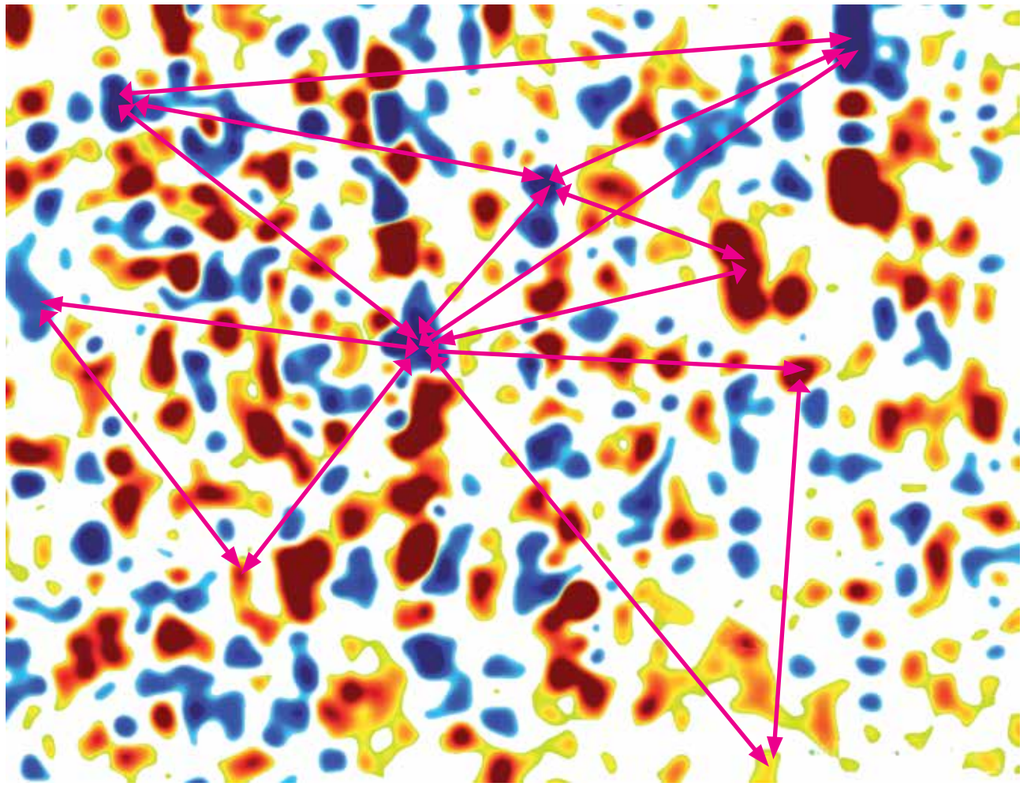


OPTIMALISEREN MET BIG DATA

hoe door de scalability wall heen te breken?

Het integreren van local search technieken in een mathematische programmerings-solver

Thierry Benoist, Julien Darlay, Bertrand Estellon, Frédéric Gardi & Romain Megel

Er bestaat geen twijfel over dat een van de meest krachtige gereedschappen van de Operationele Research (OR) het gemengde geheeltallige lineaire programmeren oftewel Mixed Integer Linear Programming (MIP), is. Het eenvoudige gebruik ervan zorgt ervoor dat OR-specialisten hier vaak een beroep op doen: de gebruiker modelleert het probleem als een geheeltallig lineair model en het MIP-programma lost dit op door middel van enumeratie en snijvlakgeneratie technieken. Deze 'modelleer en voer uit' benadering kan de inspanningen voor de ontwikkeling en het onderhoud van optimalisatiesoftware aanzienlijk verminderen. Andere zoekboomgerelateerde technologieën zoals Constraint Programming (CP) volgen dezelfde weg.

Echter, ondanks de opmerkelijke progressie in de afgelopen vijftien jaar (Bixby, 2012), lukt het MIP-solvers niet om veel grootschalige combinatorische problemen uit de praktijk van de OR aan te kunnen. Het *car sequencing* probleem (Estellon et al., 2008), een klassiek probleem in de OR-literatuur, is een goed voorbeeld. Bij dit probleem lukt het MIP-solvers tot op heden niet om toegelaten oplossingen van problemen te vinden, zelfs al hebben ze slechts duizenden binaire beslissingen. Bovendien, ook al zouden ze toegelaten oplossingen hebben gevonden dan is de lineaire relaxatie toch niet goed genoeg om de exponentiële *branch-and-bound tree* op een efficiënte manier te verkleinen. De zoektocht levert ook na uren van berekenen geen toegelaten oplossing. Dit is wat wij de *scalability wall* noemen.

