

# Vineopt and friends

Vineopt and friends

Software made by Kaj Holmberg

# Vineopt and friends

Software made by Kaj Holmberg since 2002:

# Vineopt and friends

Software made by Kaj Holmberg since 2002:

Vineopt: Visual network optimization

# Vineopt and friends

Software made by Kaj Holmberg since 2002:

Vineopt: Visual network optimization (Tcl/Tk, C, C++, fortran)

# Vineopt and friends

Software made by Kaj Holmberg since 2002:

Vineopt: Visual network optimization (**Tcl/Tk, C, C++, fortran**)

Vileopt: Visual linear optimization

# Vineopt and friends

Software made by Kaj Holmberg since 2002:

Vineopt: Visual network optimization (**Tcl/Tk, C, C++, fortran**)

Vileopt: Visual linear optimization (**Tcl/Tk**)

# Vineopt and friends

Software made by Kaj Holmberg since 2002:

Vineopt: Visual network optimization ([Tcl/Tk, C, C++, fortran](#))

Vileopt: Visual linear optimization ([Tcl/Tk](#))

Nileopt: Visual nonlinear optimization

# Vineopt and friends

Software made by Kaj Holmberg since 2002:

Vineopt: Visual network optimization (Tcl/Tk, C, C++, fortran)

Vileopt: Visual linear optimization (Tcl/Tk)

Nileopt: Visual nonlinear optimization (Tcl/Tk, SciPy)

# Vineopt and friends

Software made by Kaj Holmberg since 2002:

Vineopt: Visual network optimization (Tcl/Tk, C, C++, fortran)

Vileopt: Visual linear optimization (Tcl/Tk)

Nileopt: Visual nonlinear optimization (Tcl/Tk, SciPy)

Vigropt: Visual group optimization

# Vineopt and friends

Software made by Kaj Holmberg since 2002:

Vineopt: Visual network optimization (Tcl/Tk, C, C++, fortran)

Vileopt: Visual linear optimization (Tcl/Tk)

Nileopt: Visual nonlinear optimization (Tcl/Tk, SciPy)

Vigropt: Visual group optimization (Python, Tkinter, NumPy)

# Vineopt and friends

Software made by Kaj Holmberg since 2002:

Vineopt: Visual network optimization (Tcl/Tk, C, C++, fortran)

Vileopt: Visual linear optimization (Tcl/Tk)

Nileopt: Visual nonlinear optimization (Tcl/Tk, SciPy)

Vigropt: Visual group optimization (Python, Tkinter, NumPy)

Vikeopt (Snowplan): Visual  $k$ -Chinese postman problem solver

# Vineopt and friends

Software made by Kaj Holmberg since 2002:

Vineopt: Visual network optimization (Tcl/Tk, C, C++, fortran)

Vileopt: Visual linear optimization (Tcl/Tk)

Nileopt: Visual nonlinear optimization (Tcl/Tk, SciPy)

Vigropt: Visual group optimization (Python, Tkinter, NumPy)

Vikeopt (Snowplan): Visual  $k$ -Chinese postman problem solver (Python, Tkinter, NumPy, SciPy, Mayavi, matplotlib, networkx)

# Vineopt and friends

Software made by Kaj Holmberg since 2002:

Vineopt: Visual network optimization (Tcl/Tk, C, C++, fortran)

Vileopt: Visual linear optimization (Tcl/Tk)

Nileopt: Visual nonlinear optimization (Tcl/Tk, SciPy)

Vigropt: Visual group optimization (Python, Tkinter, NumPy)

Vikeopt (Snowplan): Visual  $k$ -Chinese postman problem solver (Python, Tkinter, NumPy, SciPy, Mayavi, matplotlib, networkx)

Viseopt: Visual Sudoku help

# Vineopt and friends

Software made by Kaj Holmberg since 2002:

Vineopt: Visual network optimization ([Tcl/Tk, C, C++, fortran](#))

Vileopt: Visual linear optimization ([Tcl/Tk](#))

Nileopt: Visual nonlinear optimization ([Tcl/Tk, SciPy](#))

Vigropt: Visual group optimization ([Python, Tkinter, NumPy](#))

Vikeopt (Snowplan): Visual  $k$ -Chinese postman problem solver ([Python, Tkinter, NumPy, SciPy, Mayavi, matplotlib, networkx](#))

Viseopt: Visual Sudoku help ([Python, Tkinter, NumPy](#))

# Vineopt and friends

Software made by Kaj Holmberg since 2002:

Vineopt: Visual network optimization (Tcl/Tk, C, C++, fortran)

Vileopt: Visual linear optimization (Tcl/Tk)

Nileopt: Visual nonlinear optimization (Tcl/Tk, SciPy)

Vigropt: Visual group optimization (Python, Tkinter, NumPy)

Vikeopt (Snowplan): Visual  $k$ -Chinese postman problem solver (Python, Tkinter, NumPy, SciPy, Mayavi, matplotlib, networkx)

Viseopt: Visual Sudoku help (Python, Tkinter, NumPy)

Small improvements each year/all the time/when I have the time.

# Vineopt and friends

Software made by Kaj Holmberg since 2002:

Vineopt: Visual network optimization (Tcl/Tk, C, C++, fortran)

Vileopt: Visual linear optimization (Tcl/Tk)

Nileopt: Visual nonlinear optimization (Tcl/Tk, SciPy)

Vigropt: Visual group optimization (Python, Tkinter, NumPy)

Vikeopt (Snowplan): Visual  $k$ -Chinese postman problem solver (Python, Tkinter, NumPy, SciPy, Mayavi, matplotlib, networkx)

Viseopt: Visual Sudoku help (Python, Tkinter, NumPy)

Small improvements each year/all the time/when I have the time.

Now large: vineopt 70 000 lines of code (excl opt).

# Vineopt and friends

Software made by Kaj Holmberg since 2002:

Vineopt: Visual network optimization (Tcl/Tk, C, C++, fortran)

Vileopt: Visual linear optimization (Tcl/Tk)

Nileopt: Visual nonlinear optimization (Tcl/Tk, SciPy)

Vigropt: Visual group optimization (Python, Tkinter, NumPy)

Vikeopt (Snowplan): Visual  $k$ -Chinese postman problem solver (Python, Tkinter, NumPy, SciPy, Mayavi, matplotlib, networkx)

Viseopt: Visual Sudoku help (Python, Tkinter, NumPy)

Small improvements each year/all the time/when I have the time.

Now large: vineopt 70 000 lines of code (excl opt).

Tcl/Tk good for graphics etc, bad for computations.

# Vineopt and friends

Software made by Kaj Holmberg since 2002:

Vineopt: Visual network optimization ([Tcl/Tk, C, C++, fortran](#))

Vileopt: Visual linear optimization ([Tcl/Tk](#))

Nileopt: Visual nonlinear optimization ([Tcl/Tk, SciPy](#))

Vigropt: Visual group optimization ([Python, Tkinter, NumPy](#))

Vikeopt (Snowplan): Visual  $k$ -Chinese postman problem solver ([Python, Tkinter, NumPy, SciPy, Mayavi, matplotlib, networkx](#))

Viseopt: Visual Sudoku help ([Python, Tkinter, NumPy](#))

Small improvements each year/all the time/when I have the time.

Now large: vineopt 70 000 lines of code (excl opt).

Tcl/Tk good for graphics etc, bad for computations.

Python better, many extensions, Tkinter for graphics.

# Vineopt and friends

Software made by Kaj Holmberg since 2002:

Vineopt: Visual network optimization (Tcl/Tk, C, C++, fortran)

Vileopt: Visual linear optimization (Tcl/Tk)

Nileopt: Visual nonlinear optimization (Tcl/Tk, SciPy)

Vigropt: Visual group optimization (Python, Tkinter, NumPy)

Vikeopt (Snowplan): Visual  $k$ -Chinese postman problem solver (Python, Tkinter, NumPy, SciPy, Mayavi, matplotlib, networkx)

Viseopt: Visual Sudoku help (Python, Tkinter, NumPy)

Small improvements each year/all the time/when I have the time.

Now large: vineopt 70 000 lines of code (excl opt).

Tcl/Tk good for graphics etc, bad for computations.

Python better, many extensions, Tkinter for graphics.

C good for fast computations,

# Vineopt and friends

Software made by Kaj Holmberg since 2002:

Vineopt: Visual network optimization (Tcl/Tk, C, C++, fortran)

Vileopt: Visual linear optimization (Tcl/Tk)

Nileopt: Visual nonlinear optimization (Tcl/Tk, SciPy)

Vigropt: Visual group optimization (Python, Tkinter, NumPy)

Vikeopt (Snowplan): Visual  $k$ -Chinese postman problem solver (Python, Tkinter, NumPy, SciPy, Mayavi, matplotlib, networkx)

Viseopt: Visual Sudoku help (Python, Tkinter, NumPy)

Small improvements each year/all the time/when I have the time.

Now large: vineopt 70 000 lines of code (excl opt).

Tcl/Tk good for graphics etc, bad for computations.

Python better, many extensions, Tkinter for graphics.

C good for fast computations, as is fortran.

# Dine

# Dine

Small starting program:

# Dine

Small starting program:  
dine

# Dine

Small starting program:  
dine (C)

## Dine

Small starting program:

dine (C)

Checks where it is run, especially if student.

# Dine

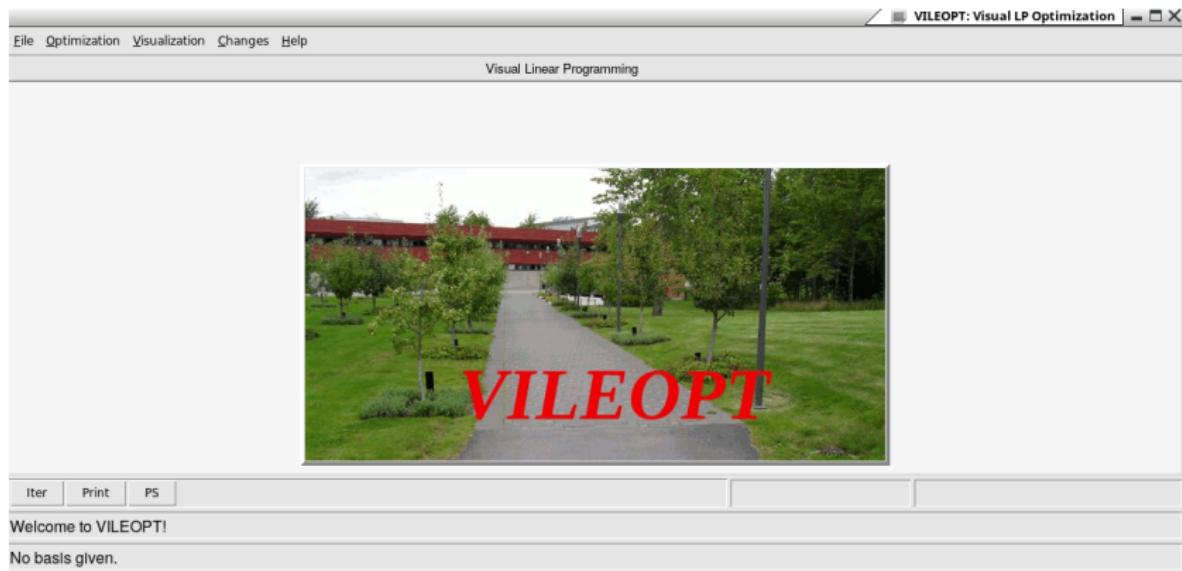
Small starting program:

dine (C)

Checks where it is run, especially if student.

```
[kaj@hohome vineopt]$ ./dine
+++++ Dine at Kaj's: Running opt labs (Vineopt, Vileopt, Nileopt) +++++
Tue May 30 11:08:13 2017
What to run?
k=0: Quit
k=1: Run Vineopt
k=2: Run Vileopt
k=3: Run Nileopt
k=4: Run Vineopt without maps
k=5: Run Vineopt for ORSS
k=6: Run Snowplan
k=7: Run Nileopt with more functions
k=8: Clean all
+100: Do not remove temporary files at exit
(Many temporary files are created, so you are advised to run dine in a separate directory.)
Give number! █
```

# Vileopt starting



Purpose: Manually controlled pivots in the simplex method.  
Create, save, read, change problem.

# Vileopt problem

LP-problem

Z =	4.0 x1	+ 3.0 x2	+ 3.0 x3	+ x4	+ x5		
s.t.	x1	+ x2	+ x3	+ x4	+ x5	+ x6	= 100.0
				5.0 x4	+ 3.0 x5	+ x7	= 20.0
	2.0 x1	+ x2	+ x3	+ 3.0 x4	+ 4.0 x5	+ x8	= 120.0
	x1,	x2,	x3,	x4,	x5,	x6,	x7, x8 >= 0

# Vileopt tableau

File Optimization Visualization Changes Help

Visual Linear Programming

```
Z = 4.0 x1 + 3.0 x2 + 3.0 x3 + x4 + x5 + x6 = 100.0
S.t.
x1 + x2 + x3 + x4 + x5 + x6 + x7 = 20.0
2.0 x1 + x2 + x3 + 3.0 x4 + 4.0 x5 + x6 + x7 + x8 = 120.0
x1, x2, x3, x4, x5, x6, x7, x8 >= 0
```

Iter Print PS Problem: ten1706. Variables: 8. Constraints: 3.

Welcome to VILEOPT!

No basis given.

# Vileopt pivot

VILEOPT: Visual LP Optimization

File Optimization Visualization Changes Help

Visual Linear Programming

Bas	z	x1	x2	x3	x4	x5	x6	x7	x8	b
z	1	-4.000	-3.000	-3.000	-1.000	-1.000	0	0	0	0
x6	0	1	1	1	1	1	0	0	0	100.000
x7	0	0	0	0	5.000	3.000	0	1	0	20.000
x8	0	2.000	1	1	3.000	4.000	0	0	1	120.000

Choose entering variable at the top and leaving variable to the left and hit pivot.

Basic variables: 6 7 8

Pivot

Iter Print PS Problem: ten1706. Variables: 8. Constraints: 3.

Welcome to VILEOPT!

Basic variables: 6 7 8.

# Vileopt iteration 2

VILEOPT: Visual LP Optimization

File Optimization Visualization Changes Help

Visual Linear Programming

Bas	z	x1	x2	x3	x4	x5	x6	x7	x8	b
z	1	0	-1.000	-1.000	5.000	7.000	0	0	2.000	240.000
x6	0	0	0.500	0.500	-0.500	-1.000	1	0	-0.500	40.000
x7	0	0	0	0	5.000	3.000	0	1	0	20.000
x1	0	1	0.500	0.500	1.500	2.000	0	0	0.500	60.000

Choose entering variable at the top and leaving variable to the left and hit pivot.

Basic variables: 6 7 1

Pivot

Iter Print PS Problem: ten1706. Variables: 8. Constraints: 3.

Welcome to VILEOPT!

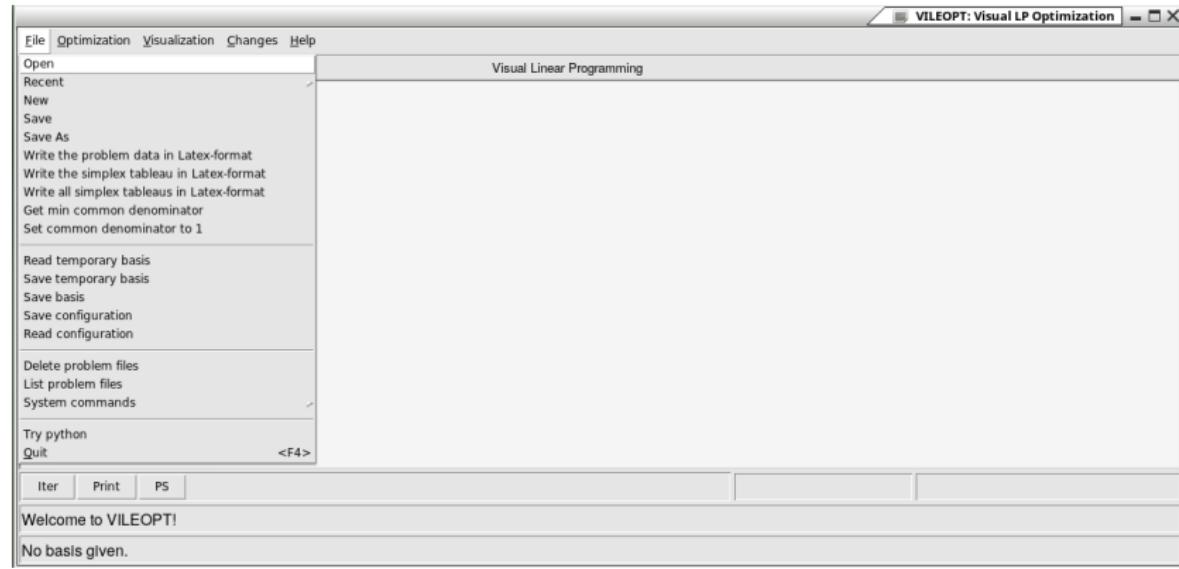
Basic variables: 6 7 1.

# Vileopt tabular change

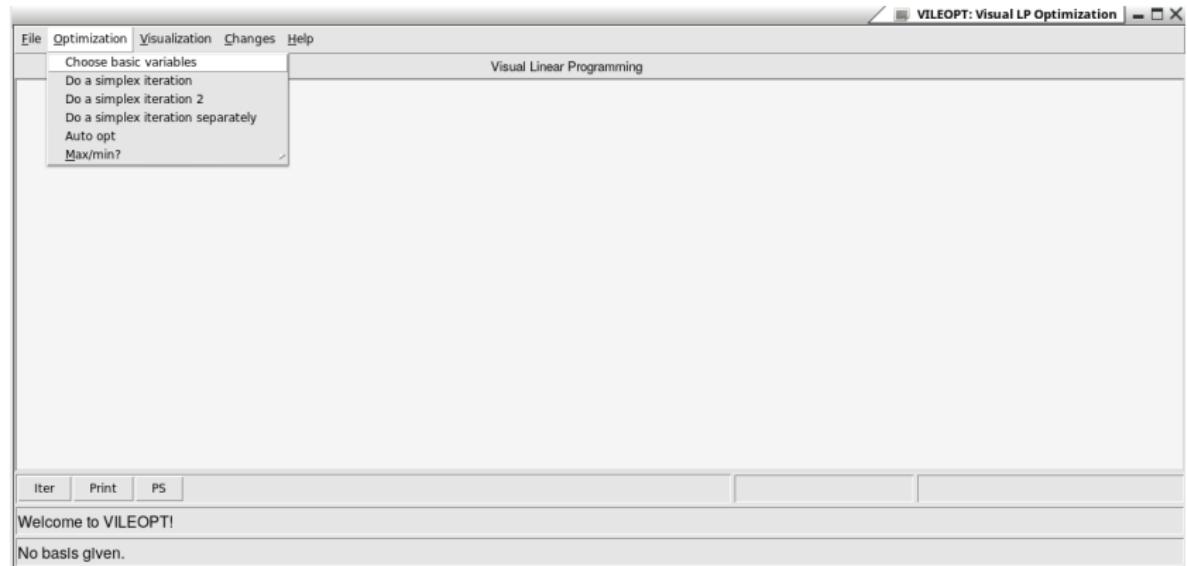
Problem data

Objective function							
c1: 4.0	c2: 3.0	c3: 3.0	c4: 1.0	c5: 1.0	c6: 0.0	c7: 0.0	c8: 0.0
Constraint coefficients							
a11: 1.0	a12: 1.0	a13: 1.0	a14: 1.0	a15: 1.0	a16: 1.0	a17: 0.0	a18: 0.0
a21: 0.0	a22: 0.0	a23: 0.0	a24: 5.0	a25: 3.0	a26: 0.0	a27: 1.0	a28: 0.0
a31: 2.0	a32: 1.0	a33: 1.0	a34: 3.0	a35: 4.0	a36: 0.0	a37: 0.0	a38: 1.0
Right-hand-side							
b1: 100.0	b2: 20.0	b3: 120.0					
<input type="button" value="Exit (no save) &lt;F3&gt;"/>				<input type="button" value="Save and exit"/>			

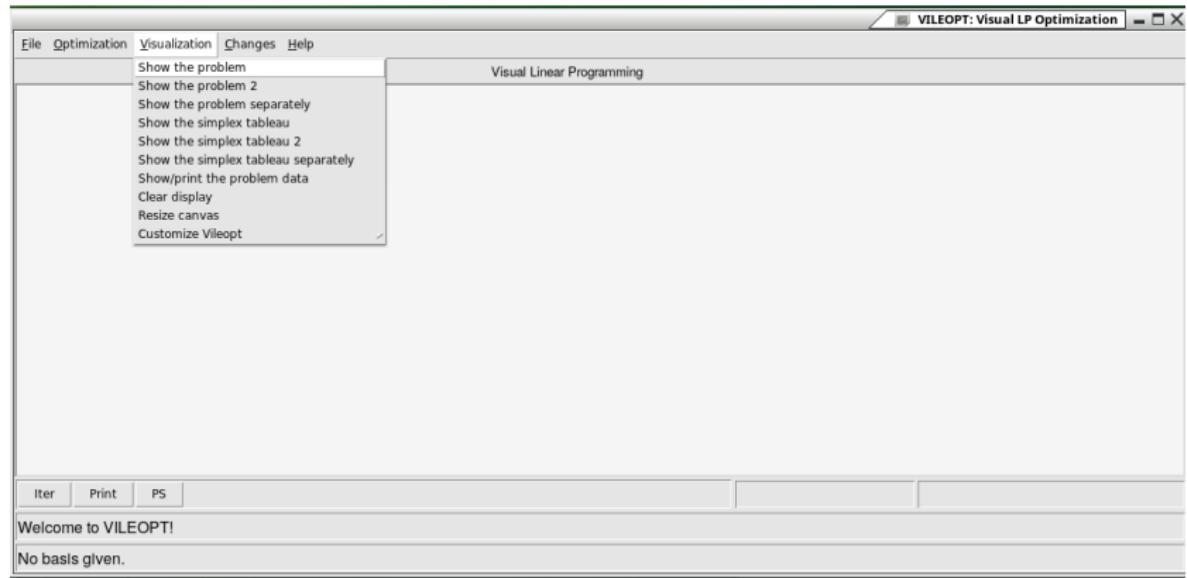
# Vileopt menu 1



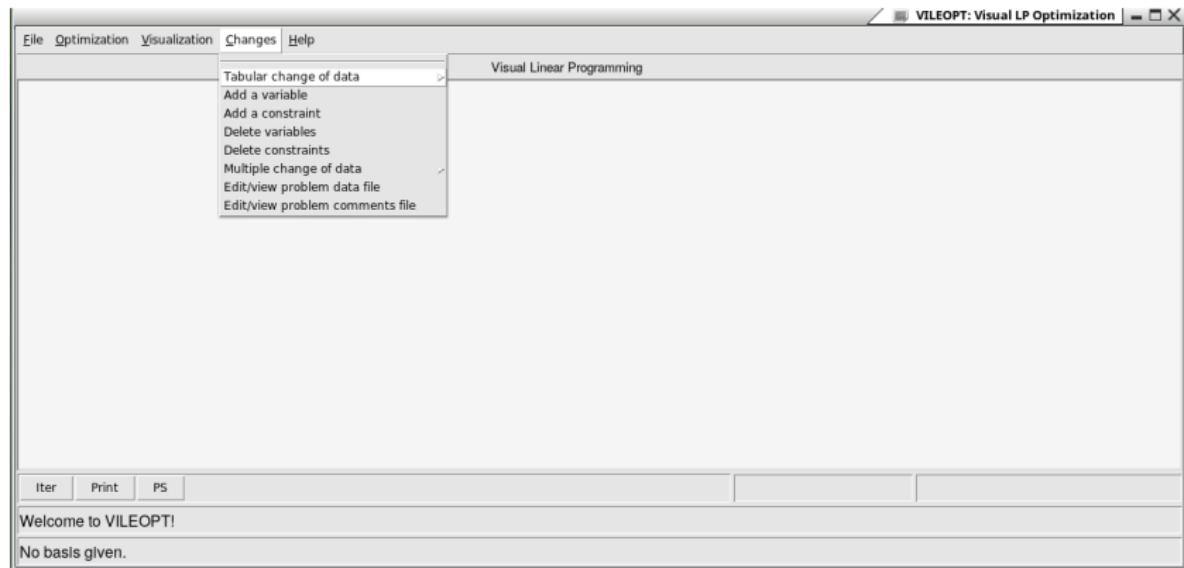
# Vileopt menu 2



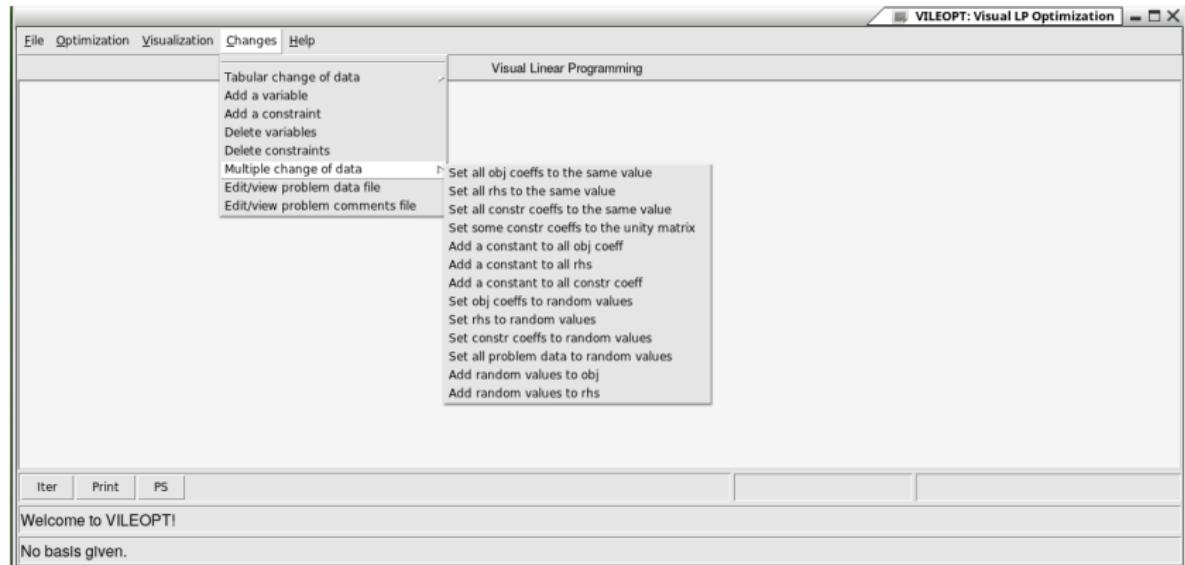
# Vileopt menu 3



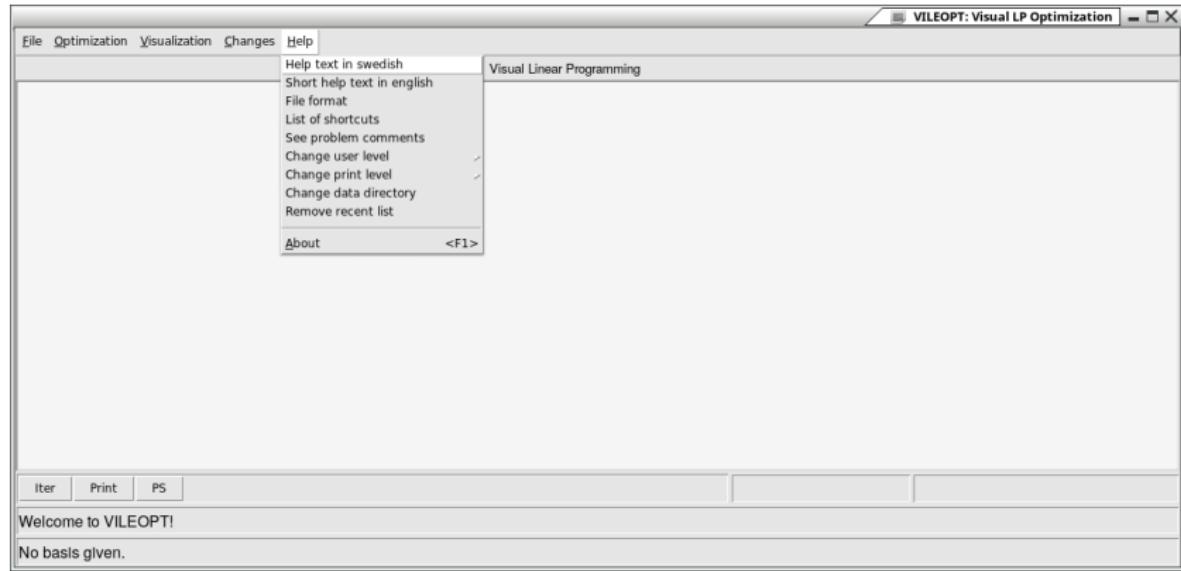
# Vileopt menu 4



# Vileopt menu 4b



# Vileopt menu 5



# Nileopt starting

NILEOPT: Nonlinear Optimization |     X

File Optimization Visualization Graphic optimization My functions Help

Nonlinear Programming

Move view of function

3D+ Up +

Left Center Right

Down

Zoom in Turn Zoom out

View angle 1

Edit Step Run Start point Clear plot Function Method Cobyla Variables: 0. Constraints: 0.

Welcome to NILEOPT!

No starting point given. Value: - . Method: Steepest descent. Good line search.

# Nileopt purpose

Illustrate solution of nonlinear optimization problems.

# Nileopt purpose

Illustrate solution of nonlinear optimization problems.

Create, read save, change problems.

# Nileopt purpose

Illustrate solution of nonlinear optimization problems.

Create, read save, change problems. Simplified AMPL-syntax.

# Nileopt purpose

Illustrate solution of nonlinear optimization problems.

Create, read save, change problems. Simplified AMPL-syntax.

Solve constrained problem (with Cobyla/Scipy).

# Nileopt purpose

Illustrate solution of nonlinear optimization problems.

Create, read save, change problems. Simplified AMPL-syntax.

Solve constrained problem (with Cobyla/Scipy).

Step through solution of unconstrained problems

# Nileopt purpose

Illustrate solution of nonlinear optimization problems.

Create, read save, change problems. Simplified AMPL-syntax.

Solve constrained problem (with Cobyla/Scipy).

Step through solution of unconstrained problems  
with many methods

# Nileopt purpose

Illustrate solution of nonlinear optimization problems.

