Večkriterijsko optimiranje z genetskimi algoritmi in diferencialno evolucijo

Delovno poročilo IJS-DP 9065

Tea Robič, Bogdan Filipič
Odsek za inteligentne sisteme
Institut "Jožef Stefan"
Jamova 39, SI-1000 Ljubljana
tea.robic@ijs.si, bogdan.filipic@ijs.si

April 2005

Povzetek

V poročilu predstavljamo osnovne pojme večkriterijskega optimiranja in dva pristopa k njegovemu reševanju. Na kratko opišemo nekaj klasičnih metod, ki so bile v preteklosti edini način reševanja večkriterijskih optimizacijskih problemov. Sledi pregled evolucijskih algoritmov za večkriterijsko optimiranje, med katerimi podrobneje predstavimo dva genetska algoritma (NSGA-II in SPEA2) ter diferencialno evolucijo (DEMO). Zaključimo z omembo ostalih metod za večkriterijsko optimiranje.

1. Uvod

V praksi se pogosto srečujemo z zahtevo po sočasnem optimiranju po različnih kriterijih. Večkrat so si kriteriji tudi konfliktni, kar pomeni, da izboljšanje rešitve po enem kriteriju povzroči njeno poslabšanje po drugih kriterijih. Takrat nimamo opravka samo z eno optimalno rešitvijo, temveč z množico optimalnih rešitev, imenovano *Pareto optimalna fronta*. Če ne poznamo dodatne informacije, ki bi podala pomembnosti kriterijev, si želimo, da bi poznali vse rešitve s Pareto optimalne fronte in bi se šele nato odločili za eno izmed njih.

Zaradi pomanjkanja metod, primernih za večkriterijsko optimiranje, so v preteklosti večkriterijske optimizacijske naloge reševali s klasičnimi metodami tako, da so nalogo pretvorili v enokriterijsko optimizacijsko nalogo in za njeno reševanje uporabljali standardne pristope. Šele z evolucijskimi algoritmi so se večkriterijskih optimizacijskih nalog lotili tako, da so v enem zagonu poiskali več rešitev, ki so aproksimirale Pareto optimalno fronto. Temu zgledu so kmalu sledile druge metahevristične metode.

Poročilo pričnemo z definicijami osnovnih pojmov, ki nastopajo v večkriterijskem optimiranju, kot sta dominantnost in Pareto optimalnost. Predstavimo dva pristopa k reševanju takšnih problemov: prednostnega in idealnega. Nato opišemo najbolj znani klasični metodi za reševanje večkriterijskih optimizacijskih problemov, metodo utežene vsote in metodo ϵ-omejitev, ter najpogosteje uporabljena genetska algoritma za večkriterijsko optimiranje, NSGA-II in SPEA2. Nadaljujemo z novim evolucijskim algoritmom, imenovanim DEMO, ki temelji na diferencialni evoluciji. Na koncu omenimo še nekatere druge metode za večkriterijsko optimiranje.

2. Večkriterijsko optimiranje

2.1. Definicije

Problem večkriterijskega optimiranja je definiran kot problem iskanja dopustnega vektorja spremenljivk, ki optimira vektorsko funkcijo, katere elementi so kriterijske funkcije. Drugače povedano, iščemo vektor $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ iz prostora spremenljivk, ki zadošča m neenakostim

$$g_i(\mathbf{x}) \geq 0; \quad i = 1, 2, \ldots, m$$

in optimira vektorsko funkcijo

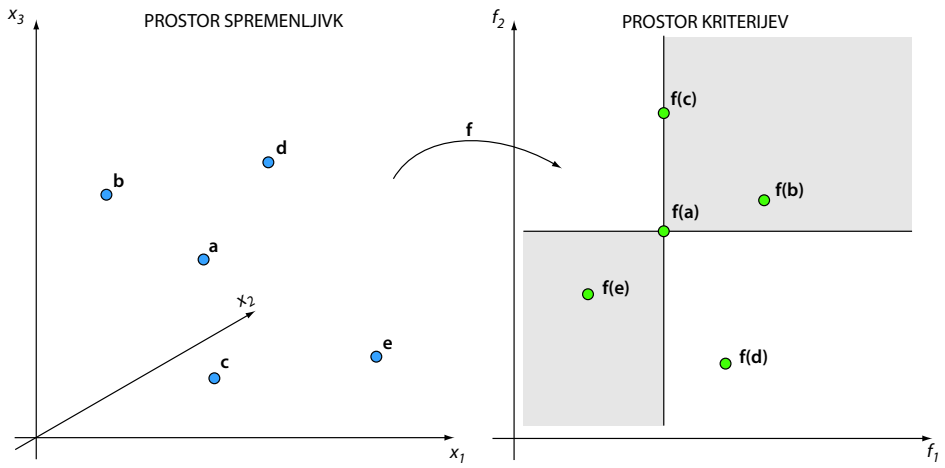
$$\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})).$$

Ker lahko vsak problem maksimizacije kriterijev enostavno prevedemo na problem minimizacije, bomo v nadaljevanju vedno predpostavljali, da želimo vektorsko funkcijo $\mathbf{f}(\mathbf{x})$ minimizirati po vseh kriterijih.

Pri enokriterijskem optimiranju je prostor kriterijev množica realnih števil $\mathbb{R}$, ki je z relacijo $\leq$ popolno urejena. Tako za poljubni rešitvi $\mathbf{x}$ in $\mathbf{y}$ enokriterijskega optimizacijskega problema velja natanko ena od trditev "$\mathbf{x}$ je boljša od $\mathbf{y}$", "$\mathbf{x}$ je slabša od $\mathbf{y}$" ali "$\mathbf{x}$ in $\mathbf{y}$ sta enakovredni". Pri večkriterijskem optimiranju pa je prostor kriterijev večdimenzionalen ($\mathbb{R}^k$). Zato za relacijo $\leq$ ne velja več popolna urejenost, temveč le delna urejenost. Dve rešitvi sta tako pogosto neprimerljivi. Večina večkriterijskih optimizacijskih algoritmov si pri primerjavi rešitev pomaga s konceptom dominantnosti.

Rešitev večkriterijskega optimizacijskega problema $\mathbf{x}$ *dominira* rešitev $\mathbf{y}$ ($\mathbf{x} \preceq \mathbf{y}$), če sta izpolnjeni naslednji zahtevi:

1. Rešitev $\mathbf{x}$ ni slabša od rešitve $\mathbf{y}$ po vseh kriterijih ($\mathbf{f}_i(\mathbf{x}) \leq \mathbf{f}_i(\mathbf{y})$ za vse $i = 1, \ldots, k$).
2. Rešitev $\mathbf{x}$ je boljša od rešitve $\mathbf{y}$ po vsaj enem kriteriju ($\mathbf{f}_j(\mathbf{x}) < \mathbf{f}_j(\mathbf{y})$ za vsaj en $j \in \{1, \ldots, k\}$).



Slika 1: Prikaz koncepta dominantnosti na primeru večkriterijske funkcije **f**

Na sliki 1 je prikazan primer večkriterijske funkcije **f**, ki tridimenzionalni prostor spremenljivk preslika v dvodimenzionalni prostor kriterijev. Ker predpostavljamo, da želimo funkcijo **f** minimizirati po vseh kriterijih, za rešitev **a** velja, da dominira rešitvi **b** in **c**, z rešitvijo **d** je neprimerljiva, medtem ko jo rešitev **e** dominira.

