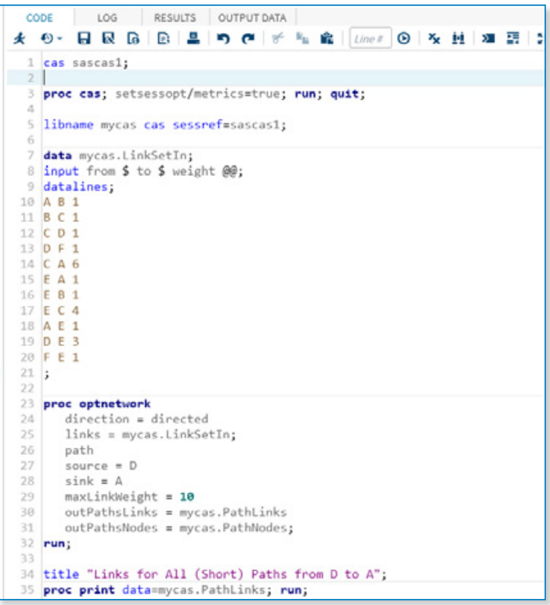
# SAS® Optimization

Find optimal solutions to complex business and planning problems faster than ever



```
cas sascas1;

proc cas; setsessopt/metrics=true; run; quit;

libname mycas cas sessref=sascas1;

data mycas.LinkSetIn;
input from $ to $ weight @@;
datalines;
A B 1
B C 1
C D 1
D F 1
C A 6
E A 1
E B 1
E C 4
A E 1
D E 3
F E 1
;

proc optnetwork
   direction = directed
   links = mycas.LinkSetIn;
   path
   source = D
   sink = A
   maxLinkWeight = 10
   outPathsLinks = mycas.PathLinks
   outPathsNodes = mycas.PathNodes;
run;

title "Links for All (Short) Paths from D to A";
proc print data=mycas.PathLinks; run;
```

## What does SAS® Optimization do?

SAS Optimization provides a powerful array of optimization, simulation and project scheduling techniques to identify the actions that will produce the best results, while operating within resource limitations and other relevant restrictions.

## Why is SAS® Optimization important?

Organizations can consider more alternative actions and scenarios, and determine the best allocation of resources and plans for accomplishing goals. Incorporating operations research analytics adds structure and repeatability to decision-making processes, lets you make the most of your analytical and BI investments, and delivers a competitive edge.

## For whom is SAS® Optimization designed?

It is designed for people in any industry with operations research (or management science) experience who build decision-guidance models by applying operations research techniques to solve real-world problems.

Figuring out what actions produce the best outcomes is key to solving everyday problems that all organizations face. Whether it's carrying the right inventory in the right store, identifying the best site for a new facility, optimizing supply chain logistics, scheduling staff or allocating resources, SAS Optimization software helps model and solve a variety of complex planning problems quickly.

Optimization is a type of prescriptive analytics within the data science discipline that finds a "best" solution from a set of feasible solutions, using mathematical algorithms that maximize or minimize a specified objective function subject to constraints. It originated in the field of operations research and management science.

SAS Optimization provides a powerful array of modeling capabilities and solution techniques to help organizations consider more alternative scenarios to determine the best course of action. Common approaches to this type of problem solving include linear programming, integer programming, stochastic programming and constraint programming. This solution benefits from parallel computation, including distributed processing across the grid and threaded processing within a single core .

With SAS Optimization, you can:

- Identify actions that will produce the best results, while operating within resource and other relevant constraints.
- Determine the optimal allocation of resources, and select the best way to achieve organizational goals.
- Add structure, consistency, adaptability and repeatability to decision-making processes.
- Avoid the hassles of dealing with niche software packages.

## Benefits

- **Drive better decision making.** SAS Optimization offers state-of-the-art methods for mathematical optimization that are integrated with a full suite of data preparation, exploration, visualization, analytics and reporting capabilities. This unifying environment enables organizations to identify and rapidly apply the best responses to complex, real-world problems.
- **Quickly solve complex optimization problems.** This solution takes advantage of SAS® Viya®, a distributed, in-memory engine, to deliver optimization modeling results at breakthrough speeds. Find optimal solutions to difficult problems faster.
- **Empower users with the programming language of their choice.** Python, Java, R and Lua programmers can take advantage of SAS Optimization capabilities without learning SAS code. They can access powerful, trusted and tested SAS algorithms from the language they're most comfortable with.

## Overview

SAS Optimization provides a powerful, intuitive algebraic optimization modeling language and an array of algorithms. You can produce a range of models, including linear, mixed-integer linear, nonlinear, quadratic, and network optimization, as well as solve constraint satisfaction problems. These sophisticated mathematical programming techniques help determine the best use of limited resources to achieve your objectives.

