



Optimizing the TracePro Optimization Process

A TracePro Webinar December 17, 2014



Presenter

Presenter

Dave Jacobsen Sr. Application Engineer Lambda Research Corporation

Moderator

Mike Gauvin Vice President of Sales and Marketing Lambda Research Corporation



Format

- A 25-30 minute presentation followed by a question and answer session
- Please submit your questions anytime using Question box in the GoToWebinar control panel





Additional Resources

Past TracePro Webinars

http://www.lambdares.com/webinars

- TracePro Tutorial Videos http://www.lambdares.com/videos
- TracePro Tutorials

http://www.lambdares.com/features/tracepro-tutorials

Information on upcoming TracePro Training Classes
 <u>http://www.lambdares.com/training/software-training</u>



Upcoming TracePro Training

- University of Applied Sciences Jena, Germany
 - Introduction to TracePro Mar. 10 -11, 2014
 - Optimization with TracePro– Mar. 12, 2014
 - Stray Light Analysis Using TracePro Mar. 13, 2015

Littleton, MA USA

- Introduction to TracePro Mar. 23 Mar. 24, 2015
- Optimization with TracePro Mar. 25, 2015
- Stray Light Analysis Using TracePro Mar. 26, 2015
- Scheme Macro Programming Mar. 27, 2015



TracePro 7.5.1

Released December 4, 2014

Customers with current maintenance and support agreements can download this new release at:

http://www.lambdares.com/CustomerSupportCenter/index.php/trace-pro/current-release



Agenda

- Introduction
- The need for an optimization process
- Optimization theory and methods
- Optimization parameters and settings
- Hybrid system optimization example
- Optimization tips
- Review and questions and answers









What is optimization?

 An act, process, or methodology of making something (as a design, system, or decision) as fully perfect, functional, or effective as possible. (Source: Merriam-Webster online dictionary)



What are some of the parameters that can optimized?

- Geometry
- Curvature
- Facets
- Position
- Angle
- Spacing
- Thickness
- Properties











What many people would like to see









What we can try to do











Why do we need an optimizer? - Brute force vs. Optimization algorithm – The goal is to optimize the reflector shown below





Optimization Goal

n: C	:\3D Optimi:	zer				В	Operar	nd list							
ix: L	EDPro							ID T	ype	Opt.	Surf	ace	Range	Weight	Target value
rati	on mode:	Variable S	Scan			✓ Config		01 li	r Profile	Similarity	<u> </u>	Receiver		1.0	{0:0:H:(-0.5,-0.44999998
riab	e list	¥.													
	Included?	Item		Var. type	🖳 Irradiance target profile definer					۵	2		- O X		
		Positi	on-Y g	RelativeVaria								Relative Pos	Value	1	
	V	Positi	on-Z g	RelativeVaria								-0.5000	0.0000		
												-0.4500	0.0000		
						1.0						-0.3330	1.0000		
										T.	TI	0.3330	1.0000		
						0.75						0.4500	0.0000		
												0.5000	0.0000		
						0.5									
_					(0.0)	0.25									
ect	list				Load B	3									
	Output?	ID	Object n	ame Objec	Position: (0,0)	0.0					5				
			Pre-pro	cessor	Direction	-0.5 -0.3	33 -0.16	67 0.	0 0.16	0.333	0.5				
		1	Refle	ctor cRadia	e Horizontal O Vertical O Path										
												Discard	Apply		
												onocaru	CAAD		



Variable range – 40mm in Y-axis and 100mm in Z-axis





Variable range – 40mm in Y-axis and 100mm in Z-axis

Scanning the entire variable range in 0.1mm increments would take $41 \times 101 = 4141$ increments.

If the raytrace time is 1minute per iteration, this would take around 70-hours to complete.





Brute force – Optimization Log after 14-hours of raytracing

ID	Err	Var.	Ti
766	0.2758536	{39,83}	9/27/201
767	0.2559533	{39,84}	9/27/201
768	0.2377989	{39,85}	9/27/201
769	0.2260526	{39,86}	9/27/201
770	0.2079248	{39,87}	9/27/201
771	0.1958354	{39,88}	9/27/201
772	0.1883438	{39,89}	9/27/201
773	0.1705532	{39,90}	9/27/201
774	0.1617426	{39,91}	9/27/201
775	0.1614642	{39,92}	9/27/201
776	0.1550782	{39,93}	9/27/201
777	Invalid	{39,94}	9/27/201





Optimization algorithm – total time of about 2 hours 20 minutes with more rays traced for each iteration - Video











Generally there are 2 types of optimizers: Global and Local.