Množica nedominiranih rešitev (imenovana tudi *nedominirana fronta*) v množici rešitev P je množica vseh tistih rešitev, ki jih ne dominira nobena rešitev iz množice P. Množica nedominiranih rešitev celotnega prostora dopustnih rešitev se imenuje *Pareto optimalna fronta*, njeni elementi pa *Pareto optimalne rešitve* (po Vilfredu Paretu, italijanskem ekonomistu in sociologu ter pionirju na področju večkriterijskega optimiranja). V primeru s slike 1 sestavljata množico nedominiranih rešitev rešitvi **d** in **e**.

2.2. Pristopi k večkriterijskem optimiranju

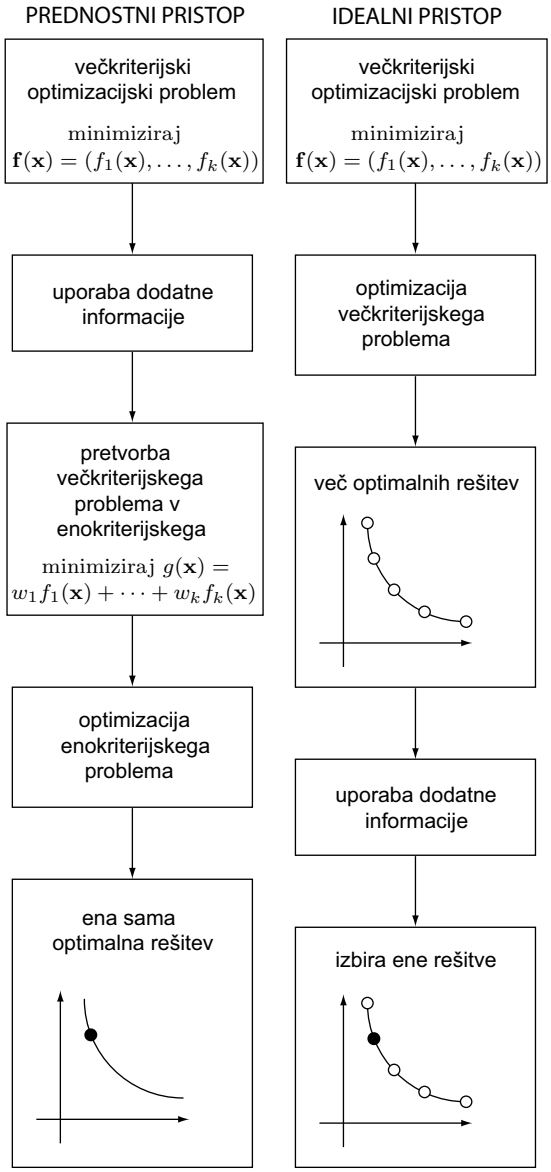
Če imamo pri večkriterijskem optimiranju opraviti s konfliktnimi kriteriji, obstaja več Pareto optimalnih rešitev. Brez dodatne informacije o pomembnosti posameznih kriterijev ne moremo povedati, katera od teh je boljša. Podobno kot pri enokriterijskem optimiranju, tudi pri večkriterijskem navadno želimo dobiti samo eno optimalno rešitev. To lahko dosežemo z dvema pristopoma, ki sta prikazana na sliki 2.

Rešitev, ki jo dobimo s *prednostnim pristopom*, je odvisna od funkcije, s katero smo večkriterijski problem pretvorili v enokriterijskega. Z uporabo drugačne funkcije bi lahko dobili drugačno rešitev. Poleg tega prednostni pristop zahteva dodatno informacijo o pomembnosti kriterijev, ki pa navadno ni poznana vnaprej.

Pri *idealnem pristopu* najprej poiščemo množico optimalnih rešitev, nato pa iz nje izberemo rešitev, ki nam najbolj ustreza. Tudi pri tem pristopu potrebujemo dodatno informacijo o problemu, vendar jo potrebujemo le za izbiro ene rešitve iz množice optimalnih in ne za gradnjo novih rešitev. Zato je idealni pristop bolj pregleden in manj subjektiven od prednostnega pristopa. Seveda, če poznamo informacijo o kriterijih, ki nam omogoča ciljno usmerjeno uporabiti prednostni pristop, ni nobenega razloga, da bi namesto tega uporabljali idealni pristop.

Pri reševanju z idealnim pristopom želimo, da večkriterijska optimizacijska metoda najde čim več Pareto optimalnih rešitev. Ker bomo izmed teh rešitev (s pomočjo informacije o pomembnosti kriterijev) izbirali najboljšo, si želimo, da so dobljene rešitve kar se da enakomerno razporejene po prostoru kriterijev. Nalogo večkriterijskega optimiranja tako lahko pretvorimo v nalogo iskanja množice nedominiranih rešitev, za katere velja (slika 3):

– da so čim bliže Pareto optimalni fronti in



Slika 2: Prednostni in idealni pristop k večkriterijskem optimiranju

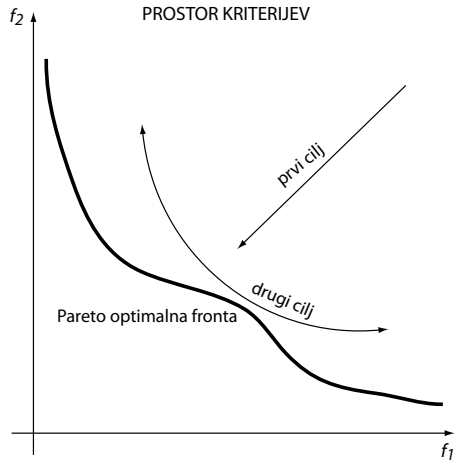
– da so enakomerno razporejene vzdolž Pareto optimalne fronte.

Ta dva cilja pa si pogosto nasprotujeta.

2.3. Metode za večkriterijsko optimiranje

Metode za večkriterijsko optimiranje lahko razdelimo v tri skupine: *klasične*, *evolucijske* in *ostale* metode. Klasične metode uporabljajo prednostni pristop in so prve metode, s katerimi so v preteklosti reševali večkriterijske optimizacijske probleme. V nadaljevanju si bomo ogledali dve najbolj priljubljeni klasični metodi: metodo utežene vsote in metodo ϵ omejitev.

Večkriterijskega optimiranja z idealnim pristopom so se najprej lotili z evolucijskimi algoritmi. Pri tem uporabljajo koncept Pareto dominantnosti, ki omogoča dosegati oba cilja večkriterijskega optimiranja. V 4. razdelku si bomo ogledali značilnosti evolucijskih algoritmov za večkriterijsko optimiranje. Podrobneje bomo opisali tri algoritme: dva genetska algoritma in diferencialno evolucijo.



Slika 3: Cilja večkriterijskega optimiranja

Med ostale metode štejemo metahevristike, kot so simulirano ohlajanje, iskanje s tabuji in podobne metode. Te so začeli uporabljati za večkriterijsko optimiranje sočasno z evolucijskimi algoritmi, a so z njimi sprva kriterije kombinirali kot v klasičnih metodah. Šele ko so v evolucijskih algoritmih začeli uporabljati idealni pristop, so tudi ostale metode sledile temu zgledu. V 5. razdelku bomo omenili nekaj ostalih metod za večkriterijsko optimiranje.

