Kvantna optimizacija energetskih sustava: Problem opredjeljenja za jedinice u proizvodnji električne energije

Parmać, Dora

Kvantna optimizacija energetskih sustava: Problem opredjeljenja za jedinice u proizvodnji električne energije

Parmać, Dora

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Dora Parmać

KVANTNA OPTIMIZACIJA ENERGETSKIH SUSTAVA: PROBLEM OPREDJELJENJA ZA JEDINICE U PROIZVODNJI ELEKTRIčNE ENERGIJE

Diplomski rad

Voditelj rada:
izv. prof. dr. sc. Matija Kazalicki

Zagreb, 2022

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Hvala mojoj obitelji koja je uvijek bila uz mene. Cijenim vašu ljubav i trud kako bi me izveli na pravi put. Hvala mom mentoru na pomoći i vodstvu pri pisanju ovog rada.

Sadržaj

Uvod

Uz rastuću potražnju za energijom i potrebu za zaštitom okoliša, pojačava se primarni interes za projektiranje, kontrolu i planiranje energetskih sustava. Novi energetski resursi integriraju se u energetske sustave koji omogućuju optimalno upravljanje, a kontrola raspoloživih resursa ključan je problem u korištenju novih tehnologija. Bez optimalnog iskorištavanja resursa, trošak ulaganja u te tehnologije ne može se opravdati. Stoga, alati i algoritmi za optimizaciju pružaju prikladan način rješavanja problema složenih energetskih sustava u ovom polju.[1]

Svjetska energetska kriza zahtijeva razvoj hibridnih obnovljivih energetskih sustava umjesto fosilnih goriva. Iako su obnovljivi izvori energije "besplatni", teško ih je predvidjeti zbog različitih čimbenika kao što su fluktuacija sunčevog zračenja, vremenske prilike i brzina vjetra. Zasad su predstavljeni hibridni energetski sustavi koji koriste kombinaciju solarne energije, vjetra i vode kako bi se poboljšala njihova pouzdanost. Razmjeri mogu varirati od male jedinice koja opskrbljuje struju za jedan dom do velike jedinice koja može napajati selo ili otok. Metode optimizacije koriste se kako bi se pronašla najjeftinija kombinacija svih generatora energije (i obnovljivih i konvencionalnih) i skladišnog kapaciteta koji može podržati očekivanu potražnju. Modeli također moraju uključivati meteorološke podatke, solarne, hidro, vjetro i baterijske sustave.

Obnovljiva energija također prirodno dovodi do distribuiranih energetskih resursa (DER) kao što su krovne solarne PV jedinice. DER-ovi mogu uključivati neobnovljivu proizvodnju, skladištenje baterija, električna vozila i druge tehnologije upravljanja energijom u domu koje uključuju internet stvari u pametnom domu (*eng. internet of things*). Metode optimizacije potrebne su kako bi se podržao rad mreže ili "pametne mreže", tj. potrebne su za pronalaženje savršene ravnoteže između pouzdanosti, dostupnosti, učinkovitosti i cijene. Optimizacija mreže kreće od proizvodnje energije, prijenosa, distribucije pa sve do upravljanja potražnjom.

Provođenje takve optimizacije u velikim razmjerima je računski skupo. Na primjer, samo pronalaženje optimalnog broja jedinica za proizvodnju energije za danu potražnju zahtijeva

vrijeme izračuna koje raste eksponencijalno s brojem varijabli u modelu. To znači da će se potrebni računski resursi udvostručiti svaki put kada se u mrežu doda novi čvor kako bi se pronašla najbolja operacija pametne mreže.

Srećom, kvantno računarstvo pruža novu paradigmu za rješavanje složenih problema optimizacije. Kvantno računalo obrađuje informacije drugačije od konvencionalnog ili 'klasičnog' računala. Dvije su temeljne razlike:

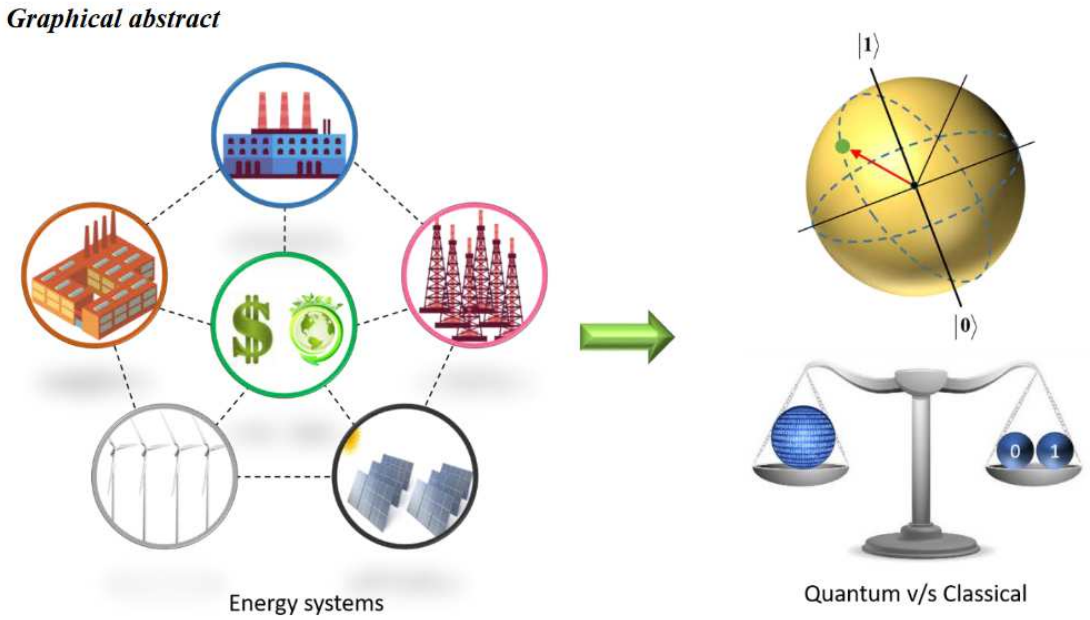
1. Klasična procesna jedinica ili bit može biti 0 ili 1, dok kvantna procesna jedinica poznata kao kvantni bit ili kubit (*eng. qubit*) može biti 0 i 1 u isto vrijeme. To omogućuje kvantnom računalu da istovremeno istražuje različita rješenja zadanog problema. Kubit preciznije definiramo u Poglavlju 1.

2. Dva ili više udaljenih kubita mogu odmah 'osjetiti' što se događa s drugim kubitima bez slanja ikakvih signala. Fenomen je poznat kao kvantna zapetljanost (*eng. quantum entanglement*), te je neophodan sastojak za kvantno ubrzanje. Pojam kvantne zapetljanosti dan je informativno da bi se intuitivno predočila razlika između kvantnog i klasičnog računarstva, te ga u ovom radu ne definiramo detaljnije.

Kvantno računarstvo je tehnologija koja mijenja igru i koja se može uhvatiti u koštac s nerješivim problemima, te proizvesti dobra rješenja u razumnom vremenu rada. Za razliku od klasičnih računala koja istražuju cijeli prostor izvedivih rješenja, kvantna računala se fokusiraju na istraživanje odabranih izvedivih podprostora, čime se inducira kvantno ubrzanje.

U ovom radu predstavit ćemo jedno od rješenja za energetske sustave inspirirano kvantnim računanjem. Upoznat ćemo se s radom kvantnog računala i njegovim glavnim razlikama s konvencionalnim CPU/GPU baziranim klasičnim računalima. Naglasak rada je na rješenju "Problema opredjeljenja jedinica" (*eng. unit commitment problem or UC problem*) koji se bavi minimiziranjem ukupnih operativnih troškova generatora koji proizvode struju. Problem se svodi na usklađivanje rada generatora da bi se zadovoljila tražena potražnja uz minimalne troškove. U Poglavlju 1 upoznat ćemo se s osnovim pojmovima kvantnog računarstva. U Poglavlju 2 dajemo matematičku formulaciju Problema opredjeljenja jedinica, te predstavljamo Algoritam optimizacije rojem čestica - PSO algoritam. PSO algoritam pokazao se kao učinkovita tehnika za rješavanje UC problema kroz zadnjih nekoliko godina, te će nam on biti osnovni algoritam kojeg ćemo nadograđivati kroz cijeli rad. Budući se UC problem svodi na određivanje koji je od generatora uključen (u stanju "ON" ili u stanju 1), a koji isključen (u stanju "OFF" ili u stanju 0) kroz dani vremenski period, u 2.3 uvodimo binarnu inačicu PSO algoritma - BPSO koji se može koristiti u diskretnim prostorima pretraživanja kako bi najlakše reprezentirali diskretna stanja 0 i 1 danih

generatora. U Poglavlju 3 proučavamo kvantno inspirirani Evolucijski algoritam - QEA, njegovu primjenu na problem ruksaka, te kvantno inspiriranu inačicu BPSO algoritma - QBPSO. Ta dva algoritma, QEA i QBPSO koriste se u Poglavlju 4 kako bi konačno opisali LAQPSO algoritam. Važno je naglasiti da spomenuti algoritmi nisu kvantni algoritmi, već vjerojatnosni algoritmi inspirirani kvantnim računanjem. LAQPSO algoritam nastaje kao kombinacija QEA i QBPSO algoritama i prema obavljenim eksperimentima, pokazao se kao (trenutno) najbolji algoritam za rješavanje UC problema. U 4.4 prikazani su rezultati simulacije (Tablica 4.6) i usporedba tri spomenuta algoritma: BPSO, IQBPSO, LAQPSO gdje možemo vidjeti očitu prednost LAQPSO algoritma nad ostalima.



Slika 0.1: Usporedba energetskih sistema [1]

Poglavlje 1

Osnovni pojmovi kvantnog računarstva

U ovom poglavlju referiramo se na knjigu *Quantum Computation and Quantum Information* [9], s naglaskom na Poglavlje 1: *Fundamental concepts*.

1.1 Jedan kubit

Bit je temeljni koncept klasičnog računanja i klasične informacije. Kvantno računanje i kvantne informacije izgrađene su na analognom konceptu, kvantnom bitu, ili skraćeno kubitu (*eng. qubit*). Kubite tretiramo kao apstraktne entitete. Ljepota tretiranja kubita kao apstraktnih entiteta je u tome što nam daje slobodu da izgradimo opću teoriju kvantnog računanja i kvantnih informacija koja za svoju realizaciju ne ovisi o specifičnom sustavu. Kao što klasični bit ima stanje 0 ili 1, možemo definirati i stanje kubita. Stanje kubita definira se kao u 1.1 tako da kubit ima beskonačno mnogo stanja, ali budući ga mjerimo (u bazi $|0\rangle$, $|1\rangle$), dobit ćemo samo $|0\rangle$ ili $|1\rangle$. Možemo primijetiti da dana stanja odgovaraju stanjima 0 i 1 klasičnog bita. Oznaka $|\rangle$ naziva se Diracova notacija, te je standardna notacija za oznaku stanja u kvantnoj mehanici. Glavna razlika između bita i kubita je u tome što kubit može biti i u nekim drugim stanjima, različitima od $|0\rangle$ i $|1\rangle$. Također, moguće je formirati linearne kombinacije stanja ili superpozicije: [9, p. 13]

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{1.1}$$

Brojevi α i β su kompleksni brojevi, iako za mnoge svrhe možemo o njima razmišljati kao o realnim brojevima. Stanje kubita možemo definirati na sljedeći način:

Definicija 1.1. *Stanje kubita je vektor norme 1 u unitarnom dvodimenzionalnom vektorskom prostoru $V|_{\mathbb{C}}$ s bazom $\{|0\rangle, |1\rangle\}$ i skalarnim produktom $\langle\cdot|\cdot\rangle$ td.:*

$$\begin{aligned} \langle 0|0\rangle &= \langle 1|1\rangle = 1 \\ \langle 0|1\rangle &= \langle 1|0\rangle = 0 \end{aligned} \tag{1.2}$$

Drugim riječima, posebna stanja $|0\rangle$ i $|1\rangle$ poznata su kao računska bazna stanja, te tvore ortonormiranu bazu za ovaj vektorski prostor.

Nadalje, možemo i ispitati je li kubit u stanju 0 ili 1; računala to rade cijelo vrijeme kada dohvaćaju sadržaj svoje memorije. Ono što ne možemo jest odrediti vrijednosti α i β. Umjesto toga, kvantna mehanika nam govori da možemo steći samo ograničene informacije o kvantnom stanju. Kada mjerimo kubit dobivamo ili rezultat 0, s vjerojatnošću $|\alpha|^2$, ili rezultat 1, s vjerojatnošću $|\beta|^2$. Prirodno, $|\alpha|^2 + |\beta|^2 = 1$, budući suma vjerojatnosti mora biti 1. Geometrijski, to možemo protumačiti kao uvjet da se stanje kubita normalizira na duljinu 1.

Sposobnost kubita da bude u stanju superpozicije protivi se zdravorazumskom razumijevanju fizičkog svijeta oko nas. Klasični bit je poput novčića: ili pismo ili glava (za nesavršene kovanice mogu postojati srednja stanja kao što je balansiranje na rubu, ali oni se u idealnom slučaju mogu zanemariti). Suprotno tome, kubit može postojati u kontinuumu stanja između $|0\rangle$ i $|1\rangle$ – dok se ne promatra. Važno je napomenuti da kada se kubit mjeri, kao rezultat mjerenja dobijemo samo 0 ili 1 s određenom vjerojatnošću. Na primjer, kubit može biti u stanju

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \tag{1.3}$$

koji, prilikom mjerenja, 50% vremena ($|\frac{1}{\sqrt{2}}|^2$) daje rezultat 0, te ostalih 50% vremena daje rezultat 1.

U ovoj fazi, jedan kubit bi se mogao koristiti za pohranjivanje beskonačne količine informacija. Ipak, baš kao i klasični bit, kubit se mora pročitati da bi se dobila njegova vrijednost za računanje. Da bismo dobili ovu vrijednost, moramo dizajnirati mjernu operaciju kako bi rezultat mogao biti samo 0 ili 1. Kao što smo napomenuli, mjerenje mijenja stanje kubita, sažimajući ga iz njegove superpozicije $|0\rangle$ i $|1\rangle$ u specifično stanje u skladu s rezultatom mjerenja (pojednostavljeno, stanje $|\psi\rangle$ "sruši" se na $|0\rangle$ ili $|1\rangle$ ovisno o vjerojatnosti). Dakle, ishod kubita nije deterministički kao u klasičnom računarstvu, već vjerojatnosni.

Unatoč ovoj neobičnosti, kubiti su izrazito stvarni i za realizaciju kubita mogu se koristiti različiti fizički sustavi. Da bismo dobili konkretan dojam, korisno je navesti nekoliko primjera kako se kubit može realizirati: kao dvije različite polarizacije fotona; kao poravnanje nuklearnog spina u jednoličnom magnetskom polju; kao dva stanja elektrona koja kruže oko jednog atoma, itd... U modelu atoma, elektron može postojati ili u tzv. "osnovnom" ili "uzbuđenom" stanju, koja ćemo nazvati $|0\rangle$ i $|1\rangle$, redom. Obasjavajući svjetlo na atom, uz odgovarajuću energiju i tijekom određenog vremena, moguće je pomicati elektron iz stanja $|0\rangle$ u stanje $|1\rangle$ i obrnuto. Ali još zanimljivije, smanjujući vrijeme tijekom kojeg obasjavamo elektron svjetlošću, elektron može na početku biti u stanju $|0\rangle$ i pomaknuti se "na pola puta" između $|0\rangle$ i $|1\rangle$, u stanje $|+\rangle$ - oznaka za stanje 1.3

Budući $|\alpha|^2 + |\beta|^2 = 1$, jednadžbu 1.1 možemo zapisati kao:

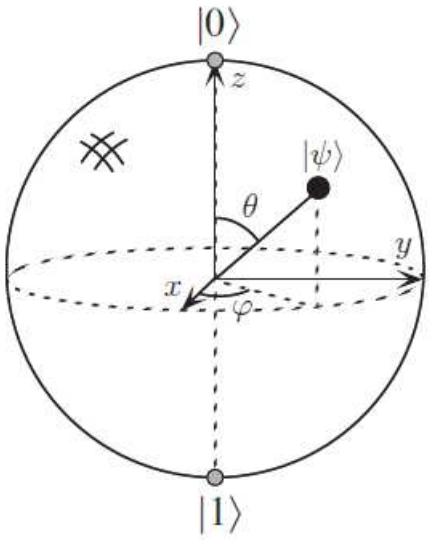
$$|\psi\rangle = e^{i\gamma}\Big(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle\Big), \tag{1.4}$$

gdje su θ, φ, γ realni brojevi. Također je važno napomenuti da možemo ignorirati faktor $e^{i\gamma}$ na početku jer "nema vidljivih efekata". [9, p. 15] Stoga prethodnu jednažbu možemo zapisati kao:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle \tag{1.5}$$

Brojevi θ i φ definiraju točku na jediničnoj trodimenzionalnoj sferi, kao što je prikazano na slici 1.1 Ova se sfera često naziva Blochova sfera, te pruža korisno sredstvo za vizualizaciju stanja jednog kubita i često služi kao izvrsna testna baza za ideje o kvantnom računanju i kvantnim informacijama. Međutim, treba imati na umu da je ova intuicija ograničena jer (zasad) ne postoji jednostavna generalizacija Blochove sfere za više kubita.

Koliko informacija predstavlja kubit? Paradoksalno, postoji beskonačnan broj točaka na jediničnoj sferi, tako da se u načelu može pohraniti cijeli tekst Shakespearea u beskonačnoj binarnom razvoju θ. Međutim, pokazalo se da ovaj zaključak može dovesti u zabludu, zbog ponašanja kubita kada se promatra. Prisjetimo se činjenice da mjerenje mijenja stanje kubita, sažimajući ga iz njegove superpozicije $|0\rangle$ i $|1\rangle$ u specifično stanje u skladu s rezultatom mjerenja. Na primjer, ako mjerenje $|+\rangle$ daje 0, tada će stanje kubita nakon mjerenja biti $|0\rangle$. Zašto se ova vrsta kolapsa pojavljuje? Nitko ne zna. Ovo ponašanje je jednostavno jedno od temeljnih postulata kvantne mehanike. Ono što je relevantno za naše svrhe je to da se iz jednog mjerenja dobiva samo jedan bit informacije o stanju ku-



Slika 1.1: Reprezentacija kubita pomoću Blochove sfere [9]

bita, čime se rješava prividni paradoks. Ako bi se izmjerilo beskonačno mnogo identičnih kubita, jedno tada bi mogli odrediti $|\alpha|^2$ i $|\beta|^2$ za kubit u stanju danom u jednadžbi 1.1.

