Niches of Population-Based Optimization Algorithms in Practice

Kalyanmoy Deb
Koenig Endowed Chair Professor
Michigan State University
East Lansing, USA
kdeb@egr.msu.edu
http://www.coin-laboratory.com

Thanks to IEEE Distinguished Lecture program

Overview

- Point versus population based optimization approaches
  - Advantages
  - Disadvantages
- Evolutionary Algorithms as Population-Based methods
- Niches of EAs
- Customization is the key for complex problems
- An extreme-scale assignment problem
- Conclusions

Optimization Methods: Point and Population Based

Point-Based

Nonlinear Programming

$$\nabla f(x) - \sum_{j=1}^{g} u_j \nabla g_j(x) - \sum_{k=1}^{h} v_k \nabla h_k(x) = 0$$

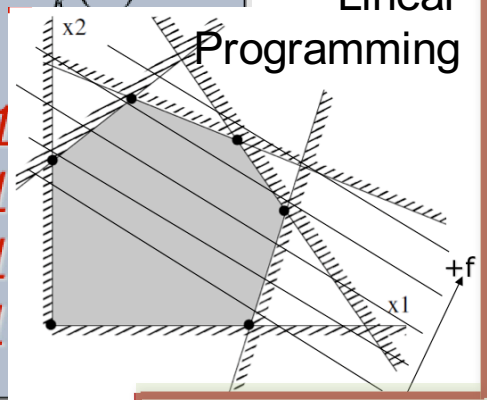
$g_j(\mathbf{x}) \geq 0 \quad j = 1$

$h_k(\mathbf{x}) = 0 \quad k = 1$

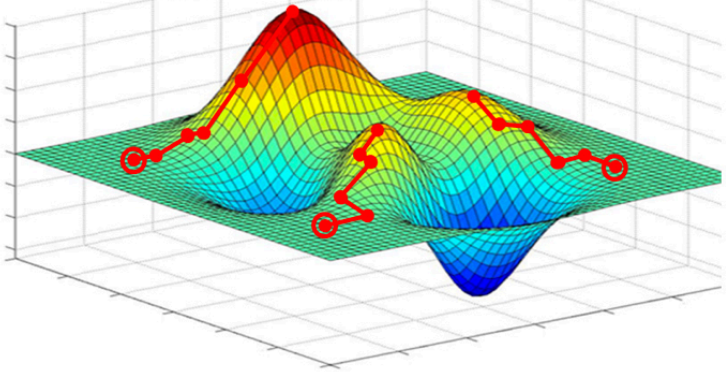
$u_j g_j(\mathbf{x}) = 0 \quad j = 1$

$u_j \geq 0 \quad j = 1$

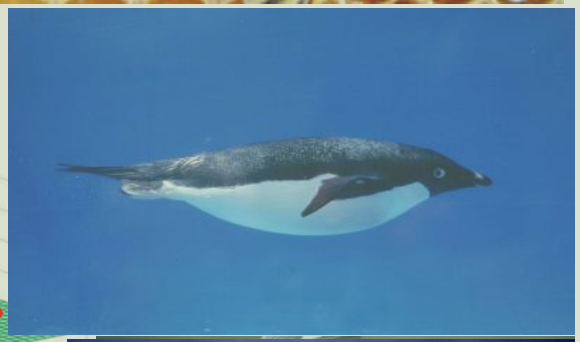
Linear Programming



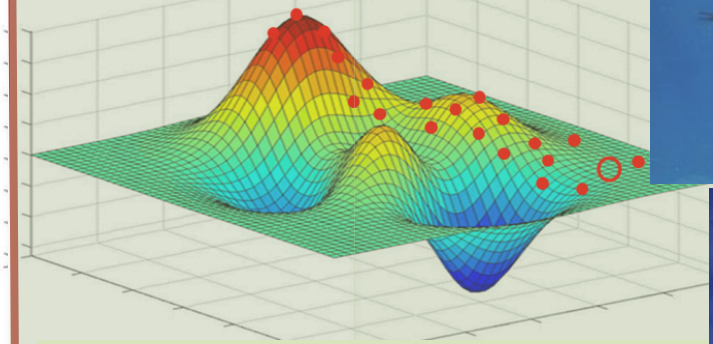
Math. optimization



Population-Based



Evolutionary optimization



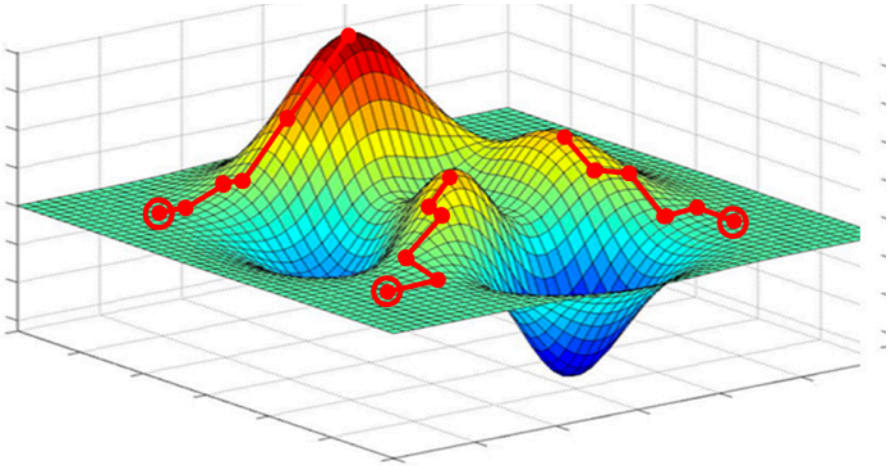
Other Meta-heuristics based methods

Point-Based Optimization Methods

1. Start with one point, $\boldsymbol{x}$
2. Update using a transition rule (T)
   a. $\boldsymbol{y} = T(\boldsymbol{x})$
3. Compare $\boldsymbol{y}$ with $\boldsymbol{x}$, if better replace: $\boldsymbol{x} \leftarrow \boldsymbol{y}$
4. Move to Start, until termination criterion is met

- Not much memory
- Local search
- No parallel processing
- Easier to do theory

"One hammer for multiple nails"

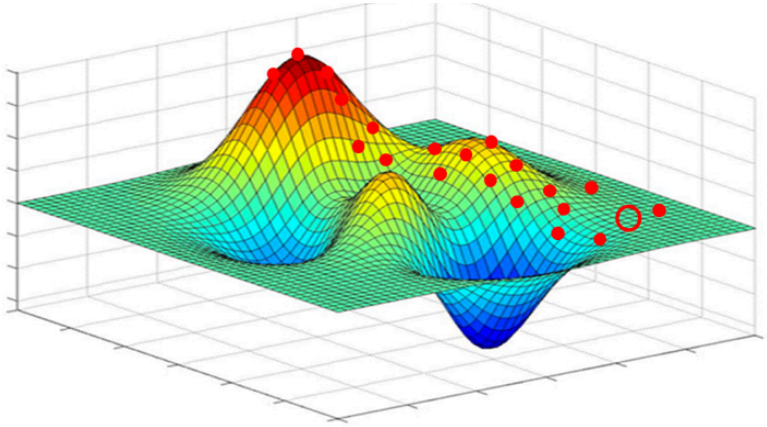


Population-Based Optimization Methods

1. Start with a population of points, $\boldsymbol{P(x)}$
2. Update using a transition rule (T)
   a. $\boldsymbol{Q(y)} = T(\boldsymbol{P(x)})$
3. Compare $\boldsymbol{Q(y)}$ with $\boldsymbol{P(x)}$ and replace: $\boldsymbol{P(x)}$ ← $\boldsymbol{(P(x),Q(y))}$
4. Move to Start, until termination criterion is met

- Requires memory
- Global search
- Parallel processing
- Not easy to do theory

"Different hammers for multiple nails"



No Free Lunch (NFL) Theorem

- In the context of *optimization*
  - Wolpert and McCardy (1997)
  - Algorithms A1 and A2
  - All possible problems **F**
  - Performances P1 and P2 using A1 and A2 for a fixed number of evaluations
  - P1 = P2
- NFL breaks down for a narrow class of problems or algorithms
- Research effort: Find the best algorithm for a class of problems
  - Unimodal, multi-modal, quadratic etc.

Current Status, Facts, and Myths

- Point-based methods:
  - Fast, local, theory-based, successful in deterministic problems, hard to modify and suit a problem
- Population-based methods:
  - "Slow", global, parallel, flexible, great potential for complex problems, easy to customize
- NFL theorem suggests a single algorithm cannot perform well on all problems
- Practical problems are complex in many ways
- Customization is the key
  - Initial knowledge and/or Derived knowledge

Population-Based Algorithm Generator

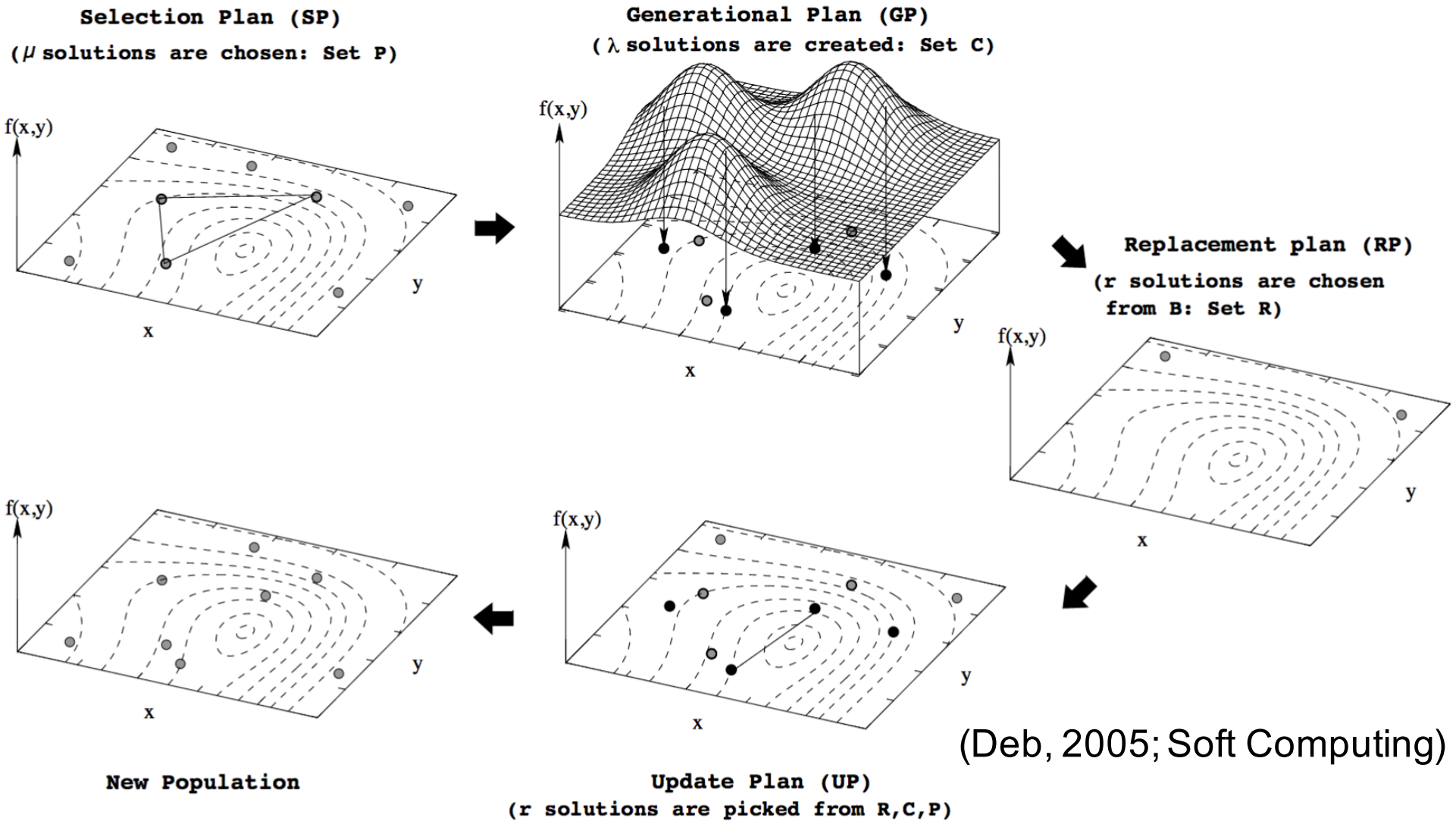
(Deb, 2005; Soft Computing)

At iteration *t*, Population *P(t)* of size *N*

1. Select the best *parent* and μ-1 other parents randomly
2. Generate λ *offspring* using a *creation* scheme
3. Choose *r* parents at random from *P(t)*
4. Form a combination of $r+\mu$ parents and λ offspring, choose best *r* solutions and replace the chosen *r* parents in Step 3 to update *P(t)*

- Requires a parametric study with μ, λ, *r* and *N*

Population-based Algorithm Generator: An Illustration

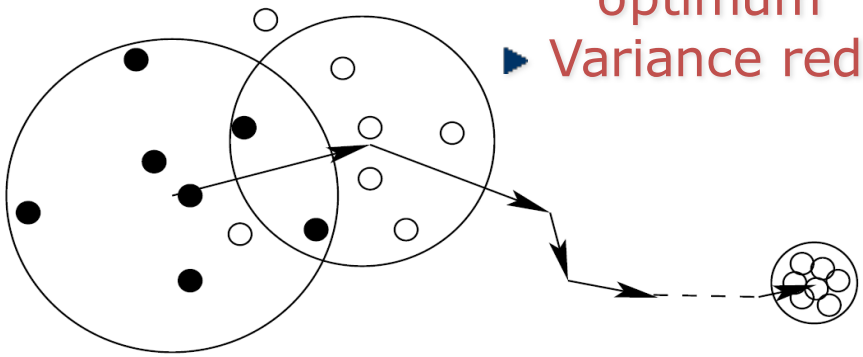


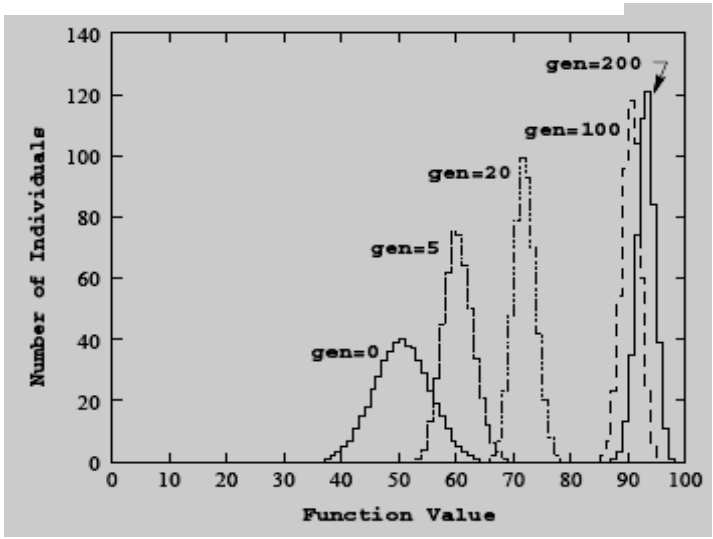
(Deb, 2005; Soft Computing)

Evolutionary Algorithm as Population-Based Optimization Method

- Mean approaches optimum
- Variance reduces

```
begin
Solution Representation
t := 0;  // generation counter
Initialize P(t);
Evaluate P(t);
while not Terminate
do
   P'(t)  := Selection (P(t));
   P''(t) := Variation (P'(t));
   Evaluate P''(t);
   P(t+1):= Survivor (P(t),P''(t));
   t := t+1;
od
end
```





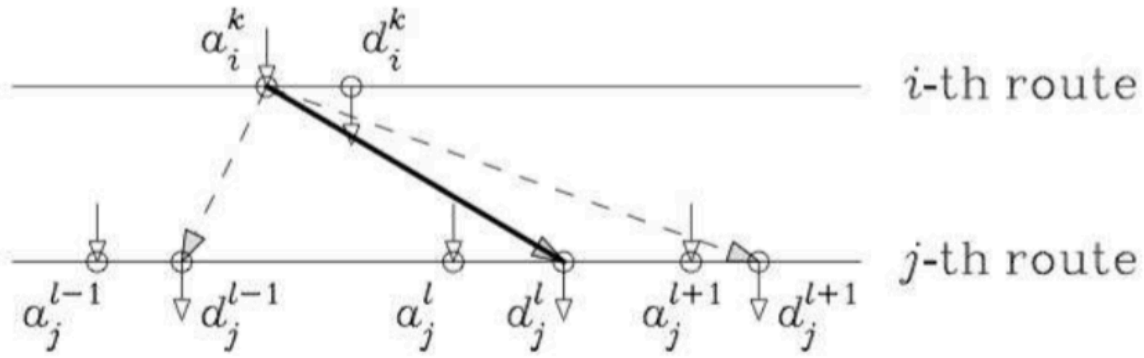
Strengths of Evolutionary Algorithms

- Population: Store a diverse set of solutions, global perspective, implicit parallelism, search for multiple solutions
- Initialization: Initial supply of diverse solutions
- Selection: Provides directions for search
- Recombination: Combines two or more evolving members to produce new and better solutions, *unique operator*
- Mutation: Local perturbation to find better solutions
- Modular, but each must work in unison with others
- Highly parallelizable
- Keep the essence, but customize for a complex problem