3. Klasične metode

Kot že rečeno, so klasične metode prve metode, s katerimi so v preteklosti reševali večkriterijske optimizacijske probleme. Sem štejemo metodo uteženе vsote in metodo ϵ-omejitev ter tudi druge, manj znane metode, kot so: metoda uteženih metrik, Bensonova metoda, metoda funkcije koristi, metode ciljnega programiranja in druge. Več o klasičnih metodah za večkriterijsko optimiranje najdemo v [7].

3.1. Metoda uteženе vsote

Metoda uteženе vsote (angl. weighted sum method) je najbolj razširjena klasična metoda za večkriterijsko optimiranje. Večkriterijski problem pretvorimo v enokriterijskega tako, da izberemo uteži w_i, ki določajo pomembnost kriterijev. Na ta način iz naloge

$$\text{optimiraj} \quad \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))$$

dobimo nalogo

$$\text{optimiraj} \quad g(\mathbf{x}) = w_1 f_1(\mathbf{x}) + \cdots + w_k f_k(\mathbf{x}),$$

ki jo lahko rešujemo z ustrezno metodo za enokriterijsko optimiranje. Ker se optimum te naloge ne spremeni, če uteži pomnožimo s konstanto, navadno predpostavljamo, da za uteži velja

$$w_i \in [0, 1] \quad \text{in} \quad \sum_{i=1}^{k} w_i = 1.$$

To je enostavna metoda, ki pa ima več slabosti. Prva očitna težava metode je, da zahteva vektor uteži. Če ne poznamo prednosti kriterijev, je določitev takšnega vektorja zahtevna naloga. Poleg tega moramo, da bi dobili več Pareto optimalnih rešitev, metodo večkrat uporabiti – vsakič z drugačno nastavitvijo uteži. Vendar je pri tem treba paziti, saj enakomerno porazdeljeni vektorji uteži ne dajo nujno enakomerno porazdeljenih točk na Pareto optimalni fronti.

Ker preslikava med vektorjem uteži in točkami na Pareto optimalni fronti navadno ni poznana, težko najdemo nastavitev uteži, ki bo dala želeno točko Pareto optimalne fronte. Poleg tega različni vektorji uteži ne dajo nujno različnih točk na Pareto optimalni fronti. Podobno tudi en vektor uteži lahko določa različne točke Pareto optimalne fronte. Če je večkriterijski optimizacijski problem konveksen, potem lahko vsako točko Pareto optimalne fronte izračunamo z metodo utežene vsote. Vendar je metoda omejena na konveksne Pareto optimalne fronte, za nekonveksne fronte pa odpove. Metoda je tudi občutljiva na razmerja med vrednostmi kriterijev, zato je dobro kriterije predhodno normirati.

Največja prednost te metode je njena enostavnost. Če rešujemo konveksen večkriterijski optimizacijski problem, pri katerem so prednosti kriterijev poznane, je metoda uteženih vsot prava izbira.

3.2. Metoda ϵ-omejitev

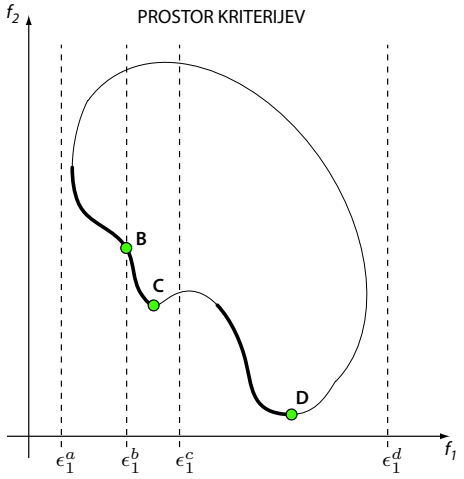
Če se želimo izogniti težavam, ki so značilne za metodo utežene vsote pri nekonveksnih Pareto optimalnih frontah, lahko uporabimo metodo ϵ-omejitev (angl. ϵ-constraint method). Haimes in sodelavci [12] so predlagali reševanje večkriterijskega optimizacijskega problema tako, da optimiramo samo en kriterij, ostale pa dodamo med omejitve. Preoblikovana optimizacijska naloga se tako glasi:

$$\begin{aligned} &\text{optimiraj} \quad f_\mu(\mathbf{x}) \\ &\text{pri pogojih} \quad f_i(\mathbf{x}) \leq \epsilon_i, \quad i = 1, \ldots, k \quad \text{in} \quad i \neq \mu. \end{aligned}$$

V tej formulaciji parameter ϵ_i označuje zgornjo mejo za vrednost f_i in ne pomeni nujno majhne vrednosti blizu nič.

Delovanje metode ϵ-omejitev je prikazano na sliki 4. V našem primeru optimiramo $f_2(\mathbf{x})$ pri pogoju $f_1(\mathbf{x}) \leq \epsilon_1$. Privzemimo najprej, da je $\epsilon_1 = \epsilon_1^b$. Prostor kriterijev je zdaj razdeljen na dve območji – območje z $f_1(\mathbf{x}) \leq \epsilon_1^b$ in območje z $f_1(\mathbf{x}) > \epsilon_1^b$. Dopustno je samo levo območje, zato iščemo minimum f_2 na tem območju. Optimum je dosežen v točki B. Podobno velja tudi za primer, ko je $\epsilon_1 = \epsilon_1^c$. Tokrat je optimalna rešitev točka C. Opazimo lahko, da oblika Pareto optimalne fronte ne vpliva na delovanje metode. Če izberemo $\epsilon_1 = \epsilon_1^a$, problem nima dopustnih rešitev. Če pa izberemo $\epsilon_1 = \epsilon_1^d$, je dopusten cel prostor in rešitev problema je točka D.

Podobno kot metoda utežene vsote tudi metoda ϵ-omejitev zahteva informacijo o kriterijih. Tokrat je ta podana v obliki vektorja omejitev. Če izberemo drugačen vektor omejitev ali če za optimiranje izberemo drug kriterij, dobimo drugačne rešitve. Ker omejujemo kriterije (in ne



Slika 4: Metoda ϵ-omejitev

spremenljivk), nam ta metoda omogoča lažje lokaliziranje rešitev v zaželenih območjih. Poleg tega nima težav z obliko Pareto optimalne fronte, saj deluje na enak način za konveksne in nekonveksne ter zvezne in diskretne fronte. Metoda tudi ne zahteva normiranja kriterijev, saj razlike v vrednosti kriterijev upoštevamo v vektorju omejitev.

4. Evolucijski algoritmi

V naravni evoluciji samo najboljši osebki prenašajo svoje gene v naslednjo generacijo. Slabe osebke (in s tem slabe gene) evolucija izloči. *Evolucijski algoritmi* (angl. evolutionary algorithms) posnemajo princip naravne evolucije, v zvezi z njimi pa se uporablja tudi za to področje značilna terminologija. Tako operirajo nad *osebki* (rešitvami v prostoru spremenljivk) in *populacijami* (množicami takšnih rešitev). Z uporabo *selekcije* in *kombiniranja* genetskih zapisov osebkov ustvarjajo iz generacije v generacijo čedalje boljše rešitve. Evolucijske algoritme uspešno uporabljajo za enokriterijsko optimiranje. Ker so populacijski, so primerni tudi za večkriterijsko optimiranje, kjer (po idealnem pristopu) želimo v enem zagonu algoritma dobiti več nedominiranih rešitev. Poleg tega so robustni – nimajo težav pri optimizacijskih nalogah z nekonveksno ali nezvezno Pareto optimalno fronto.