Als we grootschalige problemen willen aanpakken, is men strikt gesproken zelden geïnteresseerd in het vinden van de optimale oplossing. Ook zal men beseffen dat het bewijzen van het optimum een utopie is. Bovendien zijn kwalitatief hoogstaande oplossingen over het algemeen gesproken voldoende, vooral wanneer de wiskundige modellen slechts een benadering geven van de alledaagse situatie. Ten slotte, wat wij observeren als OR-beoefenaren, is dat vandaag de dag eindgebruikers steeds meer interactie willen met de beslissingsondersteunende systemen: ze willen snel kwalitatief goede oplossingen.

Rest de vraag wat OR-professionals doen als MIP- of CP-solvers ineffectief zijn? Normaal gesproken implementeren ze een specifieke *local search* heuristiek. In tegenstelling tot bovengenoemde zoekboom technieken, bestaat Local Search (LS) uit het iteratief toepassen van veranderingen aan een oplossing (stappen genoemd) teneinde de doelfunctie te verbeteren. Hoewel deze techniek incompleet is, wordt zij alom gewaardeerd omdat het de gebruikers toestaat om kwalitatief goede oplossingen te verkrijgen in korte rekentijden (in de orde van minuten voor zeer grootschalige instanties. Het ontwikkelen en implementeren van local search algoritmes is echter niet eenvoudig, zelfs niet als er al enkele raamwerken zijn ontwikkeld om de programmeur te helpen in deze taak. De algoritmische laag is vooral moeilijk te ontwikkelen omdat het een expertise in algoritmes én een behendigheid in programmeren vereist (zie Aarts en Lenstra (1997) voor een overzicht van het LS paradigma en zijn applicaties).

Deze situatie heeft geleid tot de ontwikkeling van LocalSolver[1], een mathematische programmeringssolver gebaseerd op local search. Het doel van het project, dat in 2007 is gestart, was het combineren van de eenvoud van het gebruik van een 'modelleer-en-voer uit' solver en de kracht van local search technieken voor combinatorische optimalisering. We zullen laten zien hoe het OR- beoefenaren toestaat zich te focussen op de modellering van het probleem, terwijl er gebruik wordt gemaakt van een eenvoudig formalisme, en daarna de actuele vóór-oplossing over te dragen aan een solver die gebaseerd is op efficiënte en betrouwbare local search technieken.

Modellering

LocalSolver bevat een innovatieve wiskundig gemodelleerde scripttaal voor snelle prototypering, namelijk LSP (Local Search Programming). We zullen deze taal gebruiken in onze voorbeelden hoewel sommige object-georiënteerde API's ook beschikbaar zijn voor volledige integratie (C++, Java, .NET). LocalSolvers modelleringsformalisme lijkt op klassieke wiskundige programmeringsformalismes maar is daarnaast verrijkt met algemene wiskundige operatoren. Deze staan een natuurlijke modellering van reële problemen toe en maken het daardoor eenvoudiger voor OR-deskundigen. Onderstaande regels definiëren bijvoorbeeld het model van het rugzakprobleem (*knapsack problem*) met n objecten waarvan het gewicht en de waarde is gegeven.

```
function model() {
        for [i in 1..n] x[i] <- bool();
knapsackWeight <- sum[i in 1..n](weights[i] * x[i]);
constraint knapsackWeight <= knapsackBound;

knapsackValue <- sum[i in 1..n](values[i] * x[i]);
maximize knapsackValue; }
```

