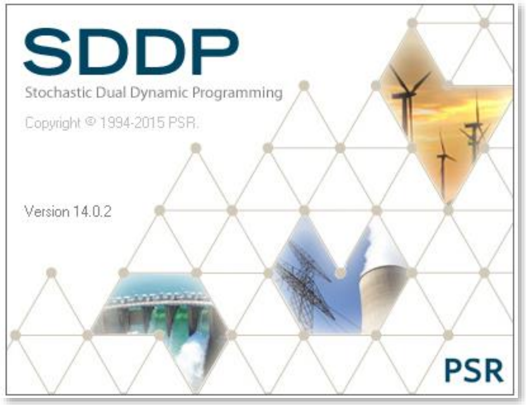
PSR

Stochastic programming in energy systems


SDDP
Stochastic Dual Dynamic Programming
Copyright © 1994-2015 PSR.
Version 14.0.2
PSR

JuMP Developers meet-up

Boston, June 13, 2017

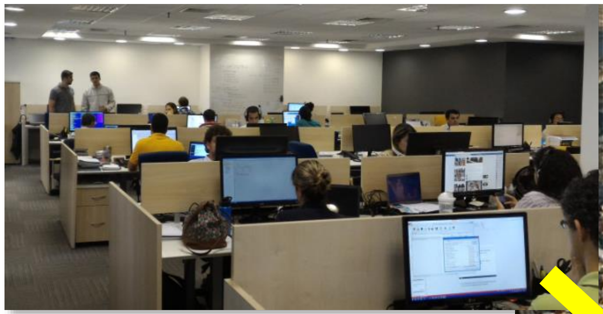
Agenda

- PSR & Problems we want/like to solve
- The begining of julia
- Projects in julia & JuMP
- Research
- SDDP + JuMP = S2
- OptFlow: Non-Linear Modelling
- Optgen: MILP & SDDiP

PSR

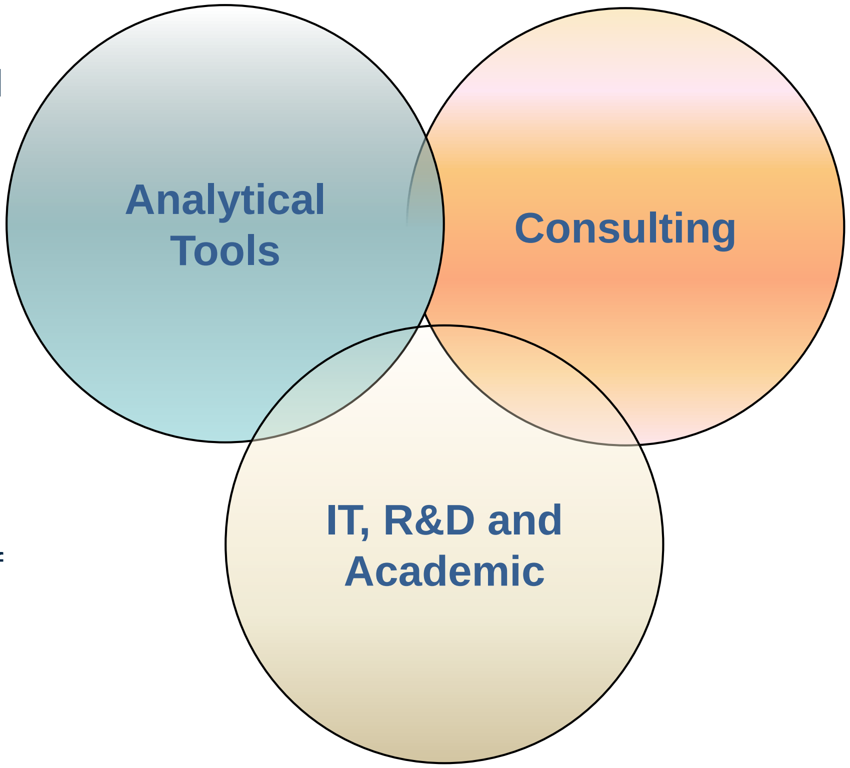
Provider of analytical solutions and consulting services in electricity and natural gas since 1987

Our team has more than 60 experts (17 PhDs, 31 MSc) in engineering, optimization, energy systems, statistics, finance, regulation, IT and environment analysis

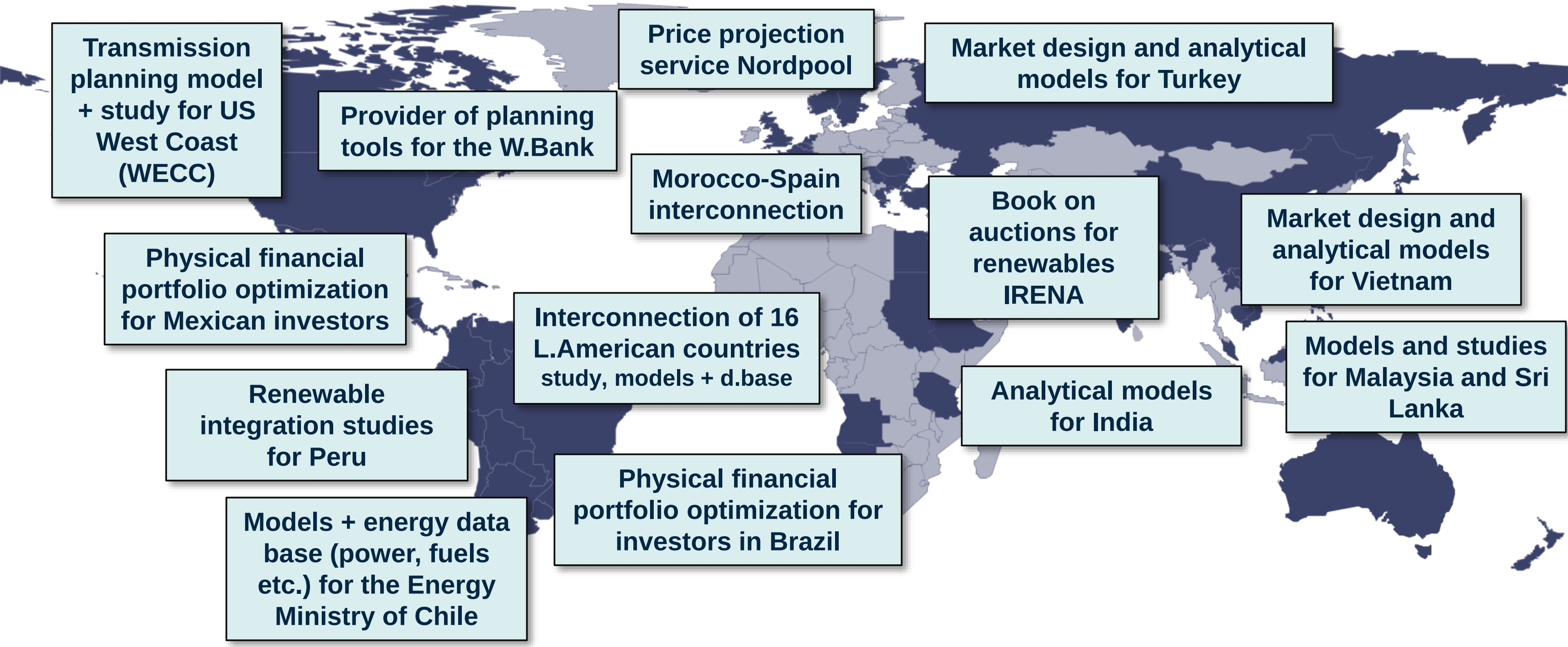


Core activities

- Development of analytical tools
  - integrated economic + technical + environmental evaluation of energy investment opportunities
- Strategic advice
  - to investors, lenders and governments (economic, financial, regulatory, risk assessment)
- Market & regulatory design
  - (e.g. organization of auctions, energy integration of Latin American countries, etc)
- Environmental analysis
  - (e.g. integrated river basin development, CO_2 emissions, due diligence, etc)
- Engineering studies
  - (e.g. generation & transmission planning, automated river-basin inventory, etc)



Some recent projects


Transmission planning model + study for US West Coast (WECC)
Provider of planning tools for the W.Bank
Price projection service Nordpool
Market design and analytical models for Turkey
Morocco-Spain interconnection
Book on auctions for renewables IRENA
Market design and analytical models for Vietnam
Physical financial portfolio optimization for Mexican investors
Interconnection of 16 L.American countries study, models + d.base
Models and studies for Malaysia and Sri Lanka
Renewable integration studies for Peru
Analytical models for India
Physical financial portfolio optimization for investors in Brazil
Models + energy data base (power, fuels etc.) for the Energy Ministry of Chile

- **Americas**: all countries in South and Central America, United States, Canada and Dominican Republic
- **Europe**: Austria, Spain, France, Norway (Nordic region), Belgium, Turkey and the Balkan region
- **Asia**: provinces in China (including Shanghai, Sichuan, Guangdong and Shandong), Philippines, Singapore, Malaysia, Kirgizstan, Sri Lanka, Tajikistan and India
- **Oceania**: New Zealand
- **Africa**: Tanzania, Namibia, Egypt, Angola, Sudan, Ethiopia and Ghana
- **More than 300 active licenses worldwide**

Main planning and scheduling tools

- **SDDP**: optimal mid and long-term stochastic production scheduling
- **OPTGEN**: optimal mid and long term capacity expansion planning
- **OPTFLOW:** optimal power flow for reactive expansion
- All models optimize complex hydrothermal systems with renewable generation, transmission networks, price-responsive loads, gas pipelines and fuel storage

The beginning

- Just arrived from US started internship in PSR (2015)
  - Me: Born in C but as a controls and telecommunication engineer had many hours of MATLAB and R
- First day: A maintenance scheduling model
  - Mario: “Do you know julia?”
  - julia+JuMP X Mosel

The beginning

$$\max \delta$$

1) δ is smaller than all reserves in all stages:

$$\delta \leq \frac{1}{d_t}\left(\sum_{i=1}^{II} \alpha_{H\,ti}(1-\varphi_{H\,i})g_{H\,ti} + \sum_{j=1}^{JJ} \alpha_{T\,j}(1-\varphi_{T\,j})g_{T\,j} + \sum_{l=1}^{LL} r_{tl} - d_t \right), \forall t \in \{1..TT\}$$

2) Maintenance lower and upper bounds per stage (representing forced stop and forced activation periods)

$$\underline{\alpha_{H}}_{ti} \leq \alpha_{H\,ti} \leq \overline{\alpha_{H\,ti}}, \forall t \in \{1..TT\}, i \in \{1..II\}$$