Create, read save, change problems. Simplified AMPL-syntax.

Solve constrained problem (with Cobyla/Scipy).

Step through solution of unconstrained problems

with many methods (steepest descent, Newton, several quasi-Newton,  
several conjugate gradient).

# Nileopt purpose

Illustrate solution of nonlinear optimization problems.

Create, read save, change problems. Simplified AMPL-syntax.

Solve constrained problem (with Cobyla/Scipy).

Step through solution of unconstrained problems

with many methods (steepest descent, Newton, several quasi-Newton,  
several conjugate gradient).

Compare the methods visually.

# Nileopt purpose

Illustrate solution of nonlinear optimization problems.

Create, read save, change problems. Simplified AMPL-syntax.

Solve constrained problem (with Cobyla/Scipy).

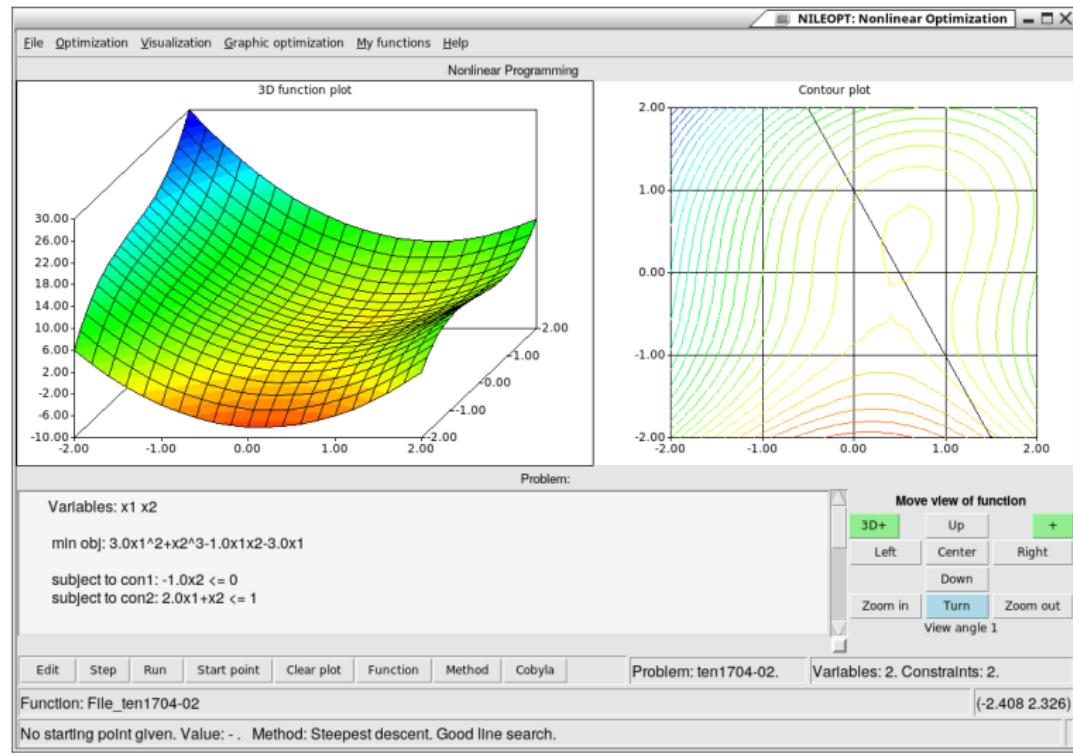
Step through solution of unconstrained problems

with many methods (steepest descent, Newton, several quasi-Newton,  
several conjugate gradient).

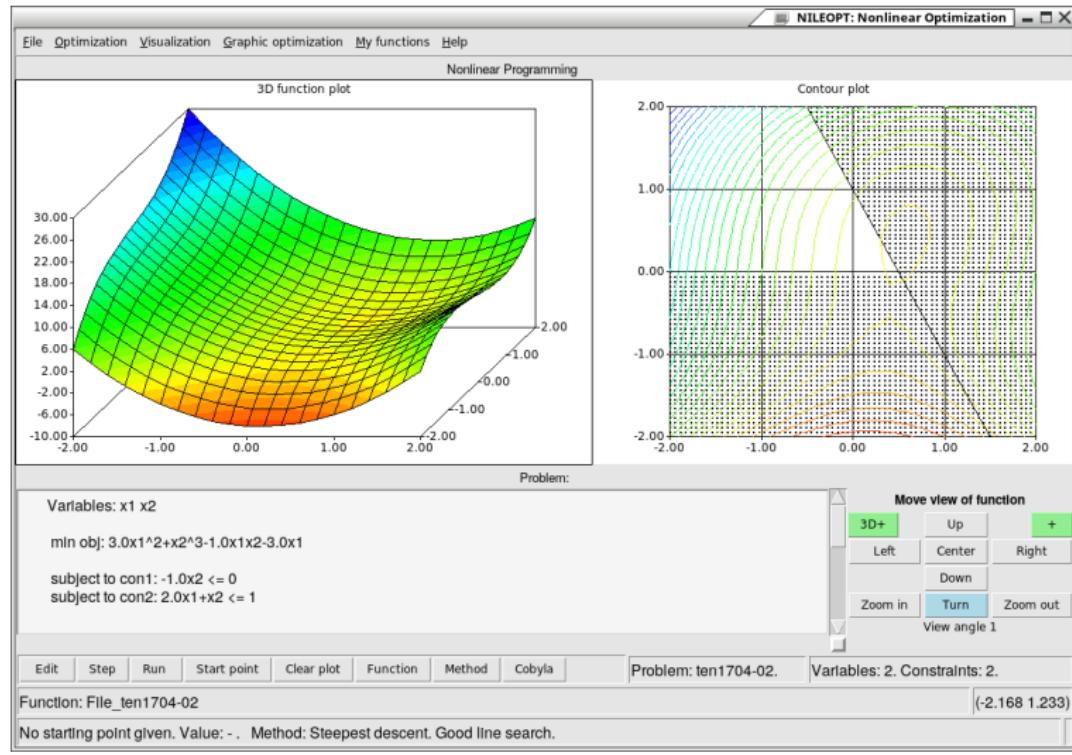
Compare the methods visually.

Show contours and 3D-image.

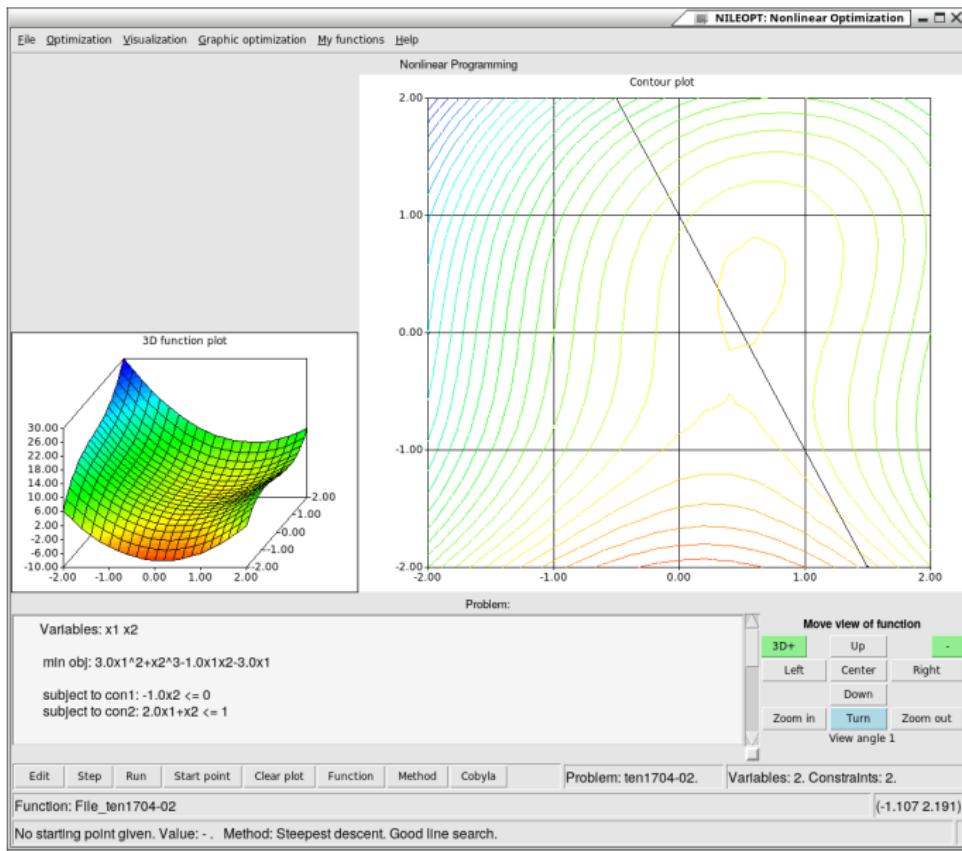
# Nileopt



# Nileopt feasible set



# Nileopt layout 2



Problem:

Variables:  $x_1 \ x_2$

min obj:  $3.0x_1^2 + x_2^3 - 1.0x_1x_2 - 3.0x_1$

subject to con1:  $-1.0x_2 \leq 0$

subject to con2:  $2.0x_1 + x_2 \leq 1$

Move view of function

3D+ Up -

Left Center Right

Down

Zoom in Turn Zoom out

View angle 1

Edit

Step

Run

Start point

Clear plot

Function

Method

Cobyla

Problem: ten1704-02.

Variables: 2. Constraints: 2.

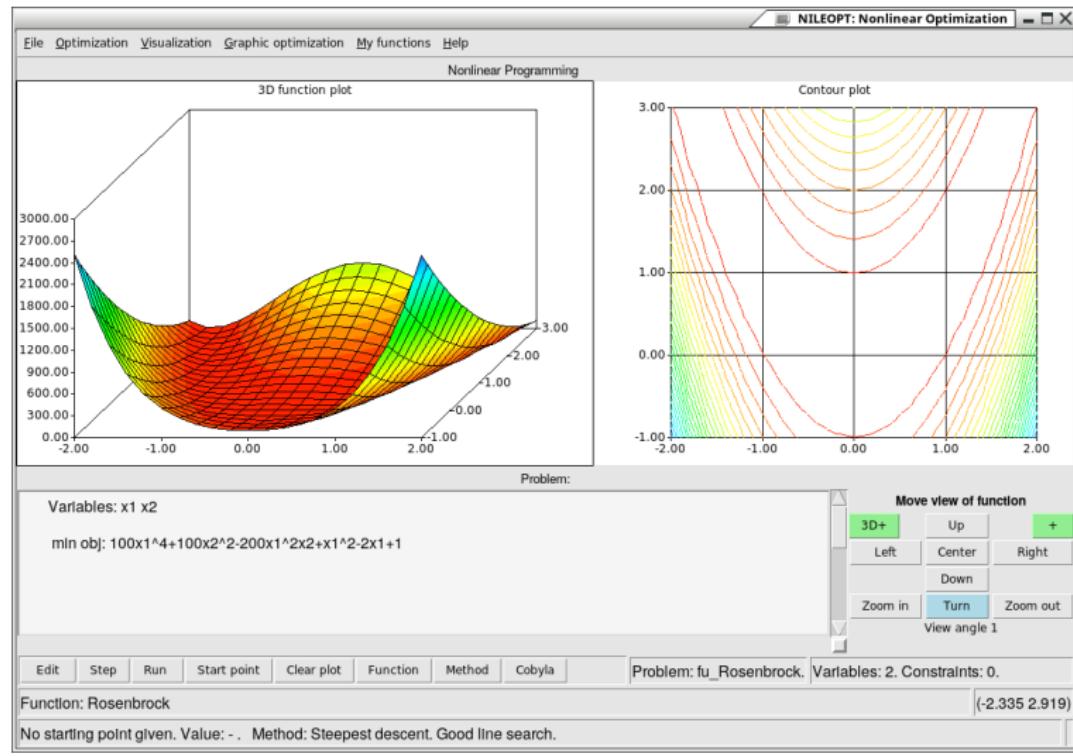
Function: File\_ten1704-02

(-1.107 2.191)

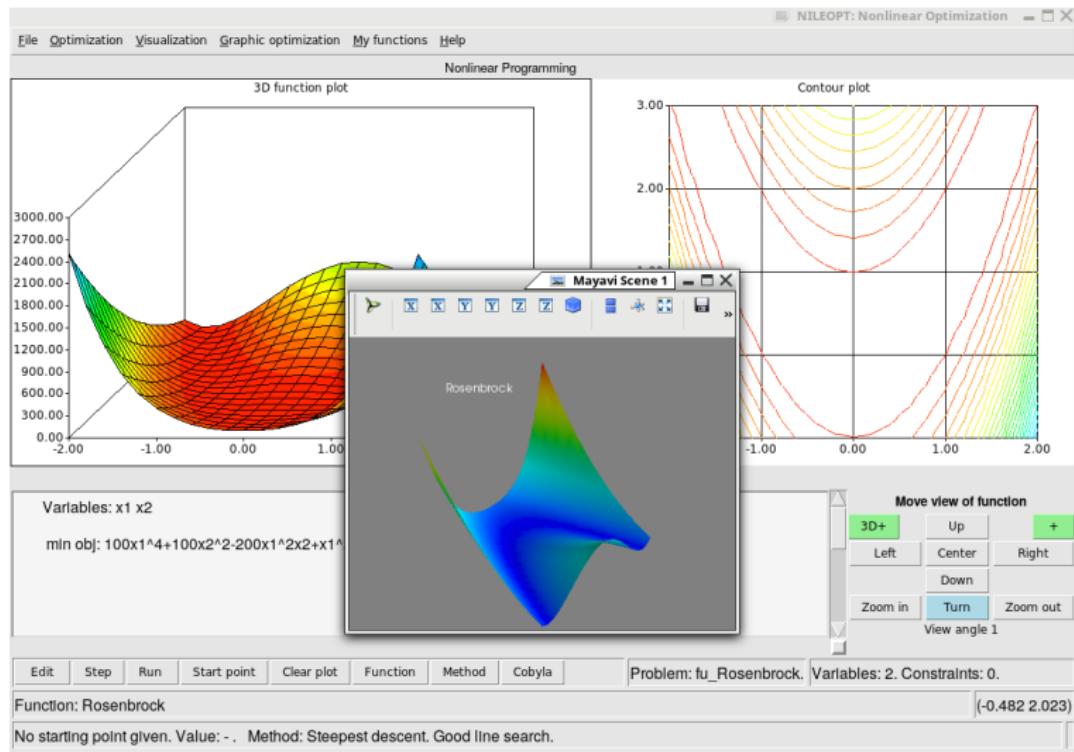
No starting point given. Value: -, Method: Steepest descent. Good line search.



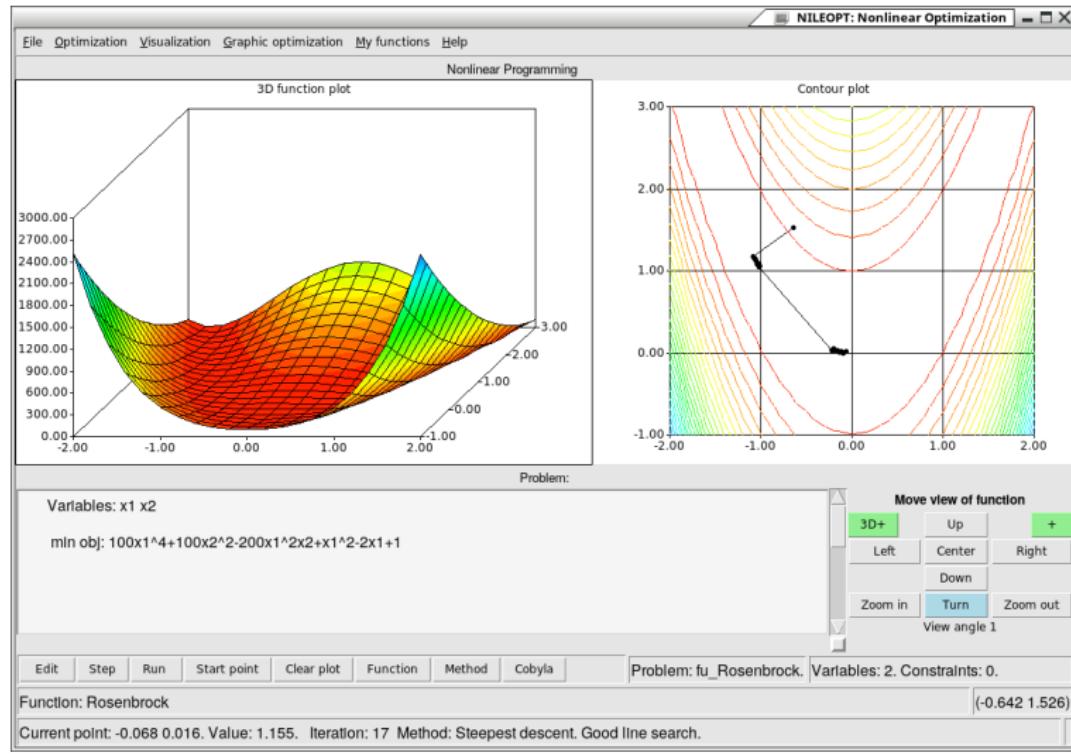
# Nileopt: Rosenbrock



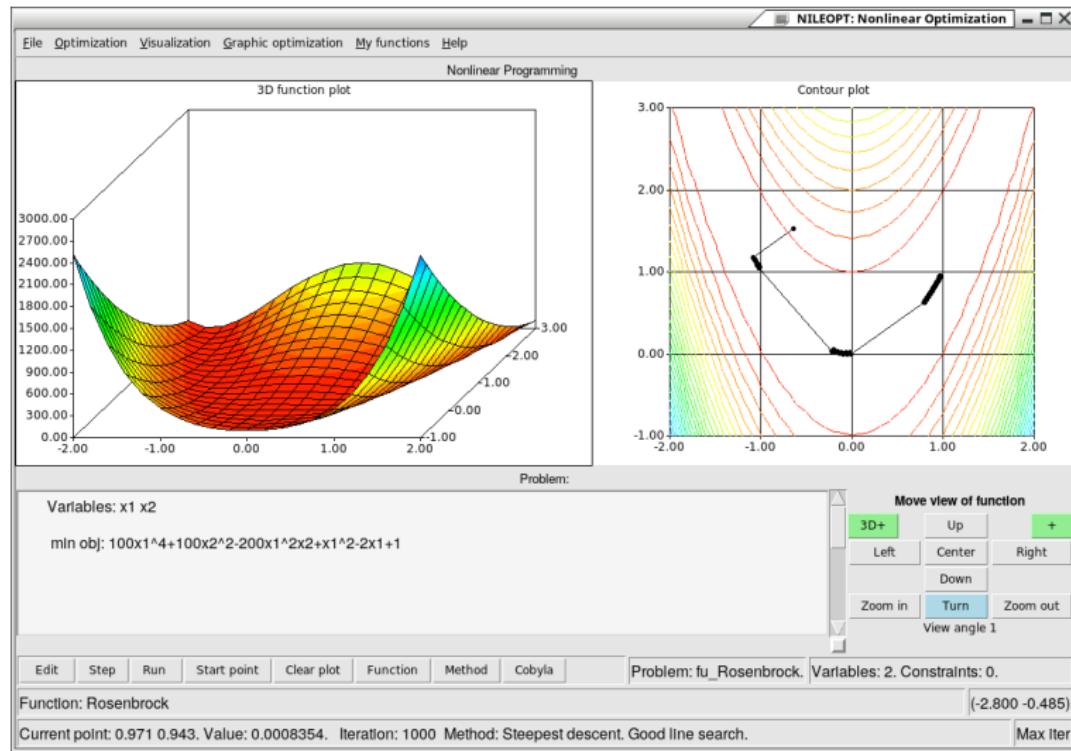
# Nileopt: Rosenbrock - 3D view



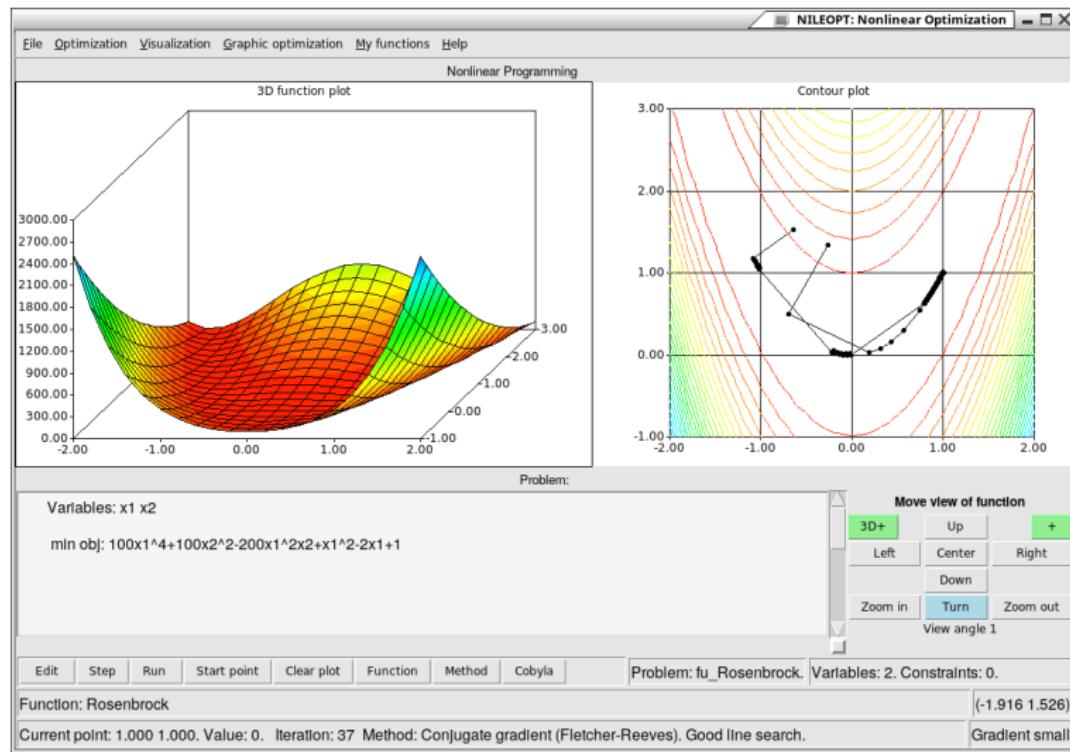
# Nileopt: Rosenbrock: Steepest descent



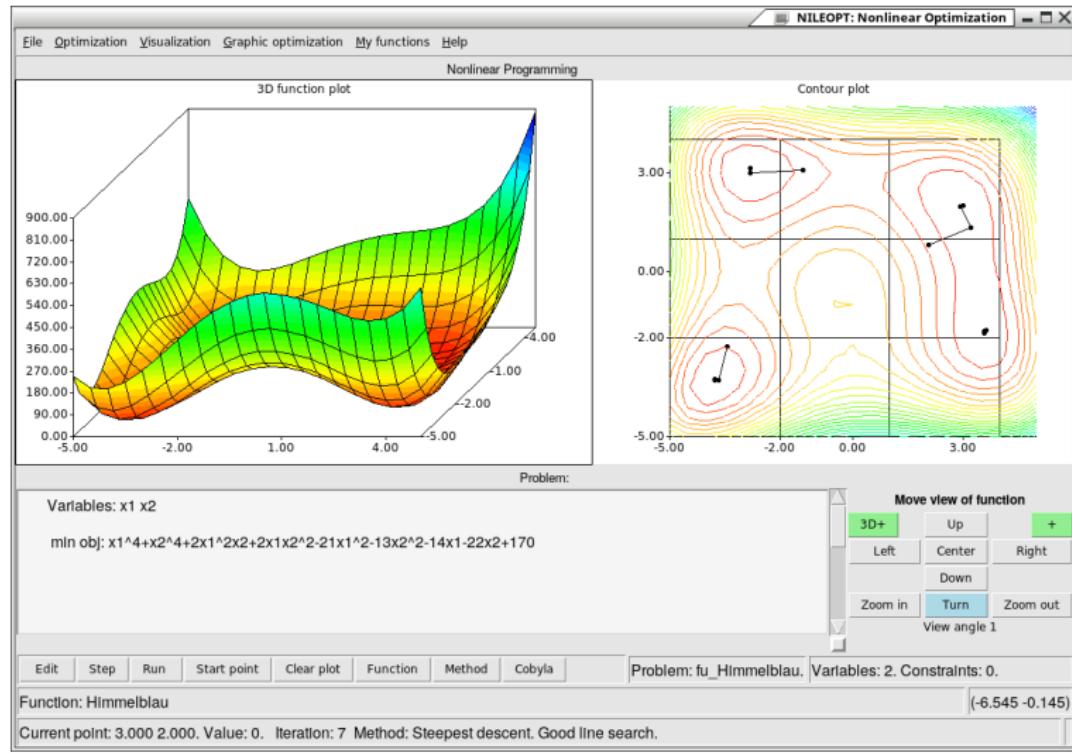
# Nileopt: Rosenbrock: Steepest descent



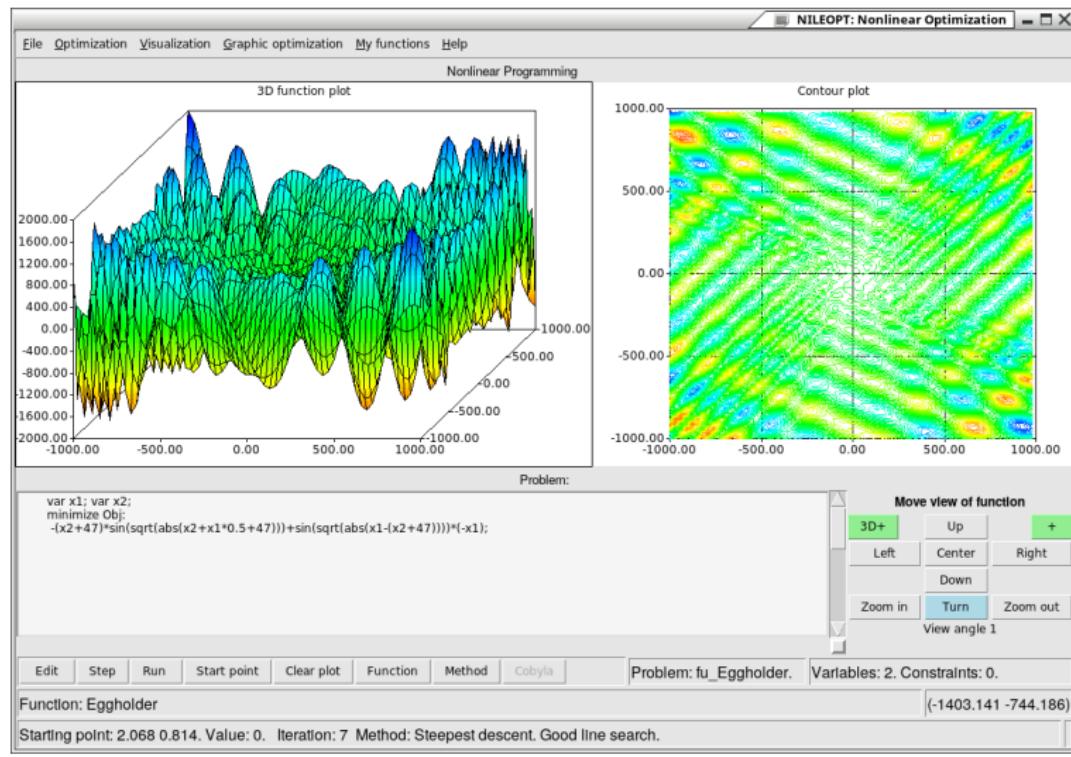
# Nileopt: Rosenbrock: Steepest descent - conjugate gradient



