Stochastic Search

An element of randomness and statistics

Hill Climbing: Min-conflict, GSAT*

Metaheuristic Search: GENET, GLS*, GGA*

** Note: not in Tsang 1993*

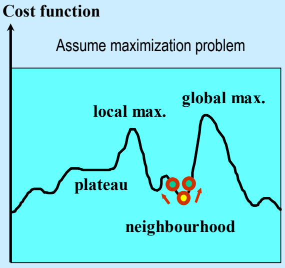
Where Stochastic Methods Sit in Constraint Satisfaction

- Problem Reduction
- Search
  - Complete search methods
    - Search Ordering
  - Stochastic methods
- Solution Synthesis

Why Stochastic Search?

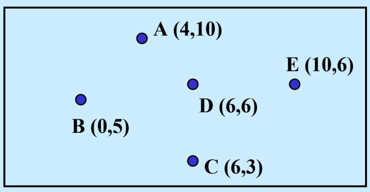
- Schedule 30 jobs to 10 machines:
  - Search space: 10^{30} leaf nodes
- Generously allow:
  - Explore one in every 10^{10} leaf nodes!
  - Examine 10^{10} nodes per second!
- Problem takes **300 years** to solve!!!
- How to contain *combinatorial explosion*?

Background: Local Search



- Ingredients:
  - Cost function
  - Neighbourhood function
  - [Optional] Strategy for visiting neighbours
    - e.g. *steepest ascent*
- Problems:
  - local optimum
  - Plateau
  - When to stop?
    - Ok with satisfiability
    - But not optimization

Example: The Travelling Salesman Problem (TSP)

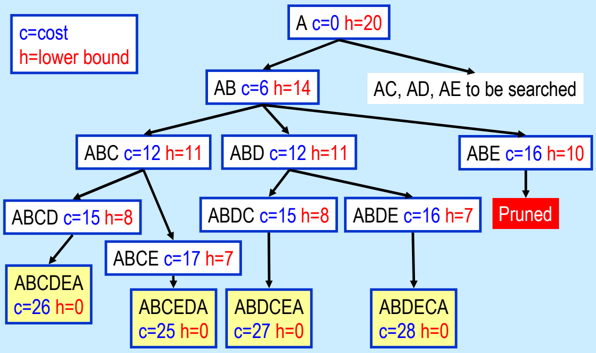


- Goal: to find *shortest route* through all cities
- Optimization involved: minimization

Distance Table for an example TSP

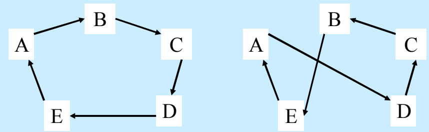
	A	B	C	D	E
A	--	6	7	4	7
B	6	--	6	6	10
C	7	6	--	3	5
D	4	6	3	--	4
E	7	10	5	4	--
Heuristic:	*4*	*6*	*3*	*3*	*4*

Branch & Bound (1)



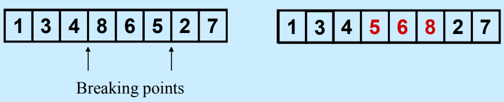
HC Example: 2-opting for TSP

- Candidate tour: a round trip route
- Neighbour: exchange two edges, change directions accordingly



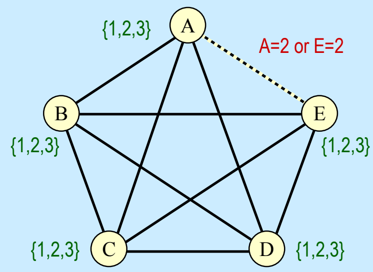
List reversing ➔ 2-Opting

- List representation:
  - A list could represent cities in sequence
- 2-Opting can be seen as sub-list reversing
  - Easy to implement

