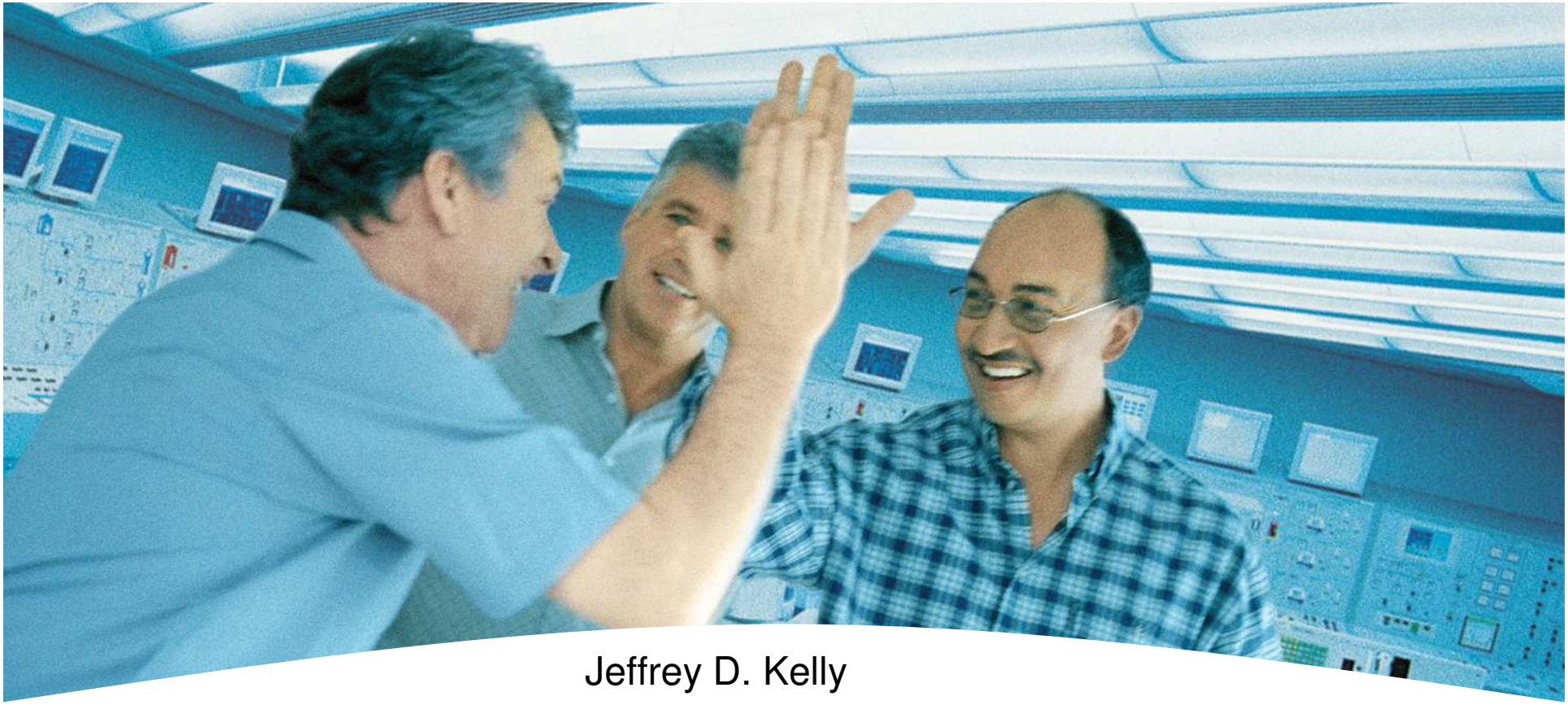


Jeffrey D. Kelly
Fields Institute - Industrial Optimization Seminar
February 2, 2010

On the Modeling and Solving of Large-Scale Manufacturing Optimization Problems (MOP's) in Diverse Industries

Honeywell

Summary of Talk

- Where, Why & What is the Manufacturing Optimization Problem (MOP)?
- How do we Model the MOP Details?
- How do we Solve the MOP Description?
- How do we Integrate the MOP Data?
- How do we Execute the MOP Decisions?, and
- How do we Improve the MOP Deployment?

Where is the MOP?

- Manufacturing found in **Continuous**, **Batch**, **Dimensional** (1D, 2D, 3D shapes) & **Discrete** Industries always involve the following details:
  - Complexity (Spatial, Temporal, Phenomenological, Technological, etc.),
  - Hierarchy (Design, Planning, Scheduling, Control, Analysis, Accounting, etc.),
  - Uncertainty (Accuracy, Ambiguity, Availability, Abstraction, Reliability, etc.).
- Manufacturing also involves the many requirements of **Production**, **Processes**, **Plants**, **Projects**, **Products**, **Policies** & of course **People**.
- Our focus is on **Complexity Management** integrated within an established Hierarchy subject to omnipresent Uncertainty (of which we know little about).

Why is the MOP Important?

- To find **better** solutions to problems **faster** & **cheaper** automatically versus manually using optimization (i.e., a systematic search).

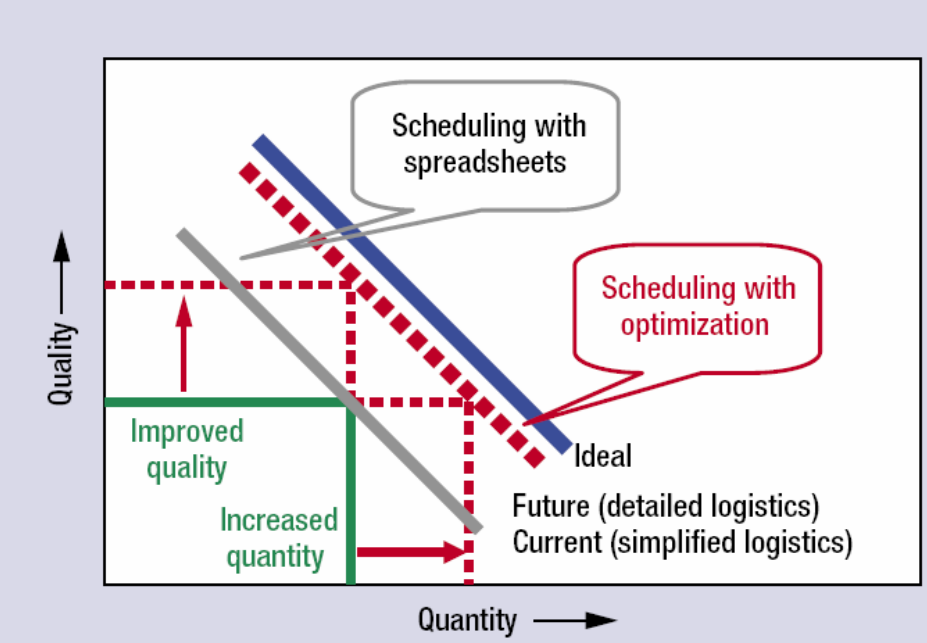


FIG. 2 Quantity-quality trade-off curve with logistics isotherms.