Čeprav se je uporaba evolucijskih algoritmov za večkriterijsko optimiranje razmahnila šele v zadnjih letih, prve zamisli o njej segajo že v šestdeseta leta prejšnjega stoletja [9]. Prvi pravi evolucijski algoritem za večkriterijsko optimiranje je Schafferjev VEGA (Vector Evaluated Genetic Algorithm) iz leta 1984 [27]. Schaffer se je naloge optimiranja po k kriterijih lotil tako, da je populacijo genetskega algoritma naključno razdelil na k podpopulacij. Vsaka od teh podpopulacij je bila ocenjena z enim od k kriterijev. Na ta način so bili vsi kriteriji uporabljeni za ocenjevanje rešitev genetskega algoritma. VEGA je dobro izpolnjeval prvi cilj večkriterijskega optimiranja (konvergenco k Pareto optimalni fronti), a ni bil sposoben v populaciji obdržati raznolikih rešitev (sčasoma so vse rešitve konvergirale le k eni Pareto optimalni rešitvi).

Schafferjevemu pionirskemu delu so čez nekaj let sledili drugi raziskovalci in nastali so številni evolucijski algoritmi za večkriterijsko optimiranje, med katerimi so najbolj znani naslednji:

- MOGA (Multiple Objective Genetic Algorithm) [10],
- NPGA (Niched-Pareto Genetic Algorithm) [15],
- NSGA (Non-Dominated Sorting Genetic Algorithm) [28] in njegov naslednik NSGA-II (Elitist Non-Dominated Sorting Genetic Algorithm) [8],
- SPEA (Strength Pareto Evolutionary Algorithm) [36] in SPEA2 [35],
- PAES (Pareto Archived Evolution Strategy) [19],
- PESA (Pareto Envelope-based Selection Algorithm) [5] in PESA-II [4],
- MOMGA (Multi-Objective Messy Genetic Algorithm) [33] in MOMGA-II [37].

Več informacij o teh (in drugih) evolucijskih algoritmih za večkriterijsko optimiranje najdemo v knjigah [7] in [3].

Izmed naštetih algoritmov sta najbolj popularna NSGA-II in SPEA2. Oba izhajata iz genetskih algoritmov, ki so tudi sicer izmed vseh evolucijskih algoritmov najpogosteje uporabljeni za večkriterijsko optimiranje. Algoritma si bomo podrobneje ogledali v nadaljevanju. Poleg genetskih algoritmov pa se v zadnjem času za večkriterijsko optimiranje uporablja tudi diferencialna evolucija. Zato bomo kot tretji primer evolucijskega algoritma za večkriterijsko optimiranje opisali nov algoritem DEMO (Differential Evolution for Multiobjective Optimization) [26], ki kot osnovo uporablja evolucijski algoritem, imenovan diferencialna evolucija.

4.1. Genetski algoritmi

Genetski algoritmi (angl. genetic algorithms) so zaradi široke uporabnosti, enostavnosti in uspešnosti pri iskanju globalnih optimumov najbolj razširjeni evolucijski algoritmi. Idejo genetskih algoritmov je prvi objavil Holland leta 1975 [14], razširil pa Goldberg leta 1989 [11].

Na sliki 5 je prikazan potek t.i. enostavnega genetskega algoritma (angl. simple genetic algorithm) [11]. Ta ponavlja osnovne korake (selekcija, križanje in mutacija) dokler ni izpolnjen ustavitveni kriterij. Rezultat algoritma je najboljši dobljeni osebek skozi celotno evolucijo.

Genetski algoritem

1. Naključno generiraj in ovrednoti začetno populacijo staršev $\mathcal{P}_0$.
2. Postavi $t = 0$.
3. Dokler ustavitveni kriterij ni izpolnjen, ponavljaj:
   3.1. Pripravi prazno populacijo potomcev $\mathcal{Q}_t$.
   3.2. Izberi nekaj staršev iz populacije $\mathcal{P}_t$.
   3.3. Izbrane starše križaj med seboj, da dobiš potomce.
   3.4. Na potomcih uporabi mutacijo in jih dodaj v populacijo potomcev $\mathcal{Q}_t$.
   3.5. Ovrednoti osebke iz populacije potomcev $\mathcal{Q}_t$.
   3.6. Postavi $t = t + 1$.
   3.7. Populacijo potomcev $\mathcal{Q}_{t-1}$ prepiši v populacijo staršev $\mathcal{P}_t$.

Slika 5: Enostavni genetski algoritem

4.1.1. NSGA-II

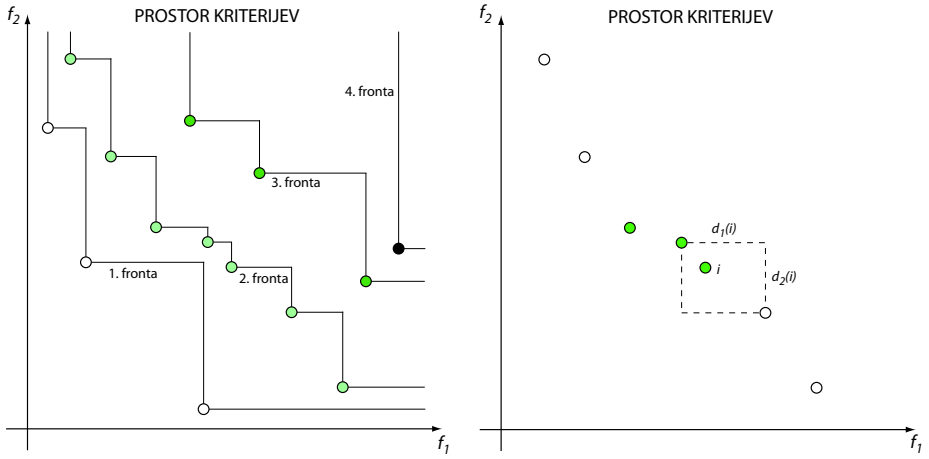
Algoritem NSGA-II (Elitist Non-Dominated Sorting Genetic Algorithm) Deba in sodelavcev iz leta 2000 [8] je zelo uspešen genetski algoritem za večkriterijsko optimiranje, ki se je v primerjalnih študijah odrezal najbolje od vseh (takrat obstoječih) evolucijskih algoritmov za večkriterijsko optimiranje. Skupaj s predhodnikom (NSGA [28]) je prvi uporabljal *nedominirano sortiranje* (angl. non-dominated sorting) in *metriko nakopičenosti* (angl. crowding distance metric).