Any SAS Optimization feature can interact with other SAS data management, analytical, exploratory or reporting capabilities. This means your organization can eliminate the assorted collection of data analysis tools that are difficult to modify and support.

Operations research analysts typically spend a lot of time struggling with data and infrastructure issues. With SAS Optimization, they can get right to the modeling with minimal distractions. An integrated process makes it easy to review models for initial validation, make adjustments and run models with new data.

### Unified modeling language supports a wide range of optimization models

With a single modeling and solution framework, you only need to learn one set of statements and commands to build a range of optimization and constraint satisfaction models.

As you build optimization models to capture the key elements of systems and associated planning problems, it's common for the type of model to change as different elements are added (integer or binary variables, network structure, etc.). Optimization models evolve as they are defined and refined. With SAS Optimization, you retain the same transparent modeling environment, even as your models change and evolve. This saves time and improves productivity.

If your models don't fit neatly within a defined type and contain different elements, you can build hybrid, customized algorithms using SAS Optimization solvers as building blocks within the SAS Optimization language.

### Powerful optimization solvers and presolvers

SAS Optimization provides a suite of optimization solvers – all streamlined for simplicity and tuned for performance. Aggressive presolvers reduce effective problem size so you can tackle large problems and solve them more quickly.

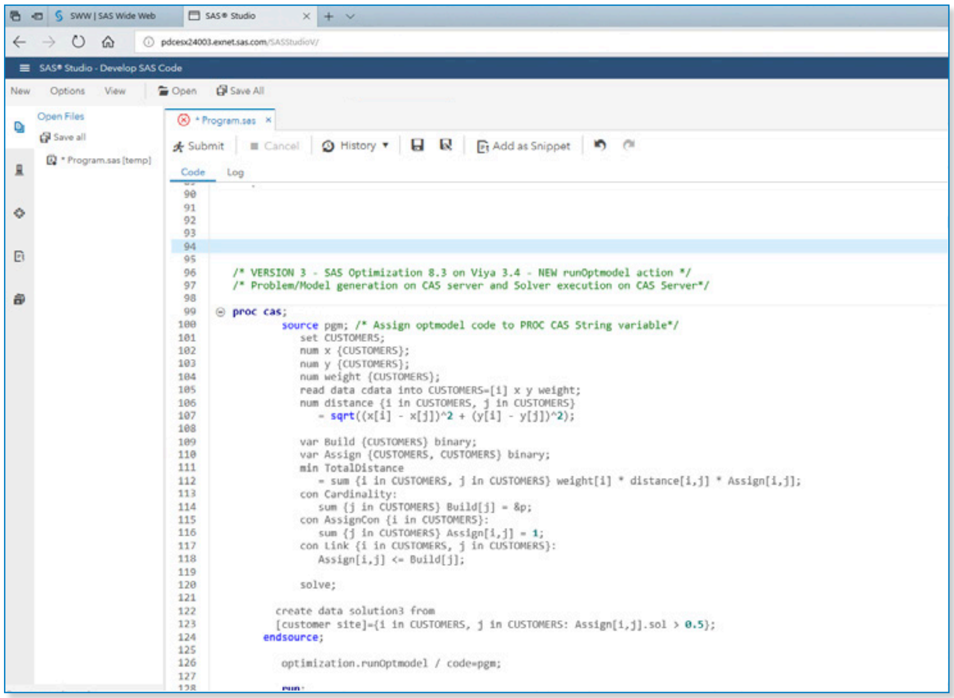
- Linear programming (LP) solvers include primal and dual simplex; network simplex; and interior point with crossover.
- General nonlinear optimization solvers include active set and interior point; concurrent solve; and the multistart algorithm.
- A distributed branch-and-bound, mixed-integer linear programming (MILP) solver has cutting planes, heuristics, conflict search and option tuning. Distributed computation reduces the time needed to find an optimal solution, even more so for larger problems.
- A state-of-the-art quadratic solver is also tailored for large-scale optimization problems.
- The DECOMP (decomposition) algorithm, applicable to LP and MILP problems, also delivers performance improvements.

Improvements in numerical stability better equip these five solvers and the decomposition algorithm to handle problems that have ill-conditioned constraint matrices.

### Network optimization

Network algorithms, accessible from both PROC OPTMODEL and PROC OPTNETWORK, enable you to investigate the characteristics of networks and find the best answers to network-oriented problems. Optimization and diagnostic algorithms include shortest path, maximum flow, minimum-cost flow, traveling salesman problem, connected and biconnected components, clique and cycle enumeration, minimum spanning tree, linear assignment and more.