Kelly, J.D., "Logistics: The Missing Link in Blend Scheduling Optimization", *Hydrocarbon Processing*, June, **2006**.

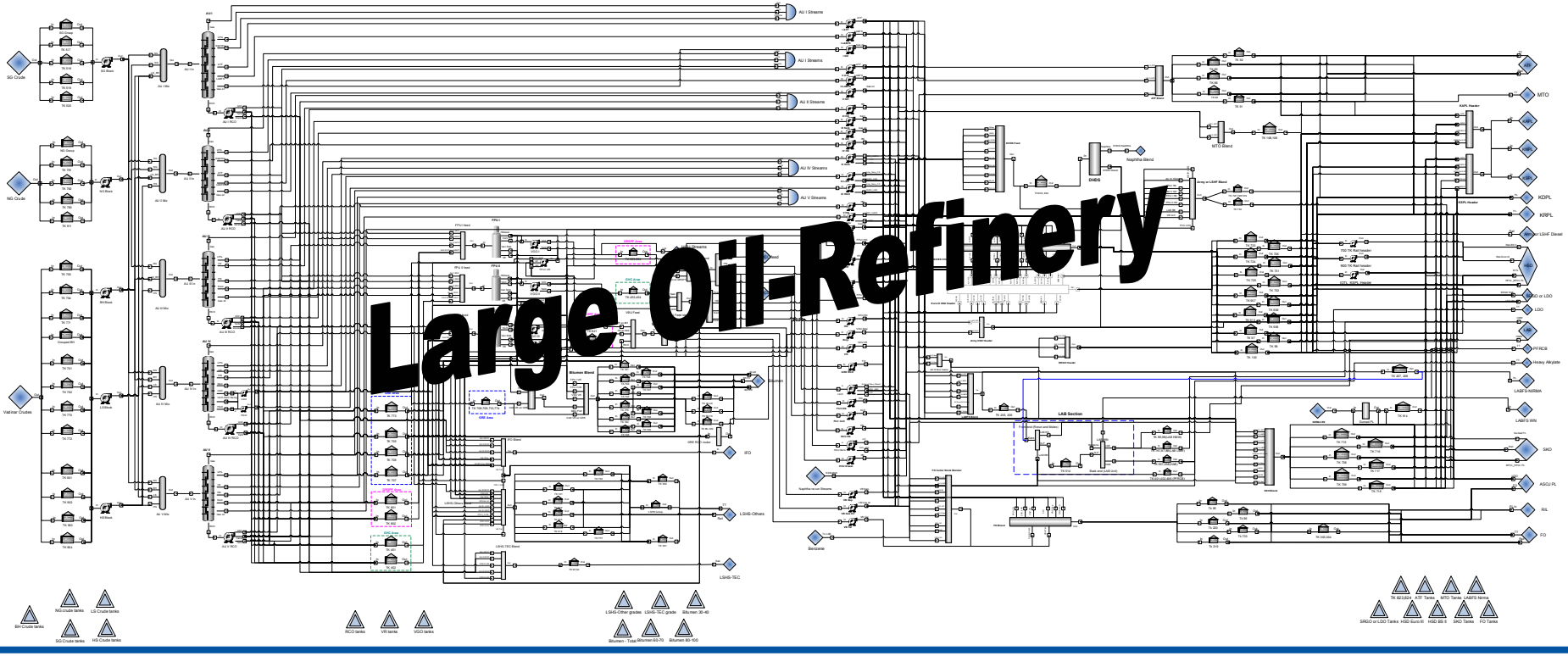
Why is the MOP Important?

- More specifically, why is optimization desired over simulation? - there are three main reasons:
  - Specificity = automatic identification of non-basic, basic & super-basic variables,
  - Feasibility = solutions are better than with user/modeler defined specificity,
  - Optimality = solutions are better than just satisfying feasibility.
- Simulation requires the modeler to apriori or "heuristically" determine the variable specifications to find "good" solutions where many specification scenarios, situations or settings are necessary requiring "lots" of effort both at modeling-time & solving-time, for example:
  - If we have the "implicit/open" tank material balance: $Hopen + Fin - Hclose - Fout = 0$
  - Then we need 4 "explicit/closed" balance constraints to be modeled in the simulation system depending on the specification of each variable at solve-time:
    - Hopen = -Fin + Hclose + Fout
    - Fin = -Hopen + Hclose + Fout
    - Hclose = Hopen + Fin – Fout
    - Fout = Hopen + Fin – Hclose;

    where the LHS variable is dependent/basic & the RHS variables are independent/non-/super-basic i.e., a "square" problem with zero DoF.

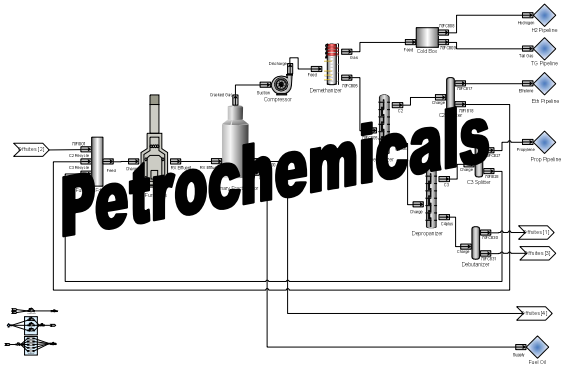
What is the MOP?