Niches of Population-based Algorithms

- Flexibility in representation of solutions
  - Avoids unnecessary fix-ups (two examples next)
  - Representation-Recombination dual is important
- For example, $x_1 \geq x_2 \geq \ldots \geq x_n$
  - Use $(x_1, p_2, p_3, \ldots, p_n)$, where $p_i = x_i / x_{i-1}$ and $p_i \in [0, 1]$ ensures $x_2 \leq x_1$
- For example, $x_1 + x_2 + \ldots + x_n = 1$
  - Update: $x_i \leftarrow x_i / \sum_j x_j$, constraint is always satisfied
- Choose one from n options: Boolean variables ($\boldsymbol{x}$)
  - Fix-up: $\sum_j x_j = 1$

Transit Network Optimization



(Chakroborty et al., 1995)

$$\text{Minimize} \quad \sum_i \sum_j \sum_k \sum_l \delta_{i,j}^{k,l}(d_j^l - a_i^k)w_{i,j}^k$$

$$+ \sum_i \sum_l \int_0^{a_i^k - a_i^{k-1}} v_{i,k}(t)\Big[(a_i^k - a_i^{k-1}) - t\Big]\, dt$$

subject to

$$s_{\max} - (d_i^k - a_i^k) \geq 0 \qquad \text{for all } i \text{ and } k,$$

$$(d_i^k - a_i^k) - s_{\min} \geq 0 \qquad \text{for all } i \text{ and } k,$$

$$T - (d_j^l - a_i^k)\delta_{i,j}^{k,l} \geq 0 \qquad \text{for all } i,\ j,\ k \text{ and } l,$$

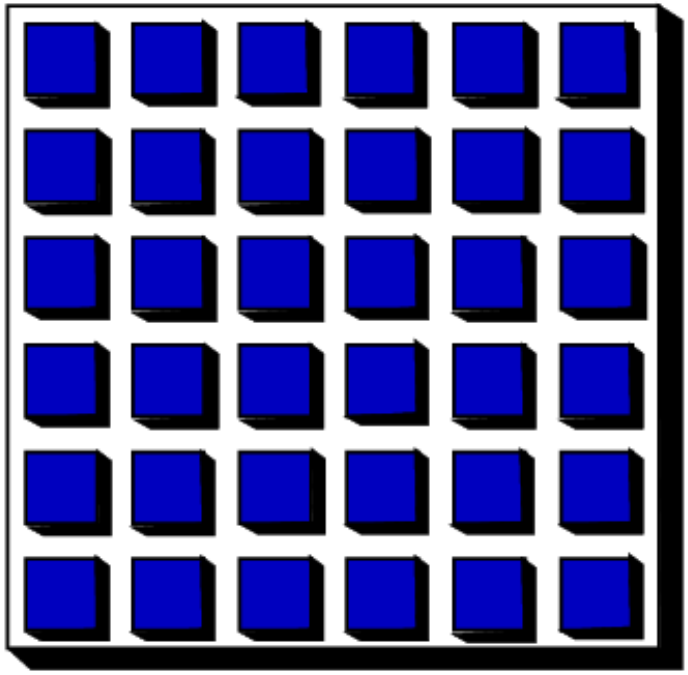
$$(d_j^l - a_i^k) + M(1 - \delta_{i,j}^{k,l}) \geq 0 \qquad \text{for all } i,\ j,\ k \text{ and } l,$$

$$h_i - (a_i^{k+1} - a_i^k) \geq 0 \qquad \text{for all } i \text{ and } k,$$

$$\sum_l \delta_{i,j}^{k,l} = 1 \qquad \text{for all } i,\ j \text{ and } k.$$

- $\delta_{i,j}^{k,l}$ is Boolean
- Difference between departure and arrival time must be +ve for the actual case, don't care for all other cases!

Innovative Representation



(Deb et al., 2003, IEEE)

- Permutation as fixed coding
- Permutation: (a-b-c-d-e)
- Fixed coding:
- (0, 0-1, 0-2, 0-3, 0-4)
- Example: (0 1 0 2 2)

_a_-> _c_a_b_ -> _c_a_d_b_-> (c-a-e-d-b)

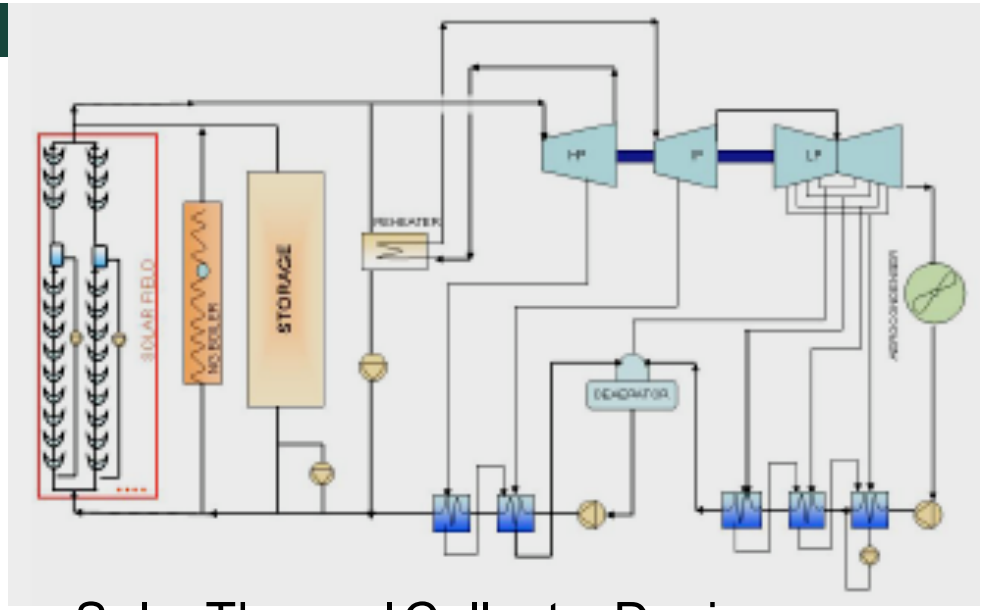
- Allows any crossover to be used
- Valid permutations are created

Niches of Population-based Algorithms (cont.)

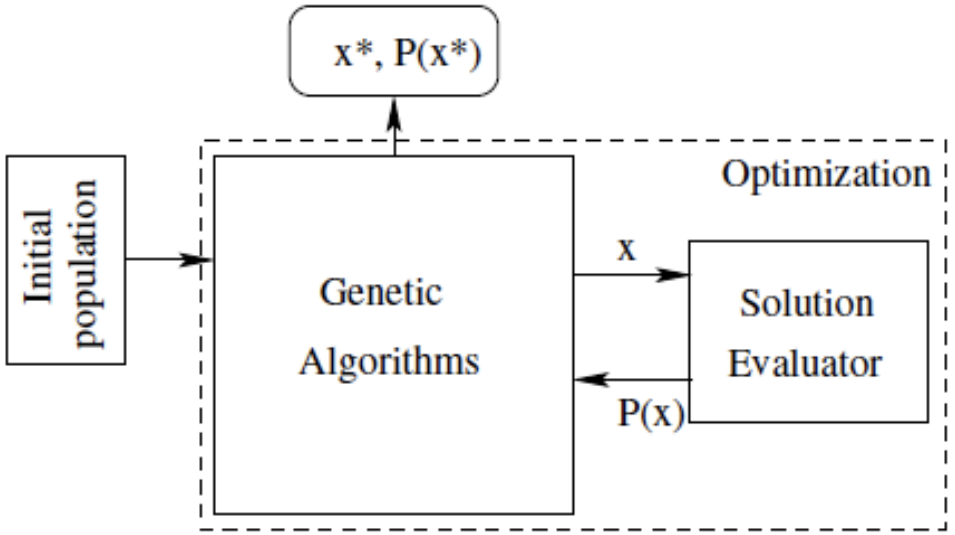
- Algorithmic flexibility
  - New problem-specific operators can be introduced
- Population approach
  - Better chance of finding global optimum
  - Potential to find and store multiple solutions
    - Multi-modal and multi-objective problems
- Direct approach (no need for gradients)
- Procedural objective and constraint functions
  - If-then-else or other procedures
- Parallel implementation

Black-box Optimization

- Maximize profit for operating a solar plant
  - Three variables
  - No mathematical description
  - Noise, isolation, multi-modality, and boundary solution
- LINDO, LGO software failed

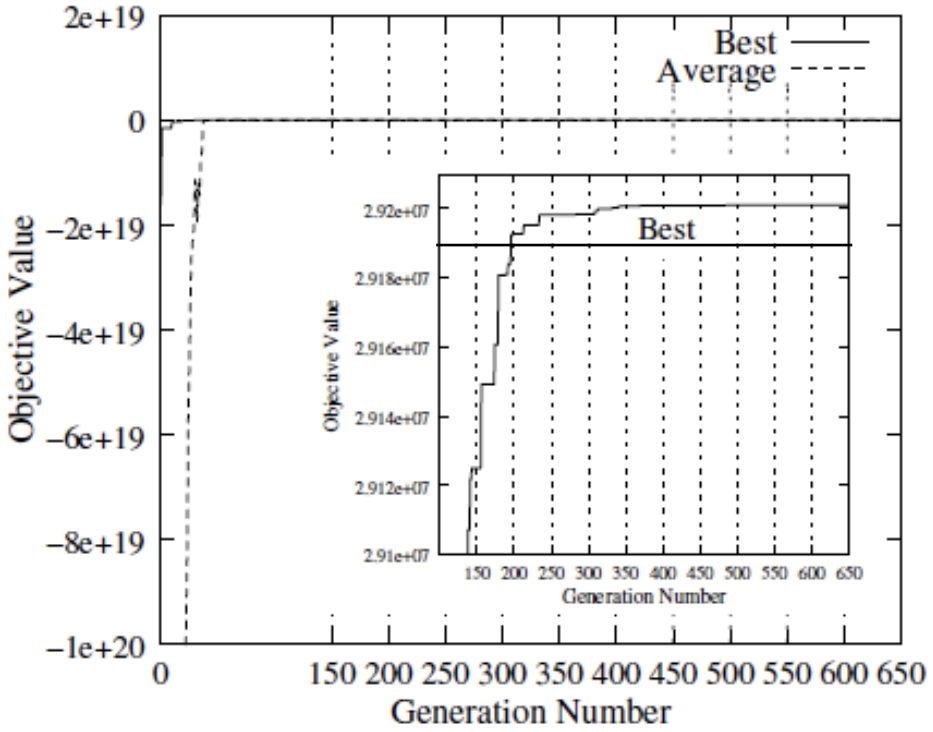


Solar Thermal Collector Design
ENDESA, Spain (Cabello et al., 2011)



Customized GA

- Gradual improvement
- Multiple runs are consistent

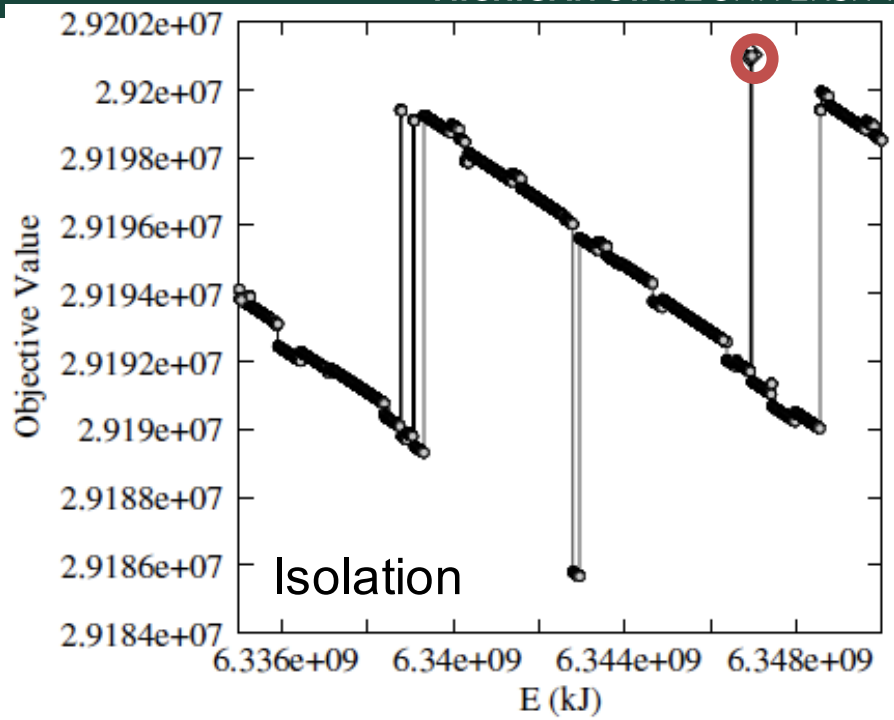


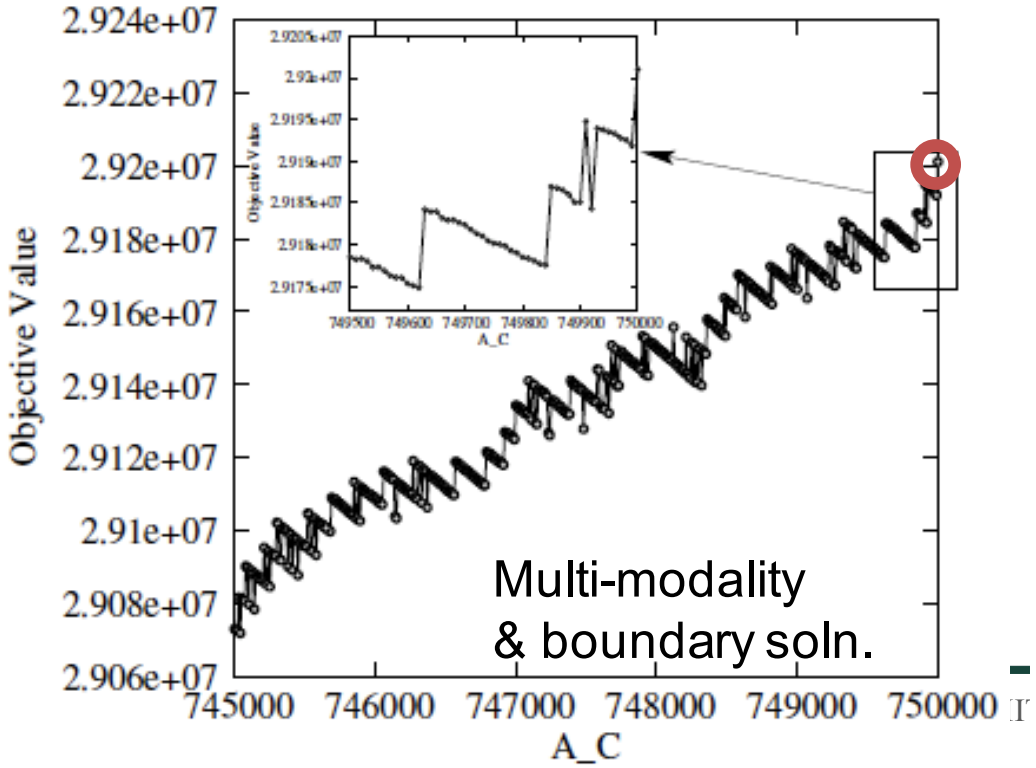
$P(\mathbf{x}^*)$ (€)	$\mathbf{x}^*$		
	A_C (m^2)	E (kJ)	P_{AUX} (kW)
29201019.61	749999.99	6346926947.98	92768.27
29201018.75	749999.94	6346929408.75	92768.26
29201018.62	749999.94	6346929669.93	92768.26
29201018.62	749999.94	6346929478.79	92768.27
29200967.67	749997.61	6347055515.19	92768.46
29200967.42	749997.62	6347055515.19	92768.46
29200957.23	749997.01	6347087739.59	92768.27
29200953.07	749997.00	6347092303.42	92768.31
29200922.51	749999.96	6346929711.66	92777.49
29200522.97	749992.62	6318673893.16	92768.70

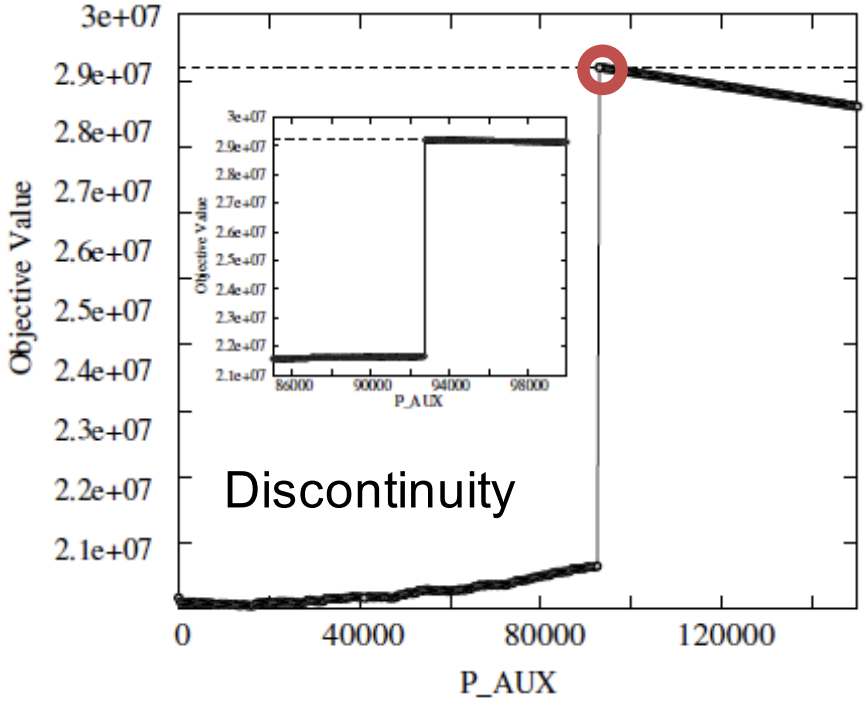
- Real-parameter GA
  - Popsize=50,
  - maxgen=150
  - P_c=0.9
  - P_m=0.333
- 10 runs
- 0.0017% difference between runs

Difficulties Negotiated Well!

