Literatuurstudie

Inhoudstafel

Neighborhood Search Algorithm for the Combined Through and Fleet Assignment Model

december 2001 – Ahuja / Goodstein / Mukherjee (Florida / Chicago / Cambridge)

Synthese

In dit artikel van 2001 onderzochten 3 universitairen en 2 United Airlines onderzoekers de mogelijkheden om sneller, betere optimale oplossingen te vinden voor het ctFAM.

Dit is een tweedelig optimalisatieprobleem waarbij men enerzijds vliegtuigen moet toewijzen aan lijnvluchten (elke lijnvlucht heeft minstens 1 vliegtuig, hetzelfde vliegtuig kan op een bepaald moment maar 1 lijnvlucht uitvoeren, uit elke luchthaven moeten minstens evenveel vluchten vertrekken als aankomen, men kan niet meer vliegtuigen van een bepaalde type toewijzen dan er werkelijk zijn). Verder aangeduid met FAM.

Anderszijds wil men op een zo voordelig mogelijke manier aansluitingen tussen lijnvluchten creeëren. Zo'n aansluiting kan als een lijnvlucht aankomt op een luchthaven en een andere een tijdje erna daar vertrekt (afhankelijk van de tijd nodig om passagiers te laten ontschepen, ..). Opnieuw zijn er beperkingen: twee verbonden lijnvluchten moeten door hetzelfde vliegtuig bediend worden, [? (1c) en (1d) p 7], capaciteitsbeperkingen (afhankelijk van "overnachtse" aansluitingen[1]). Verder aangeduid met TAM.

De optimalisatie zit hem in het maximaliseren van de winst (inkomsten – kosten van het toewijzen van een vliegtuigtype aan een lijnvlucht – kosten van het kiezen van een aansluiting).

Ze onderzochten drie methodes: een Integer Programming Formulation, een Neighbourhood Search Algorithm en een Tabu Search implementatie.

Integer Programming Formulation

Bij de Integer Programming Formulation construeren ze een connectie netwerk. Dit is een gerichte grafe bestaande uit knopen die de lijnvluchten voorstellen en verbindingen die de mogelijke aansluitingen voorstellen. Nadat deze grafe geconstrueerd is op basis van alle vliegtuigtypes, kandidaat aansluitingen en lijnvluchten kunnen een aantal constraints eenvoudig herkend worden. Door de keuze van vliegtuigtypetoewijzingen te laten wijzigen en de kost en constraints te evalueren kan er zo naar een optimale configuratie gezocht worden. Spijtig genoeg is het zoeken naar een naald in een hooiberg, waardoor deze eerste methode afviel.

Neighbourhood Search Algorithm

Hier lossen ze het probleem gradueel op. Eerst zoeken ze een semi-optimale oplossing voor het toewijzen van vliegtuigen aan lijnvluchten (= FAM). Vervolgens gebruiken ze die oplossing als vertrekpunt voor het zoeken naar aansluitingen (= TAM). Dit resultaat wordt gebruikt als vertrekpunt om nog betere oplossingen te vinden. Zo gaat men iteratief buren zoeken van het huidige resultaat die beter presteren. Het algoritme stopt als er geen twee vliegtuigtypes geswitcht kunnen worden waarbij de kost vermindert.

Als buren definieren ze de A-B swaps: men kiest een vliegtuigtype A en een vliegtuigtype B en

[1] Aangezien men alle vluchten op een "dagelijkse" basis wil formuleren, kan het zijn dat je met 1 vliegtuigtype wel 6 lijnvluchten kunt bedienen, maar dat die cirkel langer dan een dag duurt, waardoor er bv 2 werkelijke vliegtuigen ingezet moeten worden. Het aantal in te zetten vliegtuigen is dus te herkennen aan "overnight-arcs".