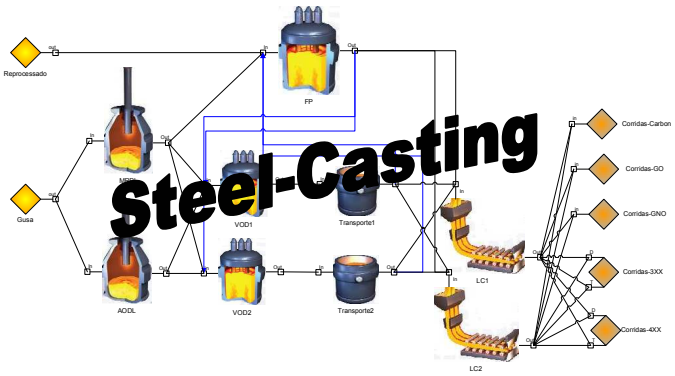
- MOP's typically involve large networks or "superstructures" similar to a decision-tree or map that represents the **capability** of the system.
- **Capability** is the combination of configuration, connectivity, capacity, compatibility, constituency, consolidation, commandability & cost.

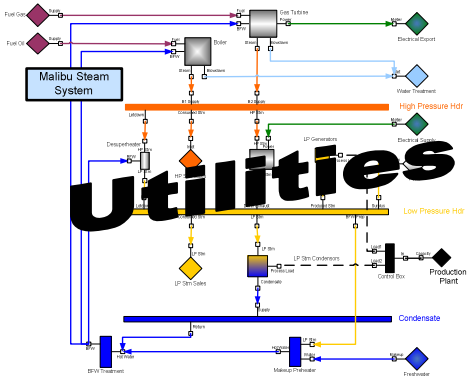


What is the MOP? (cont'd)

- MOP's may include only a portion of the overall problem such as **key bottlenecks** in the manufacturing facility, enterprise or supply-chain.
- MOP's involve to some degree or another the following **situations**:
  - Complicated environmental, governmental & international regulations,
  - Scarce & expensive resources of decreasing quality with higher demands on less costly finished goods,
  - Aggressive & sustained production at or above design or safe operating limits,
  - Tighter delivery schedules & shorter lead-times transporting over longer distances,
  - More advanced processing technologies increasingly located in harsher environments & politically unstable geographical regions.







How do we Model the MOP?

- Our approach is to describe the manufacturing using **spatial objects** of "units", "operations", "ports" & "states", **phenomenological attributes** of "quantity", "logic" & "quality" & **temporal incidences** of "time-periods" & "time-points".

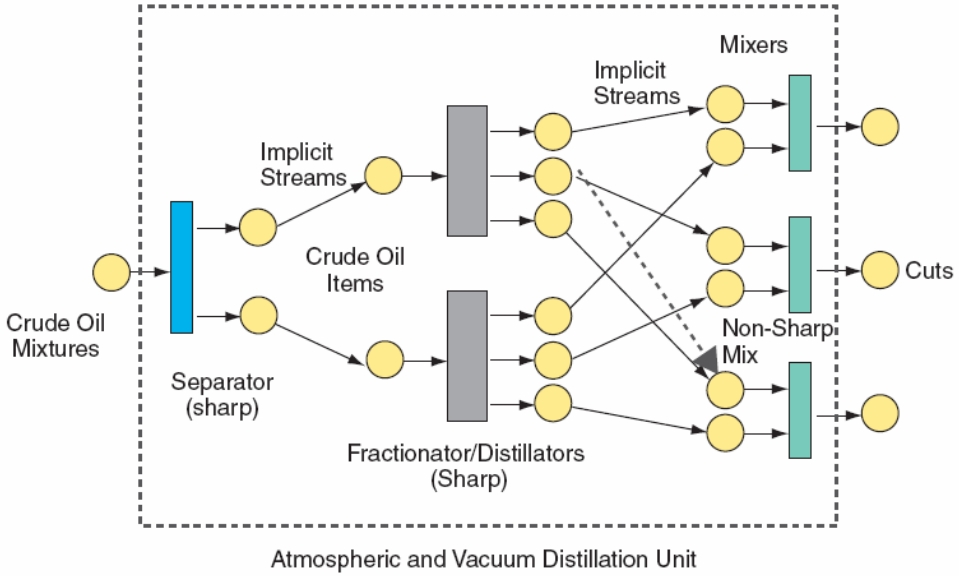


Figure 2. Anatomy of an oil-refinery atmospheric or vacuum distillation unit model.

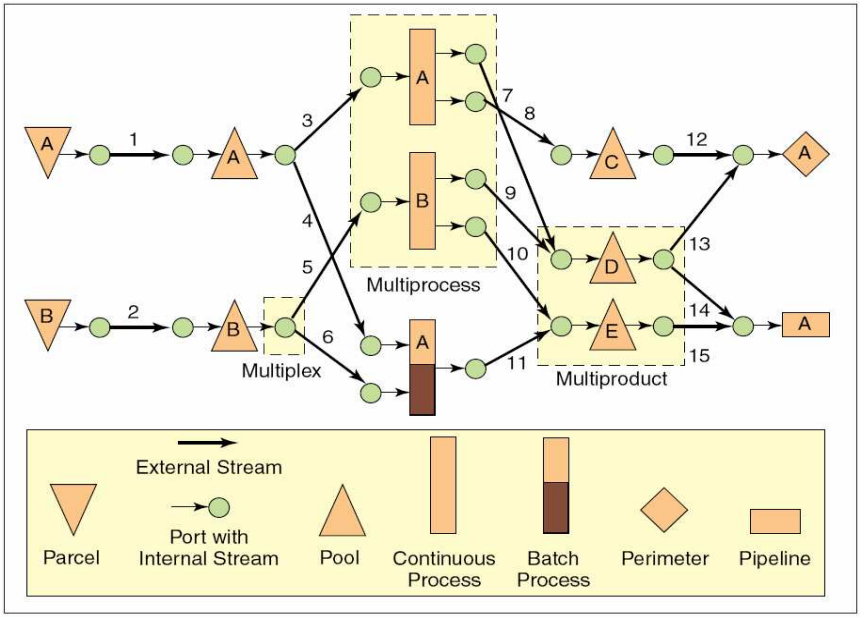


Figure 1. Production flow network for the small plant example.

Kelly, J.D., "Formulating Production Planning Models", *Chemical Engineering Progress*, January, **2004**.

Kelly, J.D., "Production Modeling for Multimodal Operations", *Chemical Engineering Progress*, March, **2004**.