You can submit PROC OPTMODEL code to the SAS Viya CAS server, so that model creation (as well as solver execution) runs on CAS. This action enables optimization models to be built where their source data resides and provides optimization modeling access from any programming language that SAS Viya supports, including Python, R, Lua, and Java.



```
/* VERSION 3 - SAS Optimization 8.3 on Viya 3.4 - NEW runOptmodel action */
/* Problem/Model generation on CAS server and Solver execution on CAS Server*/

proc cas;
        source pgm; /* Assign optmodel code to PROC CAS String variable*/
           set CUSTOMERS;
           num x {CUSTOMERS};
           num y {CUSTOMERS};
           num weight {CUSTOMERS};
           read data cdata into CUSTOMERS=[i] x y weight;
           num distance {i in CUSTOMERS, j in CUSTOMERS}
              = sqrt((x[i] - x[j])^2 + (y[i] - y[j])^2);

           var Build {CUSTOMERS} binary;
           var Assign {CUSTOMERS, CUSTOMERS} binary;
           min TotalDistance
              = sum {i in CUSTOMERS, j in CUSTOMERS} weight[i] * distance[i,j] * Assign[i,j];
           con Cardinality:
              sum {j in CUSTOMERS} Build[j] = &p;
           con AssignCon {i in CUSTOMERS}:
              sum {j in CUSTOMERS} Assign[i,j] = 1;
           con Link {i in CUSTOMERS, j in CUSTOMERS}:
              Assign[i,j] <= Build[j];

           solve;

       create data solution3 from
       [customer site]={i in CUSTOMERS, j in CUSTOMERS: Assign[i,j].sol > 0.5};
     endsource;

        optimization.runOptmodel / code=pgm;

        run;
```

You can use the new runOptmodel action (in the Optimization action set) to build and solve a facility location and customer sourcing problem. PROC OPTMODEL syntax is submitted via PROC CAS.

A path enumeration algorithm finds all paths between specified starting and ending nodes. You can specify a single starting node or all, and a single ending node or all.

The connected components algorithm provides the option to use a thin format for specifying the links in the network. This format can save memory (especially for larger networks), but because of how it stores the network connectivity information, it's not necessarily suitable for every network algorithm.

### Multistart algorithm helps identify better solutions to nonlinear problems

The multistart algorithm for nonconvex nonlinear optimization increases your chance of finding a globally optimal solution from among many locally optimal solutions. This iterative algorithm selects multiple starting points and begins optimization in parallel from each. The best solution from all starting points is reported.

### Decomposition algorithm

The decomposition algorithm (automated Dantzig-Wolfe) works well for large, appropriately structured linear and mixed-integer linear optimization problems. It decomposes the overall problem into a set of component problems, each with an exclusive set of decision variables, which are solved in parallel. Parallel solution of the subproblems is coordinated with the overall solution process, reducing time to solution significantly.

### Local search optimization

A local search optimization (LSO) solver can be used with (generally nonlinear) optimization problems that don't adhere to the assumptions that conventional optimization solvers make. Functions might be discontinuous, nonsmooth, computationally expensive to evaluate, based on black-box simulations, etc.

### Constraint programming

Constraint satisfaction problems can be solved using domain reduction/constraint propagation and a choice of search

## Key Features

### Algebraic, symbolic optimization modeling language

- Flexible algebraic syntax for intuitive model formulation.
- Support for the transparent use of SAS functions.
- Direct invocation of linear, nonlinear, quadratic and mixed-integer solvers.
- Support for the rapid prototyping of customized optimization algorithms, including support for named problems and subproblems.
- Use of industry-standard MPS/QPS format input data sets.
- Aggressive presolvers to reduce problem size.

### Powerful optimization solvers

- Linear solution algorithms: primal and dual simplex, network simplex, interior point with crossover, and concurrent solve capability.
- Mixed-integer linear programming solution algorithm: branch-and-bound integer with cutting planes, primal heuristics, conflict search and option tuning.
- Decomposition algorithm (automated Dantzig-Wolfe) for linear programming and mixed-integer linear programming problems with block-angular, block-diagonal or embedded network structure.
- Quadratic solution algorithm: interior point with state-of-the-art solver tailored for large-scale optimization problems.
- Nonlinear solution algorithms: active set, interior point. Concurrent solve capability. Multistart algorithm for nonconvex problems.