Potek algoritma si oglejmo s pomočjo slike 6. NSGA-II deluje podobno kot enostavni genetski algoritem za enokriterijsko optimiranje (slika 5). Prilagoditev za večkriterijsko optimiranje je vidna le pri selekciji. Glavni korak algoritma (4. točka na sliki 6) sestavljata selekcija in rekombinacija (križanje in mutacija) osebkov. Najprej stari populaciji staršev in potomcev ($\mathcal{P}_{t-1}$

NSGA-II

1. Naključno generiraj in ovrednoti začetno populacijo staršev $\mathcal{P}_0$.
2. Pripravi prazno začetno populacijo potomcev $\mathcal{Q}_0$.
3. Postavi $t = 0$.
4. Dokler ustavitveni kriterij ni izpolnjen, ponavljaj:
   4.1. Postavi $t = t + 1$.
   4.2. Združi stari populaciji staršev in potomcev: $\mathcal{R}_{t-1} = \mathcal{P}_{t-1} \cup \mathcal{Q}_{t-1}$.
   4.3. Izvedi nedominirano sortiranje na $\mathcal{R}_{t-1}$ in določi fronte $\mathcal{F}_i$, $i = 1, 2, \ldots$
   4.4. Pripravi novo prazno populacijo staršev $\mathcal{P}_t$.
   4.5. V populacijo $\mathcal{P}_t$ daj prvih i front, ki še gredo cele v njo.
   4.6. Fronto $\mathcal{F}_{i+1}$, ki ne gre več cela v populacijo $\mathcal{P}_t$ sortiraj z uporabo metrike nakopičenosti.
   4.7. Populacijo $\mathcal{P}_t$ dopolni z osebki iz $\mathcal{F}_{i+1}$, ki so najmanj nakopičeni.
   4.8. Populacijo potomcev $\mathcal{Q}_t$ generiraj iz populacije staršev $\mathcal{P}_t$ z uporabo turnirske selekcije, križanja in mutacije.
   4.9. Ovrednoti osebke iz populacije potomcev $\mathcal{Q}_t$.

Slika 6: Algoritem NSGA-II



Slika 7: Nedominirano sortiranje po frontah in računanje metrike nakopičenosti

in $\mathcal{Q}_{t-1}$) združimo v eno populacijo ($\mathcal{R}_{t-1}$), v kateri osebke sortiramo po frontah s t.i. nedominiranim sortiranjem. V novo populacijo staršev ($\mathcal{P}_t$) gredo po vrsti najboljše fronte. Osebke s prve fronte, ki zaradi omejene velikosti populacije ne more cela vanjo, nadalje sortiramo z metriko nakopičenosti. V populacijo dodamo tiste osebke s fronte, ki so najmanj nakopičeni. Na ta način dobimo populacijo staršev $\mathcal{P}_t$, iz katere generiramo potomce z uporabo turnirske selekcije, križanja in mutacije. Turnirska selekcija izmed dveh naključnih staršev izbere tistega, ki je boljši glede na nedominirano sortiranje. Če sta starša izenačena (ležita na isti fronti), v turnirju zmaga tisti, ki ima boljšo vrednost metrike nakopičenosti. Turnirsko selekcijo večkrat ponovimo in tako izbrane starše križamo in mutiramo, da dobimo novo populacijo potomcev $\mathcal{Q}_t$. Vse osebke iz populacije $\mathcal{Q}_t$ še ovrednotimo.

Na sliki 7 levo je prikazano nedominirano sortiranje po frontah za primer populacije s sedmimi osebki (združena populacija staršev in potomcev, ki je narisana na sliki, zato vsebuje štirinajst osebkov). Prva fronta gre v celoti v novo populacijo, medtem ko moramo iz druge fronte izbrati najboljše štiri osebke glede na metriko nakopičenosti (slika 7 desno). Le-ta vedno najbolje oceni skrajne osebke v fronti (da imajo rešitve čim večji razpon). Ostale (vmesne) osebke pa oceni glede na njihovo razdaljo do najbližjih sosedov. Če optimiramo k kriterijev, potem za vsak kriterij $j = 1, \ldots, k$ osebke najprej sortiramo po naraščajočih vrednostih f_j. Nato za vsak vmesni osebek i izračunamo razdaljo $d_j(i)$ kot razdaljo med osebku i najbližjima osebkoma p in q (za katera je $f_j(p) \leq f_j(i) \leq f_j(q)$) kot

$$d_j(i) = \frac{f_j(q) - f_j(p)}{f_j^{max} - f_j^{min}},$$

kjer sta f_j^{max} in f_j^{min} maksimalna in minimalna vrednost j-tega kriterija. Skrajnima dvema osebkoma (glede na kriterij j) določimo najvišjo možno razdaljo. Metrika nakopičenosti za osebek i je vsota teh razdalj po vseh kriterijih:

$$c(i) = \sum_{j=1}^{k} d_j(i).$$

Na sliki 7 desno je metrika nakopičenosti za osebek i enaka polovici obsega pravokotnika, ki ga določata sosednja osebka. V primeru s slike gredo v novo populacijo vsi osebki, označeni z belo barvo (trije osebki s prve fronte in štirje osebki z druge fronte, ki imajo največjo vrednost metrike nakopičenosti).

4.1.2. SPEA2

Podobno kot NSGA-II je tudi SPEA2 (Strength Pareto Evolutionary Algorithm) Zitzlerja in sodelavcev iz leta 2001 [35] popularen genetski algoritem za večkriterijsko optimiranje, ki je primerljiv z NSGA-II pri doseganju prvega cilja večkriterijskega optimiranja (konvergenca k Pareto optimalni fronti) in boljši pri doseganju drugega cilja (enakomerna razporejenost rešitev).

SPEA2 namesto populacije staršev in potomcev uporablja osnovno populacijo in arhiv konstantne velikosti, v katerega shranjuje najboljše dobljene rešitve. Vrednotenje osebkov ne poteka po frontah, tako kot pri NSGA-II, temveč z merjenjem *moči* (angl. strength), *grobe uspešnosti* (angl. raw fitness) in *gostote* (angl. density) osebkov. Moč osebka je enaka številu osebkov iz populacije in arhiva, ki jih izbrani osebek dominira (slika 8 levo). Grobo uspešnost osebka pa izračunamo kot vsoto moči vseh osebkov iz populacije in arhiva, ki ta osebek dominirajo (slika 8 desno). Ker z evolucijo številni osebki postanejo nedominirani (in imajo zato grobo uspešnost enako nič), potrebujemo dodatno informacijo o razporejenosti rešitev. Zato za vse osebke i z enako grobo uspešnostjo $R(i)$ izračunamo še gostoto $D(i)$ kot:

$$D(i) = \frac{1}{\sigma_i^k + 2},$$

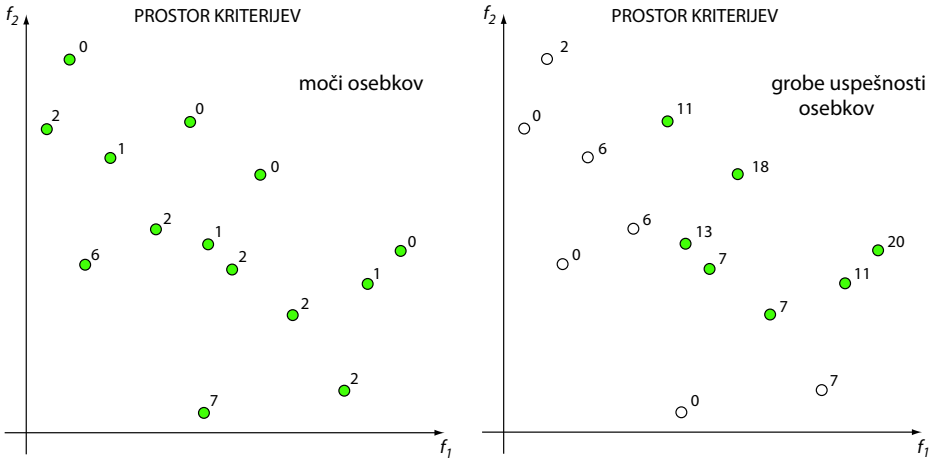
kjer je σ_i^k razdalja do k-tega osebku najbližjega soseda (k je konstanta, odvisna od velikosti populacije in arhiva). Pravo uspešnost osebka $F(i)$ dobimo tako, da seštejemo grobo uspešnost in gostoto:

$$F(i) = R(i) + D(i).$$

Čeprav je F imenovana uspešnost, je to količina, ki jo želimo minimizirati.