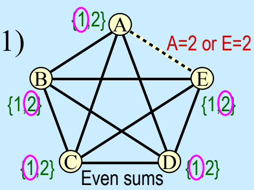


Example: Many Local Optimum

- All constraints require "*even sum*" except C_{AE}
- Only one solution
- Easy to be trapped

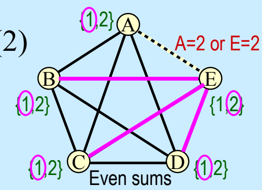


Hill Climbing in Action (1)



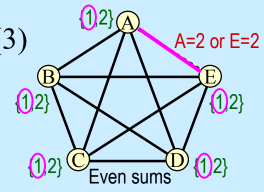
- Random start ABCDE:
  A=1,B=2,C=1,D=1,E=2
- Violations:
  - AB, BC, BD, CE, DE
- Neighbours:
  A → 2 satisfies AB, but violates AC and AD
  B → 1 satisfies AB, BC and BD, but violates BE
  C → 2 satisfies BC and CE, but violates AC and CD
  D → 2 satisfies BD and DE, but violates AD and CD
  E → 1 satisfies CE and DE, but violates AE and BE
- Move to A=1,**B=1**,C=1,D=1,E=2

Hill Climbing in Action (2)



- Current position:
  A=1,B=1,C=1,D=1,E=2
- Violations:
  - BE, CE, DE
- Neighbours:
  A → 2 satisfies none, but violates AB, AC and AD
  B → 2 satisfies BE, but violates AB, BC and BD
  C → 2 satisfies CE, but violates AC, BC and CD
  D → 2 satisfies DE, but violates AD, BD and CD
  E → 1 satisfies BE, CE and DE, but violates AE
- Move to A=1,B=1,C=1,D=1,**E=1**

Hill Climbing in Action (3)



- Current position:
  A=1,B=1,C=1,D=1,E=1
- Violations:
  - AE
- Neighbours:
  A → 2 satisfies AE, but violates AB, AC and AD
  B → 2 satisfies nothing, but violates AB, BC, BD and BE
  C → 2 satisfies nothing, but violates AC, BC, CD and CE
  D → 2 satisfies nothing, but violates AD, BD, CD and DE
  E → 2 satisfies AE, but violates BE, CE and DE
- No profitable repair possible → **local optimum** found

Min-conflict Heuristic Repair

- Start with a random complete assignment
  - May initialise with min-conflict heuristic
- Repeat until all constraints are satisfied or run out of resources:
  - Randomly pick a variable *x* that is in conflict
  - Pick value *v* in domain of *x* such that
    - <*x*, *v*> violates the least number of constraints
    - break ties randomly

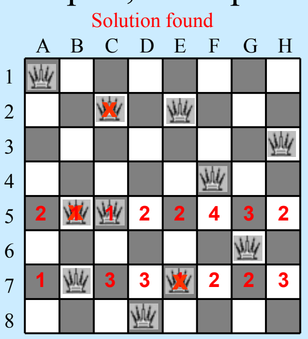
MC Heuristic Repair, Example

- Start with random assignments
- C2 attacks G6
  D8 attacks E7
- Randomly pick one, say, E7, to repair

Count number of conflicts in each square

Randomly pick a square with least attacks, say, B7

Repeat repair

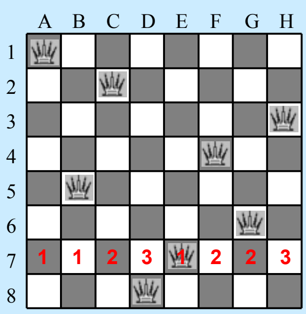


Informed Backtrack
(complete algorithm)

- VarsLeft = random complete assignment
  - May initialize with min-conflict heuristic
- VarsDone = empty set
- Do until all variables violate no constraints:
  - Remove from VarsLeft variable *x* in conflict
  - Assign min-conflict value *v* to *x*, but
    - only accept *v* if <*x*,*v*> is consistent with VarsDone
  - Add <*x*,*v*> to VarsDone
  - Backtrack when necessary