vormen lijnvluchten die bediend werden door type A om naar type B vluchten. [? methode om optimale swaps te vinden, blijft me wat onduidelijk:]. Uit het eerder besproken connectie netwerk verwijderen ze alle knopen en bogen die niet door vliegtuigtype A of vliegtuigtype B bediend worden. Dit wordt dan de A-B solution graph genoemd. Vervolgens keren ze de richting van de B-bogen om. Indien er een gesloten gericht pad bestaat, is dit een geldig pad indien een “maat” van elke knoop i maar in het pad zit als de boog tussen i en zijn maat er ook in zit. Hieruit valt dan de A-B improvement graph af te leiden. Deze improvement graph is een deelgrafe van de A-B solution graph waarbij men maar een boog tussen twee knopen trekt indien men deze wil switchen/swappen van vliegtuigtype. Er worden in de tekst 6 soorten bogen benoemd, waar ik het nut niet echt van inzie. Uit experimenten werkte dit algoritme zeer snel (seconden) en blijkbaar werden er steeds goede resultaten bereikt (hoewel ik dat uit de figuren 8 en 9 p. 31 niet echt kan uithalen, waarom kunnen veranderingen negatief worden?).

Tabu Search Algoritme

Als afsluitertje vermelden ze de resultaten van hun tabu search algoritme. Het blijft beperkt tot een korte paragraaf en de mededeling dat het waarschijnlijk betere resultaten zou halen als het beter geïmplementeerd zou worden. De methode maakt gebruik van een kort-termijn geheugen van 5 iteraties. Het algoritme deed er langer over, maar kon meer wijziging doorvoeren (= vond meer betere oplossingen).

Onduidelijkheden

[p. 5 paragraaf “Our approach..”] An A-B swap consists of changin some legs flown by [..] flown by fleet type **A** to fleet type A. Moet dit hier geen B zijn?
[p. 6 d_(ij)^f] Observe that d_(ij)^f **<** 0 for (i,j) € T en 0 otherwise. Als je de totale kost wil minimaliseren, moet dat hier toch **> 0** zijn?
[p. 11 gebruik van cycle-based property] Begrijp ik niet...

Belangrijkste conclusies

Integer programming lijkt de kortste oplossing, maar is in de praktijk niet werkbaar (veel overeenkomsten met het werken met neurale netwerken?)
Door het probleem te vertalen naar grafen, kunnen bepaalde constraints gemakkelijk herkend worden.
Het probleem opsplitsen in deelproblemen (TAM en FAM) is een goede oplossing, hoewel er moeilijkheden optreden indien je overkoepelende constraints hebt.
Werken met buuromgevingen! Goede buuromgevingen vinden!

Extra geraadpleegde bronnen

http://ocw.mit.edu/NR/rdonlyres/Civil-and-Environmental-Engineering/1-206JAirline-Schedule-PlanningSpring2003/5F9CC24D-8DDA-4419-B631-D456D5BF6339/0/lec6_fleet_assignment_2003.ppt

Detailomschrijving FAM, voorbeelden van time-line netwerken, alternatieve oplossingsmethoden (speciale gevallen zoals eilanden, en branch-and-bound technieken).

http://en.wikipedia.org/

Korte definities van Linear Programming, Branch & Bound technieken en Tabu Search technieken.

Tijdsbesteding

Eerste lezing: 4 uur

Tweede lezing, samenvatting en synthese: 4 uur

The Traveling Tournament Problem: Description And Benchmarks

2001 – Easton / Nemhauser / Trick (USA)

Synthese

TTP

Combinatorisch optimalisatie probleem waarbij je n (even) teams hebt, die je moet zien te schikken in een round-robin tournament. Zo zullen er exact 2(n-1) ronden toegewezen moeten worden.

Er zijn twee harde constraints. De eerste stelt dat elke ploeg juist 2 keer met elke andere ploeg moet spelen, één keer thuis en één keer uit. De tweede dat geen enkele ploeg mag meer dan 3 rondes na elkaar uit / thuis spelen en dat een A thuis – B uit ronde gevolgd kan worden door een A uit – B thuis ronde (= *home/away patterns / at most constraints-norepeat constraints).* Verder probeert men de afgelegde reisweg per team te minimaliseren (= *distance traveled)*.