How do we Model the MOP? (cont'd)

- Sets of **spatial objects** or "pointers" usually include:
  - Units = { Equipment, Vessels, Machines, etc. }
  - Operations = { Tasks, Jobs, Activities, etc. }
  - Ports = { Nozzles, Stubs, Connectors, etc. }
  - States = { Materials, Parts, Signals, Time, etc. }

Pole(P,o,i,n,t,e,r,s) = Spatial, Phenomenological, Temporal **Profiles**

- Sets of **phenomenological attributes** or "poles" usually include:
  - Quantity = { Flows, Rates, Holdups, Yields, Durations (Quanta) }
  - Logic = { Setups, Startups, Switchovers, Shutdowns, Statuses }
  - Quality = { Densities, Components, Properties, Conditions }
- Other spatial objects are created as set relations/cross-products/tables of Units x Operations, Unit-Operations x Port-States, etc. & these are called **projectional** objects as they are the product of **physical** (unit-port) and **procedural** (operation-state) objects & form either an "open-shop" or "closed-shop" superstructure.

Zyngier, D., Kelly, J.D., "Multiproduct Inventory Logistics Modeling in the Process Industries", Chapter 2, W. Chaovalitwongse et al. (eds.), *Optimization and Logistics Challenges in the Enterprise*, Springer, **2009**.

How do we Model the MOP? (cont'd)

- Specifically, what is "logic"?
- Logic variables & constraints manage the "discrete" & "disjunctive" (non-continuous) nature of manufacturing.

$$y_{i,t} - y_{i,t-1} - su_{i,t} + sd_{i,t} = 0 \quad \forall\, i = 1...NO \text{ and } t = 1...NT \tag{3}$$

$$y_{i,t} + y_{i,t-1} - su_{i,t} - sd_{i,t} - 2sw_{i,i,t} = 0 \quad \forall\, i = 1...NO \text{ and } t = 1...NT \tag{4}$$

$$su_{i,t} + sd_{i,t} + sw_{i,i,t} \le 1 \quad \forall\, i = 1...NO \text{ and } t = 1...NT \tag{5}$$

FSVT = fixed-size, variable-time
y = setup
su = startup
sd = shutdown
sw = switchover

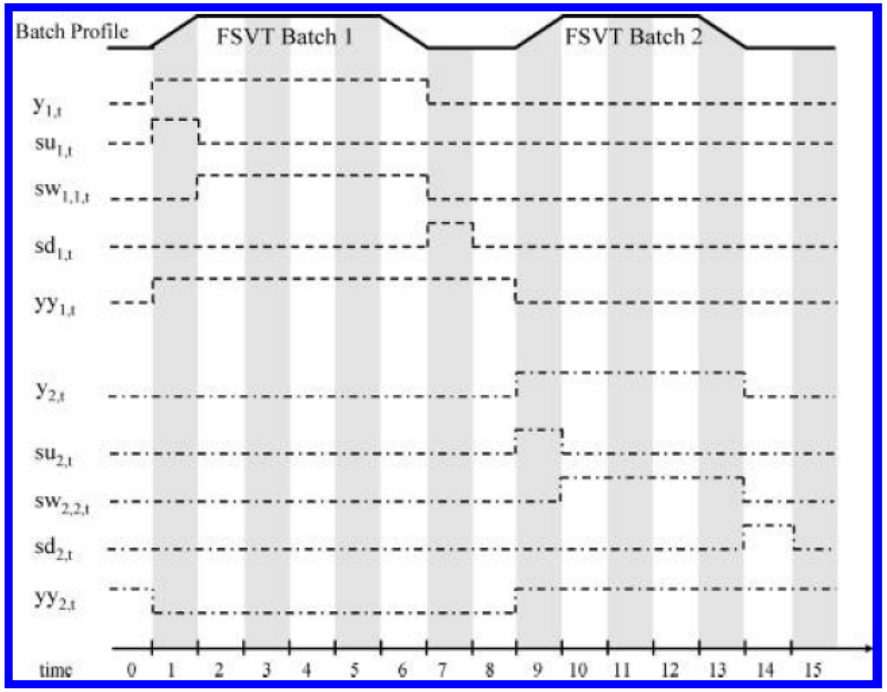


Figure 1. Relationship between $y_{i,t}$, $su_{i,t}$, $sw_{i,i,t}$, $sd_{i,t}$, and $yy_{i,t}$ for Batch-1 (—) and Batch-2 (—•—).

Kelly, J.D., Zyngier, D., "An Improved MILP Modeling of Sequence-Dependent Switchovers for Discrete-Time Scheduling Problems", *Industrial Engineering Chemistry Research*, 46, **2007**.

How do we Model the MOP? (cont'd)

- More than just algebraic & arrayic – it is an "allegory" of how manufacturing works as a whole using "arrayic-algebraic" model forms.

"Allegoric"-Form

(New School)

A "figurative" language.

Simple Tabular & Traversal Views

"Arrayic-Algebraic"-Form

(Middle-School)

"Algorithmic"-Form

(Old School)

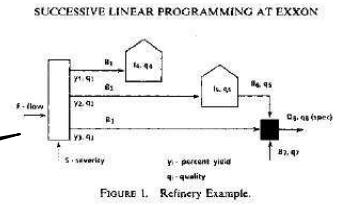


FIGURE 1. Refinery Example.

Refinery Example

The simple refinery example shown in Figure 1 illustrates the application of SLP to the nonlinear terms present in pooling and process yields. In this example, we assume that the yields, $y_i(S)$, and qualities, $q_i(S)$, of the process unit are known functions of severity, S. The severity of the process unit is a decision variable which is expanded about a base point, $\mathbf{S}$: $S = \mathbf{S} + \Delta S$. We adopt the notation,

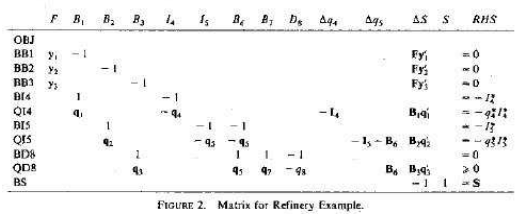
$$\mathbf{y}_i = y_i(\mathbf{S}), \qquad \mathbf{y}'_i = dy_i/dS|_{S=\mathbf{S}}, \qquad \mathbf{q}_i = q_i(\mathbf{S}), \qquad \mathbf{q}'_i = dq_i/dS|_{S=\mathbf{S}}.$$

Let B_i represent the other volumetric flows, I_i be final inventories, I_i^* be starting inventories, and q_i^* be starting qualities. The volumetric yields of the streams produced by the process unit can be expressed as Fy_i where F is the volumetric throughput of the unit and y_i is the percent yield of stream i. This nonlinear term can be expanded as: $Fy_i \simeq \mathbf{y}_i F + \mathbf{F}\mathbf{y}'_i \Delta S$ as in (3.6). Similarly the terms $B_i q_i$ become: $B_i q_i \simeq \mathbf{q}_i B_i + \mathbf{B}_i \mathbf{q}'_i \Delta S$. The final qualities q_4 and q_5, are assumed to follow the standard quality-volume relationship given in (2.1)–(2.2). For example, the final quality of tank 4 is: $q_4 = (I_4^* q_4^* + B_1 q_1)/I_4$ where $I_4 = I_4^* + B_1$.