1.2 Više kubita

Pretpostavimo da imamo dva kubita. Da su to dva klasična bita, onda bi bila četiri moguća stanja, 00, 01, 10 i 11. Sukladno tome, sustav s dva kubita ima četiri računska bazna stanja označena $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$. Par kubita također može postojati u stanju superpozicije, tako da kvantno stanje dva kubita uključuje povezivanje kompleksnog koeficijenta, koji se ponekad naziva amplituda, sa svakim računarskim osnovnim stanjem, tako da je vektor stanja koji opisuje dva kubita:

$$|\psi\rangle = \alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle + \alpha_{10}\,|10\rangle + \alpha_{11}\,|11\rangle \tag{1.6}$$

Slično kao i za jedan kubit, pojavljuje se rezultat mjerenja $x = (00, 01, 10 \text{ ili } 11)$ s vjerojatnošću $|\alpha_x|^2$, pri čemu je stanje kubita nakon mjerenja $|x\rangle$. Uvjet da je zbroj vjerojatnosti jednak jedan izražen je uvjetom normalizacije $\sum_{x\in\{0,1\}^2} |\alpha_x|^2 = 1$ gdje notacija $\{0,1\}^2$ predstavlja skup stringova duljine 2 gdje je svako slovo nula ili jedan. Za sustav s dva ku-

bita, mogli bismo izmjeriti samo podskup kubita - recimo prvi kubit: samo mjerenje prvog kubita daje 0 s vjerojatnošću $|\alpha_{00}|^2 + |\alpha_{01}|^2$, ostavljajući stanje nakon mjerenja

$$|\psi'\rangle = \frac{\alpha_{00}|00\rangle + \alpha_{01}|01\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}} \tag{1.7}$$

Primijetimo da je stanje nakon mjerenja re-normalizirano faktorom $\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}$ tako da zadovoljava uvjete normalizacije. [9, p. 16]

Općenitije, možemo promatrati sustav od n kubita. Tada su bazna stanja sustava oblika $|x_1, x_2 \dots x_n\rangle$ pa je i kvantno stanje ovog sustava specificirano s 2^n amplituda.

1.3 Model kvantnih sklopova

Promjene koje se događaju u kvantnom stanju mogu se opisati jezikom kvantnog računanja. Analogno načinu na koji se klasično računalo gradi iz električnog sklopa koji sadrži žice i logička vrata, kvantno računalo je izgrađeno od kvantnog sklopa koji sadrži "žice" i elementarna kvantna vrata za prijenos i manipuliranje kvantne informacije.

Kvantna vrata s jednim kubitom

Klasični računalni sklopovi sastoje se od žica i logičkih vrata. Žice se koriste za prijenos informacije unutar sklopova, dok logička vrata vrše manipulacije informacijama, pretvarajući ih iz jednog oblika u drugi. Uzmimo za primjer klasična logička vrata jednog bita. Jedini netrivijalni član ove klase su NOT vrata, čiji je rad definiran vlastitom tablicom istinitosti, u kojoj vrijedi: $0 \longrightarrow 1$ i $1 \longrightarrow 0$, odnosno stanja 0 i 1 se izmijenjuju. Analogno, možemo definirati kvantna NOT vrata za kubite.

Pretpostavimo da imamo neki proces koji mijenja stanje $|0\rangle$ u stanje $|1\rangle$ i obratno. Takav proces bi očito bio dobar kandidat za kvantni analogon NOT vrata. Međutim, kada specificiramo akciju vrata na stanjima $|0\rangle$ i $|1\rangle$ ne znamo što se događa u superpoziciji stanja $|0\rangle$ i $|1\rangle$ bez dodatnih znanja o svojstvima kvantnih vrata. Kao i sva kvantna vrata i NOT vrata su linearna, tj, stanje:

$$\alpha|0\rangle + \beta|1\rangle \tag{1.8}$$

pretvaraju u odgovarajuće stanje u kojem su $|0\rangle$ i $|1\rangle$ zamijenjeni:

$$\alpha|1\rangle + \beta|0\rangle \tag{1.9}$$

Zašto kvantna vrata djeluju linearno, a ne na neki nelinearan način, vrlo je zanimljivo pitanje, a odgovor uopće nije očit. Ispada da je ovo linearno ponašanje opće svojstvo kvantne mehanike i vrlo dobro motivirano empirijski; štoviše, nelinearno ponašanje može dovesti do prividnih paradoksa kao što su putovanje kroz vrijeme, komunikacija brža od brzine svjetlosti i kršenje drugog zakona termodinamike.

Postoji prikladan način predstavljanja kvantnih vrata u matričnom obliku, što izravno slijedi iz linearnosti kvantnih vrata. Zbog reverzibilnosti, kvantna vrata imaju istu količinu ulaznih i izlaznih kubita.

Pretpostavimo da definiramo matricu X koja predstavlja kvantna vrata na sljedeći način:

$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{1.10}$$

Ako je kvantno stanje $\alpha|0\rangle + \beta|1\rangle$ prikazano kao vektor:

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{1.11}$$

gdje gornji ulaz predstavlja amplitudu za $|0\rangle$ a donji ulaz amplitudu za $|1\rangle$, tada je odgovarajući izlaz kvantnih NOT vrata:

$$X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} \tag{1.12}$$

Dakle, kvantna vrata na jednom kubitu mogu se opisati $2x2$ matricom. No postoje li ograničenja na matrice koje se mogu koristiti kao kvantna vrata? Podsjetimo da uvjet normalizacije zahtijeva $|\alpha|^2 + |\beta|^2 = 1$ za kvantno stanje $\alpha|1\rangle + \beta|0\rangle$. To također mora vrijediti za kvantno stanje $|\psi'\rangle = \alpha'|1\rangle + \beta'|0\rangle$ nakon djelovanja vrata. Matrica U koja opisuje jednokubitna vrata mora biti unitarna, to jest $U^\dagger U = I$, gdje je $U^\dagger$ hermitski adjunkt od U (dobije se transponiranjem, a zatim kompleksnim konjugiranjem matrice U), a I je $2x2$ matrica identiteta. Na primjer, za NOT vrata lako je provjeriti da vrijedi dani uvjet. To je jedini uvjet na kvantnim vratima, te svaka unitarna matrica predstavlja valjana kvantna vrata.

Za razliku od klasičnih vrata, postoji mnogo netrivijalnih kvantnih vrata za jedan kubit. Navedimo nekoliko važnih primjera:

1. Z vrata koja ne mijenjaju $|0\rangle$, već mijenjaju $|1\rangle$ u $-|1\rangle$

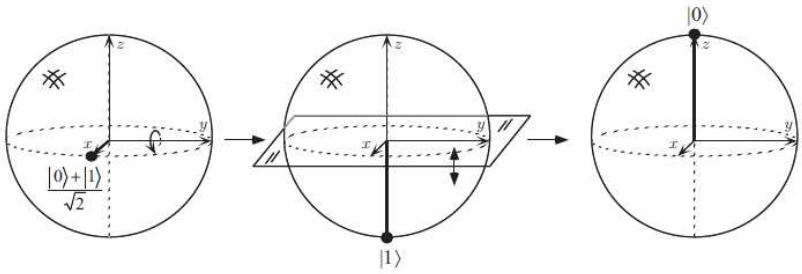
$$Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{1.13}$$

2. Hadamardova vrata koja $|0\rangle$ pretvaraju u $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$, te $|1\rangle$ pretvaraju u $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ (odnosno "na pola puta" između $|0\rangle$ i $|1\rangle$). Primijetimo da je $H^2 = I$.

$$H \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{1.14}$$

3. Rotacijska vrata predstavljaju θ-rotaciju jednog kubita θ

$$R \equiv \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \tag{1.15}$$

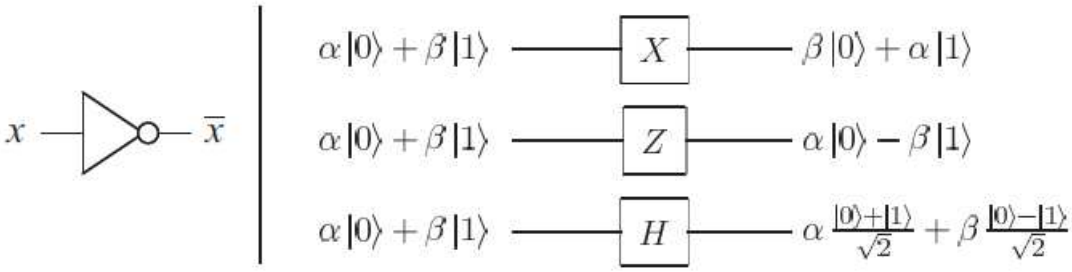


Slika 1.2: Vizualizacija Hadamardovih vrata na Blochovoj sferi [9]

Hadamardova vrata jedna su od najkorisnijih kvantnih vrata i možemo ih vizualizirati promatrajući sliku Blochove sfere. Na slici 1.2 vidimo da jednokubitna vrata odgovaraju rotacijama i refleksijama sfere. Hadamardova operacija je samo rotacija sfere oko y osi za 90°, nakon čega slijedi rotacija oko osi x za 180°. Također, na slici 1.3 vidimo usporedbu klasičnih i kvantnih logičkih vrata jednog bita.

Kvantna vrata s više kubita

Generalizirajmo sada kvantna vrata za više kubita. Slika 1.4 prikazuje pet značajnih klasičnih vrata za više bitova: AND, OR, XOR (ekskluzivno ILI), NAND i NOR vrata. Važan teoretski rezultat je da se bilo koja funkcija na bitovima može izračunati iz kompozicije samo



Slika 1.3: Klasična logička vrata i kvantna logička vrata [9]

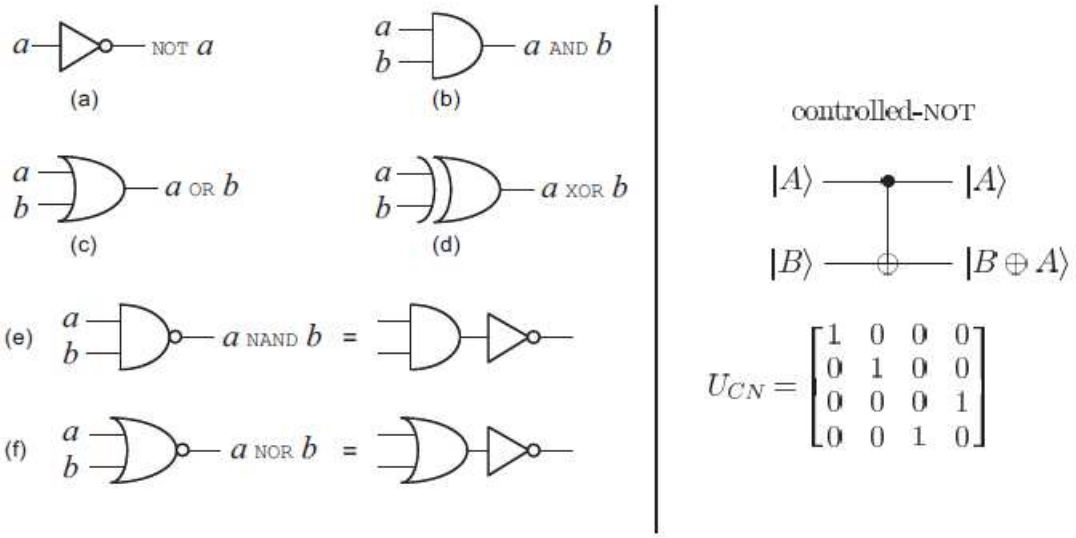
NAND vrata, te su ona poznata kao univerzalna vrata. Suprotno tome, XOR vrata, sama ili zajedno s NOT vratima nisu univerzalna. Jedan od načina kako to možemo vidjeti jest da primjena XOR vrata ne mijenja ukupni paritet bitova. Kao rezultat, bilo koji sklop koji uključuje samo NOT i XOR vrata će, ako dva ulaza x i y imaju isti paritet, dati izlaze s istim paritetom, ograničavajući klasu funkcija koje se mogu izračunati, i na taj način isključujući univerzalnost.

Prototipska kvantna logička vrata s više kubita su kontrolirana NOT ili CNOT vrata. Sadrže dva ulazna kubita, poznatija kao kontrolni (*eng. control qubit*) kubit i ciljni (*eng. target qubit*) kubit. Sklopni prikaz CNOT vrata prikazan je u gornjem desnom kutu slike 1.4; gornja linija predstavlja kontrolni kubit, dok donja linija predstavlja ciljni kubit.

Akciju danih vrata možemo opisati na sljedeći način: ako je kontrolni kubit postavljen na 0, ciljni kubit se ne mijenja. Ako je kontrolni kubit postavljen na 1, ciljni kubit se zamijeni:

$$|00\rangle \longrightarrow |00\rangle\,; \quad |01\rangle \longrightarrow |01\rangle\,; \quad |10\rangle \longrightarrow |11\rangle\,; \quad |11\rangle \longrightarrow |10\rangle \tag{1.16}$$

Još jedan način opisivanja rada kontroliranih CNOT vrata jest generalizacija klasičnih XOR vrata, budući se akcija može sažeti kao: $|A, B\rangle \longrightarrow |A, B \oplus A\rangle$, gdje je $\oplus$ zbrajanje modulo dva, što je upravo ono što rade XOR vrata, tj. na kontrolni i ciljni kubit je primijenjena XOR operacija, te je rezultat spremljen u ciljni kubit. Također, CNOT vrata možemo opisati i matričnom reprezentacijom. Lako se provjeri da prvi redak matrice U_{CN} opisuje transformaciju nad $|00\rangle$ (slično se može provjeriti i za ostala stanja). U slučaju jednog kubita, uvjet da je vjerojatnost očuvana opisana je činjenicom da je U_{CN} unitarna matrica, tj. vrijedi $U_{CN}^{\dagger} U_{CN} = I$. Primijetili smo da se CNOT vrata mogu smatrati jednim tipom generaliziranih XOR vrata, no to nije slučaj za ostala klasična vrata poput NAND ili regularnih XOR vrata, u smislu da se smatraju unitarnim vratima.. Razlog tome je što XOR i NAND vrata nisu reverzibilna. Na primjer, obzirom na izlaz $A \oplus B$ XOR vrata, nije moguće odrediti što su bili ulazi A i B; postoji jedan nepovratni gubitak informacija povezan s ire-



Slika 1.4: Lijeva strana predstavlja klasične sklopove, dok desna strana predstavlja kvantni analogon - kontrolirana CNOT vrata [9]

verzibilnošću vrata. S druge strane, unitarna kvantna vrata su uvijek reverzibilna, budući je inverz unitarne matrice također unitarna matrica. Naravno, postoji mnogo zanimljivih kvantnih vrata osim kontroliranih. Međutim, u određenom smislu kontrolirana NOT vrata i jednokubitna vrata su prototipovi za sva ostala vrata zbog sljedećeg izvanrednog rezultata univerzalnosti:

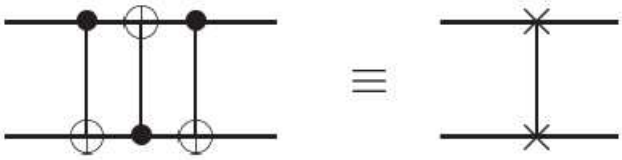
Bilo koja kvantna logička vrata s više kubita mogu biti predstavljena s CNOT i jednokubitnim vratima [9, p. 22]

Kvantni sklopovi

Već smo se upoznali s nekoliko jednostavnih kvantnih sklopova. Jedan jednostavni kvantni sklop prikazan je na slici 1.5. Svaka linija predstavlja žicu u kvantnom sklopu koja ne mora nužno predstavljati fizičku žicu; može predstavljati protok vremena, česticu kao što je foton koji se kreće s jedne lokacije na drugu kroz prostor i sl. Obično se pretpostavlja da je ulazno stanje sklopa osnovno stanje koje se sastoji od svih $|0\rangle$.

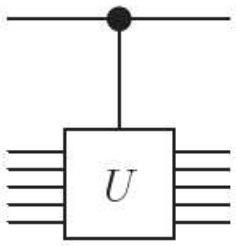
Sklop na slici 1.5 obavlja jednostavan zadatak - zamijenjuje stanja dva kubita:

$$\begin{aligned} |a,b\rangle &\longrightarrow |a, a\oplus b\rangle \\ &\longrightarrow |a\oplus(a\oplus b), a\oplus b\rangle = |b, a\oplus b\rangle \\ &\longrightarrow |b, (a\oplus b)\oplus b\rangle = |b,a\rangle \end{aligned} \tag{1.17}$$

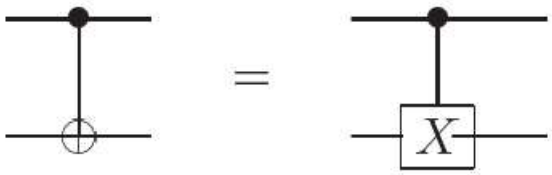


Slika 1.5: Zamjena dvaju kubita u krugu i ekvivalentna shematska oznaka [9]

Postoji nekoliko značajki dopuštenih u klasičnim sklopovima koje obično nisu prisutne kod kvantnih. Prije svega, ne dopuštamo 'petlje', odnosno povratne informacije iz jednog dijela kvantnog kruga prema drugom; kažemo da je sklop acikličan. Drugo, klasični sklopovi dopuštaju žice koje treba 'spojiti' zajedno (operacija poznata kao *FANIN*), s rezultirajućom jednom žicom koji sadrži ulaze po bitovima. Očito ova operacija nije reverzibilna pa ni unitarna, pa je ne dopuštamo u našim kvantnim krugovima. Treće, inverzna operacija, *FANOUT* , pri čemu se proizvodi nekoliko kopija bita također nije dopuštena. Zapravo, kvantna mehanika zabranjuje kopiranje kubita, čineći *FANOUT* operaciju nemogućom!



Slika 1.6: Kontrolirana U vrata [9]

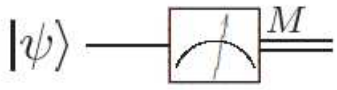


Slika 1.7: Dvije različite reprezentacije kontroliranih-NOT vrata [9]

Predstavimo još jednu značajku kvantnih sklopova koja je prikazana na slici 1.6. Pretpostavimo da je U bilo koja unitarna matrica koja djeluje na n kubita, pa se U može smatrati kvantnim vratima na tim kubitima. Tada možemo definirati kontrolirana-U vrata koja su prirodni nastavak kontroliranih-NOT vrata. Takva vrata imaju jedan kontrolni kubit, označen linijom s crnom točkom, i n ciljnih kubita, označeni okvirom U. Ako je kontrolni kubit postavljen na 0, tada se ništa ne događa ciljnim kubitima. Ako je kontrolni

kubit postavljen na 1 tada se vrata U primjenjuju na ciljne kubite. Prototipski primjer kontroliranih-U vrata su kontrolirana-NOT vrata tj. vrijedi $U = X$, kao što je prikazano na slici 1.7.