Het is een interessant probleem omdat het toelaat verschillende aanpakken te vergelijken en combinaties van methodes te verkennen. Men heeft zowel constraint programming nodig voor de home/away patterns te kunnen optimaliseren, als integer programming om een minimale reisafstand te verkrijgen.

Een strenge ondergrens kan bekomen worden door de som van de *team bounds*, deze wordt dan de **Independent Lower Bound / ILB** genoemd.

Constraint programming formulation

Voor problemen met meer dan 6 groepen moet men al gebruik maken van een zoekmethode. Een voorbeeld daarvan is om het aantal toeren door de ploegen ondernomen gradueel te laten toenemen. Verder kunnen we het aantal te onderzoeken mogelijkheden nog meer beperken door de ILB ook een minimum aantal van toeren beperking te laten bevatten.

Combination method involving column generation approaches

Hier worden de constraint programming methodes gebruikt om de variabelen te genereren, die dan gecombineerd worden via integer programming technieken.

Trivia

Het is niet duidelijk dat TTP triviaal wordt als het overeenkomstige TSP probleem triviaal is.

Er kunnen datasets gevonden worden op de http://mat.gsia.cmu.edu/TOURN site.

Er zijn nog geen optimale resultaten gevonden voor het probleem met 8 teams.

Conclusies

Goede korte inleiding tot het probleem. Het gebruik van een ILB is me nogal vaag gebleven.

Extra geraadpleegde bronnen

Tijdsbesteding

Eerste lezing: 1u

Tweede lezing, samenvatting en synthese: 30 min.

A Simulated Annealing Approach to the TTP

2003 – Anagnostopoulos / Michel / Hentenryck / Vergados (USA)

Synthese

Introductie

Deze paper stelt een Simulated Annealing techniek voor om het Traveling Tournament Probleem op te lossen. Ze geven een kortere inleiding tot het probleem, zie ook pagina 5. Ze maken gebruik van een onderverdeling tussen harde en zachte constraints, $O(n^3)$ omgevingen, een oscillatie strategie en heropwarmingen. Ze hebben een aantal experimenten uitgevoerd waaruit blijkt dat hun methode minstens even goed werkt als de bestaande integer/constraint programs en vaak zelfs beter.

Probleemomschrijving

Zie ook pagina 5.

De lokale zoekmethode

De techniek bestaat uit vier grote ontwerpbeslissingen: de opspitsing tussen harde en zachte constraints, de regels om een nabije omgeving te bepalen, het simulated annealing algoritme met de aanpassing van de temperatuur en de doelfunctie. Deze worden achtereenvolgens in meer detail besproken.

Constraints

De volgende constraints worden er onderscheiden:

- harde constraints: de constraints die bij elke configuratie moet gelden. Hieronder valt de round robin constraint.
- Zachte constraints: deze constraints moeten gelden in het uiteindelijke resultaat, maar hoeven niet te gelden in de tussenliggende configuraties. Zowel de no-repeat als de at-most constraint vallen hieronder.

Het bepalen van de omgeving

Per huidige configuratie wordt zijn omgeving berekend: alle configuraties die door het toepassen van vrij eenvoudige regels op de huidige configuratie bekomen kunnen worden. Niet alle kalenders die op deze manier bekomen worden, voldoen ook aan de zachte constraints.

1) SwapHomes: <Team_i, Team_j>
   Hier worden de home/away pattern van team i en team j onderling verwisseld. Dit is een micro-aanpassing omdat het geen wijzigingen veroorzaakt bij andere teams.
2) SwapRounds: <Rounde_k, Ronde_l>
   De wedstrijden van ronde k worden gewisseld met die van ronde i. Dit is een macro-aanpassing maar heeft geen verdere wijzigen nodig.
3) SwapTeams: <Team_i, Team_j>
   De wedstrijden van team i en team j worden gewisseld. Hierdoor moeten bepaalde wedstrijden van hun tegenspelers ook aangepast worden. Zo zullen er 4 wedstrijden per ronde aangepast moeten worden.

