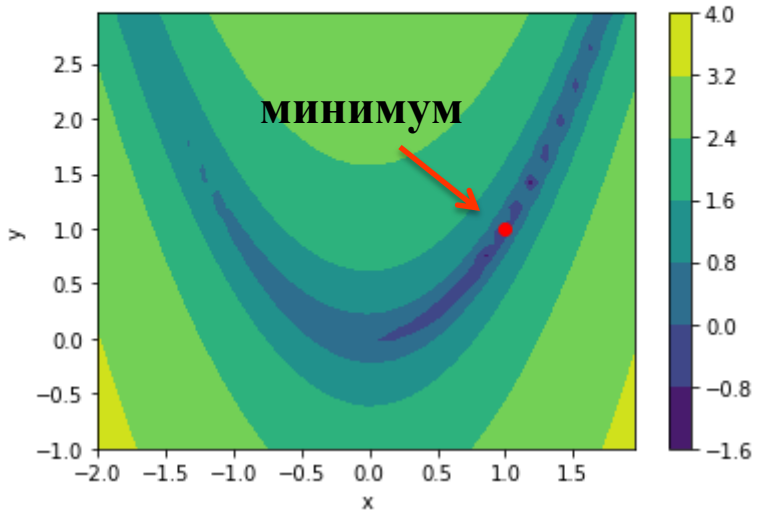
File Edit Search Source Run Debug Consoles Projects Tools View Help

Editor - C:\Users\Olcan\Dropbox\Nastava\IOA\prezentacije 2019\primer2.py

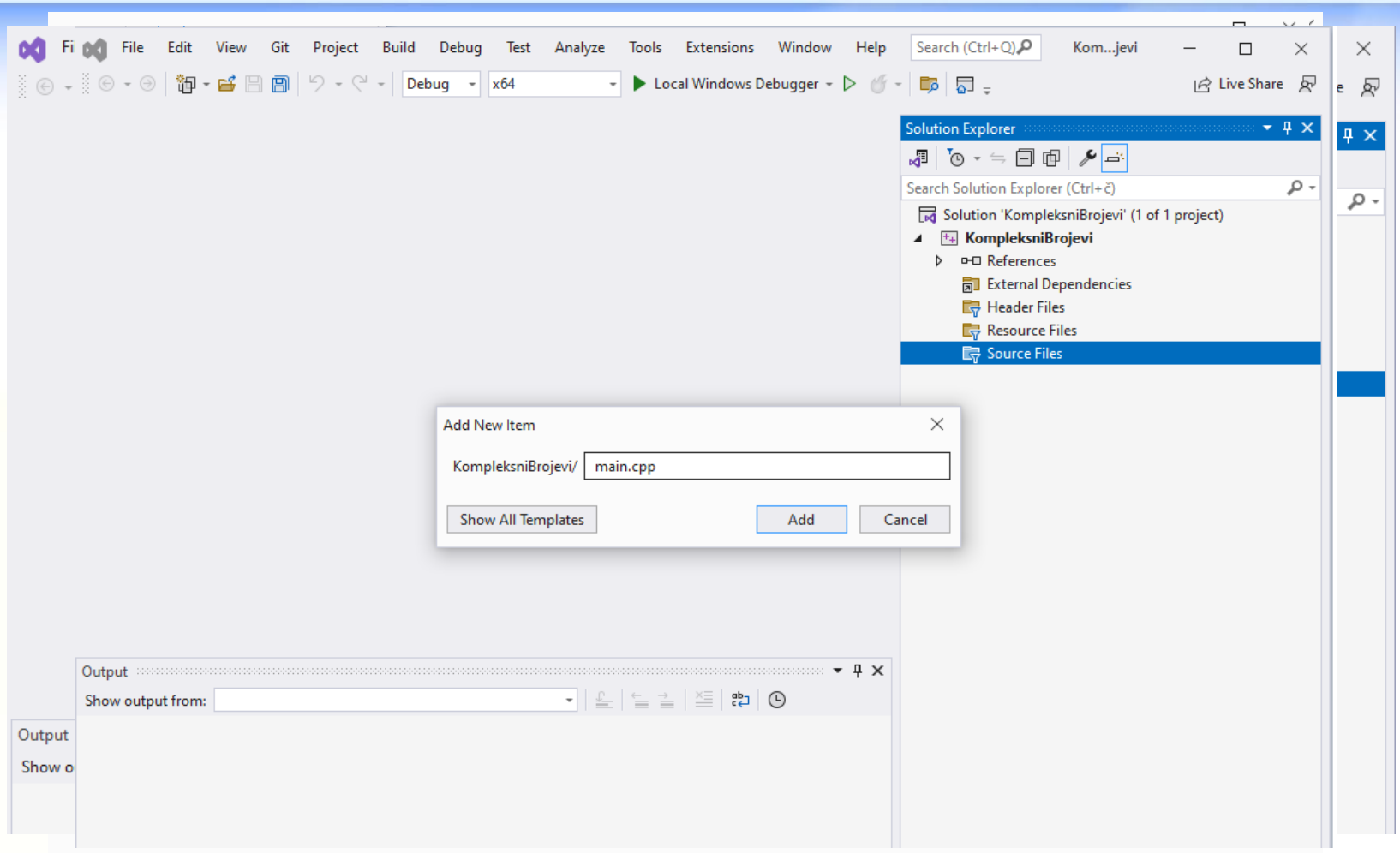
primer2.py

```
1  from scipy.optimize import minimize
2
3  import matplotlib.pyplot as plt
4  import numpy as np
5
6  def fun(x):
7      """Rosenbrock function"""
8      return (1-x[0])**2.0+100*(x[1]-x[0]**2)**2
9
10 x=np.arange(-2,2,0.11)
11 y=np.arange(-1,3,0.11)
12 x,y=np.meshgrid(x,y)
13
14 z=np.log10(fun([x,y]))
15
16 h=plt.contourf(x,y,z)
17 plt.scatter(1.0,1.0,facecolor='red')
18 plt.xlabel('x')
19 plt.ylabel('y')
20 plt.colorbar(h)
21 print(fun([1,1]))
22
23 x0 = np.array([0,0])
24 res = minimize(fun, x0, method='Nelder-Mead', options={'ftol':1e-10,'disp':True,'maxfev':1e4})
25 xmin = np.array(res.x)
26 print('xmin=(',xmin[0],xmin[1],')')
```