Global optimizers will search the entire solution space to find the best solution based on the optimization goal or merit function.

Local optimizers will find the solution closest to the starting point of the optimization process. Changing the starting conditions can change the results of the optimization process.



Examples of Global optimization methods include:

- Global Explorer
- Adaptive Simulated Annealing
- Global Synthesis
- Hammer optimization

Global optimization routines will generally have a function to allow them to escape from local solutions and sample more of the solution space in an attempt to find the best overall solution. Lens design programs such as OSLO will typically have global optimization options.





Examples of Local optimization methods include:

- Damped Least Squares
- Powell's Method
- Nelder-Mead or Downhill Simplex Method
- Variable Scanning

Local optimization routines do not have an escape function and will tend to converge on the solution closest to the starting condition. Changing the starting conditions will allow the optimization routine to sample more of the solution space and see if better solutions are available. Illumination design programs such as TracePro will typically feature local optimizers.







Solution Space for optimization problem with possible solutions





Solution Space for optimization problem with possible solutions





The Downhill Simplex, or Nelder-Mead, method for optimization was proposed by John Nelder and Roger Mead in 1965.

The Downhill Simplex method is a local optimization method, meaning it will converge to the solution closest to the starting point. It's possible that a better solution is available. Changing the initial starting conditions can be used as a test to see if a better solution is available. This is a good choice when optimizing geometry, position, and rotation where it is desirable to "jump" around the solution space to find and then refine the best choices for variable values.



The Nelder-Mead method uses the concept of a simplex, which is a special polytope of N+1 vertices in N dimensions. Examples of simplicies include a line segment on a line, a triangle on a plane, and a tetrahedron in 3-dimesional space.

A polytope is a geometric object with flat sides, which exists in any general number of dimensions. A polygon is a polytope in two dimensions, a polyhedron in three dimensions, and so on in higher dimensions (such as a polychoron in four dimensions).



2 Variables = 2 Dimensions & 3 Vertices



3 Variables = 3 Dimensions & 4 Vertices

Source: Wikipedia



A simple example for 2 variables:

For two variables, the simplex is a triangle. The algorithm compares the error function at each vertex of the triangle, rejects the vertex where the error function is highest, and replaces it with a new vertex. This forms a new triangle and the process is repeated.

The process generates a sequence of triangles where the error function at the vertices gets smaller and smaller. The size of the triangles is reduced and the local minimum is found.

The method uses reflection, expansion, contraction, and shrinkage to generate the new vertices.





Optimization Log showing Downhill-Simplex operations – 11 variables

Optimization L	og	
D	En	Var. Time
464	0.4780451	{27,9332301254823.12.134965969301.33.190652547229 9/3/2014.7.45:36 AM
465	0.4540531	27.7945396672225.12.0980833015763.33.15502443564 9/3/2014 7:47:23 AM
466	0.4683346	{27.8026080347074,12.0365509730929,33.13182755939 9/3/2014 7:49:11 AM
467	0.4838998	{27.8534823346508,12.0522831607424,33.11660124533 9/3/2014 7:50:59 AM
468	0.464882	{27.87015321589,12.0973284508893,33.1789312509364 9/3/2014 7:52:46 AM
469	0.4688423	{27.817083855682,12.0731380319655,33.117682405196 9/3/2014 7:54:33 AM
470	0.4846667	{27.7771549884363,12.001132936518,33.132598584553 9/3/2014 7:56:21 AM
471	0.4642632	{27.8974785877721,12.1175908996745,33.16567743186 9/3/2014 7:58:09 AM
472	0.4617609	{27.8056278958183,12.0874355743217,33.15436057044 9/3/2014 7:59:57 AM
473	0.4486159	{27.8029565888829,12.104069911544,33.115670081076 9/3/2014 8:01:45 AM
474	0.4514813	{27.7534100538179,12.1261395485361,33.07791528713 9/3/2014 8:03:33 AM
475	0.4565794	{27.8921623586617,12.1540442705065,33.14045989687 9/3/2014 8:05:22 AM
476	0.4897705	{27.8610439839107,12.0602334156807,33.16605566555 9/3/2014 8:07:09 AM
477	0.4548799	{27.8506787399678,12.1014179519428,33.14418277275 9/3/2014 8:08:58 AM
478	0.4631303	{27.8893213515271,12.0106616844121,33.24869386974 9/3/2014 8:10:45 AM
479	0.4615277	{27.8416130888789,12.0359642431114,33.20157836311 9/3/2014 8:12:33 AM
Graphs Trend cha	art	32.337
Color des. Init. simplex Reflection Expansion Contraction Shrinkage		20.000 10.000 0.449 1 50 100 150 200 250 300 350 400 450 489 Iteration count