For tank 5 we assume that the blending stream B_6 is not removed from the tank until the process unit has finished adding to the tank via B_2. Thus, the quality q_5 used in the blending of demand D_8 is taken to be the same as the quality of the final inventory of I_5:

$$q_5 = (I_5^* q_5^* + B_2 q_2)/(I_5 + B_6), \qquad I_5 = I_5^* + B_2 - B_6.$$

Note that equation QD8 in Figure 2 does not provide the actual quality of the blend D_8; the quality q_8 is taken to be a fixed quality which is specified as a lower bound on the quality of the blend. The Taylor series approximation used for nonlinear terms like $I_5 q_5$ is: $I_5 q_5 \simeq \mathbf{q}_5 I_5 + \mathbf{I}_5 \Delta q_5$



	F	B_1	B_2	B_3	I_4	I_5	B_6	B_7	D_8	Δq_4	Δq_5	ΔS	S	RHS
OBJ														
BB1	y_1	-1										$\mathbf{F}\mathbf{y}'_1$		$=0$
BB2	y_2		-1									$\mathbf{F}\mathbf{y}'_2$		$=0$
BB3	y_3			-1								$\mathbf{F}\mathbf{y}'_3$		$=0$
BI4		1			-1									$= -I_4^*$
QI4		q_1			$-q_4$					$-I_4$		$\mathbf{B}_1\mathbf{q}'_1$		$= -q_4^* I_4^*$
BI5			1			-1	-1							$= -I_5^*$
QI5			q_2			$-q_5$	$-q_5$				$-I_5 - B_6$	$\mathbf{B}_2\mathbf{q}'_2$		$= -q_5^* I_5^*$
BD8				1			1	1	-1					$=0$
QD8				q_3			q_5	q_7	$-q_8$		B_6	$\mathbf{B}_1\mathbf{q}'_1$		≥ 0
BS													-1	$1 = S$

FIGURE 2. Matrix for Refinery Example.

Jones, C., Baker, T.E., "MIMI/G: A Graphical Environment for Mathematical Programming and Modeling", *Interfaces*, 26, **1996**.

STR	NL.	NH.	DS.	RF.	PLL CN.	PG.	RG.	HO.	FO.
NL.						F	F		
LH1	F	F							
LH2	F	F							
LH3	F	F							
NH.								F	
DS.								F	
GO.								F	F
RD.									F
RF.						F	F		
CN1					F				
CN2					F				
CL.								F	
CH.								F	F
CN.						F	F		
PG.									
RG.									
HO.									
FO.									

Table 1: The BLENd set indicates which streams are components of other streams. The "F" is an arbitrary nonblank character. For example, RG. (regular gas) is made from NL., RF., and CN.

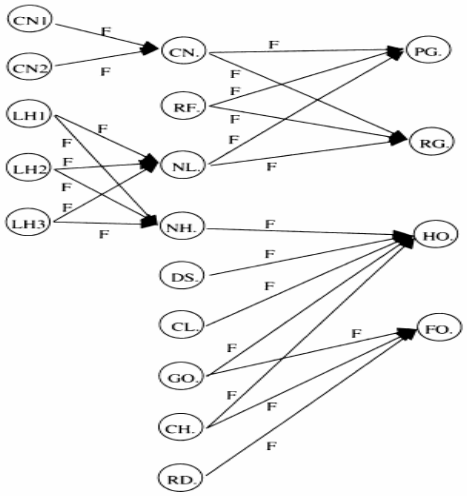


Figure 1: A graph-based representation of the blending relationship shown in Tabular form in Table 1.

Baker, T.E., Ladson, L.S., "Successive Linear Programming at Exxon", *Management Science*, 31, **1985**.

How do we Model the MOP? (cont'd)

- "Allegoric" details are developed from the **actual/material-world** to the **abstract/mathematical-world** into arrays allowing for the formation of general, dynamic, differential/integral, non-convex & non-linear algebraic calculations, constraints, complements & cuts.

- "Arrayic" details are deployed as efficient internal-memory, "home-grown" multi-dimensional (set-based) sparse arrays (MDSA) in Fortran 2003 exploiting its **whole array processing**, **dynamic allocation, operator overloading** & **parallelization** capabilities. These MDSA's then allow a "declarative" formation of the algebra in a "procedural" language.

- "Algorithmic" details are dispatched to the solver as variable and constraint dense vectors as well as first and second-order partial derivative sparse matrices (Jacobian & Hessian) derived using the novel **complex-step derivative** method (as "accurate" as analytical derivatives but computed numerically using complex-number algebra & almost as fast as finite-difference).

Martins, J.R.R.A., Sturdza, P., Alonso, J.J., "The Complex-Step Derivative Approximation", *ACM TOMS*, 29, **2003.**

How do we Model the MOP? (cont'd)