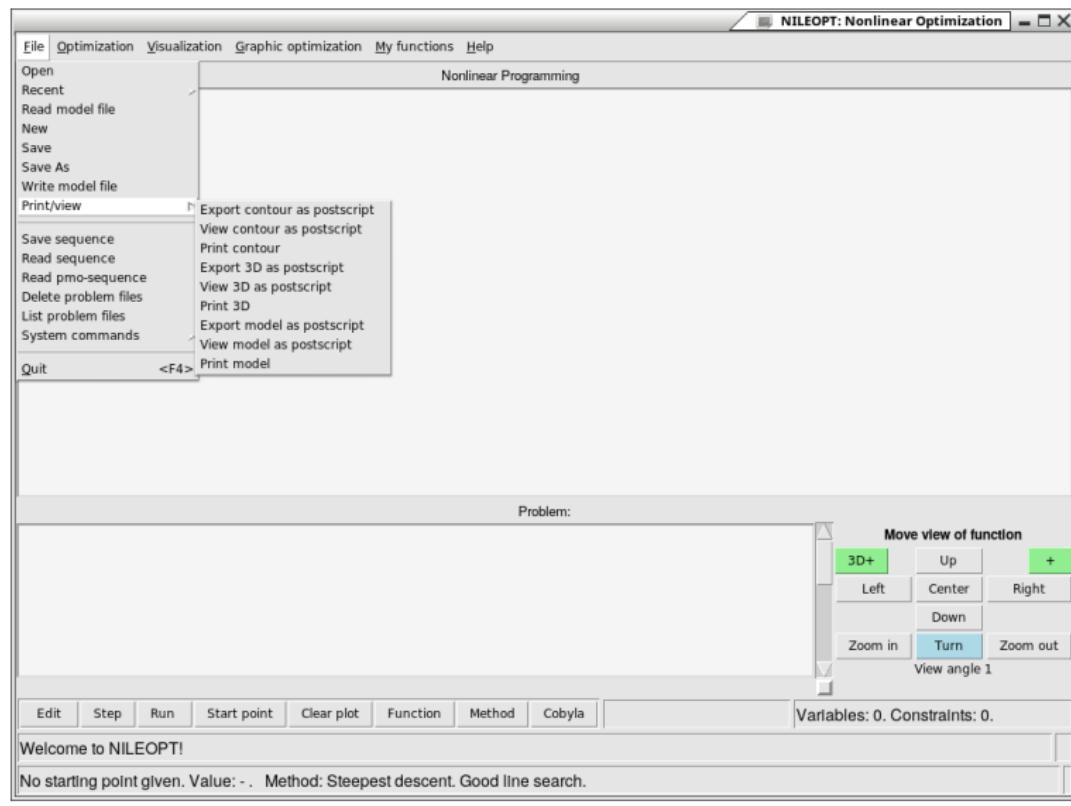
# Nileopt: Himmelblau, local optima



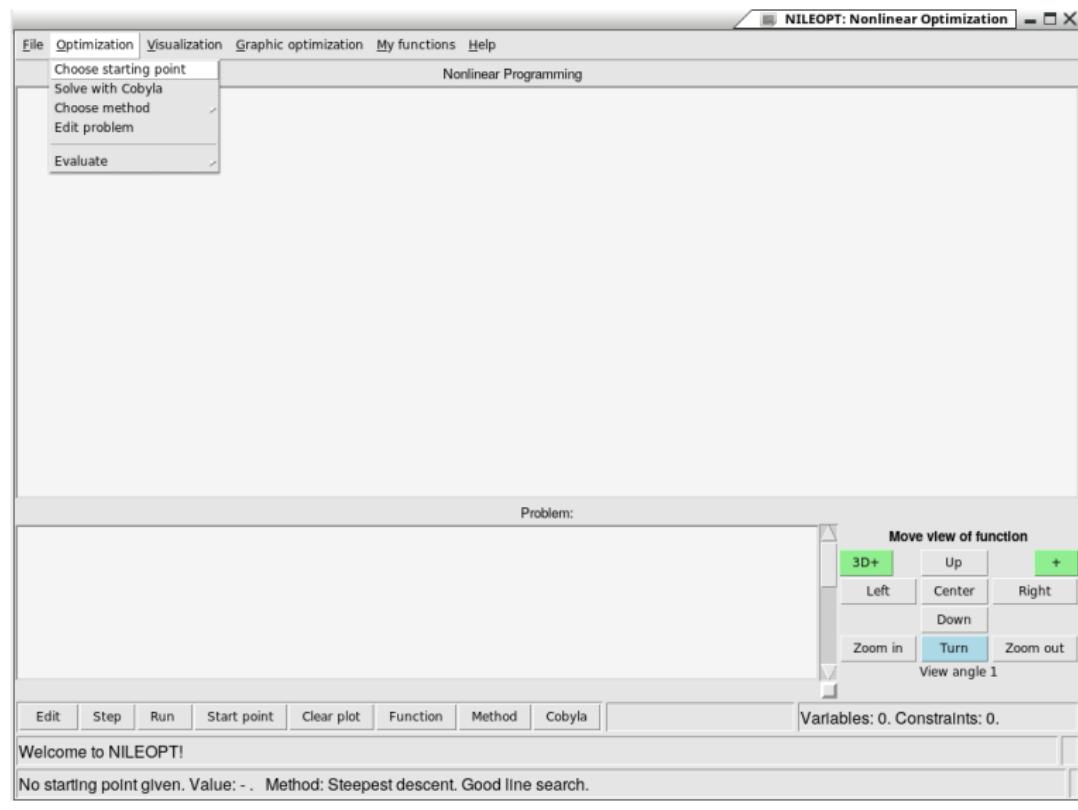
# Nileopt: Eggholder



# Nileopt menu 1



# Nileopt menu 2



# Nileopt menu 2b

NILEOPT: Nonlinear Optimization

File Optimization Visualization Graphic optimization My functions Help

Choose starting point Solve with Cobyla Choose method Edit problem

Evaluate

- Evaluate objective function gradient in current point
- Evaluate objective function Hessian in current point
- Get steepest descent direction in current point
- Get Newton direction in current point

- Evaluate current solution point in model
- Evaluate starting point in model

- Evaluate objective function gradient in starting point
- Evaluate objective function Hessian in starting point
- Get steepest descent direction in starting point
- Get Newton direction in starting point

- Evaluate starting point in given function
- Evaluate starting point in given gradient
- Evaluate starting point in given Hessian

Problem:

Move view of function

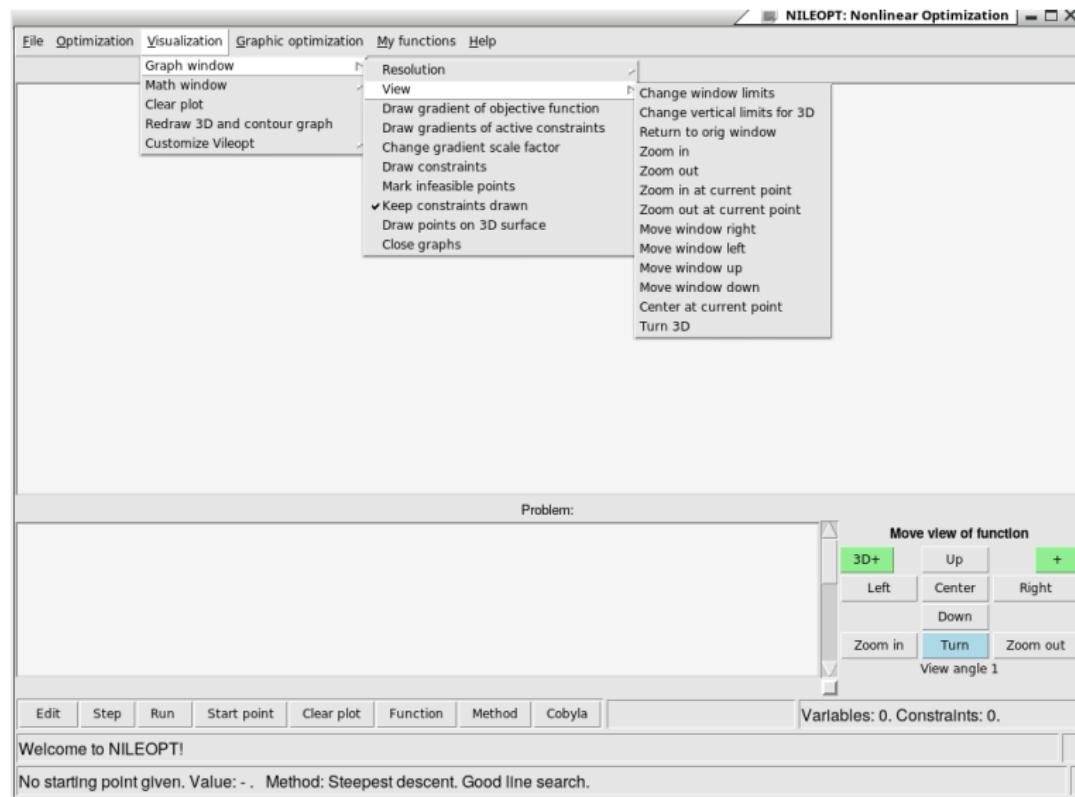
3D+	Up	+
Left	Center	Right
Down		
Zoom in	Turn	Zoom out
View angle 1		

Edit Step Run Start point Clear plot Function Method Cobyla Variables: 0. Constraints: 0.

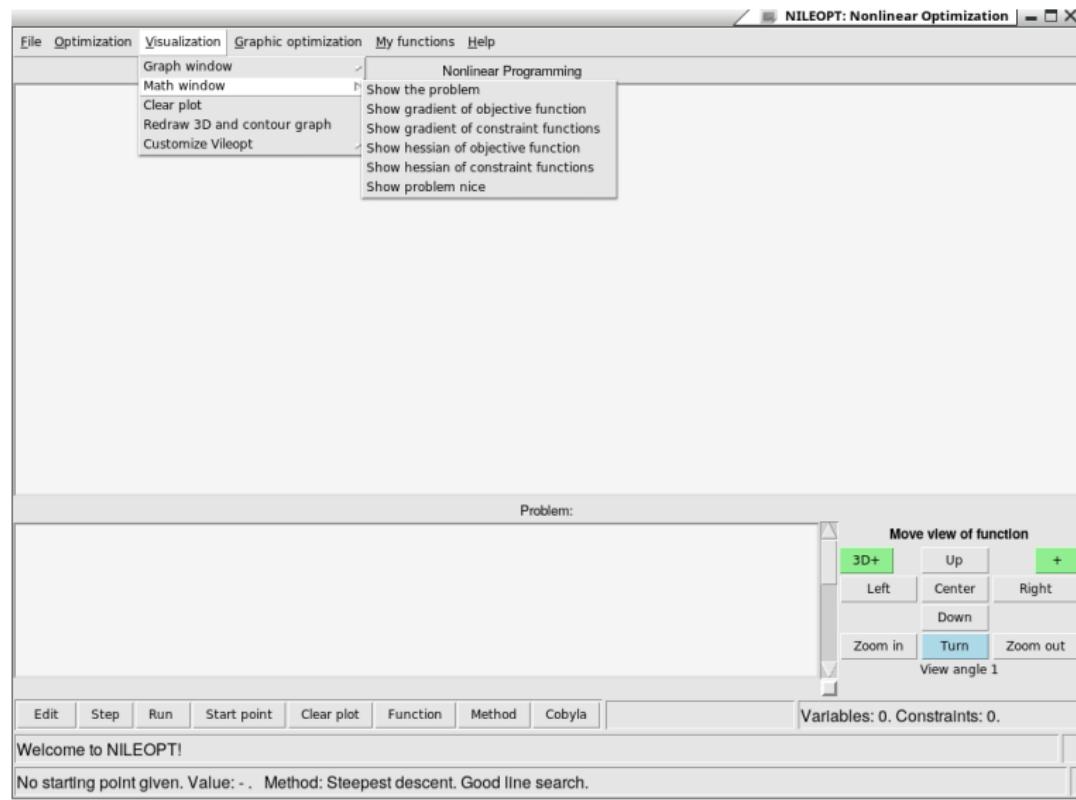
Welcome to NILEOPT!

No starting point given. Value: - . Method: Steepest descent. Good line search.