Informed Backtrack, Example

- Step 1:
  VarsLeft = {1A, 2C, 3H, 4F, 5B, 6G, 7E, 8D}
  VarsDone = {}
  * Illegal variable picked: **7E**
  * Try a value for row 7, say, **7B**
  * Backtrack if needed
  explore 7B, 7A, 7E, …
- Step 2:
  VarsLeft = {1A, 2C, 3H, 4F, 5B, 6G, 8D}
  VarsDone = {**7B**}
  * Illegal variable picked: 5B
  * Pick a value for row 5, but not any value attacking 7B



Informed Backtrack, Analysis

- *Complete search* in nature
  - Basically *ordering variables* dynamically, guided by constraint violation
  - *Ordering values* by number of conflicts involved
  - All values are explored if needed
- Benefit of *hill-climbing*
  - Changing one label at a time
  - Chance to hit a solution by chance early
- Perhaps it deserves more research

The Satisfiability Problem (SAT)

- Boolean variables (true or false)
  - i.e. all domains are {0, 1}
- Constraints: in *Conjunctive Normal Form*:
  $(X_1 \vee \neg X_2 \vee X_3) \wedge (X_2 \vee X_4 \vee \neg X_5) \wedge \ldots$
- All CSPs can be translated to CNF
  - Each label <*x*, *v*> becomes one variable *XV*
  - If *XV* = 1, then *x* takes value *v*
  - Add clauses to ensure *x* takes one value only
- Note: given CSP with **n** variables, **m** values each:
  - The CSP has m^n leave nodes to explore
  - The equivalent SAT problem will have 2^{mn} leaves

Satisfiability Problem, Example

- Suppose we say:
  A → B
  B → C
  C → ¬A
  which together refutes A
- 2-SAT problems are tractable
- Constraints may be represented by matrices
- Boolean variables
  A, B, C
- Constraints:
  ¬A ∨ B
  ¬B ∨ C
  ¬A ∨ ¬C
- Possible solutions:
  (<A,0><B,1><C,1>)
  (<A,0><B,0><C,1>)
  (<A,0><B,0><C,0>)

The GSAT Algorithm

- Parameters: *max_tries* & *max_flips*
- Do max_tries
  - Do max_flips
    - Pick an unsatisfied clause
    - Flip a variable that results in min. constraint violation
- Many many variations, including:
  - Adding random walks
  - Adding weights
  - Adding "*taboo lists*"

GSAT Example

- Boolean variables
  A, B, C
- Constraints:
  (a) ¬A ∨ B
  (b) ¬B ∨ C
  (c) ¬A ∨ ¬C
  (d) A ∨ B ∨ ¬C
- Random starting point
  (A=1, B=0, C=1)
- Solution found in step 3:
  (A=0, B=0, C=0)

		A	B	C	Violation
1.		A=1	B=0	C=1	(a), (c)
		Violated constraint picked: (c)			
neighbours		A=0	B=0	C=1	(d)
neighbours		A=1	B=0	C=0	(a)
2.		A=1	B=0	C=0	(a)
Neigh-bours		A=0	B=0	C=0	--
Neigh-bours		A=1	B=1	C=0	(b)
3.		A=0	B=0	C=0	--

GSAT in Local Optimal

- Constraints:
  (a) A ∨ ¬ B
  (b) ¬ A ∨ B
  (c) B ∨ ¬ C
  (d) ¬B ∨ C
  (e) ¬A ∨ C
  (f) A ∨ ¬C
  (g) ¬ A ∨ ¬ B ∨ ¬C
- Local optimal:
  (A=1, B=1, C=1)
  Solution missed:
  (A=0, B=0, C=0)

	A	B	C	Violation
	A=1	B=1	C=1	(g)
neighbours	A=0	B=1	C=1	(a), (f)
neighbours	A=1	B=0	C=1	(b), (c)
neighbours	A=1	B=1	C=0	(e), (e)

