У Н И В Е Р З И Т Е Т У Б Е О Г Р А Д У

МАТЕМАТИЧКИ ФАКУЛТЕТ

Мастер рад на тему:

Решавање квадратног проблема придруживања применом методе променљивих околина

Кандидат: Ана Симић

Београд, 2014

Ментор:

др Александар Савић,

Доцент Математичког факултета у Београду

Чланови комисије:

др Милан Божић,

Ванредни професор Математичког факултета у Београду

др Милан Дражић,

Ванредни професор Математичког факултета у Београду

САДРЖАЈ:

Резиме

У овом раду предлаже се метода променљивих околина за решавање квадратног проблема придруживања. Дати проблем налази примену у разним областима: производња, Интернет, електроника, комуникације и обрада сигнала. На почетку је дат преглед примењених метода из литературе за решавање датог проблема, као и историјат методе променљивих околина. Приказана је и директна примена датог проблема у области образовања. Погодно изабран систем околина пермутације даје активан допринос достизању обећавајућих региона претраге. Брза метода локалног претраживања доприноси ефикасности предложене методе. Експериментални резултати садрже тестирања на стандардним инстанцама из литературе. Добијени су резултати задовољавајућег квалитета у достижном времену извршавања. У великом броју случајева су достигнута оптимална решења која су позната из литературе.

Abstract

In this work a Variable Neighborhood Search (VNS) method for solving the Quadratic Assignment Problem (QAP) is proposed. This problem has many practical applications: manufacturing, Internet, electronics, communications and signal processing. First it is given a short survey of recent methods for solving QAP from literature and VNS developments. It is given a direct application of the QAP in education. Suitable neighborhood system based on permutation representation direct search proces toward promising search regions. Fast local search procedure gives the overall efficiency of proposed method. Experimental results are performed on standard instances from the literature. Obtained results have good quality in a reasonable running time. In many cases optimal solutions from the literature are reached.

Предговор

У првој глави овог рада дата је анализа Local Search методе, затим су приказане опште информације о функционисању неколико метахеуристичких метода. Посебно је објашњен рад методе променљивих околина. У другом поглављу је дефинисан квадратни проблем придруживања, наведени су досадашњи начини његовог решавања, разложена примена у образовању, а затим је изложена примена методе променљивих околина на решавање овог проблема. Треће поглавље садржи експерименталне резултате и њихову анализу, а четврто представља закључак рада.

Користим прилику да се захвалим ментору др Александру Савићу на правим сугестијама, које су ми знатно помогле, као и на показаном разумевању и помоћи доступној у сваком тренутку у току писања овог рада. Захвалност дугујем и члановима комисије др Милану Дражићу и др Милану Божићу на пажљивом читању и корисним примедбама, које су значајно допринеле квалитету овог рада.

Желела бих на крају да се захвалим драгим родитељима и пријатељима на огромном разумевању, стрпљењу и подршци, чиме су учинили да писање овог рада протекне лакше и ефикасније.

Кандидат,

Ана Симић

1. Увод

У овој глави описане су методе које користимо за решавање главног проблема оптимизације – проблема налажења минимума, односно максимума функције. Истакнуте су поједине предности и мане различитих метода решавања овог проблема и наглашен значај примене Методе променљивих околина, као главне методе која ће даље бити коришћена у овом раду.

У склопу већине метахеуристичких алгоритама, који ће бити разматрани у наредном одељку, примењује се локално претраживање (Local Search - LS). Ова метода решавања проблема налажења максимума функције подразумева да претражујемо околину почетног решења x, затим за свако x' из те околине израчунамо вредност f(x’) и уколико је $f(x') > f(x_{max})$ од тада се чува ново $x_{max} = x'$ и $f(x_{max}) = f(x')$. Када испитамо целу околину тачке x, прелазимо у тачку x_{max} и даљу претрагу вршимо у њеној околини. Овај поступак понављамо све док у околини x_{max} не пронађемо ни једну тачку за коју важи $f(x') > f(x_{max})$.

Ово јесте брза метода, а њен основни недостатак је што се зауставља у проналажењу локалног оптимума, који најчешће није глобални оптимум, тачније то зависи од избора почетне тачке.

1.1. Метахеуристичке методе

Да би се горе поменута метода побољшала долазило се до различитих идеја, најчешће по угледу на неке процесе у природи.

Једна од тих 'побољшаних' метода јесте **Симулирано каљење (Simulated Annealing - SA)**. Ова метода је побољшана верзија LS јер омогућава потенцијалном решењу 'излазак' из локалног оптимума и то са одређеном вероватноћом која се смањује. Основна идеја је први пут примењена у тзв. Метрополис алгоритму [Met53], а коначна верзија методе је приказана у [Kir83] . Алгоритам симулираног каљења заснива се на аналогији између симулације каљења челика и проблема оптимизације. Каљење је процес у ком се материја загрева до максимума, следи лагано хлађење у ком честице губе енергију. При свакој температури материјал достиже термичку равнотежу, међутим уколико је хлађење нагло материјал неће достићи равнотежу. У алгоритму температуру представља вероватноћа са којом прихватамо ''лоша'' решења, тј она решења за која важи неједнакост $f(x') < f(x_{max})$, док се боља решења аутоматски прихватају. Што је разлика између лошијег решења $f(x')$ и до тада пронађеног најбољег $f(x_{max})$ већа, то је вероватноћа мања. Претрага се креће од највиших вредности вероватноћа (температура), чиме се осигурава груба претрага, затим се врши постепено смањивање вероватноћа (хлађење) и на тај начин долазимо, преко све финије претраге, до могуће конвергенције ка глобалном оптимуму. Главни проблем ове методе је што у почетним итерацијама постоји могућност циклирања, тј проблем цикличног враћања у исте тачке.