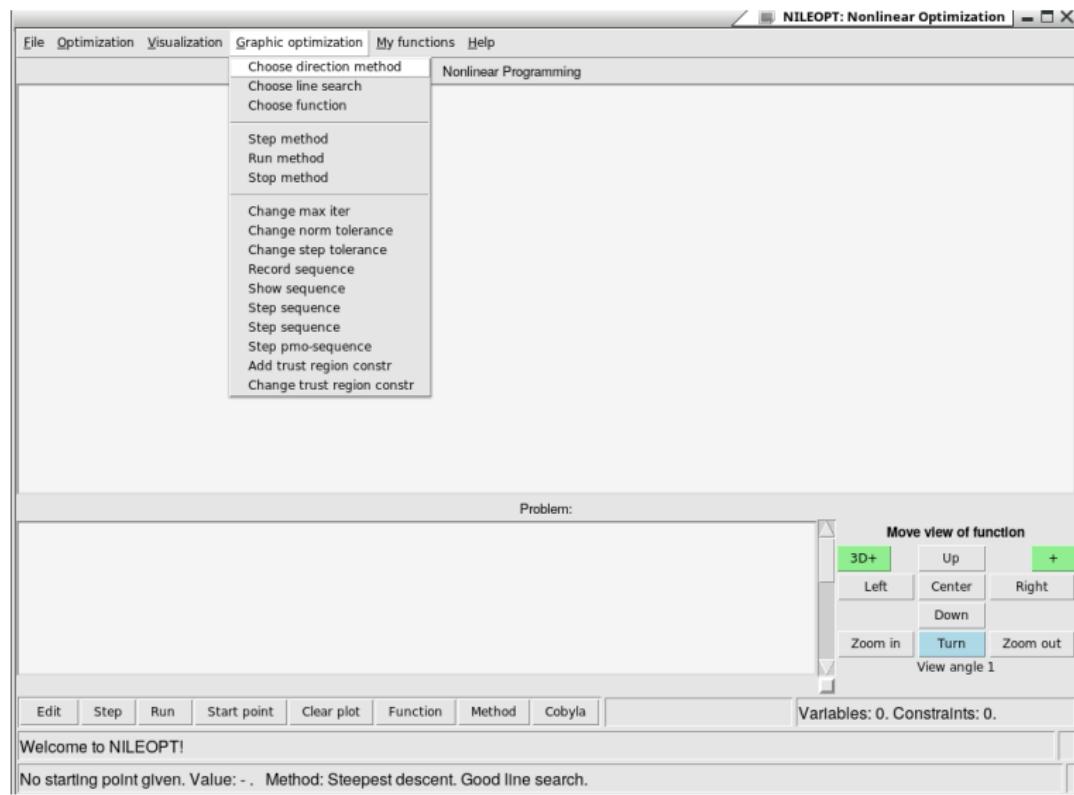
# Nileopt menu 3a



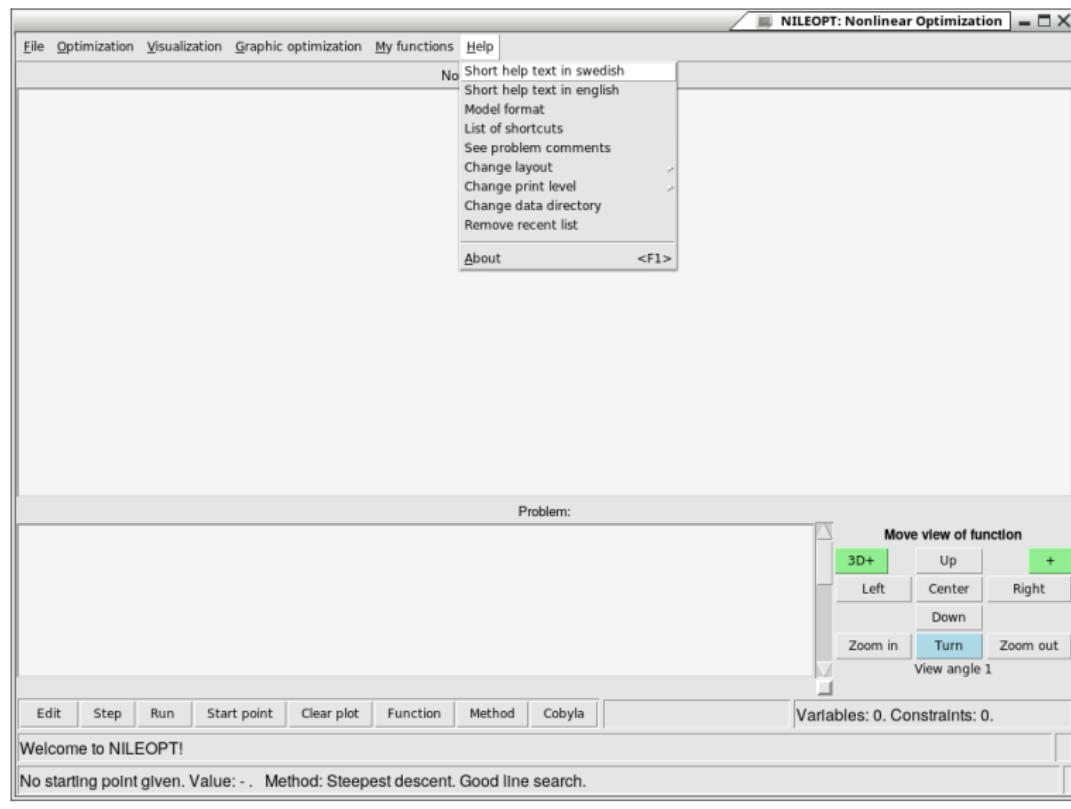
# Nileopt menu 3b



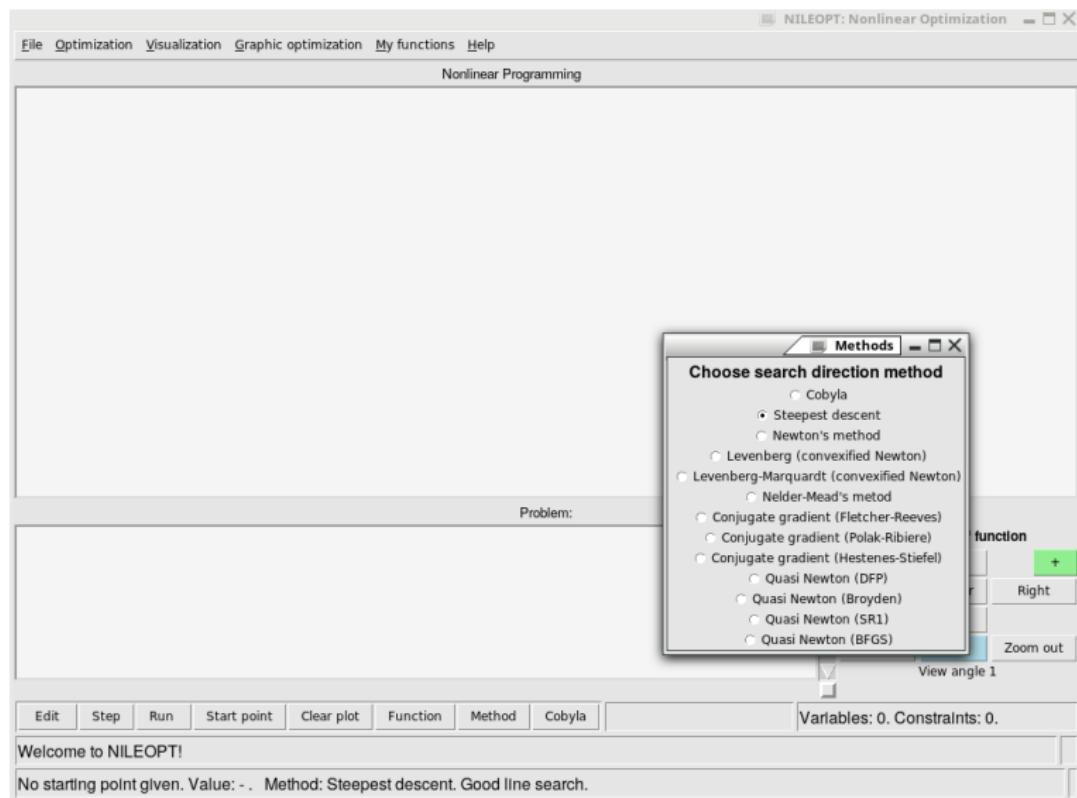
# Nileopt menu 4



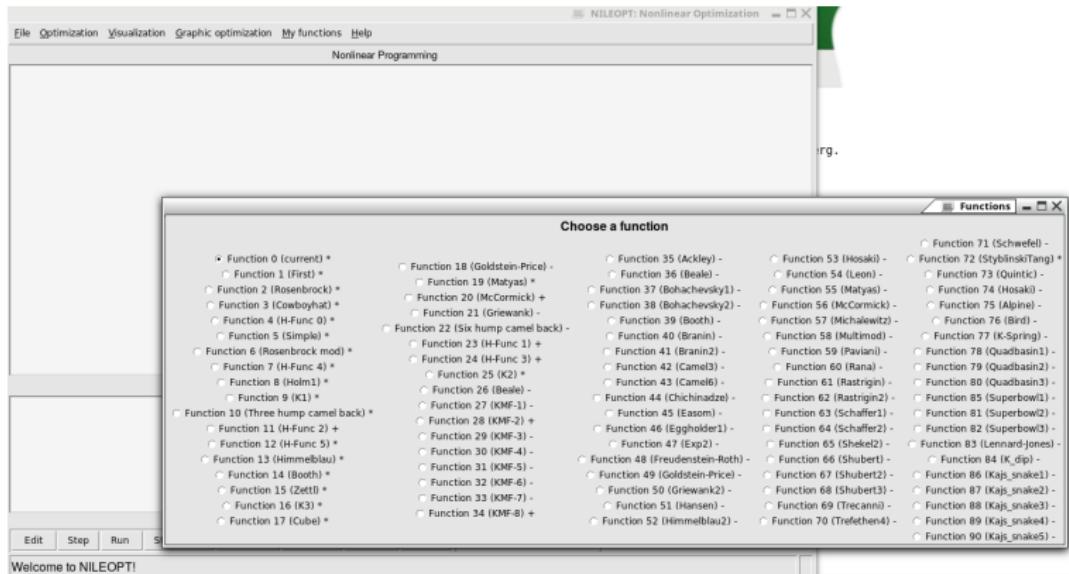
# Nileopt menu 5



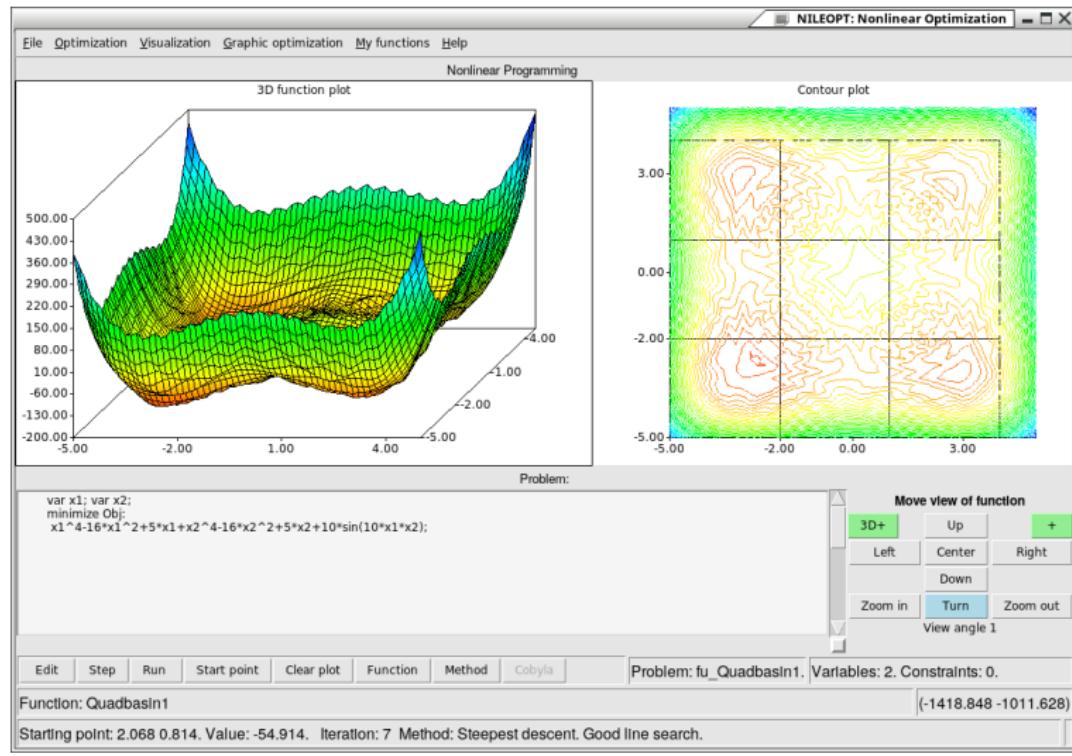
# Nileopt menu 6



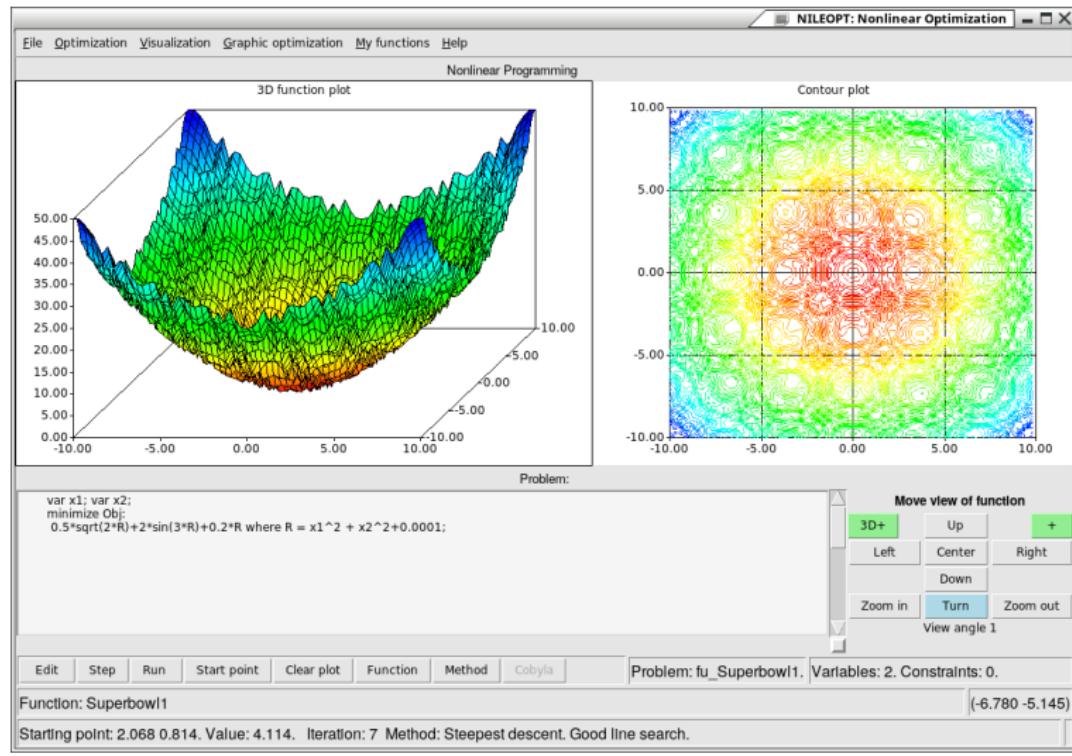
# Nileopt menu 7



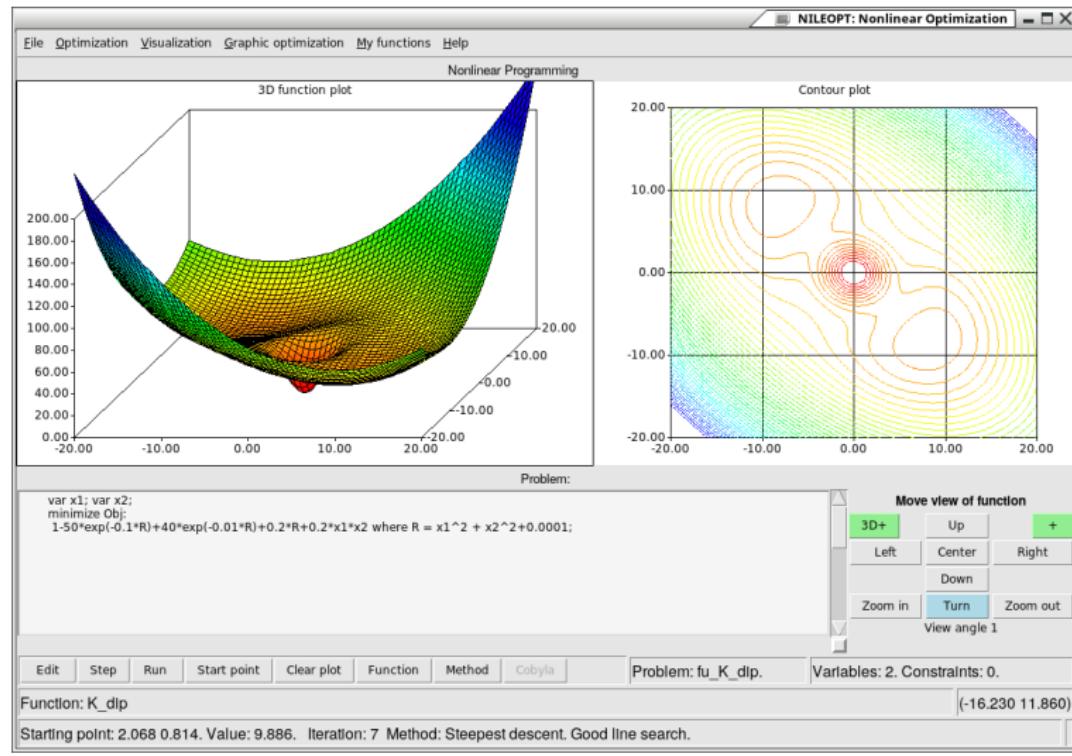
# Nileopt: Quadbasin



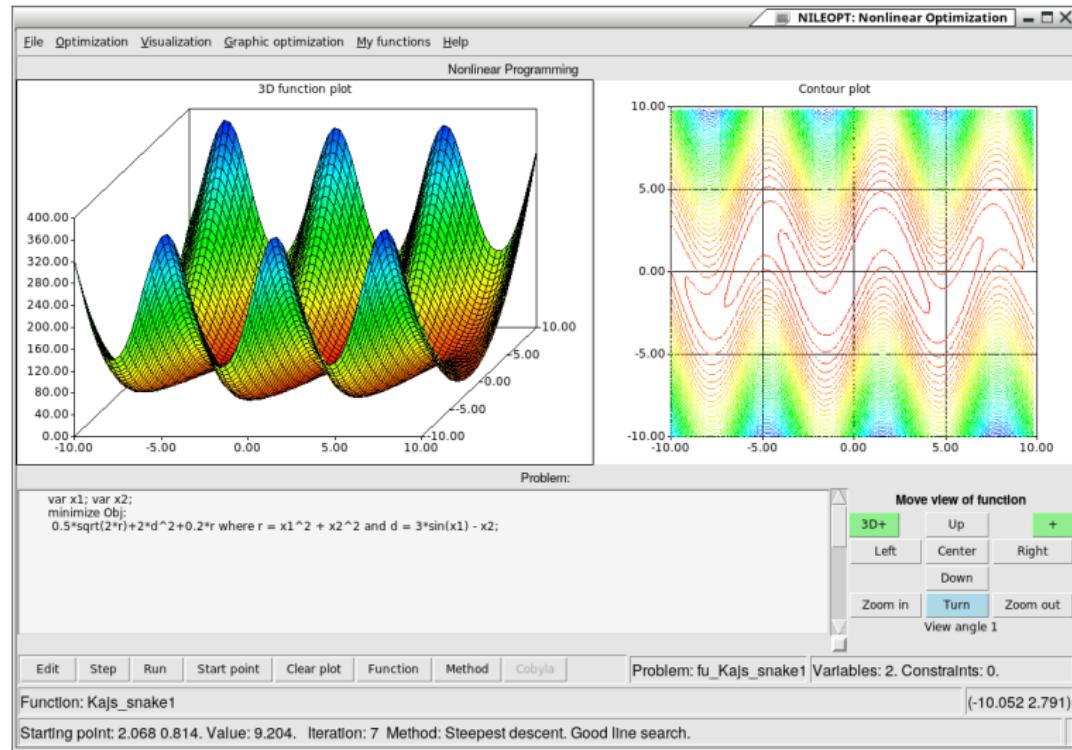
# Nileopt: Superbowl



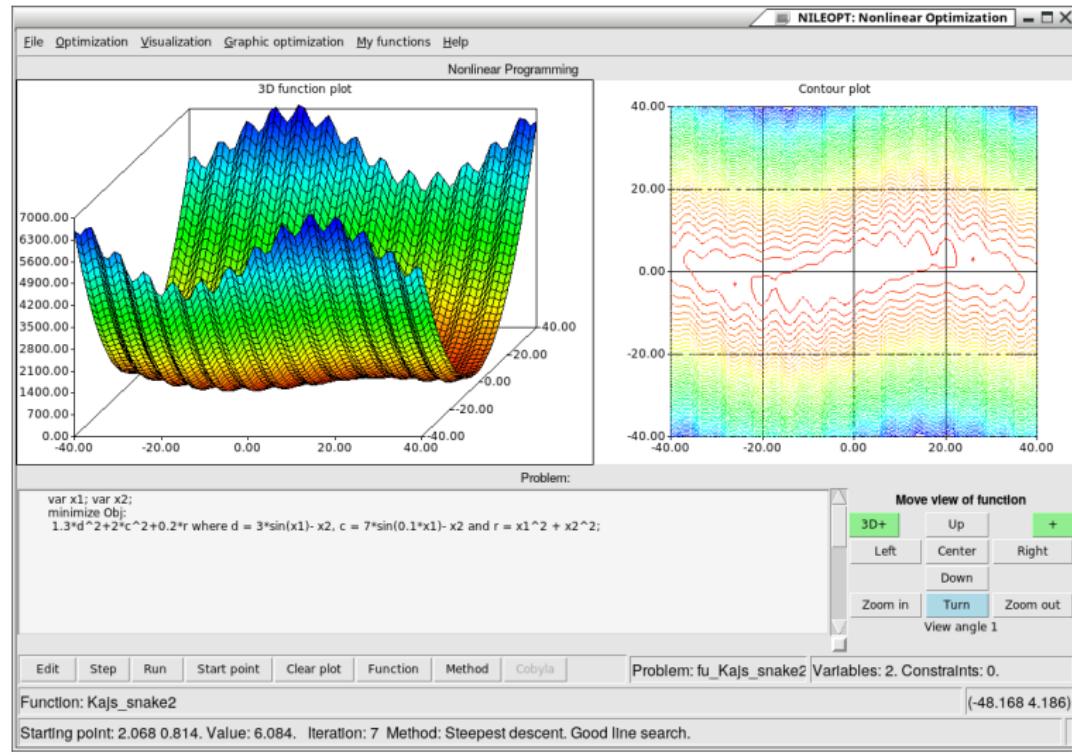
# Nileopt: Dip



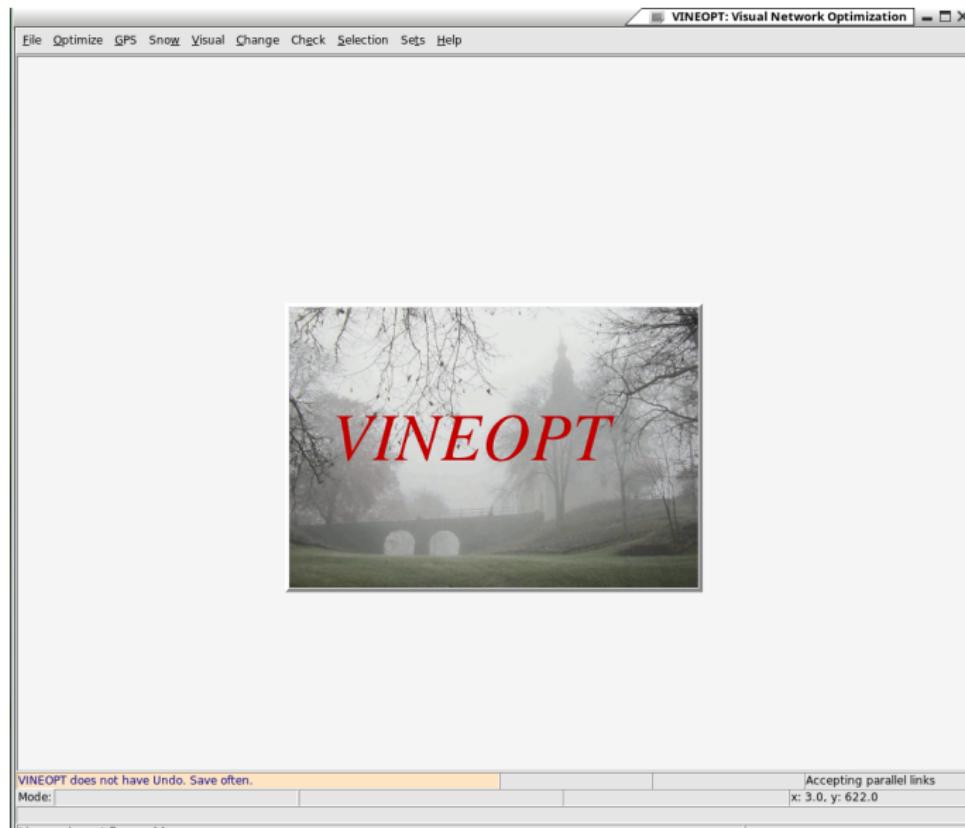
# Nileopt: Kaj's snake 1



# Nileopt: Kaj's snake 2



# Vineopt starting



# Vineopt purpose

Create, save, read, change network problems

## Vineopt purpose

Create, save, read, change network problems (of different types).

# Vineopt purpose

Create, save, read, change network problems (of different types).  
Solve network optimization problems

## Vineopt purpose

Create, save, read, change network problems (of different types).  
Solve network optimization problems (of many types).

## Vineopt purpose

- Create, save, read, change network problems (of different types).
- Solve network optimization problems (of many types).
- View the graph and solution

## Vineopt purpose

- Create, save, read, change network problems (of different types).
- Solve network optimization problems (of many types).
- View the graph and solution (in many different ways).

## Vineopt purpose

- Create, save, read, change network problems (of different types).
- Solve network optimization problems (of many types).
- View the graph and solution (in many different ways).
- Zoom, move picture.

## Vineopt purpose

- Create, save, read, change network problems (of different types).
- Solve network optimization problems (of many types).
- View the graph and solution (in many different ways).
- Zoom, move picture.
- Show OpenStreetMap as background.

## Vineopt purpose

Create, save, read, change network problems (of different types).

Solve network optimization problems (of many types).

View the graph and solution (in many different ways).

Zoom, move picture.

Show OpenStreetMap as background.

Show node/link sets with colours.

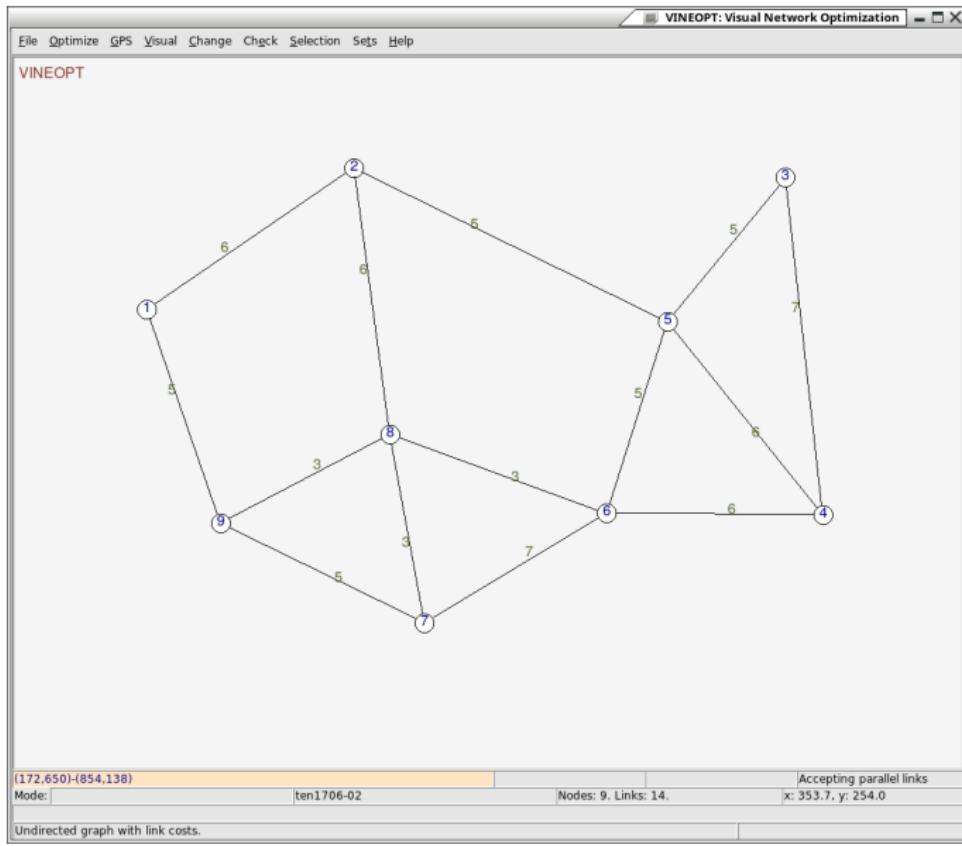
## Vineopt purpose

- Create, save, read, change network problems (of different types).
- Solve network optimization problems (of many types).
- View the graph and solution (in many different ways).
- Zoom, move picture.
- Show OpenStreetMap as background.
- Show node/link sets with colours.
- Read and show GPS-tracks.

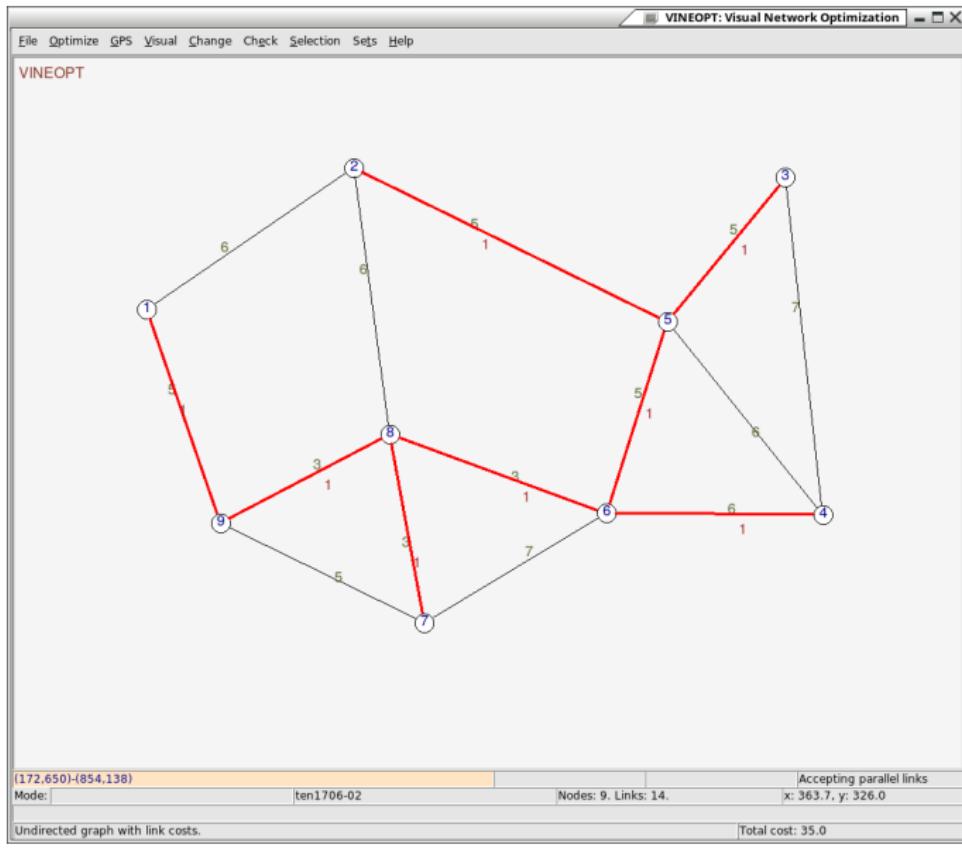
## Vineopt purpose

- Create, save, read, change network problems (of different types).
- Solve network optimization problems (of many types).
- View the graph and solution (in many different ways).
- Zoom, move picture.
- Show OpenStreetMap as background.
- Show node/link sets with colours.
- Read and show GPS-tracks.
- Used for research in IP telecom and snow removal.

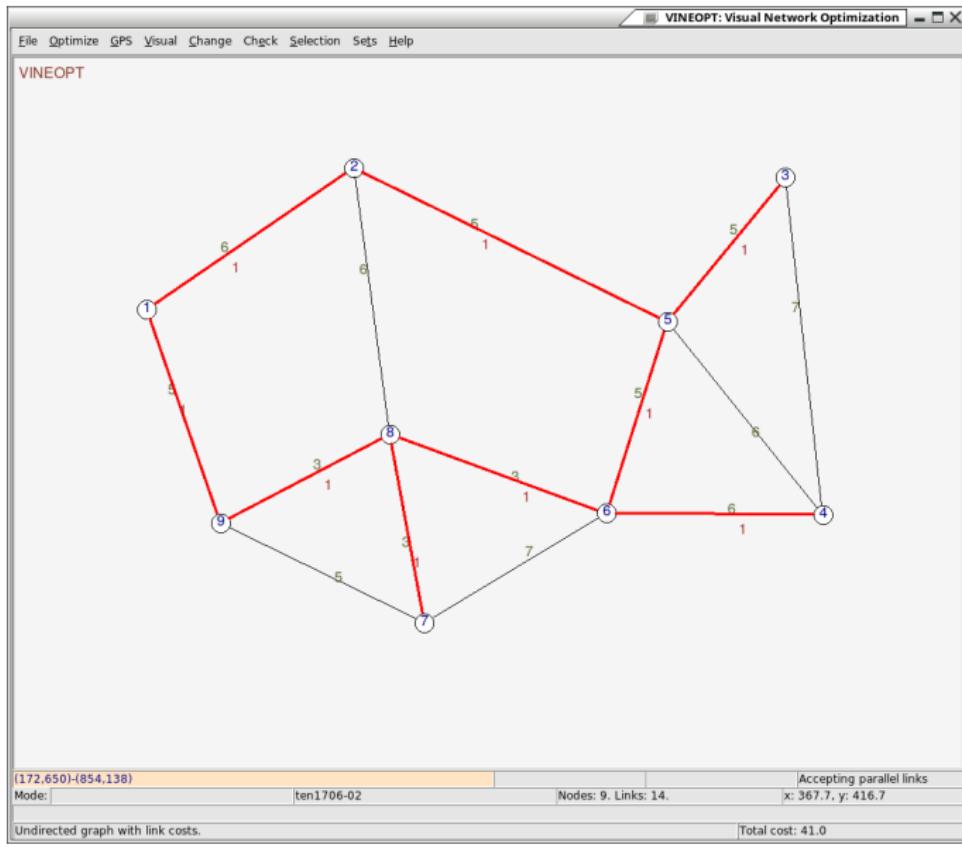
# Vineopt: undirected graph



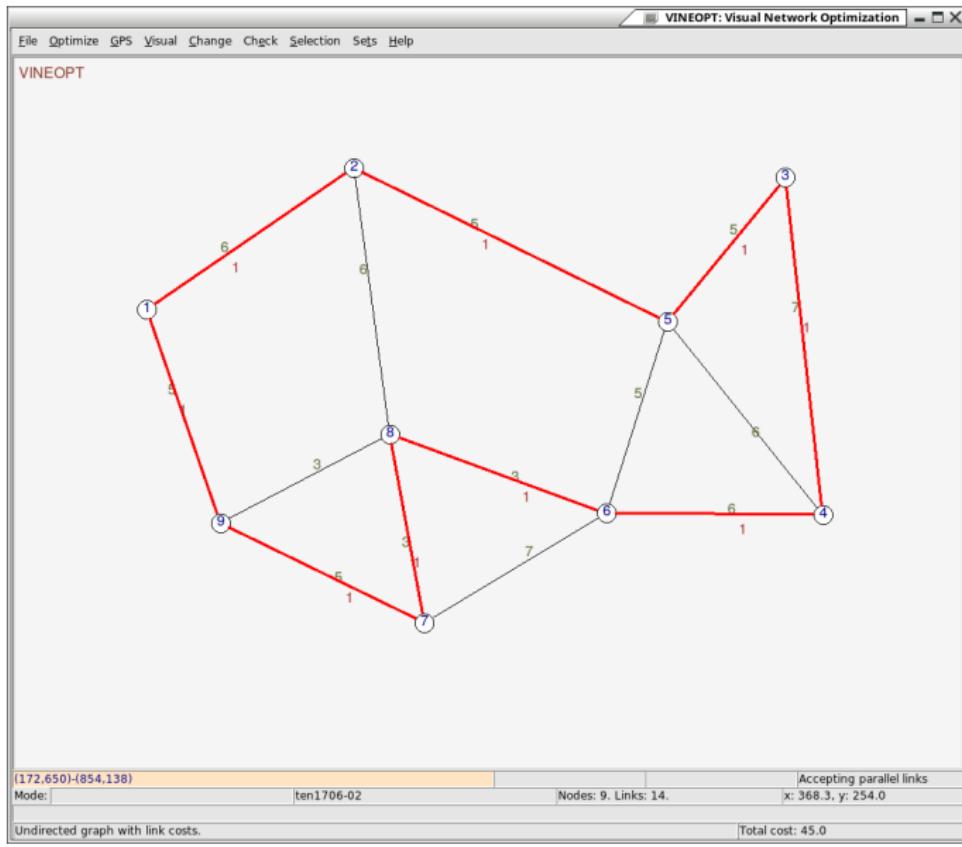
# Vineopt: MST



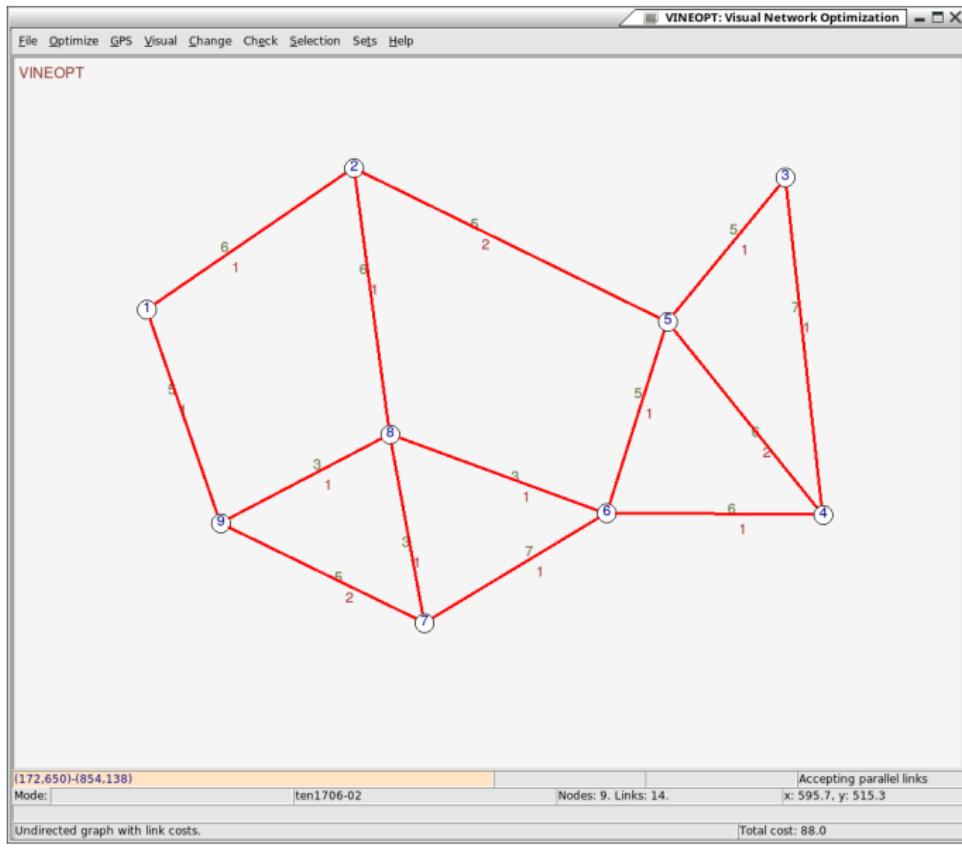
# Vineopt: 1-tree



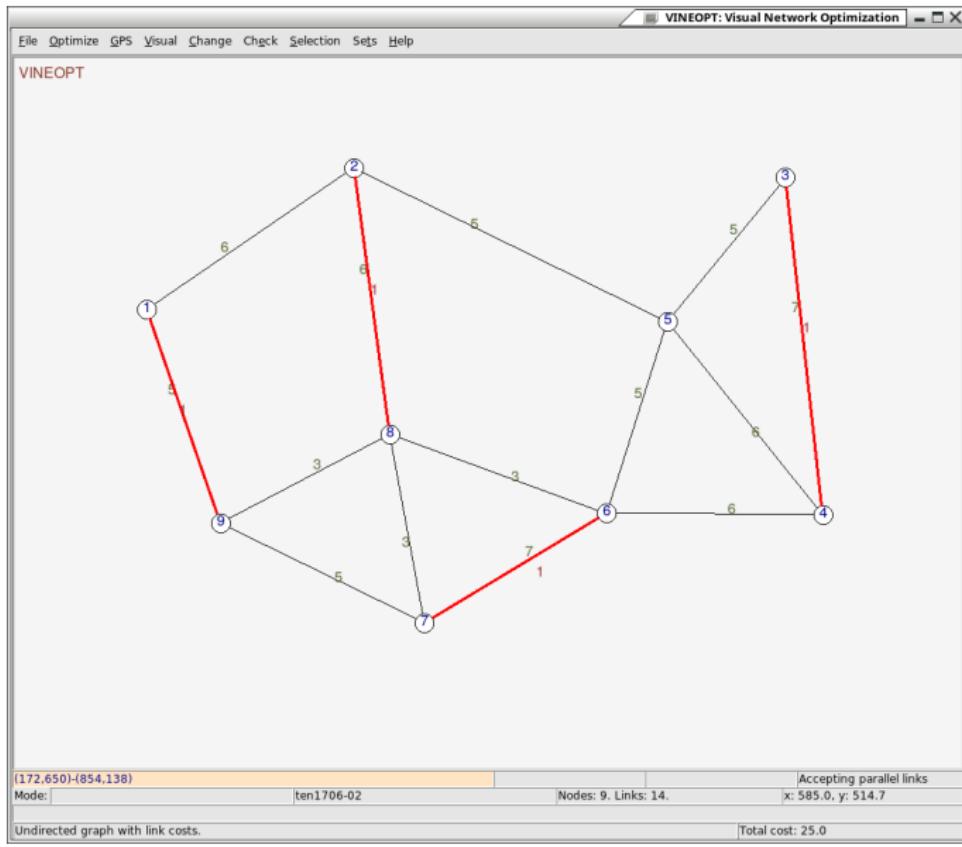
# Vineopt: TSP



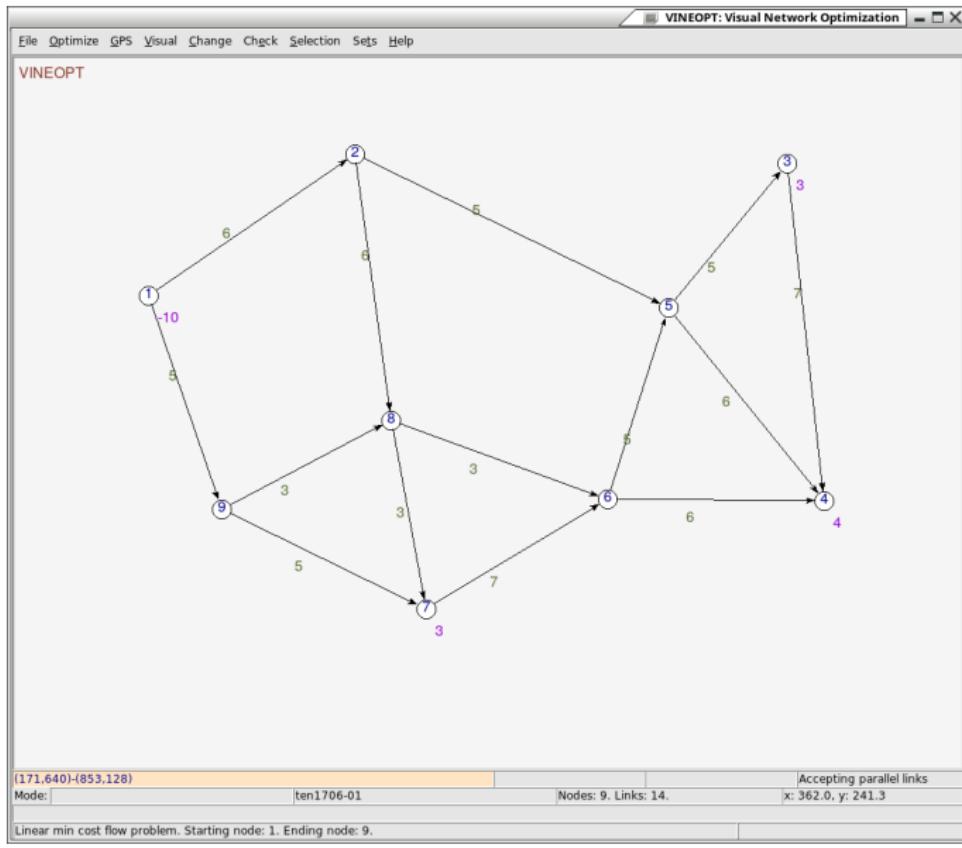
# Vineopt: Chinese postman



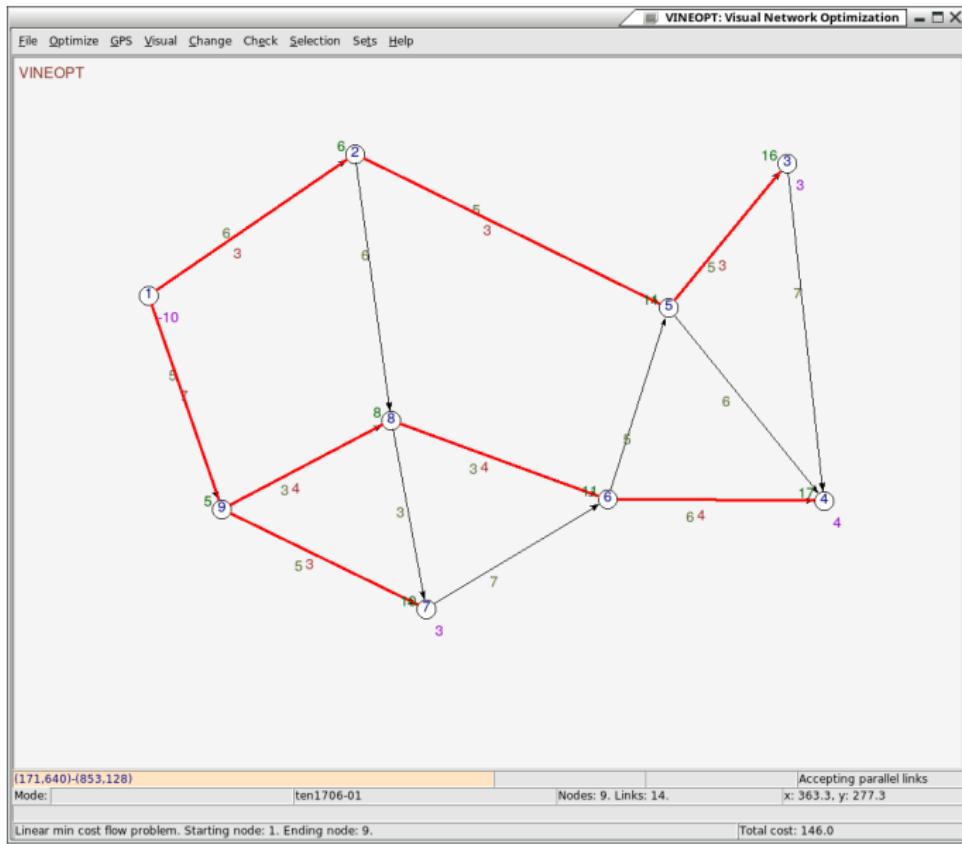
# Vineopt: matching



# Vineopt: directed network



# Vineopt: min cost flow



# Vineopt: solution in tables

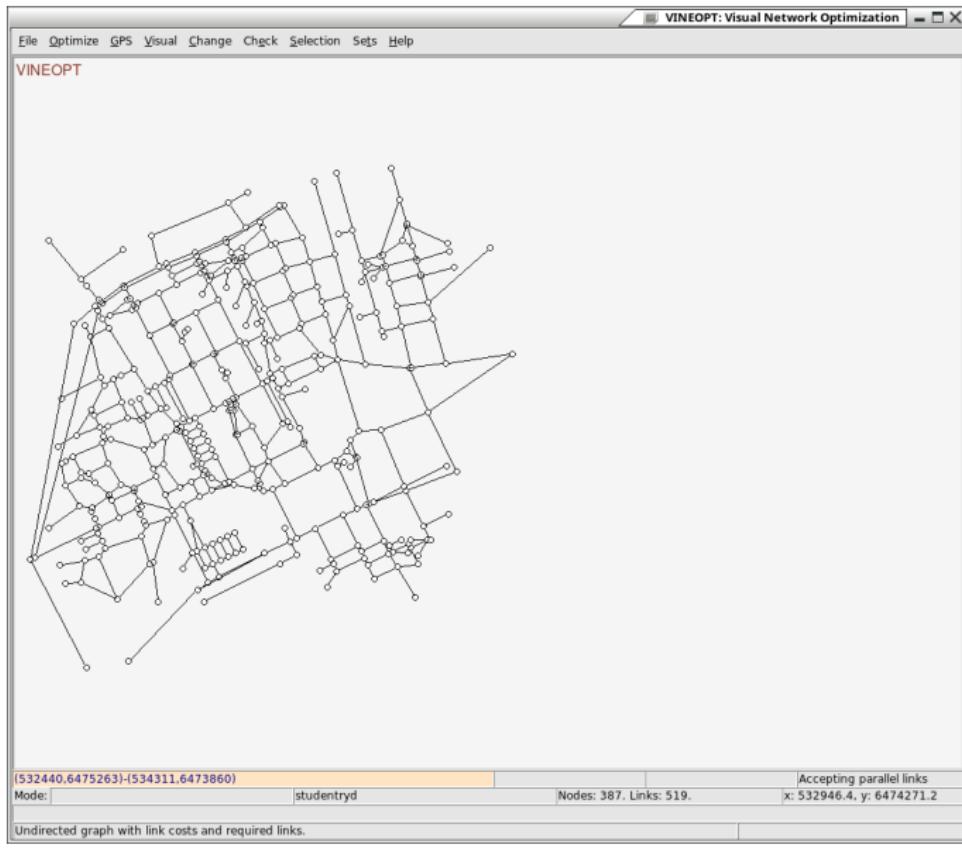
Current data					
Total cost: 146.0					
Link: 0: (1,2): Cost: 6.00,	Lowb: 0,	Cap: 99999999,	Flow: 3,	Red cost: 0,	
Link: 1: (1,9): Cost: 5.00,	Lowb: 0,	Cap: 99999999,	Flow: 7,	Red cost: 0,	
Link: 2: (2,5): Cost: 5.00,	Lowb: 0,	Cap: 99999999,	Flow: 3,	Red cost: 0,	
Link: 3: (2,8): Cost: 6.00,	Lowb: 0,	Cap: 99999999,	Flow: 0,	Red cost: 4,	
Link: 4: (3,4): Cost: 7.00,	Lowb: 0,	Cap: 99999999,	Flow: 0,	Red cost: 6,	
Link: 5: (5,3): Cost: 5.00,	Lowb: 0,	Cap: 99999999,	Flow: 3,	Red cost: 0,	
Link: 6: (5,4): Cost: 6.00,	Lowb: 0,	Cap: 99999999,	Flow: 0,	Red cost: 0,	
Link: 7: (6,4): Cost: 6.00,	Lowb: 0,	Cap: 99999999,	Flow: 4,	Red cost: 0,	
Link: 8: (6,5): Cost: 5.00,	Lowb: 0,	Cap: 99999999,	Flow: 0,	Red cost: 5,	
Link: 9: (7,6): Cost: 7.00,	Lowb: 0,	Cap: 99999999,	Flow: 0,	Red cost: 6,	
Link: 10: (8,6): Cost: 3.00,	Lowb: 0,	Cap: 99999999,	Flow: 4,	Red cost: 0,	
Link: 11: (8,7): Cost: 3.00,	Lowb: 0,	Cap: 99999999,	Flow: 0,	Red cost: 1,	
Link: 12: (9,7): Cost: 5.00,	Lowb: 0,	Cap: 99999999,	Flow: 3,	Red cost: 0,	
Link: 13: (9,8): Cost: 3.00,	Lowb: 0,	Cap: 99999999,	Flow: 4,	Red cost: 0,	