One practical problem demonstrates different difficulties



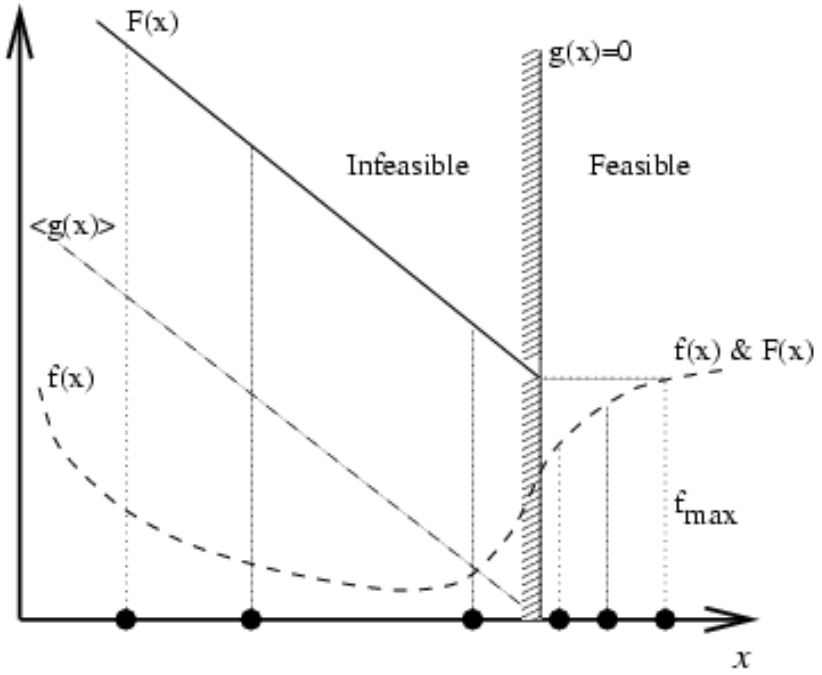




Constraint Handling

(Deb, 2000, CMAME)

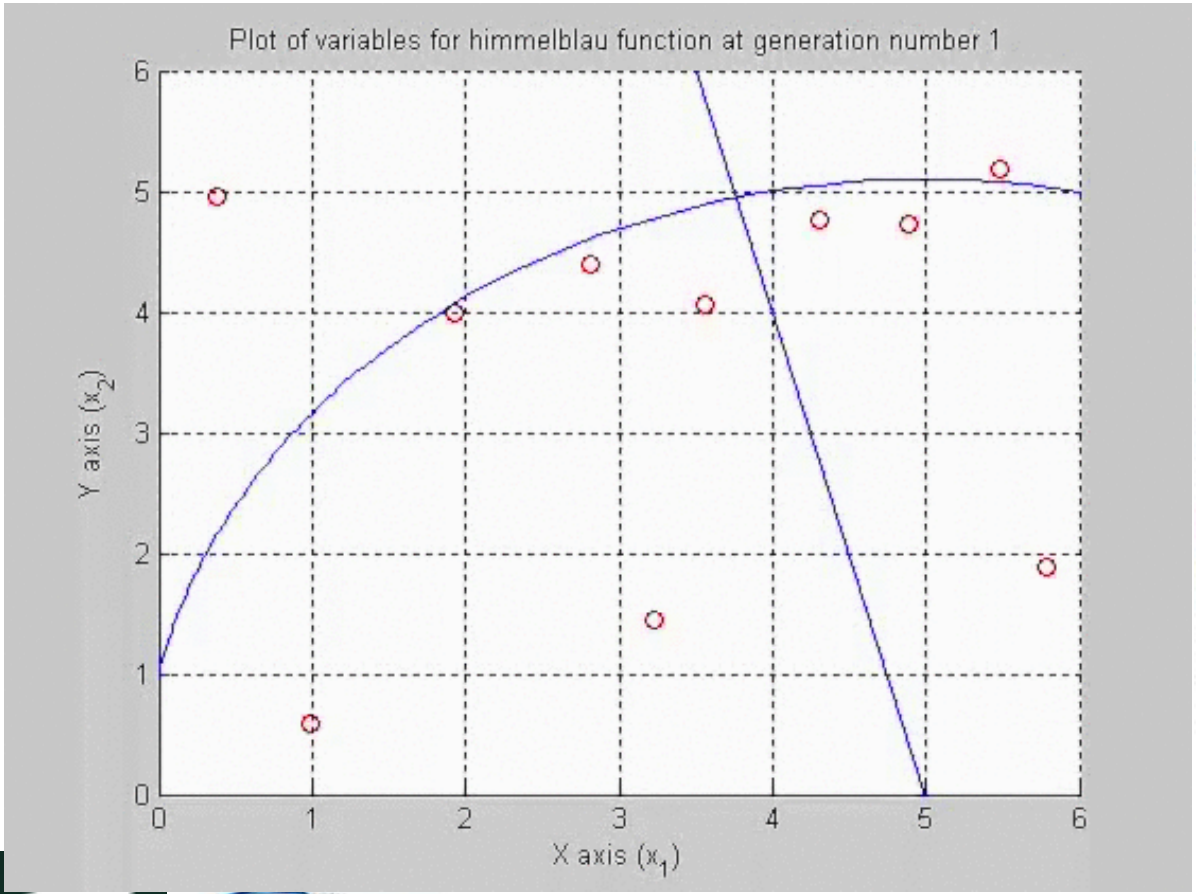
- Modify tournament sel.:
  - A feasible is better than an infeasible
  - For two feasibles, choose the one with better *f*
  - For two infeasibles, choose the one with smaller constraint violation $\left(\sum_j \langle g_j(x) \rangle\right)$

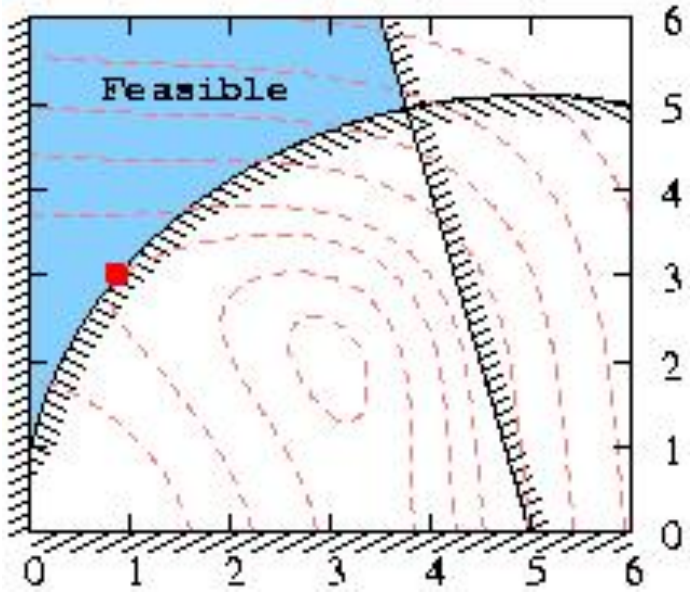


$$F(\vec{x}) = \begin{cases} f(\vec{x}) & \text{if } g_j(\vec{x}) \geqslant 0 \quad \forall j = 1, 2, \ldots, m, \\ f_{\max} + \sum_{j=1}^{m} \langle g_j(\vec{x}) \rangle & \text{otherwise.} \end{cases}$$

Only possible with a population of points

A Computer Simulation





Test Problem 1

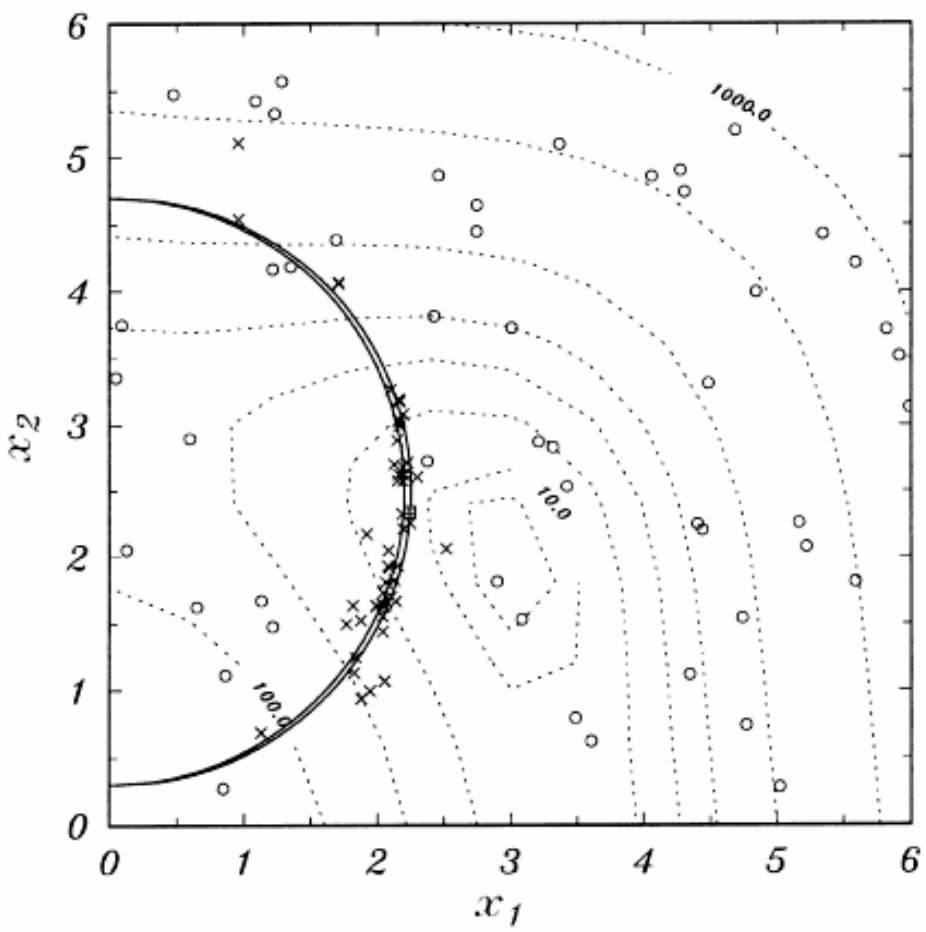
$$\begin{aligned} \text{Minimize} \quad & f_0(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \\ \text{Subject to} \quad & g_1(x) \equiv 4.84 - x_1^2 - (x_2 - 2.5)^2 \geq 0 \\ & g_2(x) \equiv (x_1 - 0.05)^2 + (x_2 - 2.5)^2 - 4.84 \geq 0 \\ & 0 \leq x_1 \leq 6, 0 \leq x_2 \leq 6 \end{aligned}$$

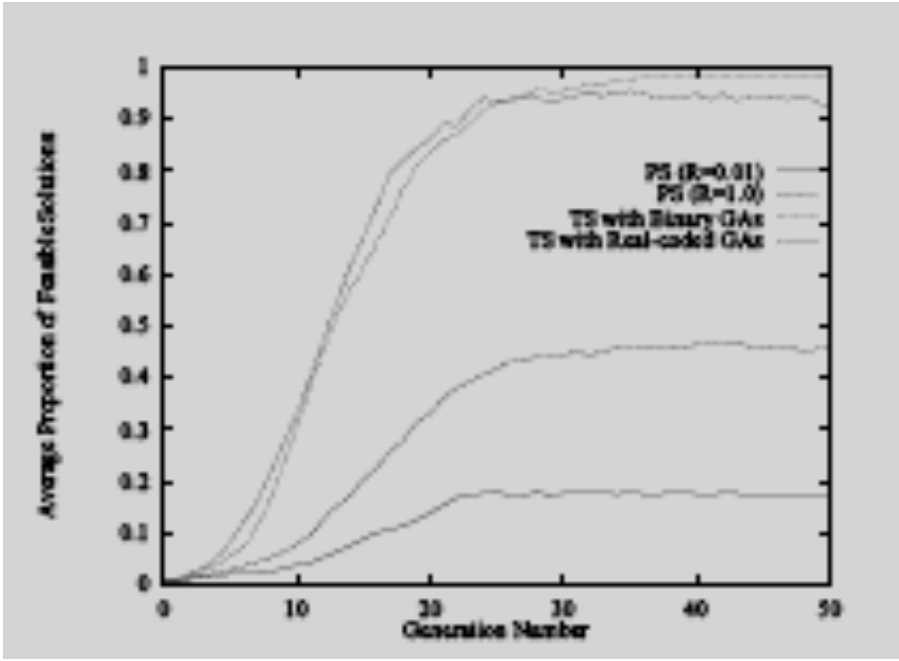
- Opt. Soln.: (2.246826, 2.381865), f=13.59085.
- 0.7% of the search space feasible

Method	ϵ ≤ 1%	≤ 2%	≤ 5%	≤ 10%	≤ 20%	≤ 50%	> 50%	Infea-sible
PS ($R = 0.01$)	5	5	7	8	15	23	8	19
PS ($R = 1$)	17	19	20	20	24	32	7	11
TS-B	2	2	5	6	9	13	37	0
TS-R	29	31	31	32	33	39	11	0

Method	Best	Median	Worst
PS ($R = 0.01$)	13.58958	24.07437	114.69033
PS ($R = 1$)	13.59108	16.35284	172.81369
TS-B	13.59658	37.90495	244.11616
TS-R	13.59085	13.61673	117.02971

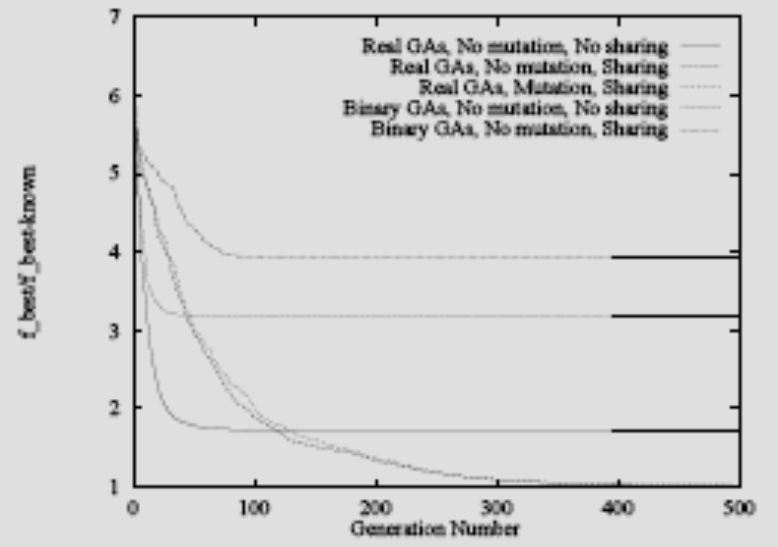
Results on Test Problem 1





Welded-Beam Design Problem

Method	Muta-tion	shar ing	ϵ ≤ 1%	≤ 2%	≤ 5%	≤ 10%	≤ 20%	≤ 50%	> 50%
Maximum generations = 500									
TS-B	No	No	0	0	0	0	0	0	50
TS-B	No	Yes	0	0	0	0	0	0	50
TS-R	No	No	0	0	1	4	8	16	34
TS-R	No	Yes	28	36	44	48	50	50	0
Maximum generations = 4,000									
TS-R	No	Yes	28	37	44	48	50	50	0
TS-R	Yes	Yes	50	50	50	50	50	50	0