Slika 1.8 prikazuje još jednu bitnu operaciju: mjerenje. Ova operacija mijenja stanje jednog kubita $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ u probabilistički klasični bit M (razlikujemo ga od kubita crtajući ga kao duplu liniju) koji je 0 s vjerojatnošću $|\alpha|^2$ ili 1 s vjerojatnošću $|\beta|^2$.



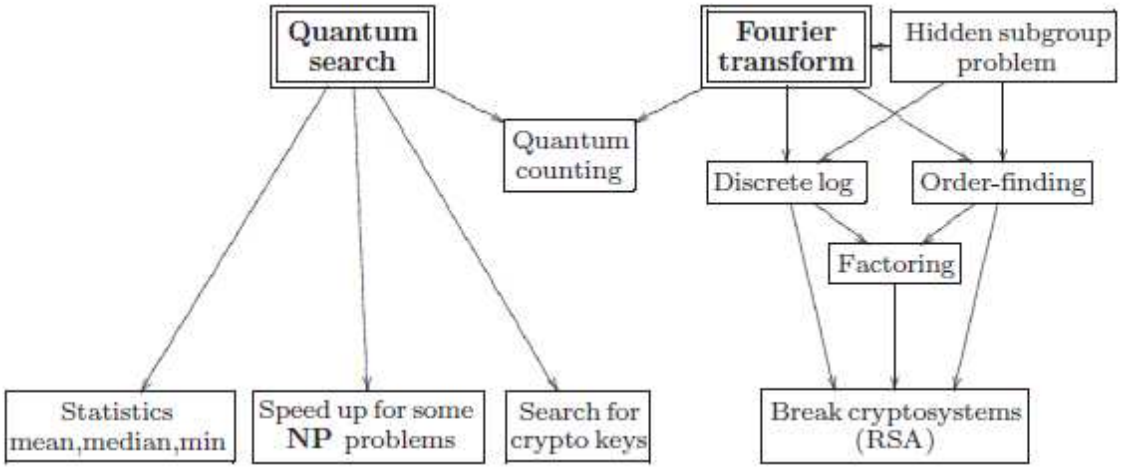
Slika 1.8: kvantni simbol za mjerenje [9]

1.4 Kvantno računanje

Kvantni algoritmi

Za sada, poznate su dvije široke klase kvantnih algoritama za efikasnije rješavanje problema koji su zahtjevni za klasična računala. Prva klasa algoritama temelji se na Shorovoj kvantnoj Fourierovoj transformaciji, a uključuje algoritme za rješavanje faktorizacije i diskretnih logaritamskih problema, pružajući zapanjujuće eksponencijalno ubrzanje u odnosu na najbolje poznate klasične algoritme. Druga klasa algoritama temelji se na Groverovom algoritmu za izvođenje kvantnog pretraživanja. Oni pružaju manje upečatljivo, ali još uvijek izvanredno kvadratno ubrzanje u odnosu na najbolje klasične algoritme. Algoritam kvantnog pretraživanja vuče svoju važnost iz raširene upotrebe tehnika temeljenih na pretraživanju klasičnih algoritama, koji u mnogim slučajevima omogućuju jednostavnu prilagodbu.

Slika 1.9 prikazuje trenutno znanje o kvantnim algoritmima. U središtu je kvantna Fourierova transformacija i kvantni algoritam pretraživanja. Kvantni algoritam brojenja je pametna kombinacija prethodna dva, koji se može koristiti za procjenu broja rješenja problema pretraživanja. Kvantni algoritam pretraživanja može se koristiti za izdvajanje statistike, kao što je minimalni element, iz neuređenog skup podataka. Također se može koristi za ubrzavanje algoritama za neke NP probleme. Kvantna Fourierova transformacija također ima mnogo zanimljivih primjena. Može se koristiti za riješavanje problema diskretnog logaritma i faktoriziracije, te pronalaženje skrivene podgrupe (generalizacija nalaženje perioda periodične funkcije). [9, p. 173]



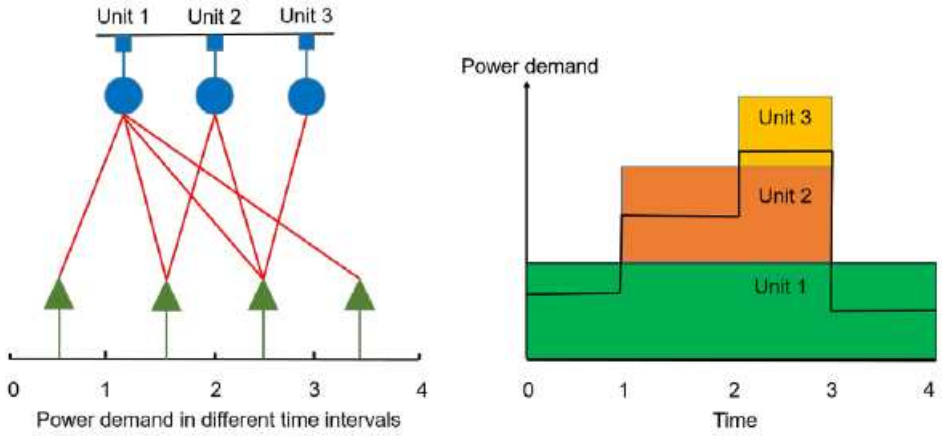
Slika 1.9: Glavni kvantni algoritmi i njihove veze [9]

Poglavlje 2

Problem opredjeljenja jedinica

Za potrebe ovog poglavlja, oslanjamo se većinom na *Unit commitment problem solution using Local Attracting Quantum PSO algorithm*. [5]

Problem opredjeljenja jedinica (*eng. unit commitment problem*) ili UC problem jedan je od najpopularnijih problema optimizacije u elektroenergetskoj industriji za rad energetskih sustava. UC problem bavi se minimiziranjem ukupnih operativnih troškova kako bi se zadovoljila procijenjena potražnja za električnom energijom u danom vremenskom roku, uz brojna ograničenja sustava i generatora. Slika 2.1 jasno ilustrira usklađivanje kombinacije jedinica s različitim opterećenjima ili zahtjevima za snagom u danom vremenskom intervalu. Dakle, procedura u kojoj su neki generatori u stanju uključeno - "ON", a neki u stanju isključeno - "OFF" može se definirati kao problem opredjeljenja jedinice.



Slika 2.1: Problem opredjeljenja s različitim opterećenjima/zahtjevima za snagom u svakom vremenskom intervalu [1]

UC problem je općenito formuliran kao veliki mješoviti cjelobrojni nelinearni problem i njegovo rješavanje je vrlo teško zbog nelinearne funkcije troškova i kombinatorne prirode skupa izvedivih rješenja. Također, dokazano je da UC nije samo NP-težak nego i NP-potpun. Za sada nije poznato polinomno rješenje UC problema.

Dva su glavna ograničenja:

1. Prva vrsta ograničenja odnosi se na elektroenergetski sustav pod kojima podrazumijevamo ograničenja prijenosa i ograničenja rezerve snage (rezervna snaga potrebna je u slučaju povećanja potražnje ili isključenja generatora iz elektroenergetskog sustava)

2. Druga vrsta ograničenja odnosi se na generatore, te uključuje ograničenje povećanja, te minimalno vrijeme uspona i zastoja

Predložene su različite tehnike za rješavanje UC problema na temelju determinističkog pristupa, metaheurističkih metoda i kombinatornih pristupa. Učinkovite tehnike optimizacije za rješavanje naprednih UC modela kada su visokoobnovljivi izvori energije integrirani u elektroenergetske sustave su od ključne važnosti. Uz sve veći broj izvora energije, problem UC-a predstavlja težak izazov zbog čega je važno razviti učinkovite metodologije za rješavanje ovog problema.

Klasične metode za rješavanje danog problema:

1. Dynamic Programming (DP)

2. Branch and Bound (BB)

3. Lagrange Relaxation (LR)

4. Priority List

5. Mixed Integer Programming (MIP)

6. Interior Point Method (IPM)

Performanse ovih metoda su prihvatljive jer su točne i jednostavne, ali samo za energetske sustave ”male” veličine.

Nedavno su umjesto klasičnih metoda korišteni meta-heuristički algoritmi jer je elektroenergetski sustav postao veći i složeniji. Primjeri meta-heurističkih metoda su:

1. Genetic Algorithm (GA)

2. Simulated Annealing (SA)
3. Ant Colony Search Algorithm (ACSA)

Istraživači su počeli poboljšavati metode optimizacije spajanjem dviju metoda zajedno jer individualni algoritmi optimizacije mogu pronaći rješenje u lokalnom minimumu:

1. Quantum Inspired Binary PSO Algorithm (QBPSO)
2. Improved Lagrangian Relaxation with Cuckoo Search Algorithm (LR-CSA)
3. Neural-Based Tabu Search (NBTS)
4. Hybrid PSO
5. Grey Wolf Optimizer (PSO-GWO)
6. Quasi-Oppositional Teaching Learning Based Algorithm (QOTLBO)

PSO algoritam ima brzu stopu konvergencije i dobro radi za mnoge složene probleme. Međutim, globalna konvergencija nije zajamčena zbog fiksne putanje i ograničene brzine svake čestice. Za rješavanje ovog nedostatka, kvantno računarstvo je uvedeno u PSO posljednjih godina, a dana istraživanja mogu se klasificirati u dva polja:

1. optimizacija rojem čestica s kvantnim ponašanjem (QPSO) - potječe iz kvantnog delta modela za PSO
2. kvantno inspirirana optimizacija rojem čestica (QIPSO) - temeljii se na QEA algoritmu, a evolucijske jednadžbe PSO algoritma koriste se za ažuriranje čestica

Kako bi se poboljšali prethodni algoritmi, uveden je novi algoritam za optimizaciju rojem kvantnih čestica s lokalnim privlačenjem - LAQPSO, nastao kombinacijom QEA i PSO algoritama. Predloženi algoritam koristi novi pristup problemu i uvodi pojmove kvantnog kuta i lokalnog atraktora koji se predlaže za definiranje rotacijskog kuta kvantnih rotacijskih vrata.[2]

U nastavku, proučit ćemo sve algoritme koji su doveli do nastanka LAQPSO algoritma.

2.1 Matematička formulacija UC problema

Cilj rješavanja problema je minimiziranje ukupnih troškova rada tijekom određenog vremenskog razdoblja, pri čemu su sva ograničenja zadovoljena. U tu svrhu mora se razviti troškovna funkcija koja se definira kao zbroj troškova goriva, troškova pokretanja i troškova

isključivanja svih generatora prema stanju uključenog ili isključenog generatora.

Ukupni trošak operacije može se izraziti na sljedeći način:

$$C_{ukupno} = \sum_{k=1}^{T} \sum_{g=1}^{N} \Big[f_{gk}(P_{gk}) + STC_{gk}(1 - U_{g(k-1)}) + SDC_{gk}(1 - U_{g(k+1)}) \Big] U_{gk} \tag{2.1}$$

gdje je:

- T vremenski horizont
- k indeks vremena
- N broj generatora
- g indeks generatora
- f_{gk} funkcija troška goriva
- U_{gk} stanje generatora g čija vrijednost može biti 0 ili 1 za sat k
- P_{gk} isporučena snaga iz generatora g po satu k
- STC_{gk} trošak potreban za pokretanje generatora g po satu k (*eng. start up cost*)
- SDC_{gk} trošak gašenja generatora g po satu k (*eng. shut down cost*)

Funkcija troška goriva kvadratna je funkcija i može se prikazati na sljedeći način:

$$f_{gk}(P_{gk}) = c_g(P_{gk})^2 + b_g(P_{gk}) + a_g \tag{2.2}$$

gdje su a_g, b_g i c_g koeficijenti troškova goriva. [5, p. 2]

Dakle, za svaki generator g, imamo T bitova u kojima su pohranjene informacije o stanju generatora - je li upaljen ili ugašen u danom trenutku. Također, primijetimo da je veličina prostora pretraživanja $T \times N$.

Tlak i temperatura toplinske jedinice moraju se podići na određenu razinu kako bi jedinica mogla biti na mreži nakon što je bila izvan upotrebe (*eng. uncommited*). Inicijalna energija potrebna za novo pokretanje stroja poznata je kao trošak pokretanja i ovisi o temperaturi ili o vremenu tijekom kojeg je generator bio izvan uporabe u elektroenergetskom sustavu.

Trošak pokretanja može se okarakterizirati sljedećom jednadžbom:

$$STC_{gk} = \begin{cases} HSC_g & \text{ako } MDT_g \leq T_g^{off} \leq MDT_g + CSH_g \\ CSC_g & \text{ako } T_g^{off} > MDT_g + CSH_g \end{cases} \tag{2.3}$$

gdje su (HSC_g, CSC_g) troškovi toplog i hladnog pokretanja generatora g, CSH_g je hladno vrijeme pokretanja generatora g, MUT_g je minimalno vrijeme tijekom kojeg generator g mora biti uključen (u stanju "ON"), MDT_g je minimalno vrijeme zastoja generatora g što je zapravo vrijeme tijekom kojeg generator mora biti isključen. T_g^{off} je vrijeme tijekom kojeg je jedinica g kontinuirano isključena (u stanju "OFF").

Funkcija opredjeljnja jedinice mora zadovoljiti skup ograničenja, a ta ograničenja mogu se podijeliti u dvije kateogije:

1. Ograničenja sustava poznatija kao ograničenja povezivanja. Svi generatori koji su u uključenom stanju "ON" i priključeni na elektroenergetski sustav, upravljani su ograničenjima spajanja zanemarujući učinkovitost ili starost generatora i uzimajući u obzir sljedeće:

   a) Zbroj snage koju isporučuju generatori jednak je zahtjevnoj snazi D_k (po satu)

   $$\sum_{g=1}^{N} P_{gk} U_{gk} = D_k \tag{2.4}$$

   b) Ograničenje rotirajuće rezerve primjenjuje se kako bi se riješila dodatna promjena u potražnji snage u situaciji isključenja generatora ili dalekovoda koji nije u funkciji, stoga se rezerva snage koristi za kompenzaciju manjka dovedene energije:

   $$\sum_{g=1}^{N} P_g^{max} U_{gk} \geq D_k + R_k \tag{2.5}$$

   gdje je D_k zahtjev za opterećenjem sustava u satu k (zahtjevna snaga), a R_k je rezerva okretanja elektroenergetskog sustava u istom satu.

2. Ograničenja generatora također poznata kao lokalna ograničenja jer su povezana s generatorom opisana su na sljedeći način:

   a) Generator ima gornje i donje granice dok proizvodi snagu i te granice se ne mogu prekoračiti

   $$P_g^{max} \geq P_{gk} \geq P_g^{min} \tag{2.6}$$

gdje su (P_g^{max}, P_g^{min}) maksimalna i minimalna snaga koja se može napajati iz jedinice g.

b) Minimalno vrijeme neprekidnog rada poznato je kao minimalno vrijeme rada (MUT)

$$T_g^{on} \geq MUT_g \tag{2.7}$$

gdje je T_g^{on} kontinuirano vrijeme tijekom kojeg je generator g uključen ili u stanju "ON".

c) Generator mora biti isključen ili u stanju "OFF" za vrijeme koje je jednako minimalnom vremenu zastoja MDT

$$T_g^{off} \geq MDT_g \tag{2.8}$$

2.2 Algoritam optimizacije rojem čestica - PSO

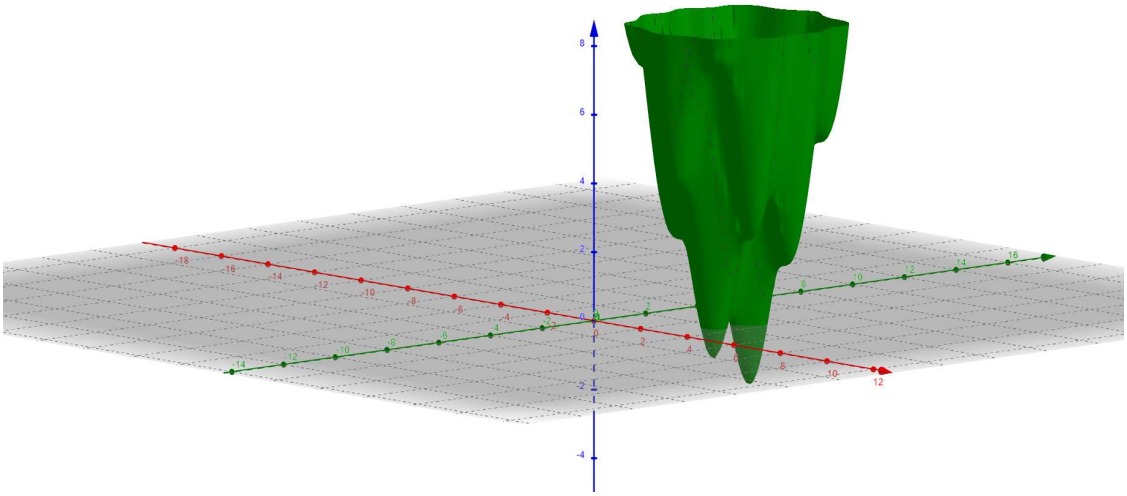
Optimizacija rojem čestica (PSO) (*eng. particle swarm optimization*) je bio-inspiriran algoritam koji je jednostavan za traženje optimalnog rješenja problema, te je jedan je od najčešće korištenih algoritama za rješavanje UC problema. Razlikuje se od ostalih algoritama optimizacije na način da je potrebna samo funkcija cilja, ne ovisi o gradijentu, te ima vrlo malo hiperparametara. U ovom radu, PSO algoritam će poslužiti kao osnovni algoritam kojeg ćemo nadograđivati kako bi opisali BPSO algoritam u 2.3, te QBPSO algoritam u 3.2.

Slika 2.2: Ilustracija algoritma optimizacije rojem čestica preko jata ptica [3]

Optimizaciju rojem čestica predložili su Kennedy i Eberhart 1995. Kao što je spomenuto u izvornom radu, sociobiolozi vjeruju da jato riba ili jato ptica koje se kreću u skupini "mogu profitirati od iskustva svih članova". Drugim riječima, dok ptica leti i nasumično traži hranu, sve ptice u jatu mogu podijeliti svoja otkrića i pomoći cijelom jatu da dobije najbolji lov. Budući možemo simulirati kretanje jata ptica, također možemo zamisliti da nam svaka ptica pomaže pronaći optimalno rješenje u visokodimenzionalnom prostoru rješenja, a najbolje rješenje koje pronađe jato je najbolje rješenje u prostoru rješenja. Ovo je heurističko rješenje jer nikada ne možemo dokazati da se pravo globalno optimalno rješenje može pronaći. Međutim, često nalazimo da je rješenje koje je pronašao PSO prilično blizu globalnom optimalnom rješenju. [10]

U praksi, PSO se najčešće koristi za pronalaženje maksimuma ili minimuma funkcije definirane u višedimenzionalnom vektorskom prostoru. Promotrimo jedan jednostavan primjer.

$$f(x,y) = (x-3.14)^2 + (y-2.72)^2 + \sin(3x+1.41) + \sin(4y-1.73) \tag{2.9}$$



Slika 2.3: Grafički prikaz funkcije 2.9

Kao što možemo vidjeti iz slike 2.4, ova funkcija izgleda kao zakrivljeni karton za jaja. To nije konveksna funkcija i stoga je teško pronaći njezin minimum jer pronađeni lokalni minimum nije nužno globalni minimum. Dakle, kako možemo pronaći minimalnu vrijednost ove funkcije? Zasigurno možemo pribjeći iscrpnom pretraživanju: ako provjerimo vrijednost za svaku točku na ravnini možemo pronaći minimalnu točku. Ili možemo samo nasumično pronaći neke točke na ravnini i vidjeti koja daje najnižu vrijednost ako

smatramo da je preskupo pretraživati svaku točku. No, ako smo pronašli točku s manjom vrijednošću, lako je pronaći još manju vrijednost oko njene blizine.