Dismiss <F3>

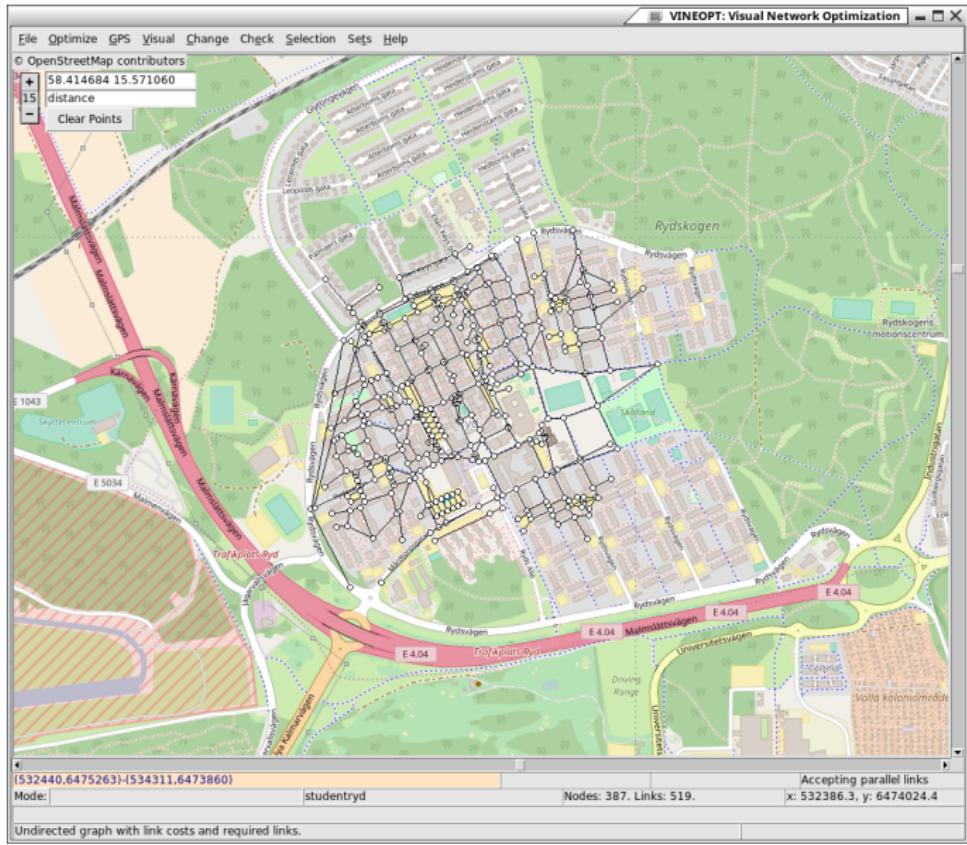
Save in file

Print

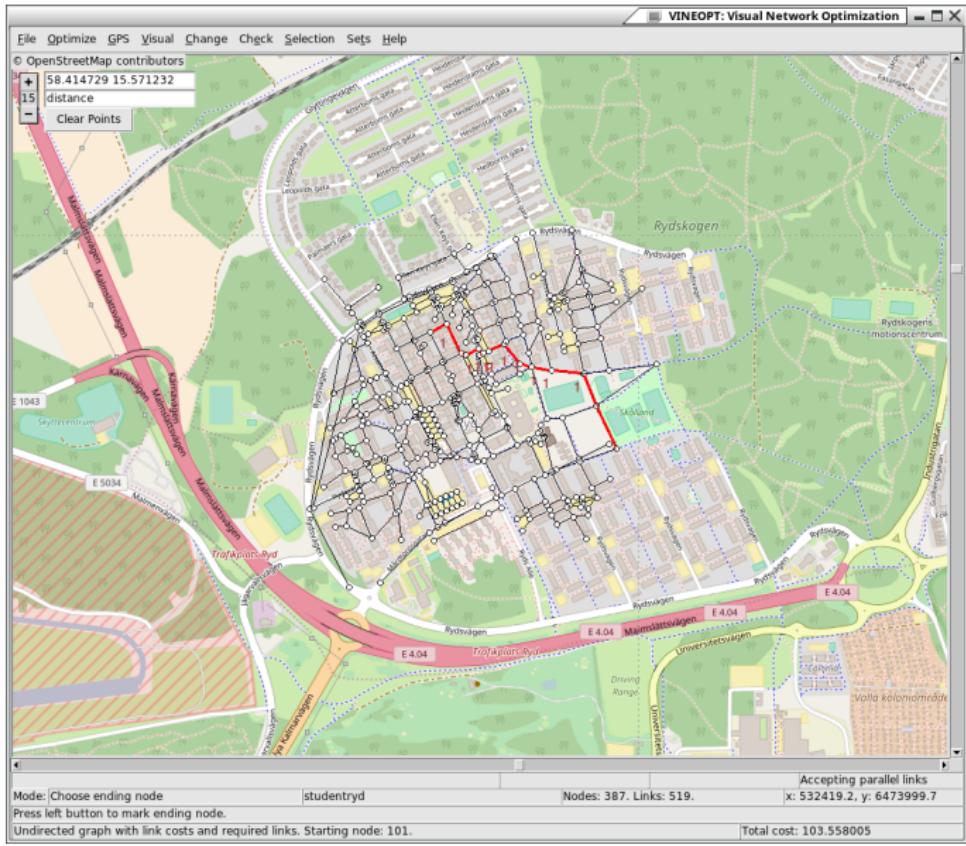
# Vineopt: Ryd - student part



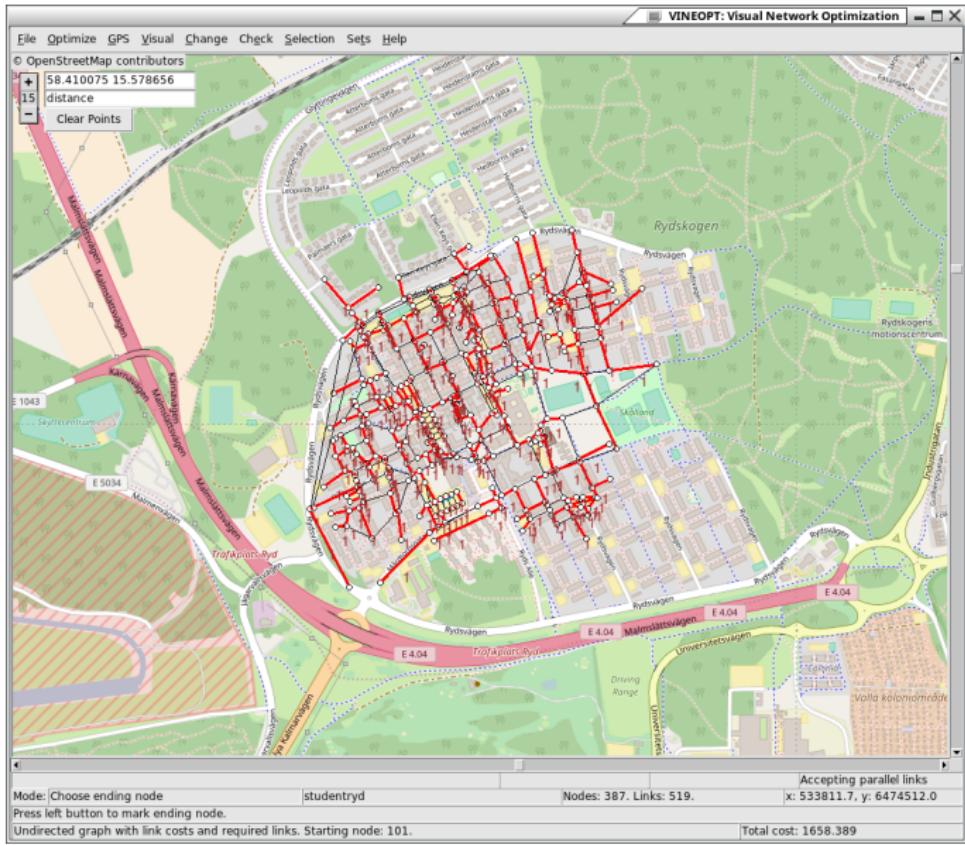
# Vineopt: Ryd - student part with OSM background



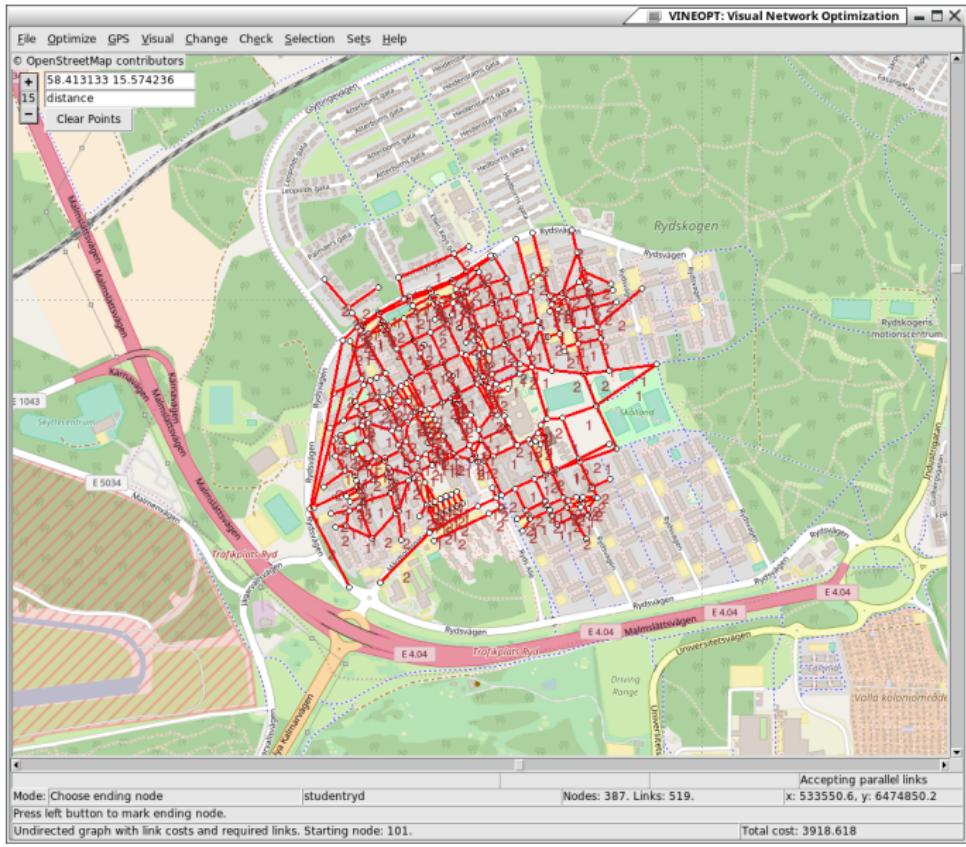
# Vineopt: shortest path in Ryd



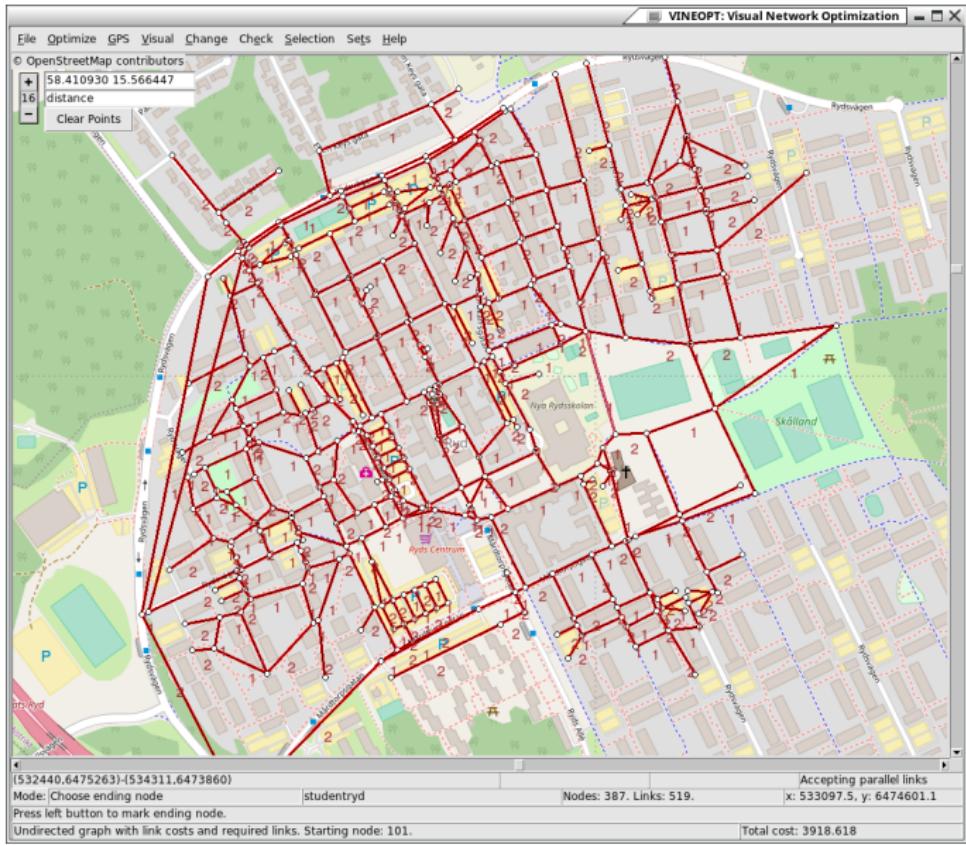
## Vineopt: MST in Ryd



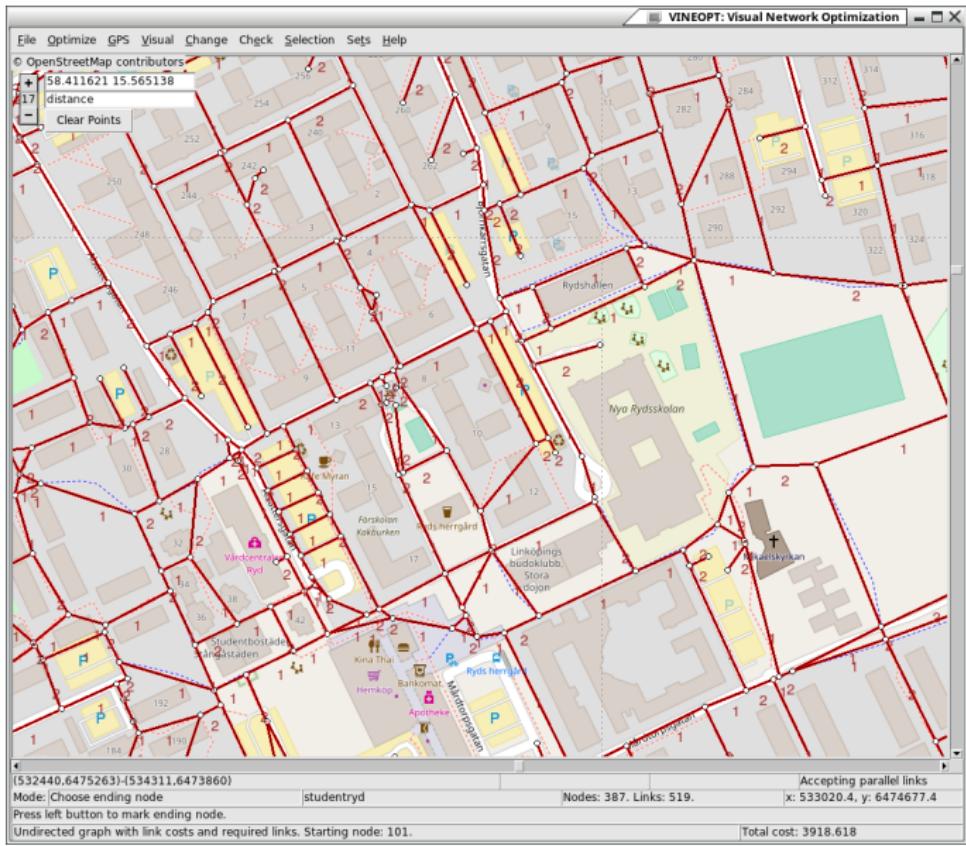
# Vineopt: Chinese postman in Ryd



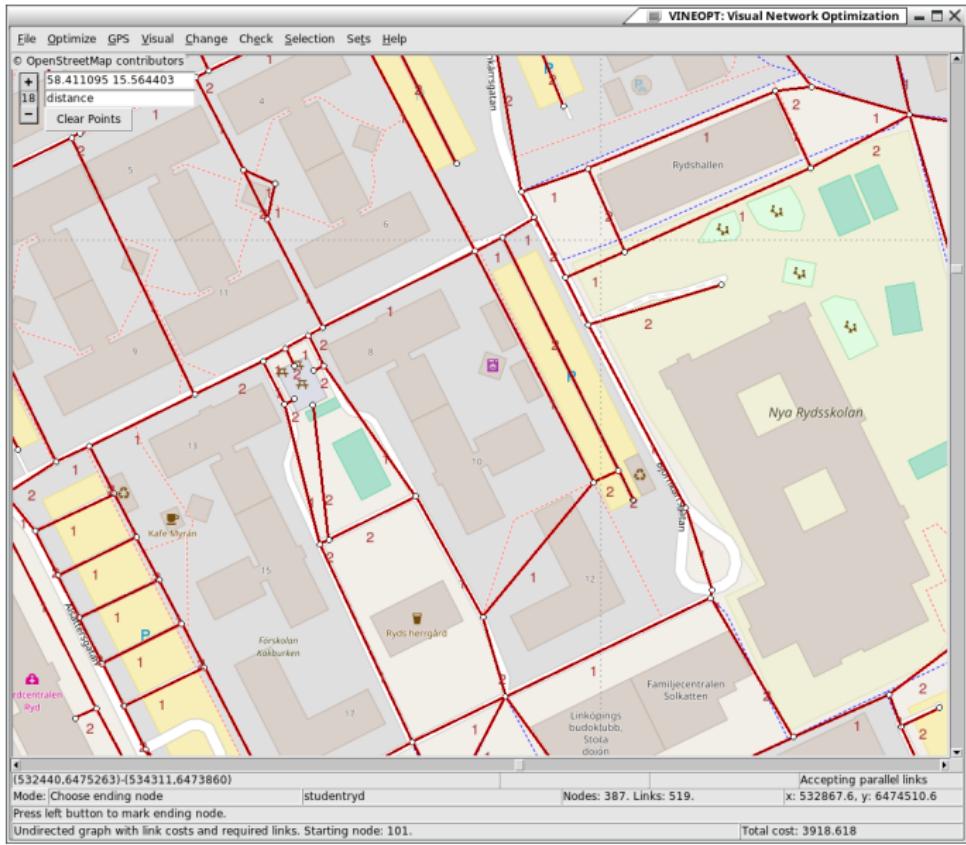
# Vineopt: Chinese postman in Ryd - zoom 1



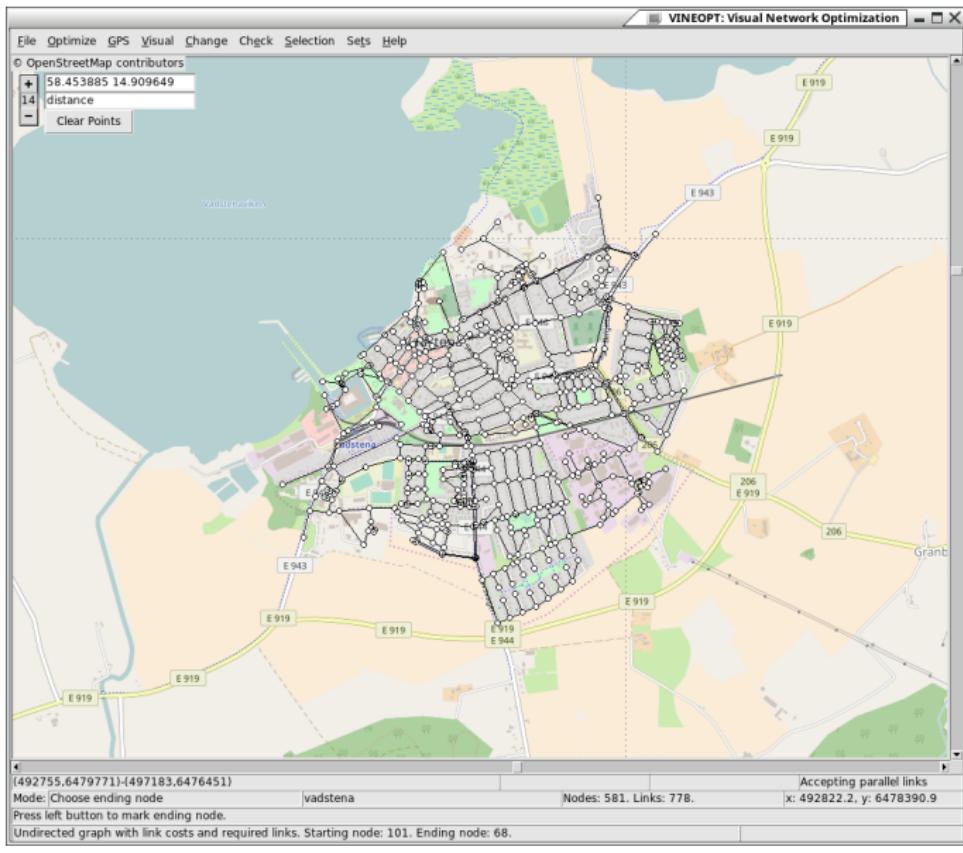
# Vineopt: Chinese postman in Ryd - zoom 2



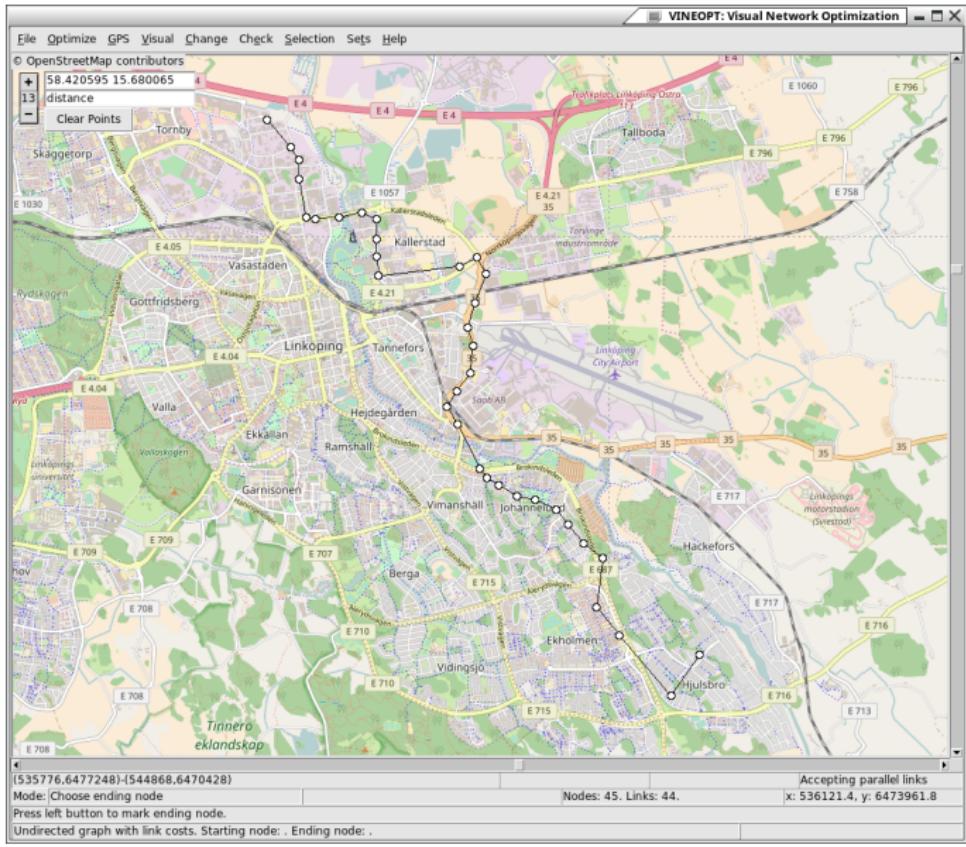
# Vineopt: Chinese postman in Ryd - zoom 3