Мрављи алгоритми (Ant Colony Optimization - ACO), популациона метода која је први пут описана у [Dor92], је метода која аналогију проналази у следећој природној појави - социјалној интелигенцији мрава. Проучавањем колективног понашања мрава долазимо до закључка о занимљивој особини ових организама - да због недовољно

развијеног чула вида, при потрази за храном остављају за собом мирисни траг, који даје информацију о исправности путање осталим мравима који би се нашли на том путу, па тако колонија мрава има за циљ да пронађе најкраћи пут до хране тј најближи извор хране. На путу којим мрави ретко пролазе феромони ће испарити. Дакле, алгоритам се своди на то да сваки мрав генерише по једно решење x, затим на свако од њих примењујемо LS, па тако добијене локалне оптимуме упоређујемо и налазимо најбоље решење. Ниво феромона са којима се сусреће на путу одређује избор решења конкретног мрава. Њима се смањује ниво феромона у зависности од параметра који означава испаравање. Ова метода није довољно ефикасна у сегменту брзине, али су шансе да достигнемо глобални оптимум знатно повећане.

Пчелињи алгоритми (Bee Colony Optimization - BCO) је популациона метода, слично као и код ACO, инспирисана понашањем организама у потрази за храном, први пут предложена 2001. године од аутора Лучић и Теодорић у раду [Luc01]. Како само име алгоритма каже, симулира социјално понашање пчела. Оне узорке хране доносе у кошницу. Ти узорци представљају потенцијална решења проблема. Затим се врши испитивање квалитета узорака, при чему свака пчела хвали своју храну дајући при том и информације о месту извора. На крају размене информација, тачније на крају сваког лета, свака од пчела се одлучује да ли наставља да хвали своје решење или се предомишља па преузима решење неке друге пчеле. Коришћењем колективног знања и међусобном разменом информација, после одређеног броја летова, током напредовања процеса претраживања, пчеле се све више окупљају око бољег решења, постепено напуштајући своја решења уколико су лоша. Постоји више варијација ове методе, а горе наведена је најпознатија и највише коришћена у пракси.

Генетски алгоритми (Genetic Algorithms - GA), су представници популационих метода са којим се најчешће сусрећемо. Концепт модерног GA први пут се помиње у [Hol75], иако компјутерску симулацију еволуције можемо приметити још 1954. године у

Принстону. Ова метода претражује разне делове области претраге, а добијена решења укршта у сврху добијања још бољег од изабрана два, баш као у природи - генетско укрштање две врсте у сврху добијања отпорнијих и јачих потомака. Тачније, свако решење симулира једну јединку, састављену од комплекса гена – репрезентација јединки, где се јединка са k гена представља вектором $(x_1,...,x_k)$. Први корак је процена организама у смислу ефикасности и прилагођености околини, а оператором селекције бирају се организми који ће учествовати у следећем кораку – укрштању. Како је врло могуће да приликом укрштања добијемо потомке који су лошији од родитеља, користе се методе селекције којима би се ова појава избегла. Један од начина је увођење тзв. стационарног генетског алгоритма. Узастопном применом селекције и укрштања се, током генерација, добијају све боље и боље јединке. Међутим, да би избегли конвергенцију ка релативно лошем локалном оптимуму морају се применити још неке технике међу којима је најважнији оператор мутације, који доприноси повећању разноврсности генетског материјала. Критеријум заустављања овог алгоритма може бити максималан број генерација или број генерација за који се није променила најбоља јединка.

Постоји још врста оптимизационих метода заснованих на процесима из природе, као што су: имуни систем, систем преживљавања, итд. Као и још популарних популационих метода, заснованих на понашању честица, бактерија, риба и других организама.

Следећа идеја за побољшање локалног претраживања је **табу претраживање (Tabu Search - TS)**, први пут описана 1989. у раду Fred Glower-a [Glow89], једино од до сада поменутих претраживања које није аналогно повезано са неким конкретним процесом у природи. Идеја се своди на то да при локалном претраживању решењу дозволимо да ''шета'' из околине, тако што ћемо после одабраног локалног оптимума вредности променљиве доделити неку другу тачку, ван претходно изабране околине, која вероватно није боља од претходно одабраног оптимума $f(x_{max})$, а затим аналогно даљу претрагу

вршити у њеној околини, памтећи при том процесу све тачке чије вредности додељујемо променљивој, да бисмо смањили могућност враћања у исту тачку. Околине тачака у које се забрањује повратак на неко време чувамо у такозваној Табу листи, која је ограничене дужине.

1.2. Метода променљивих околина

Метода променљивих околина (Variable neighborhood search - VNS) је такође метахеуристичка метода коју ћемо надаље користити за решавање проблема. Настала 1997. од аутора Младеновића и Хансена и први пут је описана у раду [Mla97].

Идеја алгоритма ове методе своди се на следећи поступак. Претрагу уобичајено крећемо од неке почетне тачке, у којој дефинишемо низ околина. Следећа фаза је такозвано размрдавање (Shaking), у којој се претражује нека k-та околина полазне тачке. Идеја је да се, уколико претраживање у блиској околини не доведе до решења, изврши прелазак у нову тачку, која је даље од тренутног локалног оптимума. Овај корак је дакле случајан одабир нове тачке x' из дате околине. На њу даље примењујемо LS, при чему добијамо нову тачку x'' и вредност f(x''). У зависности од тога да ли је ново решење x'' боље од актуелног локалног оптимума x, претрагу настављамо на један од следећих начина:

1) Уколико важи неједакост f(x'') > f(x) тада се наставак алгоритма аналогно врши са новом почетном тачком x'', x=x''.
2) У случају да је ипак боље решење већ актуелни оптимум, тј. ако важи неједнакост f(x'') < f(x) претрагу настављамо са почетном тачком x.