Variable Scanning method

- The Variable Scanning method is used to scan or step through all possible variable combinations
- Scanning the range of a variable to find a suitable starting condition for the Downhill-Simplex optimization method
- Moving a variable in fixed interval steps to monitor results
- Tolerancing
- Finding the best surface or material property for a given application by automatically scanning through all properties in a catalog and showing the simulation results for each



Variable Scanning results examples – Selecting the best result by scanning though a catalog of diffuser properties

D	Err		Var.	Time
• 1	5.6260011		{1}	8/28/2014 5
2	9.819279		{2}	8/28/2014 5
3	3.6907483		{3}	8/28/2014 5
4	8.4871414	9	{4}	8/28/2014 5
5	4.4426852		{5}	8/28/2014 5
6	10.2755047	7	{6}	8/28/2014 5
7	12.7635896	5	{7}	8/28/2014 5
 Graphs Tre View item: 	nd chart		12.764	,
Graphs Tre View item: Sum Color des Init. simple Reflection Expansion	nd chart	Error	12.764 7.500	,
Graphs Tre View item: Sum Color des Init. simple Reflection Expansion Contractio	nd chart	Error	12,764 7.500 0 1 2 3 4 5 6	7 8










Variable list

Included?

1

1

1

Item

RedPower

GreenPower

BluePower

Object

<Null>

<Null>

<Null>

Var. type

User-defined

User-defined

User-defined

Value

0.5

0.5

0.5

•

Lo Imt.

0

0

Hi Imt.

1

1

1

Optimization parameters and settings

- Variables
- Optimization operands
- Optimization settings

Random seed = 1000	Generated by timer	
Characteristic length		
Oetermined by ratio of limit	s Ratio= 0.5	
O Determined by length	Value= 0.1	
Maximum iterations number	5000 0.0001	
Model refresh after optimization		
💿 Start point 🛛 🔘 End po	int 🔘 Best point	
	OK Cancel	

Irradiance target profile definer	1.0 0.75 0.5 0.25 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.	Relative Pos. -0.5000 -0.4500 0.4500 0.5000	Value 0.0000 1.0000 0.0000 0.0000
Intrizontal O ventical O Path	(0.504,0.565)		



Variables are the parameters that are allowed to change during the optimization process. These can include:

- Control point position in 1, 2, or 3 dimensions
- Curvature
- Conic Constant
- Rotational Angle
- Distance
- Separation
- Pick-ups
- Custom or User Defined

When the variable is defined the range of the variable is specified. The range is how much the variable will be allowed to "move" during the optimization process. The range of the variable can be set to limit or control the size of the optical element.



Variables can be Absolute, Relative, or Pick-ups

- Absolute variables are defined using absolute or global coordinates of the range of variables motion. If the original variable's location is changed, the range will remain fixed.
- Relative varies are defined relative to current variable's location, so if the variable is moved, the variable range will move with the variable.
- Pick-ups define the position and movement of a variable based on the value of another variable. For example, a variable can be defined as a Pick-up to maintain a constant thickness in a material, or a specific separation between 2 components.



