Итеративна локална претрага

Никола Мајсторовић 354/2019

1. Увод

Итеративна локална претрага је метахеуристички алгоритам оптимизације који се широко користи за решавање сложених комбинаторних проблема. Комбинује локалну претрагу са итеративним пертурбацијама како би избегао заробљавање у локалним минимумима. Први пут се спомиље у раду “Lourenço, Martin i Stützlea (2001) - A beginner’s introduction to Iterated Local Search”.
У овом раду ћу објаснити основне концепте итеративне локалне претраге и њене кључне компоненте. Затим ћу описати различите верзије итеративне локалне претраге за решавање проблема трговачког путника које сам имплементирао и резултате њихове примене на познатим ТСП скуповима података.

Итеративна локална претрага представља проширење алгортма локалне претраге. Локална претрага се састоји из налажења иницијалног решења, обично неком конструктивном или стохаистичком хеуристиком, у чијем “комшилуку” се врши претрага док се не нађе локално оптимално решење. Комшилук се дефинише као скуп решења која могу да се добију од тренутног решења применом малих модификација.
Главни проблем локалне претраге је што локални оптимум који претрага нађе није и глобални оптимум. Потребно је некако побећи из локалног оптимума, покретањем локалне претраге више пута за различито иницијално решење. Уколико насумично бирамо нова иницијална решења добијамо насумичну претрагу, познату као “Random Restart” приступ који се лоше показује на већим скуповима. Уместо насумичног поновног покретања, врши се пертурбација тренутног најбољег решења и онда се на то пертурбирано решење примењује локална претрага. Пертурбација нам омогућава да сачувамо део решења и тиме “усмеримо” нашу претрагу уместо да вршимо насумичну претрагу простора решења.

Следи формални опис локалне претраге.

Нека је Ц функција цене коју хоћемо да минимализујемо. Кандидат за решење ћемо обележавати са р, а скуп свих кандидата са Р. Локална претрага коју ћемо обележити са ЛокалнаПретрага дефинише мапирање из Р у мањи скуп Р* локално оптималних решења р*.

процедура Итеративна локална претрага
	р0 = ГенеришиИницијалноРешење
	р* = ЛокалнаПретрага(р0)
	понављај
		р’= Пертурбација(р*)
		р*’ = ЛокалнаПретрага(р’)
		р* = КритеријумПрихватања(р*, р*’)
	докле год критеријум заустављања није испуњен
крај

Пертурбација нам омогућава да итеративна локална претрага налази р*’ суседе од р* у скупу Р* уместо да насумично врши претрагу.

Важно је напоменути да за специфичне проблеме, се користе специфичне имплементације горе споменутих компоненти.

2. Имплеметација итеративне локалне претраге (на примеру проблема путујућег трговца)

Прво дефинишимо проблем путујућег трговца. У потпуно повезаном тежинском графу пронаћи затворен пут тако да је сваки чвор посећен само једном, завршава се у почетном чвору и има минимналну укупну тежину.
Функција цене Ц је дужина пута, кандидат за решење р је редослед обиласка чворова а скуп Р је скуп свих комбинација обиласка. За овај проблем наивна претрага је немогућа, јер за скуп Р кардиналности 200 постоји 200! = 7.89 * 10^{374} комбинација што је огроман број. Ради поређења, процењен број атома у видљивом свемиру је око 10^{80}.

Иницијално решење: Одабир иницијалног решења може да утиче на квалитет алгоритма. Ако желимо да наш алгоритам што брже достигне неко квалитетно решење, потребно је да почнемо од што бољег иницијалног решења неком конструктивном хеуристиком. Добар пример налажења иницијалног решења је грамзиви приступ “најближег суседа”, где градимо пут тако што почнемо од једног чвора и спајамо га са најближим суседом који није посећен. Порастом величине улаза овај утицај је све већи. На мањим улазима насумична иницијализација чак може да покаже боље резултате на неким скуповима зато што има довољно времена, али порастом величине улаза захтевано време експоненцијално расте и неопходно користити неку конструктивну хеуристику за иницијализацију.