All moves are inferior to current position; unfortunately, the current position is not a solution

GSAT Variants

GSAT has many variants, e.g.:
- HSAT
  - Introducing history
- GSAT with added randomness
  - GSAT+w
  - WSAT
- MaxWalkSAT
  - For weighted MAXSAT problems

GSAT with Random Walk: GSAT+w

- With probability p
  - Pick an unsatisfied clause C
  - Pick a variable x in C
  - Flip x
- With probability $1 - p$
  - Make one GSAT step
  - (i.e. Flip a variable that results in min. constraint violation)

WSAT

- Variation of GSAT+w
- Randomly pick an unsatisfied clause C
- With probability p
  - Flip a variable x in C at random
- With probability $1 - p$
  - Make one GSAT step
  - (i.e. Flip a variable that results in min. constraint violation)

Meta-heuristics

Simulated Annealing

Tabu Search

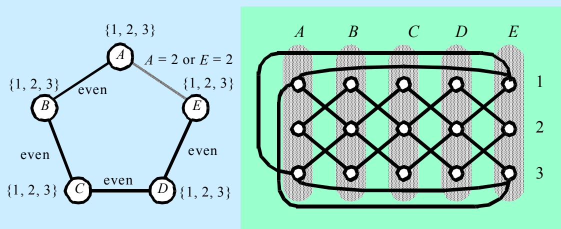
GENET, GLS

(GGA)

- Changing hill climbing behaviour
- mainly to escape local optima

Hill Climbing
e.g. 2-Opt in TSP

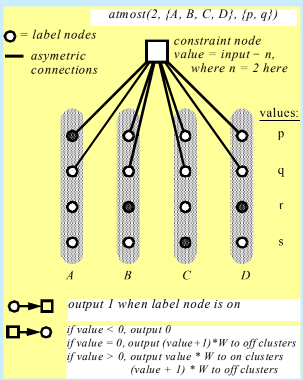
GENET: Network Representation



- Build inhibitory connections
- Let the network converge to solutions

GENET on non-binary problems

- Constraint nodes
- Label nodes
- ***Three*** states to consider (not 2!)
  - Violated
  - Under-constrained
  - At the limit



Performance of GENET

- Random binary CSPs
  - all problems up to 200 variables solved
- Random CSPs with 50 variables, up to 500 atmost constraint
- Hard graph colouring problems
- Car-sequencing problem
  - up to 200 cars, with non-binary constraints
- SAT problems

GLS: Augmented Cost Function

- Identifying solution ***features***, e.g. edges used
- Associate ***costs*** and ***penalties*** to features
- Augmented Cost Function

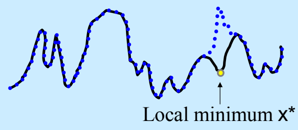
$$H(s) = G(s) + \lambda * \Sigma\, p_i * I_i(s)$$

  - where G is original cost function
  - λ is a parameter to GLS
  - p_i is penalty assoc. to feature i, initialized to 0
  - $I_i(s)$ = 1 if s exhibits feature i; 0 otherwise

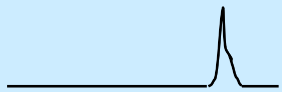
GLS & Filled Function Method

Augmented function to minimize, h' = h + f

Minimize (augmented) function h



At local minimum, add filled function f (penalty)



The GLS Algorithm

- Iterative local search
- In a local minimum
  - Select Features
    - Maximize *utility*

$$I_i(s^*) \times \frac{c_i}{1 + p_i}$$

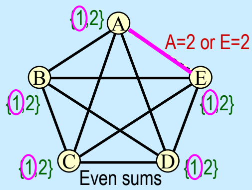
$I_i(s^*)$ = 1 if s* exhibits feature i; 0 otherwise

c_i = cost of feature i

p_i = penalty of feature I (init. to 0)