3) Ако су вредности локалног оптимума и новог решења исте, тј. $f(x'') = f(x)$, бирамо x'' за нову тачку у којој настављамо алгоритам, $x=x''$, и то са одређеном вероватноћом коју означавамо са p. Овим могућност доласка до циклирања тј враћања у исту тачку, сводимо на миниум. Уколико је l дужина ланца, онда је вероватноћа да ће до циклирања доћи p^l.

У првом случају претрага се наставља у почетној околини k_{min}, а у другом тако што изаберемо (k+1)- у околину уколико је $k<k_{max}$, а остајемо у k-тој уколико је $k=k_{max}$. У трећем случају са вероватноћом p претрага се наставља анлогно првом случају, а са вероватноћом 1-p аналогно другом случају. Систем околина које бирамо зависи од природе проблема и пресудно утиче на рад методе. Изабраћемо их тако да кардиналност сваке следеће буде већа од кардиналности претходне.

Остали детаљи разних варијанти ове методе, као и преглед примена у пракси излазе изван оквира овог рада и могу се наћи у прегледним радовима [Han08] и [Han10]

2. Квадратни проблем придруживања

Квадратни проблем придруживања, QAP (quadratic assignment problem) је један од основних проблема оптимизације. Овај проблем се своди на одређивање најбоље могуће варијанте распоређивања n елемената на n различитих локација. При томе, дефинисан је међусобна интеракција елемената и свим локацијама на које елементи треба да се распореде, задато је међусобно растојање. Приликом распоређивања елемената на одређене позиције потребно је да сума производа међусобних односа елемената и растојања између одговарајућих локација буде што је могуће мања. Математичка формулација овог проблема изгледала би овако :

$$\min_{\pi \in S_n} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} \cdot d_{\pi(i)\pi(j)}$$

при чему је S_n скуп свих пермутација са n елемената, f_{ij} однос између елемената i и j, а d_{km} је растојање између места k и места m. Напоменимо да је π једна пермутација из скупа S_n која представља распоређене елементе на одређене локације.

Једноставан пример овог проблем изгледао би овако:

Узмимо да је F матрица свих елемената f_{ij}, а D матрица чији су елементи d_{ij}.

$$F = \begin{bmatrix} 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad D = \begin{bmatrix} 2 & 3 & 1 & 1 & 1 \\ 4 & 1 & 2 & 1 & 2 \\ 3 & 3 & 2 & 2 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Дата је почетна пермутација x=(3 2 5 4 1). За ову пермутацију, функција би имала следећу вредност: $f(x)=f_{12}* d_{\pi(1)\pi(2)}+ f_{33}* d_{\pi(3)\pi(3)}+ f_{51}* d_{\pi(5)\pi(1)}=2*d_{32}+1*d_{55}+3*d_{13}=10$

Наиме, сви остали елементи матрице F су нуле, сходно томе би и вредности производа били једнаки нули.

Како се проблематика своди на тражење минимума функције, закучујемо да би у овом примеру то била вредност 6 и то у случајевима када сваки од елемената: $d_{\pi(1)\pi(2)}$, $d_{\pi(3)\pi(3)}$ и $d_{\pi(5)\pi(1)}$, узима вредност 1.

Предложени проблем се може дефинисати као проблем целобројног програмирања. Иако постоји више разних таквих формулација, њихов приказ и детаљна анализа далеко превазилазе обим овог рада. Због тога ће бити приказана само једна таква формулација целобројног квадратног програмирања. Детаљне информације, преглед и анализа разних математичких формулација за дати проблем могу се наћи у прегледном раду [Loi07].

Уколико бисмо дефинисали бинарне променљиве:

$$x_{ik}=\begin{cases}1, & k=\pi(i)\\ 0, & k\neq\pi(i)\end{cases}$$

Тада би овај проблем могао да се формулише на следећи начин:

$$\min\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ij}\cdot d_{kl}\cdot x_{ik}\cdot x_{jl}$$

$$\sum_{i=1}^{n} x_{ij}=1 \quad j=1,2,\ldots,n$$

$$\sum_{j=1}^{n} x_{ij}=1 \quad i=1,2,\ldots,n$$

$$x_{ij}\in\{0,1\} \quad i,j=1,2,\ldots,n$$

2.1. Досадашњи начини решавања QAP

Имајући у виду огроман број постојећих радова у литератури који разматрају QAP и ограничени обим овог рада, биће поменуто само неколико најновијих радова.

У раду [Dru14] за решавање овог проблема од стране аутора M. Drugan и E. Talbi предложен је такозвани адаптивни мулти-оператор, који ефективно одређује када и како рестартовати методу итеративног локалног претраживања, како не би конвергирала ка лошем локалном оптимуму. Експериментални резултати садрже резултате извршавања на стандардним QAP инстанцама.

Група аутора у раду [Han14] на проблем квадратног придруживања примењују генетски алгоритам са локалним претраживањем. За избор оператора укрштања користе се Марковљеви процеси који поспешују ток еволуционе претраге. Предложени модел Марковљевих процеса решава се такозваном методом Q-учења (Q-learning). Ефикасност овог приступа експериментално је проверена на појединачним случајевима квадратног проблема придруживања.

Аутори W. Adams и L. Waddell у раду [Ada14] користе, како и сам наслов рада каже, линеарно програмирање као увид у ефикасно решиве случајеве квадратног проблема придруживања. У раду се говори о самом проблему, као врло важном сегменту оптимизације, који је предмет изучавања већ претходних 50 година, као и о његовим различитим применама у многим областима. Рад се базира на проучавању четири

посебна случаја где су могуће ефикасне стратегије решавања овог проблема. Следе објашњења која омогућавају поједностављње или генерализацију услова који дефинишу ове специјалне случајеве.