Пертурбација: Циљ пертурбације је да се побегне из локалног минимума примењивањем промена на тренутно решење, али тако да се оно не изгуби потпуно. Ако је пертурбација мала њу ће локална претрага да врати решење назад у локални минимум, а ако је превелика губимо ефекат итеративности и претварамо у претрагу насумичног поновног покретања. Постоје различити начини да се врши пертурбација, али у овом раду сам се ослонио на тврђења из рада “Lourenço, H.R., Martin, O. and Stützle, T. (2001), A beginner’s introduction to Iterated Local Search” да “double-bridge” даје најбоље резултате са функцијама тражења локалног минимума које сам ја користио. Ова метода у најосновнијем облику функционише тако што нашу путању делимо на 4 дела, а затим их насумично рекомбинујемо.

Критеријум прихватања: Критеријум прихватања одређује да ли локални минимум пертурбираног решења постаје тренутно решење. Постоје разне варијације овог критеријума, може се прихватати побољшање као у овом раду, може се прихватати свако решење што води ка насумичној претрази, може се прихватати побољшање које је изнад негог прага ако имамо слабу пертурбацију јер слаба пертурбација и прихватање малих побољшања може да нас зароби у локалном минимуму и разне друге враијације. Могуће је и комбиновати симулирано каљење али у том случају је потребно експериментисање са вредностима температуре и коефицијентом хлађења за сваку инстанцу на којој би се покретала таква варијанта алгоритма пре него што се нађу прихватљиве вредности

Локална претрага: Локална претрага нам тражи локални минимум. За проблем путујућег путника по “Lourenço, H.R., Martin, O. and Stützle, T. (2001), A beginner’s introduction to Iterated Local Search” најбоље локалне претраге су Лин-Кернига па 3-опт па 2-опт. У овом раду су имплементиране варијанте 3-опт и 2-опт претраге. Идеја к-опт претраге где је к у овом случају 2 односно 3, је да итеративно бирамо к грана које нису суседне, уклонимо их и затим повежемо пут тако да буде краћи него што је био, ако је могуће, докле год има побољшања. У случају 2-опт постоји само један нов начин да се повеже пут, а у 3-опт постоји

7 нових начина. Овде постоји велика разика у комплексности 3-опт и 2-опт алгоритма јер 3-опт је класе O(n^3), а 2-опт је O(n^2) . Постоје варијације, да ли узимамо прво побољошање или тражимо најбоље што је временски захтевније. Порастом димензије улаза чист 3-опт постаје временски неефикасан. Идеја која је примењена у овом раду је да се посматра ограничен број најближих градова где онда добијамо алгоритам класе O(n*k^2) где је k много мање од n (на пример улаз од 1000 градова, а посматрамо 20 најближих суседа).

3. Резултати извршавања

У овом раду је анализирано осам варијација имплементације итеративне локалне претраге. Да ли је иницијално решење насумично генерисано или је коришћен грамзиви приступ, да ли је коришћен 2-опт или 3-опт (посматра се 20 суседа) и да ли су 2-опт и 3-опт узимали прво побољшање или најбоље.
Поређења су рађена на познатим скуповима са познатим оптималним решењима. За сваки улаз сваки алгоритам је позиван по 10 пута и чувани су подаци о раду тог алгоритма кроз одређен временски период, за мање скупове 60 секунди по позиву, а за веће скупове и по 600 секунди. Пореди се просечно понашање алгоритама. У случају да је за неки алгоритам x потребан просек у тренутку t, за сваки позив се чува листа побољшања кроз временски период као и тренутци када је дошло до побољшања, на основу чега се врши интерполација у тој тачки и онда се рачуна просек интерполираних вредности.
Коришћени су улазни подаци из библиотеке TSPLIB95 са сајта http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ и подаци са сајта www.math.uwaterloo.ca у којима је измењено заглавље ради лакшег читања у програму.