Upravo ovako funkcionira optimizacija roja čestica. Slično jatu ptica koja traže hranu, započinjemo s nizom nasumičnih točaka u ravnini (nazovimo ih česticama (*eng. particles*)) i pustimo ih da traže točku u kojoj funkcija postiže minimum u nasumičnim smjerovima. Svaka čestica okarakterizirana je vektorom položaja i vektorom brzine. U svakom koraku, svaka čestica računa udaljenost do točke u kojoj funkcija postiže minimum za tu točku, kao i udaljenost do točke u kojoj funkcija postiže minimum za cijeli roj čestica. Nakon određenog broja iteracija, minimalnu točku funkcije smatramo minimalnom točkom koju je ikada pronašao ovaj roj čestica.

Detalji algoritma

Pretpostavimo da imamo k čestica u n-dimenzionalnom prostoru pretraživanja. Označimo poziciju i brzinu čestice r kao $X_r = \{X_{r1}, X_{r2}, \cdots, X_{rn}\}$, $V_r = \{V_{r1}, V_{r2}, \cdots, V_{rn}\}$, redom.

Brzina i položaj čestica mogu se ažurirati pomoću sljedećih jednadžbi: [5, p. 4]

$$V_r^{m+1} = \omega V_r^m + c_1\varphi_1(Pbest_r^m - X_r^m) + c_2\varphi_2(Gbest^m - X_r^m) \tag{2.10}$$

$$X_r^{m+1} = X_r^m + V_r^{m+1} \tag{2.11}$$

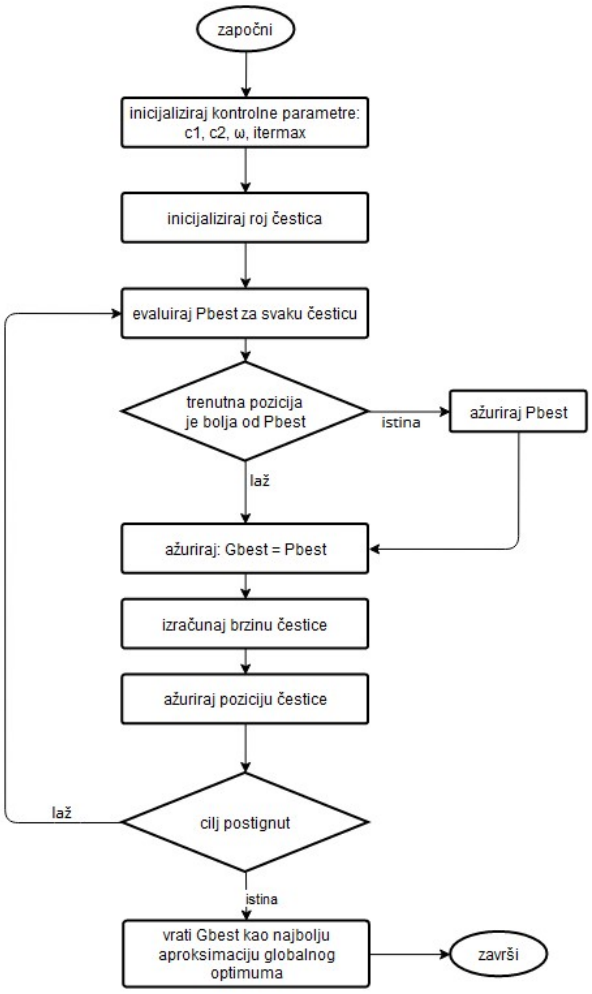
gdje:

- (X_r^m, V_r^{m+1}) predstavljaju poziciju i brzinu čestice r u trenutnoj iteraciji m
- (c_1, c_2) predstavljaju faktore ubrzanja, gdje je c_1 kognitivni parametar koji predstavlja povjerenje koje čestica ima u sebe, a c_2 je društveni parametar koji predstavlja koliko povjerenja čestica ima u svoje susjede. Najčešće se uzima $c_{1,max} = c_{2,max} = 2.5$, te $c_{1,min} = c_{2,min} = 0.5$ [11]
- (φ_1, φ_2) predstavljaju dva nasumično odabrana, uniformna broja između 0 i 1
- $Pbest_r^m = \{Pbest_{r1}^m, Pbest_{r2}^m, \cdots, Pbest_{rn}^m\}$ predstavlja vektor najboljeg osobnog iskustva čestice u kojem je zabilježena najbolja prethodna pozicija čestice u trenutnoj iteraciji m
- $Gbest^m = \{Gbest_1^m, Gbest_2^m, \cdots, Gbest_n^m\}$ predstavlja vektor globalnog najboljeg iskustva u kojem je zabilježen najbolji položaj svih čestica u trenutnoj iteraciji m

- ω predstavlja faktor inercije koji kontrolira kretanje čestica tj. određuje koliko bi čestica trebala zadržati prijašnju brzinu. ω se može uzeti kao konstantan broj ili predstaviti kao u sljedećoj jednadžbi:

$$\omega = \omega_{max} - \frac{(\omega_{max} - \omega_{min})}{iter_{max}} \times m \tag{2.12}$$

gdje je $iter_{max}$ maksimalan broj iteracija, a $(\omega_{max}, \omega_{min})$ su maksimalna i minimalna vrijednost faktora inercije. Obično se za $(\omega_{max}, \omega_{min})$ uzima 0.9 i 0.4, redom.



Slika 2.4: Algoritam optimizacije rojem čestica [7]

Vrijednosti (c_1, c_2) i ω zadaju se na početku algoritma, dok se (φ_1, φ_2) nasumično odabiru prilikom svakog računanja brzine i pozicije čestice. Roj čestica incijalizira se na način

da se dodijele random pozicije svim česticama. Primijetimo da su $Pbest_r^m$ i X_r^m dva vektora pozicije, te je njihova razlika $Pbest_r^m - X_r^m$ vektor razlike. Dodavajući ovaj izraz originalnom vektoru brzine V_r^{m+1} zapravo vraćamo česticu nazad na poziciju $Pbest_r^m$. Slično primjećujemo za razliku $Gbest^m - X_r^m$.

Jedno zanimljivo svojstvo ovog algoritma koje ga razlikuje od ostalih optimizacijskih algoritama je da ne ovisi o gradijentu ciljne funkcije. U gradijentnom spuštanju, na primjer, tražimo minimum funkcije $f(X)$ pomicanjem X u smjeru $-\Delta f(X)$ gdje funkcija najbrže pada. Za bilo koju česticu na trenutnoj poziciji X, njezino kretanje ne ovisi o tome u kojem je smjeru "nizbrdo", već samo o tome gdje su *Pbest* i *Gbest*. To čini PSO posebno prikladnim ukoliko je diferenciranje $f(X)$ teško. Još jedno svojstvo PSO-a je da se može lako paralelizirati. Kada manipuliramo s više čestica kako bismo pronašli optimalno rješenje, svaka se čestica može ažurirati paralelno i samo trebamo prikupiti ažuriranu vrijednost jednom po iteraciji. [10]

2.3 Binarni algoritam optimizacije rojem čestica - BPSO

Budući se glavni Problem opredjeljenja jedinica svodi na određivanje kada će neki generator biti uključen (u stanju 1), a kada isključen (u stanju 0), uvodi se binarna verzija PSO algoritma - Binary PSO kojeg možemo koristiti u diskretnim prostorima pretraživanja. S binarnom verzijom PSO algoritma možemo na najjednostavniji način prikazati kombinaciju rada generatora u određenom vremenskom periodu. Binarni PSO algoritam ćemo koristiti u 3.2 kada ćemo opisati vjerojatnosni BPSO algoritam inspiriran kvantnim računanjem - QBPSO. Također, tada definirani QBPSO algoritam ćemo koristiti kao osnovu za definiranje LAQPSO algoritma u Poglavlju 4.

Razlika između PSO i BPSO pristupa je ta što je u BPSO algoritmu, vektor položaja $X_r = \{X_{r1}, X_{r2}, \cdots, X_{rn}\}$ binarni, odnosno svaka komponenta vektora X_r može poprimiti vrijednost 0 ili 1. Stoga, uvodimo dodatnu varijablu *Sigmoid Limiting Transformation (SLT)* na sljedeći način:

$$S(V_{ri}^{m+1}) = \frac{1}{1 + exp(V_{ri}^{m+1})} \tag{2.13}$$

gdje je $S(V_{ri}^{m+1})$ sigma funkcija i-tog elementa čestice r, te se V_{ri}^{m+1} računa prema 2.10.

Vrijednost $S(V_{ri}^{m+1})$ interpretira se kao granični parametar pomoću kojeg određujemo poziciju svake komponente X_{ri} čestice r.

Pozicija čestice se tada može ažurirati na sljedeći način: [5, p. 4]

$$X_{ri}^{m+1} = \begin{cases} 1 & \text{ako } H_{ri} < S(V_{ri}^{m+1}) \\ 0 & \text{inače} \end{cases} \tag{2.14}$$

gdje je $r = 1, 2, \cdots, k$, $i = 1, 2, \cdots, n$, H_{ri} nasumično uniforman odabran broj, između $[0, 1]$. Ako je H_{ri} manji od $S(V_{ri}^{m+1})$, tada je pozicija i-tog elementa čestice r u iteraciji $m + 1$ jednaka 1, a inače 0. [5, p. 4]

Možemo zaključiti da se u BPSO inačici mijenja samo funkcija za određivanje položaja pojedine X_{ri} komponente čestice r, dok sve ostalo (računanje brzine V_{ri} čestice r, faktori ubrzanja (c_1, c_2), nasumično odabrani parametri (φ_1, φ_2), vektor najboljeg osobnog iskustva $Pbest_r^m$, vektor najboljeg globalnog iskustva $Gbest^m$, faktor inercije ω) ostaje isto kao u 2.10 i 2.11 PSO algoritma.

Poglavlje 3

Kvantna algoritamska hibridizacija

3.1 Kvantno inspirirani evolucijski algoritam - QEA

Kako bi objasnili evolucijski algoritam koristimo se člankom *Quantum-Inspired Evolutionary Algorithm for a Class of Combinatorial Optimization*. [6]

Evolucijski algoritam inspiriran kvantnim računanjem bit će nam potreban u Poglavlju 4 kako bismo uspješno opisali LAQPSO algoritam.

Evolucijski algoritmi (EA) su uglavnom stohastičke metode pretraživanja i optimizacije koje su inspirirane principima prirodne biološke evolucije. U usporedbi s tradicionalnim metodama optimizacije, kao što su strategije temeljene na proračunu i enumerativne strategije, EA su robusni, globalni i općenito se mogu primijeniti bez pribjegavanja heuristikama specifičnim za domenu, iako te heuristike utječu na performanse. Tri glavne metode evolucijskog računanja koje su uspostavljene tijekom posljednjih 45 godina su genetski algoritmi (GA), evolucijsko programiranje (EP), te strategije evolucije (ES).

EA rade na skupu potencijalnih rješenja, primjenjujući princip preživljavanja najsposobnijih kako bi proizveli bolje aproksimacije. Na svakoj generaciji EA stvara se novi skup aproksimacija postupkom odabira pojedinaca prema njihovoj razini sposobnosti u domeni problema i njihove reprodukcije koja je implementirana pomoću operatora varijacije. Pod operatorom varijacije porazumijevamo proces kreiranja novih pojedinaca iz starih. Ovaj proces može dovesti do evolucije populacije jedinki (skupa mogućih rješenja) koje su bolje prilagođene svojoj okolini od jedinki od kojih su stvorene, baš kao u prirodnoj prilagodbi.

Što karakterizira evolucijske algoritme?

- Reprezentacija pojedinca elementom iz skupa potencijalnih rješenja

- Funkcija evaluacije (ili *fitness* funkcija) je funkcija na skupu mogućih rješenja koja opisuje koliko je neko rješenje dobro
- Dinamika populacije predstavlja metodu koja omogućava da iz trenutnog skupa potencijalnih rješenja konstruiramo novi skup potencijalnih rješenja

Kako bismo imali dobru ravnotežu između istraživanja prostora rješenja i primjene funkcije evaluacije na pronađene vrijednosti, prethodne 3 komponente treba pravilno definirati.

U radu [6] predložen je novi evolucijski algoritam, nazvan kvantno inspiriran evolucijski algoritam (QEA), koji se temelji na konceptu i principima kvantnog računanja kao što su kvantni bit i superpozicija stanja s kojima smo se već upoznali u Poglavlju 1. Kao i EA, QEA je također karakteriziran odabirom (reprezentacijom) pojedinca, evaluacijskom funkcijom i dinamikom populacije. Međutim, umjesto binarnog, numeričkog ili simboličkog prikaza, QEA koristi novu oznaku za vjerojatnostni prikaz koja se temelji na konceptu kubita, te uvodi pojam kubit populacije i kubit pojedinca kao niza kubita. Treba napomenuti da iako se QEA temelji na konceptu kvantnog računanja, QEA nije kvantni algoritam, već novi evolucijski algoritam za klasično računalo.

Definicija 3.1. *Kubit je najmanja jedinica informacije u QEA, koja je definirana parom brojeva* (α, β) *kao*

$$q = [\alpha, \beta]^T = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{3.1}$$

gdje su $\alpha, \beta \in \mathbb{R}$*, te vrijedi* $|\alpha|^2 + |\beta|^2 = 1$*.* $|\alpha|^2$ *je vjerojatnost da će se kubit naći u stanju* 0*,* a $|\beta|^2$ *je vjerojatnost da će se kubit naći u stanju* 1*. Kubit može biti u stanju* 1*, u stanju* 0 *ili u bilo kojoj superpoziciji ova dva.*

Definicija 3.2. *Populaciju kubit pojedinaca u generaciji m definiramo kao:*

$$Q(m) = [q_1^m, q_2^m, \cdots, q_k^m]$$

gdje je q_r^m *kubit pojedinac definiran na sljedeći način:*

$$q_r^m = [q_{r1}^m, q_{r2}^m, \cdots, q_{rn}^m] = \left[\begin{array}{c|c|c|c} \alpha_{r1}^m & \alpha_{r2}^m & \cdots & \alpha_{rn}^m \\ \beta_{r1}^m & \beta_{r2}^m & \cdots & \beta_{rn}^m \end{array}\right]$$

gdje je $|\alpha_{rn}|^2 + |\beta_{rn}|^2 = 1$*,* $r = 1, 2, \ldots, k$*, a* n *je broj kubita.*

Kažemo da QEA održava populaciju kubit pojedinaca $Q(m)$ na generaciji m, gdje je k veličina populacije.

Kao što smo objasnili u Poglavlju 1.3, djelovanje kvantnih vrata mijenja stanje kubita. Kvantna vrata su reverzibilna vrata i mogu se predstaviti kao unitarni operator U koji djeluje na bazi kubitnih stanja te zadovoljava $U^{\dagger}U = UU^{\dagger}$. Također smo se upoznali s nekoliko primjera nekoliko kvantnih vrata koje ćemo sad proširiti s još jednim primjerom kojeg uvodi QEA: Q-vrata.

Definicija 3.3. *Q-vrata su rotacijska vrata pomoću kojih ažuriramo (rotiramo) kubit. Ažurirani kubit treba zadovoljiti uvjet normalizacije* $|\alpha'_i|^2 + |\beta'_i|^2 = 1$, *gdje su* α' *i* β' *vrijednosti ažuriranog kubita. Q-vrata definiramo kao:*

$$U(\Delta\theta^m_{ri}) = \begin{bmatrix} \cos\left(\Delta\theta^m_{ri}\right) & -\sin\left(\Delta\theta^m_{ri}\right) \\ \sin\left(\Delta\theta^m_{ri}\right) & \cos\left(\Delta\theta^m_{ri}\right) \end{bmatrix} \tag{3.2}$$

gdje je $\Delta\theta^m_{ri}$, $i = 1, 2, \cdots, n$, *kut rotacije i-tog kubita čestice r. Q-vrata se još nazivaju varijacijski operator.*

Q-vrata se koriste kako bismo dobili evoluciju populacije. Kut $\Delta\theta_{ri}$ definira se ovisno o specifičnom problemu. U nastavku, dan je primjer definicije $\Delta\theta_{ri}$ kuta za problem ruksaka.

Napomena 3.4. *Uloga varijacijskog operatora je kreiranje novih pojedinaca iz starih.*

Napomena 3.5. *NOT vrata, kontrolirana NOT vrata ili Hadamardova vrata iz Poglavlja 1.3 mogu se koristiti kao Q-vrata.*

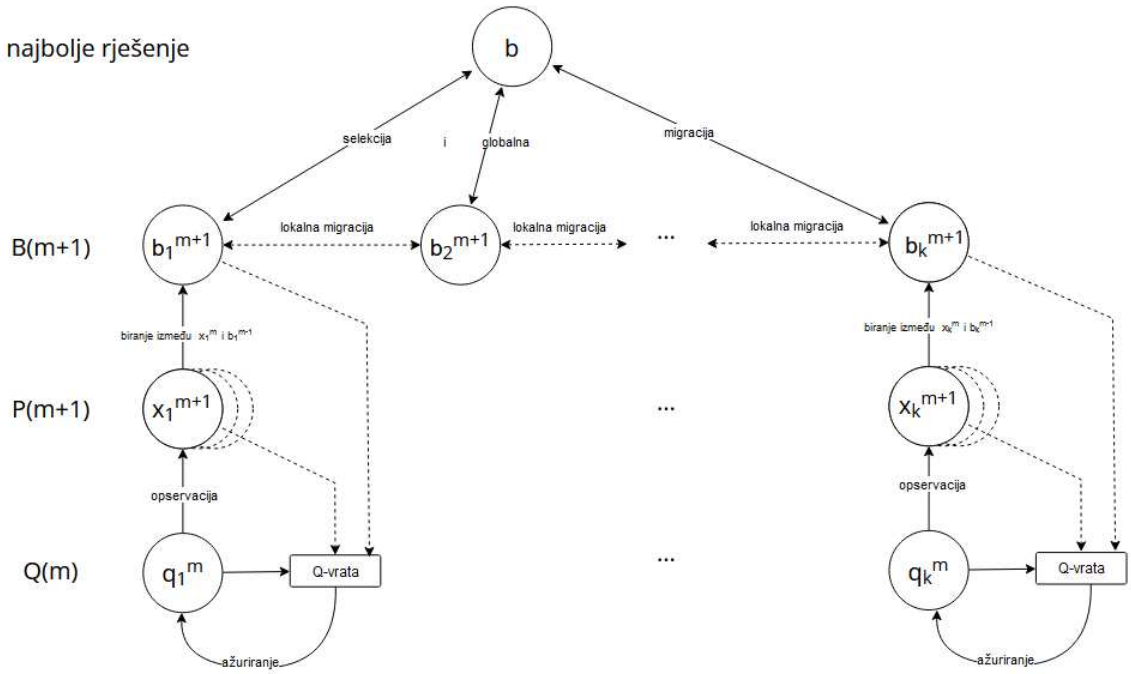
Detalji algoritma

Proučimo korake algoritma:

1. U prvom koraku inicijaliziramo početnu populaciju kubita, $Q(0)$ - skup potencijalnih rješenja. To znači da za svaku česticu q^0_r, $r = 1, 2, \cdots, k$ inicijaliziramo α^0_{ri} i β^0_{ri}, $i = 1, 2, \cdots, n$ na $\frac{1}{\sqrt{2}}$.