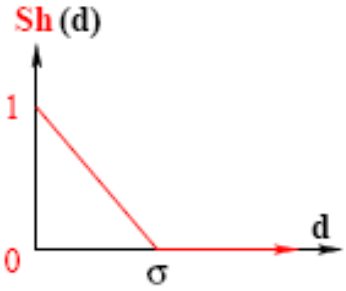
Case	Best	Median	Worst
1	4.939027	7.183082	12.084255
2	3.820984	8.899963	14.298933
3	2.442714	3.834121	7.444246
4	2.381191	2.392892	2.645833
5	2.381187	2.392025	2.645833
6	2.381447	2.382634	2.383554

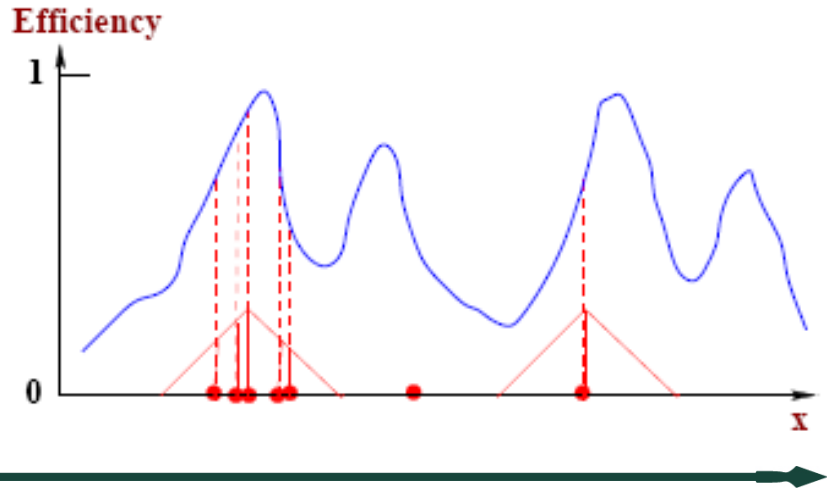
(Deb, 2000)

Multi-Modal Optimization

Multiple niches (human and animal) coexist in nature

- By sharing resources (land, food, etc.)
- How to mimmick the concept in EAs?
- Reduce selection pressure for crowded solutions
- Use a sharing function based on two-armed bandit problem



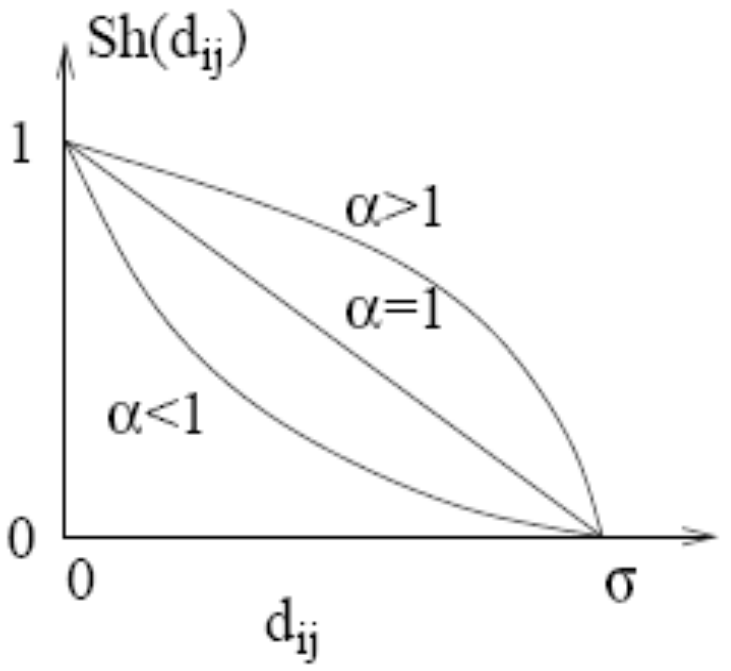


Sharing Function

- Goldberg and Richardson (1997)
- d is a distance measure betwe two solns.
  - Phenotypic distance: $d(x_i,x_i)$, x: variable
  - Genotypic distance:

    $d(s_i,s_i)$, s: string
- Calculate niche count, $nc_i=\sum_j Sh(d_{ij})$
- Shared fitness: $f_i' = f_i/nc_i$
- Use proportionate selection opeartor

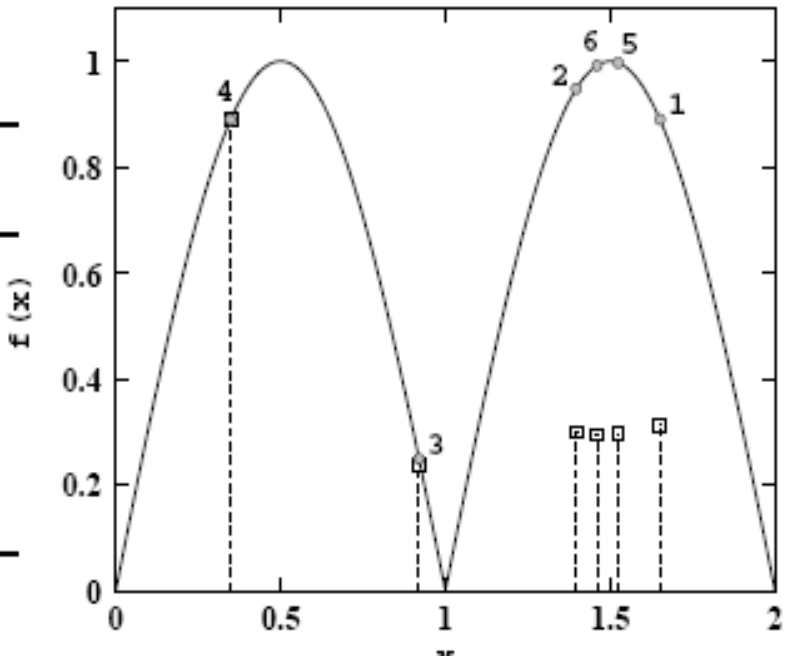
$$Sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma}\right)^{\alpha} & \text{if } d_{ij} \le \alpha, \\ 0 & \text{otherwise.} \end{cases}$$



An Example: Phenotypic Sharing

- Maximize $f(x) = |\sin(\pi x)|$, $x \in [0, 2]$
- $nc_i = \sum_{j=1}^{N} Sh(d_{ij})$

Sol. i	String	$x^{(i)}$	f_i	nc_i	f_i'
1	110100	1.651	0.890	2.856	0.312
2	101100	1.397	0.948	3.160	0.300
3	011101	0.921	0.246	1.048	0.235
4	001011	0.349	0.890	1.000	0.890
5	110000	1.524	0.997	3.364	0.296
6	101110	1.460	0.992	3.364	0.295

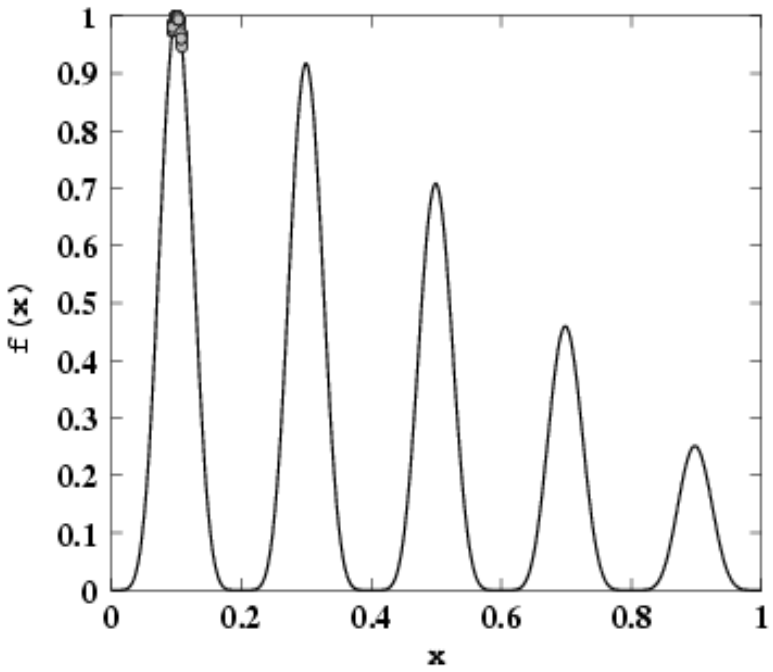


(Deb and Goldberg, 1989)

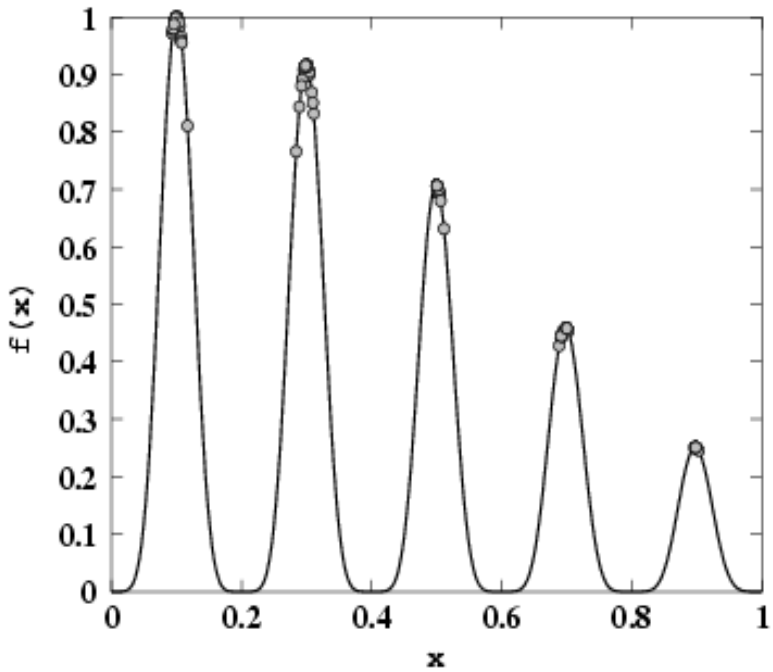
Simulation Results

- Inclusion of niche-formation strategy

Without Sharing



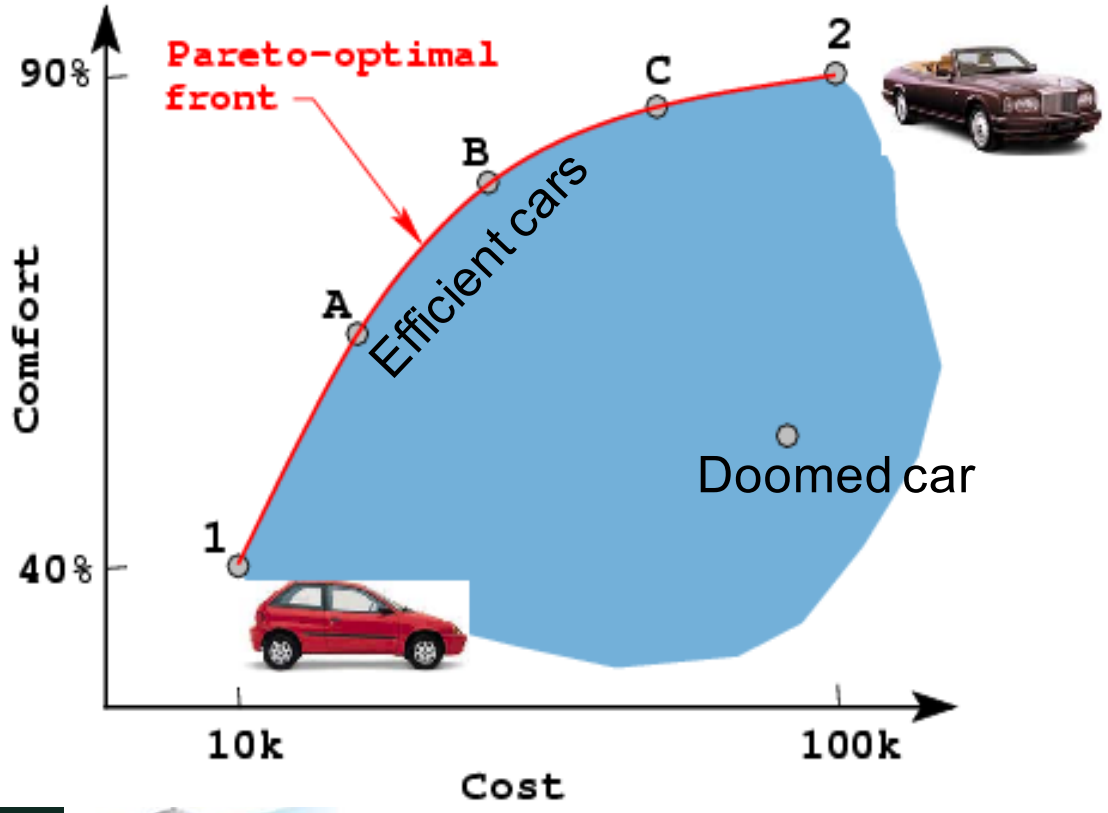
With Sharing



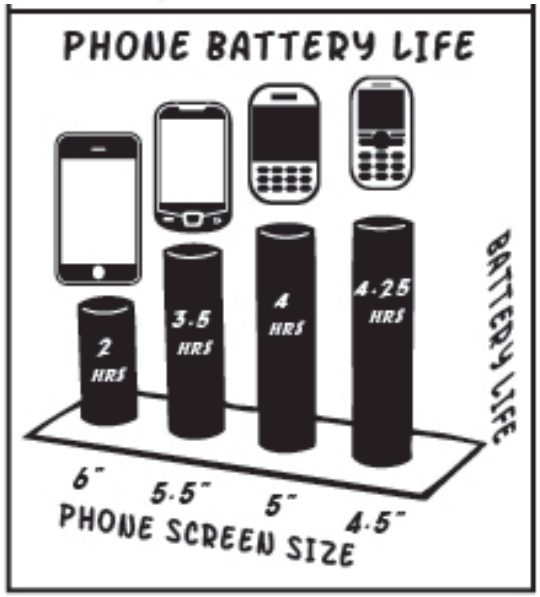
More results in (Li et al., in press, IEEE TEVC)

Multi-Objective Optimization:

Handling multiple conflicting objectives



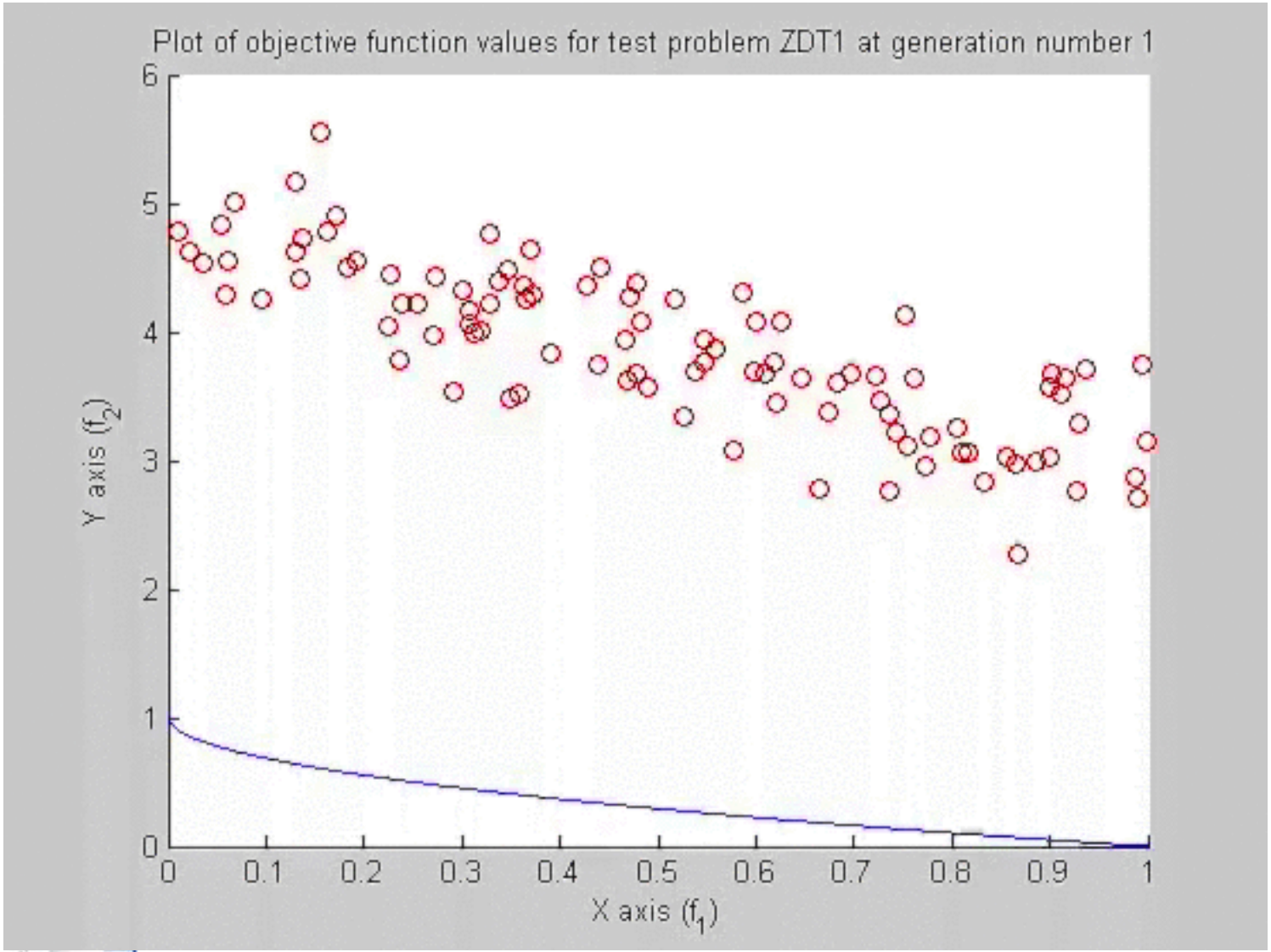
Have you wondered?