Deze eerste drie vrij eenvoudige regels zijn niet voldoende om alle goede configuraties te kunnen bereiken. Daarom worden er nog twee meer algemene regels gedefinieerd.

4) Partial Swap Rounds: <Team_i, Ronde_k, Ronde_l>
   Hier wordt een wedstrijd van team i in ronde k gewisseld met rond i. Hierdoor moeten de kalenders van 3 andere teams in die rondes nog aangepast worden. Het vinden van die teams kan via het opstellen van een grafe die de wedstrijden van team i in ronde k en l bevat. De knopen stellen de teams voor en een boog bestaat tussen team a en team b indien deze spelen in ronde k of in ronde l. Alle kalenders van de teams die betrokken zijn in de cirkel waar Team i inzit, moeten aangepast worden. [p. 7 heb ik het vinden van de teams goed begrepen?]

5) Partial Swap Teams: <Team_i, Team_j, Ronde_k>
   De wedstrijd in ronde k van team i wordt gewisseld met die van team j. In het slechtste geval moeten er dan per ronde voor 4 ploegen aanpassingen gemaakt worden.

Simulated Annealing

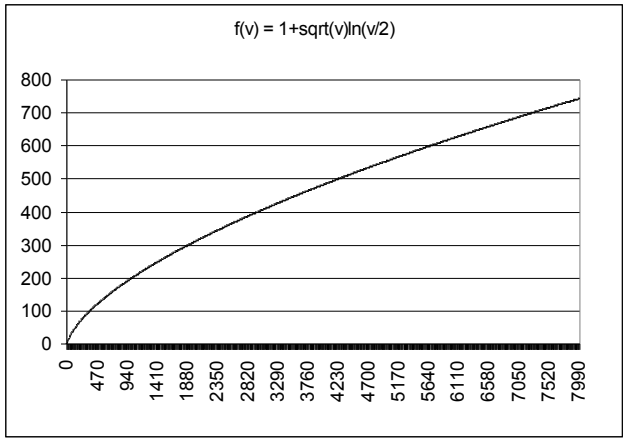
Het gebruikte algoritme vertrekt van een goede (aan zowel harde als zachte constraints voldaan) kalender. Deze initiële situatie wordt door een eenvoudig backtrackalgoritme bekomen.

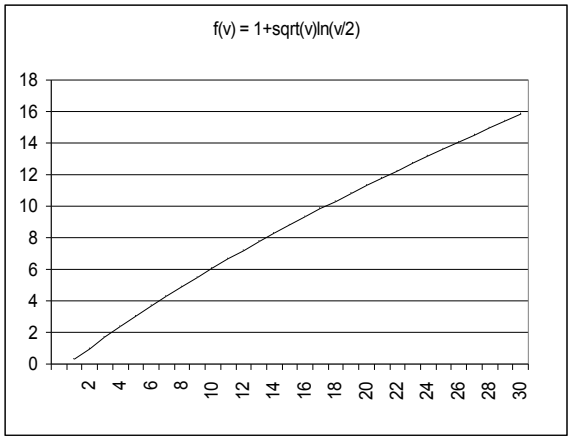
Vervolgens kiest het algoritme willekeurig een bepaalde configuratie uit zijn omgeving en berekent diens kost. Indien de winst (oude kost – nieuwe kost > 0) positief is onthoudt hij ze, en anders onthoudt hij ze met een bepaalde kans (afhankelijk van de temperatuur). De temperatuur wordt op twee manieren beïnvloed: indien er na THRESHHOLD_1 nog geen verbeterende stap ondernomen is, verlaag je de temperatuur. Indien er geen verbeteringen meer gevonden werden, gaat de temperatuur een aantal keren opnieuw verhoogd worden, totdat het verhogen van de temperatuur ook geen betere resultaten meer geeft (> THRESHHOLD_2).