2. Sa $P(m) = \{x^m_1, x^m_2, \cdots, x^m_k\}$ smo označili binarni skup potencijalnih rješenja. Za trenutnu generaciju $m = 0$ generiramo binarna rješenja u $P(0) = \{x^0_1, x^0_2, \cdots, x^0_k\}$ promatrajući stanja $Q(0)$. To znači da za svaki kubit pojedinac q^0_r zapravo generiramo potencijalno binarno rješenje. Jedno binarno rješenje $x^0_r = \{x^0_{r1}, x^0_{r2}, \cdots, x^0_{rn}\}$, $r = 1, 2, \cdots, k$ je binarni niz duljine n gdje je svaka komponenta 0 ili 1. Vrijednost pojedine komponente odabire se na način da se za svaki bit u binarnom nizu x^0_r, $r = 1, 2, \cdots, k$ u $P(0)$, generira random broj r iz intervala $[0, 1]$: ako je $r < |\beta_{ri}|^2$, tada je $x^0_{ri} = 1$, a inače je $x^0_{ri} = 0$.

3. Primjenjujemo *fitness* funkciju na svako potencijalno binarno rješenje x^0_r iz $P(0)$.



Slika 3.1: Cjelokupna struktura QEA algoritma [6]

4. Sa $B(m) = \{b_1^m, b_2^m, \cdots, b_k^m\}$ smo označili trenutna najbolja binarna rješenja svake čestice. U početnoj generaciji $m = 0$ vrijedi $b_r^0 = x_r^0, \forall r = 1, 2, \cdots, k$, odnosno $P(0) = B(0)$.

5. U while petlji, binarna rješenja u $P(m)$ generiraju se promatrajući stanja $Q(m-1)$ kao u koraku 2).

6. Primjenjujemo *fitness* funkciju na svako binarno rješenje x_r^m iz $P(m)$.

7. Kubit pojedinci iz populacije $Q(m)$ ažuriraju se primjenom odgovarajućih Q-vrata. Kut θ_{ri}^m definira se slično kao u 3.1. Uspoređujemo nova generirana rješenja x_r^m s trenutnim najboljim rješenjima b_r^m. Ako se potencijalno binarno rješenje x_{ri}^m pojedine čestice "udaljava" od trenutnog najboljeg binarnog rješenja b_{ri}^m, odaberemo kut tako da se približimo b_{ri}^m. Npr. ako je $x_{ri}^m = 0, b_{ri}^m = 1$, želimo da se 0 približi 1. S ovim kutem, ažuriramo vjerojatnosti kubita - želimo postići veću vjerojatnost da se čestica približi najboljem rješenju.

8. Uspoređujemo odgovarajuća binarna rješenja iz $P(m)$ i $B(m-1)$ s $B(m)$. Ako se u $P(m)$ ili $B(m-1)$ nalazi bolje rješenje za danu česticu, spremamo ga u $B(m)$ na odgovarajuće mjesto; inače ne radimo ništa.

9. Sa b smo označili varijablu u kojoj je spremljeno trenutno najbolje rješenje. Ako je novo najbolje rješenje b_r^m iz $B(m)$ bolje od trenutnog najboljeg rješenja b, spremljeno rješenje b zamjenjuje se novim najboljim rješenjem b_r^m.

10. Ako je zadovoljen uvjet migracije, najbolje rješenje b se globno "migrira" ka $B(m)$ ili se najbolje rješenje b_r^m unutar rješenja u $B(m)$ lokalno migrira ka $B(m)$.

Definicija 3.6. *Globalna migracija implementirana je zamjenom svih rješenja u $B(m)$ s b, a lokalna migracija implementirana je zamjenom nekih rješenja iz $B(m)$ s najboljim rješenjem iz $B(m)$. Uvjet migracije je dizajn parametar, a proces migracije može potaknuti varijacije vjerojatnosti kubit pojedinca.*

Binarna rješenja u $P(m)$ odbacuju se na kraju petlje jer će $P(m+1)$ nastati promatranjem $Q(m)$ u koraku 7). Sve dok se ne ispuni uvjet prekida, QEA nastavlja s radom.

Algoritam 1 QEA procedura

begin
$m \leftarrow 0$

1. inicijaliziraj početnu populaciju $Q(0)$ prema 1
2. generiraj binarna potencijalna rješenja $P(0)$ promatrajući stanja $Q(0)$ prema 2
3. evaluiraj $P(0)$
4. spremi binarna potencijalna rješenja $P(0)$ u trenutna najbolja binarna rješenja $B(0)$
   while (**not** uvjet prekida) **do**
   begin
   $m \leftarrow m + 1$
5. generiraj $P(m)$ promatrajući stanja $Q(m-1)$ prema 2
6. evaluiraj $P(m)$
7. ažuriraj kubit pojedince iz populacije $Q(m)$ koristeći Q-vrata iz 3.3
8. usporedi rješenja iz $B(m-1)$ i $P(m)$, te pohrani bolje rješenje u $B(m)$
9. usporedi novo najbolje rješenje b_r^m iz $B(m)$ s trenutnim najboljim rješenjem b;
   ako je bolje, zamijeni b s b_r^m
10. **if** (uvjet migracije)
    then migriraj b ili b_r^m ka $B(m)$ globalno ili lokalno, redom

   end

end

Primjena QEA algoritma na problem ruksaka

Primjenimo opisani algoritam na konkretan primjer - problem ruksaka. Problem ruksaka možemo opisati kao problem u kojem između ponuđenih predmeta biramo one predmete koji imaju najveću vrijednost (profit) tako da ne prijeđemo unaprijed definiran kapacitet ruksaka.

Pretpostavimo da imamo skup od n predmeta i ruksak kapaciteta C. Trebamo odabrati podskup svih predmeta kako bi maksimizirali profit $f(x)$:

$$f(x) = \sum_{i=1}^{n} p_i x_i \quad \text{tako da} \quad \sum_{i=1}^{n} w_i x_i \leq C \tag{3.3}$$

gdje je:

- $X = (x_1, x_2, \cdots, x_n)$ td. $x_i = 0$ ili $x_i = 1$, za $i = 1, 2, \cdots, n$ (ako je $x_i = 1$ tada je predmet i u ruksaku)
- p_i je vrijednost (profit) predmeta i
- w_i je težina predmeta i

QEA algoritam za problem ruksaka sastoji se od osnovne strukture za QEA uz dodanu metodu "popravi" kako bi se zadovoljio uvjet ograničenja ruksaka. Kubit duljine n predstavlja linearnu superpoziciju rješenja problema. Duljina kubit pojedinca jednaka je broju predmeta. Algoritam se može opisati na sljedeći način:

1. Korak inicijalizacije isti je kao u 1 - predmet i može biti odabran za ruksak s vjerojatnošću $|\beta_i|^2$ ili $(1 - |\alpha_i|^2)$.
2. Isto kao u 2 - za svaki bit u binarnom nizu x_i^0, $i = 1, 2, \cdots, n$ u $P(0)$, generira se random broj r iz intervala $[0, 1]$: ako je $r < |\beta_i|^2$, tada je $x_i^0 = 1$, a inače je $x_i^0 = 0$. Zbog jednostavnosti, umjesto x_i^m i q_i^m, koristimo x i q, redom.

Algoritam 2 generiraj(x)

```
begin
    i ← 0
    while (i < n) do
    begin
        i ← i + 1
        if random[0, 1] < |β_ri^2|
        then x_i ← 1
        else x_i ← 0
    end
end
```

3. Kada binarni niz prekrši ograničenje kapaciteta, odnosno ako je ruksak prepunjen, koristi se sljedeća metoda da "popravimo" stanje ruksaka:

Algoritam 3 popravi(x)

```
begin
    ruksak_prepunjen ← false
    if Σ_{i=1}^{n} w_i x_i > C
    then ruksak_prepunjen ← true
    while (ruksak_prepunjen)
    begin
        odaberi i-ti predmet iz ruksaka
        x_i ← 0
        if Σ_{i=1}^{n} w_i x_i ≤ C
        then ruksak_prepunjen ← false
    end
    while (not ruksak_prepunjen)
    begin
        odaberi j-ti predmet iz ruksaka
        x_j ← 1
        if Σ_{i=1}^{n} w_i x_i > C
        then ruksak_prepunjen ← true
    end
    end
    x_j ← 0
end
```

4. Isto kao u 3 - primjenjujemo *fitness* funkciju na svako binarno rješenje x_i iz $P(0)$

5. Isto kao u 4 - sa $B(m) = \{b_1, b_2, \cdots, b_n\}$ smo označili trenutna najbolja binarna rješenja svakog predmeta. U početnoj generaciji $m = 0$ vrijedi $b_i = x_i, \forall i = 1, 2, \cdots, n$, odnosno $P(0) = B(0)$.

6. Isto kao u 5 - binarna rješenja u $P(m)$ generiraju se promatrajući stanja $Q(m-1)$ kao u koraku 2.

7. Isto kao u 3 - "popravljamo" stanje ruksaka ako je prepunjen

8. Isto kao u 6 - primjenjujemo *fitness* funkciju na svako binarno rješenje x_i iz $P(m)$

9. Ažuriramo kubit pojedinace koristeći metodu **ažuriraj**:

Algoritam 4 ažuriraj(q)

begin
  $i \leftarrow 0$
  while ($i < n$) **do**
  begin
    $i \leftarrow i + 1$
    odredi $\Delta\theta_i$ pomoću pregledne tablice 3.1
    odredi (α'_i,β'_i) na sljedeći način:
    if (q se nalazi u 1. ili 3. kvadrantu)
    then $[\alpha'_i \ \beta'_i]^T = U(\Delta\theta_i[\alpha_i \ \beta_i]^T)$
    else $[\alpha'_i \ \beta'_i]^T = U(-\Delta\theta_i[\alpha_i \ \beta_i]^T)$

  end
  $q \leftarrow q'$
end

Za problem ruksaka, vrijednosti $\Delta\theta_i$ koje se koriste za rotacijska vrata definirane su u tablici 3.1.

x_i	b_i	$fitness(x) \geq fitness(b)$	$\Delta\theta_i$
0	0	Laž	θ_1
0	0	Istina	θ_2
0	1	Laž	θ_3
0	1	Istina	θ_4
1	0	Laž	θ_5
1	0	Istina	θ_6
1	1	Laž	θ_7
1	1	Istina	θ_8

Tablica 3.1: Pregledna tablica za odredivanje kuta rotacije

Definirajmo vektor kuta kao $\Theta = [\theta_1, \theta_2, \cdots, \theta_8]^T$. Tada možemo definirati preglednu (*eng. lookup*) tablicu s vrijednostima: $\theta_1 = 0$, $\theta_2 = 0$, $\theta_3 = 0.01\pi$, $\theta_4 = 0$, $\theta_5 = -0.01\pi$, $\theta_6 = 0$, $\theta_7 = 0$, $\theta_8 = 0$, a $B = (b_1, b_2, b_3, \cdots, b_n)$ je najbolje rješenje gdje:

- *fitness* funkcija predstavlja funkciju cilja koja kao ulaz uzima potencijalno rješenje problema, te vraća vrijednost koja nam govori koliko je to rješenje dobro.
- b_i i x_i predstavljaju $i - ti$ bit trenutnog najboljeg rješenja b i binarnog rješenja x

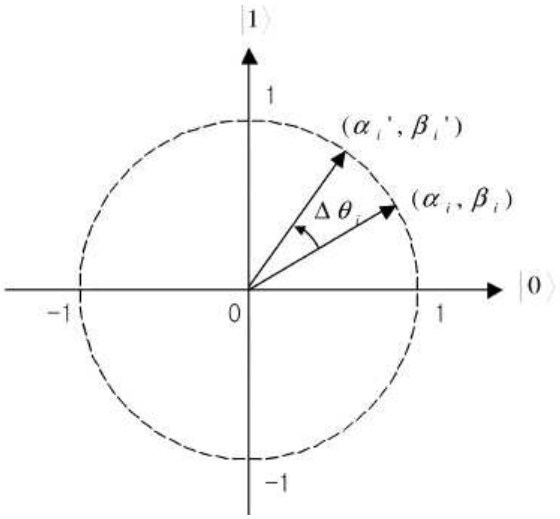
Ako su x_i i b_i 0 i 1 redom, i ako je vrijednost $fitness(x) \geq fitness(b)$ neistinita:

a) ako se kubit nalazi u 1. ili 3. kvadrantu (kao na slici 3.2), vrijednost θ_3 postavljena je na pozitivnu vrijednost kako bi se povećala vjerojatnost stanja $|1\rangle$
b) ako se kubit nalazi u 2. ili 4. kvadrantu, tada bi se -θ_3 trebala koristiti kako bi se povećala vjerojatnost stanja $|1\rangle$

Ako su npr. x_i i b_i 1 i 0 redom, i ako je vrijednost $fitness(x) \geq fitness(b)$ neistinita:

a) ako se kubit nalazi u 1. ili 3. kvadrantu (kao na slici 3.2), vrijednost θ_5 postavljena je na negativnu vrijednost kako bi se povećala vjerojatnost stanja $|0\rangle$

b) ako se kubit nalazi u 2. ili 4. kvadrantu, tada bi se -θ_5 trebala koristiti kako bi se povećala vjerojatnost stanja $|0\rangle$



Slika 3.2: Prikaz rotacijskih vrata za kubit pojedinca [6]

Ako je dvosmisleno odabrati pozitivan ili negativan broj za vrijednosti parametara kuta preporuča se da se vrijednosti postave na 0. Najčešće korištene vrijednosti su $\theta_3 = 0.01\pi$, $\theta_5 = -0.01\pi$, te 0 za ostatak. Veličina $\Delta\theta_i$ ima utjecaj na brzinu konvergencije, ali ako je prevelika, rješenja se mogu razilaziti ili se prerano približavati lokalnom optimumu. Preporučuju se vrijednosti od 0.001π do 0.05π za veličinu $\Delta\theta_i$, iako ovise o konkretnom problemu. Predznak $\Delta\theta_i$ određuje smjer konvergencije. Opravdanje za točno ovaj odabir kuta dano je u [6].

Rotacijska vrata $U(\Delta\theta_i)$ koriste se kao varijacijski operator za ažuriranje kubit pojedinca q. (α_i, β_i) i-tog kubita dobiju se prema sljedećem:

$$\begin{bmatrix} \alpha_i' \\ \beta_i' \end{bmatrix} = \begin{bmatrix} cos(\Delta\theta_i) & -sin(\Delta\theta_i) \\ sin(\Delta\theta_i) & cos(\Delta\theta_i) \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} \tag{3.4}$$

10. Isto kao u 8 - uspoređujemo odgovarajuća binarna rješenja iz $P(m)$ i $B(m-1)$ s $B(m)$. Ako se u $P(m)$ ili $B(m-1)$ nalazi bolje rješenje za dani predmet, spremamo ga u $B(m)$ na odgovarajuće mjesto; inače ne radimo ništa.

11. Isto kao u 9 - ako je novo najbolje rješenje b_i iz $B(m)$ bolje od trenutnog najboljeg rješenja b, spremljeno rješenje b zamjenjuje se novim najboljim rješenjem b_i.

12. Isto kao u 10 - ako je zadovoljen uvjet migracije, najbolje rješenje b se globno "migrira" ka $B(m)$ ili se najbolje rješenje b_i unutar rješenja u $B(m)$ lokalno migrira ka $B(m)$.

Algoritam 5 QEA procedura za Problem ruksaka

begin
$m \leftarrow 0$

1. inicijaliziraj početnu populaciju $Q(0)$
2. **generiraj** $P(0)$ promatrajući stanja $Q(0)$
3. **popravi** $P(0)$
4. evaluiraj $P(0)$
5. spremi binarna potencijalna rješenja $P(0)$ u trenutna najbolja binarna rješenja $B(0)$
   while ($m <$ max. broj generacija) **do**
   begin
   $m \leftarrow m + 1$
6. **generiraj** $P(m)$ promatrajući stanja $Q(m-1)$
7. **popravi** $P(m)$
8. evaluiraj $P(m)$
9. **ažuriraj** kubit pojedince iz populacije $Q(m)$ koristeći Q-vrata
10. usporedi rješenja iz $B(m-1)$ i $P(m)$, te pohrani bolje rješenje u $B(m)$
11. usporedi novo najbolje rješenje b_i iz $B(m)$ s trenutnim najboljim rješenjem b;
    ako je bolje, zamijeni b s b_i
12. **if** (period migracije)
    then migriraj b ili b_i ka $B(m)$ globalno ili lokalno, redom

   end

end

3.2 Kvantno inspirirani binarni algoritam optimizacije rojem čestica - QBPSO

U Poglavlju 2.3 predstavili smo binarnu verziju PSO algoritma - BPSO algoritam. U prethodnom poglavlju upoznali smo se s radom evolucijskog algoritma i Q-vratima čija

je uloga rotacija kubita kako bismo dobili evoluciju populacije. Sada možemo iskoristiti to znanje kako bismo predstavili binarnu verziju PSO algoritma inspiriranu kvantnim računarstvom, tzv. QBPSO algoritam. QBPSO algoritam potreban nam je u Poglavlju 4 gdje konačno opisujemo LAQPSO algoritam za rješavanje Problema opredjeljenja jedinica.