### Network optimization

- Optimization and diagnostic algorithms include:
  - Connected components and biconnected components (with articulation points).
  - Clique and cycle enumeration.
  - Transitive closure.
  - Minimum cut.
  - Minimum spanning tree.
  - Summary.
  - Linear assignment.
  - Minimum-cost network flow.
  - Shortest path.
  - Traveling salesman problem.
  - Path enumeration.

### Local search optimization and constraint programming

- Local search optimization: hybrid parallel algorithm, including generic algorithms, global GA-type heuristics and pattern search. Multiobjective optimization identifies a set of non-dominated solutions, for which no other solution delivers better values for all objectives.
- Solve constraint satisfaction problems using domain reduction/constraint propagation and a choice of search strategies (look ahead and backtracking).

### Distributed, accessible and cloud-ready

- Runs on SAS Viya, a scalable and distributed in-memory engine of the SAS Platform.
- Distributes analysis and data tasks across multiple computing nodes.
- Provides fast, concurrent, multiuser access to data in memory.
- Includes fault tolerance for high availability.
- Lets you add the power of SAS Analytics to other applications using SAS Viya REST APIs.

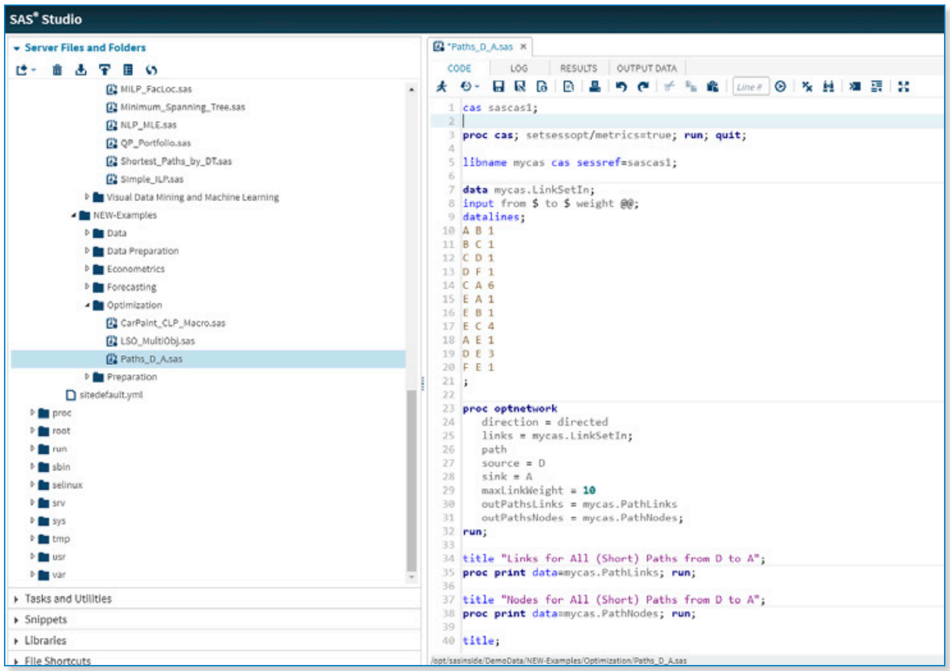
strategies such as look-ahead and back-tracking. The CLP procedure solves standard constraint satisfaction problems in distributed mode. The only categorical exception is the OBJECTIVE statement in PROC CLP, which isn't supported in distributed mode. Also, if you use the EVALVARSEL statement to specify multiple variable selection strategy, then the strategies will execute in distributed mode only if the number of nodes available is at least as great as the number of specified strategies. Otherwise they will execute sequentially on one node.

## Accessible, cloud-enabled, in-memory engine

SAS Optimization takes advantage of the SAS Viya engine for even quicker insights. SAS Viya brings high availability, fast in-memory processing, the ability to code from open source languages and native cloud support to the SAS Platform. SAS Optimization is available for both public and private cloud delivery in a scalable and elastic environment. Processing can spin up or down as needed. Solve your largest optimization and planning problems using the most appropriate computing resources.