De doelfunctie

De kost van een bepaalde stap wordt berekend aan de hand van een doelfunctie. Indien er fouten gemaakt worden tegen de zwakke constraints, wordt er een penalty term toegevoegd aan de kost. Er wordt een bepaald gewicht toegekend aan deze penalty term. In de onderstaande figuren stelt f(v) de penalty term voor. [p. 8 staat dat f(1)=1, volgens mij is f(1)=0.31 en f(2)=1]

De strategische oscillatie





Hiermee wordt het laten afnemen en toenemen van het gewicht van de penalty term. Indien er geen fouten gevonden werden, zal het gewicht verder verkleind worden (gedeeld door gamma), indien er wel gevonden worden, zal het vergroot worden (maal delta). Hierdoor zal het algoritme convergeren naar de selectie van kalenders die ook voldoen aan de zwakke constraints.

Experimenten

Het standaard algoritme

Toegepast op de 'National League benchmark'. De parameters van het algoritme waren zo gezet er er weinig oscillatie mogelijk was, een zeer trage afkoeling en een mogelijkheid voor het systeem om lage temperaturen te behalen. De resultaten verbeteren de best gekende resultaten tot 2-5%. Er wordt ook aangetoond dat van de 50 runs die ze per aantal ploegen gedaan hebben, de slechtste oplossing nog steeds beter is dan de best gekende resultaten.

De aparte componenten

Er werden een aantal varianten op het TTSA geformuleerd, oa TTSA met enkel gedeeltelijke zetten, TTSA zonder heropwarmingen, TTSA met een Metropolis algoritme met een bepaalde ingestelde temperatuur T ipv de simulated annealing. Vervolgens werden de algoritmen toegepast op het probleem van grootte 10. De standaard TTSA bleef de beste methode, gevolgd door de enkel gedeeltelijke zetten, de zonder heropwarmingen en de Metropolis variant met de laagste temperatuur. Doordat de resultaten van de standaard TTSA en de enkel gedeelde zetten zo dicht bij elkaar lagen, werden er nog extra experimenten gedaan waarbij duidelijk werd dat volledige wissels enkel doorslaggevend waren in een korte zoektijd, maar dat hun effect na langere tijd marginaal is.

De kwaliteit van de oplossing tov de tijd

Wanneer men de best gevonden oplossing uitzet tegenover de tijd voor het probleem van 12 ploegen, ziet men dat de snelste verbetering gebeurd in de eerste 1000 seconden, waarna het algoritme ofwel zeer langzaam nog een beetje verbeterd (neem1000m over 1000 seconden), ofwel lang constant blijft om dan nog een 2000m ineens te verbeteren (ergste geval bleef 2500 sec constant om dan 4000m te verbeteren.)

Snel afkoelen

Bij een snelle afkoeling, werden er haast onmiddellijk veel betere resultaten gevonden dan via het standaard algoritme. Op (zeer; 25500 sec) langere termijn was het standaard TTSA beter of even goed.[p. 13-15 gebruik van cycle-based property, wat wordt hiermee bedoeld?]

Conclusies

Het is niet duidelijk of de 5 voorgestelde 'swap'-regels alle oplossingen bestrijken. Moet ik nog eens diep over nadenken.

De experimenten laten uitschijnen dat er véél tijd over gaat om tot een aanvaardbaar resultaat te komen.

Extra geraadpleegde bronnen

Tijdsbesteding

Eerste lezing: 4uur

Samenvatting: 2uur

Synthese: 3 uur

Solving The Traveling Tournament Problem: a combined integer programming and constraint programming approach.

2003 – Easton / Nemhauser / Trick (USA)

Synthese

Introduction

Constraint programming werd reeds succesvol toegepast voor het oplossen van uit en thuis patronen. Integer programming methodes zijn goed in staat om grote instanties van het TSP of routeringsproblemen op te lossen. Het Traveling Tournament Problem heeft beide nodig, wat het een uitdagend probleem maakt om verschillende combinatietechnieken uit te proberen.

The Traveling Tournament Problem

De definitie voor het TTP gaat als volgt:

Gegeven een set van n teams T={t_1, ... , t_n} waarbij n een even getal is; D een symmetrische nxn integer afstandsmatrix en l,u natuurlijke getallen waarbij l <= u. l stelt het minumum aantal opeenvolgende uit/thuis wedstrijden en u het maximum.

