GENO™ 1.0

A Global Numerical Optimizer

GAUSS™

What is GENO™?

- GENO is an acronym for:
  ***G**eneral **E**volutionary **N**umerical **O**ptimizer*
- GENO is a real-coded genetic algorithm that can be used to solve uni- or multi-objective optimization problems. The problems presented may be static or dynamic in character; they may be unconstrained or constrained by equality or inequality constraints, coupled with upper and lower bounds on the variables. The variables themselves may assume real or discrete values in any combination. In fact, except for the relatively benign requirement that, if present, all equation constraints should preferably be affine in the current control, the algorithm does not require the problem presented to have any other special structure.

Practical Applications

- GENO has been tested on many optimization problems from well-known test suites that cover a wide range of problem-types. GENO consistently out-performs many algorithms of its genre in terms of solution quality. Some practical examples solved include:
  - Pressure Vessel Design
  - Oligopolist Market Equilibrium
  - Efficient Portfolio Selection
  - Decentralised Economic Planning

Platforms/Requirements

- Windows, UNIX, Linux, Mac OS
- Requires GAUSS™ 6.0 or above

Product Attributes

- GENO may be specialized for various classes of problems such as the general static optimization problem, the general dynamic optimization problem, the mixed integer problem, and the two-point boundary value problem, by mere choice of a few parameters. These properties are easily pre-set at the problem set-up stage of the solution process. GENO includes a quantization mechanism that significantly enhances the rate of convergence as well as the quality of the final solution. So if your model involves:
  - static Optimization
  - dynamic Optimization
  - robust Optimization
  - mixed Integer Optimization
  - multi-objective Optimization
  - and more . . .

  chances are GENO can help.
- GENO is well documented and easy to use
- Examples programs are available to kick-start your fruitful adventure with GENO.

Pricing Information

Contact Aptech or your local Dealer for pricing and information. See our website for the Dealer nearest you: http://www.Aptech.com

Aptech Systems Inc.

P.O. Box 250 ♦ Black Diamond, WA 98010 USA

Phone: (425) 432-7855 ♦ FAX: (425) 432-7832 ♦ info@Aptech.com ♦ URL: www.Aptech.com

A Comparative Capability Map of Some Existing NLP Solvers [1]

Constrained NLP Solver	Applicability [2]																		Convexity Required	Principal Methods	Input Formats	Special Requirements
	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18				
DONLOP2	✓																		No	Sequential Quadratic Programming	Fortran, AMPL	
LANCELOT	✓																		No	Sequential Quadratic Programming	SIF, AMPL	
LOQO	✓																		Yes	Infeasible primal-dual interior-point	AMPL, Matlab	
MINOS	✓																		No	Sequential linearly constrained algo	GAMS, AMPL	Modest Nonlinearity
KNITRO	✓																		No	primal-dual interior-point	AMPL	Modest Free Variables
SNOPT	✓																		No	Sequential Quadratic Programming	Fortran, GAMS, AMPL	
CONOPT	✓																		No	Feasible Path Method	GAMS	
FSQP	✓																		No	Sequential Quadratic Programming	AMPL	
HQP/OMUSES	✓																		No	Newton-type SQP, integer-point	SIF, C++	
MOSEK	✓																		Yes	Best Interior-point method	AMPL	
GENECOP	✓	✓	✓																No	Genetic Algorithm	C	
COBYLA2	✓	✓	✓																No	SLP with Gradient estimates	Fortran	Inequality Constraints Only
BARON	✓			✓			✓												No	Branch and Reduce	Baron Model	Functions are Factorable
BNB	✓			✓			✓												No	Branch and Bound	Matlab	
MINLP_BB	✓			✓			✓												No	Branch and Bound, SQP	AMPL	
SBB	✓			✓			✓												No	Branch and Bound	GAMS	
MITTLP	✓			✓			✓												Yes	Extended Cutting Plane	C	
ALPHAECP	✓			✓			✓												No	Extended Cutting Plane	Fortran, LP	Functions are Pseudo-convex
AUGMENTED										✓									No	Interior-point, augmented system	Standard SLP	Linear Programs Only
MSLIP										✓									No	Nested Benders Decomposition	Standard SLP	Linear Programs Only
GENO	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	No	Multi-objective Genetic Algorithm	GAUSS, C++, Mathematica	

[1] **Source**: Zhang, H (2001). *Improving Constrained Nonlinear Search Algorithms Through Constraint Relaxation*. Unpublished MSc Thesis, University of Illinois at Urbana-Champaign.

[2] See Table below for an explanation of the NLP classes C1, C2, . . . , C18.

A Classification of Constrained Non-linear Programming Problems [1]

CONSTRAINED NLP CLASS	VARIABLE TYPE	FUNCTION TYPE
C1	Continuous	Continuous and Differentiable
C2	Continuous	Continuous and Non-differentiable
C3	Continuous	Discontinuous
C4	Discrete	Continuous and Differentiable
C5	Discrete	Continuous and Non-differentiable
C6	Discrete	Discontinuous
C7	Mixed-integer	Continuous and Differentiable
C8	Mixed-integer	Continuous and Non-differentiable
C9	Mixed-integer	Discontinuous
C10	Continuous	Stochastic Objective and Constraints
C11	Discrete	Stochastic Objective and Constraints
C12	Mixed-integer	Stochastic Objective and Constraints
C13	Continuous	Vector Objective Function
C14	Discrete	Vector Objective Function
C15	Mixed-integer	Vector Objective Function
C16	Continuous	Dynamic Constraints
C17	Discrete	Dynamic Constraints
C18	Mixed-integer	Dynamic Constraints

[1] Global Optimisation problems are very heterogeneous: they encompass the usual categories of mathematical programming models, specifically including linear models and the broad nonlinear category. Unsurprisingly, there is no universally accepted way of categorizing nonlinear programs. This classification is essentially that suggested by Honghai Zhang in his MSc Thesis (University of Illinois at Urbana-Champagne, 2001) to which the classes C13 through C18 have been added. Classes C16-18 were included in order to emphasize the fact that GENO is equally applicable to dynamic optimisation problems, despite the fact that the discrete-time dynamic programming problem can in principle be converted to a static program by the technique of variable stacking.