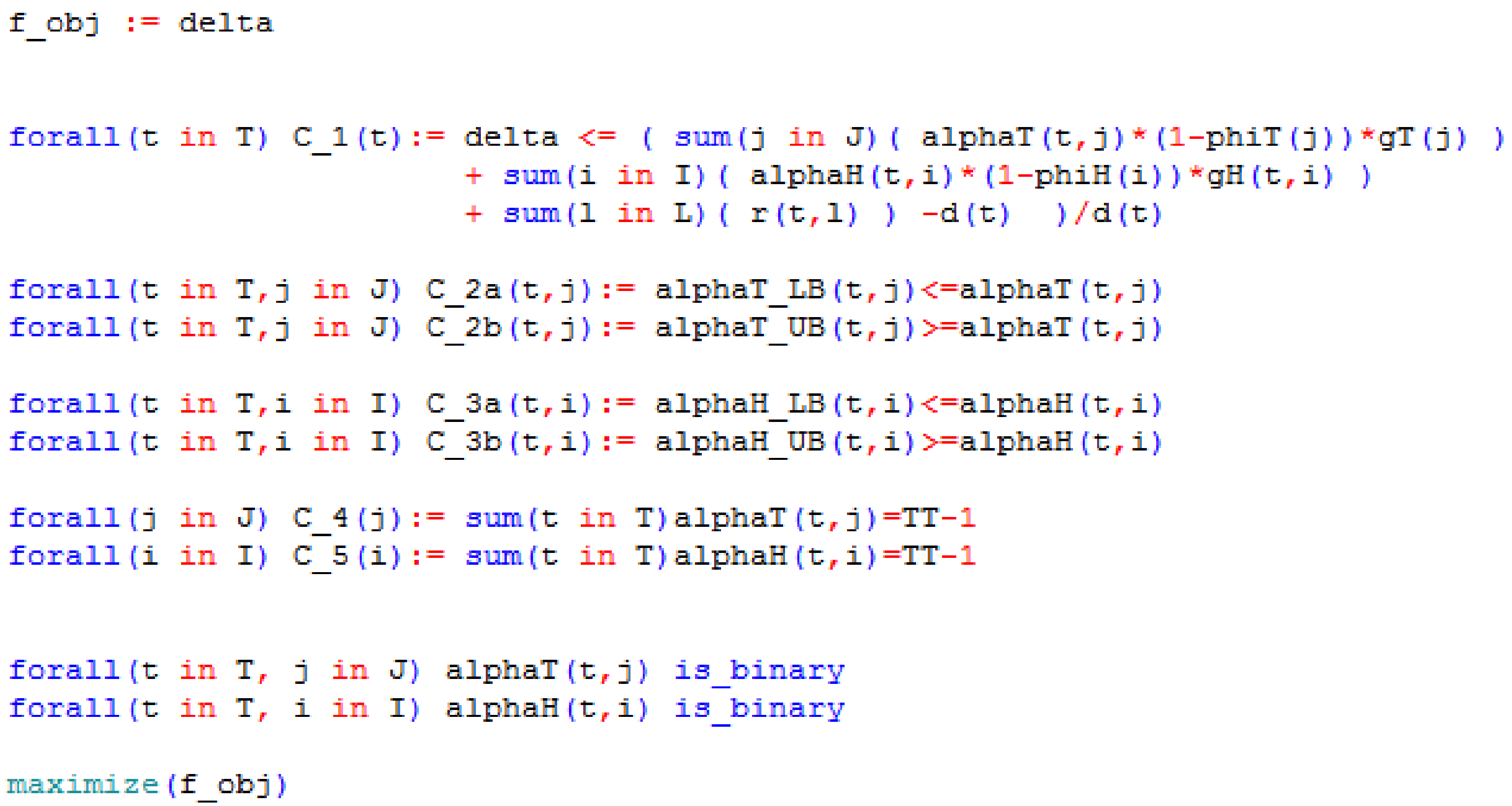
$$\underline{\alpha_{T}}_{tj} \leq \alpha_{T\,tj} \leq \overline{\alpha_{T\,tj}}, \forall t \in \{1..TT\}, j \in \{1..JJ\}$$

3) Maintenance obligation, once per cycle

$$\sum_{t=1}^{TT} \alpha_{H\,ti} = TT - 1, \forall i \in \{1..II\}$$

$$\sum_{t=1}^{TT} \alpha_{T\,tj} = TT - 1, \forall j \in \{1..JJ\}$$

The beginning



```
f_obj := delta


forall(t in T) C_1(t):= delta <= ( sum(j in J)( alphaT(t,j)*(1-phiT(j))*gT(j) )
                               + sum(i in I)( alphaH(t,i)*(1-phiH(i))*gH(t,i) )
                               + sum(l in L)( r(t,l) ) -d(t)  )/d(t)

forall(t in T,j in J) C_2a(t,j):= alphaT_LB(t,j)<=alphaT(t,j)
forall(t in T,j in J) C_2b(t,j):= alphaT_UB(t,j)>=alphaT(t,j)

forall(t in T,i in I) C_3a(t,i):= alphaH_LB(t,i)<=alphaH(t,i)
forall(t in T,i in I) C_3b(t,i):= alphaH_UB(t,i)>=alphaH(t,i)

forall(j in J) C_4(j):= sum(t in T)alphaT(t,j)=TT-1
forall(i in I) C_5(i):= sum(t in T)alphaH(t,i)=TT-1


forall(t in T, j in J) alphaT(t,j) is_binary
forall(t in T, i in I) alphaH(t,i) is_binary

maximize(f_obj)
```

The beginning

```
@objective(m, Max, delta);
```

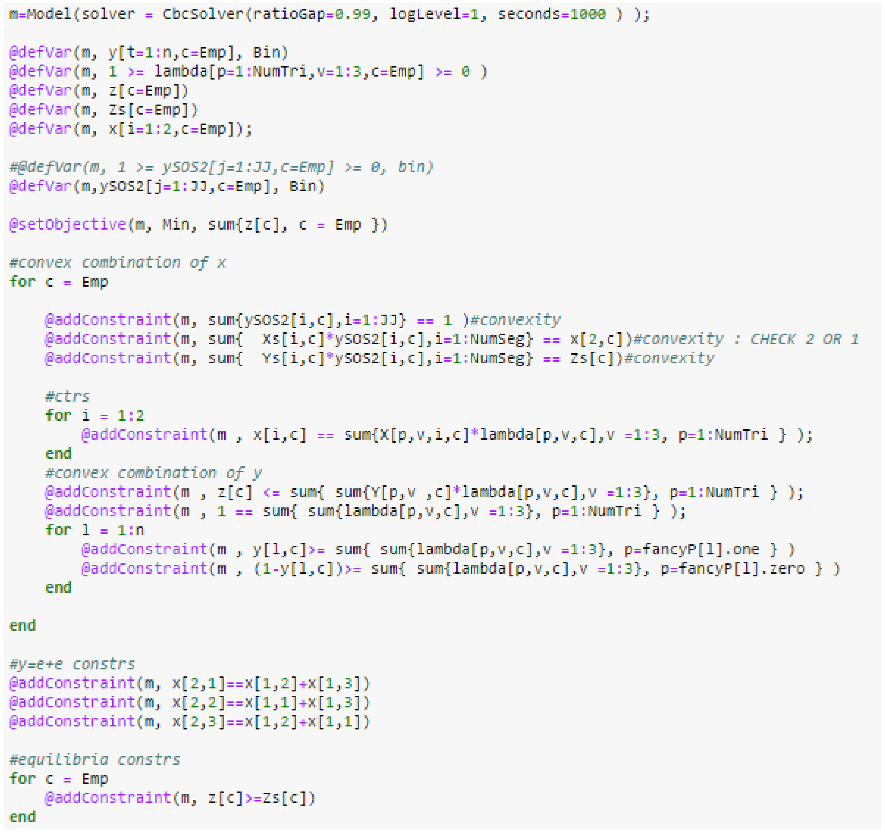
```
for t = 1:TT
    @constraint(m ,delta <=(  sum{( (1-phiH[i])*gH[t,i]*alphaH[t,i] ),i = 1:II}
        + sum{(1-phiT[j])*gT[j]*alphaT[t,j] ,j =1:JJ} + sum{( r[t,l] ),l=1:LL} -d[t]  )/d[t])
end;
```

```
for t = 1:TT, j = 1:JJ
    @constraint(m, alphaT[t,j] >= alphaT_LB[t,j])
    @constraint(m, alphaT[t,j] <= alphaT_UB[t,j])
    @constraint(m, alphaH[t,j] >= alphaH_LB[t,j])
    @constraint(m, alphaH[t,j] <= alphaH_UB[t,j])
end
```

```
for i = 1:II
    @constraint(m, sum{alphaH[t,i],t=1:TT} == TT-1)
end
for j = 1:JJ
    @constraint(m, sum{alphaT[t,j],t=1:TT} == TT-1)
end
```

The beginning

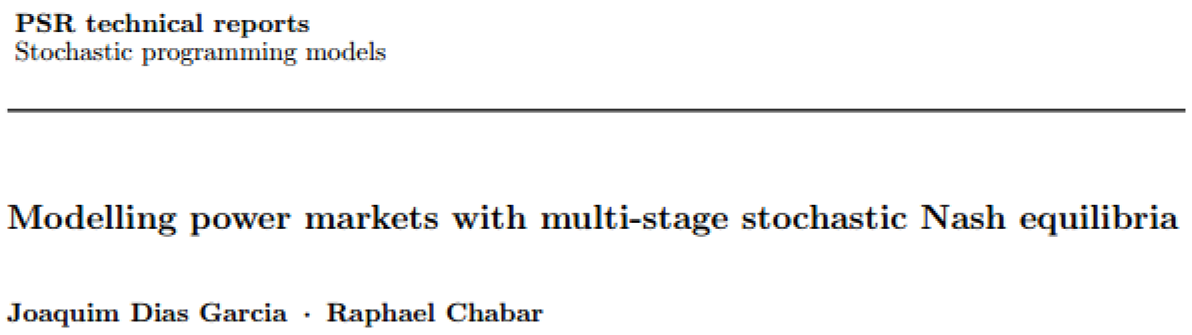
- Bivariate functions as MILP from paper:
  - **Mixed-Integer Models for Nonseparable Piecewise Linear Optimization: Unifying Framework and Extensions**
    - Vielma, Ahmed, Nemhauser



```
m=Model(solver = CbcSolver(ratioGap=0.99, logLevel=1, seconds=1000 ) );

@defVar(m, y[t=1:n,c=Emp], Bin)
@defVar(m, 1 >= lambda[p=1:NumTri,v=1:3,c=Emp] >= 0 )
@defVar(m, z[c=Emp])
@defVar(m, Zs[c=Emp])
@defVar(m, x[i=1:2,c=Emp]);

#@defVar(m, 1 >= ySOS2[j=1:JJ,c=Emp] >= 0, bin)
@defVar(m,ySOS2[j=1:JJ,c=Emp], Bin)

@setObjective(m, Min, sum{z[c], c = Emp })

#convex combination of x
for c = Emp

    @addConstraint(m, sum{ySOS2[i,c],i=1:JJ} == 1 )#convexity
    @addConstraint(m, sum{  Xs[i,c]*ySOS2[i,c],i=1:NumSeg} == x[2,c])#convexity : CHECK 2 OR 1
    @addConstraint(m, sum{  Ys[i,c]*ySOS2[i,c],i=1:NumSeg} == Zs[c])#convexity

    #ctrs
    for i = 1:2
        @addConstraint(m , x[i,c] == sum{X[p,v,i,c]*lambda[p,v,c],v =1:3, p=1:NumTri } );
    end
    #convex combination of y
    @addConstraint(m , z[c] <= sum{ sum{Y[p,v ,c]*lambda[p,v,c],v =1:3}, p=1:NumTri } );
    @addConstraint(m , 1 == sum{ sum{lambda[p,v,c],v =1:3}, p=1:NumTri } );
    for l = 1:n
        @addConstraint(m , y[l,c]>= sum{ sum{lambda[p,v,c],v =1:3}, p=fancyP[l].one } )
        @addConstraint(m , (1-y[l,c])>= sum{ sum{lambda[p,v,c],v =1:3}, p=fancyP[l].zero } )
    end

end

#y=e+e constrs
@addConstraint(m, x[2,1]==x[1,2]+x[1,3])
@addConstraint(m, x[2,2]==x[1,1]+x[1,3])
@addConstraint(m, x[2,3]==x[1,2]+x[1,1])

#equilibria constrs
for c = Emp
    @addConstraint(m, z[c]>=Zs[c])
end
```