У раду [Tos13] група аутора наводи значај квадратног проблема придруживања. Аутори истичу да је овај проблем са 10-20 елемената могуће егзактно решити на вишепроцесорским рачунарима у року од неколико дана, користећи паралелне алгоритме. Међутим, проблем није решив на овај начин уколико има око 100 или више елемената. У раду се износи истраживање различитих варијација паралелних генетских алгоритама у сврху решавања овог проблема, користећи добро познате инстанце библиотеке QAPLIB. Повећањем димензије проблема уз повећање броја процесора и величине популације аутори су пронашли решења која су упоредива са најбољим познатим решењима из литературе. У циљу да се упореде генетски алгоритми, на сваком појединачном процесору се извршава посебан генетски алгоритам, при чему је коришћен модел острва. Овај модел мења десет процената решења предложених у иницијалним фазама оптимизације. На крају се експериментално показује побољшање које овај алгоритам доноси при решавању квадратног проблема придруживања са великим бројем елемената.

У раду [Tas13] описују се два метахеуристичка алгоритма за решавање квадратног проблема придруживања. Први алгоритам се заснива на итеративној примени тзв. ''похлепне'' (greedy) хеуристике која има две компоненте – уништавање и изградња. Састоји се из неколико корака који се итеративно понављају, све док су сви од унапред постављених критеријума испуњени. Такође се представља други, такозвани дискретни диференцијални еволуциони алгоритам, прилагођен специјално за решавање датог проблема. Овај рад је богат примерима из праксе, при чему се објашњава зашто су предложени алгоритми упоредиви у односу на друге поменуте методе за решавање предложеног проблема.

2.2. Примена проблема у образовању

Дати проблем налази примену у разним областима: електроника, Интернет, комуникације, обрада сигнала, производња итд. Овде ће бити разложене поједине примене овог проблема у методици наставе математике и образовању уопште.

Први пример овог проблема, са којим се свакодневно сусрећу наставници у школама, изнет је у наставку и односи се на распоређивање одељења по кабинетима у току једне радне недеље.

Уколико сва одељења у школи посматрамо као елементе нашег квадратног проблема, јасно је да је локација проблема кабинет који је потребно доделити сваком одељењу, тј појединачно сваком елементу проблема. Онда би сваки распоред одељења по кабинетима представљао једну пермутацију елемената S_n, где n означава број кабинета, односно број одељења. Са f_{ij} дефинишемо број ђака који у одређеном тренутку, пре почетка одређеног часа, прелазе из кабинета i у кабинет j. Како из неких предмета постоји више кабинета, да би се избегле компликације, посматраћемо сваки од кабинета појединачно, баш као да се ради о различитим предметима, чиме избегавамо појаву пермутација са понављањем. Уколико за функцију циља при распоређивању ученика по кабинетима дефинишемо укупан пређени пут свих ученика, њеном минимизацијом се смањује могућност инцидената и конфликата између ђака на школским одморима, а уједно даје ученицима више времена за припрему пред наредни час.

Следећи пример овог проблема, лако препознатљив наставницима математике, можемо уочити уколико посматрамо проблем тражења најбољег могућег редоследа предавања лекција у току једне школске године. Узмимо да појединачна лекција представља елемент претходно дефинисаног квадратног проблема. Термин за предавање појединачне лекције, тј одређени час у школској години, представља локацију коју придружујемо елементу. Онда је сваки редослед предавања лекција заправо једна пермутација ових елемената, S_n, где је n број лекција, тј број часова. Сваке две појединачне лекције, i $\in \{1,2,\dots,n\}$ и j $\in \{1,2,..,n\}$, узајамно су дефинисане коефицијентом каузалности f_{ij}. Овај коефицијент одређујемо узимајући у обзир садржај лекција. Што је су сличније лекције по садржају, њихов коефицијент каузалности биће већи. Растојање између две локације, које у овом случају представља удаљеност два термина за предавање лекција, дефинишемо као значајност - блискост лекција i и j по термину и означавамо га са $d_{ij.}$ Иако постоје разне могућности за задавање дате формуле, најједноставнији облик би изгледао као:

$$d_{ij} = \frac{1}{|i-j|}$$

Јасно је да што су лекције ближе по термину значајност је већа, а уколико су суседне она достиже свој максимум.

Битно је разјаснити појам коефицијента каузалности, за који јесте неопходна сличност, међутим, није и довољна. На пример, области:

- Квадратна функција
- Експоненцијална и логаритамска функција
- Тригонометријске функције

јесу сличне, али је коефицијент каузалности врло мали, јер су излагања једне од поменутих лекција скоро независна од излагања осталих.

Са друге стране, узмимо за пример области, означавајући прву са i а другу са j:

- Квадратне једначине
- Квадратна функција

Овде примећујемо да је f_{ij} највеће могуће, а f_{ji} блиско нули, јер да бисмо излагали и проучавали област Квадратне функције скоро да је неопходно познавати област Квадратне једначине, док за упознавање ученика са појмовима прве области скоро да није важно да ли су претходно упознати са појмовима квадратне функције .

Приликом тражења најпогоднијег решења, закључујемо да је повољно да што већи број суседних лекција буде блиско по садржају, тј да је за појединачне лекције коефицијент каузалности f_{ij} већи. Такође је јасно да је потребно да што већи број лекција буде блиско по термину, тачније да значајност d_{ij} буде већа. Јасно је, дакле, да ћемо функцију преформулисати, јер се у квадратном проблему придруживања ради о минимизацији, а нама је потребна максимимизација. Ово решавамо тако што ћемо сваки од елемената f_{ij} помножити са коефицијентом -1.