In dit basisvoorbeeld zijn binaire beslissingsvariabelen x[i] geïntroduceerd met het bool()-statement. Vervolgens is het gewicht in de *knapsack* geïntroduceerd als een som-expressie en is er een voorwaarde gesteld aan deze waarde. Tenslotte is de waarde in de *knapsack* gedefinieerd als een maximalisatie-doel. Merk op dat verscheidene doelfuncties toegevoegd kunnen worden, welke kunnen worden geïnterpreteerd als een lexicografische doelfunctie. Het cruciale punt hier is dat er verder niets meer hoeft te worden gedefiniëerd en vooral dat er geen oplossingsomgevingen hoeven worden meegegeven. Alleen dit model wordt meegegeven aan de solver, verder niets. Het is de verantwoordelijkheid van de solver om een search algoritme toe te passen dat kan werken aan de abstracte combinatorische structuur welke is geïnduceerd door het model van de gebruiker. In eerste

ARITHMETIC	LOGICAL	RELATIONAL
sum, min, max, prod, div, mod, log, exp, pow, sin, cos, tan, abs, dist, sqrt, floor, ceil, round	and, or, not, xor, if, at	==, !=, <=, >=, <, >

Tabel 1. Mathematische operatoren beschikbaar in LocalSolver

instantie voor het vinden van een toegelaten oplossing en vervolgens voor het iteratief verbeteren van deze oplossing. De belangrijkste principes van deze stappen worden gegeven in de volgende sectie.

Hoewel dit eenvoudige *knapsack*-voorbeeld alleen gebruik maakt van linaire expressies, staan de onderliggende oplostechnieken het gebruik toe van nonlineaire operatoren, inclusief conditionele expressies (*als A, dan B of C*, geschreven als A ? B : C) of zelfs *array indexering* (de expressie A[N] staat voor het *N-de element in array A*) toe. Tabel 1 geeft een lijst van beschikbare operatoren. Het introduceren van logische, rekenkundige of relationele operatoren heeft twee belangrijke voordelen in een local search verband: expressiviteit en doeltreffendheid. Met dergelijke *low-level* operatoren is modelleren gemakkelijker dan met de basis MIP syntax, terwijl het behouden van de basis syntax zorgt voor een snellere acceptatie door beginners (vooral voor degenen die niet vertrouwd zijn met het computerprogrammeren). Bovendien kunnen de invarianten die geïnduceerd zijn door deze operatoren benut worden door de interne algoritmes van de LS solver om daarmee de local search te versnellen.

Beschouw bijvoorbeeld het *P*-mediaan probleem (Beasley, 1985), dat bestaat uit het selecteren van een deelverzameling *S* van *P* steden uit *N* (voor bijvoorbeeld het vestigen van openbare voorzieningen), zo dat de som van de afstanden van elke stad naar elke dichtstbijzijnde stad in *S* geminimaliseerd wordt. In onderstaand model laat het gebruik van conditionele en min-operatoren toe dat het model bijna letterlijk geformuleerd kan worden als hierboven. Deze eenvoud levert een model op dat alleen gefocust is op de relevante beslissingsvariabelen, namelijk het selecteren of niet selecteren van elke stad x[i]. Na het gelijkstellen van de som van deze variabelen aan *P*, wordt de minDistance[i] van elke stad *i* naar de dichtstbijzijnde stad in *S* geschreven als het minimum van de afstanden naar de andere steden. De afstand naar steden buiten *S* worden beschouwd als oneindig (InfiniteDistance kan worden gesteld als maximumwaarde in de afstandsmatrix). De doelfunctie zal de som zijn van deze minDistance[i].

Model P-median probleem

```
function model() {
    x[1..N] <- bool();
    constraint sum[i in 1..N](x[i]) == P;

    minDistance[i in 1..N] <- min[j in 1..N]
            (x[j] ? distance[i][j] : InfiniteDistance);
    minimize sum[i in 1..N] (minDistance[i]); }
```

Alles wat de solver moet doen is in enkele seconden een kwalitatief goede oplossing vinden van deze probleemstelling (een gemiddeld verschil van 0,6% van de 40 instanties van de OR-Library met een tijdslimiet van één minuut). In de volgende sectie zullen we het interne mechanisme beschrijven dat dit mogelijk maakt.