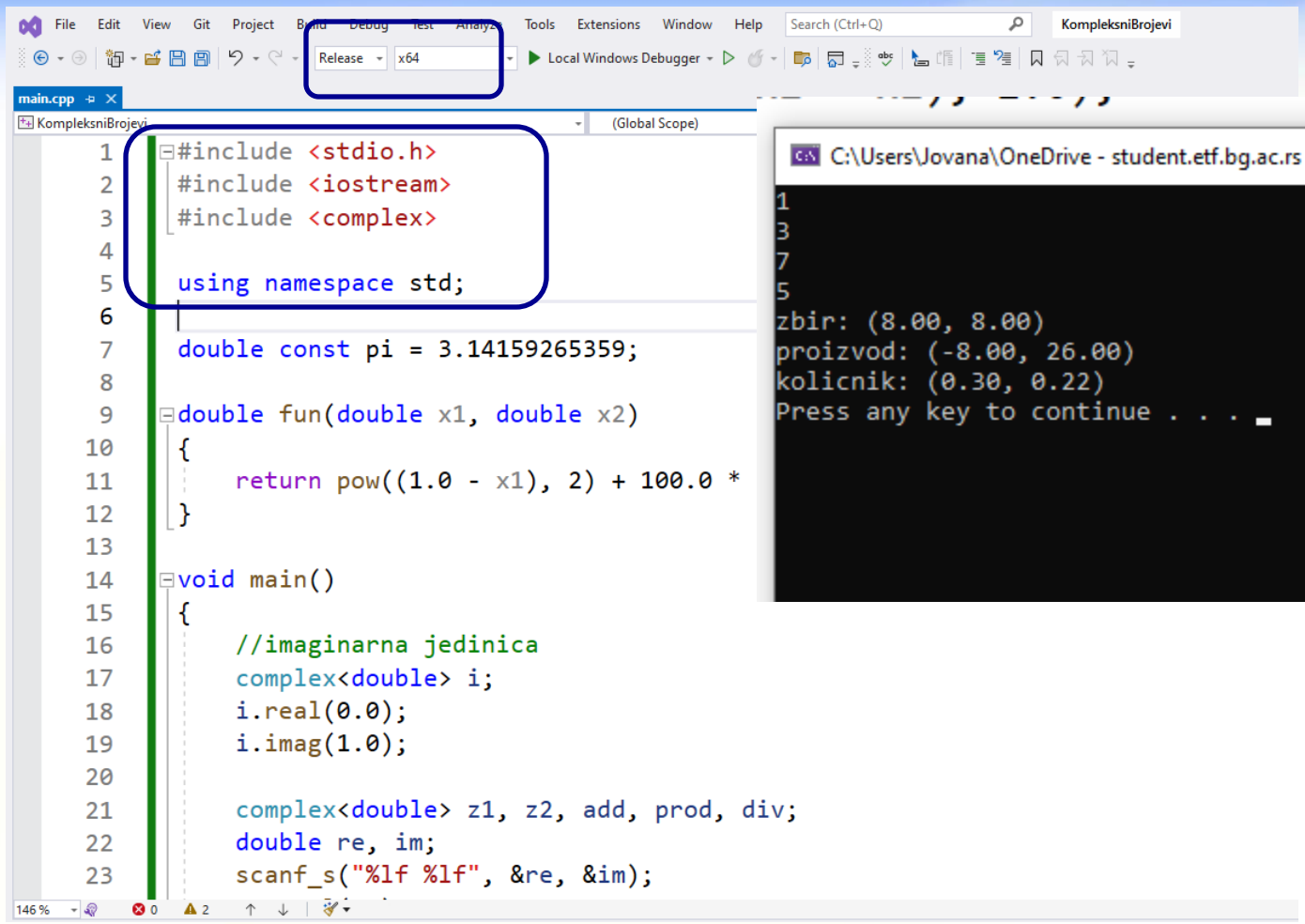
```
Optimization terminated successfully.
         Current function value: 0.000000
         Iterations: 79
         Function evaluations: 146
xmin = ( 1.0000043858986165 , 1.0000106409916478 )
```