2.3. Решавање QAP методом променљивих околина

У овом одељку биће представљена имплементација методе променљивих околина за решавање квадратног проблема придруживања.

Први корак је дефинисање погодне k-околине у којој ће се вршити такозвани процес размрдавања. Околине дефинишемо на следећи начин. На почетку бирамо к случајних позиција у пермутацији. Затим пермутујемо елементе са изабраних позиција, такође на случајан начин. На тај начин се добија нова пермутација, која се разликује од претходне у највише k елемената.

Како су пермутације низови елемената над којима се врши ова метода, јасно је да је неопходно да k буде мање или једнако n, при чему је n број елемената пермутације. У случају да се ради са пермутацијама које имају велики број елемената, k_{max} смо ограничили на следећи начин: $k_{max} = \min\{n,30\}$. Такође k не може бити мањи од 2, а све околине величине 2 већ разматрамо у локалном претраживању, па је дефинисано $k_{min}=3$. Означимо са $N^k(x)$ скуп свих могућих решења из k-те околине тачке x. Околине се најчешће бирају тако да задовољавају следећу неједнакост о кардиналности: $|N^{kmin}(x)| < |N^{kmin+1}(x)| <\ldots< |N^{kmax}(x)|$. Јасно је да применом ове методе од једне пермутације добијамо такође пермутацију истих елемената, при чему је k! број могућих мешања које можемо применити.

Комплетна метода примењена на овај проблем описана је на следећи начин. Дата је пермутација $x=(x_1,x_2,...,x_n)$. За дате вредности k, које задовољавају $3\leq k\leq n$, у пермутацији на случајан начин бирамо k_{min} елемената и вршимо Shaking, мешајући тачно тих k_{min} елемената, док остали елементи пермутације остају непромењени. Исход је пермутација x'. Потом уводимо променљиву x'', којој иницијално додељујемо вредност x'. У наредном кораку упоређујемо вредности f(x'') и f(x), при чему испитујемо да ли ће доћи до замене

почетне пермутације x са x''. Основа ове методе је прелазак са једног решења на друго, при чему је јасно да никада не прелазимо са бољег на горе решење, већ искључиво обрнуто. Дакле, ако важи f(x'') < f(x), сматрамо бољом новодобијену пермутацију и додељујемо јој улогу почетне x, x = x''. Ако насупрот, важи неједнакост f(x'') > f(x), ново решење није боље и до замене неће доћи. Проблем се јавља само у случају када су ове вредности једнаке. Уколико бисмо задржали променљиву на првобитном решењу, шансе да се постигне напредак у наставку алгоритма биле би смањене. Међутим, уколико би сваки пут када су вредности минимума функције једнаке, мењали вредност пермутације x, постојала би велика вероватноћа да ћемо поново доћи у исто решење. Сходно томе, да бисмо вероватноћу циклирања свели на минимум, уводимо у алгоритам нову функцију, коју ћемо позивати увек у оваквим случајевима. Она ће ипак вршити замену x са x'', али са одређеном вероватноћом p. Уколико до замене дође, претрага се наставља у k_{min} околини, у супротном претраживање се наставља са околином k+1, уколико је $k<k_{max}$, а остајемо у k-тој уколико је $k=k_{max}$.

У следећем кораку, на тако добијену пермутацију примењујемо LS, на начин описан у наставку. Почетни корак је замена прва два елемента пермутације, па вредност функције новодобијене пермутације упоређујемо са актуелним и уколико је вредност коју има функција нове пермутације мања од вредности актуелне, тада за x'' именујемо новопронађену пермутацију. Уколико је до замене дошло, претрага се врши од почетка, а ако се замена не изврши, настављамо локалну претрагу тако што место мењају први и трећи елемен актуелне пермутације и аналогно вршимо испитивање. Претрага ће се вршити све док не извршимо замене свака два елемента пермутације. Локално претраживање имаману што сваки пут за нову пермутацију претрагу врши од почетка.

Критеријум заустављања дате методе променљивих околина је тренутак када достигнемо *maxit* итерација. У том тренутку штампа се актуелно решење *x*.

3. Експериментални резултати

Изложени експериментални резултати добијени су на рачунару Toshiba Satellite C55-A-172, са процесором Intel (R) Celeron (R), 1005M, 1.9GHz, који има 1 језгро, а метода променљивих околина имплементирана је у програмском језику C.

У табелама 1 - 5 резултати су тестирани на инстанцама малих димензија. Прва колона у овим табелама садржи назив инстанци, у којима је означена димензија проблема. У другој колони је дато оптимално решење приказано у QAPLIB (http://anjos.mgi.polymtl.ca/qaplib). Најбољи и просечни добијени резултати методе променљивих околина у свих двадесет извршавања дати су редом у трећој и четвртој колони. Колона **gr** означава релативну грешку просечног решења у односу на оптимално решење, дату у процентима, а **sigma** представа његово средње квадратно одступање, такође у процентима. Колона **t** је време извршавања дато у секундама.

У табелама 6 - 7 су дати резултати извршавања на инстанцама већих димензија. Подаци су представљени на сличан начин, с тим што оптимално решење није познато, па се у одговарајућој колони приказује најбоље познато решење из литературе, а и релативна грешка се рачуна у односу на то решење методе. Број итерација је maxit=1000, p=0.4, а околине су задате на следећи начин: $k_{min}=3$, $k_{max}=\min\{30,n\}$.