Most practical problems are multi-objective in nature

(Deb, 2001, WILEY)

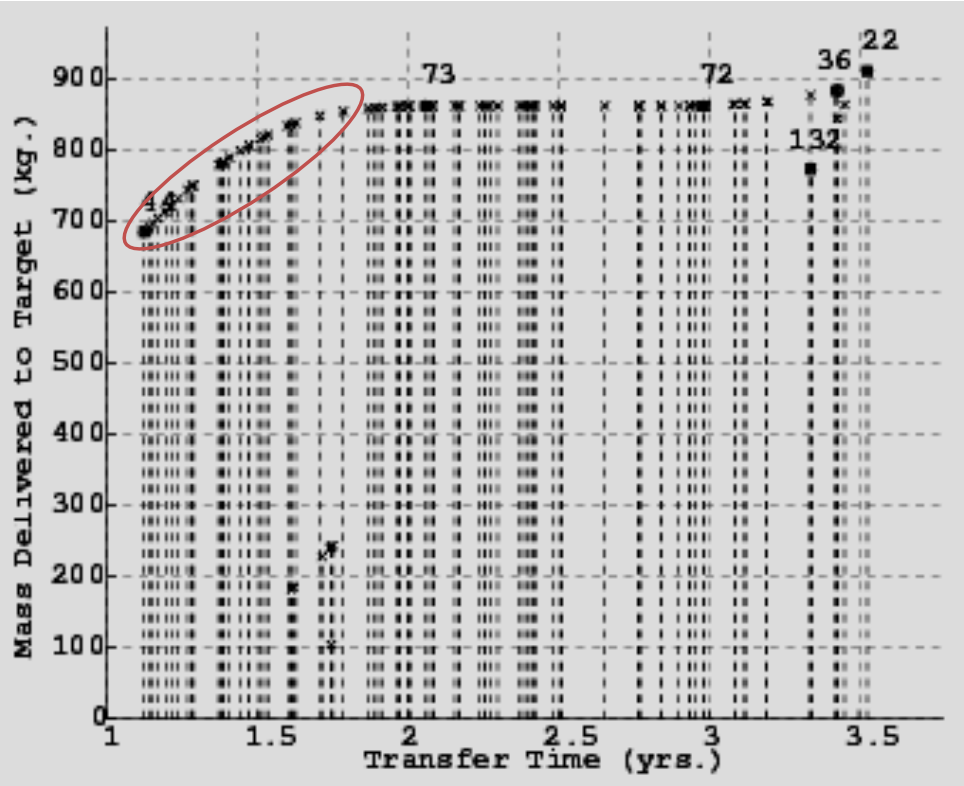
Simulation on ZDT1

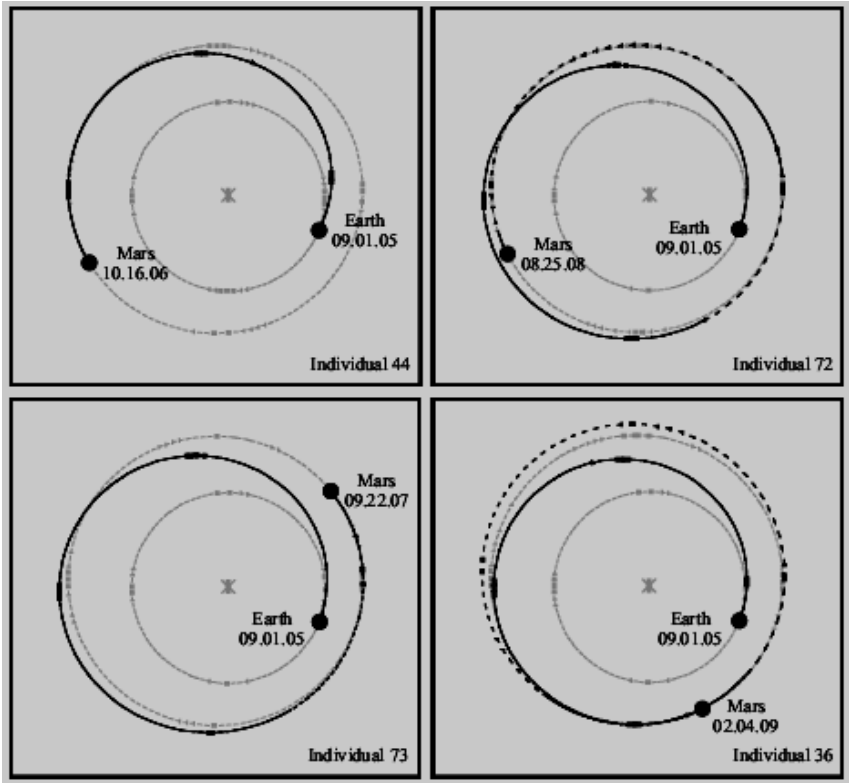


Inter-planetary Trajectory Optimization

Jet Propulsion Laboratory
California Institute of Technology

- Find a set of trade-off solutions, then choose one
- More confident decision-making (Coverstone-Carroll et al., 2000)





BEACON
An NSF Center for the Study of Evolution in Action

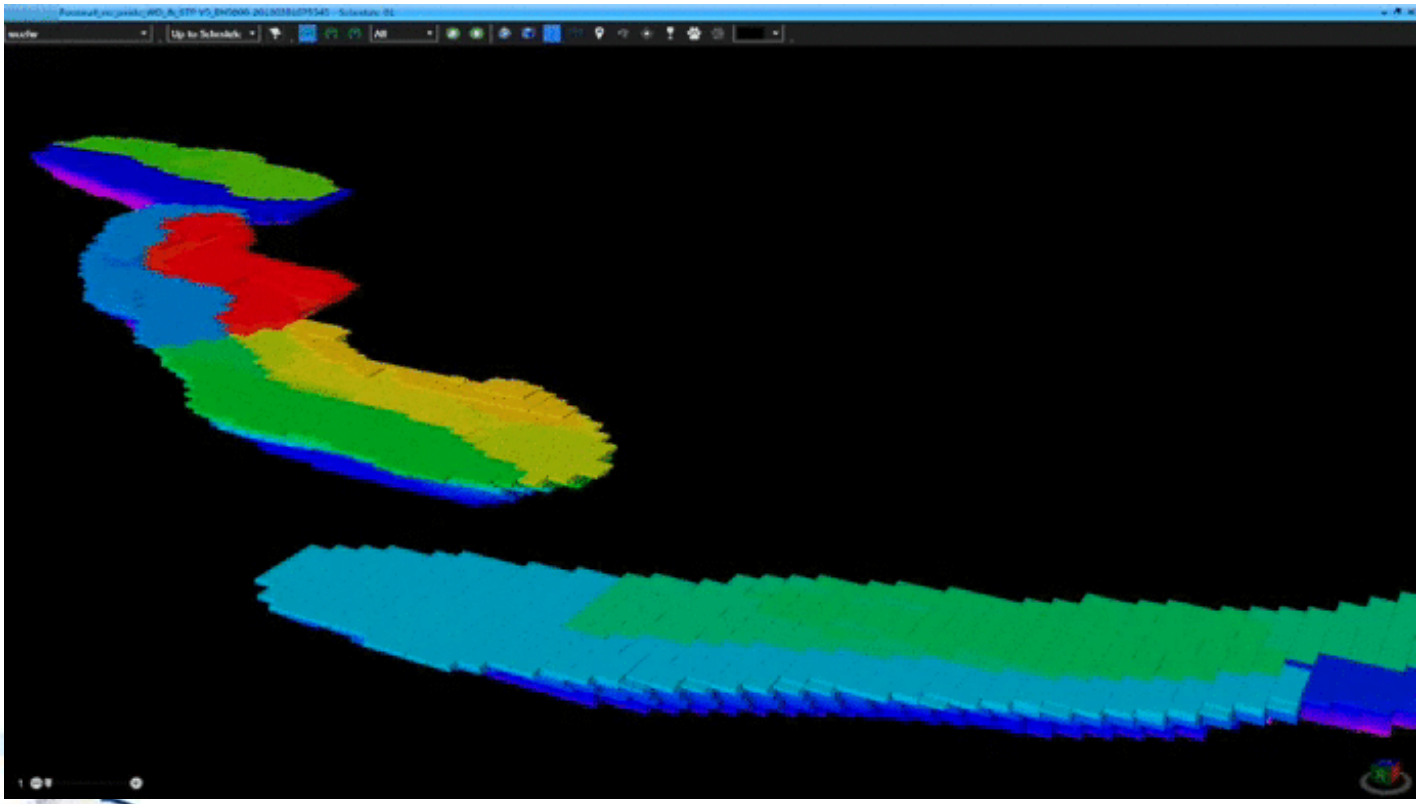
Mine Scheduler in Australia

EVORELUTION AGILE SCHEDULING

MAPTEK

- 3D scheduling for three objectives
- Each simulation takes 3-4 hours

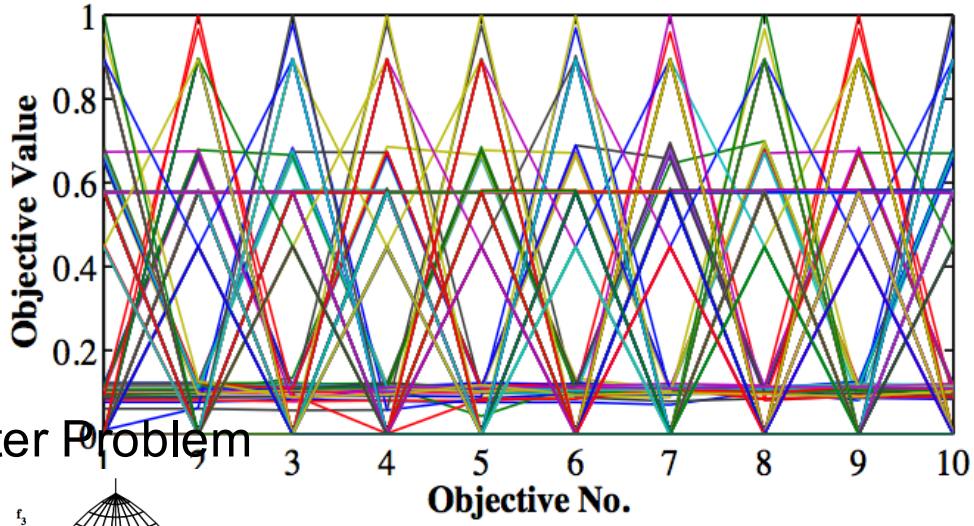
(Deb et al., 2003, IEEE)



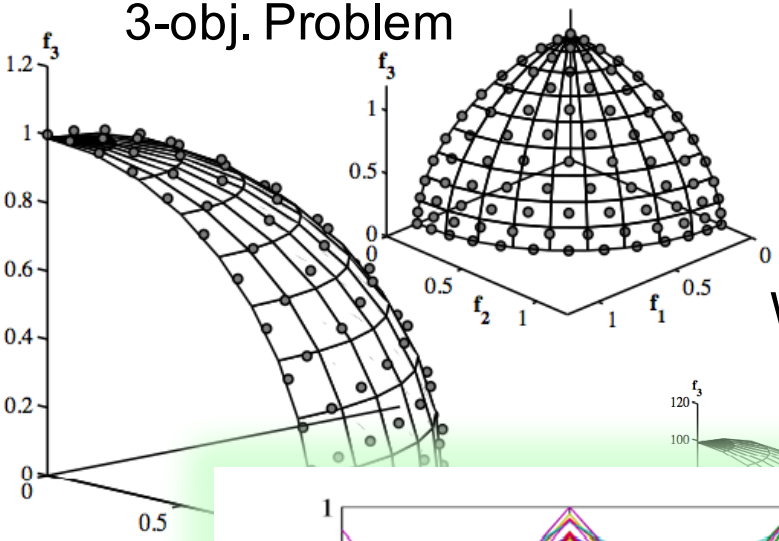
Millions of blocks sequenced

Many-Objective EMO: NSGA-III (IEEE TEC August 2014)