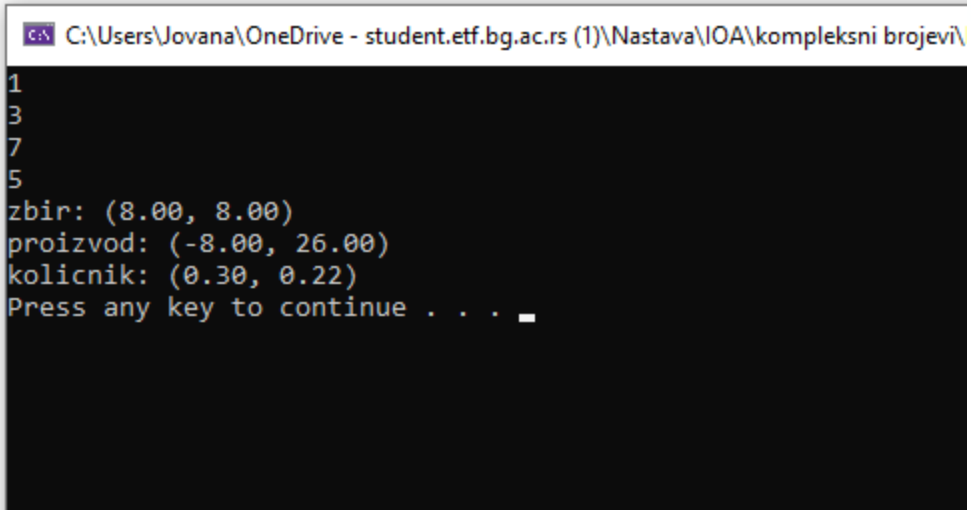
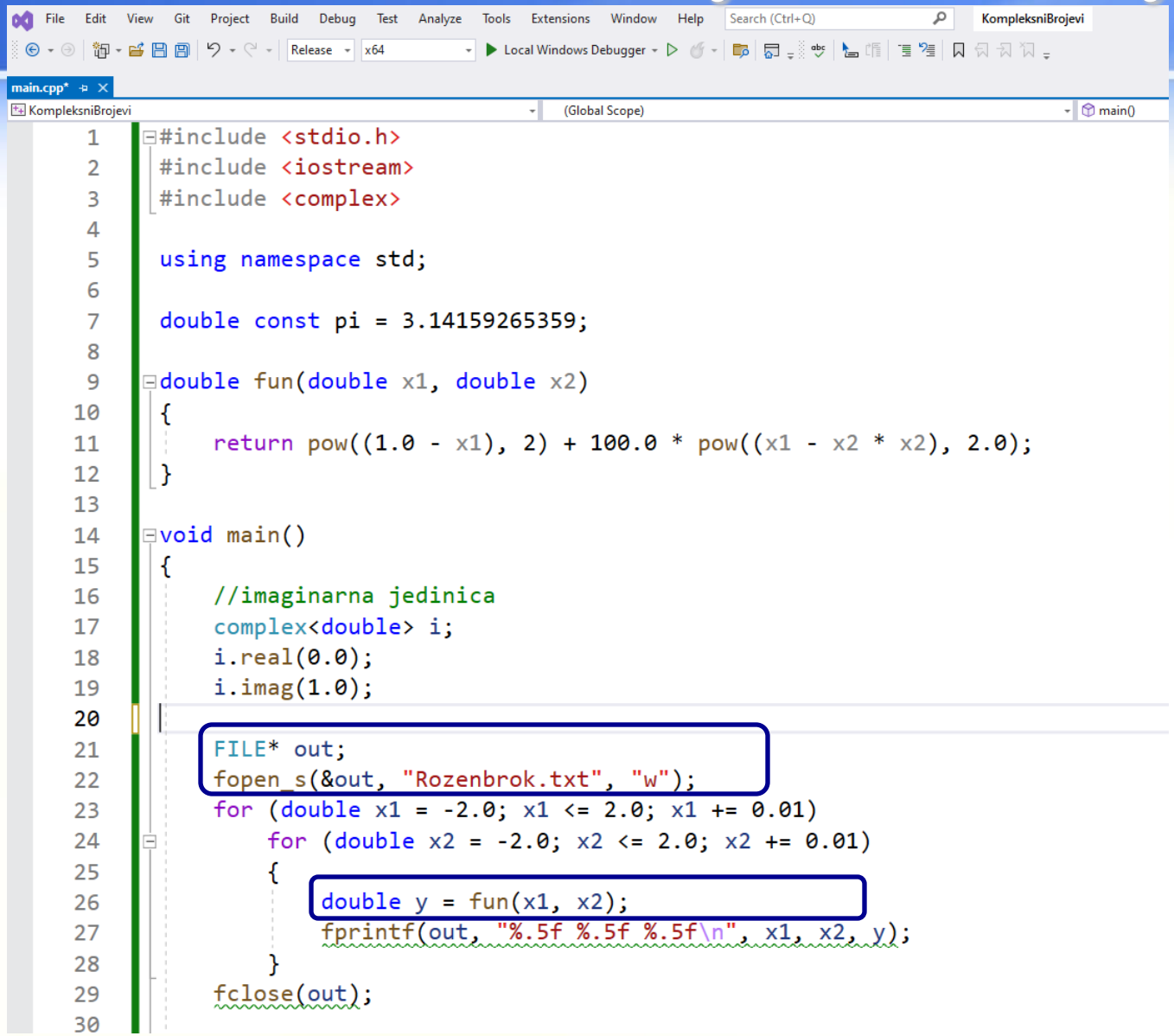


Visual Studio 2022
Отварање пројекта



Рад са комплексним бројевима





Уписивање у датотеку