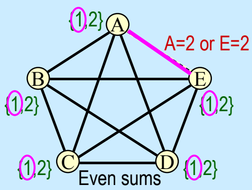
  - Increase penalties (strengthen constraints)
- Resume Local Search from Local Minimum

GLS in Action (1)



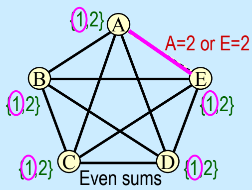
- Local optimum:
  A=1,B=1,C=1,D=1,E=1
- Violations:
  - AE
- Neighbours:
  A → 2 satisfies AE, but violates AB, AC and AD
  B → 2 satisfies nothing, but violates AB, BC, BD and BE
  C → 2 satisfies nothing, but violates AC, BC, CD and CE
  D → 2 satisfies nothing, but violates AD, BD, CD and DE
  E → 2 satisfies AE, but violates BE, CE and DE
- No profitable repair possible → **local optimum** found

GLS in Action (2)



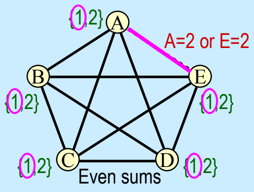
- Local optimum:
  A=1,B=1,C=1,D=1,E=1
- Violations:
  - AE
- Features: I_{XY} = 1 means constraint on XY is violated
  I_{AE} = true in this case
- Let cost for all features c_{XY} be 1
- Let λ = 1
- Penalty p_{AE} is incremented (from 0) to 1
  Since only AE is violated

GLS in Action (3)



- Current position:
  A=1,B=1,C=1,D=1,E=1
- Violations: AE
- $H(s) = G(s) + \lambda \cdot \Sigma\, p_i \cdot I_i(s)$
  $= 1 + 1 \cdot 1 = 2$
- Neighbours:
  A → 2 satisfies AE (2), but violates AB, AC and AD (3)
  B → 2 satisfies nothing, but violates AB, BC, BD and BE
  C → 2 satisfies nothing, but violates AC, BC, CD and CE
  D → 2 satisfies nothing, but violates AD, BD, CD and DE
  E → 2 satisfies AE (2), but violates BE, CE and DE (3)
- No profitable repair possible; penalise again

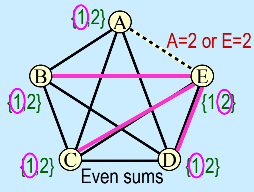
GLS in Action (4)



- Current position:
  A=1,B=1,C=1,D=1,E=1
- Violations: AE
- $H(s) = G(s) + \lambda \cdot \Sigma\, p_i \cdot I_i(s)$
  $= 1 + 1 \cdot 2 = 3$

- Neighbours:
  A → 2 satisfies AE (3), but violates AB, AC and AD (3)
  B → 2 satisfies nothing, but violates AB, BC, BD and BE
  C → 2 satisfies nothing, but violates AC, BC, CD and CE
  D → 2 satisfies nothing, but violates AD, BD, CD and DE
  E → 2 satisfies AE (3), but violates BE, CE and DE (3)
- May make A=2 or E=2, if **sideways moves** allowed

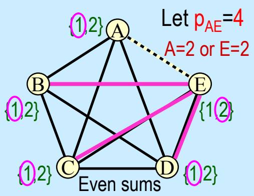
GLS in Action (5)



- Current position:
  A=1,B=1,C=1,D=1,E=2
- Violations: BE, CE, DE
- $H(s) = G(s) + \lambda \cdot \Sigma\, p_i \cdot I_i(s)$
  $= 3 + 0 \cdot 3 = 3$

- Neighbours:
  A → 2 satisfies nothing, but violates AB, AC and AD (3),
  B → 2 satisfies BE (1), but violates AB, BC and BD (3)
  C → 2 satisfies CE (1), but violates AC, BC and CD (3)
  D → 2 satisfies DE (1), but violates AD, BD and CD (3)
  E → 1 satisfies BE, CE and DE (3), but violates AE (3)
- May make E=1