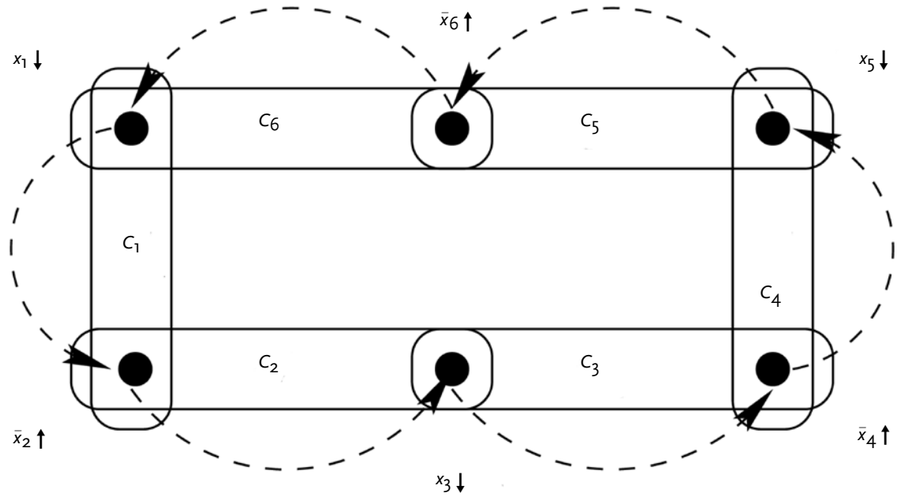
Oplossen

Onze benadering van een autonoom Local Search werd bepaald door het volgende fundamentele principe: *de LS solver moet werken zoals een LS beoefenaar werkt*. Dit impliceert dat LocalSolver gestructureerde stappen uitvoert om te zorgen dat de haalbaarheid van de oplossingen bij elke iteratie gehandhaafd blijft, waarbij de evaluatie wordt versneld door het gebruik van invarianten die feitelijk zijn geïnduceerd door de structuur van het model. De sleutelcomponenten voor de efficiëntie van het algoritme zijn: een incrementele machine die in staat is om zeer snel de impact van een transformatie van de oplossing te evalueren; meervoudige autonome stappen die de combinatorische structuur van het model gebruiken om *toegelaten* omgevingen van een oplossing te ontdekken; een globale adaptieve zoekstrategie die de zoekopdracht leidt naar kwalitatief goede oplossingen.

De incrementele machine is gebaseerd op een representatie van het model als een directe a-cyclische graaf (DAG), waarvan de wortels de beslissingen zijn en de bladeren de beperkingen en de doelstellingen. De interne knopen van deze DAG zijn de operatoren uit tabel 1. Het toevoegen van stappen aan een bestaande oplossing bestaat uit het wijzigen van de waarden van de beslissingen (wortels) en het evalueren van de beperkingen en doelstellingen (bladeren) door het verspreiden van deze wijzigingen in de DAG. Door een bijzonder geoptimaliseerd algoritme kunnen er miljoenen stappen per minuut worden berekend in *real-life* modellen.

De meest eenvoudige generieke stap is de verandering van de waarde van een beslissing, zoals bijvoorbeeld het veranderen van de waarde van een x[i] beslissing in ons P-mediaan voorbeeld. Echter, als we van één toegelaten oplossing naar een andere willen gaan, zoals in de meeste local search algoritmes, zijn er meer gestructureerde stappen nodig. De solver construeert dergelijke stappen op grond van het analyseren van de structuur van het model. Bijvoorbeeld, de kardinaliteits-voorwaarde in het P-mediaan probleem wordt gebruikt voor het ontwerpen van stappen die het aantal geselecteerde steden constant houdt en daarmee de toelaatbaarheid van de oplossing waarborgt. Meer algemeen, het volgen van *ejection chains* of *ejection cycles* in de *constraint hypergraph* (figuur 1) levert zeer krachtige stappen op. Naast deze kleine omgevingen, kunnen grotere omgevingen worden verkend door het combineren van kleinere omgevingen binnen een verwerp & repareer mechanisme. Tenslotte kan de identificatie van speciale combinatorische structuren de activatie van specifieke omgevingen op gang brengen. Merk daarbij op dat deze omgevingen op verschillende manieren kunnen worden verkend: vrijwel willekeurig of door zich te richten op stappen met een grotere kans op succes.

De verkenning van de zoekruimte is verdeeld langs verschillende lijnen die periodiek onderling worden gesynchroniseerd. De globale verandering van de zoekopdracht wordt gewaarborgd door simulated annealing met *reheating* en herstart mechanismen. Statistieken over de prestaties van de stappen worden benut om de algehele prestaties van de zoekopdracht te verbeteren.