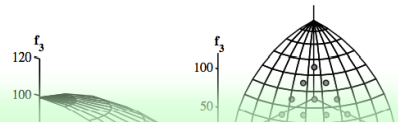
10-obj. Problem

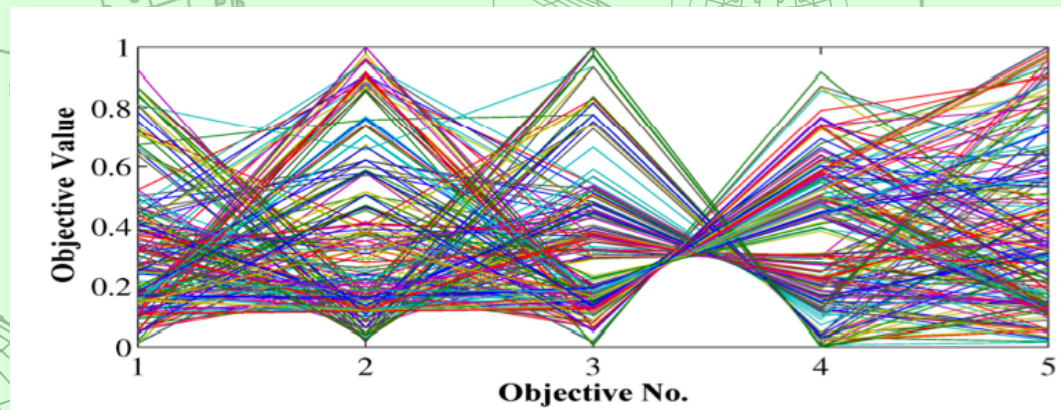


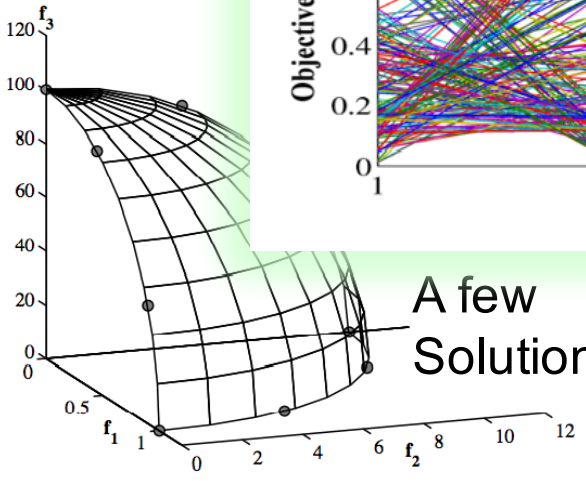
3-obj. Problem



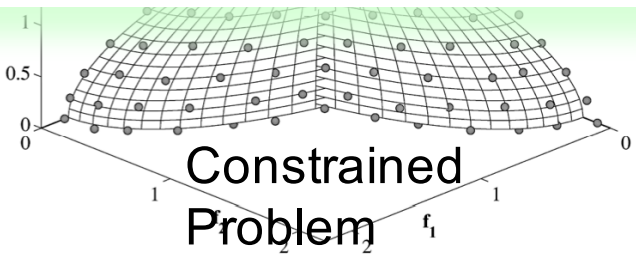
Water Problem



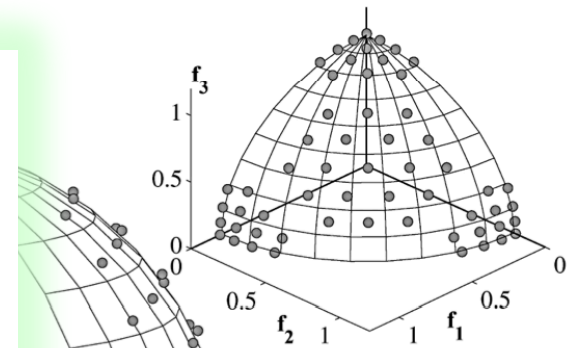




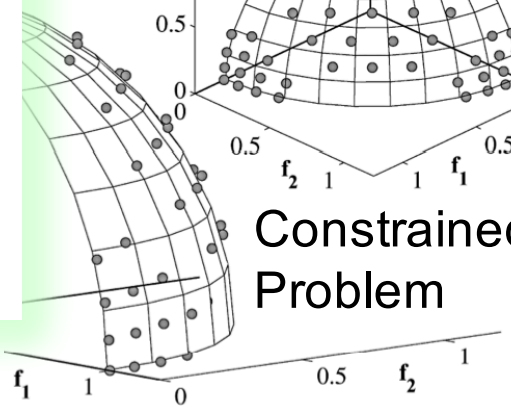
A few Solutions



Constrained Problem

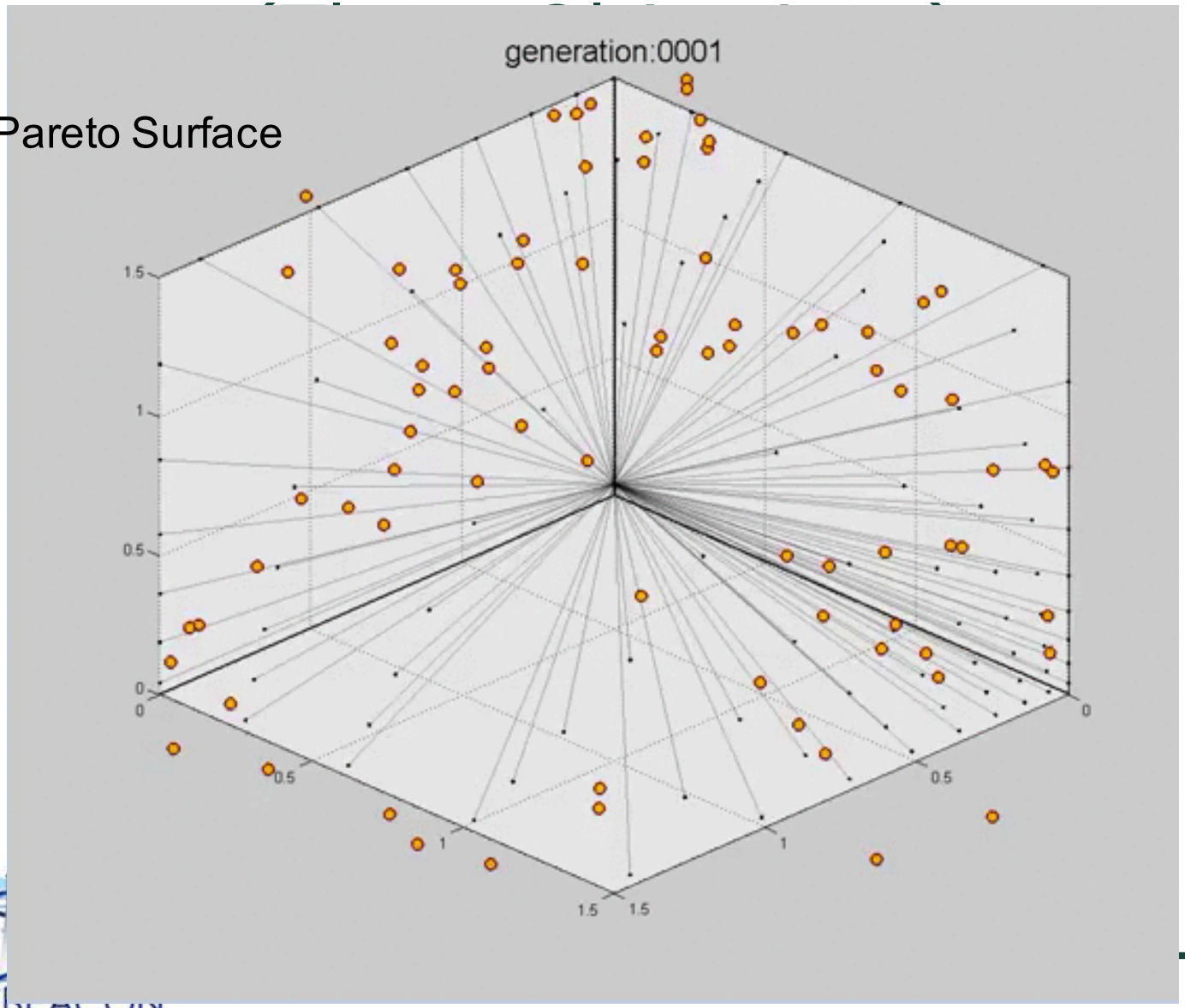


Constrained Problem



NSGA-III on DTLZ2

A 3-D Pareto Surface

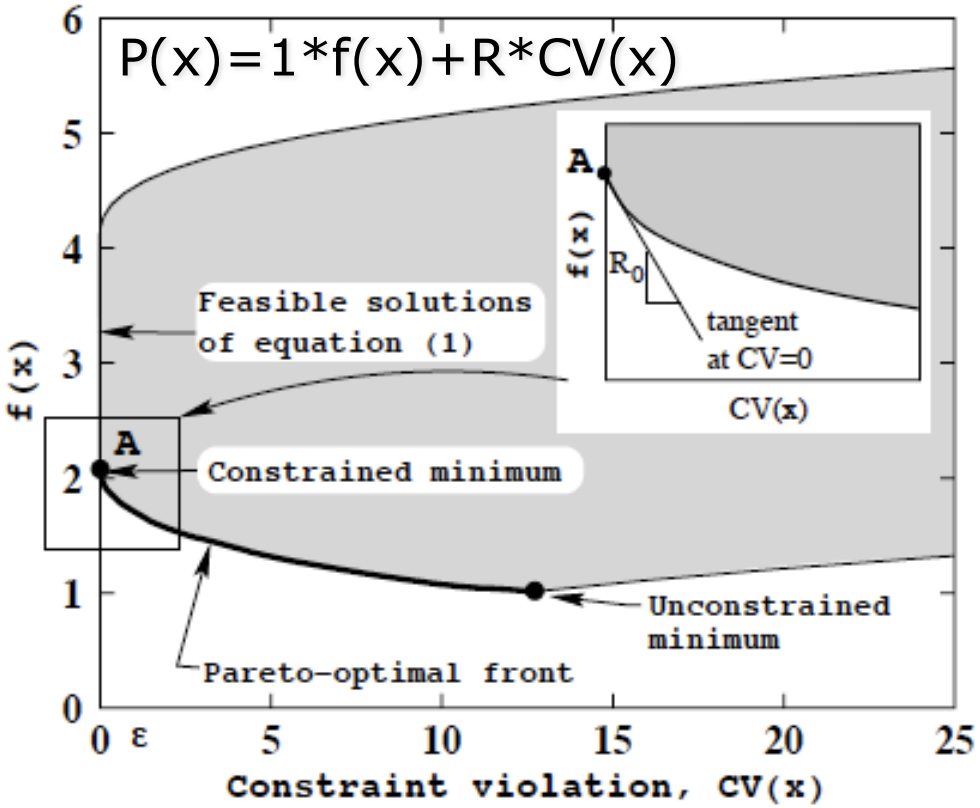


BEACON
An NSF Center for the Study of Evolution in Action

Hybrid Point-Population Based Approaches

(Deb and Datta, CEC-10)

- Best approach, if done well
- Pop-based approach to get R
- Classical penalized approach to find a local solution
- Improvements of one or two-orders of magnitude



Prob.	Zavala, Aguirre & Diharce [52]			Takahama & Sakai [45]			Brest [6]			Proposed Hybrid Approach		
	Best	Median	Worst	Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
g01	80,776	90,343	96,669	18,594	19,502	19,917	51,685	55,211	57,151	**2,630**	**3,722**	**4,857**
g02	87,419	93,359	99,654	1,08,303	114347	1,29,255	1,75,090	2,26,789	2,53,197	**26,156**	**50,048**	**63,536**
g04	93,147	1,03,308	1,109,15	12,771	13,719	14,466	56,730	62,506	67,383	**1,210**	**1,449**	**2,295**
g06	95,944	1,09,795	1,30,293	5,037	5,733	**6,243**	31,410	34,586	37,033	**1,514**	**4,149**	11,735
g07	1,14,709	1,38,767	2,08,751	60,873	67,946	75,569	1,84,927	1,97,901	2,21,866	**15,645**	**30,409**	**64,732**
g08	2,270	4,282	5,433	**621**	**881**	**1,173**	1,905	4,044	4,777	822	1,226	2,008
g09	94,593	1,03,857	1,19,718	19,234	21,080	21,987	79,296	89,372	98,062	**2,732**	**4,850**	**5,864**
g10	1,09,243	1,35,735	1,93,426	87,848	92,807	**1,07,794**	2,03,851	2,20,676	2,64,575	**7,905**	**49,102**	1,80,446
g12	482	6,158	9,928	2,901	4,269	5,620	**364**	6,899	10,424	496	**504**	**504**
g18	97,157	1,07,690	1,24,217	46,856	57,910	60,108	1,39,131	1,69,638	1,91,345	**4,493**	**7,267**	**10,219**
g24	11,081	18,278	6,33,378	1,959	2,451	**2,739**	9,359	12,844	14,827	**1,092**	**1,716**	2,890

Extreme-Scale Optimization

PTC INDUSTRIES
Aspire. Innovate. Achieve.

- Allocate batches of molten metal for making castings
- Heat: One melt of W kg
- N casting, j-th one requiring r_j copies and having w_j kg
- Given: W, N, r_j and w_j
- Metal utilization: Avg. Ratio of used metal to W
  - Max. possible: 100% util.
- Find a feasible casting schedule to achieve a given target in metal utilization (η)

- A typical assignment problem
- Often found in practice
- Large-scale problem: ~50,000 variables

(Deb and Myburgh, 2016)

Opt. Problem Formulation

- H computed from total required metal
- Variables: n = N x H
- Constraints: N+H

$$\text{Maximize} \quad f(\mathbf{x}) = \frac{1}{H}\sum_{i=1}^{H}\frac{1}{W_i}\sum_{j=1}^{N} w_j x_{ij},$$

$$\text{Subject to} \quad \sum_{j=1}^{N} w_j x_{ij} \le W_i, \quad \text{for } i = 1, 2, \ldots, H,$$

$$\sum_{i=1}^{H} x_{ij} = r_j, \quad \text{for } j = 1, 2, \ldots, N,$$

$$x_{ij} \ge 0 \text{ and is an integer.} \qquad \text{(ILP)}$$

An Example:

LINGO MILP Solver

Heat No.	j = Order Number 1	2	3	4	5	6	7	8	9	N 10	Utilization/ Cruc. Size	Efficiency (%)
i = 1	0	1	1	0	0	0	2	1	0	0	623/650	95.85
2	2	0	0	0	1	0	0	0	2	0	615/650	94.62
3	1	0	0	1	3	1	0	0	0	0	611/650	94.00
4	2	0	0	0	1	0	0	1	0	0	645/650	99.23
5	0	0	0	1	0	2	0	0	1	6	612/650	94.15
6	1	1	0	0	2	1	0	0	0	0	591/650	90.92
7	0	0	2	2	1	0	0	0	2	0	585/650	90.00
8	0	3	0	0	0	1	0	0	1	0	611/650	94.00
9	0	2	3	0	1	0	0	0	0	0	650/650	100.00
H 10	1	0	0	5	0	0	0	0	1	0	635/650	97.69
Demand=	7	7	6	9	9	5	2	2	7	6	Average	f(x)= 95.05

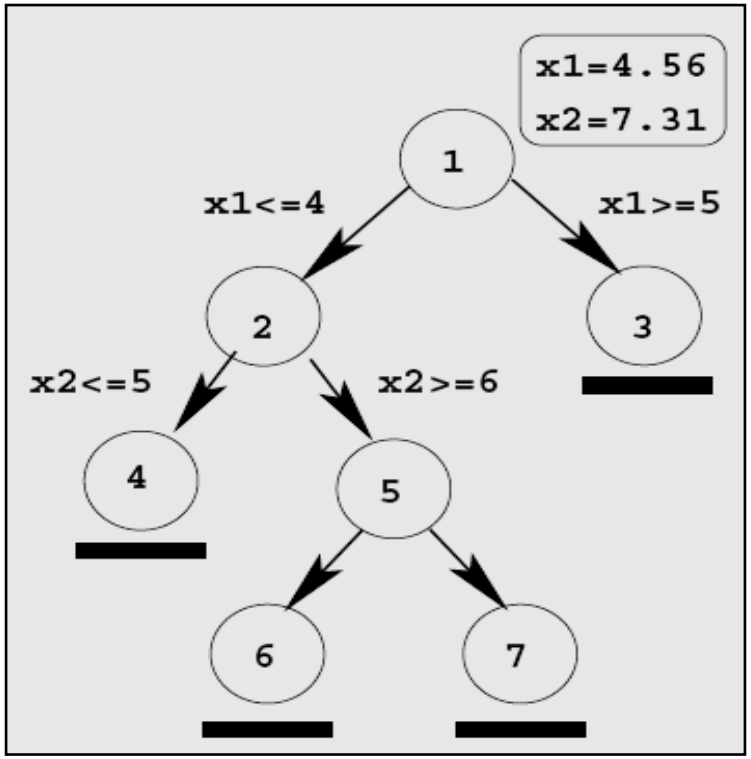
Heats

Classical Approaches

- An integer linear program (ILP)
- Multi-dim. knapsack problem (MKP)
- NP-Hard problem
  - Polynomial time algorithm to optimum not possible

Difficult to Beat

- Without integer restrictions
  - Dantzig's Simplex or Karmarkar's Predictor-Corrector method
- Branch-and-bound or branch-and-cut method
  - A fix-up
  - Exponential method



Industry-standard Softwares:
IBM's CPLEX, Ocatve' glpk, Gurobi optimizer

A Small-Sized Problem Using CPLEX and glpk

No.	1	2	3	4	5	6	7	8	9	10	Total
Wt. (kg)	175	145	65	55	95	75	195	20	125	50	
# Copies	20	20	20	20	20	20	20	20	20	20	200
Total (kg)	3500	2900	1300	1100	1900	1500	3900	400	2500	1000	20,000

kg.