Табела испод представља просечне резултате алгориритама на познатим инстанцама. Плава колона у првој табели представља оптималне вредности. У другој табели је приказана разлика просечног решења и оптималног у процентима.

	berlin52	eil101	qa194	a280	d493	uy734	pr1002
ILS_first2opt_random	7544.36590190408	641.139377522996	9390.23624452063	2593.33248518148	35512.7343927584	81535.4903268417	273044.908494189
ILS_best2opt_random	7544.36590190408	643.535066450906	9381.12749128424	2623.06973173412	35768.9613357844	83175.4847161045	278286.1548868
ILS_first2opt_greedy	7544.36590190408	640.954904263627	9377.21229593208	2594.63084236341	35435.8699776761	81697.9384542119	269657.307117308
ILS_best2opt_greedy	7544.36590190408	640.580537398499	9388.50347356828	2653.7649913935	35677.5707212235	82339.777766136	268512.843904552
ILS_first3opt_random	7544.36590190408	640.398774312324	9358.1986296937	2593.30120968376	35330.1275432251	80781.8212184081	268015.144420974
ILS_best3opt_random	7544.36590190408	640.393565335527	9360.53917527275	2595.42110478268	35277.9087902555	80198.3606737637	265497.607271534
ILS_first3opt_greedy	7544.36590190408	640.398774312324	9368.55421298499	2600.22330449665	35387.3765929938	80613.8154479336	266614.301823267
ILS_best3opt_greedy	7544.36590190408	640.767720831063	9365.9676502994	2589.37427581082	35278.6177166827	80122.6708587452	263793.562380314
optimal	7542	629	9352	2579	35002	79114	259045

%	berlin52	eil101	qa194	a280	d493	uy734	pr1002
ILS_first2opt_random	0.0313696884656549	1.92994873179587	0.408856335763803	0.55573808381078	1.45915774172447	3.0607608347975	5.40443108115925
ILS_best2opt_random	0.0313696884656549	2.31082137534277	0.311457349061591	1.70879145925243	2.19119289121878	5.13371175279281	7.42772679912756
ILS_first2opt_greedy	0.0313696884656549	1.90062070963863	0.269592557015403	0.606081518550214	1.23955767577881	3.26609507067256	4.09670409284411
ILS_best2opt_greedy	0.0313696884656549	1.84110292503959	0.39032798939564	2.89899152359442	1.9300917696803	4.07737918211189	3.65490316530023
ILS_first3opt_random	0.0313696884656549	1.81220577302448	0.066281326921507	0.554525385178757	0.937453697574706	2.10812399626881	3.46277458394256
ILS_best3opt_random	0.0313696884656549	1.81137763680874	0.0913085465435209	0.636723721701426	0.788265785542245	1.37063057583197	2.49092137332664
ILS_first3opt_greedy	0.0313696884656549	1.81220577302448	0.17701254261109	0.822927665632034	1.10101306495001	1.89576490625375	2.92200267261171
ILS_best3opt_greedy	0.0313696884656549	1.87086181733911	0.149354686691624	0.402259628182245	0.790291173883482	1.27495874149354	1.83310327561389

Из горе приказаног можемо да закључимо да порастом величине улаза расте разлика између резултата 2-опт и 3-опт претраге и да је 3-опт претрага боља. Поред тога приметно је да растом величине улаза грамзиви старт даје боље резлтате у односу верзију са насумичним стартом, али и да за 3-опт алгоритам и довољно велике улазе битније да ли је верзија са првим побољшањем или најбољим. Најоптималније решење даје итеративна локална претрага где за иницијализацију користимо похлепни алгоритам, а за локалну минизациону функцију користимо 3-опт са најбољим побољшањем.

У првој табели испод је приказано колико је просечно времена потребно сваком алгоритму да дође до горе споменутих резултата. У другој табели је уписано просечно време потребно алгоритмима да дођу на 90% оптималног решења, односно до решења које се за 10% разликује од оптималног

seconds	berlin52	eil101	qa194	a280	d493	uy734	pr1002
ILS_first2opt_random	1.49454696178436	15.5077168941498	174.460510206223	186.184761381149	578.728665328026	555.018562602997	559.102843141556
ILS_best2opt_random	0.844648098945618	19.4827996015549	155.045850372314	251.184587621689	567.459129261971	585.592204070091	556.821401405334
ILS_first2opt_greedy	1.48088343143463	24.0865840673447	174.417463755608	225.094945669174	571.124394798279	573.977546811104	531.583602261543
ILS_best2opt_greedy	0.714182281494141	14.0847898483276	114.575677371025	248.179135251045	576.916093420982	567.094829511643	483.653462362289
ILS_first3opt_random	22.593861246109	71.2108244895935	195.168039393425	236.959774923325	581.836356544495	556.874341869354	570.590377140045
ILS_best3opt_random	26.4506250858307	61.535800909996	229.73707678318	185.118763685226	581.105003261566	584.429497122765	593.09471924305
ILS_first3opt_greedy	19.2388916492462	65.748922252655	209.869903540611	263.135261249542	563.716010570526	562.416553497315	552.867181372643
ILS_best3opt_greedy	11.2205861330032	86.2924347400665	151.414110684395	213.27703807354	569.085765624046	591.3885658741	591.594997048378