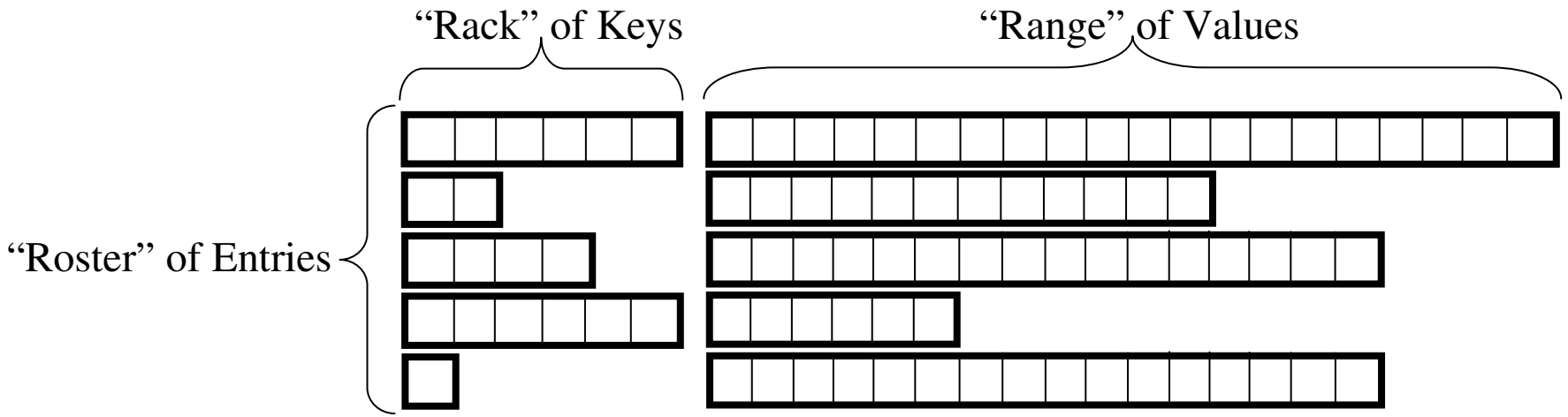
- **Our "home-grown" MDSA's are classified into the following library or language "resources":**
  - **Series-Sets** = vector of integer head, tail & stride tuple
  - **Simple-Sets** = vector of integer keys with integer value tuple
  - **Symbol-Sets** = vector of string keys with integer value tuple
  - **Lists** = tensor of integer keys with integer value tuple
  - **Parameters** = tensor of integer keys with real value tuple
  - **Catalogs** = tensor of integer keys with string value tuple
  - **Variables** = tensor of integer keys with real or complex value tuple
  - **Constraints** = tensor of integer keys with real or complex value tuple
  - **Terms** = tensor of integer keys with variable-address value tuple
  - **Derivatives** = matrix in sparse row-ordered coordinate storage format
  - **Expressions** = tensor of integer keys with RPN operator-operand tuple
  - **Functions** = tensor of integer keys with DLL-address value
  - **Rules** = tensor of integer keys with WHEN()-THEN() tuple

How do we Model the MOP? (cont'd)

- **Each MDSA resource is a hybrid of an "associative-array" or hash-table/dictionary/map with an "appendable-array" or value-tuple both of varying length or arity (similar to Google's Bigtable design).**

Chang, F., et. al. "Bigtable: A Distributed Storage System for Structured Data", *OSDI '06*, **2006.**

- **Accessing the non-zero entries to perform the "whole array" algebra can either be random/spot versus running/sequential and hence requires no lexicographic sorting required for heterogeneous tensor operations/manipulations (Bader & Kolda).**



Bader, B.W., Kolda, T.G., "Algorithm 862: MATLAB Tensor Classes for Fast Algorithm Prototyping", *ACM TOMS*, 32, **2006.**

How do we Model the MOP? (cont'd)

- **MDSA resources are declared, inserted, viewed & updated (accessed) using the following routines:**
  - **Reserve() = allocate memory for the resources dynamically**
  - **Roster() = procure a unique roster within the resource i.e., primary key**
  - **Register() = declare a roster with its rank (dimension of rack) & range**
  - **Refer() = point to or reference a particular key tuple for a roster – reference index is "memorized" for future accesses (iterations/successions).**
  - **Receive() = insert or update a roster reference value tuple given its key tuple**
  - **Retrieve() = view a roster reference value tuple given its key tuple**
  - **Restrain() = populate the lower & upper bounds for a variable or constraint**
  - **Ratio() = define constraint-variable 1st-order derivative elements symbolically**
  - **Restrict() = prevent any further inserts of key/value entries for the roster.**

How do we Model the MOP? (cont'd)

- **Why have our own "set-based" modeling-system (AML) instead of using GAMS, AIMMS, AMPL, LINGO, MPL, MOSEL, MATLAB, OPL, OML, OptimJ, ZIMPL, etc.?**
  - **PRO: Uses drastically less internal-memory (<20% of AML) to instantiate a problem instance thus enabling "multiple problem" parallelization i.e., executing many problem instances on different CPU's.**
  - **PRO: Is significantly faster (>20-times) to generate/iterate a problem instance using the whole-array processing for a discretized spatial, phenomenological or temporal dimension & the memorization of reference/hash/map-indices after iteration 1.**
  - **PRO: Has surprisingly little code (<20,000 LOC in Fortran 2003) to support & availability/re-use of existing "classical" Fortran/C language scientific utilities is high.**
  - **PRO: Can be used in either "reverse" & "callback" communication modes when interfacing to non-linear/iterative solvers i.e., can be embedded as a subroutine.**
  - **PRO: Easily extensible with Fortran 2003 DLL's to maintain propriety of code.**
  - **PRO: Straightforward to interface new solvers using their API's by modelers.**
  - **PRO: No royalties for each copy sold & has a less costly development environment.**
  - ***CON: Harder to formulate the arrayic-algebraic constraints because it does not have a "meta-language" (language within a language) to automate/abstract the programming details of the manipulations i.e., sparsity of Jacobian entered manually by modeler.***

How do we Model the MOP? (cont'd)

- **What are the other forms of modeling besides** set-based**?**
- Structure-based **(object-oriented) modeling e.g., NOVA-MS, Milano, SIMULA, Modelica, gPROMS, ASCEND, etc. useful for process simulation or optimization but limited to the local details of sub-models or modules (globally unaware of superstructure) & difficult to "sparsify".**
  - **Unit(u).Operation(o).Port(p).State(s).FLOW.Time(t) = … opposed to FLOW(u,o,p,s,t) = … where FLOW is a MDSA variable roster.**
  - **Desirable for data-modeling but difficult for data-manipulation.**

Mougin, P., Ducasse, S., "OOPAL: Integrating Array Programming in Object-Oriented Programming", *ACM*, **2003.**

- Scalar-based **modeling e.g., Excel, RPMS, PIMS, GRTMPS, PCOMP, MATLAB (no native sparse tensors) encode the pointers inside the symbol (or cell) name of the variable.**
  - **FLOWuops(t) = … where "uops" is hard-coded for each u,o,p,s instance.**

How do we Solve the MOP?