GLS in Action (6)



- Current position:
  A=1,B=1,C=1,D=1,E=2
- Violations: AE
- $H(s) = G(s) + \lambda \cdot \Sigma\, p_i \cdot I_i(s)$
  $= 3 + 3 \cdot 0 = 3$

- Neighbours:
  A → 2 satisfies nothing, but violates AB, AC and AD (3),
  B → 2 satisfies BE (1), but violates AB, BC and BD (3)
  C → 2 satisfies CE (1), but violates AC, BC and CD (3)
  D → 2 satisfies DE (1), but violates AD, BD and CD (3)
  E → 1 satisfies BE, CE and DE (3), but violates AE (4)
- Local optima reached, change p_{BE}, p_{CE} or p_{DE} to 1

GLS on TSP

Features:
- n^2 Features
- cost = distance given
- e.g. tour [1,5,3,4,6,2]

- Local search: 2-opting
- ➢ $\lambda = a \times g(t^*) / n$

a = parameter to tune, within (0, 1]

n = # of cities

t* = first local minimum produced by local search; g(t*) = cost of t*

	1	2	3	4	5	6
1					X	
2	X					
3				X		
4						X
5			X			
6		X				

Fast Local Search (FLS)

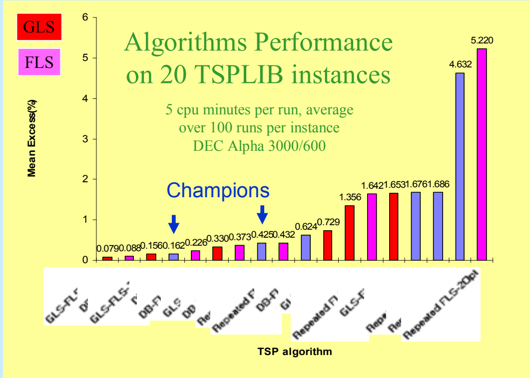
- For speeding up local search
  – through reduced neighbourhood
- Method:
  – **associate activation bit to problem features**
  – **Only active features examined for hill climbing**
- Cost for speed-up: lost of solution quality
- Rescue: solution quality compensated by GLS

Results, GLS-FLS on RLFAP

- No algorithm reported on all 25 RLFAPs
  – GLS against the best obtained by all algorithms
- GLS on 25 benchmark RLFAPs
  – ***Better than best published solutions in 5***
  – ***Equal best solutions in 18***
  – ***Close to best solutions in 2***
- All GLS-FLS variants found good results
  – Best results in GLS-FLS-S4 with Circular List
  – With mean cpu time: 4.73 minutes

TSP: GLS vs Specialised Methods

- Repeated Lin-Kernighan (LK) (1973)
- Iterated Lin-Kernighan (Johnson 1990)
  - also known as double-bridge moves
  - considered champion (Zachariasen & Dam 96)
- TSPs tested: 48-1002 cities
  - 5 & 30 cpu minutes on DEC Alpha 3000/600
  - GLS+FLS+2-Opting found better results with lower variance
- GLS found its place in TSP research



GENET/GLS Applications (1)

- Constraint Satisfaction (GENET), incl.
  - Graph colouring, Car-sequencing
- Travelling Salesman
- Radio Length Frequency Assignment
- BT's work force scheduling
- Other Applications
  - Quadratic assignment,
  - Function optimization (F6)
  - Network routing
  - Max. channel assignment
- SAT / MAXSAT

GENET/GLS Applications (2)

- Vehicle Routing
  - Strathclyde University and ILOG
  - Commercial Product: ***ILOG Dispatcher***
  - BT sponsored Case Studentship in Essex
- Logic Programming (Melbourne, Singapore, Chinese Univ. of Hong Kong)
- Train scheduling (King's College, London)
- Bus scheduling (Leeds)
- Bin Packing (University of Copenhagen)