The beginning

- Optfolio: Portfolio optimization
  - Initially JuMP+CBC, first motivation for Xpress.jl
- Graduation project
  - SDDP Notebook
  - **Modelling power markets with multi-stage stochastic Nash equilibria**
  - www.psr-inc.com/publications/scientific-production/papers/

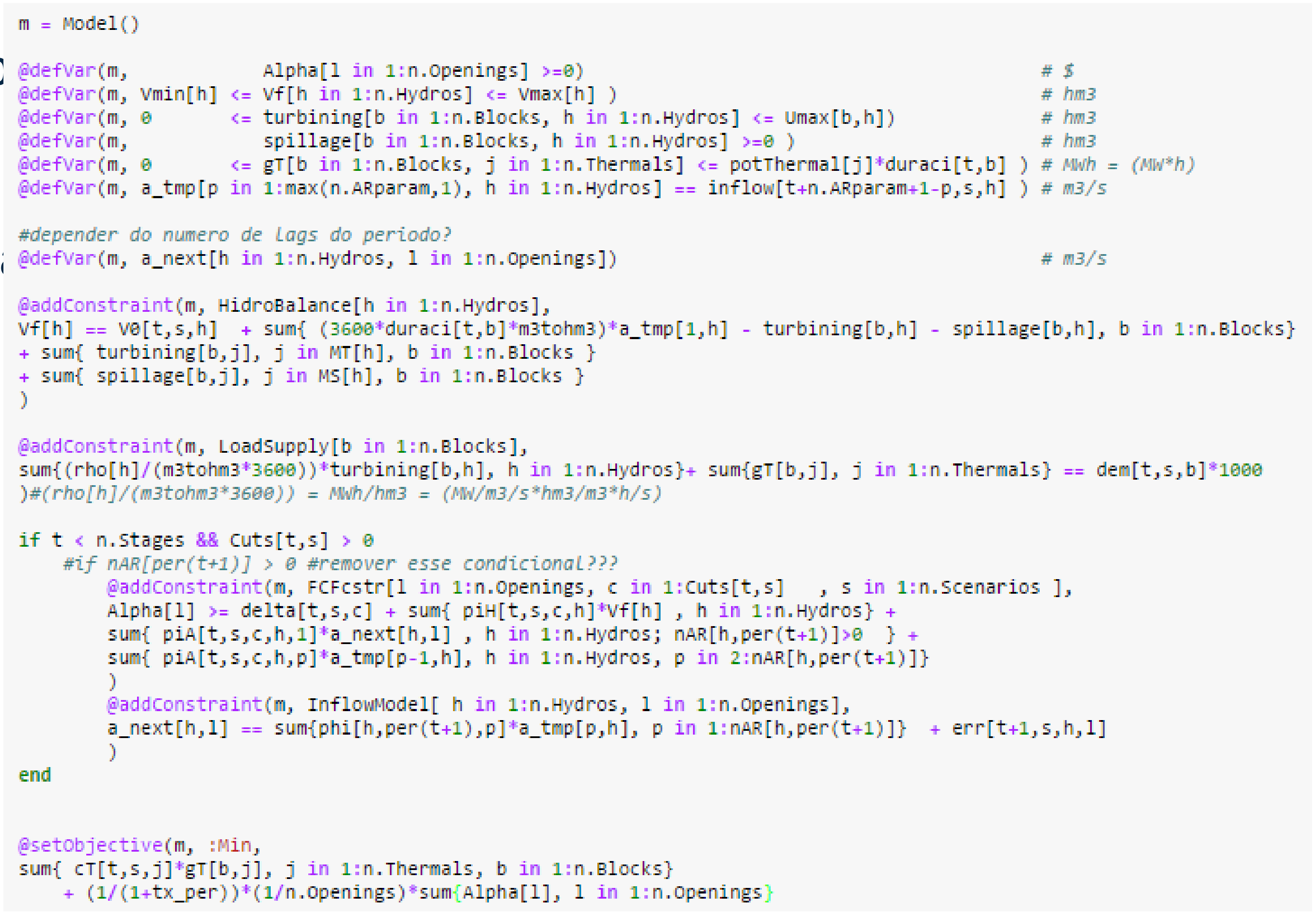

PSR technical reports
Stochastic programming models

Modelling power markets with multi-stage stochastic Nash equilibria

Joaquim Dias Garcia · Raphael Chabar

The beginning

- Op
  -
- Gr
  -
  -



```
m = Model()

@defVar(m,            Alpha[l in 1:n.Openings] >=0)                                       # $
@defVar(m, Vmin[h] <= Vf[h in 1:n.Hydros] <= Vmax[h] )                                    # hm3
@defVar(m, 0       <= turbining[b in 1:n.Blocks, h in 1:n.Hydros] <= Umax[b,h])           # hm3
@defVar(m,            spillage[b in 1:n.Blocks, h in 1:n.Hydros] >=0 )                    # hm3
@defVar(m, 0       <= gT[b in 1:n.Blocks, j in 1:n.Thermals] <= potThermal[j]*duraci[t,b] ) # MWh = (MW*h)
@defVar(m, a_tmp[p in 1:max(n.ARparam,1), h in 1:n.Hydros] == inflow[t+n.ARparam+1-p,s,h] ) # m3/s

#depender do numero de lags do periodo?
@defVar(m, a_next[h in 1:n.Hydros, l in 1:n.Openings])                                    # m3/s

@addConstraint(m, HidroBalance[h in 1:n.Hydros],
Vf[h] == V0[t,s,h]  + sum{ (3600*duraci[t,b]*m3tohm3)*a_tmp[1,h] - turbining[b,h] - spillage[b,h], b in 1:n.Blocks}
+ sum{ turbining[b,j], j in MT[h], b in 1:n.Blocks }
+ sum{ spillage[b,j], j in MS[h], b in 1:n.Blocks }
)

@addConstraint(m, LoadSupply[b in 1:n.Blocks],
sum{(rho[h]/(m3tohm3*3600))*turbining[b,h], h in 1:n.Hydros}+ sum{gT[b,j], j in 1:n.Thermals} == dem[t,s,b]*1000
)#(rho[h]/(m3tohm3*3600)) = MWh/hm3 = (MW/m3/s*hm3/m3*h/s)

if t < n.Stages && Cuts[t,s] > 0
    #if nAR[per(t+1)] > 0 #remover esse condicional???
        @addConstraint(m, FCFcstr[l in 1:n.Openings, c in 1:Cuts[t,s]   , s in 1:n.Scenarios ],
        Alpha[l] >= delta[t,s,c] + sum{ piH[t,s,c,h]*Vf[h] , h in 1:n.Hydros} +
        sum{ piA[t,s,c,h,1]*a_next[h,l] , h in 1:n.Hydros; nAR[h,per(t+1)]>0  } +
        sum{ piA[t,s,c,h,p]*a_tmp[p-1,h], h in 1:n.Hydros, p in 2:nAR[h,per(t+1)]}
        )
        @addConstraint(m, InflowModel[ h in 1:n.Hydros, l in 1:n.Openings],
        a_next[h,l] == sum{phi[h,per(t+1),p]*a_tmp[p,h], p in 1:nAR[h,per(t+1)]}  + err[t+1,s,h,l]
        )
end


@setObjective(m, :Min,
sum{ cT[t,s,j]*gT[b,j], j in 1:n.Thermals, b in 1:n.Blocks}
    + (1/(1+tx_per))*(1/n.Openings)*sum{Alpha[l], l in 1:n.Openings}
```

PSR

The beginning

- Optfolio: Portfolio optimization
  - Initially JuMP+CBC, first motivation for Xpress.jl
- Graduation project
  - SDDP Notebook
  - Multi stage Nash equilibria
- In parallel:
  - Monte Carlo methods for reliability analysis (Luiza & Camila)
  - Optimal power flow (with help from Lujan, Lucas & Igor)

Research

- Optgen (SDDiP)
- Reliability analysis
  - Graduation project
- Optflow
  - **Progressive Hedging Applied to the Problem of Expansion Planning of Reactive Power Support Equipment** (online but in Portuguese)
  - Conic Models -> Decomposition
- Immediate cost function
  - **Analytical representation of immediate cost functions in SDDP** (online)
  - In fortran
  - Pre computation of hyperplane representation (https://github.com/JuliaPolyhedra/Polyhedra.jl)
- ICSP:
  - **Representation of uncertainties in fuel cost and load growth in SDDP-based hydrothermal scheduling** (online)
  - Nash equilibria (online)

What we have been doing in julia & JuMP

- SDDP + Optgen
  - PERU energy ministry
- Portfolio Optimization (TNC)
  - Optfolio + PreOptgen (as auxiliary routine of HERA)
  - Magdalena Basin in Colombia
  - https://global.nature.org/content/power-of-rivers?src=r.v_powerofrivers

A Better Way to Harness the Power of Rivers

How system-scale planning and management of hydropower can yield economic, financial and environmental benefits

What we have been doing in julia & JuMP

- SDDP + Optgen
  - PERU energy ministry
- Portfolio Optimization (TNC)
  - Optfolio + PreOptgen (as auxiliary routine of HERA)
- Evaluating Latin American Interconnections
  - For BID (Inter-American Development Bank)
- Refactoring Genesys from NWPCC
  - Large scale reliability evaluation model

SDDP – Power system operation modeling

- Physical parameters
  - Hydro (detailed topology (cascades), hydro production, reservoirs modeling, operative constraints etc.)
  - Thermal (efficiency curves, combined cycle plants, multiple fuel plants, fuel availability constraints, GHG emission factors, unit commitment decisions etc.)
  - Renewables (Wind, biomass, solar etc. represented scenarios)
  - Transmission Network (Linearized power flow model with quadratic losses, security constraints etc.)
- Stochastic parameters
  - Hydro inflows and renewable generation - Multivariate stochastic model
  - Uncertainty on fuel costs - Markov chains (hybrid SDDP/SDP model)
  - Wholesale energy market prices - Markov chains
  - Generation & transmission equipment outages - Monte Carlo

SDDP – Power system operation modeling

- Multi-stage: time coupling due to energy reservoirs
- Stochastic: multiple parameters are uncertain
- Solving the deterministic equivalent LP is not feasible
  - Too many scenarios and stages: the scenario tree grow too fast
- SDDP stands for Stochastic Dual Dynamic Programming, an algorithm developed by Mario Pereira (PSR founder and president)
  - ICSP: 5 sessions and 22 talks
  - julia
    - https://github.com/odow/SDDP.jl
    - https://github.com/blegat/StructDualDynProg.jl
    - https://github.com/JuliaOpt/StochDynamicProgramming.jl
    - Many others...