Poboljšanja u odnosu na BPSO:

1. Za ažuriranje brzine čestice iz 2.10, ne koristi se više sigma funkcija $S(V_{ri}^{m+1})$. Sada koristimo pohranjenu vjerojatnost $|\beta|^2$ u $r-tom$ u individualnom kubitu za ažuriranje vektora položaja čestice. Stoga možemo modificirati definiciju 2.14 tako da se $i-ti$ element čestice r ažurira prema sljedećem:

$$X_{ri}^{m+1} = \begin{cases} 1 & \text{ako } H_{ri} < |\beta_{ri}|^2 \\ 0 & \text{inače} \end{cases} \tag{3.5}$$

   gdje je $r = 1, 2, 3, \ldots, k$, $i = 1, 2, 3, \ldots$, n, a H_{ri} je nasumično odabran broj iz intervala $[0, 1]$. [5]

2. Faktor inercije ω i faktori $(c1, c2)$ ovdje su izostavljeni, te se umjesto njih koristi rotacijski kut.

3. Uvode se nova rotacijska vrata kao operator varijacije. Dosadašnja rotacijska vrata iz Definicije 3.2 zahtijevala su unaprijed određenu tablicu za određivanje kuta rotacije. Sada, rotacijski kut možemo odrediti bez tablice:

$$\Delta\theta_{ri}^{m} = \theta^{m} \times \{\gamma_{1r}^{m} \times (Pbest_{ri}^{m} - X_{ri}^{m}) + \gamma_{2r}^{m} \times (Gbest_{i}^{m} - X_{ri}^{m})\} \tag{3.6}$$

   gdje je θ veličina kuta rotacije, $(Pbest_{ri}^{m}, Gbest_{i}^{m})$ lokalni i globalni najbolji položaji, a (y_{1r}, y_{2r}) su određeni sljedećim jednadžbama: [8]

$$\gamma_{1r} = \begin{cases} 0 & \text{ako } fitness(X_r^m) \geq fitness(Pbest_r^m) \\ 1 & \text{inače} \end{cases} \tag{3.7}$$

$$\gamma_{2r} = \begin{cases} 0 & \text{ako } fitness(X_r^m) \geq fitness(Gbest^m) \\ 1 & \text{inače} \end{cases} \tag{3.8}$$

   Svaka se čestica može približiti optimalnom rješenju kroz vlastito iskustvo i iskustva svojih susjeda. Veličina kuta rotacije može utjecati na kvalitetu rješenja i brzinu konvergencije. Stoga, ispravan odabir kuta rotacije rezultira manjim brojem iteracija u pronalaženju optimalnog rješenja. Općenito, vrijednosti iz intervala $[0.001\pi, 0.05\pi]$

preporučuju se za veličinu kuta rotacije, no kako bi poboljšali karakteristike konvergencije, predlaže se dinamičko računanje kuta θ tako da se veličina kuta monotono smanjuje od θ_{max} do θ_{min} duž iteracije kako slijedi:

$$\theta = \theta_{max} - \frac{\theta_{max} - \theta_{min}}{iter_{max}} \times m \tag{3.9}$$

gdje je $iter_{max}$ maksimalan broj iteracija, a m je trenutna iteracija.

Možemo primijetit da je ažuriranje vektora položaja čestice slično kao u koraku 2 QEA algoritma, no da se odabir kuta razlikuje (u QEA fiksiramo kut, a ovdje ga dinamički računamo). Prednost QBPSO nad BPSO algoritmom je izostavljanje kontrolnih parametara ω, (c_1, c_2). Točan odabir tih parametara često zahtijeva metodu pokušaja i pogreške za svaki novi UC problem. Budući QBPSO može naći rješenje bez spomenutih parametara, te uz korištenje principa kvantnog računanja, možemo očekivati bolju učinkovitost čak i s malom populacijom u usporedbi s konvencionalnim PSO algoritmima. [8]

Postupak predloženog QBPSO algoritma može biti sažet na sljedeći način:

Algoritam 6 QBPSO algoritam

begin

1. inicijaliziraj kubit pojedinca i poziciju populacije
2. postavi inicijalne *Pbest* i *Gbest*

do while
for $r = 1$ to k

3. ažuriraj kubit pojedinca r-te čestice
4. izmijeni poziciju r-te čestice
5. ažuriraj *Pbest* r-te čestice

$r = r + 1$

6. ažuriraj *Gbest*

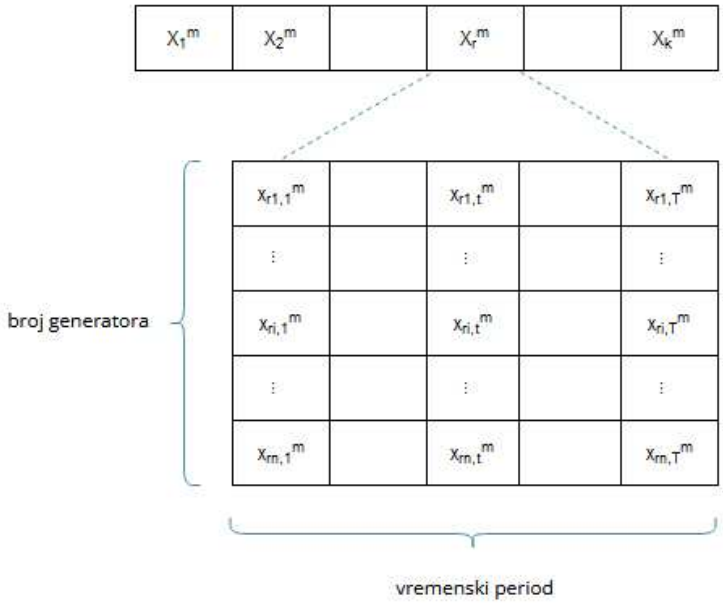
dok se ne ispune uvjeti za prekid

end

Primjena QBPSO algoritma na UC problem

Problem opredjeljenja jedinica podrazumijeva određivanje stanja (uključeno/isključeno) generatora. Struktura populacije predloženog QBPSO algoritma za UC problem prikazana

je na slici 3.3. Sa k označimo veličinu populacije, sa n označimo broj generatora, a s T označimo vremenski horizont. $x_{ri,t}$, postavljen je na 1 ako je generator i, čestice r, u satu t uključen, a inače je postavljen na 0, za sve $r = 1, 2, \cdots, k$, $i = 1, 2, \cdots, n$, $t = 1, 2, \cdots, T$.



Slika 3.3: Struktura populacije u QBPSO algoritmu za UC problem [8]

1. U koraku 1 inicijaliziramo $\alpha^0_{ri,t}$ i $\beta^0_{ri,t}$ na $\frac{1}{\sqrt{2}}$ za svaki kubit pojedinac. Nakon generiranja random broja $H_{ri,t}$, inicijalna vrijednost i-tog elementa čestice r u satu t je jednaka 1 ako je $H_{ri,t} < \frac{1}{2}$, a inače je jednaka 0.

2. Za svaku česticu r, inicijalni $Pbest_r$ postavljen je na svoju inicijalnu poziciju, a inicijalni $Gbest$ se postavlja na poziciju čestice s minimalnim troškom.

3. Kubit pojedinci ažuriraju se primjenom rotacijskih vrata. Veličina rotacijskog kuta određuje se prema 3.9, pa možemo odrediti rotacijski kut za pojedini kubit tako da:

$$\Delta\theta^{m+1}_{ri,t} = \theta^m \times \{\gamma^m_{1r} \times (Pbest^m_{ri,t} - X^m_{ri,t}) + \gamma^m_{2r} \times (Gbest^m_{i,t} - X^m_{ri,t})\} \tag{3.10}$$

Tada možemo dobiti novi par (α, β) svakog kubita kao:

$$\begin{bmatrix} \alpha^{m+1}_{ri,t} \\ \beta^{m+1}_{ri,t} \end{bmatrix} = \begin{bmatrix} cos(\Delta\theta^{m+1}_{ri,t}) & -sin(\Delta\theta^{m+1}_{ri,t}) \\ sin(\Delta\theta^{m+1}_{ri,t}) & cos(\Delta\theta^{m+1}_{ri,t}) \end{bmatrix} \begin{bmatrix} \alpha^m_{ri,t} \\ \beta^m_{ri,t} \end{bmatrix} \tag{3.11}$$

Ažurirani kubit mora zadovoljavati uvjet normalizacije: $|\alpha^m_{ri,t}|^2 + |\beta^m_{ri,t}|^2 = 1$.

4. Vektor pozicije r-te čestice u iteraciji m dobije se kao:

$$X_{ri,t}^{m+1} = \begin{cases} 1 & \text{ako } H_{ri,t} < |\beta_{ri,t}|^2 \\ 0 & \text{inače} \end{cases} \tag{3.12}$$

5. Ažuriramo $Pbest_r^m$ prema sljedećem:

$$Pbest_r^{m+1} = \begin{cases} X_r^{m+1} & \text{ako } fitness(X_r^{m+1}) < fitness(Pbest_r^m) \\ Pbest_r^m & \text{inače} \end{cases} \tag{3.13}$$

   $Gbest_r^{m+1}$ se postavlja na najbolju vrijednost od $Pbest_r^{m+1}$

6. Uvjet za prekid je ispunjen ako je dostignut unaprijed određen, maksimalan broj iteracija.

Poboljšani QBPSO algoritam - IQBPSO

QBPSO algoritam možda neće uspjeti u pronalaženju optimalne vrijednosti rješenja. Ova verzija PSO algoritma naziva se IQBPSO (*eng. improved QBPSO algorithm*) u kojoj promatramo *fitness* vrijednosti u *Pbest* za prvu polovicu iteracija, te *fitness* vrijednosti u *Gbest* u drugoj polovici iteracija:

$$\gamma_{1r} = \begin{cases} 0 & \text{ako } m \geq iter_{max}/2 \\ 1 & \text{inače} \end{cases} \tag{3.14}$$

$$\gamma_{2r} = \begin{cases} 1 & \text{ako } m \geq iter_{max}/2 \\ 0 & \text{inače} \end{cases} \tag{3.15}$$

dok se rotacijski kut ažurira kao kod 3.9. Uvodimo ga kako bi u 4.4 jasno mogli pokazati prednost IQBPSO nad QBPSO algoritmom, te prednost LAQPSO nad IQBPSO.

Poglavlje 4

Algoritam optimizacije rojem čestica s lokalnim privlačenjem - LAQPSO

Algoritme koje smo proučavali u Poglavlju 3 - QEA i QBPSO sada kombiniramo kako bi napokon opisali LAQPSO algoritam. Dodatno, uvodimo koncept lokalnog privlačenja. U tu svrhu, upoznajmo se s osnovnim pojmovima koristeći članak *A new quantum particle swarm optimization algorithm with local attracting* [2].

4.1 Kvantni kut

Uzimajući u obzir uvjet normalizacije ($|\alpha|^2 + |\beta|^2 = 1$), kvantni kut možemo definirati na sljedeći način:

$$q_{ri}^m = \begin{bmatrix} \alpha_{ri}^m \\ \beta_{ri}^m \end{bmatrix} \longrightarrow \theta_{ri}^m : \begin{cases} |q_{ri}^m\rangle = \cos\theta_{ri}^m |0\rangle + \sin\theta_{ri}^m |0\rangle \\ \theta_{ri}^m = \arctan\frac{\alpha_{ri}^m}{\beta_{ri}^m} \end{cases} \tag{4.1}$$

Posljedično, kvantni kut definira kubit. Tada se populacija kubit pojedinaca, kao i pojam kubit pojedinca iz Poglavlja 3.1, definicija 3.2 mogu interpretirati u formi kvantnog kuta:

$$q_r^m = [q_{r1}^m, q_{r2}^m, \cdots, q_{rn}^m] \tag{4.2}$$

$$\downarrow$$

$$\theta_r^m = [\theta_{r1}^m, \theta_{r2}^m, \cdots, \theta_{rn}^m] \tag{4.3}$$

$$Q(m) = [q_1^m, q_2^m, \cdots, q_k^m] \tag{4.4}$$

$$\downarrow$$

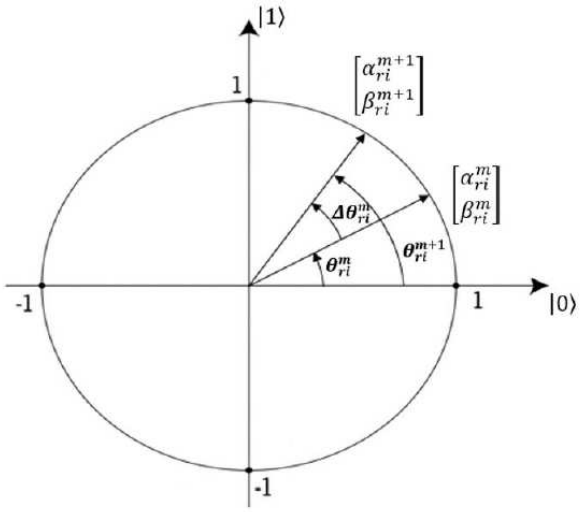
$$\theta(m) = [\theta_1^m, \theta_2^m, \cdots, \theta_k^m] \tag{4.5}$$

Također, operator rotacije kvantnog kuta definiran u 3.3 sada možemo matrično prikazati u drugoj bazi preko kuteva.

$$q_{ri}^{m+1} = \begin{bmatrix} \alpha_{ri}^{m+1} \\ \beta_{ri}^{m+1} \end{bmatrix} = \begin{bmatrix} cos(\Delta\theta_{ri}^m) & -sin(\Delta\theta_{ri}^m) \\ sin(\Delta\theta_{ri}^m) & cos(\Delta\theta_{ri}^m) \end{bmatrix} \begin{bmatrix} \alpha_{ri}^m \\ \beta_{ri}^m \end{bmatrix} \tag{4.6}$$

$$\Downarrow$$

$$\theta_{ri}^{m+1} = \Delta\theta_{ri}^m + \theta_{ri}^m \tag{4.7}$$



Slika 4.1: Kvantni kut [5]

4.2 Lokalno privlačenje

U PSO algoritmu, čestice se nasumično raspoređuju u prostoru pretraživanja, a brzina se ažurira na temelju iskustva svake čestice (svakog pojedinca) i iskustva svih susjeda. Jednadžbe 2.10 i 2.11 pokazuju da se stanje svake čestice može prikazati vektorom pozicije i vektorom brzine koji opisuju putanju čestice. Dodatno, uvodimo novi vektor $P_r^m = [P_{r1}^m, P_{r2}^m, \cdots, P_{rn}^m]$ kojeg nazivamo lokalni atraktor.

Pojedinu komponentu lokalnog atraktora u diskretnom binarnom prostoru možemo definirati na sljedeći način:

$$P_{ri}^m = \lambda_{ri}^m \cdot Pbest_{ri}^m + (1 - \lambda_{ri}^m) \cdot Gbest_i^m \tag{4.8}$$

gdje je λ_{ri}^m nasumično odabran, prirodan broj koji može biti 0 ili 1. Možemo zaključiti da se točka P_{ri}^m nalazi između $Pbest_r^m$ i $Gbest^m$ u binarnom prostoru. Prilikom konvergencije PSO algoritma, r-ta čestica pomiče se prema P_r^m, dok se njena brzina smanjuje. Konvergencija je zagarantirana budući se čestica r centrira oko točke P_r^m, te time smanjuje raznolikost populacije.

4.3 LAQPSO algoritam

U LAQPSO algoritmu, kvantni kut koristi se za kodiranje kubita, te se sve čestice u populaciji smatraju rojem kvantnih kutova.

Prvi korak je inicijalizacija. Svaki kvantni kut $\theta_{ri}^0, i = 1, 2, \cdots n$ svake čestice $\theta_r^0, r = 1, 2, \cdots, k$ u populaciji $Q(m)$ inicijaliziran je s $\pi/4$ (ovaj postupak ekvivalentan je inicijaliziranju svake amplitude QEA algoritma s $1/\sqrt{2}$). To znači da svaka čestica predstavlja linearnu superpoziciju svih mogućih stanja s istom vjerojatnošću $1/\sqrt{2}^n$. Slično kao u 3.5 kvantni kut možemo generirati na sljedeći način:

$$X_{ri}^{m+1} = \begin{cases} 0 & \text{ako } rand < |cos(\theta_{ri}^m)|^2 \\ 1 & \text{inače} \end{cases} \tag{4.9}$$

gdje je *rand* nasumično odabran, uniformno distribuiran broj između 0 i 1.

Prilikom ove operacije, r-ta čestica θ_r^m daje binarno rješenje $X_r^m = [X_{r1}^m, X_{r2}^m, \cdots, X_{rn}^m]$ duljine n. Tada možemo evaluirati *fitness* vrijednost svake čestice i pronaći *Pbest* i *Gbest*.

U evolucijskom procesu LAQPSO algoritma, kvantni kut ažuriramo pomoću kvantnih rotacijskih vrata gdje je rotacijski kut definiran pomoću lokalnog atraktora.

1. Da bismo odredili smjer rotacijskog kuta, definiramo ga pomoću lokalnog atraktora i položaja čestice trenutnog roja:

$$DIR(\theta_{ri}^m) = P_{ri}^m - X_{ri}^m \tag{4.10}$$

2. Da bismo odredili veličinu rotacijskog kuta, prvo definiramo kvantne kuteve koji odgovaraju lokalnom atraktoru P_r^m, tj. lokalni atraktor može se dobiti sa 100% vjerojatnošću promatranjem unaprijed definiranog kuta $\phi_r^m = [\phi_{r1}^m, \phi_{r2}^m, \cdots, \phi_{rn}^m]$. Ako je vrijednost ϕ_{ri}^m u intervalu $[0, \pi/2]$, prema 4.9, možemo zaključiti da ukoliko je P_{ri}^m jednaka 1 ili 0, pripadajući kvantni kut je $\pi/2$ ili 0, redom. Dakle, vrijednost kvantnog kuta može se dobiti kao:

$$\phi_{ri}^m = \frac{\pi}{2} P_{ri}^m \tag{4.11}$$

Nakon toga, koristimo ϕ_{ri}^m i θ_{ri}^m kako bi odredili vrijednost rotacijskog kuta:

$$|\Delta\theta_{ri}^m| = a \cdot |\phi_{ri}^m - \theta_{ri}^m| \cdot rand \tag{4.12}$$

gdje je a faktor kontrakcije koji može podesiti veličinu kuta rotacije.
Konačno, rotacijski kut može se opisati slijedećom jednadžbom:

$$\Delta\theta_{ri}^m = DIR(\theta_{ri}^m) \cdot |\Delta\theta_{ri}^m| \tag{4.13}$$

Primijetimo da je razlika između zadnja 3 opisana algoritma (QEA, QBPSO, LAQPSO) u definiranju rotacijskog kuta. Možemo pretpostaviti da LAQPSO algoritam ima veću učinkovitost pretraživanja u usporedbi s ostalim pristupima zbog svoje sposobnosti automatske adaptacije na različite karakteristike prostora pretraživanja. Rezultate potvrđujemo u Poglavlju 4.4

Algoritam 7 LAQPSO algoritam

inicijaliziraj roj kvantnih kuteva $\theta(0) = [\theta_1^0, \theta_2^0, \cdots, \theta_k^0]$

repeat

1. odredi binarni roj promatrajući stanja $\theta(m)$ koristeći 4.9: $X^m = [X_1^m, X_2^m, \cdots, X_k^m]$
2. evaluiraj *fitness* vrijednosti svih čestica: $fitness(X_r^m)$
3. if $fitness(X_r^m) < fitness(Pbest_r^{m-1})$ then
   ažuriraj: $Pbest_r^m = X_r^m$
   end if
4. if $fitness(X_r^m) < fitness(Gbest^{m-1})$ then
   ažuriraj: $Gbest^m = X_r^m$
   end if
5. generiraj lokalni atraktor koristeći 4.8: $P(m) = [P_1^m, P_2^m, \cdots, P_k^m]$
6. definiraj smjer i vrijednost rotacijskog kuta, te evaluiraj rotacijski kut $\Delta\theta(m) = [\Delta\theta_1^m, \Delta\theta_2^m, \cdots, \Delta\theta_k^m]$ koristeći 4.13
7. ažuriraj roj kvantnih kuteva: $\theta(m+1) = \theta(m) + \Delta\theta(m)$

until ispunjeni su uvjeti za prekid ili je dostignut unaprijed određen broj iteracija

ispiši najbolje individualno i globalno rješenje

4.4 Simulacija i rezultati

Rezultati predstavljeni u ovom poglavlju preuzeti su iz [4, p. 7] kako bi se prikazala razlika između algoritma predloženog u ovom radu, LAQPSO i ostalih.