Oglejmo si še enkrat primer s slike 8. Denimo, da želimo izmed vseh osebkov izbrati sedem najboljših (označeni so z belo barvo). Izbor najboljših šestih osebkov je enostaven – vzamemo tiste, ki imajo najmanjšo grobo uspešnost. Izmed treh osebkov z grobo uspešnostjo enako sedem pa se odločimo za tistega, ki ima najmanjšo gostoto.

Na sliki 9 vidimo oris algoritma SPEA2. Oglejmo si njegov glavni del (4. točka). Najprej nedominirane osebke iz stare populacije in arhiva prepišemo v nov arhiv $\mathcal{A}_t$. Če jih je preveč, iz arhiva odstranimo tiste, ki so najbliže drugim osebkom. Če je osebkov premalo, arhiv dopolnimo z dominiranimi osebki iz populacije in starega arhiva, ki imajo najmanjšo uspešnost. Če ustavitveni kriterij še ni izpolnjen, iz arhiva $\mathcal{A}_t$ generiramo novo populacijo $\mathcal{P}_t$ z uporabo turnirske selekcije, križanja in mutacije (podobno kot pri NSGA-II). Na koncu vse osebke iz populacije in arhiva ovrednotimo glede na moč, grobo uspešnost in gostoto.



Slika 8: Računanje moči in grobih uspešnosti osebkov v algoritmu SPEA2

SPEA2

1. Naključno generiraj in ovrednoti začetno populacijo $\mathcal{P}_0$.
2. Pripravi prazen začetni arhiv $\mathcal{A}_0$.
3. Postavi $t = 0$.
4. Ponavljaj:
   4.1. Postavi $t = t + 1$.
   4.2. V nov arhiv $\mathcal{A}_t$ prepiši vse nedominirane osebke iz $\mathcal{P}_{t-1}$ in $\mathcal{A}_{t-1}$.
   4.3. Če je nedominiranih osebkov več kot je velikost novega arhiva, izloči osebke, ki so blizu drugim osebkom.
   4.4. Sicer pa, če nedominirani osebki ne zapolnijo arhiva, arhiv dopolni z najboljšimi dominiranimi osebki iz $\mathcal{P}_{t-1}$ in $\mathcal{A}_{t-1}$.
   4.5. Če je zaustavitveni kriterij izpolnjen, končaj.
   4.6. Sicer populacijo $\mathcal{P}_t$ generiraj iz arhiva $\mathcal{A}_t$ z uporabo turnirske selekcije, križanja in mutacije.
   4.7. Ovrednoti vse osebke iz populacije $\mathcal{P}_t$ in arhiva $\mathcal{A}_t$.

Slika 9: Algoritem SPEA2

4.2. Diferencialna evolucija

Diferencialna evolucija (angl. differential evolution) je novejši evolucijski algoritem Pricea in Storna iz leta 1997 [25], ki se uspešno uporablja za optimiranje numeričnih funkcij. Kot lahko razberemo s slike 10, je to zelo enostaven algoritem, katerega največja slabost je, da ni primeren za kombinatorično optimiranje. Osebki morajo namreč biti zapisani v obliki vektorjev, da jih lahko seštevamo in množimo s skalarjem, kot to zahteva postopek za generiranje kandidata. Diferencialna evolucija je t.i. *algoritem s stabilnim stanjem* (angl. steady-state algorithm), pri katerem novi osebki (kandidati) ne sestavljajo nove populacije, ampak jih dodajamo v obstoječo populacijo. Ta način (ki ga uporabljajo tudi nekateri genetski algoritmi) pospešuje konvergenco, saj lahko pravkar izračunane kakovostne osebke takoj uporabimo za generiranje novih kandidatov.

Diferencialna evolucija

1. Naključno generiraj in ovrednoti začetno populacijo $\mathcal{P}$.
2. Dokler ustavitveni kriterij ni izpolnjen, ponavljaj:
   2.1. Za vsak osebek P_i ($i = 1, \ldots,$ *popSize*) iz $\mathcal{P}$ ponavljaj:
      (a) Iz starša P_i generiraj kandidata C.
      (b) Kandidata ovrednoti.
      (c) Kandidat zamenja starša, če je boljši od njega. Sicer kandidata zavržemo.
   2.2. Naključno oštevilči osebke iz $\mathcal{P}$.

Slika 10: Diferencialna evolucija

Ker se je diferencialna evolucija na številnih enokriterijskih optimizacijskih problemih izkazala kot bolj uspešna od nekaterih drugih evolucijskih algoritmov, so jo kmalu pričeli uporabljati tudi za večkriterijsko optimiranje [1, 22, 34]. Pri tem predstavlja največjo težavo primerjava med kandidatom in njegovim staršem (točka 2.1.(b) na sliki 10). Zato so omenjeni pristopi najprej generirali vse kandidate ene populacije in se šele nato (na različne načine) odločili, katere osebke bodo ohranili za naslednjo populacijo.

4.2.1. DEMO

V tem razdelku si bomo ogledali nov algoritem za večkriterijsko optimiranje, imenovan DEMO (Differential Evolution for Multiobjective Optimization) [26], ki je v primerjavi z NSGA-II in SPEA dosegel zelo dobre rezultate na petih testnih problemih. Na sliki 11 vidimo njegov potek. Generiranje kandidatov poteka po shemi DE/1/rand/bin, ki je opisana na sliki 12 (za

DEMO

1. Naključno generiraj in ovrednoti začetno populacijo $\mathcal{P}$.
2. Dokler zaustavitveni kriterij ni izpolnjen, ponavljaj:
   2.1. Za vsak osebek P_i ($i = 1, \ldots,$ *popSize*) iz $\mathcal{P}$ ponavljaj:
      (a) Iz starša P_i generiraj kandidata C.
      (b) Kandidata ovrednoti.
      (c) Če kandidat dominira starša, ga zamenja v populaciji.
      Če je kandidat dominiran od starša, ga zavrzi.
      Sicer kandidata dodaj v populacijo.
   2.2. Če ima populacija več osebkov kot *popSize*,
   jo zmanjšaj na velikost *popSize*.
   2.3. Naključno oštevilči osebke iz $\mathcal{P}$.

Slika 11: Algoritem DEMO

Generiranje kandidata

Vhod: Osebek P_i

1. Naključno izberi osebke $P_{i_1}, P_{i_2}, P_{i_3}$ iz $\mathcal{P}$, tako da so i, i_1, i_2 in i_3 paroma različni.
2. Kandidata izračunaj kot $C = P_{i_1} + F \cdot (P_{i_2} - P_{i_3})$, kjer je F faktor skaliranja.
3. Kandidata modificiraj z binarnim križanjem z njegovim staršem.