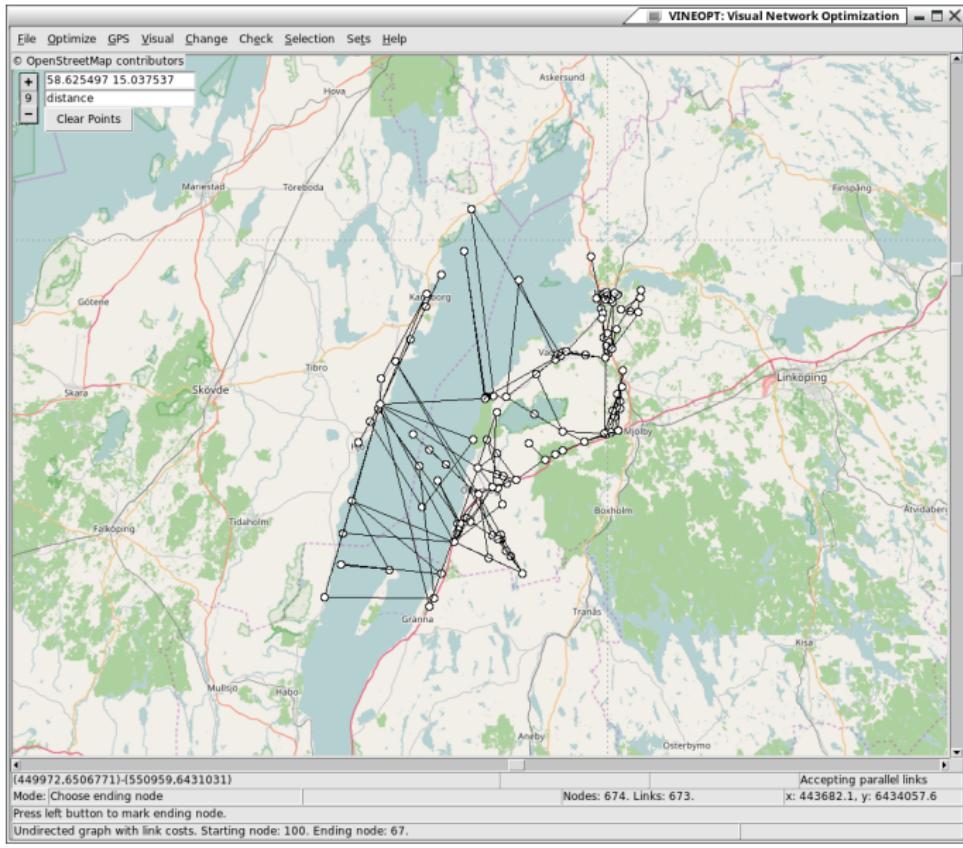
# Vineopt: Vadstena



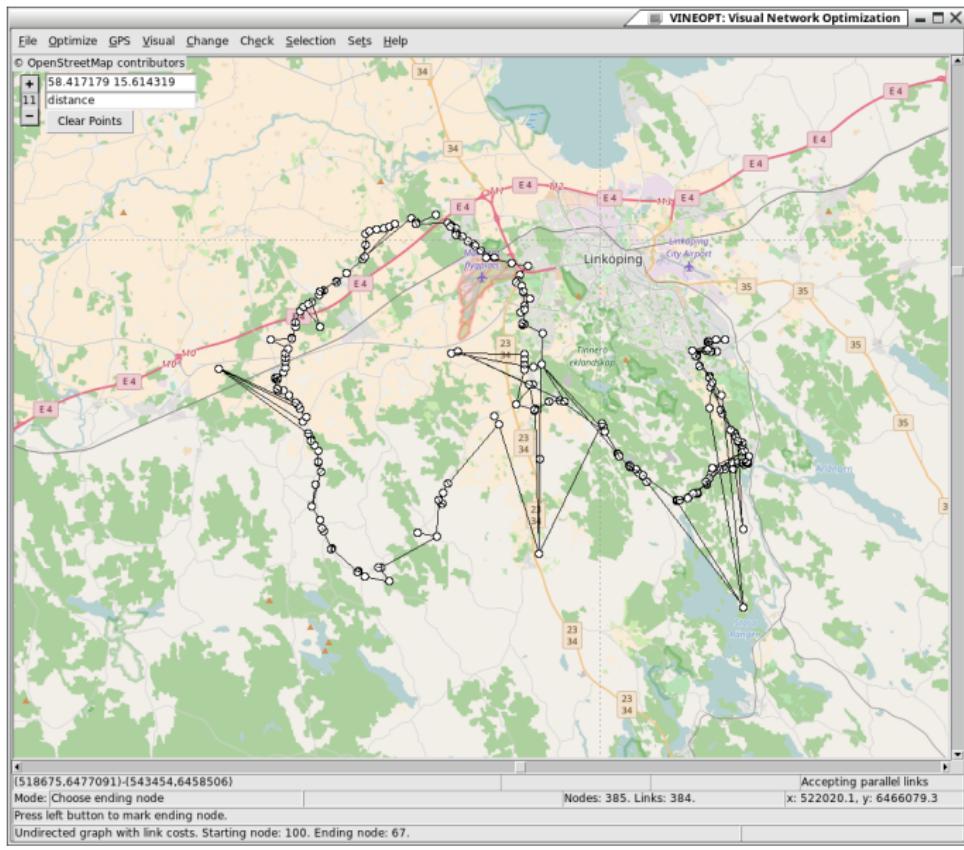
# Vineopt: GPS-track: home - Tornby



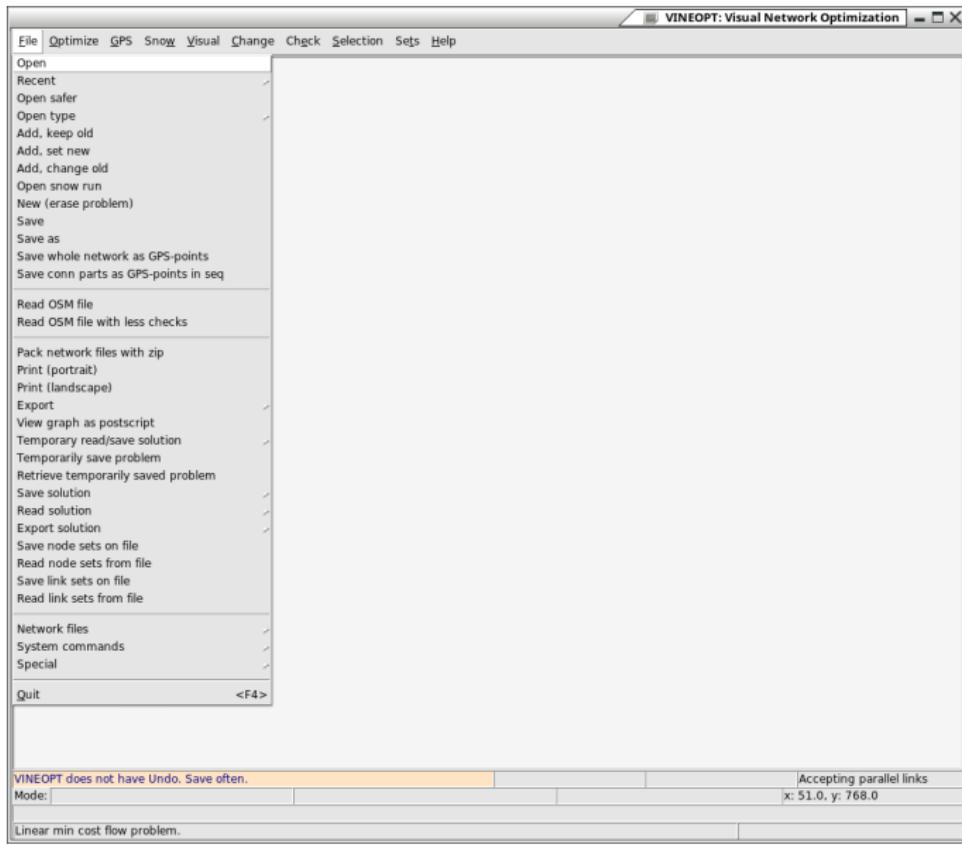
# Vineopt: bad GPS-track: Halvvättern



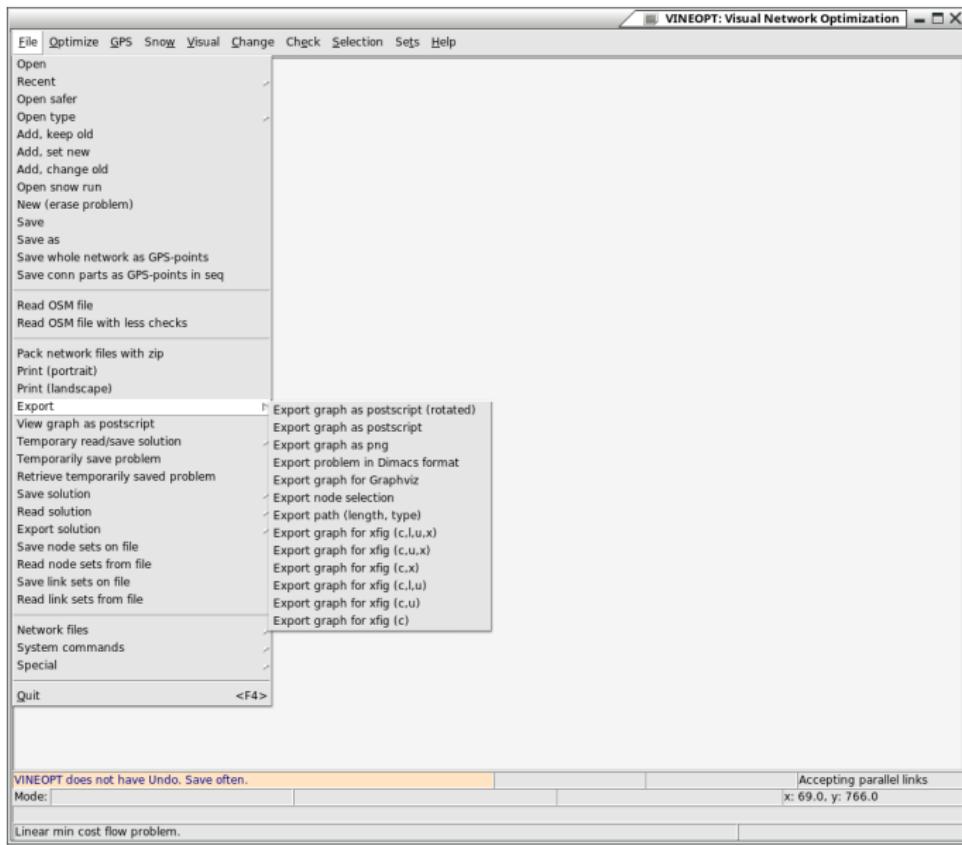
# Vineopt: GPS-track with few errors: round trip