Absolute vs. Relative variable examples





Relative Variable





Absolute Variable





Pick-up variable examples

Use to maintain a constant wall thickness in a reflector





Number of variables to use: Not enough variables example





Tir *

÷

40





Number of variables to use: Too many variables example



D	Err	Var.	Time
334	0.1436474	{29.1461608573769,11.0902660969301,33.27975821109	9/4/2014 6:25:0
335	0.1421101	{29.1399424202344,11.0986100242262,33.27604511825	9/4/2014 6:27:2
336	0.1292306	{29.1498909129324,11.0992001154868,33.27349194726	9/4/2014 6:29:4
337	0.1510294	{29.1446963941115,11.1021197384325,33.28208259461	9/4/2014 6:32:0
338	0.1394355	{29.1503034933705,11.1096671068245,33.27594475247	9/4/2014 6:34:2
339	0.1396255	{29.1561797907375,11.0990652387584,33.27285257455	9/4/2014 6:36:4
340	0.1380887	{29.164383003869,11.0931939614876,33.260210213596	9/4/2014 6:39:0
341	0.13268	{29.1583203165442,11.1069482229099,33.26778794680	9/4/2014 6:41:3
342	0.127192	{29.1511229390619,11.1039724061891,33.27193367268	9/4/2014 6:43:5
343	0.1459687	{29.1372479194852,11.1066713041117,33.27398463063	9/4/2014 6:46:1
344	0.1315886	{29.1599829987355,11.0970550670123,33.26575339439	9/4/2014 6:48:3
345	0.1258516	{29.1694982987483,11.0916292979605,33.27310306415	9/4/2014 6:50:5
346	0.1454274	{29.1856234883798,11.0841082948848,33.27266228091	9/4/2014 6:53:1
347	0.1401825	{29.1627069464358,11.1081584325085,33.26672447034	9/4/2014 6:55:4
348	0.1476395	{29.1447516353192,11.0972113655632,33.27641033173	9/4/2014 6:58:0
		m	•
aphs Trend iew item: um	chart	0.525	
Color des. Init. simplex Reflection Expansion Contraction Shrinkage			







Number of variables to use: Adequate number of variables example



	D	Err	Var.	Tir -
83 0.1676582		0.1676582	{50.7625366907251.28.4959833135397}	9/24/2014
	84 0.1677401		{50.7631887280393,28.4964215771236}	9/24/2014
85 0.1676401		0.1676401	{50.7626997000536,28.4960928794357}	9/24/2014
86 0.1672671		0.1672671	{50.7630257187107,28.4963120112276}	9/24/2014
87 0.1677401		0.1677401	{50.7631887280393,28.4964215771235}	9/24/2014
	88 0.1672892		{50.7632382542312,28.4966294510169}	9/24/2014
89 0.1675911		0.1675911	{50.7632949957996,28.4965802970182}	9/24/2014
90 0.1672671		0.1672671	{50.7630504818067,28.4964159481742}	9/24/2014
91 0.1672672		0.1672672	{50.7628379462863,28.496098508385}	9/24/2014
92 0.1672672		0.1672672	{50.7629380232725,28.4962312440429}	9/24/2014
93 0.1672671		0.1672671	{50.7629442140465,28.4962572282796}	9/24/2014
94 0.1672671		0.1672671	{50.7630381002587,28.4963639797009}	9/24/2014
L.				Þ
Graphs	Trend	I chart		
View it	em:		0.617	
Sum		-		
Color Init. s	r des. simplex		0.400	

Contraction

Shrinkage



Iteration count

Number of variables to use: Adequate number of variables example Model:[Luminaire Example, enough variables, optimized, 2.oml] Irradiance/Illuminance Map:[Luminaire Example, enough variables, optimized, 2.oml] 🕀 🧹 Axrich2-13W LED Total - Illuminance Map for Absorbed Flux Target Receiver Global Coordinates 🗄 🧹 Target E V Reflecto lux 580-551 522 -400 -200 0 200 400 600 493 500 550 464 400 400 500 435-450 300 300 406 400 200 200 377 350 348-(millimeters) 100 100 300 319n 250 290--100 -100 200 261-150 -200 -200 232 100 -300 -300 203-50 174--400 400 145--500 500 -500-500 0 -400 -200 0 200 400 Horizontal 116-Vertical X (millimeters) 87-58-29-Min:64.141. Max:561.43. Ave:381.15 Model Source Luminance Total Flux:381.15 lm, Flux/Emitted Flux:0.38119, 1507680 Incident Rays



Optimization Operands

Optimization operands are used to define the target or goal of the optimization process. Some examples include:

- Flux
- CIE color coordinates
- Irradiance
- Irradiance Profiles
- Intensity
- Candela or Intensity Profiles
- Uniformity
- Beam Width
- User Defined or Custom



Varying the starting point of the optimization process – Initial design and optimization goal







Varying the starting point of the optimization process













Varying the starting point of the optimization process





Varying the starting point of the optimization process



Improve the results by adding a second optimization target – use the lower right corner starting condition from the previous example

Keep the Irradiance Profile target from the previous examples, but add an additional Flux operand with a target goal of 750 lumens.

The two operands can be weighted so that contribution of each can be varied. In this case the were set to have similar contributions to the overall error function.