Figuur 1. Een *ejection cycle* met zes boolean variabelen x_1, x_3, x_5 (waarvan de huidige waarde 1 is) en $\bar{x}_2$, $\bar{x}_4$, $\bar{x}_6$ (waarvan de huidige waarde 0 is), en zes beperkende sommaties C_1, ..., C_6. Elke variabele behoort tot twee sommaties (bijvoorbeeld, x_1 behoort tot C_1 en C_6). Nu geldt dat x_1, x_3, x_5 zijn afgenomen (↓), terwijl $\bar{x}_2$, $\bar{x}_4$, $\bar{x}_6$ zijn toegenomen (↑). Deze stap bewaart de waarden van de sommaties, en dus de toegelatenheid van de voorwaarden.

Testen en conclusies

Ondanks hun schijnbare conceptuele eenvoud leveren de uitgangspunten van de voorgaande sectie opmerkelijk goede resultaten in de praktijk, vooral voor grootschalige combinatorische optimaliseringsproblemen die buiten het gebied van de state-of-the-art wiskundige programmerings solvers vallen. Bij een aantal van de moeilijkste

MIPLIB-voorbeelden[2], overtrof LocalSolver onlangs de beste MIP-solvers. Een voorbeeld is het *car sequency* probleem[3], dat bestaat uit het plannen van auto's naar verf- en assemblagelijnen rekening houdend met sequency voorwaarden. De oplossing van LocalSolver was na 10 seconden beter dan die van de beste MIP-solvers in één dag rekentijd.

Het vermogen om grootschalige combinatorische problemen aan te pakken op deze model-and-run wijze werd geïllustreerd tijdens de Roadef/Euro Challenge 2012[4], waar de opdracht was om de processen op Google-servers te herschikken, rekening houdend met verschillende bronnen en afhankelijkheidsvoorwaarden. Als 25ste gerangschikt van de 82 deelnemende teams was LocalSolver de enige model-and-run, general-purpose solver die zich kwalificeerde voor de finaleronde. Het team van LocalSolver paste een honderd regels bevattend model toe dat binnen één dag tijd was geschreven. Meer resultaten op zowel academische als industriële problemen kunnen worden gevonden in Benoist et al. (2011) of op de website van LocalSolver.[5]

In 2012 was LocalSolver gereed om van een onderzoeksproject over te gaan naar een commercieel product. Heden ten dage wordt het in verschillende industrieën over de hele wereld gebruikt, van het maximaliseren van televisiereclame-inkomsten in Frankrijk tot het optimaliseren van de toelevering van een bakkerij in Japan (een niet-lineair probleem met drie miljoen 0-1 beslissingen!). LocalSolver ontwikkelt zich in de richting van een volledige, alles-in-een programmerings solver met alle optimalizatietechnieken (Mixed-integer en niet-lineaire programmeringstechnieken, constraint programming en satisfiability technieken, local en direct search technieken). De volgende versie 4.0, die gepland staat voor het eind van 2013, zal de eerste stap zijn in de richting van deze nieuwe generatie wiskunde programmerings solvers voor grootschalige mixed-variable niet-convexe optimalisatie. Deze nieuwe versie zal verscheidene nieuwe belangrijke functies bieden vanuit functioneel en technisch oogpunt: kleine omgeving stappen om te optimaliseren over continue of gemengde beslissingen; verkenning van grote, exponentieel grote omgevingen bij 0-1 of gemengde beslissingen gebruikmakend van enkele zoekboomtechnieken (bijvoorbeeld afrondingsheuristieken op basis van lineaire relaxatie); verkenning van grote omgevingen bij continue beslissingen door het herzien van opeenvolgende lineaire programmeertechnieken voor niet-lineair programmeren (gebaseerd op een simplexalgoritme); berekening van ondergrenzen gecombineerd met voorwaarden propagatie en duale lineaire relaxatie.

Noten

1. www.localsolver.com
2. www.localsolver.com/misc/LocalSolver_ECCO_2013.pdf
3. www.challenge.roadef.org/2005/en
4. http://localsolver.com/new.html?id=3
5. www.localsolver.com

Literatuur

Aarts, E., Lenstra, J. (1997). *Local search in combinatorial optimization.* John Wiley & Sons.