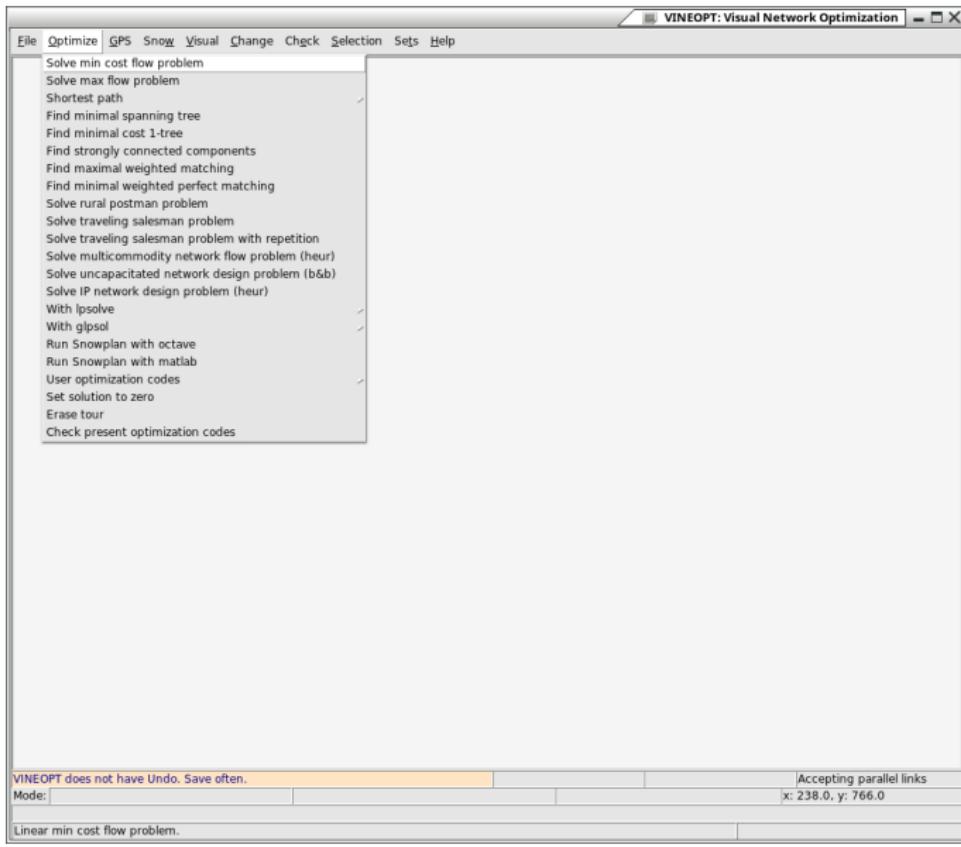
# Vineopt menu 1



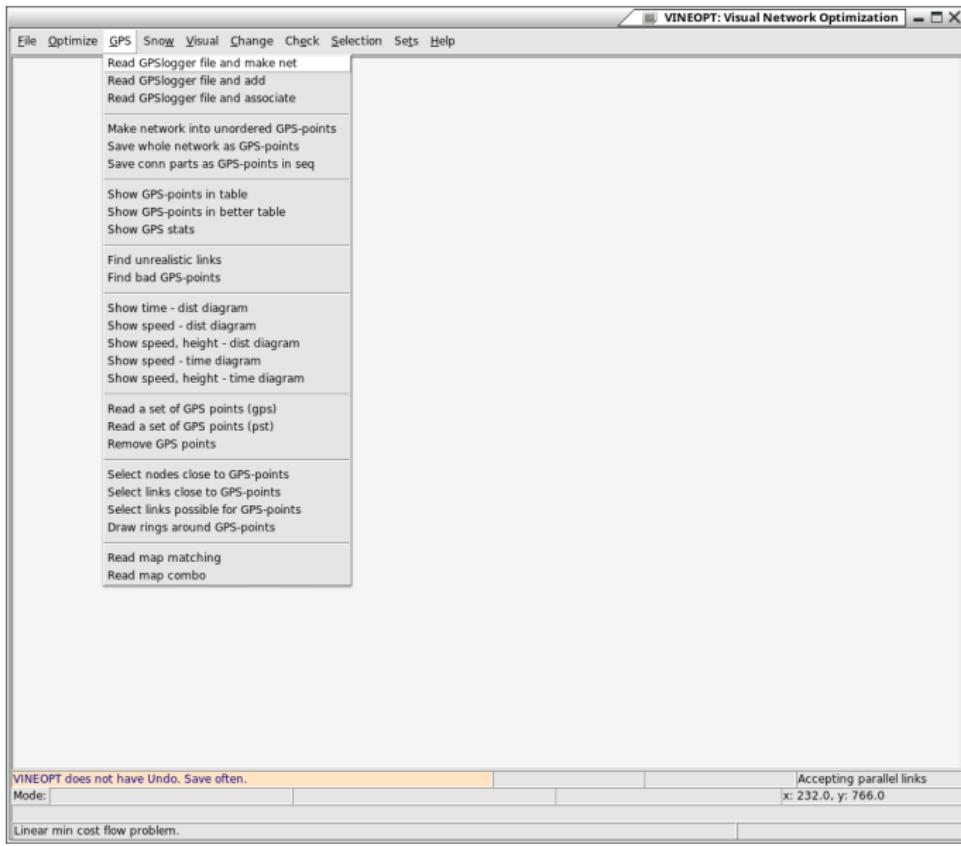
# Vineopt menu 1b



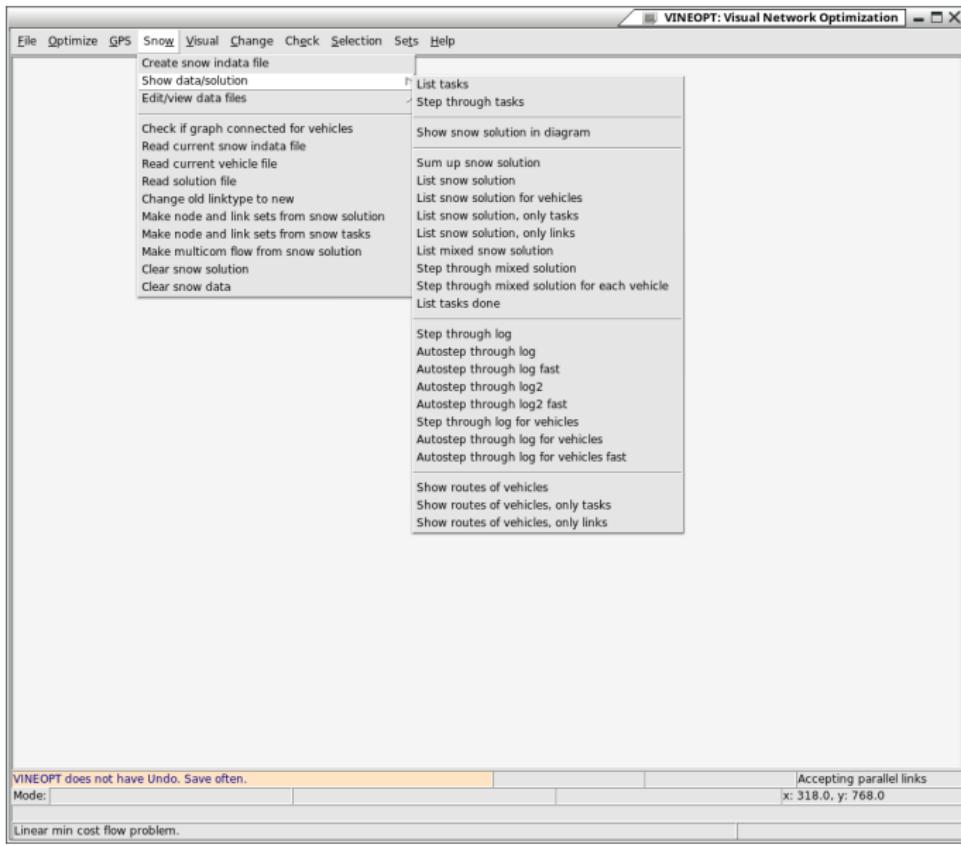
## Vineopt menu 2



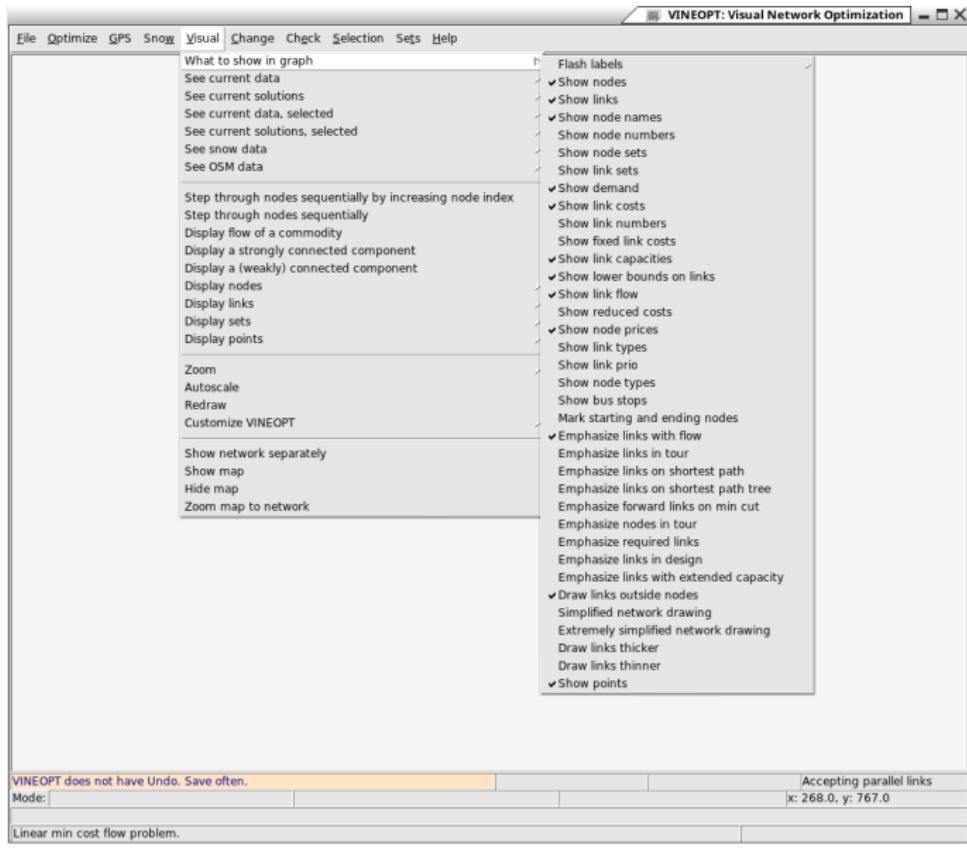
# Vineopt menu 3



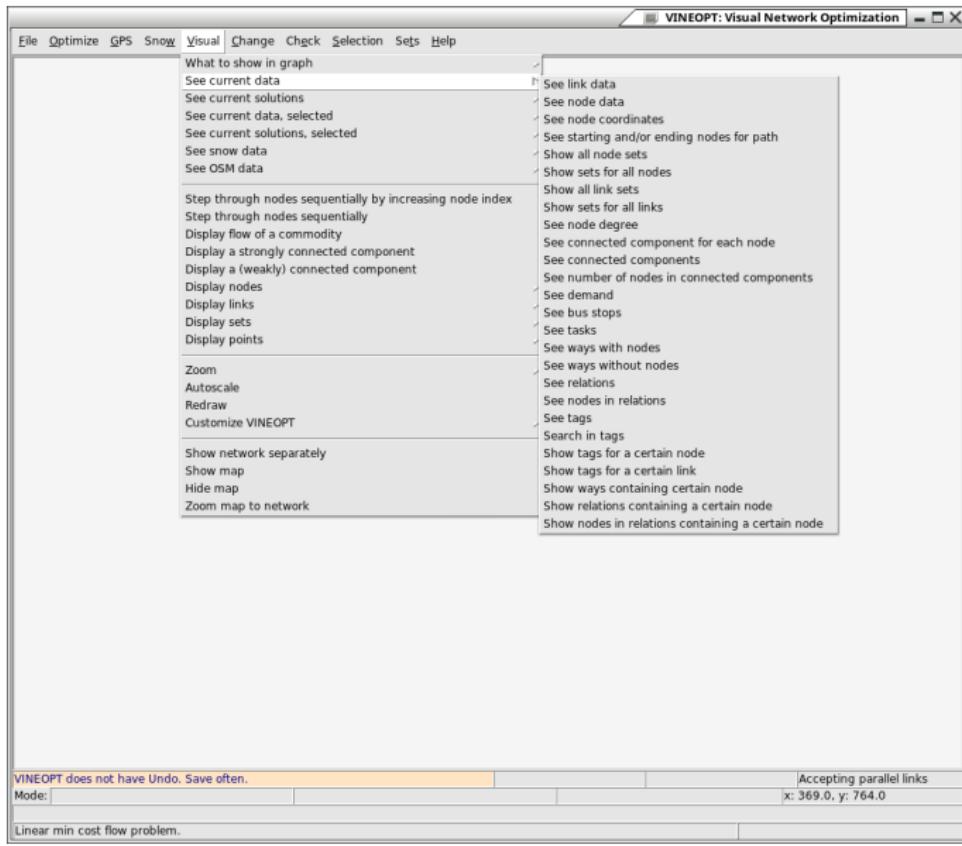
# Vineopt menu 4



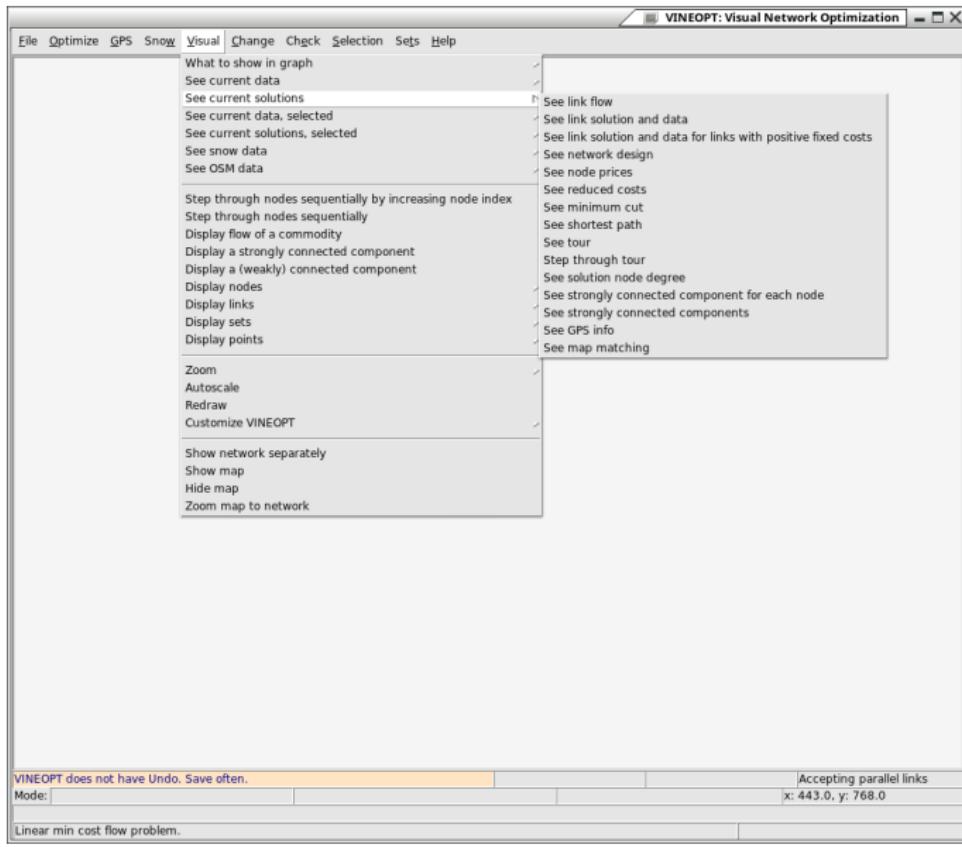
# Vineopt menu 5a



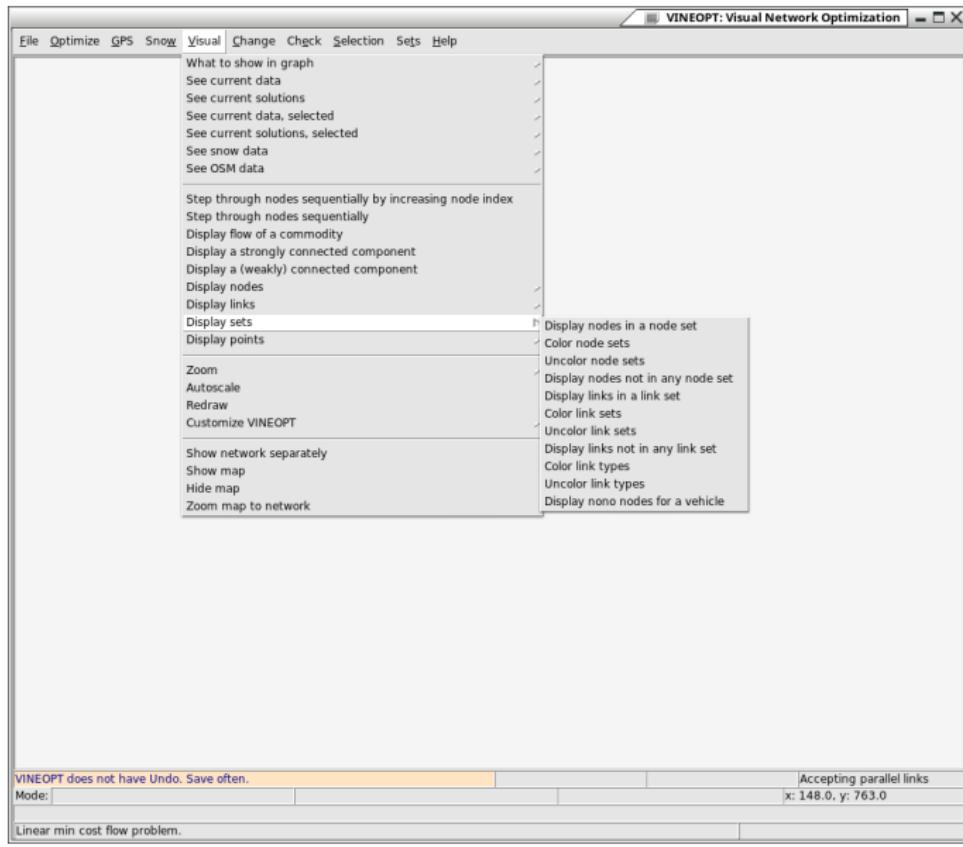
# Vineopt menu 5b



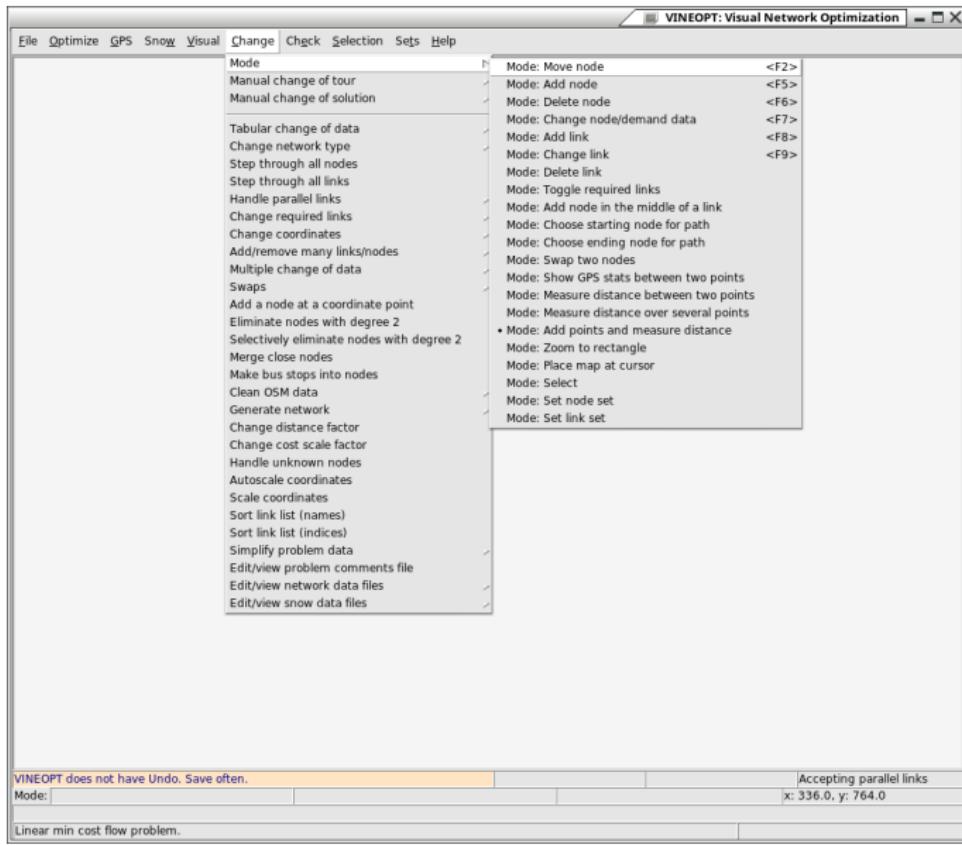
# Vineopt menu 5c



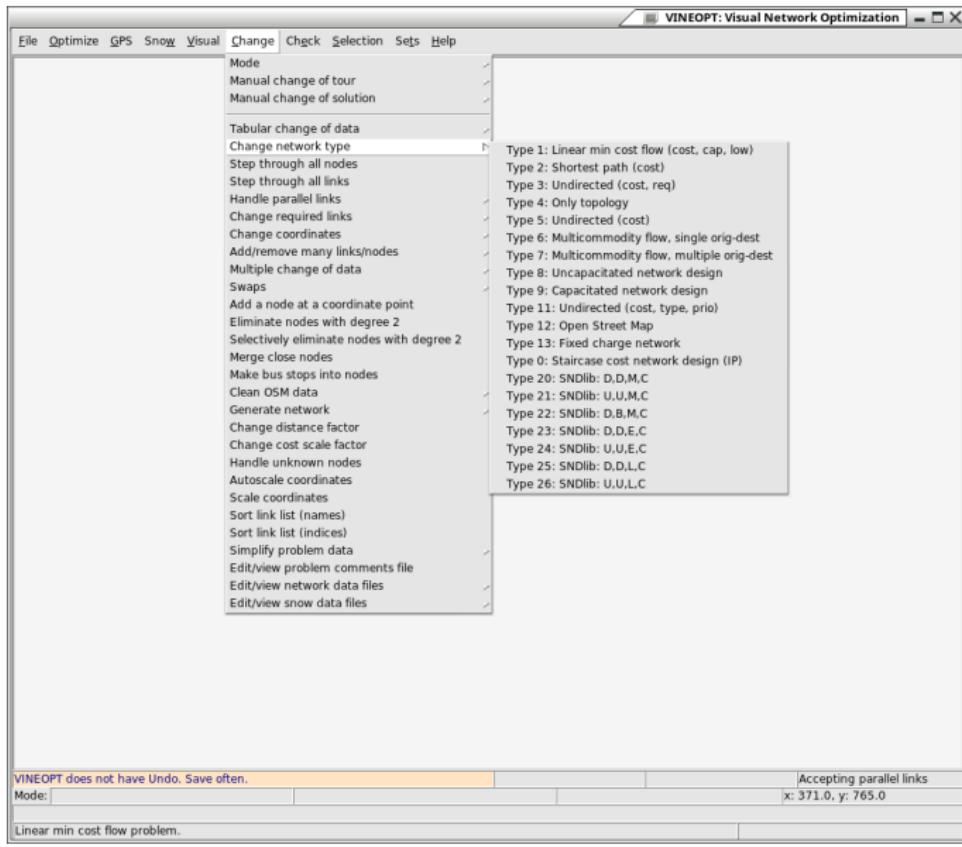
# Vineopt menu 5d



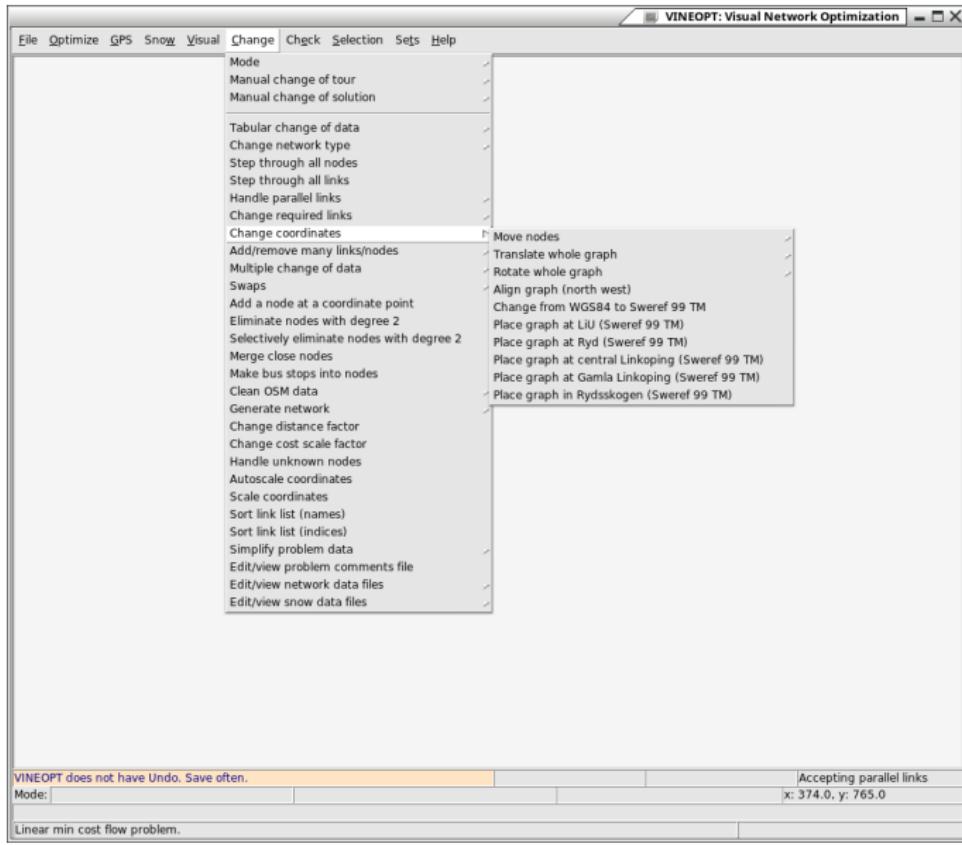
# Vineopt menu 6a



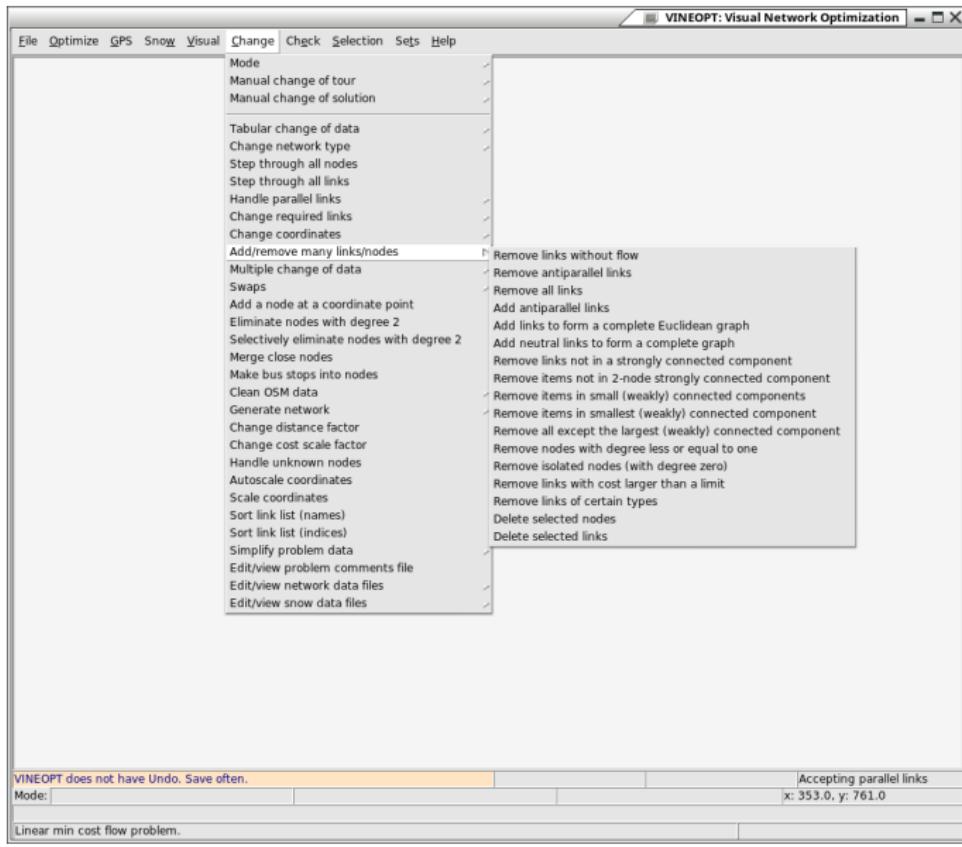
# Vineopt menu 6b



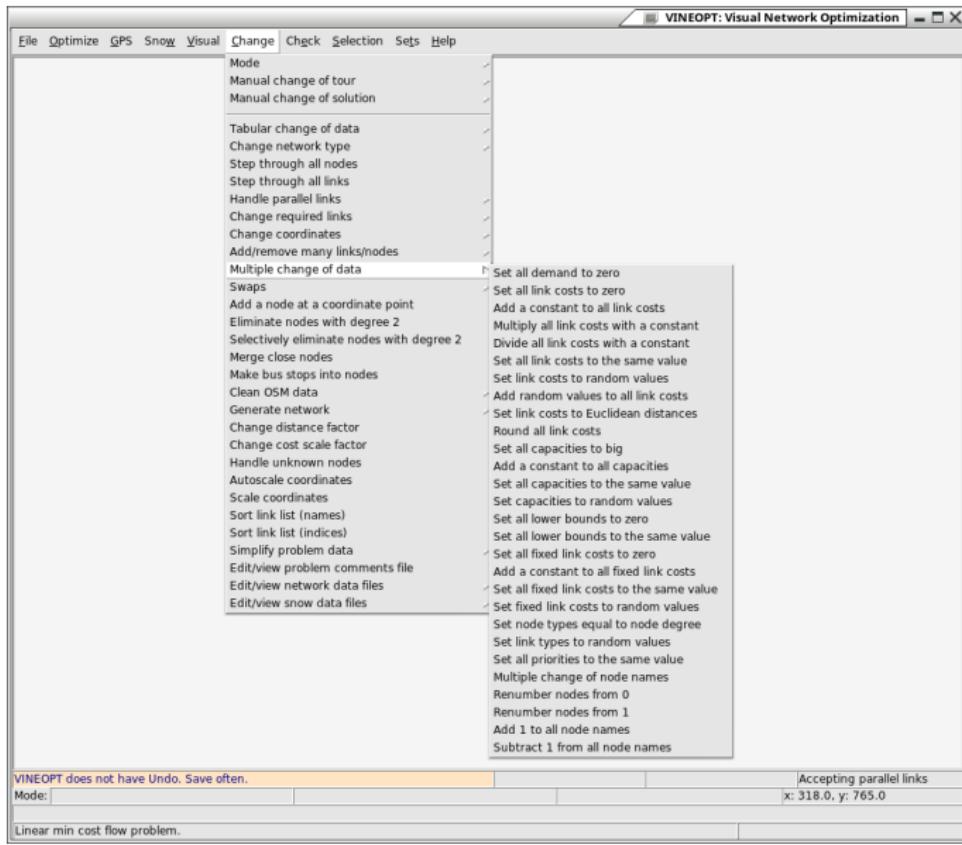
# Vineopt menu 6c



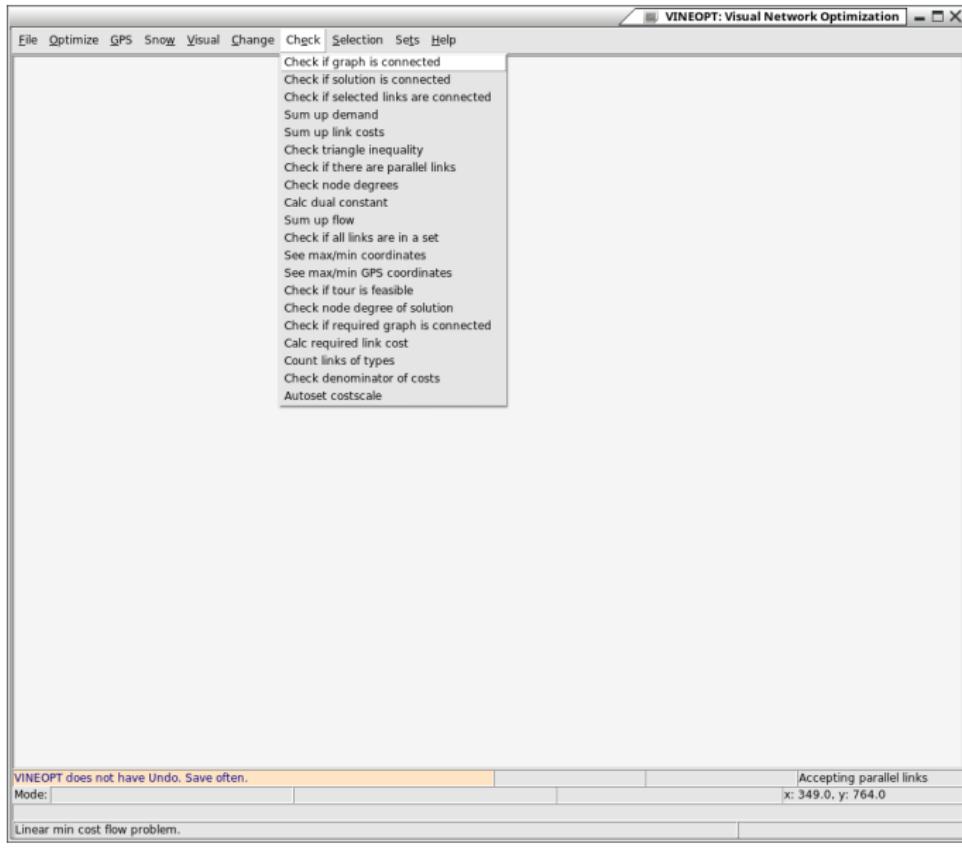
# Vineopt menu 6d



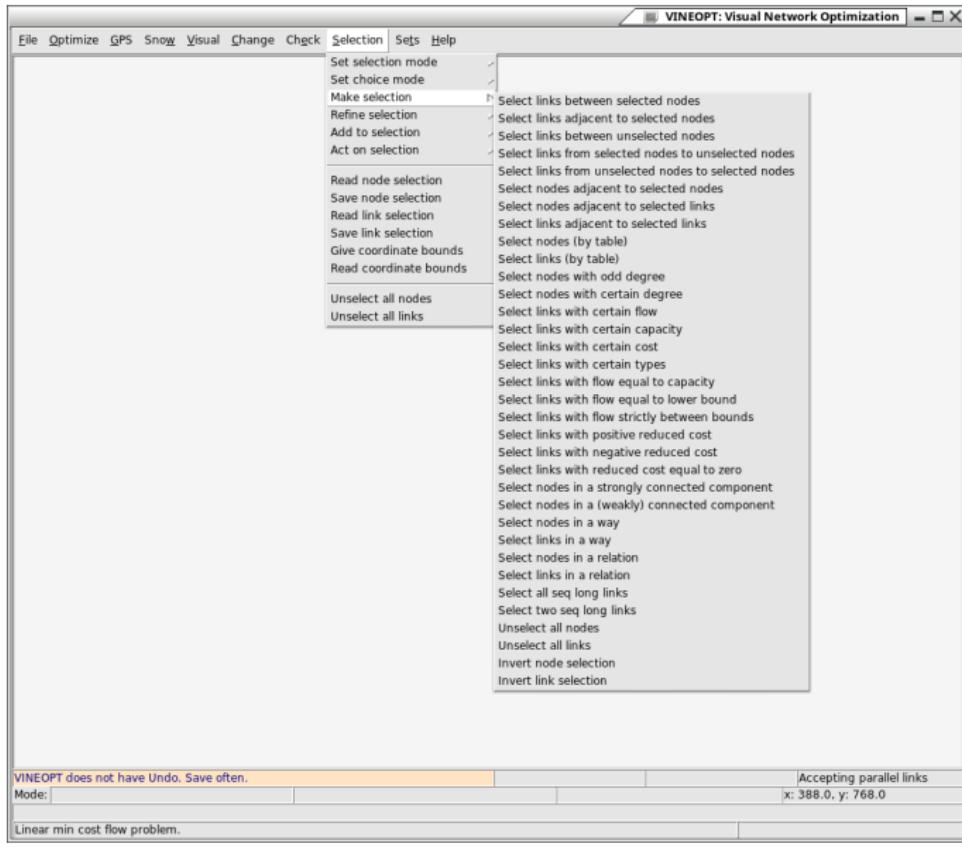
# Vineopt menu 6e



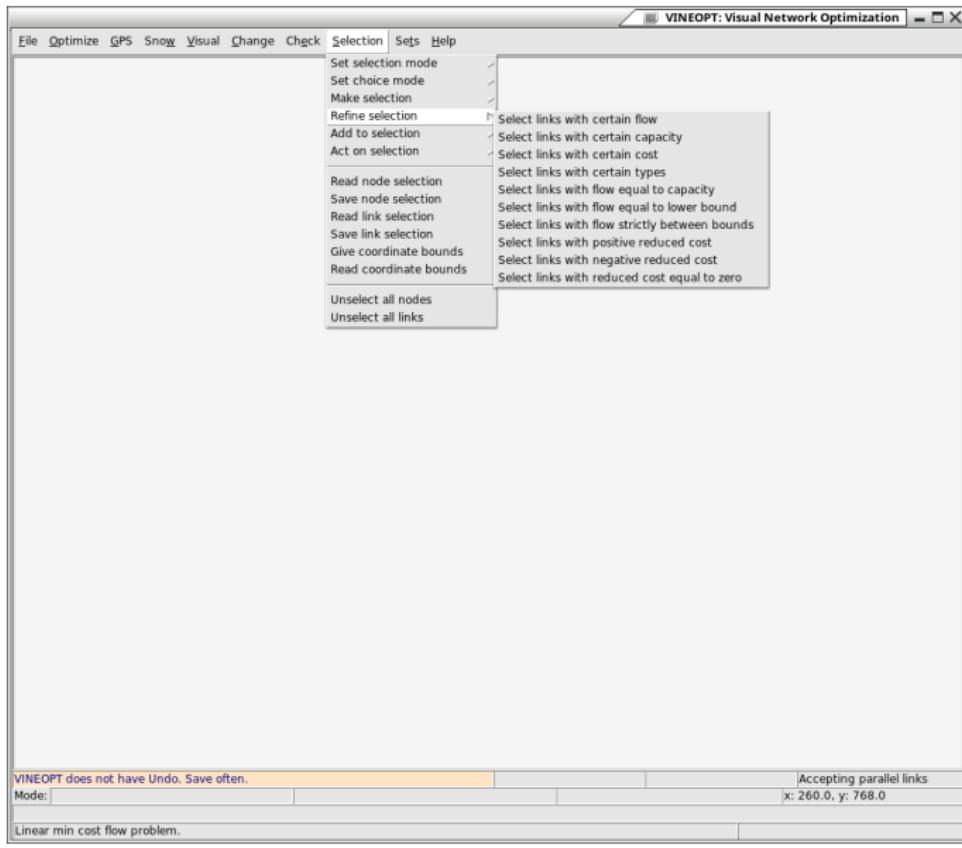
# Vineopt menu 7



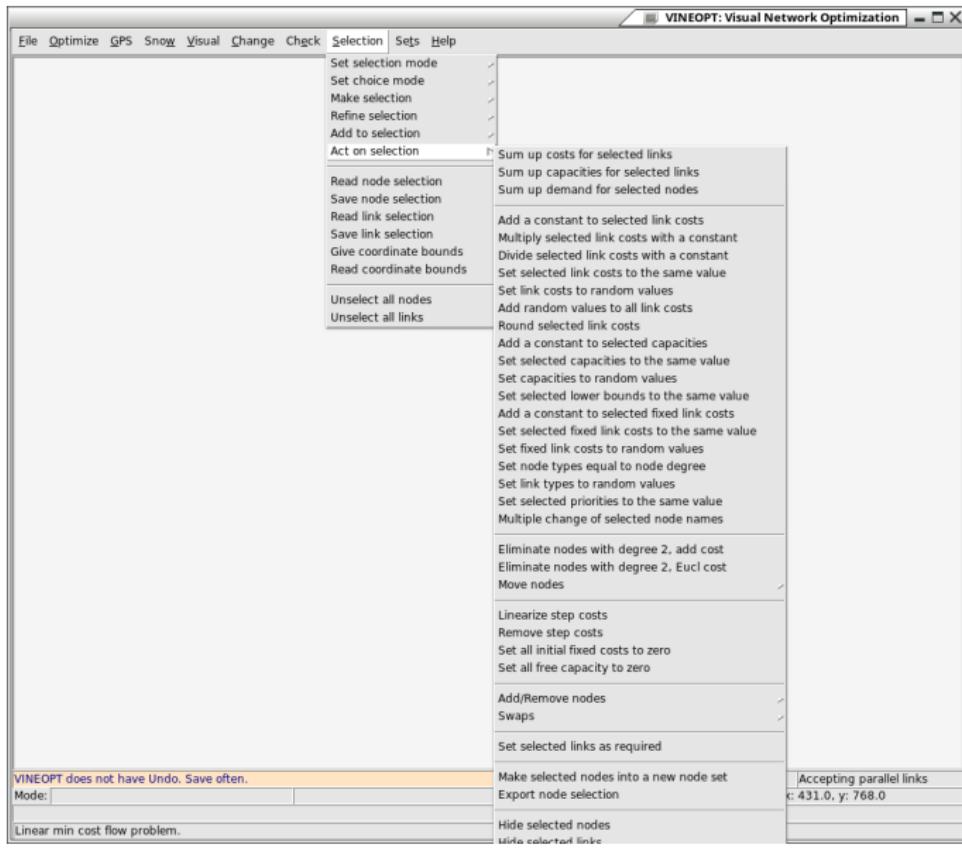
# Vineopt menu 8a



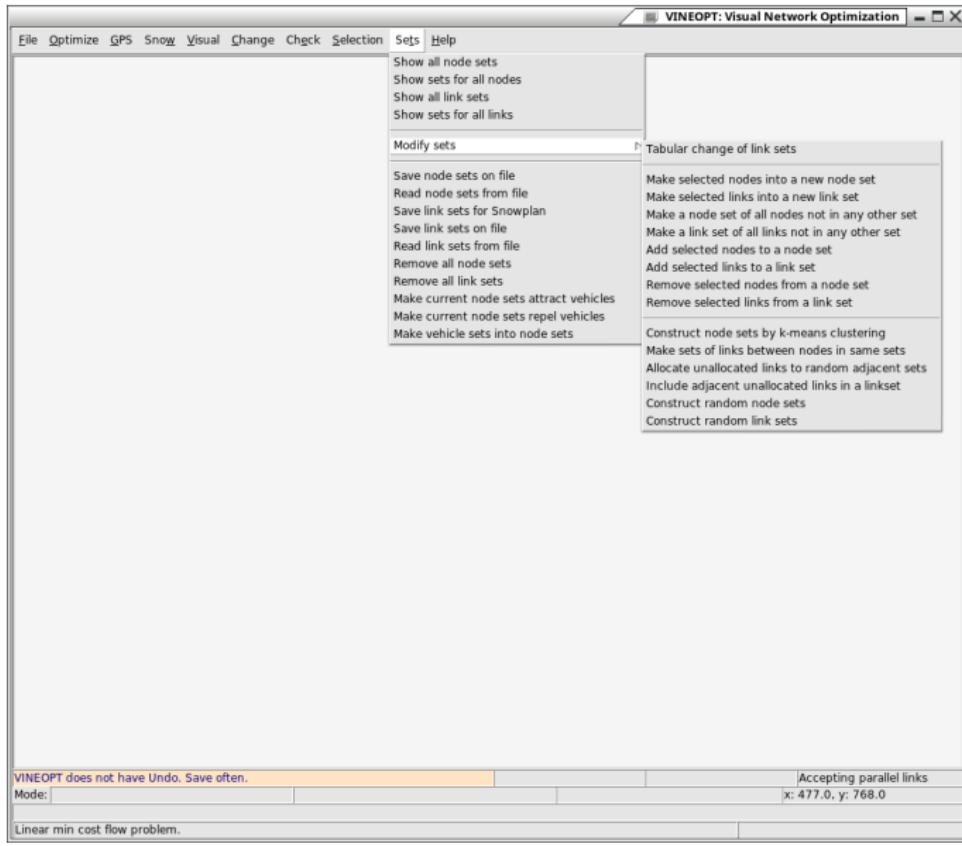
## Vineopt menu 8b



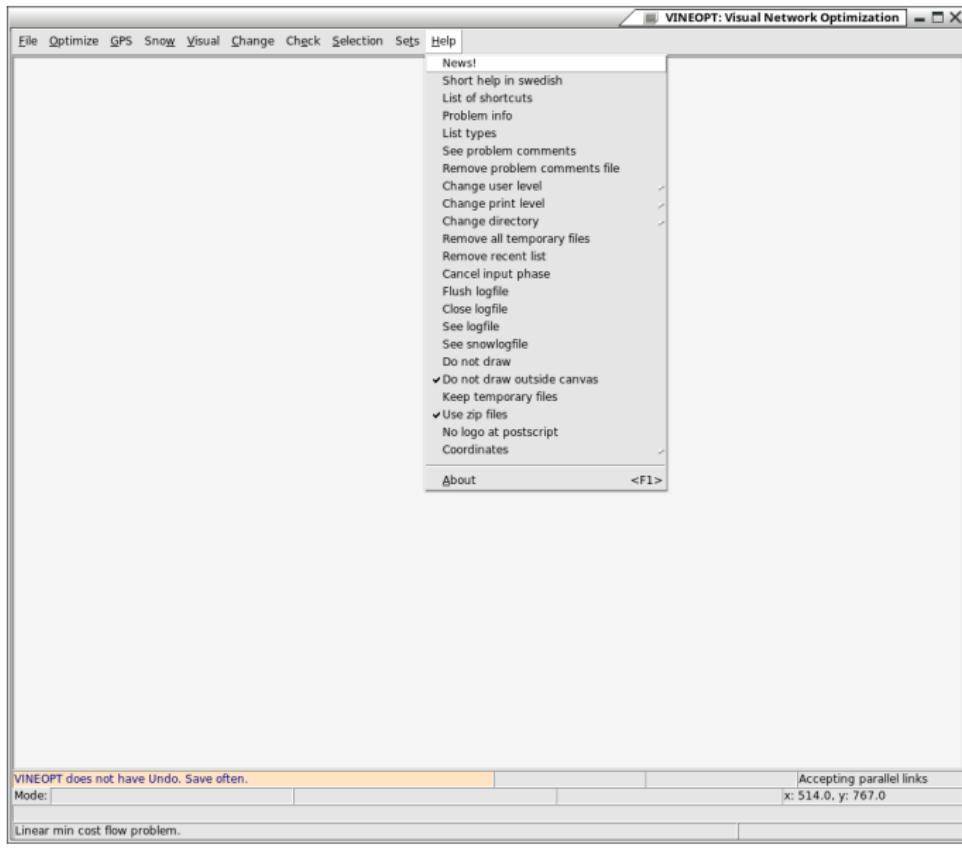
# Vineopt menu 8c



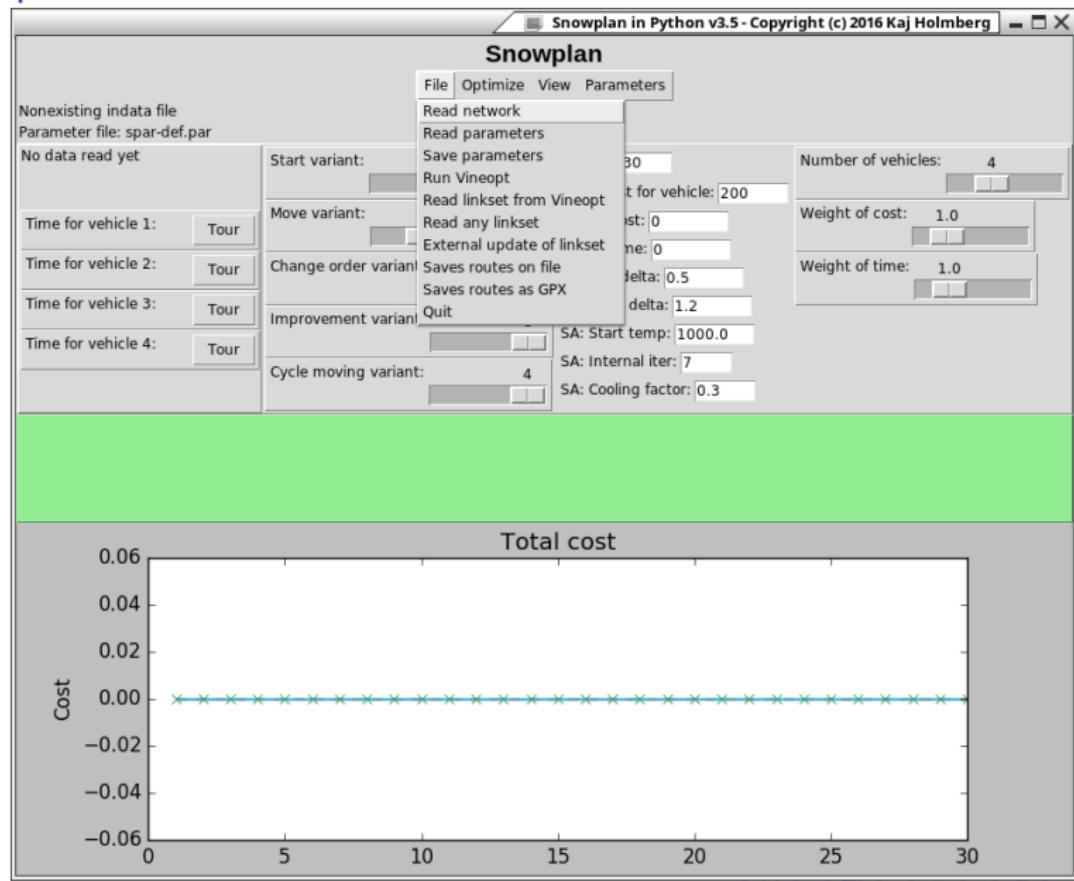
# Vineopt menu 9



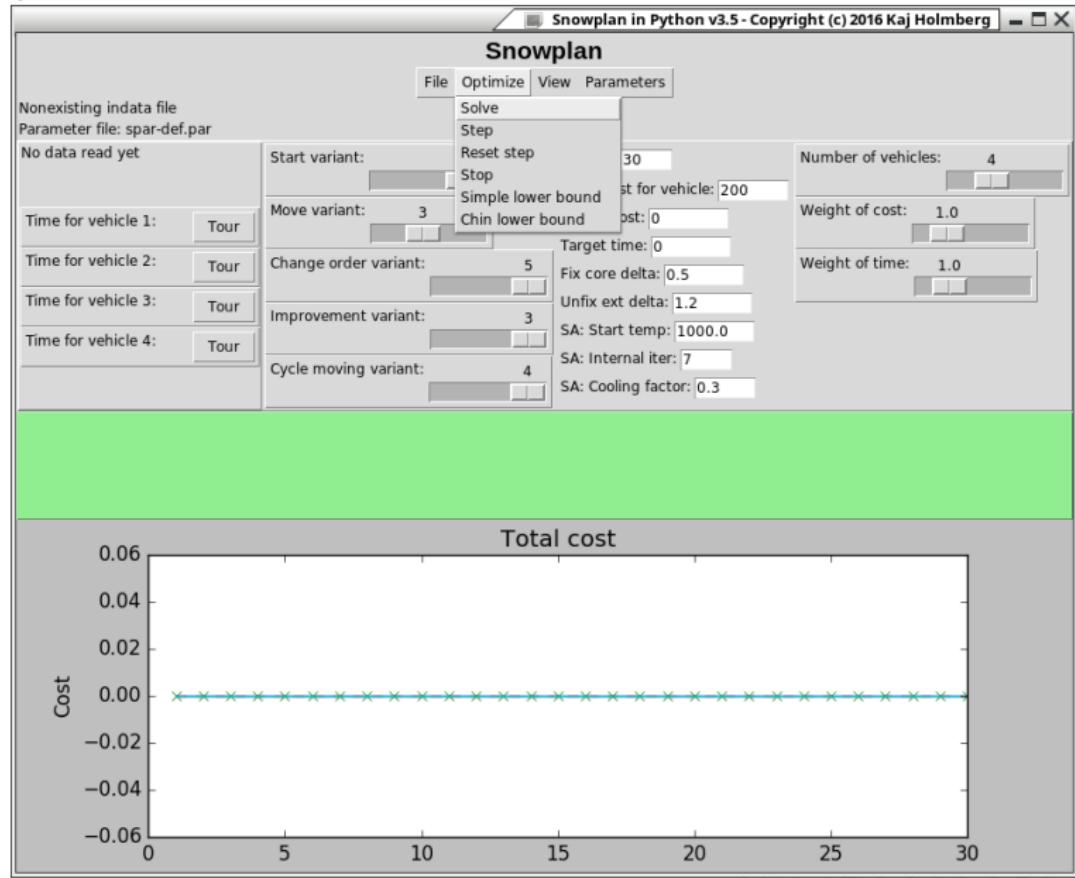
# Vineopt menu 10



# Snowplan menu 1



## Snowplan menu 2



# Snowplan info

Parameters:

Start variant:

- 0: Start from current solution.
- 1: Create a random allocation.
- 2: Start from Vineopt allocation.
- 3: Let vehicle 1 to do everything.
- 4: Sort by x for starting node.
- 5: Sort by x for ending node.
- 6: Sort by x for middle point.
- 7: Sort by y for starting node.
- 8: Sort by y for ending node.
- 9: Sort by y for middle point.