Табела 1 Решења за инстанце где је познато оптимално решење

Inst	Opt	Rez	Sr	gr (%)	sigma (%)	t
bur26a	5426670	opt	5434464,45	0,144	0,062	0,96705
bur26b	3817852	3821357	3825184,45	0,192	0,039	0,895275
bur26c	5426795	opt	5429645,60	0,053	0,088	1,05465
bur26d	3821225	opt	3824524,90	0,086	0,126	0,9939
bur26e	5386879	opt	5389350,00	0,046	0,070	1,09275
bur26f	3782044	opt	3784775,70	0,072	0,117	1,09845
bur26g	10117172	10118177	10120378,65	0,032	0,015	1,1442
bur26h	7098658	7098905	7108469,20	0,138	0,202	1,0932
chr12a	9552	opt	9801,80	2,615	2,636	0,17145
chr12b	9742	opt	9760,00	0,185	0,804	0,17085
chr12c	11156	opt	11505,20	3,130	2,474	0,16095
chr15a	9896	9978	10548,40	6,546	4,690	0,282225
chr15b	7990	opt	8592,20	7,537	6,281	0,304275
chr15c	9504	9780	11270,70	18,146	5,960	0,2445
chr18a	11098	opt	12997,20	17,113	7,858	0,443175
chr18b	1534	opt	1561,80	1,812	1,622	0,455025
chr20a	2192	2354	2547,00	15,590	4,066	0,633525
chr20b	2298	2508	2653,60	14,943	3,268	0,572175
chr20c	14142	opt	16570,50	17,172	9,744	0,85725
chr22a	6156	6230	6456,20	4,833	1,555	0,99345
chr22b	6194	6310	6540,90	5,532	1,587	0,75705
chr25a	3796	4048	4612,00	20,572	5,771	1,5357

Табела 2 Решења за инстанце где је познато оптимално решење

Inst	opt	rez	Sr	gr (%)	sigma (%)	t
els19	17212548	17937024	17942895,40	4,242	0,143	0,638175
esc16a	68	opt	68,00	0,000	0,000	0,384975
esc16b	292	opt	292,00	0,000	0,000	0,46305
esc16c	160	opt	160,00	0,000	0,000	0,37305
esc16d	16	opt	16,00	0,000	0,000	0,455925
esc16e	28	opt	28,00	0,000	0,000	0,42105
esc16g	26	opt	26,00	0,000	0,000	0,344175
esc16h	996	opt	996,00	0,000	0,000	0,2184
esc16i	14	opt	14,00	0,000	0,000	0,3183
esc16j	8	opt	8,00	0,000	0,000	0,464325
esc32e	2	opt	2,00	0,000	0,000	6,0432
esc32f	2	opt	2,00	0,000	0,000	6,028425
had12	1652	opt	1652,00	0,000	0,000	0,192525
had14	2724	opt	2724,00	0,000	0,000	0,29565
had16	3720	opt	3720,30	0,008	0,019	0,4365
had18	5358	opt	5359,20	0,022	0,067	0,542625
had20	6922	opt	6925,00	0,043	0,106	0,77505

У првој табели дата су оптимална решења, која су постигнута у 12 случајева. Чак и када није постигнуто оптимално решење, грешка у процентима је мала, осим у појединачним случајевима. Највећа грешка прави се код инстанци chr18a и chr20c. За ове две табеле примећујемо да су решења добијена у врло кратком времену. У првој не прелазе 1,5357 секунди, које је потребно за извршавање инстанце chr25a .

У другој табели, гледајући овај параметар, постоје две инстанце које се издвајају, els19 и esc32f, чије време прелази 6 секунди. Посматрајући резултате, уочавамо да се у другој табели сви резултати, осим у случају инстанце els19, поклапају са оптималним решењем.

Табела 3 Решења за инстанце где је познато оптимално решење

Inst	Opt	Rez	Sr	gr (%)	sigma (%)	t
esc32a	130	134	140,70	8,077	2,781	2,254
esc32b	168	168	176,40	5,000	5,952	2,987
esc32c	642	642	642,00	0,000	0,000	1,540
esc32d	200	200	200,40	0,200	0,399	1,939
esc32g	6	6	6,00	0,000	0,000	2,997
esc32h	438	438	438,60	0,137	0,209	2,627
esc64a	116	116	116,00	0,000	0,000	37,729
esc128	64	64	64,30	0,469	1,111	754,894
kra30a	88900	88900	91661,00	3,106	1,300	2,245
kra30b	91420	92160	92871,00	1,580	0,587	2,383
lipa20a	3683	3683	3722,35	1,068	0,879	0,425
lipa20b	27076	27076	27076,00	0,000	0,000	0,526
lipa30a	13178	13178	13374,20	1,489	0,496	1,679
lipa30b	151426	151426	159268,00	5,179	6,735	1,650
lipa40a	31538	31873	31921,50	1,214	0,081	4,954
lipa40b	476581	476581	513923,60	7,836	8,037	4,854
lipa50a	62093	62093	62742,30	1,046	0,241	10,875
lipa50b	1210244	1210244	1369308,35	13,143	6,709	11,294
lipa60a	107218	108146	108207,45	0,923	0,029	25,056
lipa60b	2520135	2520135	2972917,20	17,967	3,498	22,604
lipa70a	169755	171072	171140,25	0,816	0,024	44,014
lipa70b	4603200	4603200	5463496,00	18,689	3,618	42,905
lipa80a	253195	254830	255012,10	0,717	0,030	74,654
lipa80b	7763962	7763962	9122501,75	17,498	6,257	75,561
lipa90a	360630	362877	363036,05	0,667	0,021	118,166
lipa90b	12490441	15061886	15092531,85	20,790	0,110	113,957