Gevraagd is een double round robin tournament op alle teams uit T zodat l en u niet overschreden worden en de totaal afgelegde afstand die door alle teams afgelegd moet worden, minimaal is.

OPGELET: hier wordt er geen rekening gehouden met de repeat constraint!

Indien je u=n-1 neemt, dan komt de toer van elke team erop neer om het TSP op te lossen. Als u=1 vallen de constraints weg.

The Single Team Problem

We definieren als strenge ondergrens de som van alle optimale toeren per team (dus dus geen rekening houden met onderlinge afhankelijkheden): Independent Lower Bound.

Vervolgens definieren ze een algoritme die voor een gegeven team diens optimale toer gaat berekenen voor een gegeven l en u. Het algemene geval is Npcomplete. In het speciale geval dat u=2 kan er een O(n^3) algoritme gevonden worden. Deze definieert een grafe waarin er zodanig bogen met een kost gedefinieerd staan, dat als je de 'minimale gewogen perfecte match' berekend, hieruit een optimale kalender uit te halen valt.

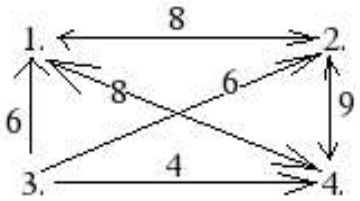
Bv. Gegeven 4 steden, l=1 en u=2.

De afstandmatrix ziet er als volgt uit:

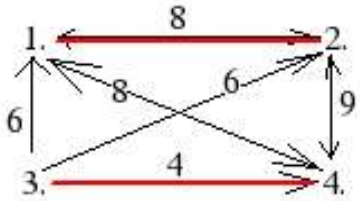
D = [0 2 3 3;
2 0 3 4;
3 3 0 2;
3 4 2 0]

Vervolgens berekenen we voor ploeg 3 de optimale kalender.

De grafe ziet er als volgt uit:



De minimale gewogen perfect match kan er zo inzien:



Dus alle combinaties waarbij 3-1-2-3 en 3-4 voorkomen, kunnen gebruikt worden:

1 2 3 4 3 3 OF 3 1 2 3 4 3 OF 3 3 1 2 3 4 OF 3 3 2 1 3 4
en de wedstrijden 1 en 2 kunnen natuurlijk omgewisseld worden.

Solution Methodology

De volledige oplossingsmethode vertrekt van de optimale wedstrijdkalenders per ploeg en probeert via een Branche-and-bound algoritme dan correcte kalenders te bekomen met een zo laag mogelijke totale weg.

Ze behalen redelijk goede resultaten, waarbij hun meest optimale oplossing tot 5-6% hoger liggen dan de ILB.

Computational Results

Ze benadrukken dat ze hier geen rekening gehouden hebben met de no-repeat constraints.

Ze hebben een goede oplossing gevonden voor het probleem met 8 ploegen, waarbij ze 20 processoren ongeveer 4 dagen hebben laten lopen... Heel tijdsinventief dus.

Conclusies

- Tijdsintensieve benadering van het probleem
- De beginvoorwaarden zijn hier het belangrijkste. Het zoeken in de omgevingen zelf wordt via een branch en bound algoritme gedaan.

Extra geraadpleegde bronnen

Tijdsbesteding

1ste lezing en synthese: 3u

A Composite-Neighborhood Tabu Search Approach to the TTP

mei 2006 – Gaspero / Schaerf (Italie)

Synthese

Historisch overzicht

Het TTP werd het eerst geformuleerd door *Easton* in 2001. Het werk ging verder op de beschrijving van het TSP (*Junger* 1995) en Tournament scheduling (*Henz* 2001). Vervolgens werd er gekeken in hoeverre de huidige programmeertechnieken een antwoord op het probleem konden formuleren: Integer en Constraint programming technieken, *Easton* in 2003. *Anagnostopoulos* in 2005 verkrijgt zeer goede resultaten in veel minder tijd met de Simulated Annealing approach en Lim doet het via een Two Stage approach. In 2004,2005 beschrijft men de mirrored version van TTP, waarbij er een bijkomende constraint wordt opgelegd, waarbij de kalender uit 2 gespiegelde delen bestaat.