- **Solvers can be both commercial & public-domain in the form of linear programs (LP), mixed-logic linear programs (MLP), non-linear programs (NLP), constraint programs (CP) & meta-heuristics (MH) such as genetic algorithms, simulated annealing, differential evolution, etc. written in any compiled machine-code language such as Fortran, C or C++.**

- **Ultimately our modeling-system distills the problem into two dense vectors of continuous & discrete variable (x+) & constraint (f(x)+…) values & later a sparse matrix of derivatives (J,H) with the constraint algebra as follows:**

Linear Part: $IR * (r+ - r-) + ID * (d+ - d-) + IS+ * s+ + IS- * s- + A * x + b$

  - **where the "r"'s are the "reliefs", "d"'s are the "deviations" from target, "s"'s are "slacks/surpluses" & "b"'s are the constraint "balances".**

Non-Linear Part: $f(x) + F(x) * x + Q * w + T * v + D * u$

  - **where "f(x)"'s are the non-linear terms, "F(x)"'s are the non-linear coefficients, "w"'s are the quad-linear terms, "v"'s are the tri-linear terms & "u"'s are the bi-linear terms i.e., the "multi-linear" terms of x * x * x * x.**

How do we Solve the MOP? (cont'd)

- **Our routines to distill the "algorithmic" description from the "arrayic-algebraic" details are:**
  - Sparsity() = manually defined by the modeler (numerically too slow)
  - **Separability() = partitions the sparsity-pattern into groups**
  - **Sensitivity() = FD/CS Jacobian/Hessian calculations**
  - **Stationarity() = estimates linearity of constraints**
  - **Shapability() = assesses convexity/concavity of constraints**
  - **Shrinkability() = performs a primal pre-solve with "terms"**
  - **Stackability() = re-arranges Jacobian into row or column order**
  - **Scatterability() = pre-orders the Jacobian for factorization**
  - **Scalability() = scales variables & constraints using Jacobian**
  - **Startability() = determines heuristically good starting values**
  - **Solvability() = infeasibility analysis**
  - **Sensability() = observability, redundancy & precision estimates**

How do we Solve the MOP? (cont'd)

- **How do we derive the LP sub-problem in standard form from the non-linear algebra?**
- **Each constraint in the form $f(x) + \ldots + A*x + b = 0$ is called three times at startup of the problem definition:**
  - **First time with $x = 0$ to calculate b.**
  - **Second & third time with $x = x0$ & $x = x0 + h$ (or $x = Re(x0) + Im(h)$) to compute the Jacobian.**
- **During our primal pre-solve the constraints are called at each "pass" to update the b-balances as we successively fix variables in linear singleton constraints & substitute out variables in linear doubleton constraints (connection or transfer equations) i.e., $x(i) – x(j) = 0$ then $x(j)$ is replaced with $x(i)$ everywhere $x(j)$ appears in a constraint significantly reducing both the number of variables & constraints.**

How do we Solve the MOP? (cont'd)

- **How do we solve large-scale MINLP MOP's?**
- **"Scheduling" in continuous & batch-process industries is an excellent example where we have all three quantity, logic & quality phenomenon that need to be decided on over a relatively long time-horizon with a mix of "small & big bucket" time-intervals (restriction on level of contention).**
- **Our novel & effective approach applied to many diverse industrial sites, is to implement a "truncated" Bender's Decomposition with two sub-problems solved recursively:**
  - **Solve a "Logistics" MOP (quantity-logic) using MLP then**
  - **Solve a "Quality" MOP (quantity-quality) using NLP then**
  - **Iterate until useful (optimized & feasible) solutions are obtained.**

Kelly, J.D. "The Unit-Operation-Stock Superstructure (UOSS) and the Quantity-Logic-Quality Paradigm (QLQP) for Production Scheduling in the Process Industries. *MISTA 2005,* 327–333, **2005**.

How do we Solve the MOP? (cont'd)

- **How do we handle "infeasible" or "inconsistent" MOP's?**
- **Searching for Irreducible Infeasibility Subsets (IIS, Chinneck) is unfortunately not practical for industrial-sized MOP's nor is solving the Primal Simplex "Phase I" to obtain marginal-values on slacks (Primal too slow).**

Chinneck, J.W., "Feasibility and Infeasibility in Optimization", *Springer*, 2008.

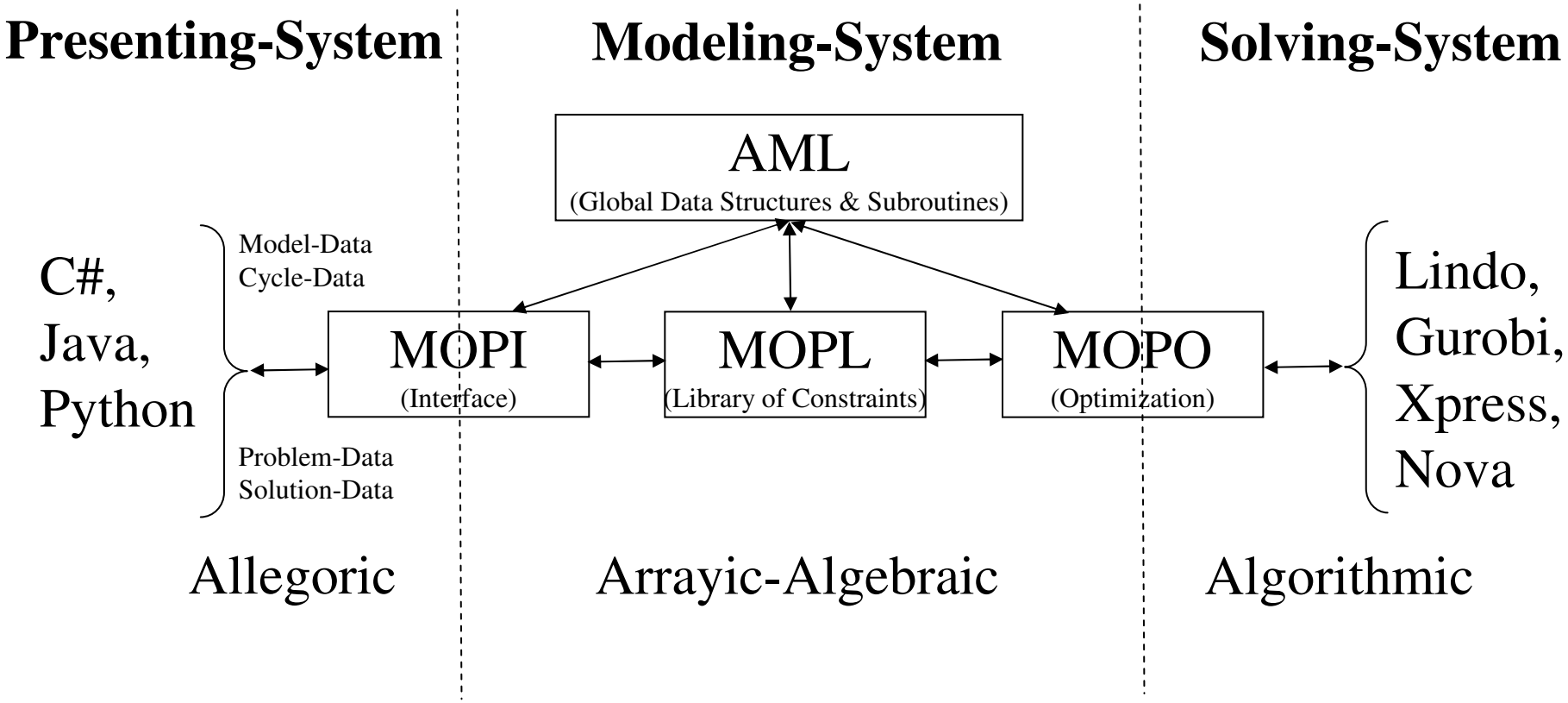
- **We employ the following two techniques:**
  - **Given our automated but "baroque" generation of the problem (UOPSS/QLQP), primal pre-solve has proven to be very effective on the first LP sub-problem of either the MLP or NLP to detect, isolate & identify inconsistencies especially in linear constraints.**
  - **Penalty, artificial or elastic variables are added to specific modeler-defined constraints & are minimized in our "Archimedean" objective function. These penalties provide a useful level of isolation & identification in addition to being displayed graphically.**