Табела 4 Решења за инстанце где је познато оптимално решење

Inst	Opt	Rez	Sr	gr (%)	sigma (%)	t
nug12	578	578	578,40	0,069	0,207	0,107
nug14	1014	1014	1017,80	0,375	0,304	0,145
nug15	1150	1150	1150,60	0,052	0,080	0,187
nug16a	1610	1610	1622,20	0,758	0,527	0,214
nug16b	1240	1240	1242,40	0,194	0,580	0,246
nug17	1732	1732	1741,60	0,554	0,413	0,271
nug18	1930	1930	1945,10	0,782	0,413	0,346
nug20	2570	2570	2590,50	0,798	0,650	0,500
nug21	2438	2438	2449,30	0,463	0,314	0,566
nug22	3596	3596	3617,70	0,603	0,487	0,689
nug24	3488	3488	3503,90	0,456	0,382	1,014
nug25	3744	3744	3760,70	0,446	0,334	1,231
nug27	5234	5234	5298,60	1,234	0,785	1,162
nug28	5166	5190	5247,20	1,567	0,455	1,333
nug30	6124	6144	6191,10	1,094	0,547	2,467
rou12	235528	235528	236157,20	0,267	0,551	0,089
rou15	354210	354210	359826,50	1,586	0,906	0,172
rou20	725522	725582	732326,90	0,938	0,581	0,383
scr12	31410	31410	31410,00	0,000	0,000	0,070
scr15	51140	51140	51669,50	1,035	1,793	0,137
scr20	110030	110058	111777,70	1,588	0,855	0,318
ste36a	9526	9628	9887,50	3,766	1,771	4,240
ste36b	15852	16098	16969,00	6,963	3,164	4,154
ste36c	8239110	8318122	8471605,20	2,804	1,628	4,284

Табела 5 Решења за инстанце где је познато оптимално решење

Inst	Opt	Rez	Sr	gr (%)	sigma (%)	t
tai10a	135028	135028	135058,60	0,023	0,099	0,092
tai10b	1183760	1183760	1183760,00	0,000	0,000	0,066
tai12a	224416	224416	226419,20	0,893	1,368	0,102
tai12b	39464925	39464925	39552017,00	0,221	0,440	0,106
tai15a	388214	388214	391007,40	0,720	0,466	0,193
tai15b	51765268	51765268	51812767,25	0,092	0,101	0,164
tai17a	491812	491812	499266,90	1,516	0,786	0,223
tai20a	703482	706786	719171,50	2,222	0,824	0,361
tai20b	122455319	122455319	123251150,85	0,650	0,479	0,437
tai25a	1167256	1183573	1193187,50	2,210	0,417	1,042
tai25b	344355646	344355646	347814678,70	1,004	1,309	1,041
tai30b	637117113	646061548	663748876,30	4,142	2,821	2,593
tho30	149936	150454	151789,40	1,233	0,574	1,942

Када посматрамо трећу табелу, примећујемо да време варира од 0,425 секунди, постигнуто у случају lipa20a инстанце, до 754,894 секунди, постигнуто за инстанцу esc128 . Највећа вредност грешке у овој табели достигнута је у случају инстанце lipa90b и износи 20,790 процената.

Увиђамо да се време извршавања у четвртој и петој табели се креће у временском интервалу од 0,066 до 4,284 секунди, где су ''екстремне вредности'' постигнуте редом у инстанцама tai10b и ste36c, а максимална достигнута вредност грешке износи 6,963 процената.

Табела 6 Решења за инстанце веће димензије

Inst	Najbolje	Rez	Sr	gr (%)	sigma (%)	t
sko42	15812	15858	16009,10	1,244	0,607	10,583
sko49	23386	23464	23616,90	0,986	0,349	21,018
sko56	34458	34636	34868,60	1,189	0,511	34,910
sko64	48498	48776	49094,40	1,226	0,300	64,087
sko72	66256	66744	67094,10	1,262	0,424	100,084
sko81	90998	91590	92106,30	1,215	0,379	167,419
sko90	115534	115930	116819,40	1,110	0,379	235,255
sko100a	152002	152512	153818,80	1,193	0,409	342,646
sko100b	153890	154384	155620,00	1,122	0,336	370,004
sko100c	147862	148988	149747,80	1,272	0,357	371,264
sko100d	149576	150668	151425,50	1,233	0,327	373,749
sko100e	149150	150268	151184,10	1,360	0,321	357,102
sko100f	149036	150134	150740,90	1,141	0,254	367,212

У табелама 6 и 7 ради се са инстанцама већих димензија, сходно томе време извршавања је приметно веће, посебно у појединачним случајевима.

У шестој табели најдуже време извршавања постиже се за инстанцу sko100d и износи 373,749 секунди. Увиђа се и да се највећа грешка прави се код инстанце sko100e и износи 1,36 процената.

Табела 7 Решења за инстанце веће димензије

Inst	Najbolje	Rez	Sr	gr (%)	sigma (%)	t
tai30a	1818146	1855412	1870275,10	2,851	0,465	1,418
tai35a	2422002	2467174	2492780,40	2,903	0,487	2,373
tai35b	283315445	284531699	293371492,60	3,536	2,234	3,950
tai40a	3139370	3195106	3235747,00	3,047	0,455	4,369
tai40b	637250948	637250948	668016388,70	4,828	2,357	8,591
tai50a	4938796	5060050	5095230,30	3,150	0,364	10,959
tai50b	458821517	462215064	470579470,90	2,550	1,509	16,353
tai60a	7205962	7381712	7436690,30	3,184	0,320	22,301
tai60b	608215054	608684370	625684759,20	2,870	1,842	54,996
tai64c	1855928	1855928	1855928,00	0,000	0,000	25,020
tai80a	13499184	13813164	13885487,00	2,850	0,255	70,158
tai80b	818415043	823373409	848506987,60	3,659	1,212	142,393
tai100a	21052466	21484680	21598123,80	2,581	0,247	167,642
tai100b	1185996137	1192684655	1229265801,65	3,631	1,550	365,378
tai150b	498896643	507586604	511938322,80	2,599	0,585	2050,076
tai256c	44759294	44839216	44873574,90	0,256	0,034	6857,238
tho40	240516	242456	244937,30	1,830	0,553	6,747
tho150	8133398	8236796	8281129,10	1,809	0,355	1882,277
wil50	48816	48878	49058,70	0,497	0,223	13,549
wil100	273038	274108	274940,00	0,696	0,210	246,022