Omschrijving van het TTP

Default.

In deze paper onderzoeken ze hoe ze het TTP kunnen oplossen (voor instancies van minstens 12 ploegen), gebruikmakend van zogenaamde *tabu lists* (taboe lijsten).

Local Search for TTP

Search space

De volledige oplossingsverzameling bestaat uit alle correcte kalenders, ook kalenders die niet voldoen aan de no repeat of at most constraints worden opgenomen.

Cost function

Is de gewogen som van de fouten die er tegen de constraints gemaakt worden, als de afstand. De gewichten worden tijdens het algoritme dynamisch aangepast en op elk moment wegen de constraintfouten zwaarder door dan de afstand.

Beginsituatie

Deze wordt gevonden in twee stappen, eerst wordt er een willekeurig tornooikalender patroon[2] genomen. Dit probleem is nauw verwant aan het vinden van een 1-factorizatie van een complete niet-gerichte grafe. Vervolgens kan de 1-factorizatie omgezet worden in een tornooi kalender patroon, waarbij ze gebruikmaken van een canonical pattern. Om verschillende beginsituaties te verkrijgen, worden geldige varianten geformuleerd (random verwisselen van home/away, rondes, ...)

Definitie van de omgeving

Ze gebruiken dezelfde mutatieregels als Anagnostopoulos, te weten de *SwapHomes, SwapTeams, SwapRounds, SwapMatches*[3] en *SwapMatchRound*[4]. Voor de SwapMatches en de SwapMatchRound moeten extra aanpassingen gemaakt worden bij andere ploegen/teams om tot een geldige tornooikalender te komen. Deze reeks van wijzigingen noemen ze de *repair chain*.

---

[2] Het zogenaamde *Tournament pattern*, een geldige tornooikalender waarbij anonieme nummers van 0 tot n-1 gebruikt worden als teams.

[3] aka Partial Swap Rounds. Het verschil met Anagnostopoulos is dat hier een ganse rij gewisseld wordt.

[4] aka Partial Swap Teams

Analyse van de omgeving

Aangezien er lijsten bijgehouden worden van configuraties die niet meer gewenst zijn, is het noodzakelijk om equivalente configuraties te kunnen herkennen. De eenvoudigste zijn de symmetrisch te herkennen configuraties. Bij de SwapMatchRound is <t, r_1, r_2> hetzelfde als de <t_i, r_1, r_2> als 1 <= i <= k als de repair chain van <t, r_1, r_2> de teams 1 tot k bevat. We nemen voor de move <t, r_1, r_2> dus telkens de kleinste t wiens repair chains dezelfden zijn. Na experimenten werd het duidelijk dat er bij het toepassen van SwapMatches nog andere equivalente configuraties mogelijk zijn, waardoor het voldoende is om elke SwapMatches met een repair chain van lengte 2 toe te laten.

Kwaliteit van de wijzigingen

Na experimenten werd de gemiddelde wijziging van de afstand bekeken per mutatieregel. SwapHomes presteert het beste, wat intuitief verklaard kan worden door zijn lokale werking. De meer globale mutaties, zoals SwapTeams en SwapRounds bleken zeer grote wijzigingen te veroorzaken. SwapMatches en SwapMatchRound bleken de minste wijziging te veroorzaken als de mutatie zo volledig mogelijk een wijziging aanbracht: hoe langer de mutatie, hoe volledige de omwisseling is van twee teams. [p. 8 Heb ik dit goed begrepen?]

