Programsko inženjerstvo temeljeno na pretraživanju

(eng. Search Based Software Engineering - SBSE)

Student: Goran Mauša

Mentori: Tihana Galinac Grbac

Bojana Dalbelo Bašić

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Uvod

- Programsko inženjerstvo temeljeno na pretraživanju
  - skupina pretraživačkih optimizacijskih algoritama upotrijebljenih u programskom inženjerstvu
  - Termin prvi put upotrijebili Harman i Jones 2001. godine
  - SBSE tehnike su korištene u gotovo svim fazama životnog ciklusa programskog proizvoda, a ponajviše u testiranju [3]

- Primjenjivo gdjegod postoji:
  - veliko područje pretrage (mogućih rješenja)
  - veći broj jednako dobrih ili neusporedivih rješenja
  - više-ciljna problematika

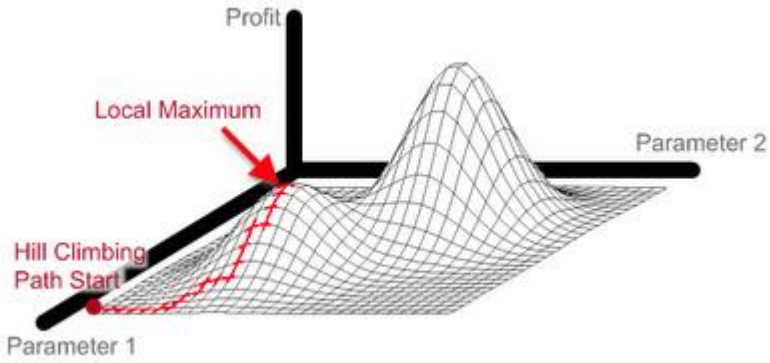
Motivacija za primjenu

- Podaci u programskom inženjerstvu često su nepotpuni, neprecizni i raspršeni
- Nedostatak opće analitičke teorije opisa promatrane pojave
- Metode inteligentne obrade podataka sve češće

- Neke područja primjene u najnovijim istraživanjima:
  - Automatsko ispravljanje programskog koda [4-9]
  - Optimizacija testnih podataka [10-14]
  - Problem sljedeće generacije (eng. next-release problem) [3, 15, 16]
  - Generiranje mutanata višeg reda [17, 18]
  - Klastering modula programa [19]
  - Alokacija ljudstva i zadataka [20]