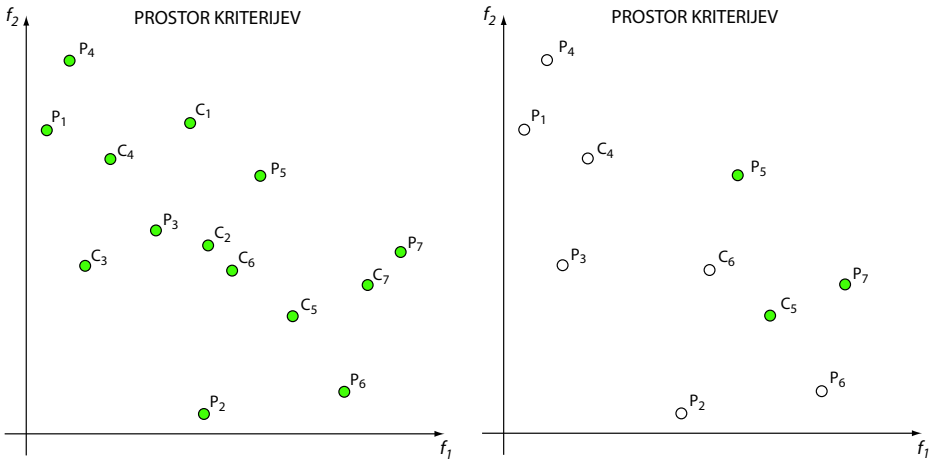
Izhod: Kandidat C

Slika 12: Generiranje kandidata po shemi DE/rand/1/bin

več informacij o shemah diferencialne evolucije glej [24]). Računanje kandidata je enako kot pri diferencialni evoluciji za enokriterijsko optimiranje.

DEMO vsako generacijo ponavlja naslednje korake. Najprej iz vsakega osebka v populaciji generira kandidata s postopkom, ki je opisan na sliki 12. Kandidata ovrednoti in primerja z njegovim staršem. Če kandidat dominira starša, ga zamenja v populaciji. Če je kandidat dominiran s strani starša, ga algoritem takoj zavrže. Sicer sta med seboj nedominirana, zato algoritem kandidata doda v populacijo. Ko to ponovi za vse osebke iz populacije, je velikost populacije med *popSize* in $2 \cdot$ *popSize*, zato jo mora zmanjšati na velikost *popSize*. Temu postopku rečemo tudi klestenje (angl. truncation). Za klestenje uporabi nedominirano sortiranje in metriko nakopičenosti na enak način kot NSGA-II.

Delovanje algoritma DEMO si lahko ogledamo tudi na sliki 13. Na levi polovici slike so hkrati prikazani vsi osebki populacije (P_i, $i = 1, \ldots, 7$) in kandidati (C_i, $i = 1, \ldots, 7$), kjer je



Slika 13: Klestenje populacije z algoritmom DEMO

kandidat C_i potomec osebka P_i (za vse $i = 1, \ldots, 7$). Na sliki 13 desno pa so narisani samo osebki, ki po enem koraku algoritma sestavljajo razširjeno populacijo. Po klestenju populacije z uporabo nedominiranega sortiranja in metrike nakopičenosti ostanejo v populaciji samo še osebki, ki so označeni z belo barvo.

DEMO obstaja v treh različicah. V različici DEMO/parent kandidata primerjamo z njegovim staršem (to je različica, ki smo jo opisali zgoraj). V DEMO/closest/dec, ga primerjamo z najbližjim osebkov v prostoru spremenljivk, pri čemer bližino merimo z evklidsko metriko. V različici DEMO/closest/obj pa kandidata primerjamo z najbližjim osebkom v prostoru kriterijev (spet uporabljamo evklidsko metriko).

5. Ostale metode

Za reševanje problemov večkriterijskega optimiranja poleg evolucijskih algoritmov uporabljajo tudi druge metahevristike. Med njimi najdemo:

- simulirano ohlajanje [6],
- iskanje s tabuji [13] ,
- optimiranje s kolonijami mravelj [16],
- porazdeljeno učenje s spodbujanjem (angl. distributed reinforcement learning) [23] in
- memetske algoritme (angl. memetic algorithms) [17, 20].

Več o naštetih in drugih metahevristikah za večkriterijsko optimiranje lahko preberemo v [3].

5.1. Simulirano ohlajanje

Simulirano ohlajanje (angl. simulated annealing) je izmed naštetih metod največkrat uporabljena metahevristika za večkriterijsko optimiranje. To je različica lokalnega iskanja, ki z neko verjetnostjo dovoli izbiro tudi trenutno slabših rešitev (da se ne ustavimo prehitro v nekem lokalnem optimumu). Ta verjetnost se sčasoma manjša. Ideja simuliranega ohlajanja prihaja iz termodinamike, saj molekule z višjo temperaturo lažje preidejo v druga stanja. Če temperaturo zmanjšujemo (sistem ohlajamo), je takšnih preskokov vedno manj.

S simuliranim ohlajanjem so sprva večkriterijske optimizacijske naloge reševali s prednostnim pristopom (kombiniranjem več kriterijev v enega samega). Kasneje so se pojavile številne različice, ki uporabljajo koncept Pareto dominantnosti, med njimi: PSA (Pareto Simulated Annealing) [6], UMOSA (Ulungu Multiobjective Simulated Annealing) [32], SMOSA (Suppaptinarm Multiobjective Simulated Annealing) [31], WMOSA (Weight-based Multiobjective Simulated Annealing) in PDMOSA (Pareto-domination Based Multiobjective Simulated Annealing) (oba [29])[1].

5.2. Iskanje s tabuji

Iskanje s tabuji (angl. tabu search) je optimizacijska metoda, izpeljana iz lokalne optimizacije, ki uporablja seznam prepovedanih rešitev (tabujev) zato, da se v preiskovanju ne zacikla. Na vsakem koraku izberemo najboljšega soseda trenutne rešitve, ki ni na seznamu tabujev. To storimo, tudi če je nova rešitev slabša od trenutne. Najboljšo dobljeno rešitev imamo shranjeno posebej in jo v vsakem koraku primerjamo z novo rešitvijo. Po vsakem koraku osvežimo tudi seznam tabujev, ki vsebuje pred kratkim pregledane rešitve.

Ker se iskanje s tabuji giblje v bližini trenutne rešitve, je največji izziv pri uporabi te metode za večkriterijsko optimiranje obdržati raznolikost rešitev (drugi cilj večkriterijskega

[1]Navedeni akronimi so povzeti po primerjalni študiji [30].

optimiranja). Iskanje s tabuji se v večkriterijskem optimiranju pojavlja kot samostojna metoda [13, 2] in tudi kot hibridni pristop (v kombinaciji z evolucijskimi algoritmi) [21, 18].

6. Zaključek

Večkriterijsko optimiranje je zaradi zahteve po hkratnem optimiranju več konfliktnih kriterijev mnogo zahtevnejše od enokriterijskega. Zato ni presenetljivo, da so v preteklosti večkriterijske probleme reševali s pretvorbo v enokriterijske. Evolucijski algoritmi so kot populacijske metode zelo primerni za reševanje problemov, pri katerih želimo v enem samem zagonu algoritma dobiti več optimalnih rešitev. Z njihovimi uspehi na nalogah večkriterijskega optimiranja je to področje v kratkem času pridobilo veliko zanimanje raziskovalcev s celega sveta. Počasi se pozornost z razvoja novih algoritmov preusmerja na težave večkriterijskega optimiranja, kot so:

- predstavitev rešitev v problemih z velikim številom kriterijev,
- iskanje ustreznih metrik za vrednotenje množic nedominiranih rešitev,
- definiranje testnih problemov,
- obravnavanje omejitev v večkriterijskih optimizacijskih problemih,
- dokazi konvergence ipd.