Definitie van de verschillende omgevingen

Gebaseerd op de voorgaande opmerkingen worden er 4 omgevingen gedefinieerd. De eerste bestaat uit alle omgevingen die bekomen kunnen worden door het muteren volgens alle regels. De tweede negeert SwapTeams en SwapRounds. De derde is een striktere omgeving van de tweede, waarbij voor SwapMatches en SwapMatchRound enkel tot lengte 4 respectievelijk 6 gegaan wordt[5]. De laatste omgeving gebruikt een speciale versie van SwapMatches: de *shortest chain closure*. [p. 11 geen idee wat ze met shortes chain closure bedoelen. Is dit relevant ;)?] Men maakt de opmerking dat de laatste omgeving veel groter is dan de derde.

Samenhang van de omgeving

Er bestaat nog geen theoretisch framework voor het bewijzen dat een omgeving voldoende verbonden is. Door hun omgevingen te testen op NL8[6], waarbij ze steeds de reeds gekende optimale afstand bereikten, hebben ze een sterk vermoeden dat hun omgevingen voldoende verbonden zijn.

Solution Meta-Heuristic for TTP

Er werden 4 solvers uitgewerkt die elk vertrokken van een van de 4 hierboven beschreven omgevingen. Elke solver had verder de volgende instellingen.

- De volledige omgeving werd door de solver bezocht. Elke configuratie die niet voorkwam in de tabu list werd geevalueerd.
- De tabu list zelf is dynamisch. De lengte varieeert random binnen een bepaald interval.
- Een move is tabu als het herkend wordt in de tabu list.
- Een tabu move wordt pas verwijderd van de lijst indien het het best gekende resultaat zou verbeteren. [p. 11 Gaat de solver dan zelfs voor alle tabu waardes hun resultaat berekenen? En dat telkens opnieuw?]
- Wanneer er een maximum aantal *idle iterations* zijn gebeurd, zonder verbetering, herstart de solver vanaf het best gekende resultaat. De solver stopt volledig wanneer er een maximaal aantal *idle rounds* gepasseerd zijn (of bij een time-out)

[5] Deze bovengrenzen werden experimenteel vastgelegd, en in het artikel gaan ze er niet verder op in.
[6] Zie de TTP website van Trick

- De penalty term in de objective function wordt afhankelijk van de zoekgeschiedenis verzwaard of verhoogd. [7]

Experimental Analysis

Benchmark instances

De solvers werden getest op alle gegevens die gevonden kunnen worden op de TTP website. Ze hebben zich geconcentreerd op gegevens van minstens 10 ploegen.

Comparison methodology, parameter setting and implementation

Snurk.

Neighborhood Comparison

Snurk.

Unlimited Time experiments

Door hun beste solver (degene die werkt met de vierde omgeving) meer tijd te geven om tot een resultaat te komen, verkregen ze kostenresultaten waren vergelijkbaar met die uit de literatuur (Lim, 2005 en Anagnostopoulos, 2005).

Beste resultaten

Ze zijn altijd een klein beetje slechter dan de best gevonden resultaten, maar ze hebben wel als eerste goede resultaten verkregen voor andere gegevens.

Conclusions and further work

De gebruikte omgeving bestaat uit een compositie vertrokken uit 5 basis omgevingen die het resultaat zijn van een analytische studie van de kardinaliteit, de overlapping en de gemiddelde kwaliteit van elke aparte omgeving.

Conclusies

- Aangezien dat de mutatieregels zonder al te veel aantekeningen overgenomen zijn uit Anagnostopoulos, kunnen we vermoeden dat die redelijk juist zijn.
- Kort samengevat proberen ze de omgeving zo klein mogelijk te houden en laten ze een solver erop los die ALLE mogelijkheden afgaat (zolang ze niet op de tabu list staan).
- De tabulijst gaat afhankelijk van het probleem herkenningspunten opslaan van configuraties die niet meer bekeken mogen worden. Dit kunnen zowel moves als uitkomsten zijn.

Extra geraadpleegde bronnen

Tijdsbesteding

Eerste lezing: 4 uur / Samenvatting en synthese: 4 uur

[7] Ze hanteren nooit de term temperatuur, hoewel me het hetzelfde mechanisme lijkt als gebruikt bij Anagnostopoulos.

Synthese

Conclusies

Extra geraadpleegde bronnen

Tijdsbesteding