**TO LEARN MORE »**

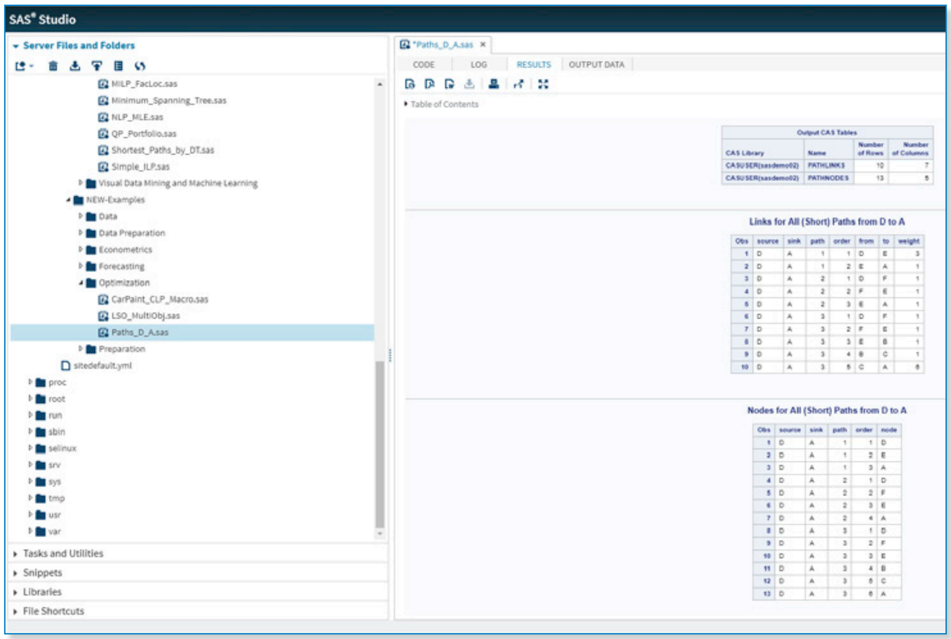
To learn more about SAS Optimization, view screenshots and see other related materials, please visit sas.com/optimization. SAS Optimization includes a license to SAS/OR. Find details on SAS/OR procedures and capabilities at sas.com/or.



```
cas sascas1;

proc cas; setsessopt/metrics=true; run; quit;

libname mycas cas sessref=sascas1;

data mycas.LinkSetIn;
input from $ to $ weight @@;
datalines;
A B 1
B C 1
C D 1
D F 1
C A 6
E A 1
E B 1
E C 4
A E 1
D E 3
F E 1
;

proc optnetwork
   direction = directed
   links = mycas.LinkSetIn;
   path
   source = D
   sink = A
   maxLinkWeight = 10
   outPathsLinks = mycas.PathLinks
   outPathsNodes = mycas.PathNodes;
run;

title "Links for All (Short) Paths from D to A";
proc print data=mycas.PathLinks; run;

title "Nodes for All (Short) Paths from D to A";
proc print data=mycas.PathNodes; run;

title;
```

/opt/sasinside/DemoData/NEW-Examples/Optimization/Paths_D_A.sas

The SAS Studio coding interface in SAS Optimization provides syntax help such as auto-complete so you can quickly experiment with powerful SAS Optimization solvers. Here is an example of code that loads network data for use with the path algorithm in the OPTNETWORK procedure to find all paths that start at node D and end at node A.



Output CAS Tables

| CAS Library | Name | Number of Rows | Number of Columns |
|---|---|---|---|
| CASUSER(sasdemo02) | PATHLINKS | 10 | 7 |
| CASUSER(sasdemo02) | PATHNODES | 13 | 5 |

Links for All (Short) Paths from D to A

| Obs | source | sink | path | order | from | to | weight |
|---|---|---|---|---|---|---|---|
| 1 | D | A | 1 | 1 | D | E | 3 |
| 2 | D | A | 1 | 2 | E | A | 1 |
| 3 | D | A | 2 | 1 | D | F | 1 |
| 4 | D | A | 2 | 2 | F | E | 1 |
| 5 | D | A | 2 | 3 | E | A | 1 |
| 6 | D | A | 3 | 1 | D | F | 1 |
| 7 | D | A | 3 | 2 | F | E | 1 |
| 8 | D | A | 3 | 3 | E | B | 1 |
| 9 | D | A | 3 | 4 | B | C | 1 |
| 10 | D | A | 3 | 5 | C | A | 6 |

Nodes for All (Short) Paths from D to A

| Obs | source | sink | path | order | node |
|---|---|---|---|---|---|
| 1 | D | A | 1 | 1 | D |
| 2 | D | A | 1 | 2 | E |
| 3 | D | A | 1 | 3 | A |
| 4 | D | A | 2 | 1 | D |
| 5 | D | A | 2 | 2 | F |
| 6 | D | A | 2 | 3 | E |
| 7 | D | A | 2 | 4 | A |
| 8 | D | A | 3 | 1 | D |
| 9 | D | A | 3 | 2 | F |
| 10 | D | A | 3 | 3 | E |
| 11 | D | A | 3 | 4 | B |
| 12 | D | A | 3 | 5 | C |
| 13 | D | A | 3 | 6 | A |

Output from running the path algorithm in PROC OPTNETWORK.

To contact your local SAS office, please visit: sas.com/offices

THE POWER TO KNOW®