- 10: Sort by distance from the origin to starting node.
- 11: Sort by distance from the origin to ending node.
- 12: Sort by distance from the origin to middle point.

- 13: Use kmeans, limits to highest number.
- 14: Use kmeans, limits to lowest number.
- 15: Use kmeans, limits randomly.
- 16: Use kmeans, limits min dist to center.

Move variant:

- 0: No change of vehdo.
- 1: Swap two random links.
- 2: Change some random links.

- 3: Move a link from a vehicle with largest number of links to a vehicle with lowest.
- 4: Move a link between two random vehicles that have both nodes in common.
- 5: Swap between two random vehicles that have both nodes in common.

- 6: Move a link, when one node is in common.
- 7: Swap a link, when one node is in common.

Change order variant:

- 0: No change of order.
- 1: Swap two random elements.
- 2: Create a completely new order.
- 3: Shift the order to the right.
- 4: Shift the order to the left.

- 5: Swap one that has most with one that has least.

Improvement variant:

- 0: No move of cycles.
- 1: Move cycles, one random try.

- 2: Move cycles, terminate when there is no improvement.
- 3: Move cycles, try all combinations.

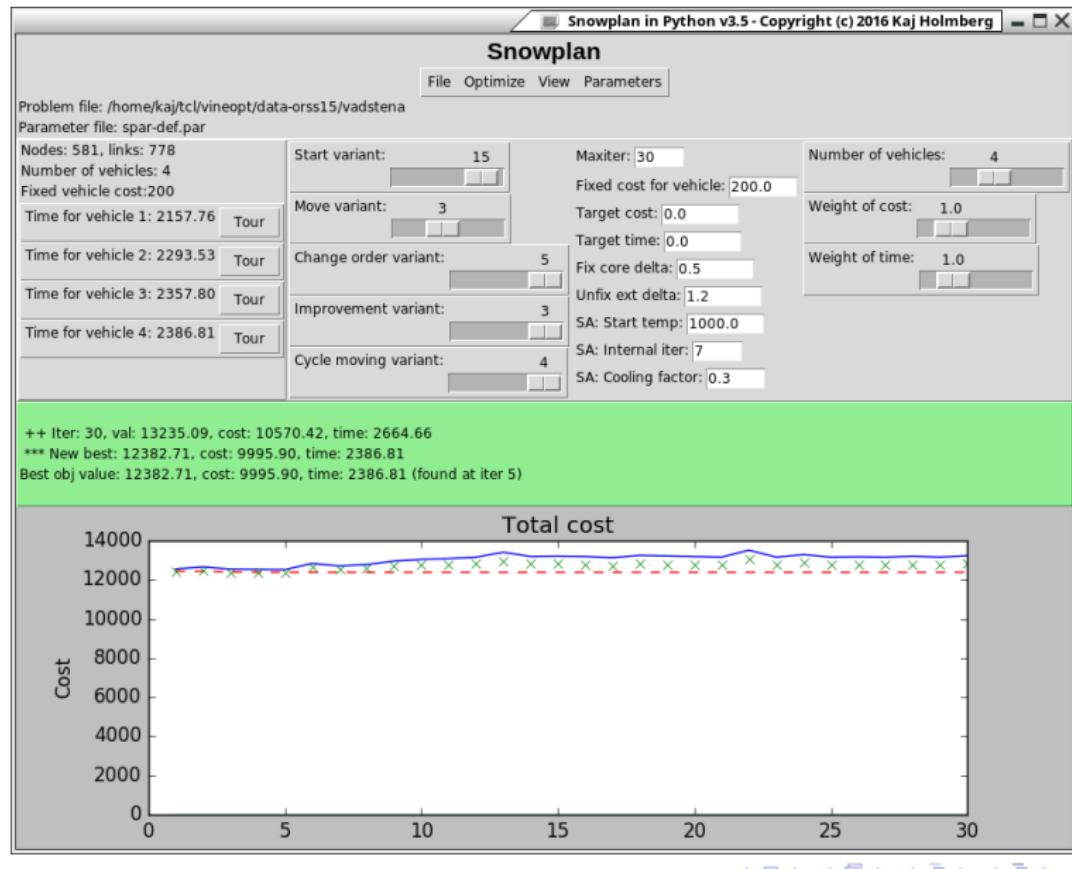
Cycle moving variant:

- 0: No move of cycle.

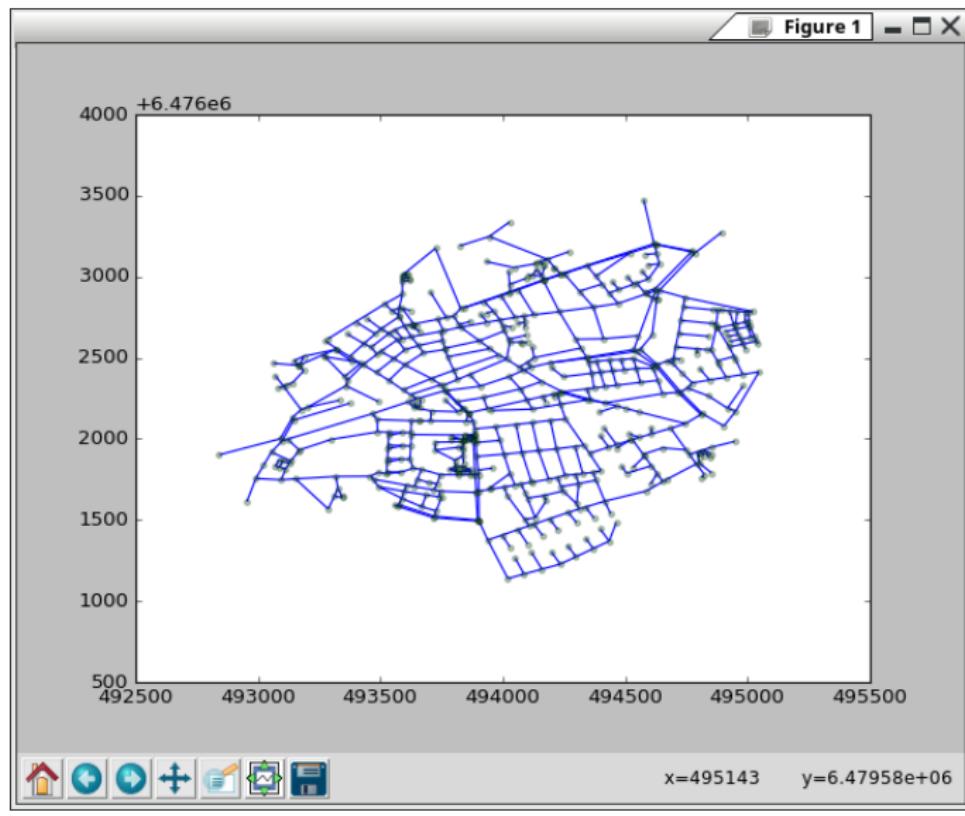
- 1: Try to move a random cycle between two random vehicles.
- 2: Try to move a cycle from a long tour to a short one.
- 3: Try to move a cycle between two given vehicles.
- 4: Try to move a cycle from a costly tour to a cheap one.

Don't forget to hit Update after change of a parameter.

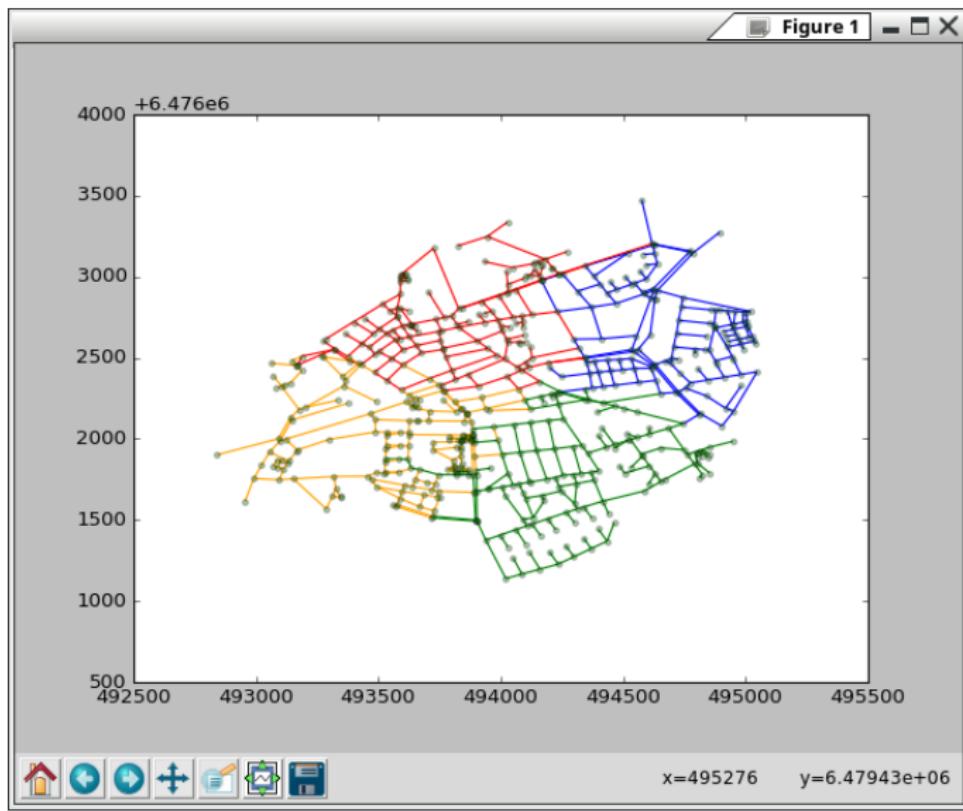
# Snowplan: a run - Vadstena



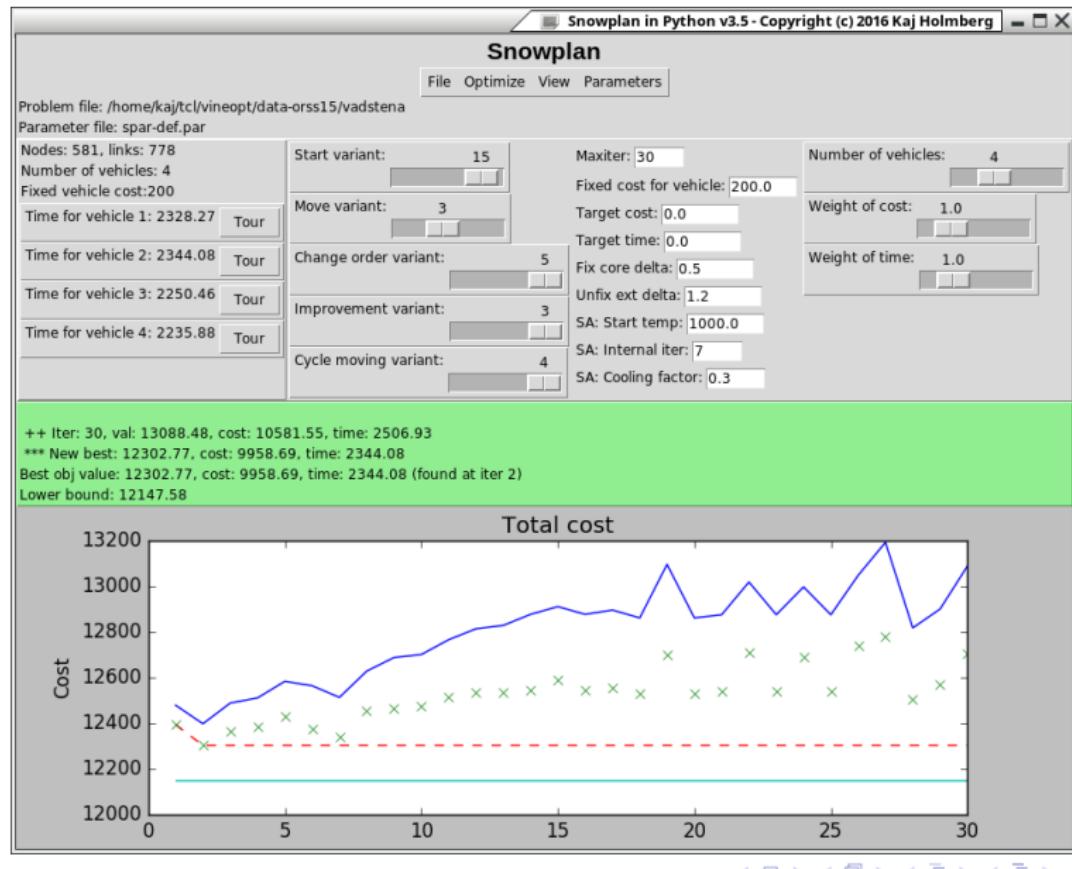
# Snowplan: Vadstena network



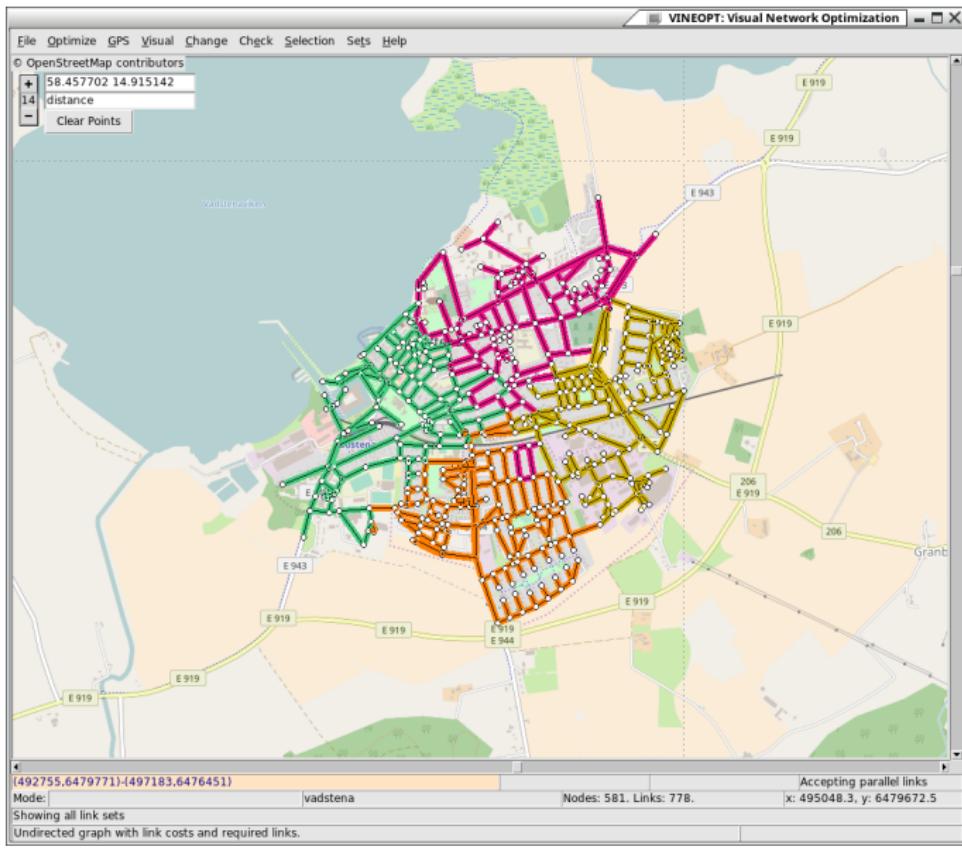
# Snowplan: Vadstena allocation



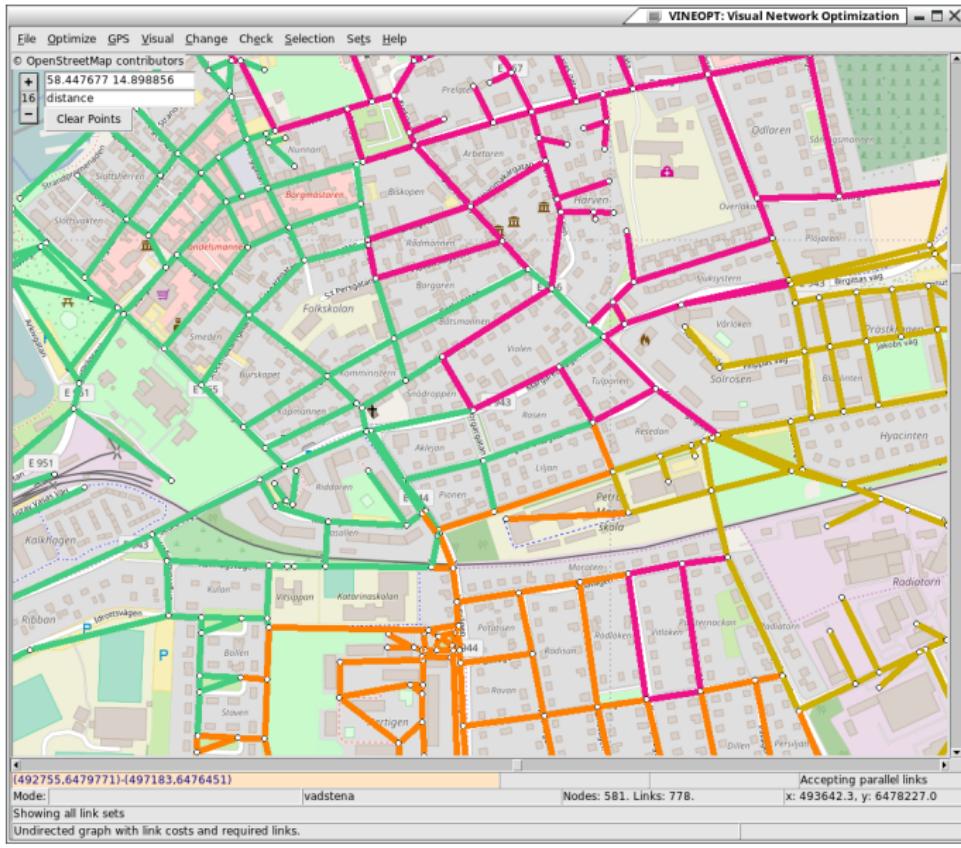
# Snowplan: a run with lower bound - Vadstena



# Vineopt with allocation - started from Snowplan



# Vineopt with allocation - zoom for manual change



# Vigropt: students not allocated

Unallocated persons:

Persons not allocated:	Use	Ed	Del
AL-MUFTI SUMMIA	Use	Ed	Del
ASPGREN ANTON	Use	Ed	Del
BÄNGERIUS SEBASTIAN	Use	Ed	Del
CLAESSEN MARTIN	Use	Ed	Del
DUFBÄCK DENNIS	Use	Ed	Del
EDHOLM LOVISA	Use	Ed	Del
ESKILSSON SOFIE	Use	Ed	Del
FLEMING THEODOR	Use	Ed	Del
FRÖBERG FELIX	Use	Ed	Del
GUDJONSSON KARL FREDRIK	Use	Ed	Del
HABBE LISA	Use	Ed	Del
HELLSING MATTIAS	Use	Ed	Del
HERO EK PONTUS	Use	Ed	Del
HÄKANSSON FREDRIK	Use	Ed	Del
JOHANNSSEN FABIAN	Use	Ed	Del
JOHANSSON CHRISTOFFER	Use	Ed	Del
KRONKVIST EMMA	Use	Ed	Del
KVIST HANNA	Use	Ed	Del
LESTANDER TIM	Use	Ed	Del

Optimized group formation - Copyright (c) 2016 Kaj Holmberg

Choose group for AL-MUFTI SUMMIA (asp: 99.0 1.0 92.0 ) prev groups: 0 0 0 0 2  
Suggested group: 1

Group 1	Group 2	Group 3	Group 4	Group 5	Group 6
Number of persons: 0	Number of persons: 0	Number of persons: 0	Number of persons: 0	Number of persons: 0	Number of persons: 0
Sum of asp 1: 0.00	Sum of asp 1: 0.00	Sum of asp 1: 0.00	Sum of asp 1: 0.00	Sum of asp 1: 0.00	Sum of asp 1: 0.00
Sum of asp 2: 0.00	Sum of asp 2: 0.00	Sum of asp 2: 0.00	Sum of asp 2: 0.00	Sum of asp 2: 0.00	Sum of asp 2: 0.00
Sum of asp 3: 0.00	Sum of asp 3: 0.00	Sum of asp 3: 0.00	Sum of asp 3: 0.00	Sum of asp 3: 0.00	Sum of asp 3: 0.00
Sum of repetition: 0.00	Sum of repetition: 0.00	Sum of repetition: 0.00	Sum of repetition: 0.00	Sum of repetition: 0.00	Sum of repetition: 0.00
Total penalty: 211.00	Total penalty: 211.00	Total penalty: 211.00	Total penalty: 211.00	Total penalty: 211.00	Total penalty: 211.00
Total cost: 0.00 Pers: 40. In group: min 6, max 7. Date: 2017-05-30					
Neighborhood: Move Swap Update					
Weight of number: 10	Weight of Aspect 1 1	Number of groups: 6			
<input type="button" value=""/>	<input type="button" value=""/>	<input type="button" value=""/>			
Weight of prev groups: 100	Weight of Aspect 2 10	<input type="button" value=""/>			
<input type="button" value=""/>	<input type="button" value=""/>	<input type="button" value=""/>			
Weight of Aspect 3 1					
<input type="button" value=""/>					
Read data from file/home/ka/tcl/basgr/16-vt-it/ft6-vt16.dat					

## Vigopt: students allocated by heuristic

# Viseopt

Filled In

	1	2	3	4	5	6	7	8	9
1			4	8					
2		7	9				5		
3	6		1					2	
4				5	9				
5		8	1						
6	2				6	7			
7		5		7			6		
8				9					
9		1		4		3	9		

Num for row and col      Cols for num and row      Rows for num and col

1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
2 3    6 5 3 3 4 1	5 4    2 2 3 4    1	1        3 3 4 3 6 1
2        3 5 4 4    5 2	3 3    3 4    4 6 4 2	3 2 4 6 6 3 3    2
3        4 4 4 3 4    3	6 5 5 7 6 6 4 5    3	6 7 3 6 7 6    4 5 3
4 3 3 3 4 5    5 4	4 4 5 7 4 4 5    4	4 5    5 4 5 4 6 4
5 5    5 5 3 3 3 5	4 4    4 3    3 2 5	2 4    3 6 4    1 5
4 1 3 5    4 4 6	3 4    3 3    4 2 6	3 3 2 4    3 4 6
4 5    2 4 4    3 7	3    3 4 2    4 3 7	4    3 2    4 4 2 7
4 5 4 4 6    4 5 5 8	3 3 3    3 5 6 4 8	4 3    4 3 4 4 5 8
2 3    4 3    3 9	2 3    3 3 2    9	2 3    2 4    2    9

Do one row   Do one col   Do one sq   Do one num   Do one   Do all   Do two test   Do ext test

Get numbers   Read   Save   Count   Clear   Quit   Run GLPK   Check   Info    Alt disp

Num: None 1 2 3 4 5 6 7 8 9 :

Row: None 1 2 3 4 5 6 7 8 9 :

Col: None 1 2 3 4 5 6 7 8 9 :

SqCol: None 1 2 3 :   SqRow: None 1 2 3 :

Filled: 23 (of 81)   Checking Indata feasibility: Indata feasible

# Viseopt: one step

Filled In

1	2	3	4	5	6	7	8	9
		4	8			7		
2		7	9			5		
3	6		1				2	
4	1				5	9		
5		8	1					
6	2			6	7			
7		5	7		6			
8			9			5		
9		1	4		3	9		

Num for row and col      Cols for num and row      Rows for num and col

1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
2 3    6 4 3 2    1	4 4    2 3 3    1	3 3    3 3 4 3 3 1
2    3 5 4 4    5 2	3 3    3 4    4 6 4 2	3 2 4 6 6 3 3    2
3    4 4 4 3 3    3	5 5 5 6 6 6 4 5    3	5 7 3 6 7 6    4 4 3
3 3 3 4 5    4 4	4 4 4 7 4 4 4    4	3 5    5 4 5 4 5 4
5 5    5 5 3 3 3 5	4 4    4 3    2 5	2 4    2 5 4    5
4 1 3 5    4 4 6	2 4    3 3    4 2 6	3 3 2 4    3    3 6
4 5    2 4 4    3 7	2 3 2    3 3 7	3    3 2    3    2    7
4 5 4 3 5    4 5 8	3 3 3    3 5 5 4 8	4 3    4 3 4 4 4 8
2 3    4 3    3 9	2 3    3 3 2    9	2 3    2 4    2    9

Do one row   Do one col   Do one sq   Do one num   Do one   Do all   Do two test   Do ext test

Get numbers   Read   Save   Count   Clear   Quit   Run GLPK   Check   Info    Alt disp

Num: None 1 2 3 4 5 6 7 8 9 :

Row: None 1 2 3 4 5 6 7 8 9 :

Col: None 1 2 3 4 5 6 7 8 9 :

SqCol: None 1 2 3 :   SqRow: None 1 2 3 :

Filled: 26 (of 81)   Do a one pass col fixation: Fixed position (1,9) to 7

# Viseopt: full solution by GLPK

Filled In

1	2	3	4	5	6	7	8	9	
1	5	2	4	8	9	1	6	3	7
2	3	7	9	6	2	4	8	5	1
3	6	8	1	5	3	7	9	4	2
4	1	4	6	7	8	2	5	9	3
5	7	9	8	1	5	3	4	2	6
6	2	5	3	9	4	6	7	1	8
7	9	3	5	2	7	8	1	6	4
8	4	6	7	3	1	9	2	8	5
9	8	1	2	4	6	5	3	7	9

Num for row and col

1	2	3	4	5	6	7	8	9
1	4	4		2	3	3	1	
2	3	3	2	3	5	4	2	
3	4	5	5	4	4	4	3	
4	4	4	4	7	4	4	4	
5	3	4	4	3		2	5	
6	2	4	3	3		4	2	6
7		2	3	2		3	3	7
8		3	3	3	5	5	4	8
9		2	3	3	2		2	3

Cols for num and row

1	2	3	4	5	6	7	8	9
1	4	4		2	3	3	1	
2	3	3	2	3	5	4	2	
3	4	5	5	4	4	4	3	
4	4	4	4	7	4	4	4	
5	3	4	4	3		2	5	
6	2	4	3	3		4	2	6
7		2	3	2		3	3	7
8		3	3	3	5	5	4	8
9		2	3	3	2		2	3

Rows for num and col

1	2	3	4	5	6	7	8	9
1		3	3	4	3	3	1	
2	2	4	4	4	3	3	2	
3	4	3	5	6	6	3	3	
4	3	5	4	5	4	5	4	
5	2	3	2	5	4	5	4	
6	3	3	2	4	3	3	6	
7	3	3	2	3	2	7		
8	4	3	4	4	4	4	8	
9	2	3	2	4	2	9		

Do one row | Do one col | Do one sq | Do one num | Do one | Do all | Do two test | Do ext test |  Alt disp

Get numbers | Read | Save | Count | Clear | Quit | Run GLPK | Check | Info

Num: None 1 2 3 4 5 6 7 8 9 :  
Row: None 1 2 3 4 5 6 7 8 9 :  
Col: None 1 2 3 4 5 6 7 8 9 :  
SqCol: None 1 2 3 : SqRow: None 1 2 3 :

Filled: 81 (of 81) Solved problem with GLPK! No unique num fixation! No unique row fixation! No unique col fixation! No unique square fixation!

# Vineopt homepage

Visual network optimization

Visual Linear Programming

Bas	z	x1	x2	x3	x4	x5	b	
	z	1	0	3.000	0	4.000	0	12.000
	x3	0	0	-2.000	1	-2.000	0	-1.000
	x1	0	1	1	0	1	0	3.000
	x5	0	0	1	0	0	1	2.000

Choose entering variable at the top and leaving variable to the left and hit pivot.  
Basic variables: 3 1 5

Pivot

**Vineopt and friends**

VINEOPT is a program for visualization and optimization of graph and network flow problems. VIEOPT is a teaching tool that helps doing pivots in the simplex method. NILEOPT is a program for visualization and optimization of nonlinear optimization problems. The codes are used in university courses given at the Optimization Division of the Department of Mathematics at Linköping University. The main purpose is illustration, but VINEOPT and NILEOPT contains possibilities of running efficient optimization codes.

The main codes are written in Tk/Tk by Kaj Holmberg, but the optimization codes are written in C and Python.

When VINEOPT and NILEOPT are used for academic use, they use optimization codes that are publicly available and free for academic use. There are also several codes made by Kaj Holmberg.

New additions to the family are VIEOPT, which is used for forming student groups, and VIKEOPT, which solves a weighted version of the k-Chinese postman problem. These codes are written in Python, using NumPy, SciPy and

**News**

2015  
VINEOPT, VIEOPT added

October 2015  
New version of VINEOPT.

# Vineopt homepage

\*\*\* New best: 1000.83, cost: 953.99, time: 46.84  
Best obj value: 1000.83, cost: 953.99, time: 46.84 (found at iter 12)  
Lower bound: 981.29

Total cost

Cost

0 5 10 15 20 25 30

1060  
1050  
1040  
1030  
1020  
1010  
1000  
990  
980

Vineopt and friends

VINEOPT is a program for visualization and optimization of graph and network flow problems. VILEOPT is a teaching tool that helps doing pivots in the simplex method. NILEOPT is a program for visualization and optimization of nonlinear optimization problems. The codes are used in university courses given at the Optimization Division of the Department of Mathematics at Linköping University. The main purpose is illustration, but VINEOPT and NILEOPT contains possibilities of running efficient optimization codes.

The main codes are written in Tcl/Tk by Kaj Holmberg, but the optimization codes are written in C and Python.

When VINEOPT and NILEOPT are used for academic use, they use optimization codes that are publicly available and free for academic use. There are also several codes made by Kaj Holmberg.

New additions to the family are VIGEOPT, which is used for forming student groups, and VIKEOPT, which solves a weighted version of the k-Chinese postman problem. These codes are written in Python, using NumPy, SciPy and

News

2015  
VIGEOPT, VIKEOPT added

October 2015  
New version of VINEOPT

The end

Questions?

kaj.holmberg@liu.se