Algoritmi BPSO, IQBPSO i LAQPSO simulirani su korištenjem MATLAB R2017b okruženja kako bi se riješio problem opredjeljnja jedinica (UCP problem). Korišteno računalo u simulaciji ima sljedeće značajke: Core i5 CPU i 8GB RAM-a. Za provjeru potencijalnosti i sposobnosti tri algoritama u rješavanju UCP-a, IEEE 26 proizvodni sustav za ispitivanje jedinica koristi se kao energetski sustav. Ovaj sustav ispitivanja sastoji se od 26 jedinica (generatora) za opskrbu potražnje kroz vremenski period od 24 sata.
Kriterij za ograničenje rotirajuće rezerve u ovom sustavu jednak je najvećem kapacitetu angažiranih proizvodnih jedinica - $400MW$. Dimenzija prostora za pretraživanje iznosi 26×24, gdje je 26 broj proizvodnih jedinica, a 24 je vremenski horizont (broj sati). Tablica 4.1 predstavlja zahtjev za opterećenjem sustava tijekom 24 sata. Tablica 4.3 prikazuje parametre generatora. Veličina populacije postavljena je na 100, dok je maksimalni broj iteracija 500. Optimalni izbori parametara za 3 spomenuta algoritma dana su kao $c_1 = c_2 = 1.9, \omega_{max} = 0.9, \omega_{min} = 0.4$ za BPSO, $\theta_{max} = 0.1\pi, \theta_{min} = 0.05\pi$ za IQBPSO i $a = 0.1$ za LAQPSO. Tablica 4.2 prikazuje usporedbu ukupnih operativnih troškova

BPSO, IQBPSO, IRL i Binary/Real PSO (BRPSO) algoritama, gdje je zadnji trošak rezultat LAQPSO algoritma - $720921. Tablice 4.4, 4.5, 4.6 prikazuju rezultate simulacije BPSO, IQBPSO i LAQPSO algoritma. Slika 4.2 predstavlja konvergenciju ka rješenju sva 3 algoritma, gdje x os predstavlja broj iteracija, a y os ukupan operativni trošak ($). Možemo primijetiti da je LAQPSO algoritam dosta brži u usporedbi s ostala dva algoritma.

Rezultati simulacije pokazali su da je najbolje rješenje za UCP postigao LAQPSO algoritam u usporedbi s BPSO, QBPSO, IQBPSO. Također, sva ograničenja su zadovoljena. Ovo pokazuje da LAQPSO algoritam ima bolje performanse za postizanje cilja minimiziranja ukupnog operativnog troška od ostalih algoritama koji se koriste za ovaj testni sustav. Može se pokazati da LAQPSO ima izvrsnu karakteristiku konvergencije za smanjenje ukupnih operativnih troškova i učinkovitiji je od ostalih.

Sat	**Potražnja (MW)**	**Sat**	**Potražnja (MW)**
1	1700	13	2590
2	1730	14	2550
3	1690	15	2620
4	1700	16	2650
5	1750	17	2550
6	1850	18	2530
7	2000	19	2500
8	2430	20	2550
9	2540	21	2600
10	2600	22	2480
11	2670	23	2200
12	2590	24	1840

Tablica 4.1: Potražnja u IEEE 26 simulacijskom sustavu generatora [4]

Pristup	**Ukupan trošak** ($)
ILR	725,996.9
BRPSO	721208
BPSO	721958
IQBPSO	721261
LAQPSO	720921

Tablica 4.2: Ukupni operativni trošak različitih algoritama [4]

Jedinica (generator)	**a ($/h)**	**b ($/MWh)**	**c ($/M$W^2$h)**	P_{max} **(MW)**	P_{min} **(MW)**	**MUT (h)**	**MDT (h)**	**HSC ($)**	**CSC ($)**	**CSH (h)**	**Početno stanje (h)**
1	311.9102	7.5031	0.0019	400	100	8	5	500	500	10	10
2	310.0021	7.4921	0.0019	400	100	8	5	500	500	10	10
3	177.0575	10.8616	0.0015	350	140	8	5	300	300	8	10
4	260.176	23.2	0.0026	197	68.95	5	4	200	200	8	-4
5	259.649	23.1	0.0026	197	68.95	5	4	200	200	8	-4
6	259.131	23	0.0026	197	68.95	5	4	200	200	8	-4
7	143.5972	10.7583	0.0049	155	54.25	5	3	150	150	6	5
8	134.3179	10.7367	0.0048	155	54.25	5	3	150	150	6	5
9	143.0288	10.7154	0.0047	155	54.25	5	3	150	150	6	5
10	142.7348	10.694	0.0046	155	54.25	5	3	150	150	6	5
11	218.7752	18.2	0.006	100	25	4	2	70	70	4	-3
12	218.335	18.1	0.0061	100	25	4	2	70	70	4	-3
13	217.8952	18	0.0062	100	25	4	2	70	70	4	-3
14	81.6259	13.4073	0.0093	76	15.2	3	2	50	50	3	3
15	81.4681	13.3805	0.0091	76	15.2	3	2	50	50	3	3
16	81.298	13.3538	0.0089	76	15.2	3	2	50	50	3	3
17	81.1364	13.3272	0.0088	76	15.2	3	2	50	50	3	3
18	118.8206	37.8896	0.0143	20	4	0	0	20	20	2	-1
19	118.4576	37.777	0.0136	20	4	0	0	20	20	2	-1
20	118.1083	37.6637	0.0126	20	4	0	0	20	20	2	-1
21	117.7551	37.551	0.012	20	4	0	0	20	20	2	-1
22	24.8882	26.0611	0.0285	12	2.4	0	0	0	0	1	-1
23	24.7605	25.9318	0.0284	12	2.4	0	0	0	0	1	-1
24	24.6382	25.8027	0.028	12	2.4	0	0	0	0	1	-1
25	24.411	25.6753	0.0265	12	2.4	0	0	0	0	1	-1
26	24.3891	25.5472	0.0253	12	2.4	0	0	0	0	1	-1

Tablica 4.3: Parametri generatora IEEE 26 simulacijskog sustava [4]

Tablica 4.4: Izlazna snaga generatora BPSO algoritma

Sat	**U(1)**	**U(2)**	**U(3)**	**U(4)**	**U(5)**	**U(6)**	**U(7)**	**U(8)**	**U(9)**	**U(10)**
						MW				
1	400	400	339.2813	0	0	0	117.1972	120.9045	125.2009	130.2161
2	400	400	350	0	0	0	120.7345	124.4859	128.8429	133.9367
3	400	400	335.6071	0	0	0	116.0429	119.7357	124.0124	129.0019
4	400	400	338.2315	0	0	0	116.8674	120.5705	124.8614	129.8692
5	400	400	350	0	0	0	122.2489	126.0192	130.4022	135.5297
6	400	400	350	0	0	0	140.5688	144.5676	149.2643	154.7992
7	400	400	350	0	68.95	0	155	155	155	155
8	400	400	350	0	68.95	68.95	155	155	155	155
9	400	400	350	68.95	68.95	68.95	155	155	155	155
10	400	400	350	68.95	68.95	80.9	155	155	155	155
11	400	400	350	71.5429	91.5992	111.2579	155	155	155	155
12	400	400	350	68.95	68.95	70.9	155	155	155	155
13	400	400	350	68.95	68.95	70.9	155	155	155	155
14	400	400	350	68.95	68.95	68.95	155	155	155	155
15	400	400	350	68.95	72.7321	92.3179	155	155	155	155
16	400	400	350	68.95	84.9086	104.5414	155	155	155	155
17	400	400	350	68.95	68.95	68.95	155	155	155	155
18	400	400	350	68.95	68.95	68.95	155	155	155	155
19	400	400	350	68.95	68.95	68.95	155	155	155	155
20	400	400	350	68.95	68.95	68.95	155	155	155	155
21	400	400	350	68.95	68.95	79.3	155	155	155	155
22	400	400	350	68.95	68.95	68.95	155	155	155	155
23	400	400	350	0	68.95	68.95	155	155	155	155
24	400	400	350	0	0	0	138.1262	142.0945	146.7494	152.2299

Sat	**U(11)**	**U(12)**	**U(13)**	**U(14)**	**U(15)**	**U(16)**	**U(17)**	**U(18)**	**U(19)**	**U(20)**
						MW				
1	0	0	0	15.2	15.2	15.2	15.2	0	0	0
2	0	0	0	15.2	15.2	15.2	15.2	0	0	0
3	0	0	0	15.2	15.2	15.2	15.2	0	0	0
4	0	0	0	15.2	15.2	15.2	15.2	0	0	4
5	0	0	25	15.2	15.2	15.2	15.2	0	0	0
6	25	0	25	15.2	15.2	15.2	15.2	0	0	0
7	25	0	25	24.1235	26.1793	28.1096	30.2376	0	0	0
8	61.1096	67.8816	74.7088	76	76	76	76	4	4	0
9	79.9895	86.3296	92.831	76	76	76	76	0	0	0
10	100	100	100	76	76	76	76	0	0	0
11	100	100	100	76	76	76	76	4	4	4
12	100	100	100	76	76	76	76	0	0	0
13	100	100	100	76	76	76	76	0	0	0
14	83.3943	89.6565	96.0992	76	76	76	76	0	0	0
15	100	100	100	76	76	76	76	0	0	0
16	100	100	100	76	76	76	76	0	4	4
17	83.3943	89.6565	96.0992	76	76	76	76	0	0	0
18	76.5846	83.0026	89.5628	76	76	76	76	0	0	0
19	66.3701	73.0218	79.7581	76	76	76	76	0	0	0
20	83.3943	89.6565	96.0992	76	76	76	76	0	0	0
21	100	100	100	76	76	76	76	0	0	0
22	59.5604	66.3679	73.2217	76	76	76	76	0	0	0
23	0	25	25	55.859	58.682	61.1571	64.0018	0	0	0
24	0	25	25	15.2	15.2	15.2	15.2	0	0	0

Sat	U(21)	U(22)	U(23)	U(24)	U(25)	U(26)	Rezerva	Potražnja	STC ($)
					MW				
1	4	0	0	0	2.4	0	406	1700	20
2	4	0	2.4	2.4	0	2.4	400	1730	
3	0	0	0	0	2.4	2.4	408	1690	
4	0	0	0	2.4	2.4	0	418	1700	20
5	0	0	0	0	0	0	424	1750	70
6	0	0	0	0	0	0	424.0001	1850	70
7	0	0	0	0	2.4	0	483	2000	200
8	4	0	0	2.4	0	0	410	2430	330
9	0	0	0	0	0	0	424.9999	2540	200
10	0	0	2.4	2.4	2.4	0	401	2600	
11	0	0	2.4	2.4	2.4	0	403	2670	
12	0	0	0	2.4	2.4	2.4	411	2590	
13	0	0	2.4	0	2.4	2.4	411	2590	
14	0	0	0	0	0	0	415	2550	
15	0	2.4	2.4	2.4	2.4	2.4	405	2620	
16	0	2.4	2.4	2.4	0	2.4	403	2650	40
17	0	0	0	0	0	0	415	2550	
18	0	0	0	0	0	0	435	2530	
19	0	0	0	0	0	0	465	2500	
20	0	0	0	0	0	0	415	2550	
21	4	0	0	2.4	0	2.4	409	2600	20
22	0	0	0	0	0	0	485	2480	
23	0	0	0	2.4	0	0	480.0001	2200	
24	0	0	0	0	0	0	434	1840	
									1030
Ukupni operativni trošak = $721958									

Tablica 4.5: Izlazna snaga generatora IQBPSO algoritma

Sat	U(1)	U(2)	U(3)	U(4)	U(5)	U(6)	U(7)	U(8)	U(9)	U(10)
						MW				
1	400	400	331.1455	0	0	0	114.6412	118.3165	122.5693	127.5276
2	400	400	344.2678	0	0	0	118.7638	122.4906	126.8139	131.8639
3	400	400	333.42	0	0	0	115.3558	119.04	123.305	128.2792
4	400	400	336.7443	0	0	0	116.4002	120.0975	124.3803	129.3777
5	400	400	350	0	0	0	122.2489	126.0192	130.4022	135.5297
6	400	400	350	0	0	0	140.5688	144.5676	149.2643	154.7992
7	400	400	350	0	0	0	155	155	155	155
8	400	400	350	68.95	68.95	68.95	155	155	155	155
9	400	400	350	68.95	68.95	68.95	155	155	155	155
10	400	400	350	68.95	68.95	80.9	155	155	155	155
11	400	400	350	70.7501	90.7972	110.4528	155	155	155	155
12	400	400	350	68.95	68.95	70.9	155	155	155	155
13	400	400	350	68.95	68.95	70.9	155	155	155	155
14	400	400	350	68.95	68.95	68.95	155	155	155	155
15	400	400	350	68.95	73.1313	92.7187	155	155	155	155
16	400	400	350	68.95	84.9086	104.5414	155	155	155	155
17	400	400	350	68.95	68.95	68.95	155	155	155	155
18	400	400	350	68.95	68.95	68.95	155	155	155	155
19	400	400	350	68.95	68.95	68.95	155	155	155	155
20	400	400	350	68.95	68.95	68.95	155	155	155	155
21	400	400	350	68.95	68.95	80.9	155	155	155	155
22	400	400	350	68.95	68.95	68.95	155	155	155	155
23	400	400	350	0	68.95	68.95	155	155	155	155
24	400	400	350	0	0	0	138.1262	142.0945	146.7494	152.2299

Sat	U(11)	U(12)	U(13)	U(14)	U(15)	U(16)	U(17)	U(18)	U(19)	U(20)
					MW					
1	0	0	25	15.2	15.2	15.2	15.2	0	0	0
2	0	0	25	15.2	15.2	15.2	15.2	0	0	0
3	0	0	25	0	15.2	15.2	15.2	0	0	0
4	0	0	25	0	15.2	15.2	15.2	0	0	0
5	0	0	25	15.2	15.2	15.2	15.2	0	0	0
6	0	25	25	15.2	15.2	15.2	15.2	0	0	0
7	25	25	25	33.798	36.0876	38.184	40.5305	0	0	4
8	42.5362	49.7331	56.8807	76	76	76	76	0	0	0
9	79.9895	86.3296	92.831	76	76	76	76	0	0	0
10	100	100	100	76	76	76	76	0	0	0
11	100	100	100	76	76	76	76	0	4	4
12	100	100	100	76	76	76	76	0	0	0
13	100	100	100	76	76	76	76	0	0	0
14	83.3943	89.6565	96.0992	76	76	76	76	0	0	0
15	100	100	100	76	76	76	76	0	0	0
16	100	100	100	76	76	76	76	0	0	4
17	83.3943	89.6565	96.0992	76	76	76	76	0	0	0
18	76.5846	83.0026	89.5628	76	76	76	76	0	0	0
19	66.3701	73.0218	79.7581	76	76	76	76	0	0	0
20	83.3943	89.6565	96.0992	76	76	76	76	0	0	0
21	100	100	100	76	76	76	76	0	0	0
22	59.5604	66.3679	73.2217	76	76	76	76	0	0	0
23	25	25	25	65.3397	68.3919	71.0298	74.0886	0	0	0
24	0	25	25	15.2	15.2	15.2	15.2	0	0	0

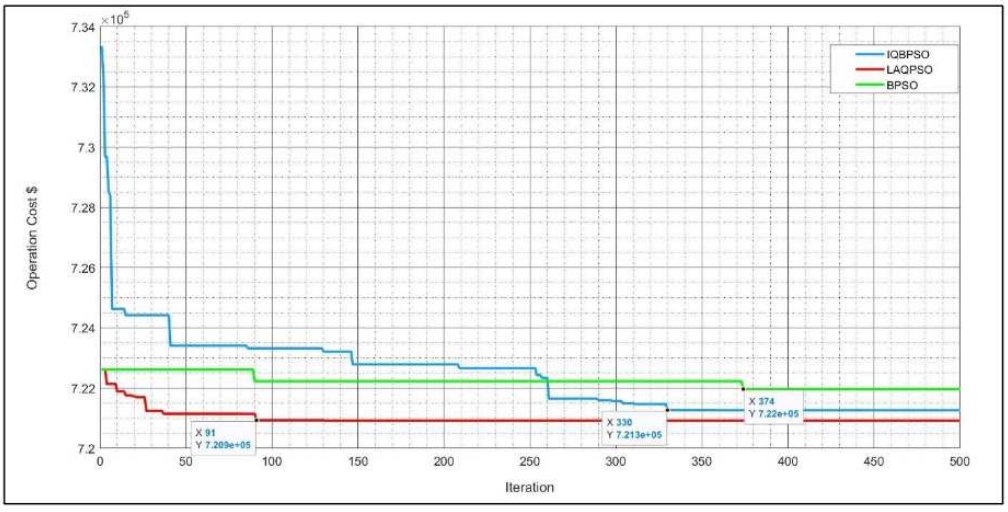
Sat	U(21)	U(22)	U(23)	U(24)	U(25)	U(26)	Rezerva	Potražnja	STC ($)
					MW				
1	0	0	0	0	0	0	474	1700	70
2	0	0	0	0	0	0	444	1730	
3	0	0	0	0	0	0	408	1690	
4	0	0	0	2.4	2.4	2.4	410	1700	
5	0	0	0	0	0	0	424	1750	50
6	0	0	0	0	0	0	424	1850	70
7	0	2.4	0	0	0	0	406	2000	90
8	0	0	0	2.4	0	0	535	2430	600
9	0	0	0	0	0	0	425	2540	
10	0	0	0	2.4	2.4	2.4	401	2600	
11	4	2.4	2.4	2.4	2.4	2.4	415	2670	60
12	0	0	0	2.4	2.4	2.4	411	2590	
13	0	0	0	0	2.4	2.4	411	2590	
14	0	0	0	0	0	0	415	2550	
15	4	2.4	0	2.4	2.4	0	401	2620	20
16	4	0	2.4	2.4	2.4	2.4	403	2650	20
17	0	0	0	0	0	0	415	2550	
18	0	0	0	0	0	0	435	2530	
19	0	0	0	0	0	0	465	2500	
20	0	0	0	0	0	0	415	2550	
21	0	0	0	2.4	2.4	2.4	401	2600	
22	0	0	0	0	0	0	485	2480	
23	0	0	2.4	0	2.4	2.4	407	2200	
24	0	0	0	0	0	0	434	1840	
									980
Ukupni operativni trošak = $721261									