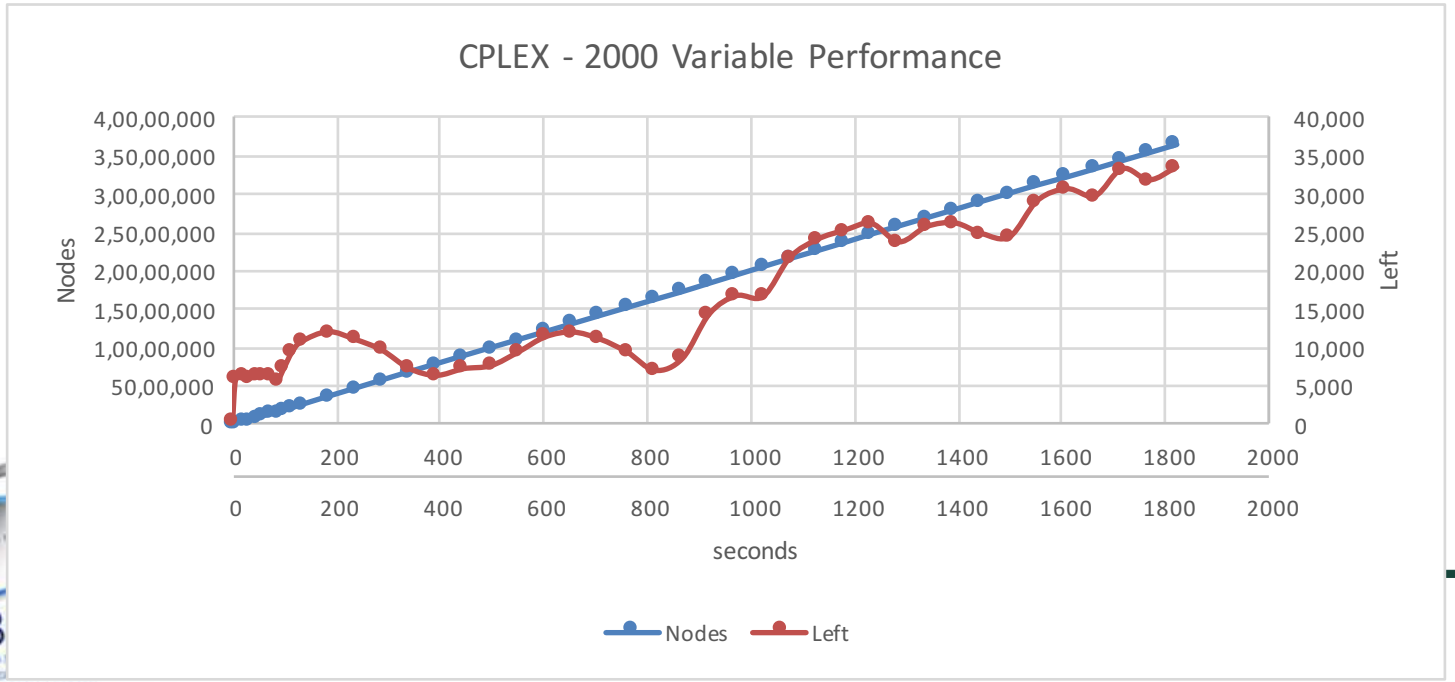
- With W=650 kg and Target η=99.7%, H=31 heats
- Total variables n=10 x 31 = 310

Method	Time (s)		# Evals.		Metal	Time (s)		# Evals.		Metal
	Avg.	SD	Avg.	SD	Util	Avg.	SD	Avg.	SD	Util
	320-Variable					310-Variable				
glpk	1.24	0.22	44,675	0.00	96.154%	–	–	–	–	–
CPLEX	0.03	0.00	184	0.00	96.154%	0.05	0.00	249	0.00	99.256%
	1000-Variable					2000-Variable				
CPLEX	0.13	0.00	947	0.00	99.462%	–	–	–	–	–

- # Copies=65 (63) require 64,650 kg metal, H=100 heats, Total variables n=10 x 100 = 1,000

A ILP from Practice Using IBM's CPLEX

- 1,800-var: 3342 nodes, 1.22 s
- 2,000-var: 37M nodes, DNC in 30 min, 32,704 nodes remaining
- Does not scale up with variables, as branch-and-cut is exponential



Proposed Population-Based ILP Method

- Step 1: Custom Initialization of a population to handle all equality constraints
  - Custom Mutation2 to attempt to fix inequality constraints
- Step 2: Assign Fitness based on objective and constraints
- Step 3: Tournament Selection to choose two parent solutions
- Step 4: Custom Recombination to combine heat-wise partial solutions into a child solution
- Step 5: Custom Mutation1 to fix equality constraints
- Step 6: Custom Mutation2 to fix inequality constraints
- Step 7: If $F(x^{best}) \geq \eta$, Go to Step 2, else print x^{best}

Target metal util.

Custom Recombination Operator

Parent 1:

Weight	154	136	57	55	67	83	187	20	123	50	Metal Used	
Parent 1	0	1	0	0	0	0	1	1	0	0	343	
	2	0	0	0	1	0	1	0	2	0	808	
	1	1	0	1	3	1	0	0	0	0	629	
	0	0	2	1	0	2	0	2	1	4	698	
Demand	3	2	2	2	4	3	2	3	3	4	0.953	Infeasible
									Fitness:		-63.6	

Parent 2:

Weight	154	136	57	55	67	83	187	20	123	50	Metal Used	
Parent 2	1	0	1	2	2	0	0	1	0	3	625	
	1	1	0	0	1	1	1	2	0	0	667	
	1	0	0	0	0	1	0	0	3	0	606	
	0	1	1	0	1	1	1	0	0	1	580	
Demand	3	2	2	2	4	3	2	3	3	4	0.953	Infeasible
									Fitness:		0.269	

Offspring:

Weight	154	136	57	55	67	83	187	20	123	50	Metal Used	
Offspring	1	0	1	2	2	0	0	1	0	3	625	
	1	1	0	0	1	1	1	2	0	0	667	
	1	1	0	1	3	1	0	0	0	0	629	
	0	1	1	0	1	1	1	0	0	1	580	
Demand	3	3	2	3	7	3	2	3	0	4	0.962	Infeasible
											0.278	

Ideal recomb. Operator; Complexity: O(*NH*)

Can easily extend to multi-parent recombination

Custom Mutation1 Operator

	Offspring:										Metal
Weight	154	136	57	55	67	83	187	20	123	50	Used
	1	0	1	2	2	0	0	1	0	3	625
	1	1	0	0	1	1	1	2	0	0	667
	1	1	0	1	3	1	0	0	0	0	629
	0	1	1	0	1	1	1	0	0	1	580
Demand	3	3	2	3	7	3	2	3	0	4	

	1	0	1	2	1	0	0	1	0	3	558
	1	0	0	0	1	1	1	2	1	0	654
	1	1	0	0	2	1	0	0	1	0	630
	0	1	1	0	0	1	1	0	1	1	636
Demand	3	2	2	2	4	3	2	3	3	4	
		3→2		3→2	7→4				0→3		

- All equality constraints are guaranteed to be satisfied
  - Complexity: O(N^2H)

Custom Mutation2 Operator

Mutation 1:

Weight	154	136	57	55	67	83	187	20	123	50	Metal Used
	1	0	1	2	1	0	0	1	0	3	558
	1	0	0	0	1	1	1	2	1	0	654
	1	1	0	0	2	1	0	0	1	0	630
	0	1	1	0	0	1	1	0	1	1	636
Demand	3	2	2	2	4	3	2	3	3	4	

Mutation 2:

Weight	154	136	57	55	67	83	187	20	123	50	Metal Used	
	1	0	1	2	1	0	0	2	0	3	578	1→2
	1	0	0	0	1	1	1	1	1	0	634	2→1
	1	1	0	0	2	1	0	0	1	0	630	
	0	1	1	0	0	1	1	0	1	1	636	
Demand	3	2	2	2	4	3	2	3	3	4		

- All inequality constraints are not guaranteed to be satisfied; F(x) takes care of constraint violation
  - Complexity: O(*NH*)

Small-Sized Problem Revisited

- Recall:
  - CPLEX and glpk could not solve more than 1,000 variables

Method	Time (s)		# Evals.		Metal	Time (s)		# Evals.		Metal
	Avg.	SD	Avg.	SD	Util	Avg.	SD	Avg.	SD	Util
	320-Variable					310-Variable				
glpk	1.24	0.22	44,675	0.00	96.154%	–	–	–	–	–
CPLEX	0.03	0.00	184	0.00	96.154%	0.05	0.00	249	0.00	99.256%
PILP	0.02	0.00	26	5.06	96.154%	0.04	0.01	150	48.08	99.256%
	1000-Variable					2000-Variable				
CPLEX	0.13	0.00	947	0.00	99.462%	–	–	–	–	–
PILP	0.05	0.004	220	37.71	99.462%	0.19	0.010	224	18.38	99.596%

- PILP requires 220 (avg.) FEs (popsize = 20 and 11 gens.)
- PILP solves 2,000 variable problem in 0.19 seconds, which CPLEX could not solve!

Properties of PILP:

1M variables

1. Random solutions are not feasible

- 1M variables, 56,352,140 kg casting
- ***W***=650 (x10) and 500 (x13) kg on alternating days
- 10,000 random solutions:
  - *F(x)*: -7.029M, -7.22M, -7.38M

Target *F(x)*=0.997

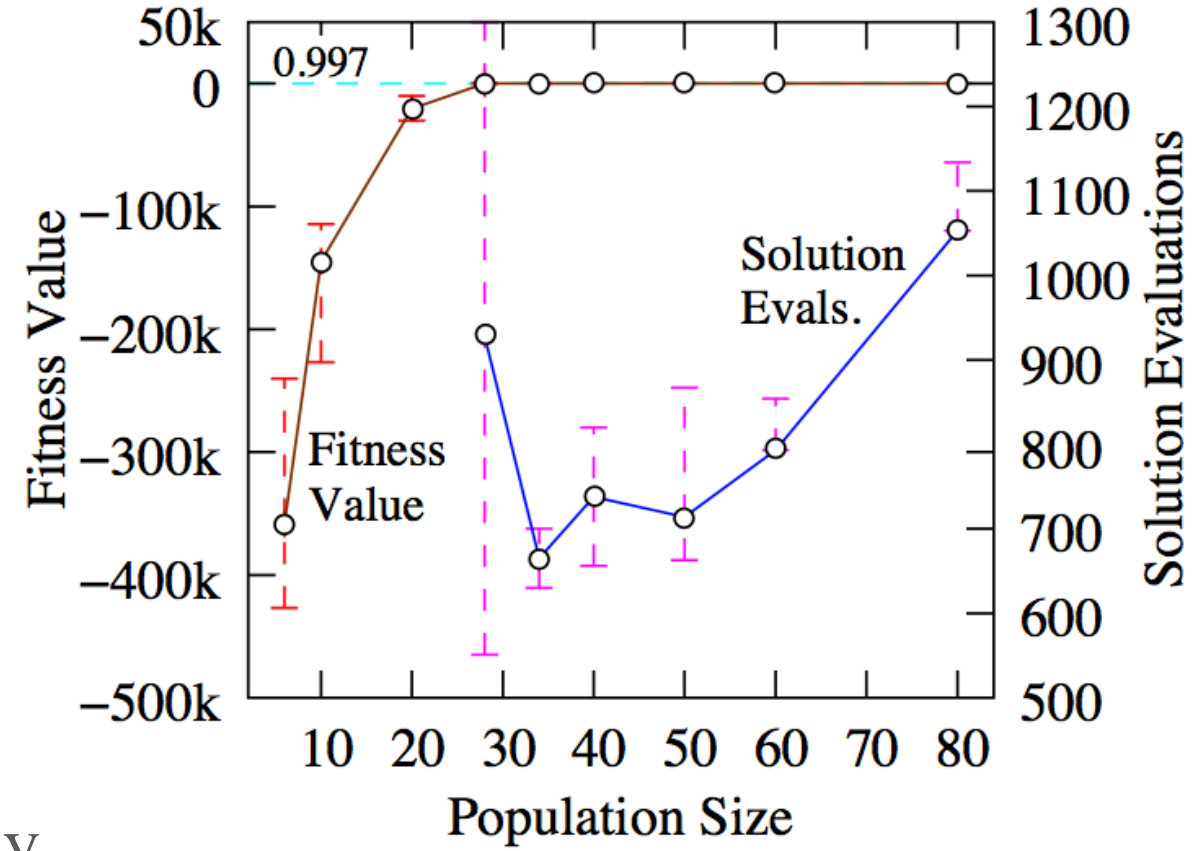
- Mutation1 and 2 bring down F(x):
  - *F(x)*: -1.39M, -1.41M, -1.45M
- A tiny part of the search space is feasible or near-feasible
  - Random or repairs thereof are not enough

Properties of PILP:

1M variables

2. A Critical Population Size is essential

- More is redundant
- Less is inadequate
- Typical performance of a population-based approach
- Recombination op. needs a sample of points to work properly



Clearly demonstrates the need of a population-based algorithm

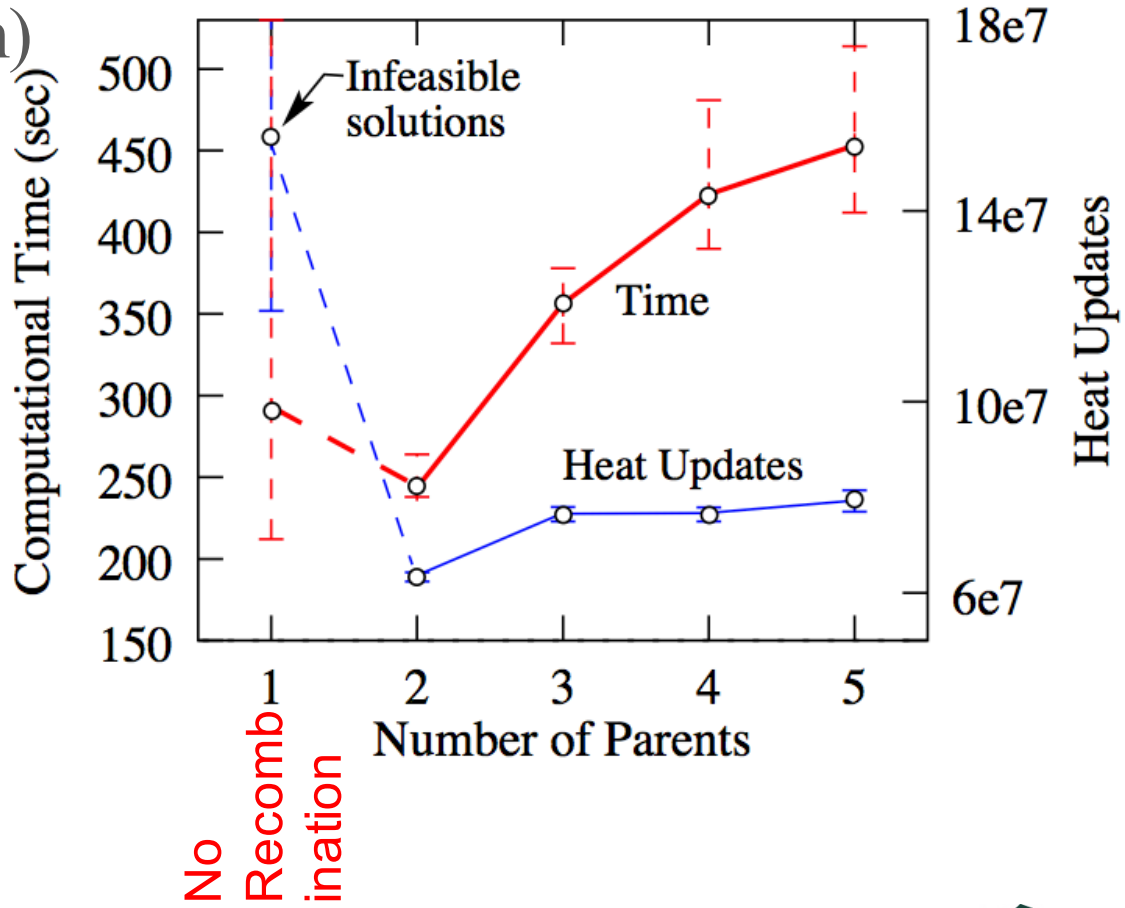
Properties of PILP:

1M variables

3. Recombination is Essential

- Multi-parent implementation
- NP=1 (no recombination)
  - No feasible solution found
- NP=2 performs best
- NP>2 found greedy for 60 popsize
  - Target always found