Multi-stage economic dispatch and SDDP

- Classical dispatch problem:
  - Objective $\min \sum c_j g_j + \beta(v_{t+1}, a^s_{t+1})$
  - Water Balance $v_{t+1} = v_t + a_t - u - s$
  - Load Balance $\sum g_j + \sum \rho_i u_i = d - \sum r_k$
  - AR model $a^s_{t+1} = \phi_1 a_t + \phi_2 a_{t-1} + \xi^s_{t+1}$
- And much more...
- Complex maintenance and hard to experiment and add features

SDDP characteristics

Iterative procedure

1. forward simulation: finds new states and provides upper bound
2. backward recursion: updates FCFs and provides lower bound
3. convergence check (LB in UB confidence interval)

Distributed processing

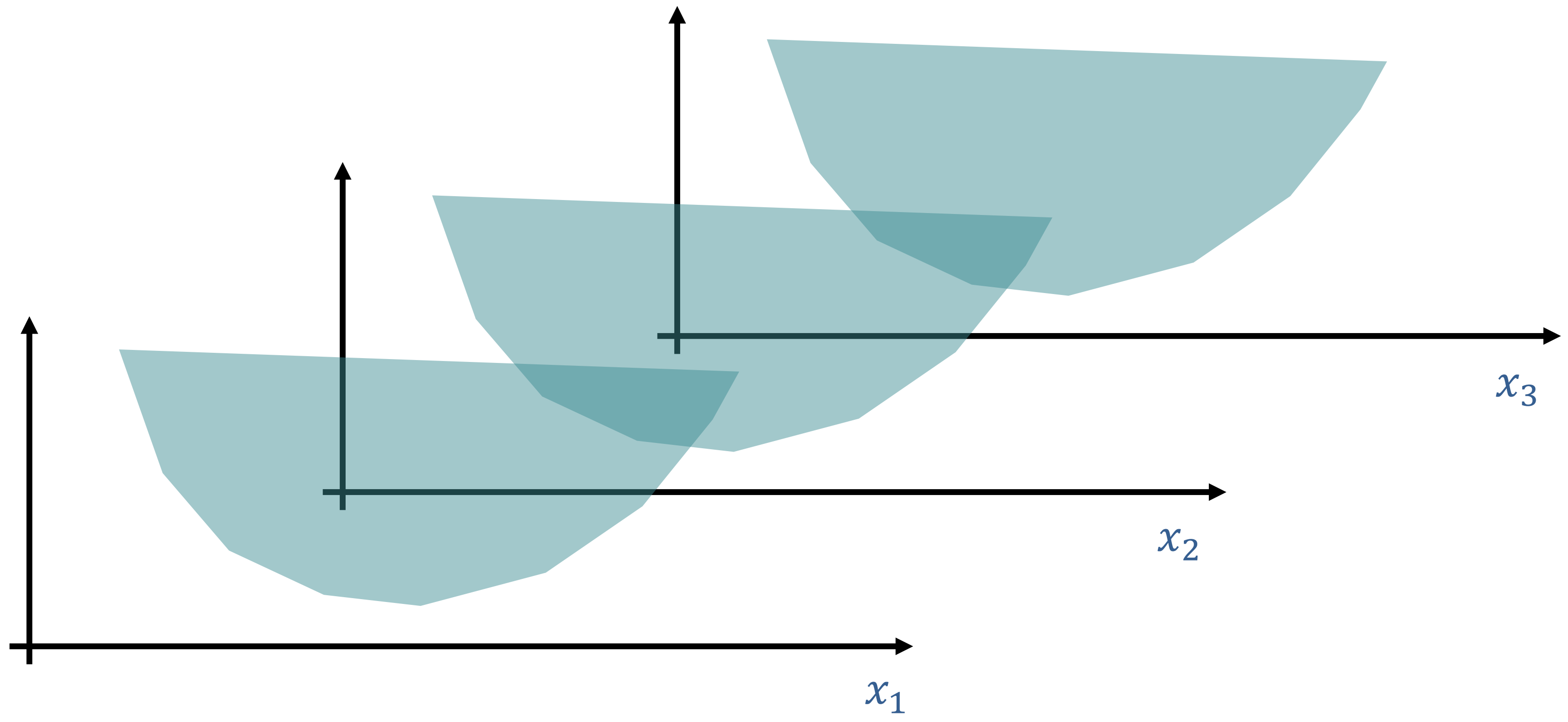
- The one-stage subproblems in both forward and backward steps can be solved simultaneously, which allows the application of distributed processing
- SDDP has been running on computer networks since 2001; from 2006, in a cloud system with AWS

SDDP – Applications

- Valuation studies for new projects (hydro, thermal and renewables) and transmission expansion
- Assessment of regional markets and international interconnections
- System price forecast
- Mid and long term production planning
- Regulatory studies and market design simulations
- Water resources analysis (eg. hydropower vs. irrigation) and evaluation of different hydro operative rules
- Natural gas x electricity coordination

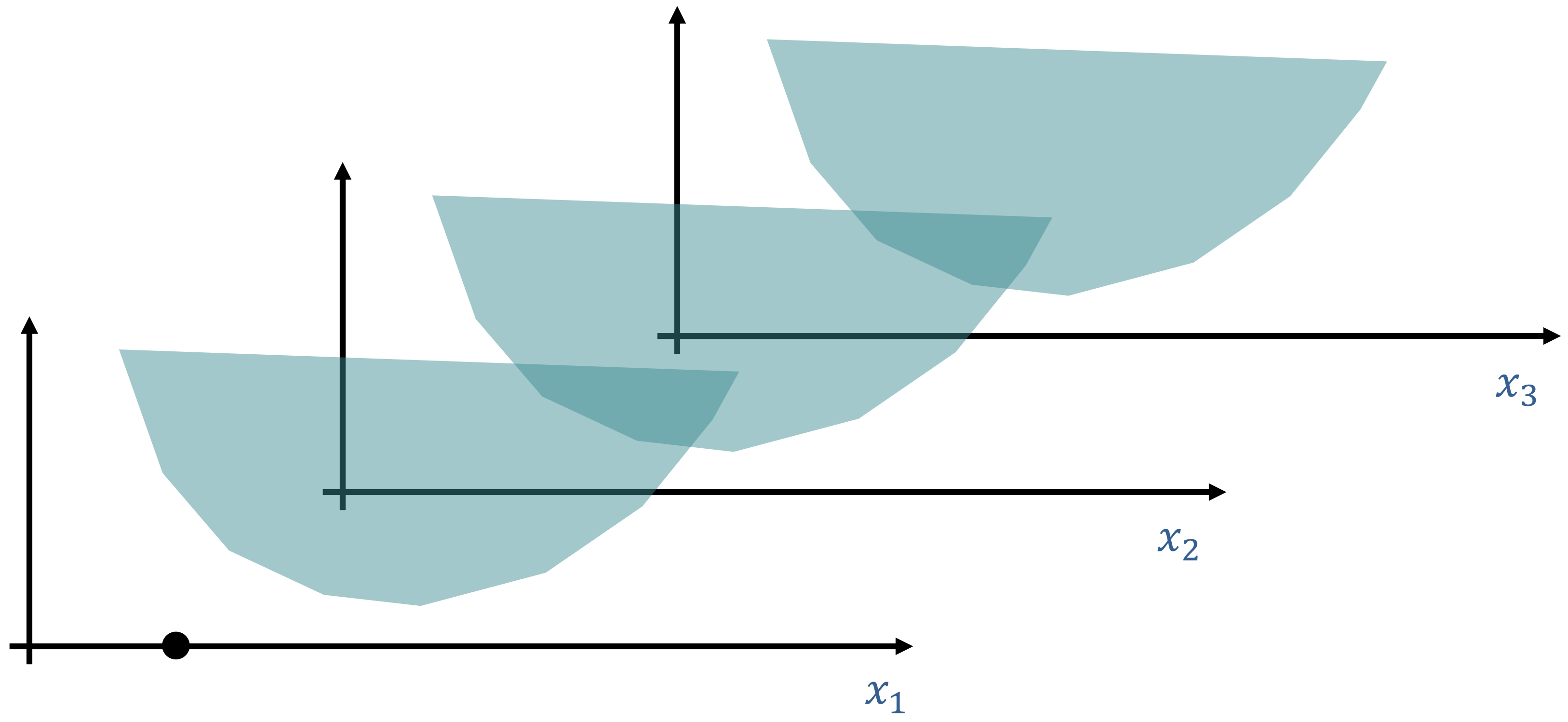
SDDP

Future cost function (cost-to-go)