Preduvjeti

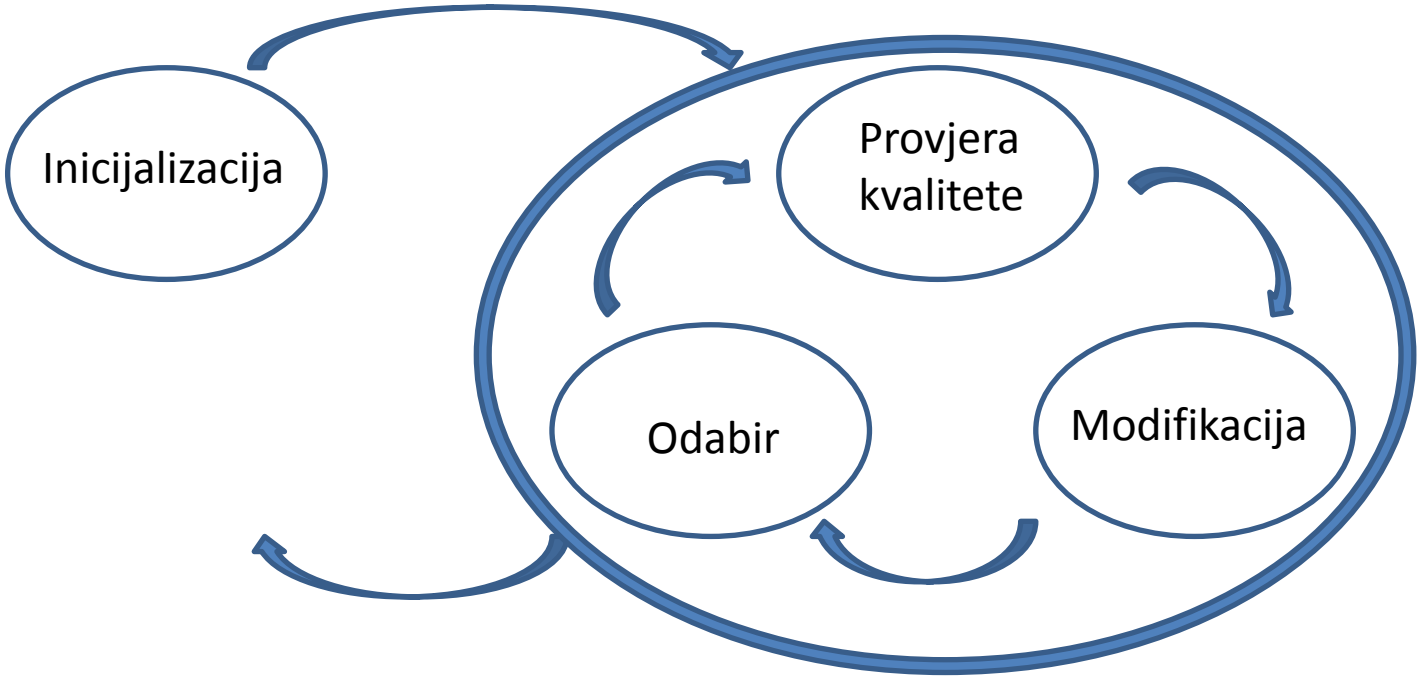
- Područje pretraživanja
  - promjenjivi parametri za kandidate rješenja



- Funkcija dobrote
  - mjera za usporedbu kandidata
  - vodi pretragu prema čim boljem rješenju
  - definirana samom problematikom

Općeniti koraci algoritama

1. **Inicijalizacija** – obično nasumično odabrani kandidati rješenja
2. **Provjera kvalitete** – kandidata rješenja izračunom funkcije dobrote
3. **Modifikacija** – kandidata rješenja nasumičnim manjim promjenama
4. **Odabir** – kandidata rješenja prema funkciji prikladnosti i sukladno odabranom algoritmu

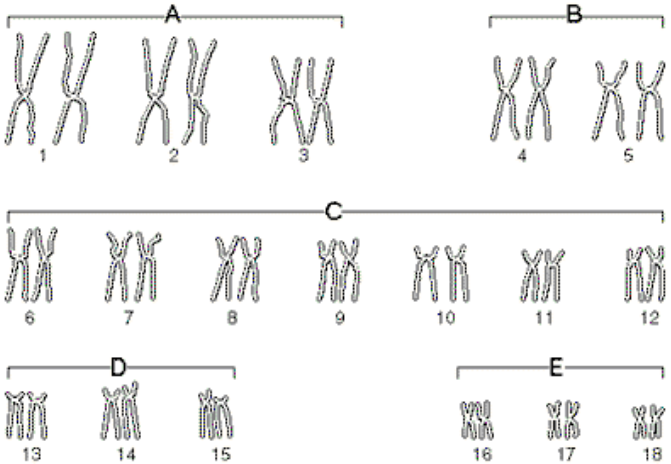


Podjela

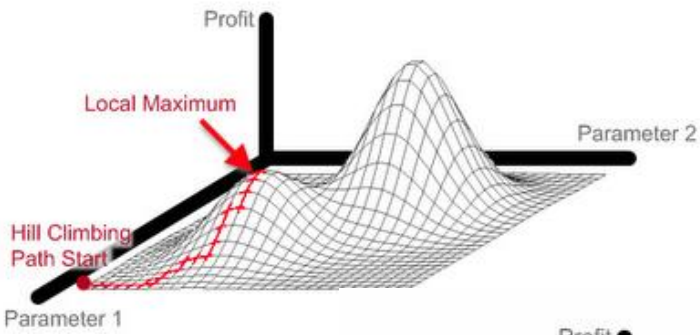
- **Lokalni**
  - potraga za lokalnim optimumom, gdje mogu postati zarobljeni
- **Globalni**
  - potraga za globalnim optimumom, ali uz veće zahtjeve računanja