Tablica 4.6: Izlazna snaga generatora LAQPSO algoritma

Sat	**U(1)**	**U(2)**	**U(3)**	**U(4)**	**U(5)**	**U(6)**	**U(7)**	**U(8)**	**U(9)**	**U(10)**
						MW				
1	400	400	338.9314	0	0	0	117.0873	120.7931	125.0877	130.1004
2	400	400	344.2678	0	0	0	118.7638	122.4906	126.8139	131.8639
3	400	400	333.42	0	0	0	115.3558	119.04	123.305	128.2792
4	400	400	336.7443	0	0	0	116.4002	120.0975	124.3803	129.3777
5	400	400	350	0	0	0	122.2489	126.0192	130.4022	135.5297
6	400	400	350	0	0	0	140.5688	144.5676	149.2643	154.7992
7	400	400	350	0	0	0	155	155	155	155
8	400	400	350	68.95	68.95	68.95	155	155	155	155
9	400	400	350	68.95	68.95	68.95	155	155	155	155
10	400	400	350	68.95	68.95	80.9	155	155	155	155
11	400	400	350	71.5429	91.5992	111.2579	155	155	155	155
12	400	400	350	68.95	68.95	70.9	155	155	155	155
13	400	400	350	68.95	68.95	70.9	155	155	155	155
14	400	400	350	68.95	68.95	68.95	155	155	155	155
15	400	400	350	68.95	72.7321	92.3179	155	155	155	155
16	400	400	350	68.95	84.9086	104.5414	155	155	155	155
17	400	400	350	68.95	68.95	68.95	155	155	155	155
18	400	400	350	68.95	68.95	68.95	155	155	155	155
19	400	400	350	68.95	68.95	68.95	155	155	155	155
20	400	400	350	68.95	68.95	68.95	155	155	155	155
21	400	400	350	68.95	68.95	80.9	155	155	155	155
22	400	400	350	68.95	68.95	68.95	155	155	155	155
23	400	400	350	0	0	68.95	155	155	155	155
24	400	400	350	0	0	0	138.1262	142.0945	146.7494	152.2299

Sat	**U(11)**	**U(12)**	**U(13)**	**U(14)**	**U(15)**	**U(16)**	**U(17)**	**U(18)**	**U(19)**	**U(20)**
						MW				
1	0	0	0	15.2	15.2	15.2	15.2	0	0	0
2	0	0	25	15.2	15.2	15.2	15.2	0	0	0
3	0	0	25	0	15.2	15.2	15.2	0	0	0
4	0	0	25	0	15.2	15.2	15.2	0	0	0
5	0	0	25	15.2	15.2	15.2	15.2	0	0	0
6	0	25	25	15.2	15.2	15.2	15.2	0	0	0
7	25	25	25	33.798	36.0876	38.184	40.5305	0	4	0
8	42.5362	49.7331	56.8807	76	76	76	76	0	0	0
9	79.9895	86.3296	92.831	76	76	76	76	0	0	0
10	100	100	100	76	76	76	76	0	0	0
11	100	100	100	76	76	76	76	0	4	4
12	100	100	100	76	76	76	76	0	0	0
13	100	100	100	76	76	76	76	0	0	0
14	83.3943	89.6565	96.0992	76	76	76	76	0	0	0
15	100	100	100	76	76	76	76	0	0	0
16	100	100	100	76	76	76	76	0	0	4
17	83.3943	89.6565	96.0992	76	76	76	76	0	0	0
18	76.5846	83.0026	89.5628	76	76	76	76	0	0	0
19	66.3701	73.0218	79.7581	76	76	76	76	0	0	0
20	83.3943	89.6565	96.0992	76	76	76	76	0	0	0
21	100	100	100	76	76	76	76	0	0	0
22	59.5604	66.3679	73.2217	76	76	76	76	0	0	0
23	25	25	25	65.3397	68.3919	71.0298	74.0886	0	0	0
24	0	25	25	15.2	15.2	15.2	15.2	0	0	0

Sat	U(21)	U(22)	U(23)	U(24)	U(25)	U(26)	Rezerva	Potražnja	STC ($)
					MW				
1	0	0	0	2.4	2.4	2.4	410	1700	
2	0	0	0	0	0	0	444	1730	70
3	0	0	0	0	0	0	408	1690	
4	0	0	2.4	0	0	0	410	1700	
5	0	0	0	0	0	0	424	1750	50
6	0	0	0	0	0	0	424	1850	70
7	0	2.4	0	0	0	0	406	2000	70
8	0	0	0	0	0	0	535	2430	600
9	0	0	0	0	0	0	425	2540	
10	0	0	0	2.4	2.4	2.4	401	2600	
11	4	2.4	0	2.4	2.4	2.4	403	2670	60
12	0	0	0	2.4	2.4	2.4	411	2590	
13	0	0	0	2.4	2.4	2.4	411	2590	
14	0	0	0	0	0	0	415	2550	
15	0	2.4	2.4	2.4	2.4	2.4	405	2620	
16	4	0	2.4	2.4	2.4	2.4	403	2650	60
17	0	0	0	0	0	0	415	2550	
18	0	0	0	0	0	0	435	2530	
19	0	0	0	0	0	0	465	2500	
20	0	0	0	0	0	0	415	2550	
21	0	0	2.4	0	2.4	2.4	401	2600	
22	0	0	0	0	0	0	485	2480	
23	0	0	2.4	2.4	2.4	0	407	2200	
24	0	0	0	0	0	0	434	1840	
									980
Ukupni operativni trošak = $720921									



Slika 4.2: Konvergencija BPSO, IQBPSO, LAQPSO algoritama [4]

Poglavlje 5

Zaključak

Rješavanje Problema opredjeljenja jedinica ili UCP zahtijeva složenu proceduru odlučivanja u elektroenergetskom sustavu. Da bi se pojednostavilo donošenje odluka, te riješili kompleksni problemi velikih dimenzija, glavnu ulogu pri traženju optimalnijeg rješenja imaju inteligentne tehnike. Jedna od njih je LAQPSO algoritam. LAQPSO je hibridni algoritam nastao kombinacijom kvantnog računanja i PSO algoritma s lokalnim atraktorom. Lokalni atraktor pomogao je PSO algoritmu da prevlada nedostatke kao što su sporija brzina konvergencije i zarobljenost u lokalnim optimalnim rješenjima.

IEEE 26 energetski sustav proizvodnih jedinica koristi se za provjeru učinkovitosti LAQPSO algoritma. Pri simulaciji velikih energetskih sustava, rezultati usporedbe dokazuju superiornost, fleksibilnost, afektivnost, robusnost i veliki potencijal LAQPSO algoritma među ostalim algoritmima kao što su BSO, IQBPSO i drugi algoritmi za rješavanje UC problema. Rezultati su potvrdili da LAQPSO algoritam ima najbolje performanse (sposobnost i brzinu konvergencije) pri rješavanju složenih i nelinearnih problema. Također, sva ograničenja UC problema su zadovoljena, te se povećala ušteda. LAQPSO je za sada, zasigurno jedan od najefektivnijih algoritama za rješavanje problema opredjeljenja jedinica.

Bibliografija

[1] Fengqi You Akshay Ajagekar, *Quantum computing for energy systems optimization: Challenges and opportunities*, Cornell University, Ithaca, New York 14853, USA (2019), 1–34, https://arxiv.org/ftp/arxiv/papers/2003/2003.00254.pdf.

[2] Y. Fei D. Shao, S. Hu, *A new quantum particle swarm optimization algorithm with local attracting*, (2016), http://www.nnw.cz/doi/2016/2016_26_028.pdf.

[3] Kostas Hatalis, *Swarm Optimization: Goodbye Gradients*, (2018), https://www.datasciencecentral.com/profiles/blogs/swarm-optimization-goodbye-gradients.

[4] Ali Abduladheem; Ali Nasser Hussain. Ismail, *Solving the unit commitment problem in large systems using hybrid PSO algorithms*, IOP Conference Series. Materials Science and Engineering; **1105** (2021), br. 1, 1–15, https://www.proquest.com/openview/34dd20b3c8fbfe9b986b3e3c33cb4cfc/1?pq-origsite=gscholar&cbl=4998670.

[5] Ali N Ismail, Ali A; Hussain, *Unit commitment problem solution using Local Attracting Quantum PSO algorithm*, (2020), https://www.proquest.com/docview/2562029998.

[6] Jong Hwan Kim Kuk-Hyun Han, *Quantum-inspired evolutionary algorithm for a class of combinatorial optimization*, IEEE Transactions on Evolutionary Computation **6** (2002), br. 6, 580–593, https://ieeexplore.ieee.org/document/1134125.

[7] Lu Le, Hai Bang Ly, Binh Pham, Lê Vucng, Tuan Phạm, Hung Nguyen, Xuan Tuan Tran i Tien Thinh Le, *Hybrid Artificial Intelligence Approaches for Predicting Buckling Damage of Steel Columns Under Axial Compression*, Materials **12** (2019), 1670.

[8] Yun Won Jeong Jong Bae Park Se Hwan Jang Kwang Y. Lee, *A New Quantum-Inspired Binary PSO: Application to Unit Commitment Problems for Power Systems*,

IEEE Transactions on power systems (2010), https://ieeexplore.ieee.org/document/5428772.

[9] Isaac L. Chuang Michael A. Nielsen, *Quantum Computation and Quantum Information*, Cambridge University Press, New York, 2010, http://mmrc.amss.cas.cn/tlb/201702/W020170224608149940643.pdf.

[10] Adrian Tam, *A Gentle Introduction to Particle Swarm Optimization*, (2021), https://machinelearningmastery.com/a-gentle-introduction-to-particle-swarm-optimization/.

[11] Unknown, *Particle Swarm Optimization*, https://web2.qatar.cmu.edu/~gdicaro/15382-Spring18/hw/hw3-files/pso-book-extract.pdf.

Sažetak

Problem opredjeljenja jedinice ili *unit commitment problem - UC* je nelinearni problem mješovitog cjelobrojnog programiranja koji se koristi za minimiziranje operativnih troškova proizvodnih jedinica u elektroenergetskom sustavu. Može se opisati kao procedura uključivanja i isključivanja generatora kako bi se povećala izlazna snaga i zadovoljila tržišna potražnja. Naravno, prilikom odlučivanja koji od generatora će biti uključeni, a koji ne, niz ograničenja elektroenergetskog sustava mora biti zadovoljeno.

Na početku rada, upoznajemo se s radom i osnovnim pojmovima kvantnog računanja kao što su kvantni bit ili kubit, te superpozicija. Također se upoznajemo s definicijom Problema opredjeljenja jedinica i najpoznatijim algoritmom za njegovo rješavanje - algoritmom optimizacije rojem čestica (*eng. particle swarm optimization, PSO*). PSO algoritam koristi se za pronalaženje maksimuma ili minimuma funkcije definirane u višedimenzionalnom vremenskom prostoru. Sastoji od jata pojedinaca (čestica) koji se kreću u prostoru pretraživanja kako bi pronašli rješenje za dani problem. U obzir se uzima kretanje svake čestice prema brzini i prošlom iskustvu čestice, kao i iskustvo susjednih čestica.

Nadalje, predlaže se binarni algoritam optimizacije rojem čestica ili BPSO. Zašto binarna inačica? Pa upravo je ona najjednostavnija za reprezentaciju stanja generatora: 0 - isključeno i 1 - uključeno Razlika između PSO i BPSO pristupa je u definiranju vektora položaja čestice, koji je u BPSO binaran.

Kako bismo uspješno opisali rad LAQPSO algoritma, uvodimo pojam kvantno inspiriranog evolucijskog algortima - QEA. Temelji se na konceptima i principima kvantnog računarstva kao što su kvantni bit i superpozicija stanja. Okarakteriziran je odabirom (reprezentacijom) pojedinca, evaluacijskom funkcijom i dinamikom populacije. Uvodi se nova, vjerojatnosna oznaka za prikaz kubita, te pojam kubit pojedinca kao niza kubita.

Iako su pristupi temeljeni na BPSO algoritmu uspješno primijenjeni na kombinatoriku problema optimizacije u raznim područjima, BPSO algoritam ima nekoliko nedostataka od kojih je jedan preuranjena konvergencija pri rješavanju težih problema. Kako bi se

prevladali spomenuti nedostatci, uveden je novi binarni PSO algoritam inspiriran kvantnim računarstvom - QBPSO algoritam. Predloženi QBPSO kombinira konvencionalni BPSO s konceptima kvantnog računarstva. QBPSO koristi pojam kubit pojedinca za probabilistički prikaz, koji zamjenjuje postupak ažuriranja brzine u optimizaciji roja čestica. Predlažu se i novu rotacijska vrata, odnosno vrata za rotaciju koordinata za ažuriranje kubit pojedinaca u kombinaciji s dinamičkim kutom rotacije za određivanje veličine kuta.

Kako bi se performanse poboljšale i ukupni minimalni operativni trošak smanjio, algoritam roja kvantnih čestica lokalnog privlačenja - LAQPSO za rješavanje UC problema nastaje na temelju kvantnog evolucijskog algoritma (QEA) i QBPSO algoritma. Novi kvantni mehanizam prikaza bitova nazvan kvantni kut koristi se za kodiranje rješenja, te se uvodi pojam lokalnog atraktora koji je zadužen za automatsko određivanje rotacijskog kuta kvantnih rotacijskih vrata. Tijekom procesa traženja globalnog rješenja, veličina kuta rotacije prilagođava se važnom parametru zvanom koeficijent kontrakcije, koji može kvantitativno odrediti kompromis između sposobnosti istraživanja i sposobnosti eksploatacije.

Predloženi algoritam primjenjuje se za rješavanje UC problema za energetski sustav od 26 jedinica (generatora). Uspoređuje se s binarnim PSO (BPSO), poboljšanim kvantnim BPSO (IQBPSO) i drugim tehnikama kako bi se pokazala učinkovitost i točnost predloženog algoritma. Rezultati pokazuju vrhunske performanse LAQPSO algoritma za minimiziranje ukupnih troškova u usporedbi s drugim navedenim algoritmima.

Summary

Unit commitment problem (UC) is a nonlinear problem of mixed integer programming used to minimize the operating costs of generating units in the power system. It can be described as a procedure to turn the generator on and off to increase output power and meet market demand. When deciding which of the generators will be included and which will not, a number of limitations of the power system must be met.

At the beginning of the paper, we get acquainted with the work and basic concepts of quantum computing such as quantum bit or qubit, and superposition. We are also introduced to the definition of the Unit Commitmen Problem and the most well-known algorithm for its solution - the particle swarm optimization (PSO) algorithm. The PSO algorithm is used to find the maximum or minimum of a function defined in multidimensional vector space. It consists of a swarm of individuals (particles) moving in the search space to find a solution to a given problem. The movement of each particle according to the velocity and past experience of the particle is taken into account, as well as the experience of neighboring particles.

Furthermore, a binary particle swarm optimization algorithm or BPSO is proposed. The difference between the PSO and BPSO approaches is in defining the particle position vector, which is binary in BPSO.

In order to successfully describe the work of the LAQPSO algorithm, we introduce the concept of a quantum-inspired evolutionary algorithm - QEA. It is based on the concepts and principles of quantum computing such as quantum bit and state superposition. It is characterized by the selection (representation) of the individual, the evaluation function and the dynamics of the population. A new, probabilistic notation for representation is introduced, which is based on the concept of qubits and the notion of the qubit of an individual as a series of qubits.

Although BPSO-based approaches have been successfully applied to the combinatorics of optimization problems in various fields, the BPSO algorithm has several shortcomings,

one of which is premature convergence in solving more difficult problems. In order to overcome the mentioned shortcomings, a new binary PSO algorithm inspired by quantum computing - QBPSO algorithm - was introduced. The proposed QBPSO combines conventional BPSO with concepts of quantum computing. QBPSO uses the term qubit individual for probabilistic representation, which replaces the speed update process in particle swarm optimization. A new rotary gate is also proposed, ie a gate for the rotation of coordinates to update the qubit of individuals in combination with a dynamic rotation angle to determine the size of the angle.

In order to improve performance and reduce the overall minimum operating cost, the local attraction quantum particle swarm algorithm - LAQPSO for UC problem solving is based on the quantum evolution algorithm (QEA) and the PSO algorithm. A new quantum bit expression mechanism called quantum angle is used to encode the solution to the particle, and the notion of a local attractor is introduced which is in charge of automatically determining the rotational angle of a quantum rotating gate. During the process of finding a global solution, the magnitude of the rotation angle is adjusted to an important parameter called the contraction coefficient, which can quantify the trade-off between exploration capability and exploitation capability.

The proposed algorithm is applied to solve UC problems for a 26-unit (generator) power system. It is compared with binary PSO (BPSO), improved quantum BPSO (IQBPSO) and other techniques to demonstrate the efficiency and accuracy of the proposed algorithm. The results show the superior performance of the LAQPSO algorithm to minimize total costs compared to the other listed algorithms.

Životopis

Rođena sam 09.12.1995. godine u Metkoviću. Osnovnu školu don Mihovila Pavlinovića završavam 2010. godine, te upisujem Jezičnu gimnaziju Metković. Pri završetku gimnazije, 2014. godine upisujem preddiplomski studij na Prirodoslovno-matematičkom fakultetu u Splitu, smjer Matematika i Informatika. Zadnji semestar preddiplomskog studija provodim na Technische Üniversitat u Beču gdje pišem završni preddiplomski rad. 2019. godine upisujem diplomski studij na Prirodoslovno-matematičkom fakultetu u Zagrebu, smjer Računarstvo i matematika.