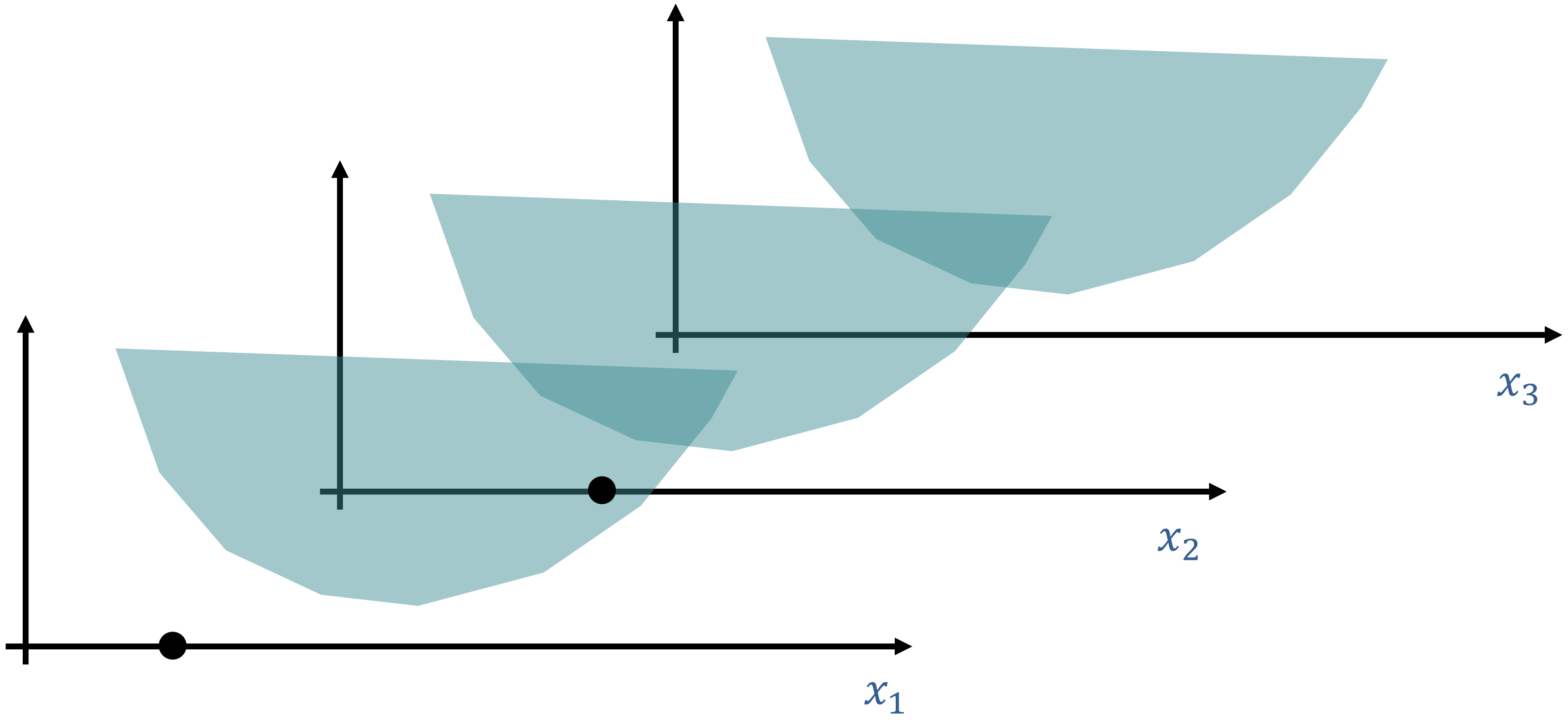
SDDP

Forward iteration 1



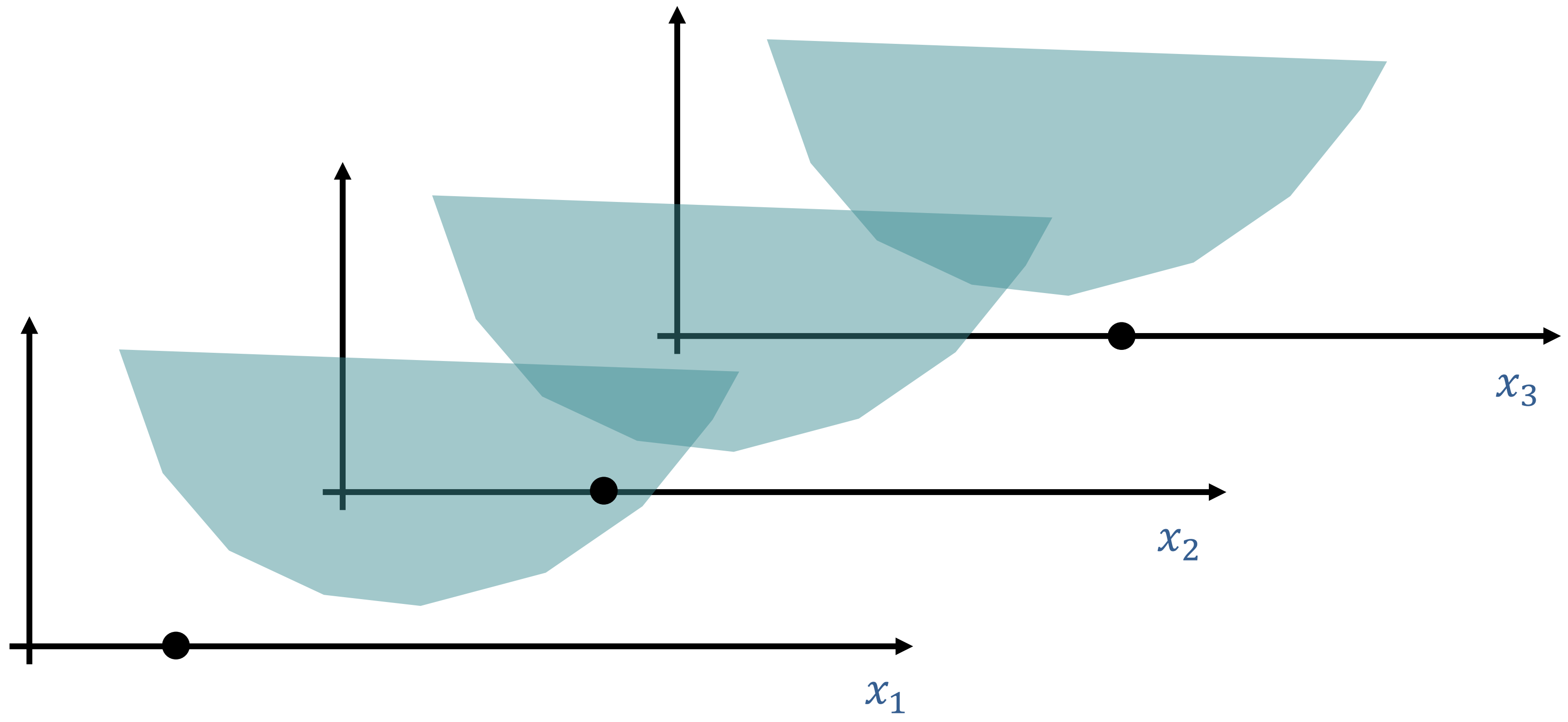
SDDP

Forward iteration 1



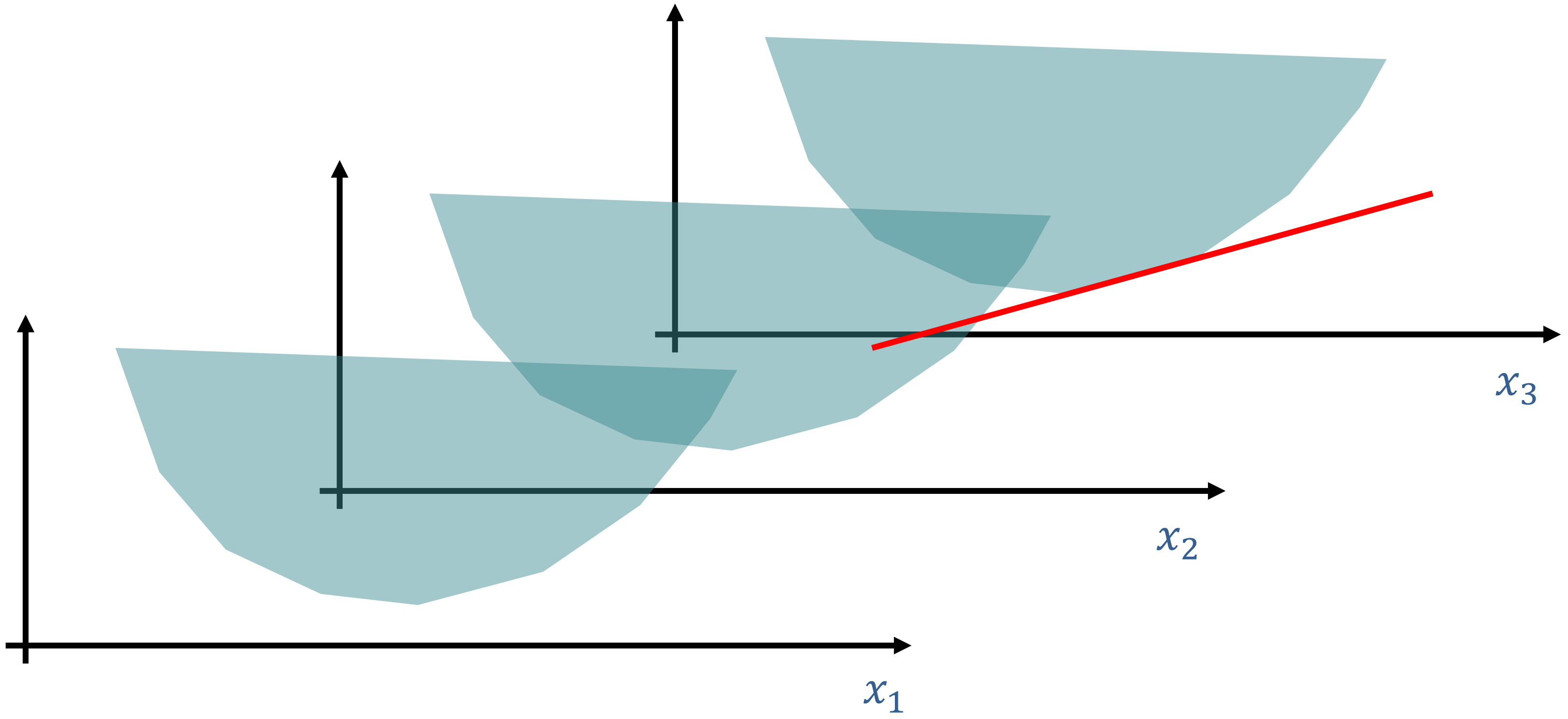
SDDP

Forward iteration 1



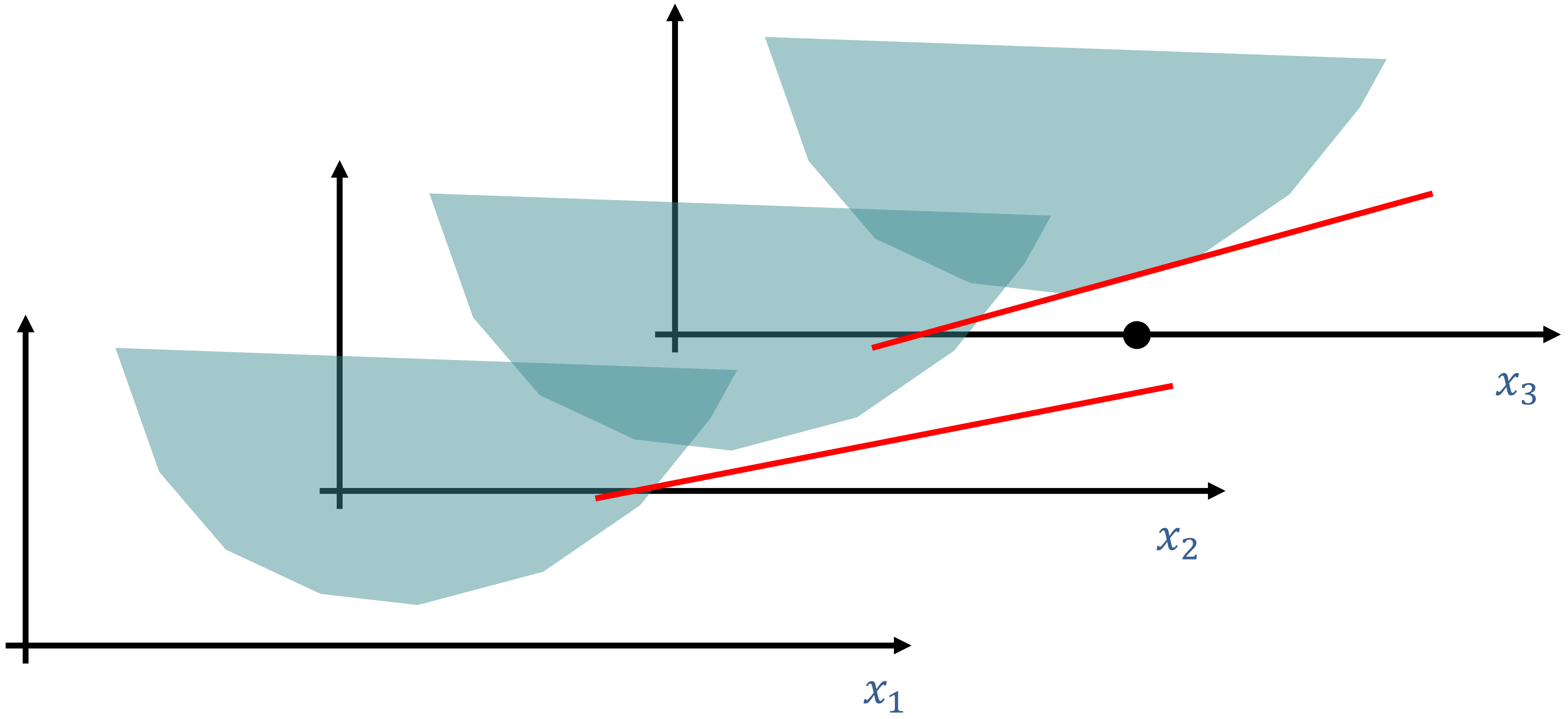
SDDP

Backward iteration 1



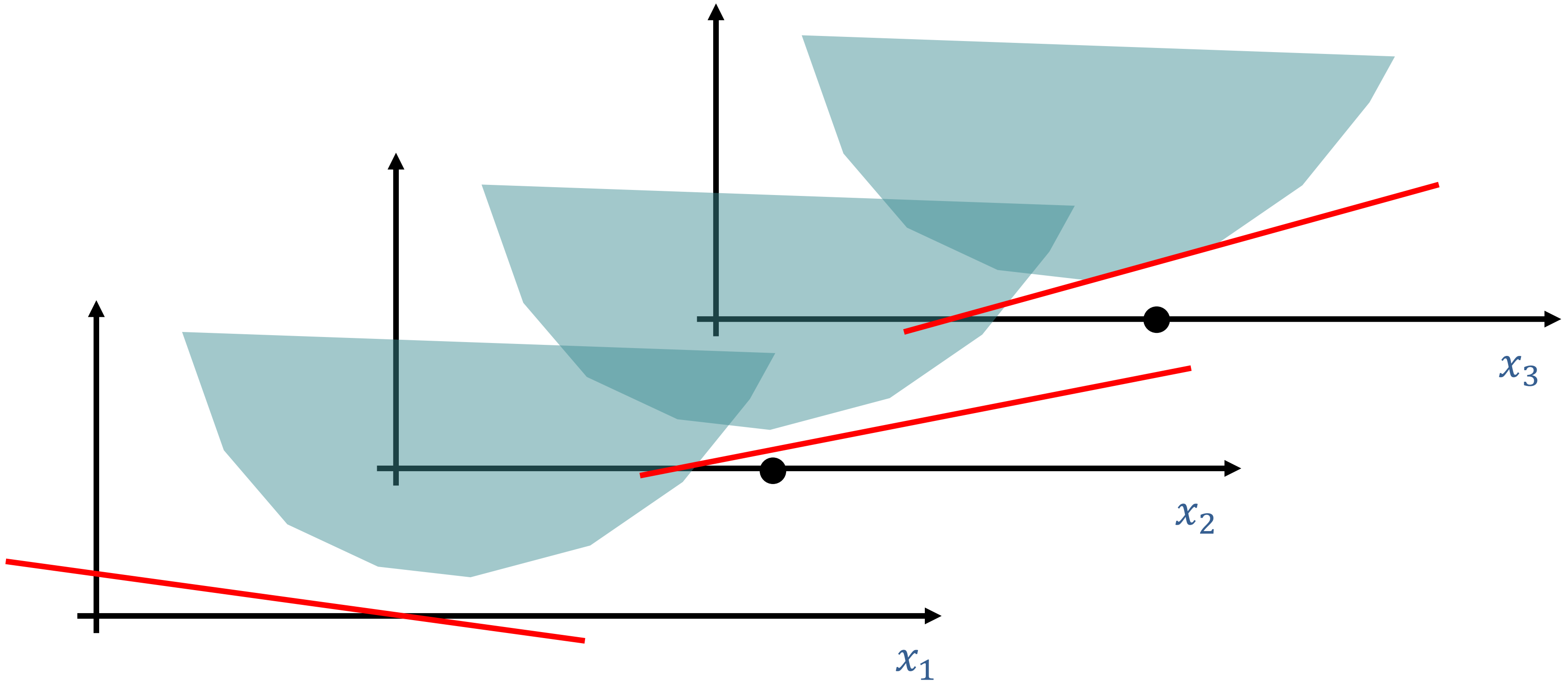
SDDP

Backward iteration 1



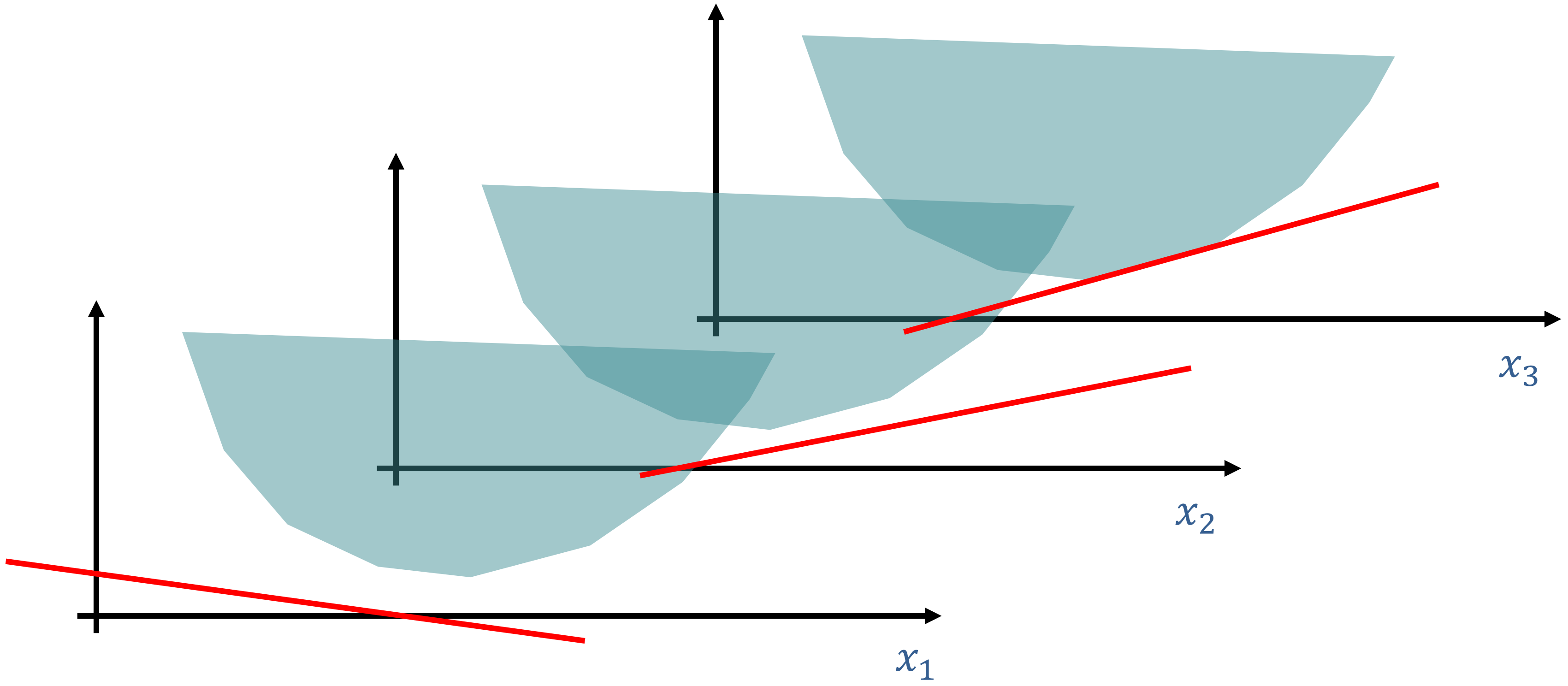
SDDP

Backward iteration 1



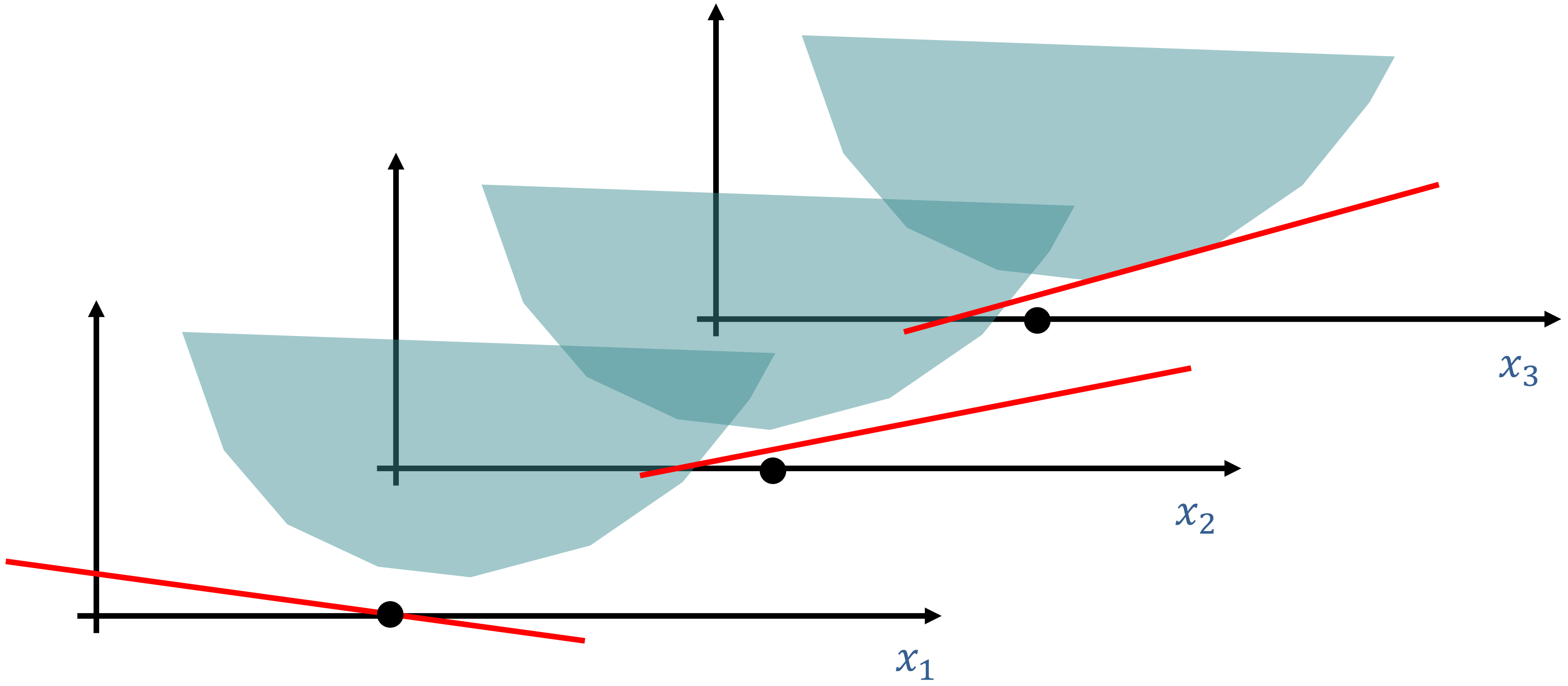
SDDP

Backward iteration 1



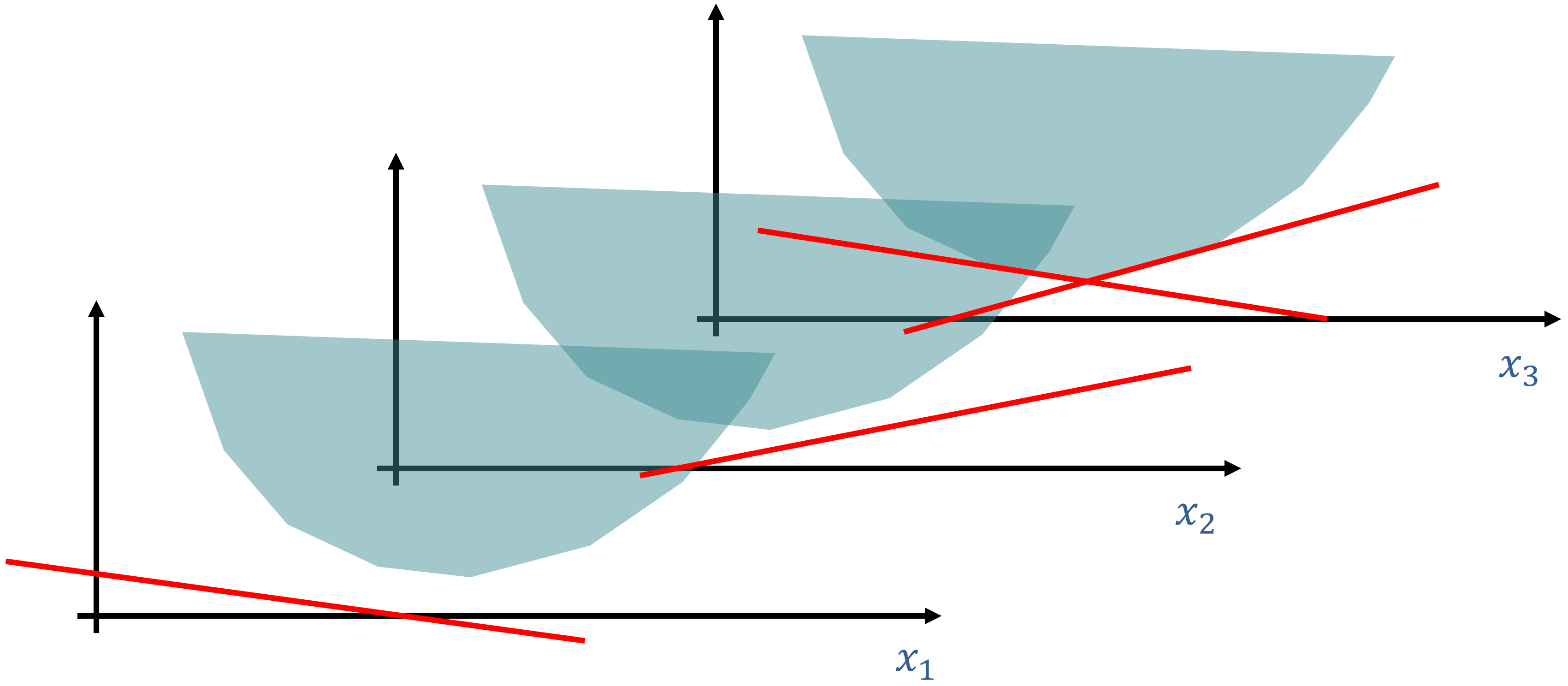
SDDP

Forward iteration 2



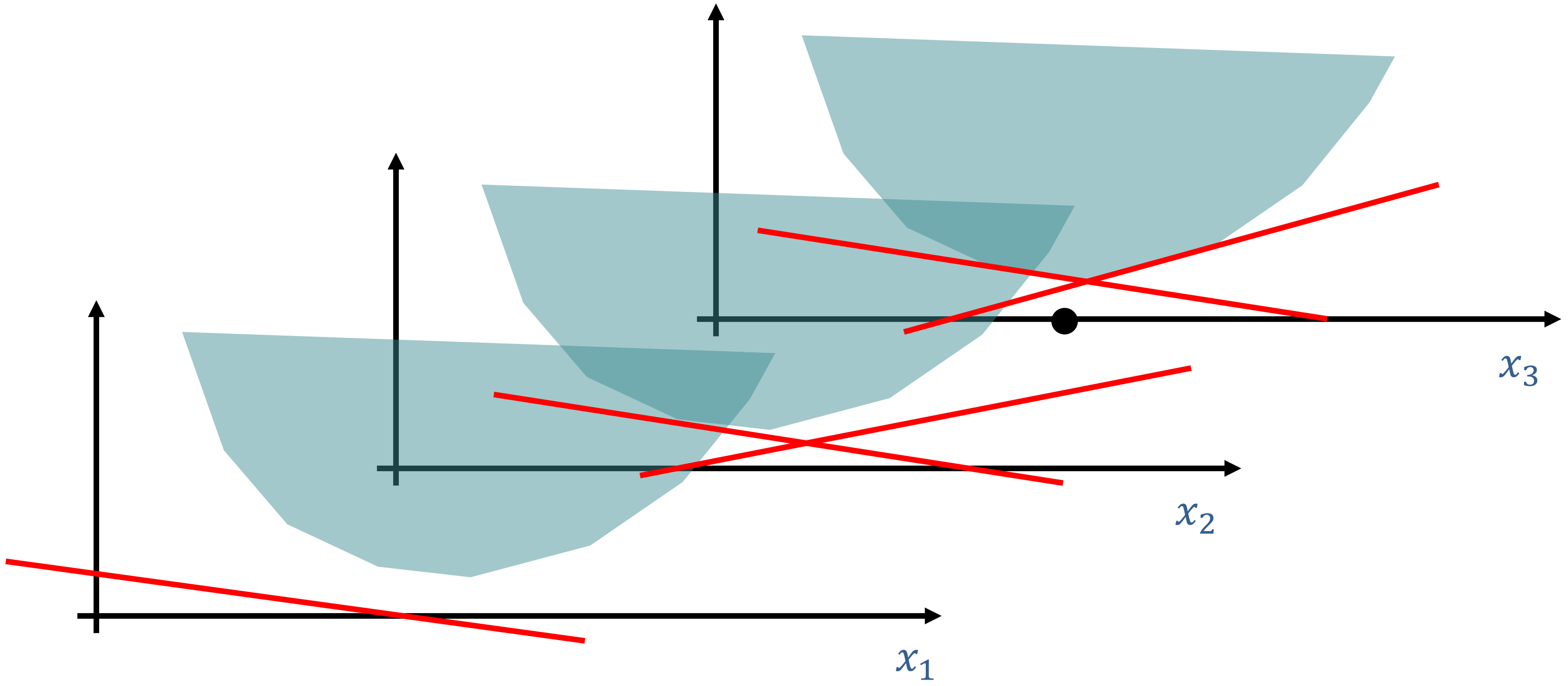
SDDP

Backward iteration 2



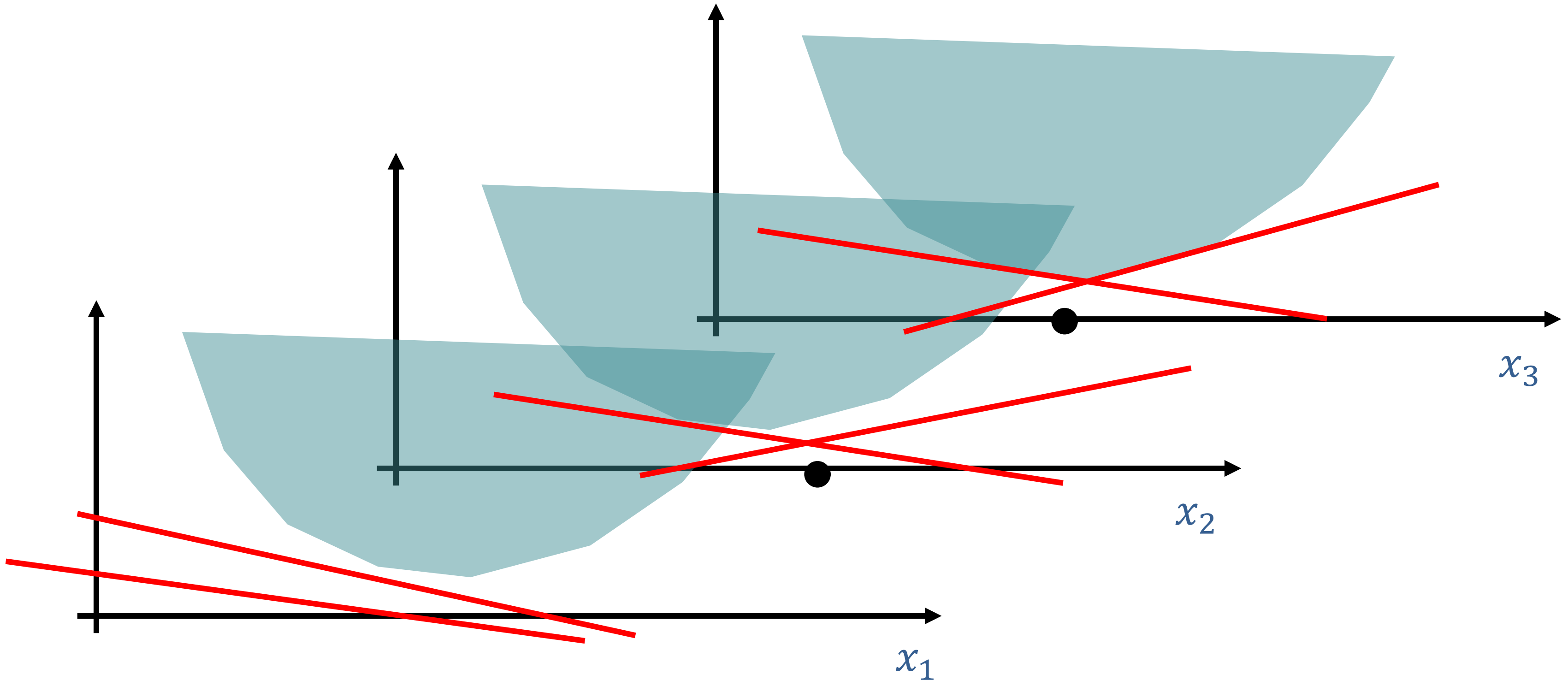
SDDP

Backward iteration 2

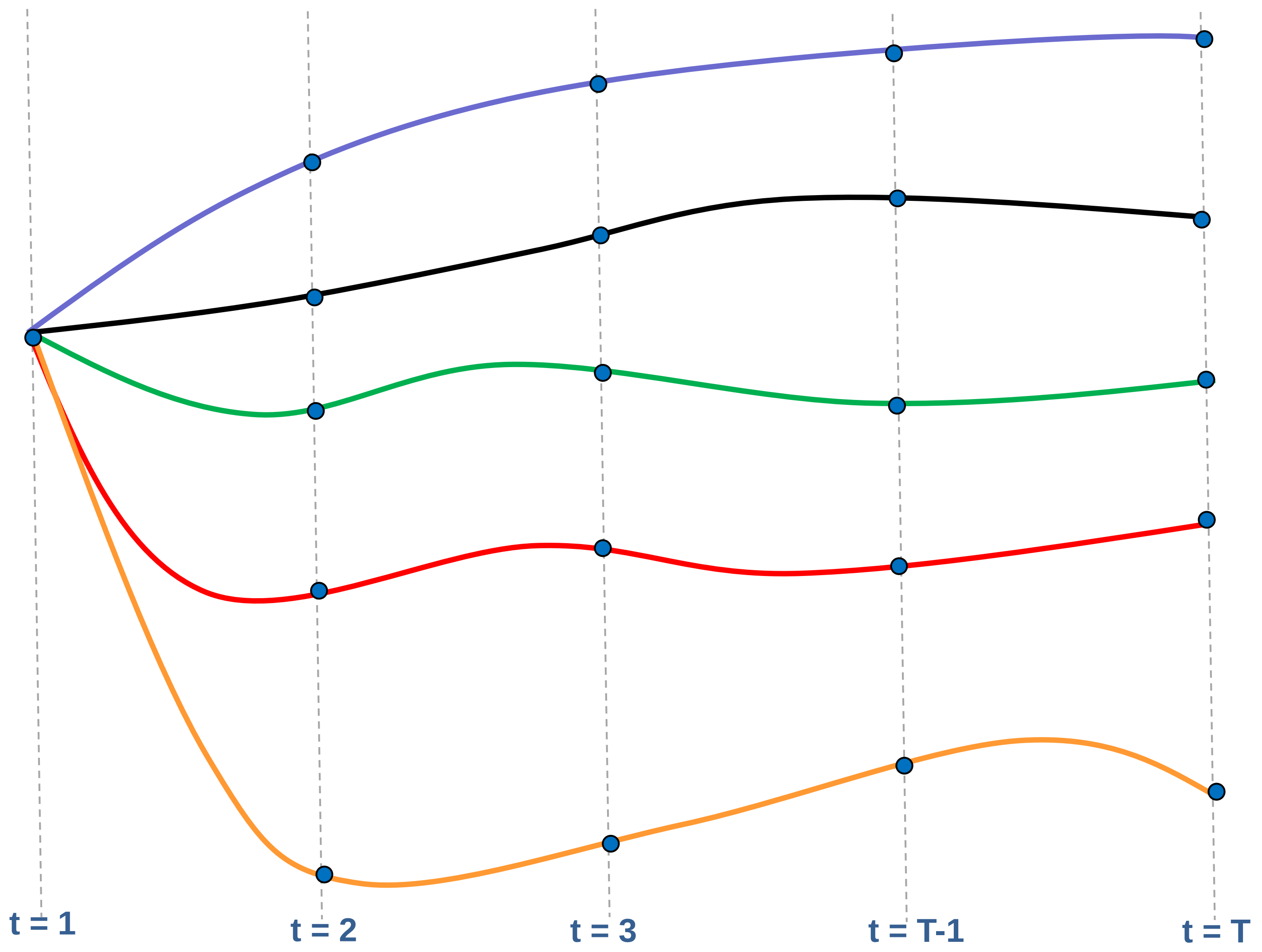


SDDP

Backward iteration 2



SDDP: distributed processing



SDDP – Implementation

- Why Julia & JuMP
  - Easy to prototype, test new features
  - Good performance
  - Miles: “I want to model and solve LP/MIP within a programming language, but python is to slow and C++ is too low level”
  - Miles:”I want to implement optimization algorithms in a fast, high-level language designed for numerical computing”
- EX: Central America (2500 vars, 25000 cons, 60 stages, 100 scenarios, 5 iterations. More than 200 build problems, 1.000.000 solved in 20min)