Heat Updates: # of Variable Changes

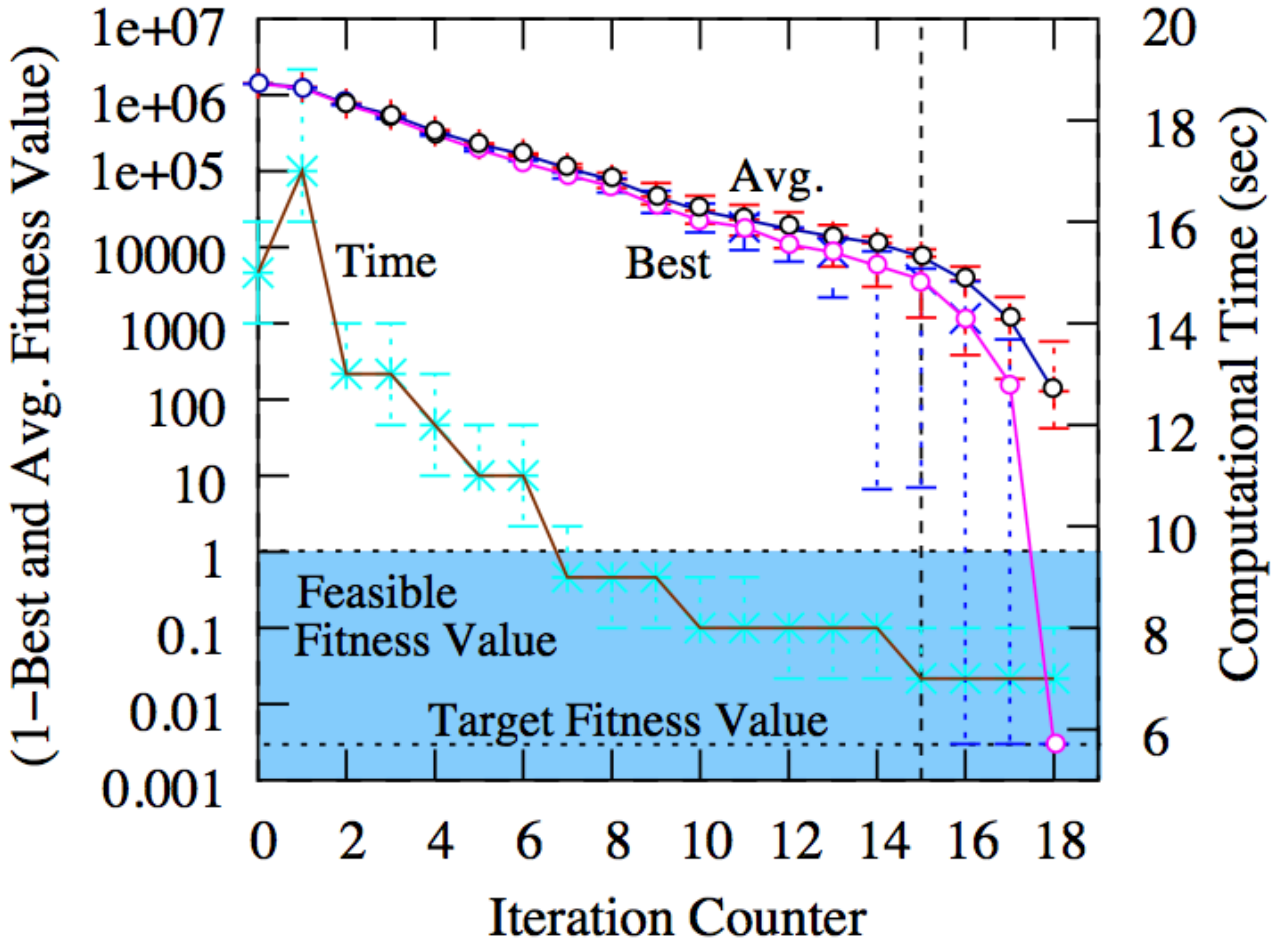


Properties of PILP:

1M variables

4. Exponential Progress towards Feasible Region

- A typical run
- Popsize = 40
- 18 iterations of PILP to find 99.7% util.
- Exponential reduction in fitness

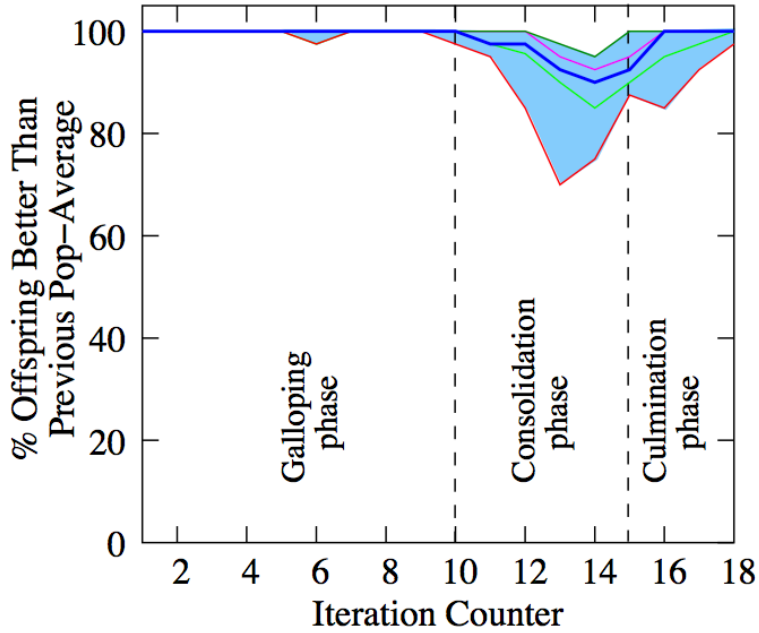


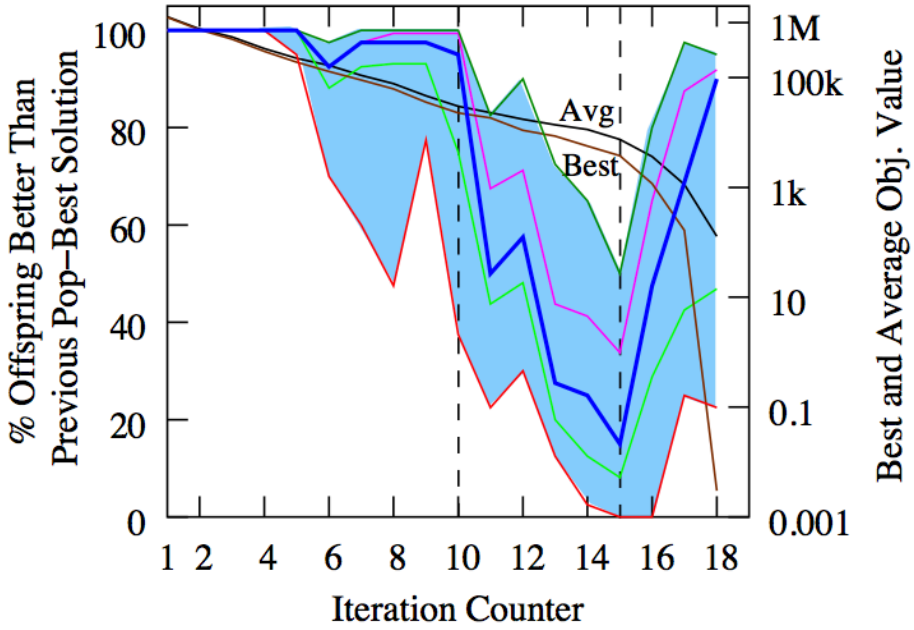
Properties of PILP:

5. Exploring Dynamics of PILP

- Continuous improvement with generations
- >70% offspring better than previous pop. average
- Three phases:
  1. Galloping phase
  2. Consolidation phase
  3. Culmination phase
- Phase 2 most difficult pd.
- After phase 2, near-optimal solutions

1M variables



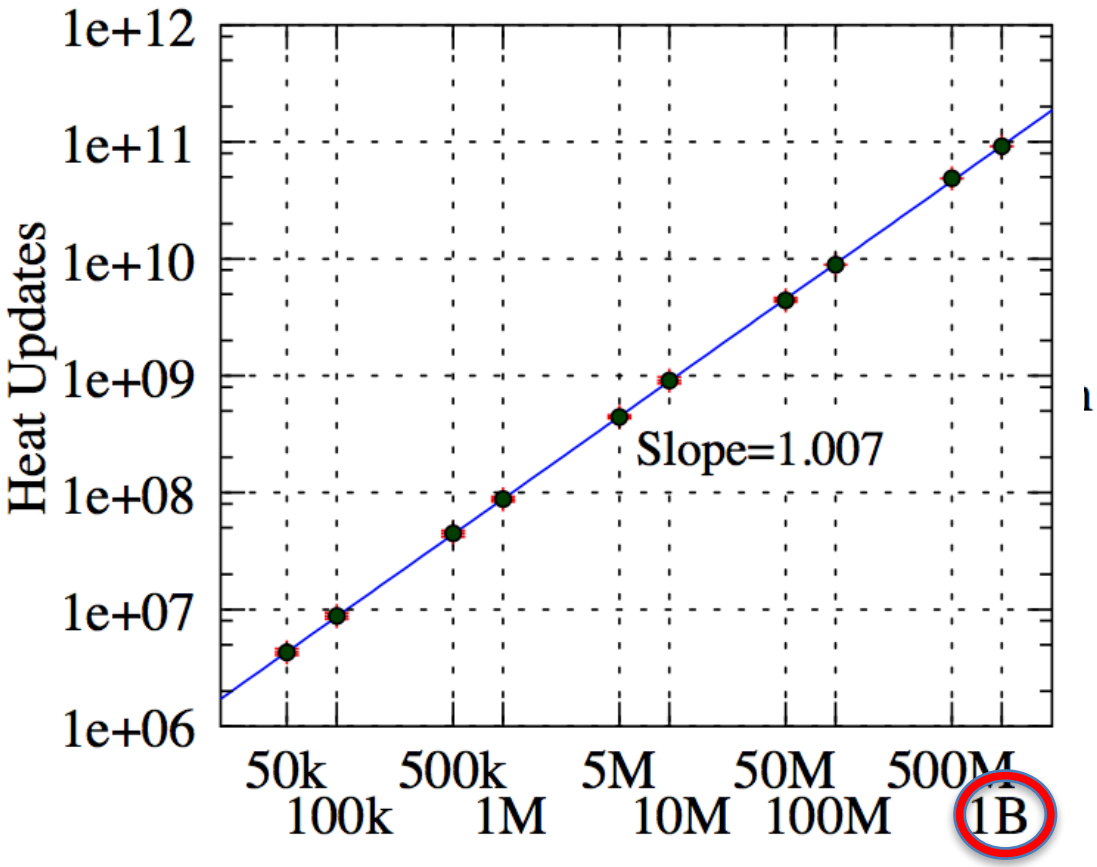


Further Results of PILP:

Computer: 2 × Intel 8-Core Xeon-2640V3 2.66 GHz, 16 threads with 16 × 16GB DDR4

6. PILP is Scalable

- 50k to 1B variables (Time: $5.34(10^{-5})n^{1.11}$)
- Multi-knapsack problem is NP-hard
- Poly. Time for approximate solutions:
  - 99.7% utilization
- Popsize: 60 for all problems

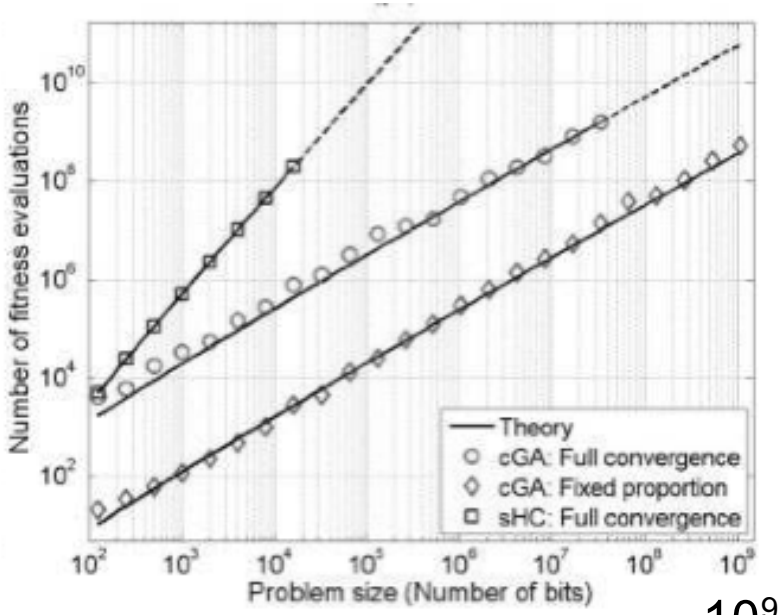


How much is a Billion?

- 4GBytes for a solution, 240GB RAM for a population

Other Billion-variable Studies using EC

- Goldberg, Sastry and Llora (2006) (Compact GA):
  - Noisy, one-max, Boolean-variable problem
  - Full convergence: 34M variable
  - 50.1% convergence: Up to 1B variable



10^9

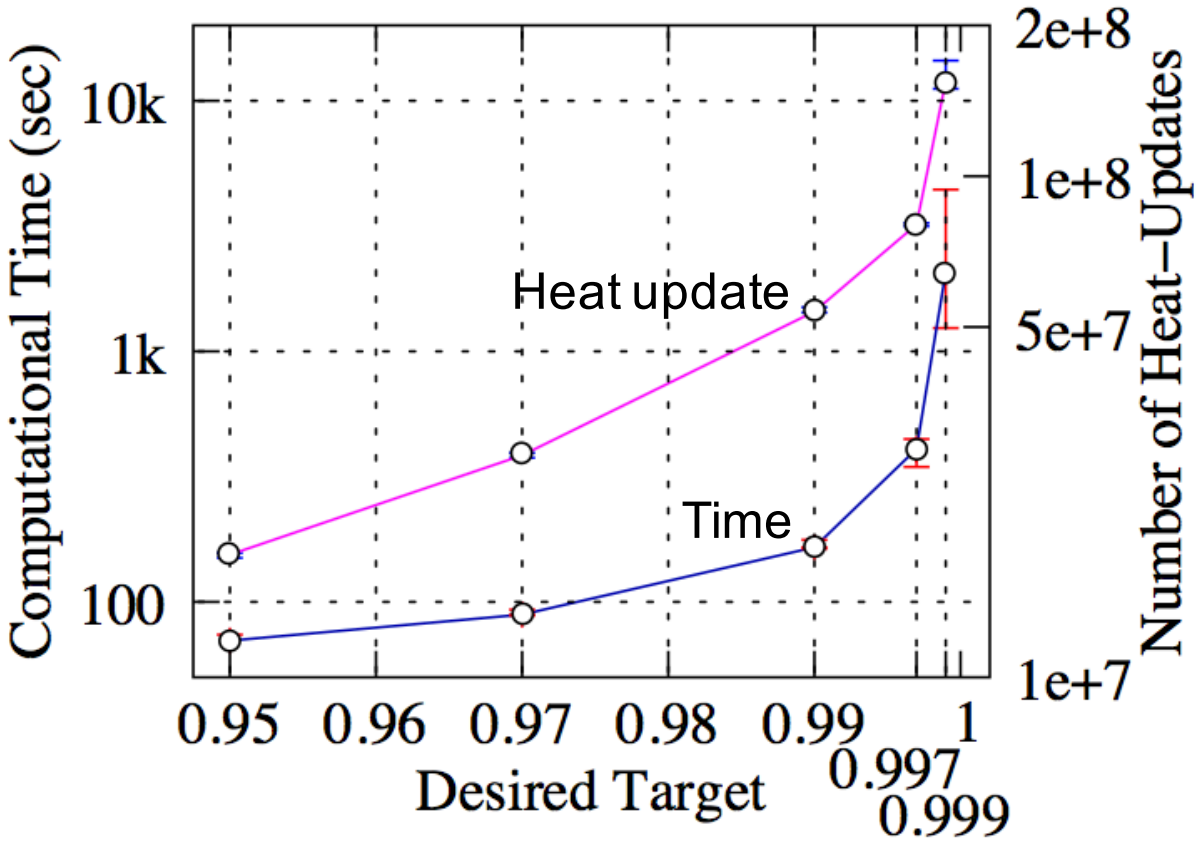
- Suwannik and Chongstitvatana, (2008), Iturriaga and Nesmachnow (2012) 64% conv. with GPU
- Wang et al. (2013) (CMA-ES)
  - 2 real-var. embedded in n=1B variable (n-2 variables do not contribute to the objective function!)

More Results of PILP:

1M Variables

7. PILP is Exponential in Accuracy

- For more accuracy, exponentially more time and heat-updates
- Agrees with NP-hardness of MKP problems



Why Does PILP Work?

1M variables

- Random solutions are not feasible
- Certain intelligent individual fix-ups do not make them feasible as well
- Mutations alone (intelligent repairs) cannot find feasible solutions
- A critical population size is essential to introduce adequate diversity
- Recombination of multiple partial solutions is a key operator
  - Recombination+Mutations+Adequate popsize make it happen
- PILP exponentially moves towards feasible region
  - PILP finds high proportion of improved solutions every gen.

BEACON
An NSF Center for the Study of Evolution in Action

Generic Assignment Problem

- PILP can solve problems of following type:

$$\begin{aligned}
&\textbf{Maximize} && f(\mathbf{x}) = \sum_{i=1}^{H} \sum_{j=1}^{N} a_{ij} x_{ij}, \\
&\textbf{Subject to} && \sum_{j=1}^{N} b_{ij} x_{ij} \leq c_i, \quad \text{for } i = 1, 2, \ldots, H, \\
& && \sum_{i=1}^{H} d_{ij} x_{ij} = e_j, \quad \text{for } j = 1, 2, \ldots, N, \\
& && x_{ij} \geq 0 \text{ and is an integer.} \quad \text{for all } (i, j).
\end{aligned}$$

- Multiply-constrained knap-sack problem and its relaxations (Kellerer et al., 2004)
- Cutting stock problem
- Multiple subset problem (Martello and Toth, 1990)
- Linear assignment problems (Akgul, 1992)
- Currently investigating binary knap-sack problems

Conclusions

- Point-based methods has their niche
- Same is true for population-based methods
- No hope for a single method to be efficient for all problems
- Customized optimization is the key
  - EC based population methods are flexible
- Large-scale optimization through customization
- Solved a billion-variable real-world problem to near-optimality for the first time
  - A triumph for Evolutionary Algorithms

Further Information: http://www.coin-laboratory.com