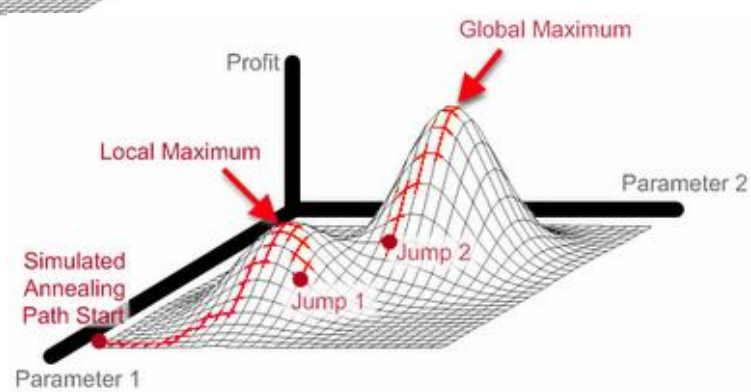
- **Bazirani na jednom stanju**
  - samo jedan kandidat rješenja istovremeno
- **Bazirani na populaciji**
  - cijela populacija kandidata rješenja istovremeno
  - evolucija kao inspiracija => evolucijski algoritmi
  - korak **Odabira** sadrži mutaciju i rekombinaciju najprilagođenijih roditelja

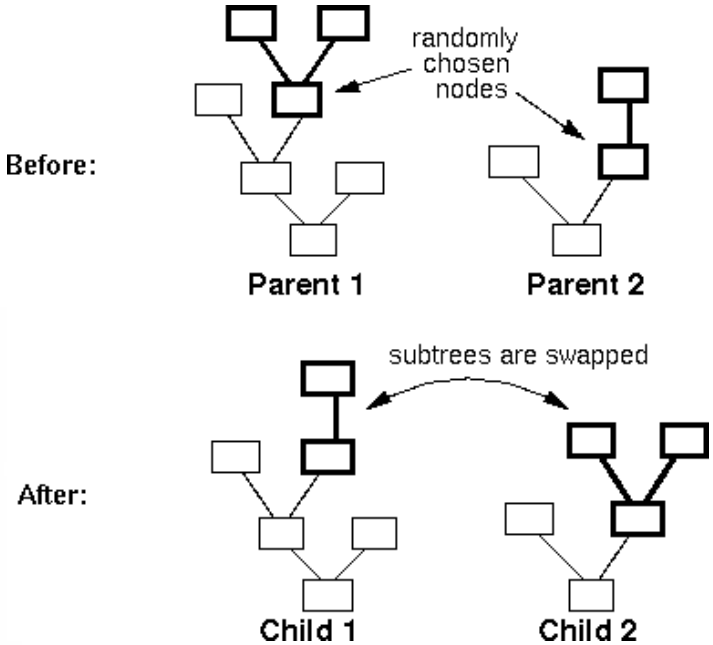
Algoritmi



- Najčešći su [1]:
  - Genetski algoritmi
  - Genetsko programiranje
  - Penjanje uz brijeg (eng. hill climbing)
  - Simulirano žarenje (eng. simulated annealing)







Opširnije o primjeni programskog inženjerstva temeljenog na pretraživanju

Automatsko ispravljanje programskog koda

- Težak posao koji oduzima mnogo vremena i sredstava (90% troškova nakon isporuke proizvoda)
- Ideja:
  1. Locirati regiju koja uzrokuje grešku
  2. Modifikacije duž puta gdje leži pogreška kako bi ju uklonili
  3. Zadržati funkcionalnost