Ob strmo naraščajočem številu objav na temo večkriterijskega optimiranja se ni bati, da bi zanimanje za to področje kmalu usahnilo.

Literatura

[1] H. A. Abbass, R. Sarker in C. Newton. PDE: A pareto-frontier differential evolution approach for multi-objective optimization problems. V *Proceedings of the Congress on Evolutionary Computation 2001 (CEC'2001)*, volume 2, str. 971–978, Piscataway, New Jersey, 2001. IEEE Service Center.

[2] J. Balicki in Z. Kitowski. Multicriteria evolutionary algorithm with tabu search for task assignment. V *First International Conference on Evolutionary Multi-Criterion Optimization*, str. 373–384. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[3] C. A. Coello Coello, D. A. van Veldhuizen in G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, 2002.

[4] D. W. Corne, N. R. Jerram, J. D. Knowles in M. J. Oates. PESA-II: Region-based selection in evolutionary multiobjective optimization. V *Proceedings of the Genetic in Evolutionary Computation Conference (GECCO-2001)*, str. 283–290, San Francisco, CA, 2001. Morgan Kaufmann.

[5] D. W. Corne, J. D. Knowles in M. J. Oates. The pareto envelope-based selection algorithm for multiobjective optimization. V *Proceedings of the Parallel Problem Solving from Nature VI Conference*, str. 839–848, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.

[6] P. Czyzak in A. Jaszkiewicz. Pareto simulated annealing—A metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7: 34–47, 1998.

[7] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001.

[8] K. Deb, S. Agrawal, A. Pratab in T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. KanGAL Report 200001, Indian Institute of Technology, Kanpur, India, 2000.

[9] L. J. Fogel, A. J. Owens in M. J. Walsh. *Artificial Intelligence Through Simulated Evolution*. Wiley, New York, 1966.

[10] C. M. Fonseca in P. J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion in generalization. V *Proceedings of the Fifth International Conference on Genetic Algorithms*, str. 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[11] D. E. Goldberg. *Genetic Algorithms in Search, Optimization in Machine Learning*. Addison-Wesley, 1989.

[12] Y. Y. Haimes, L. S. Lasdon in D. A. Wismer. On a bicriterion formulation of the problems of integrated system identification in system optimization. *IEEE Transactions on Systems, Man in Cybernetics*, 1(3): 296–297, 1971.

[13] M. P. Hansen. Tabu search in multiobjective optimisation: MOTS. V *Proceedings of the 13th International Conference on Multiple Criteria Decision Making (MCDM'97)*, Cape Town, South Africa, 1997.

[14] J. H. Holland. *Adaptation in Natural in Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[15] J. Horn, N. Nafpliotis in D. E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. V *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, str. 82–87, Piscataway, New Jersey, 1994. IEEE Service Center.

[16] S. Iredi, D. Merkle in M. Middendorf. Bi-criterion optimization with multi colony ant algorithms. V *First International Conference on Evolutionary Multi-Criterion Optimization*, str. 359–372. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[17] A. Jaszkiewicz. Genetic local search for multiple objective combinatorial optimization. Technical Report RA-014/98, Institute of Computing Science, Poznan University of Technology, 1998.

[18] E. F. Khor, K. C. Tan in T. H. Lee. Tabu-based exploratory evolutionary algorithm for effective multi-objective optimization. V *First International Conference on Evolutionary Multi-Criterion Optimization*, str. 344–358. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[19] J. D. Knowles in D. W. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2): 149–172, 2000.

[20] J. D. Knowles in D. W. Corne. M-PAES: A memetic algorithm for multiobjective optimization. V *2000 Congress on Evolutionary Computation*, volume 1, str. 325–332, Piscataway, New Jersey, 2000. IEEE Service Center.

[21] S. Kurahashi in T. Terano. A genetic algorithm with tabu search for multimodal in multiobjective function optimization. V *Proceedings of the Genetic in Evolutionary Computation Conference (GECCO'2000)*, str. 291–298. Morgan Kaufmann, 2000.

[22] N. K. Madavan. Multiobjective optimization using a pareto differential evolution approach. V *Congress on Evolutionary Computation (CEC'2002)*, volume 2, str. 1145–1150, Piscataway, New Jersey, 2002. IEEE Service Center.

[23] C. E. Mariano in E. F. Morales. Distributed reinforcement learning for multiple objective optimization problems. V *2000 Congress on Evolutionary Computation*, volume 1, str. 188–195, Piscataway, New Jersey, 2000. IEEE Service Center.

[24] K. V. Price in R. Storn. Differential evolution homepage. http://www.icsi.berkeley.edu /~storn/code.html.

[25] K. V. Price in R. Storn. Differential evolution – A simple evolution strategy for fast optimization. *Dr. Dobb's Journal*, 22(4): 18–24, 1997.

[26] T. Robič in B. Filipič. DEMO: Differential evolution for multiobjective optimization. V *Third International Conference on Evolutionary Multi-Criterion Optimization, EMO 2005*, str. 520–533. Springer-Verlag. Lecture Notes in Computer Science No. 3410, 2005.

[27] J. D. Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, 1984.

[28] N. Srinivas in K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3): 221–248, 1994.

[29] B. Suman. Multiobjective simulated annealing–A metaheuristic technique for multiobjective optimization of a constrained problem. *Foundations of Computing in Decision Sciences*, 27(3): 171–, 2002.

[30] B. Suman. Study of simulated annealing based algorithms for multiobjective optimization of a constrained problem. *Computers in Chemical Engineering*, 28(9): 1849–1871, 2004.

[31] A. Suppapitnarm, K. A. Seffen, G. T. Parks in P. J. Clarkson. A simulated annealing algorithm for multiobjective optimization. *Engineering Optimization*, 33(1): 59–85, 2000.

[32] E. L. Ulungu, J. Teghem, Ph. Fortemps in D. Tuyttens. MOSA method: A tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 8(4): 221–236, 1999.

[33] D. A. Van Veldhuizen in G. B. Lamont. Multiobjective optimization with messy genetic algorithms. V *Proceedings of the 2000 ACM Symposium on Applied Computing*, str. 470–476, Villa Olmo, Como, Italy, 2000. ACM.

[34] F. Xue, A. C. Sanderson in R. J. Graves. Pareto-based multi-objective differential evolution. V *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, volume 2, str. 862–869, Canberra, Australia, 2003. IEEE Press.

[35] E. Zitzler, M. Laumanns in L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. Tik report no. 103, Computer Engineering in Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 2001.

[36] E. Zitzler in L. Thiele. An evolutionary algorithm for multiobjective optimization: The strength pareto approach. Technical Report 43, Computer Engineering in Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 1998.

[37] J. B. Zydallis, D. A. Van Veldhuizen in G. B. Lamont. A statistical comparison of multiobjective evolutionary algorithms including the MOMGA–II. V *First International Conference on Evolutionary Multi-Criterion Optimization*, str. 226–240. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.