У седмој табели најмање време извршавања постигнуто је у случају tai30a инстанце и оно износи 1,418 секунди. Насупрот томе, најдуже време извршавања постиже се у случају tai256c инстанце са највећом димензијом проблема у табели, kra32 и износи 6857,238.

4. Закључак

У овом раду је приказана метода променљивих околина која је прилагођена за решавање квадратног проблема придруживања. Дати приступ је примењен на овај NP-тежак проблем, који има значајну примену у пракси. Такође је дат преглед досадашњих новијих метода за решавање овог проблема.

Такође су описана два примера конкретне примене датог проблема у области образовања. Први пример се односи на проблем редоследа предавања лекција у току једне школске године, а други представља проблем распореда одељења по учионицама у току једне школске недеље.

Дати проблем је тежак за решавање и то је главни разлог зашто постојеће егзактне методе могу решити само инстанце релативно малих димензија. Због тога је предложена метода променљивих околина врло значајна јер се добијају оптимална или квалитетна субоптимална метехуристичка решења на свим инстанцама из литературе у разумном времену извршавања. Пројектовани систем околина је директно прилагођен решаваном проблему решењима која имају облик пермутације. На тај начин су сва добијена решења коректна. Процедура размрдавања је на природан начин примењена на дати систем околина. Такође је ефикасно имплементирана метода локалног претраживања.

Правци даљег развоја може бити:

- Прилагођавање дате имплементације за извршавање на моћнијим вишепроцесорским рачунарским системима

- Модификација предложене методе за решавање неких сличних проблема, као на пример би-квадратног проблема придруживања
- Решавање датог проблема помоћу неких других метахеуристика
- Коришћење ове методе уз евентуалне ситне измене за решавање неких конкретних проблема из области образовања.

Литература

[Ada14] Adams W., Waddell L., “Linear programming insights into solvable cases of the quadratic assignment problem”, Discerte Optimization, Vol. 14, pp. 46-60 (2014).

[Dor92] Dorigo M., Stützle T., “Ant Colony Optimization: Overview and Recent”, Ph.D.Thesis, Politecico di Milano (1992).

[Dru14] Drugan M., Talbi E., ''Adaptive Multi-operator Meteheuristic for QAP'', Evolve - A Bridge between Probability, Set Oriented Numeric, and Evolutionary ComputationV Advances in Intelligent System and Computing, Vol. 288, pp. 149-163 (2014).

[Glo89] Glower F., “Tabu search - part one”, ORSA Journal of computing, Vol. 1, No 3 (1989).

[Kir83] Kirkpatrick S., Gellat C., Vecchi M., "Optimization by simulated annealing", Science, Vol. 220, pp. 671-680 (1983).

[Luc01] Lucic P., Teodorovic D., ''Bee System: Modeling Combinatorial Optimization Transportation Engineering Problems by Swarm Intelligence'', In The Preprints of the TRISTAN IV Triennial Symposium on Transportation Analysis, pp. 441-445 (2001).

[Han08] Hansen P., Mladenovic N., Moreno Perez JA., "Variable neighborhood search: methods and applications", 4OR, Vol. 6, pp. 319-360 (2008).

[Han10] Hansen P., Mladenovic N., Moreno Perez JA., "Variable neighborhood search: algorithms and applications", Annals of Operations Research, Vol. 175, pp. 367-407 (2010).

[Han14] Handoko S. D., Nguyen D. T., Yuan Z., Chuin Lau H., "Reinforcement learning for adaptive operator selection in memetic search applied to quadratic assignment problem", GECCO Comp 14, Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion, pp. 193 - 194 (2014).

[Hol75] Holland J.H., "Adaptation in Natural and Artificial Systems", The University of Michigan Press, Ann Arbor (1975).

[Huss14] Saifullah Hussin M., Stutzle T., "Tabu search vs Simulated annealing as a function of the size of the quadratic assighment problem instances", Computers & Operations Research, Vol. 43, March 2014, pp. 286-291 (2014).

[Mla97] Mladenovic N., Hansen P.., "Variable neighborhood search", Computers & Operations Research, Vol. 24, No. 11, pp. 1097–1100 (1997).

[Met53] Metropolis N., Rosenbluth A.W., Rosenbluth M.N., Teller A.H., Teller E., "Equation of state calculation by fast computing machines", Journal of Chemical Physics, Vol. 21, pp. 1087-1091 (1953).

[Tas13] Tasgetiren M.F., Suganthan P.N. , Dizbay I.E., "Metaheuristic algorithms for the quadratic assignment problem", Computational Intelligence In Production And Logistics Systems (CIPLS), 2013 IEEE Workshop on, pp. 131-137 (2013).

[Tos13] Tosun U., Dokeroglu T., Cosar A., "A robust Island Parallel Genetic Algorithm for the Quadratic Assignment Problem", International Journal of Production Research, Vol 51, pp. 4117-4133 (2013).