- Forrest et al. [5] uspješno ispravili 11 C programa “sa police” (eng. off-the-shelf) kombinacijom genetskog programiranja i metode analize programa
- Schulte et al. [7] riješili problem na razini asembler koda koristeći genetsko računanje gotovo jednako uspješno kao i na razini izvornog koda (eng. source code)

Optimizacija testnih podataka

- Primjena evolucijskog testiranja – testiranje temeljeno na pretraživanju (eng. Search Based Testing)
- 3 aspekta traženja optimalnog testnog seta:
  1. Generacija podataka iz početka
  2. Regeneracija podataka uz već postojeće podatke
  3. Minimizacija podataka izbacivanjem reduntantnih

- McMinn i Harman [11] koristili evolucijsko testiranje kao globalni, penjanje uz brijeg kao lokalni i hibridni algoritam (eng. memetic) za generiranje podataka za strukturno testiranje
- Yoo i Harman [13] koristili penjanje uz brijeg sa već postojećim podacima i povećali efikasnost za 2 reda veličine uz jednaku strukturnu rasprostranjenost
- Yoo i Harman [14] koristili hibridni algoritam - gramzivi (eng. greedy) + genetski za više-ciljnu (eng. multi-objective) minimizaciju testnih podataka, naročito uspješno za veće proizvode

Problem sljedeće generacije

- Multi-objective problem odabira zahtjeva koje će zadovoljiti čim veći broj ljudi sljedeća generacija programskog proizvoda
- SBSE nudi više skoro optimalnih rješenja za donositelja odluke

- Durillo et al. [3] žele zadovoljiti čim veći broj kupaca uz minimalne troškove algoritmima:
  - NSGA-II – najveći broj rješenja
  - MOCell – najveći raspon rješenja
  - PAES – najlošiji rezultati
- Zhang et al. [15] koristili NSGA-II i dvo-arhivni (eng. **two-archive**) **algoritam** kako bi zadovoljili čim veći broj suprotstavljenih dioničara
- Finkelstein et al. [16] postigli jednake rezultate kao [15] sa nasumičnim podacima i bolje rezultate korištenjem NSGA-II sa realnim Motorola podacima

Generiranje mutanata višeg reda

- 90% pogrešaka koje prežive testiranje su kompleksne
- Mutanti nastali kombinacijom jednostavnih grešaka mogu otkriti takve suptilne, ali opasne greške

- Jia i Harman [17] traže mutante višeg reda algoritmima:
  - greedy algoritam – nađe mutante najvišeg reda
  - **genetski algoritam** – nađe najviše mutanata
  - penjanje uz brijeg – nađe mutante najveće dobrote
- Langdon et al. [18] uspješno pronalaze jednostavne greške koje se u kombinaciji međusobno maskiraju koristeći spoj Monte Carlo sempliranja, NSGA-II i genetskog programiranja

Zaključak

- Mogućnost šire upotrebe unutar programskog inženjerstva
  - relativno mlada grana
  - ne koriste sva područja programskog inženjerstva ove metode podjednako
  - postoje problemi za koje ove metode nisu razmatrane

- Velik broj optimizacijskih algoritama
  - ne postoji idealni algoritam
  - nisu istraženi svi algoritmi
  - mogućnost daljnjeg unapređenja samih algoritama
  - hibridni se pokazuju najboljima