seconds	berlin52	eil101	qa194	a280	d493	uy734	pr1002
ILS_first2opt_random	0.00993223190307617	0.0834702968597412	0.703526544570923	2.89011352062225	20.6217116355896	77.1322036981583	229.691431808472
ILS_best2opt_random	0.0108559370040894	0.105888319015503	0.654465413093567	4.55104582309723	14.5117879390717	66.1818404912949	178.759615969658
ILS_first2opt_greedy	0.00109872817993164	0.016052770614624	0.13853006362915	0.141137671470642	1.71238088607788	7.35027749538422	14.6912924289703
ILS_best2opt_greedy	0.0024813175201416	0.0178296327590942	0.133595132827759	0.233785510063171	1.63361921310425	7.16028831005096	18.6795905590057
ILS_first3opt_random	0.124328851699829	0.806930303573608	4.52212409973145	10.46591796875	43.8635298252106	110.492009735107	231.579806923866
ILS_best3opt_random	0.175743532180786	0.733901381492615	3.34541368484497	7.34129059314728	28.7177122831345	65.6178233146668	123.800431156158
ILS_first3opt_greedy	0.0235902547836304	0.144361543655396	0.811585187911987	1.16502709388733	3.90922830104828	11.8241365671158	15.6951093673706
ILS_best3opt_greedy	0.0568509817123413	0.23855357170105	0.869441962242126	0.844670176506043	5.4215576171875	11.407128739357	23.0917153835297

Извести неки паметан закључак из ових података….

У табели испод је приказан коефицијент варијације алгоритама на датим улазима.

%	berlin52	eil101	qa194	a280	d493	uy734	pr1002
ILS_first2opt_random	1.62E-14	0.144716449331183	0.3234273626221	0.417593092504413	0.376070112389615	0.542536963281427	0.571424428397191
ILS_best2opt_random	9.34E-15	0.506128280399342	0.23365756841845	0.498606957272646	0.469611265234395	0.401420706090218	0.862893161895839
ILS_first2opt_greedy	1.75E-14	0.142039570766	0.106719933096123	0.392879650138816	0.259122598203924	0.39038316338743	0.280786880036491
ILS_best2opt_greedy	2.50E-14	0.115191298267398	0.173223525910288	0.283823474237461	0.273891964889199	0.34982690072682	0.156641347711588
ILS_first3opt_random	1.62E-14	0.0876875971997971	0.0988119850532166	0.329472902506903	0.223869947378462	0.264255525640767	0.390959397762471
ILS_best3opt_random	8.52E-15	0.0852481031743401	0.113295000307319	0.385338648644115	0.232139992667618	0.210723407622129	0.292265428109264
ILS_first3opt_greedy	7.62E-15	0.0876875971997681	0.131159184658061	0.330292457668282	0.320293792687627	0.221393015618866	0.371435969908587
ILS_best3opt_greedy	8.52E-15	0.132580235477915	0.132792211090425	0.092323486429442	0.287384037011765	0.20732977121867	0.220641312437021

Из горе приказаног, можемо приметити да варијације алгоритама где се користи 3-опт теже већој стабилности него алгоритми где се користи 2-опт, као и да у варијацијама где се користи 2-опт насумично иницјално решење даје лошију стабилност док код 3-опт алгоритма не мора увек да значи.

Да ли углавити овде негде неуспеле експерименте са адаптивним ИЛС на великим инстанцама?

Неки закључак рада, нпр како би 3-опт био још бољи кад не би посматрао само 20 најближих него на пример и 5 најдаљих, јер постоји категорија ТСП скупа на којем не ради добро и претпостављам да је то разлог.

На крају референце на радове које сам читао кад сам припремао пројекат.