How do we Integrate the MOP Data?

- **We decompose the diverse data requirements into:**
  - **Model-Data + Cycle-Data = Problem-Data ==solve==> Solution-Data**
- **Model-data provides the "prototype" whereas cycle-data (cases, changes) provides the "provisos" and overloads model-data to provide the problem-data "profiles" i.e., "pole(p,o,i,n,t,e,r,s) = profile" MDSA's.**
- **The data is "presented" & "persisted" in several allegoric "forms":**
  - **Graph/Relation/Object/Network-Based (*Traversal*), Paragraph/Sentence/Word-Based (*Textual*) & Line/List/Sheet/Table-Based (*Tabular*).**
  - **These data-forms can then be dispatched from its source data depot as simple files/text streams in CSV or XML formats.**
  - **These files/streams are imported then lexed, parsed & expanded into the necessary MDSA's as problem-data & then after solving,**
  - **Solution-data is then exported into these non-array-based forms.**

How do we Integrate the MOP Data? (cont'd)

- **System Architecture delineated …**



How do we Execute the MOP Decisions?

- **We define three execution environments called "in-line", "on-line" & "off-line".**
- ***In-line* MOP's have "variable feedback" in terms of past/present data but no or very little "parameter feedback" & with "actuation/manipulation" (planning & scheduling) – exhibits "steady-state offsets".**

Kelly, J.D., Zyngier, D., "Continuously Improve the Performance of Planning & Scheduling Models with Parameter Feedback", *FOCAPO*, **2008**.

- ***On-line* MOP's have both variable & parameter feedback (i.e., bias updating) using past/present data with actuation (regulatory & maneuvering control).**
- ***Off-line* MOP's may or may not have either or both "variable" & "parameter" feedback without direct actuation or with indirect actuation (design & analysis).**

How do we Improve the MOP Deployment?

- **Kaizen, POMAI, DMAIC, PECA, Six Sigma, TPS, etc. enable the necessary "continuous-improvement" we call our** Plan-Perform-Perfect-Loop **(also addresses Uncertainty).**

Kelly, J.D., "Modeling Production-Chain Information", *Chemical Engineering Progress*, February, **2005**.

- **The core idea is that the modeling, solving, etc. of a MOP "is not it, but a part of it" & needs to be refined & revisited regularly in context with the overall business objectives & obstacles.**

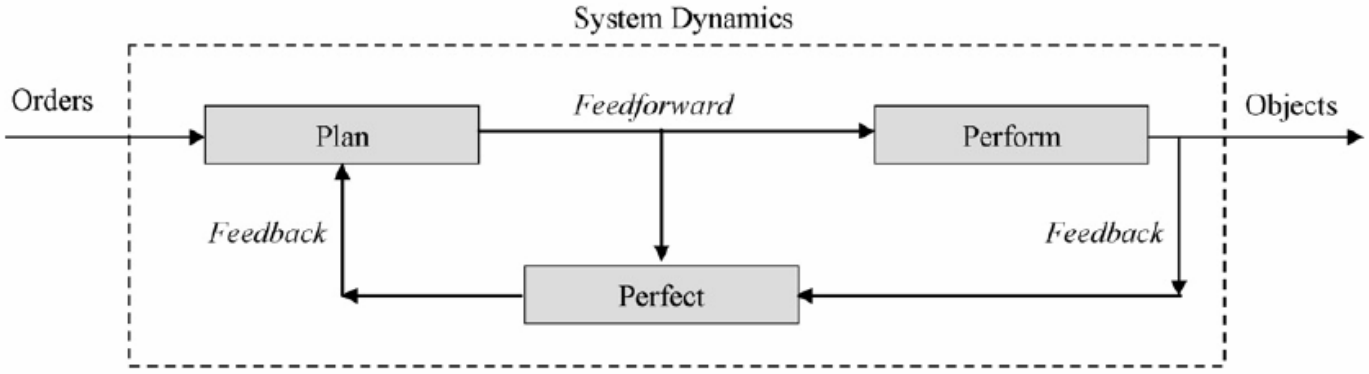


Fig. 14. The plan–perform–perfect-loop of the HDH with orders-to-objects system dynamics.

Kelly, J.D., Zyngier, D., "Hierarchical Decomposition Heuristic for Scheduling: Coordinated Reasoning for Decentralized & Distributed Decision-Making", *Computers & Chemical Engineering*, 32, **2008**.

- THANK YOU!