Literatura

- [1] Mark Harman and Afshin Mansouri. Search based software engineering: Introduction to the special issue of the ieee transactions on software engineering. 36(6):737–741, 2010.
- [2] Sean Luke. Essentials of Metaheuristics. Lulu, 2009. Available for free at http://cs.gmu.edu/sean/book/metaheuristics/.
- [3] Juan J. Durillo, Yuanyuan Zhang, Enrique Alba, Mark Harman, and Antonio J. Nebro. A study of the bi-objective next release problem. Empirical Software Engineering, 16(1):29–60, February 2011.
- [4] Westley Weimer, ThanhVu Nguyen, Claire Le Goues, and Stephanie Forrest. Automatically finding patches using genetic programming. In Proceedings of the 31st International Conference on Software Engineering, ICSE ’09, pages 364–374,Washington, DC, USA, 2009. IEEE Computer Society.
- [5] Ethan Fast, Claire Le Goues, Stephanie Forrest, and Westley Weimer. Designing better fitness functions for automated program repair. In Proceedings of the 12th annual conference on Genetic and evolutionary computation, GECCO ’10, pages 965–972, New York, NY, USA, 2010. ACM.
- [6] Stephanie Forrest, ThanhVu Nguyen, Westley Weimer, and Claire Le Goues. A genetic programming approach to automated software repair. In Proceedings of the 11th Annual conference on Genetic and evolutionary computation, GECCO ’09, pages 947–954, New York, NY, USA, 2009. ACM.
- [7] Eric Schulte, Stephanie Forrest, andWestleyWeimer. Automated program repair through the evolution of assembly code. In Proceedings of the IEEE/ACM international conference on Automated software engineering, ASE ’10, pages 313–316, New York, NY, USA, 2010. ACM.

Literatura

- [8] ThanhVu Nguyen, Westley Weimer, Claire Le Goues, and Stephanie Forrest. Using execution paths to evolve software patches. In Proceedings of the IEEE International Conference on Software Testing, Verification, and ValidationWorkshops, ICSTW'09, pages 152–153,Washington, DC, USA, 2009. IEEE Computer Society.
- [9] Westley Weimer, Stephanie Forrest, Claire Le Goues, and ThanhVu Nguyen. Automatic program repair with evolutionary computation.
- Commun. ACM, 53:109–116, May 2010.
- [10] Cu D. Nguyen, Anna Perini, Paolo Tonella, Simon Miles, Mark Harman, and Michael Luck. Evolutionary testing of autonomous software agents. In Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1, AAMAS '09, pages 521–528, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems.
- [11] Mark Harman and Phil McMinn. A theoretical and empirical study of search-based testing: Local, global, and hybrid search. IEEE Trans. Softw. Eng., 36:226–247, March 2010.
- [12] Phil McMinn, Mark Harman, Kiran Lakhotia, Youssef Hassoun, and Joachim Wegener. Input domain reduction through irrelevant variable removal and its effect on local, global and hybrid search-based structural test data generation. IEEE Transactions on Software Engineering, 2011.
- [13] Shin Yoo and Mark Harman. Test data regeneration: Generating new test data from existing test data. Journal of Software Testing, Verification and Reliability, To appear.

Literatura

- [14] Shin Yoo and Mark Harman. Using hybrid algorithm for pareto efficient multi-objective test suite minimisation. J. Syst. Softw., 83:689–701, April 2010.
- [15] Yuanyuan Zhang, Mark Harman, Anthony Finkelstein, and S. AfshinMansouri. Comparing the performance of metaheuristics for the analysis of multi-stakeholder tradeoffs in requirements optimisation. Information and Software Technology, 53(7):761–773, July 2011.
- [16] Anthony Finkelstein, Mark Harman, S. Afshin Mansouri, Jian Ren, and Yuanyuan Zhang. Asearch based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making. Requir. Eng., 14:231–245, October 2009.
- [17] Yue Jia and Mark Harman. Higher order mutation testing. Inf. Softw. Technol., 51:1379–1393, October 2009.
- [18] William B. Langdon, Mark Harman, and Yue Jia. Efficient multi-objective higher order mutation testing with genetic programming. J. Syst. Softw., 83:2416–2430, December 2010.
- [19] Kata Praditwong,MarkHarman, and Xin Yao. Softwaremodule clustering as a multi-objective search problem. IEEE Trans. Softw. Eng., 37:264–282, March 2011.
- [20] Massimiliano Di Penta, Mark Harman, and Giuliano Antoniol. The use of search-based optimization techniques to schedule and staff software projects: an approach and an empirical study. Softw. Pract. Exper., 41:495–519, April 2011.