Improve the results by adding a second optimization target



Initial Optimization

After adding second optimization operand

TracePro

Improve the results by adding a second optimization target



Initial Optimization

After adding second optimization

operand



Optimization Settings

The optimization settings can be used to control how the optimization process runs. Changes in these settings can sometimes result in improvements in the final design. Wrong choices can lead to poor results.

Examples of optimization settings that can be varied include:

- Optimization type
- Characteristic Length Ratio of Limits and Length
- Stopping conditions
- Number of rays traced
- Accurate source model geometric or rayfile



Optimization Method

Choose the optimization method that best suits the application.

•Optimizing geometry or position – choose the Downhill-Simple (Nelder-Mead) method and allow the optimizer to search through a range of variable.



Characteristic Length

The Characteristic Length is an estimate of the size of the solution space for an optimization process. It is used when defining the initial simplex. Each vertex of the initial simplex is a variable set that is a function of the Characteristic Length and a random number.

Random seed = 1000	Ge Ge	enerated by timer
Characteristic length		
Oetermined by ratio of limits	Ratio=	0.5
Determined by length	Value=	0.1
Stop condition		
Maximum iterations number:	5000	
Iteration tolerance:	0.0001	
Model refresh after optimization		
Start point O End point	nt 🔘 Be	st point



Different Characteristic Length Examples





2 Variable Simplex – Iterations





2 Variable Simplex – Iterations













Stopping Conditions

The stopping conditions determine when the optimization process will be considered finished or complete. Possible stopping conditions include:

•Goal is reached – the process stops when the goal is reached

•Number of iterations – the process will stop after a user defined number of iterations

•Iteration tolerance – the process stops when the variation in results from one iteration to the next falls below a certain level



Number of rays to trace

Trace enough rays to get an accurate result in the analysis tools. If too few rays are traced the graphs can be "noisy" and the results will be difficult for the optimizer to interpret.





Accurate Source Model

It is very important to have a source model that is as accurate as possible. Source models can include rayfiles, source property files, and full 3D solid models of the source. A bad source model will lead to poor results.

Some factors to consider in a source model include: size, shape, angular distribution, spatial distribution, spectrum/color, and number of rays.









Hybrid System Optimization Example



Example: Hybrid System – Lens and Reflector

The Goal: Optimize the shape of a side emitting LED lens and reflector combination





Example: Hybrid System – Lens and Reflector




Optimization Goal – Candela profile from 45 to 80 degrees and from -45 to -80 degrees with as little output between those lobes as possible

h: (fix: s erati riab	C:\3D Optimiz SEL ion mode: C Ile list	er Optimization		▼ Config	ID Type Opt. Surface ▶ 01 Can Profile Similarity ▼	Range Weight Exiting ray 1.0	Target value ((-180,-90,-80,-70,-45,45,7
	Included?	Item Position-Y	Object Ctrl Pnt:0@S	🖳 Candela target definer			
		Position-Z	Ctrl Pnt:0@S	Profile chooser	V Symmetric input		
		Position-Y Position-Z	Ctrl Pnt:1@S	45.0 215.0		Angle Value -180.0000 0.0000 -90.0000 0.0000	
				50.0		-80.0000 1.0000 -70.0000 1.0000 -45.0000 0.0000 45.0000 1.0000 70.0000 1.0000 80.0000 1.0000	
ject	list			180.0	0.25	90.0000 0.0000	
	Output?	ID Ob	ject name e-processor	Selected azimuth: 0			
		2	Lens c	Plothing			
		3	Object 3 c	Rectangular	(57.379,1.428)	scard Apply	Start



TracePro



Example: Hybrid System – Lens and Reflector

Candela Profile-Before and after optimization



Before optimization



Lens Profile– Before and after optimization



Before optimization



Add a reflector to the lens assembly









Optimization Goal – Uniform Candela Profile from +/- 20-degrees falling to zero at +/- 25-degrees