Bixby, R. (2012). A brief history of linear and mixed-integer programming computation. In M. Grötschel (Ed.), *Optimization Stories, 21st International Symposium on Mathematical Programming, Berlin, August 19-24, 2012,* (pp. 107-121). Bielefeld: Documenta Mathematica.

Beasley, J. (1985). A note on solving large p-median problems. *European Journal of Operational Research, 21*(2), 270-273.

Benoist, T., Estellon, B., Gardi, F., Megel, R., Nouioua, K. (2011). Localsolver 1.x: a black-box local-search solver for 0-1 programming. *4OR-Quarterly Journal of Operations Research* 9(3), 299-316.

Estellon, B., Gardi, F., Nouiouna, K. (2008). *Two local search approaches for solving real-life car sequencing problems. European Journal of Operational Research,* 191(3), 928-944

Thierry Benoist, Julien Darlay, Bertrand Estellon, Frédéric Gardi & Romain Megel zijn werkzaam bij LocalSolver te Parijs.
E-mail: <contact@localsolver.com>

Johan van Leeuwaarden

column

Wiskunde is *à la mode*

In 2011 sprak ik bij de finale van de Nederlandse Wiskunde Olympiade. Van de ruim 5.000 leerlingen die meededen aan de voorronde kwamen op die dag de beste 150 naar de finaleronde om zich te kwalificeren voor het Nederlands team. Tijdens de wedstrijd gaf ik mijn presentatie. Niet voor de finalisten, die waren driftig in de weer met sommen, maar voor hun ouders.

De meeste ouders keken aanvankelijk wat angstig, maar raakten gaandeweg geboeid, zeker toen ik na wat formules aangaf dat wiskunde ook nuttig kan zijn, en dat de maatschappij wiskundigen hard nodig heeft. Ik begrijp de ouders ook wel. Je zult maar een kind hebben dat wiskunde leuk vindt, en zelfs overweegt het te gaan studeren. Wat zeg je tegen zo'n kind, als ouder? Zijn er geen andere studies, waar je later wat meer mee kunt? Ik ben geneigd om nee te zeggen.

Wiskunde is niet stoer in Nederland. Nog steeds laten velen op feestjes geen kans onbenut om te zeggen hoe slecht ze wel niet in wiskunde waren. Vreemd, want met onze sportprestaties doen we in de regel het tegenovergestelde. Het tij lijkt echter wel te keren. De laatste jaren gaan meer scholieren wiskunde studeren. In 2006 nog lag de jaarlijkse instroom in Nederland onder de 200, maar sindsdien is er een duidelijke opwaartse trend zichtbaar. De instroom is sinds 2006 grofweg verdrievoudigd. Wiskunde wordt steeds populairder.

Deze ontwikkeling laat zich moeilijk verklaren. Ligt het aan de toegenomen media-aandacht, of betere voorlichting op de middelbare school? Dat zal vast helpen. Maar ik denk dat het bovenal een natuurlijke ontwikkeling is. Een kwestie van vraag en aanbod. Uit een onderzoek van *Elsevier* van deze zomer bleek dat wiskundigen vrijwel meteen een vaste baan vinden, in de huidige arbeidsmarkt vrij zeldzaam. De maatschappij – bedrijfsleven, onderwijs en wetenschap – ontvangt je met open armen.

Goed om te weten, en ook wel stoer, zou je zeggen. Maar zie hier de spagaat van de wiskundige. Enerzijds is er het verlangen naar die abstracte wereld, met de uitdagende som en de prachtige formule, en anderzijds is er de roep uit de echte wereld van de toepassing. En toch kan dat samengaan. Als wij voorlichting geven aan middelbare scholieren, of beter nog, aan hun ouders, dan kunnen we eerst een prachtige stelling bewijzen, dan een maatschappelijk relevante toepassing laten zien, en op de laatste slide de harde carrièrecijfers tonen. Daarmee zeggen we niets te veel, en kiest een scholier voor een wiskundeopleiding, dan kunnen we de drie elementen van de presentatie ook waarmaken.

Johan van Leeuwaarden is hoogleraar wiskunde aan de TU Eindhoven.
E-mail: <j.s.h.v.leeuwaarden@TUE.nl>