- Very good results: more than 70% of time in (Xpress) solver!
  - Thanks to TimerOutputs
- Trivially working in clusters on amazon with MPI

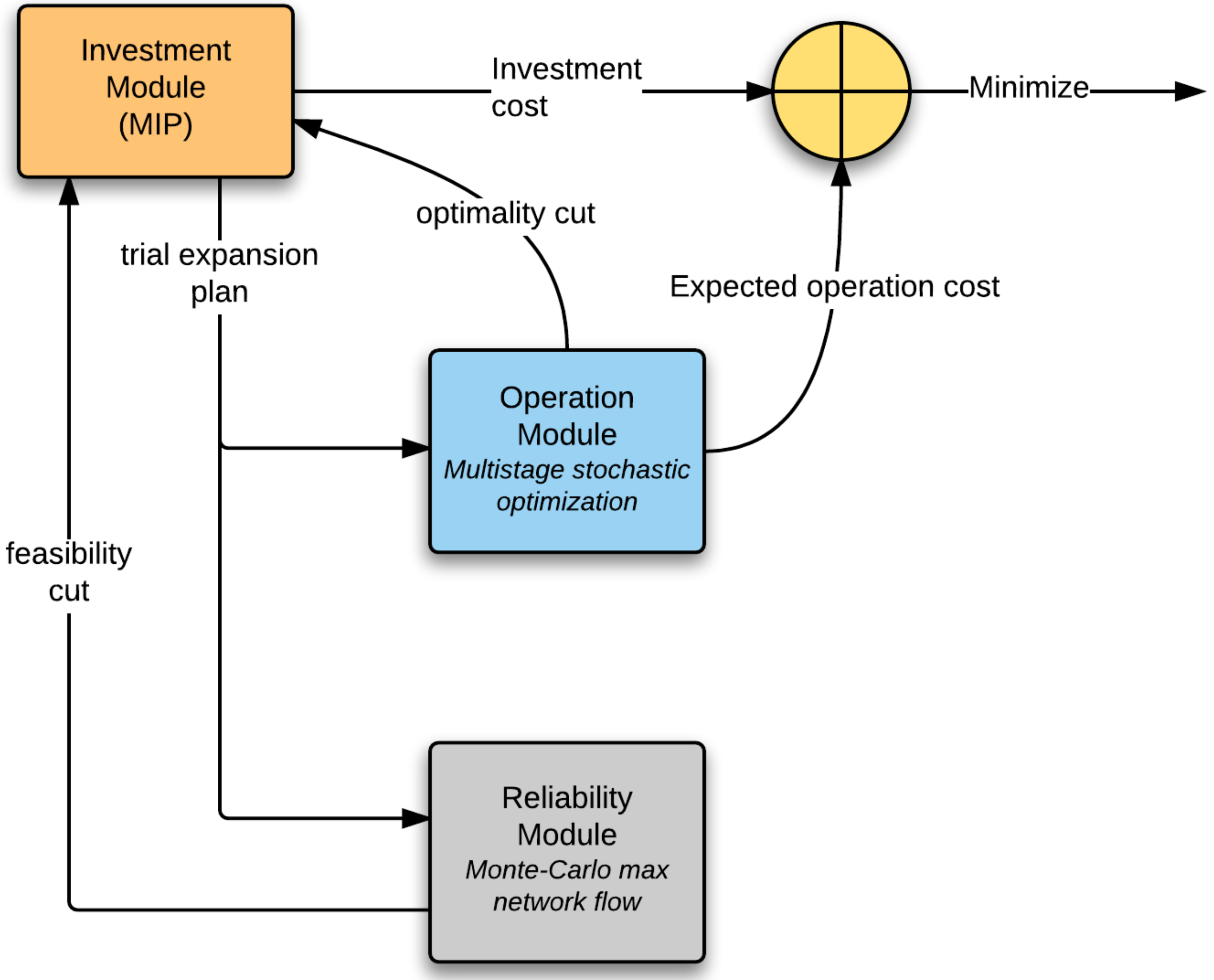
SDDP – Implementation

- Moreover
  - Miles: “Make it easy to access low-level features. Don’t get in the user’s way”
- A big problem: cut relaxation (selection)
  - Adding cuts as extra julia constraints: too slow
    - Use MPB lower level
- Rewriting many problems: (small problem: build in 4.5ms, solve in 0.4ms)
  - Too expensive to write problems for each scenario
  - Build once per stage!
  - What about cuts? Well make copies of a model, not so good
  - Remove constraints (also variables, changecoeffs, fixvalues)
- Vary parameters: tolerances etc
- More: fixglobals, inplace getters (getsolutions 6%)

OptGen – Overview

- Optimal mid and long term capacity expansion planning
- Horizon: monthly/weekly stages; up to several decades
- Detailed representation of hydro, thermal, renewable plants' production and investments
- FORTRAN coded
- Mosel: subproblem

Integrated G&T Planning (SDDP – OptGen integration)



OptGen – Applications

- Used by the World Bank globally and by utilities, market operators, regulators and investors in the several studies:
- Generation expansion planning studies including regional interconnections links with Central America systems (Panamá, Costa Rica, Nicaragua, Honduras, El Salvador e Guatemala) for the horizon 2009-2023
- Re-evaluation of the generation-transmission integrated studies with Bolivian system (10 years horizon period, 2009-2018)
- Expansion studies of Egypt-Sudan-Ethiopian interconnection system to provide an economic evaluation to justify the expansion of the interconnection and the construction of large reservoirs
- Studies for the economic evaluation for the construction of the second transmission line connecting all six Central American countries (Panamá, Costa Rica, Nicaragua, Honduras, El Salvador and Guatemala)
- Evaluation of the generation expansion planning for the Dominican Republic, horizon of 10 years (2007-2016)

OptGen – Julia

- Completely new formulation focusing on energy reserve:
  - Investments are now compared to their required reserve
  - Approach completely adapted to renewable generation and battery projects
  - Test different decomposition approaches
- Already used in real studies:
  - Saudi Arabia
    - 100% thermal system wanting to invest in renewable sources, many minimum generation commitment constraints
  - Colombia
  - Chile

OptGen – Julia

- Yearly subproblems
  - MILP solved via forward or backward single iterations heuristic
- JuMP HEAVILY used
  - Detailed information such as renewables, reserves and commitment
- Motivated work on Xpress.jl
  - IIS
  - Callbacks
- Easily converted in decomposition technique SDDiP
  - Generating cuts and solving problem in parallel (MPI) in large scale (AWS)
- Orders of magnitude faster for some classes of problems

OptFlow – Julia

- Originally focused on non-linear optimal power flow for reactive expansion & investment
- Taylor made IPM
  - Extremely fast
  - Hard to add new constraints and prototype new ideas
- New version: based in JuMP desigend to accomodate non-linear power flow and its convex relaxtaions
  - Ready to generate valid cuts for decompostion algorithms
  - Easy implementation of progressive hedging

OptFlow – Julia

- Similar to PowerModels.jl…
- Abstracts on constraints classes:
  - Non-linear rectangular
  - Non-linear Polar
  - Convex SOCP
  - Convex SDP (complex and real)

OptFlow – Julia

- Some testing results (12600 problems solved sequentially)

TCPU(s)	NL Polar (Ipopt)	NL Retangular (Ipopt)	SOCP (Xpress)
Input	30.13	37.17	93.95
Write Model	19.80	22.66	136.44
Solver	1568.99	4340.25	1378.83
Output	16.38	18.39	46.22
Total	1635.31	4418.47	1655.44

- FORTRAN: 162s

The end

- Thanks!