Optimization Log

ID	Err	Var.	Tir	
119	0.0199202	{1.79053731687621,5.54612160714186,4.832046127497	9/26/201	
120	0.0171559	{1.77307953397855,5.52386735434377,4.852554466930	9/26/201	
121	0.0176932	{1.77305807210709,5.5967518156108,4.7705382818242	9/26/201	
122	0.0169368	{1.77505933562437,5.60028714794389,4.778938847050	9/26/201	
123	0.0169563	{1.77178588756396,5.53571202564896,4.872162501818	9/26/201	
124	0.0151074	{1.76187348191704,5.64456341886273,4.734139584296	9/26/201	
125	0.0150634	{1.74935470309527,5.72861864856312,4.646885729376	9/26/201	
126	0.0189741	{1.76980060711756,5.55723127095197,4.841335458496		
127	0.0158646	{1.77224370585971,5.58687167944609,4.788237575992	9/26/201	
128	0.0165392	<pre>{1.77374977003619,5.68435693016716,4.691596309295 {1.75954519221555,5.66597785342138,4.662218789623</pre>		
129	0.0174064			
130	Evaluating/	{1.77631303136533,5.57602060553131,4.823242331614	9/26/201	
([+	
Graphs Trend View item: Sum Color des. Init. simplex Reflection Expansion	d chart			



Candela Profile-Before and after optimization



Before optimization



Lens Profile– Before and after optimization



Before optimization



The Goal: Optimize Lens and Reflector as a system







Optimization Goal – Uniform Candela Profile from +/- 20-degrees falling to zero at +/- 25-degrees





Optimization Log – Combined optimization

	D	Err	Var.	Time
	222	0.0120452	{2.08826248704555,5.13311571061494,5.436819398779	9/30/2014 12:1
	223	0.0122266	{2.11422807162778,5.17023859308962,5.504443583648	9/30/2014 12:1
	224	0.0117545	{2.09899878725224,5.13194097939899,5.449811736646	9/30/2014 12:1
	225	0.01122	{2.12943392510403,5.16058202728438,5.515604188623	9/30/2014 12:2
	226	0.0112118	{2.1163523874342,5.14563031450146,5.4871029476266	9/30/2014 12:2
	227	0.0104732	{2.15327522409423,5.17799944961345,5.566719753892	9/30/2014 12:2
	228	0.0106691	{2.1294219669029,5.15273677278261,5.5208286149675	9/30/2014 12:3
	229	0.0113296	{2.14544623120845,5.17466424648279,5.555201301969	9/30/2014 12:3
	230	0.0110579	{2.15328595118785,5.18366775069393,5.566270265848	9/30/2014 12:3
•	231	0.010694	{2.1662378707255,5.1999153703691,5.60609151124261	9/30/2014 12:3
	232	Invalid	{2.18451662579519,5.20792422388173,5.633731454263	9/30/2014 12:4
	233	Evaluating/	{2.23759703262766.5.25402127971595.5.744753717731	9/30/2014 12:4
		-		
•			III	+
< Graph	ns Trenc	i chart	III :	+
 Graph View Sum 	ns Trenc v item:	i chart	0.599	
 Graph View Sum Co Ini Re E∖ 	ns Trenc v item: 1 olor des. it. simplex effection	i chart	0.599 0.400 0.200	
 Graph View Sum Co Ini Re E) 	hs Trenc v item: n olor des. it. simplex effection xpansion	j chart		200 231



Candela Profile-Before and after combined optimization



Before optimization



Lens and Reflector Profile – Before and after combined optimization



Before optimization



Candela Profile - 2 different optimization procedures



Separate optimization

Combined optimization



Lens and Reflector Profile – 2 different optimization procedures



Separate optimization

Combined optimization



Photorealistic Rendering – 2 different optimization procedures



Combined optimization

Separate optimization



Photorealistic Rendering – 2 different optimization procedures



Combined optimization

Separate optimization



Optimization Tips

- Start with a good initial design if possible
- Use accurate models including geometry and properties
- Use accurate source models
- Define enough variables so that the model is not over or under constrained
- Set the Characteristic Length to adequately sample the solution space
- Define achievable optimization operands or goals



Optimization Tips

- Trace enough rays so that the analysis maps are not noisy and the optimizer can make accurate decisions
- Change optimization parameters to check for better solutions
- Know the capabilities of your optical analysis and optimization software



Summary and Questions

Software based optimization allows the user to easily search a large range of solutions to find the best result for a given problem:

- ✓ Luminaire design process time can be shortened considerably
- Designs can be tested "virtually", cutting down on the need for physical prototypes
- \checkmark A large number of solutions can be searched in a short period of time
- In addition to geometric shape optimization can also include position, rotational angle, and properties
- \checkmark Tolerancing can also be accomplished

For more information or to sign up for our free 30-day trial please visit us at:

www.lambdares.com

Phone: +1 978-486-0766 E-mail: sales@lambdares.com