Remarks: parameters in GLS

- Local search strategy
  - Needed in HC, SA, Tabu Search
- Features, costs
  - Sometimes come naturally from problem spec.
- Main parameter λ
  - Experimental results sometimes sensitive to λ
- *Less tuning to do than GA, NN, Tabu (& SA)*

Components in GLS

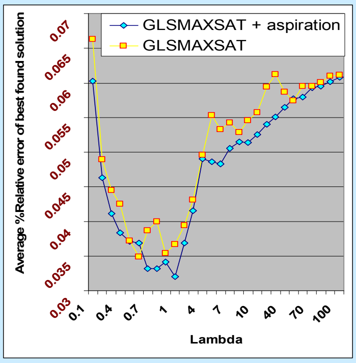
- Local search strategy
  - Also needed in HC, SA, Tabu Search
- Features, costs
  - Sometimes come naturally from cost function
- Main parameter: λ
  - Experimental results sometimes sensitive to λ
  - Our practice: $\lambda = \mathbf{a} \times g$(first local optimum)
- *Question: how to tune* **a** *(λ -coefficient)?*

Patrick Mills: GLS+

- **Aspiration**: if **G**(*s*) is better than best so far, then move to *s* even if **H**(*s*) is inferior
  - Work for MaxSAT and QAP but not SAT
  - Result generally improved at high λ value
- **Random moves**:
  - With probability *Pr* make random move
  - Results improved in QAP at low λ value
  - No effect on GLS → SAT / MAX SAT
- Combining Aspiration and Randomness:
  - GLS performance less sensitive to λ in QAP, [Max]Sat
- Where else / when will it work?

GLS + Aspiration

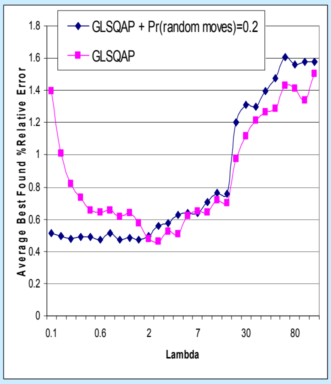
- Aspiration: if **G**(*s*) is better than best so far, then move to *s* even if **H**(*s*) is inferior
- Work for MaxSAT and QAP but not SAT
- Result generally improved at high λ value



G: Original Cost Function H: Augmented Cost Function

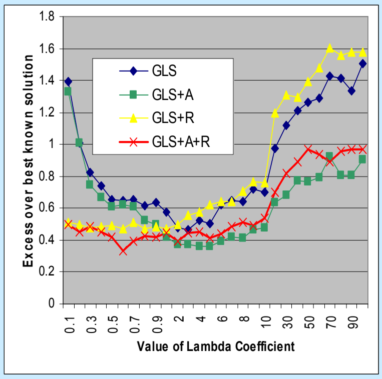
GLS + Randomness

- With probability *Pr* make random move
- Results improved in QAP at low λ value
- No effect on GLS → SAT / MAX SAT
- Randomness: when is it useful?



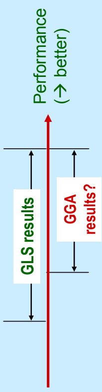
GLS + Aspiration + Randomness

- Result: performance is less sensitive to λ value
- Aspiration should become a standard feature of GLS
- Randomness sometimes helps
- Where/when will they succeed?



Guided Genetic Algorithm Overview

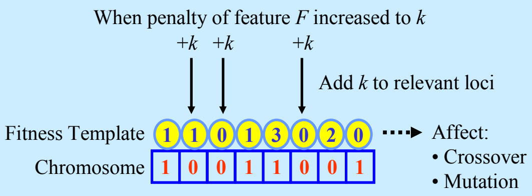
- Guided GA: Hybrid GLS + GA
- Using *Guided Local Search* as *meta-heuristic* for Genetic Algorithms
- Aims:
  - To extend the domain of GLS
  - To improve efficiency & effectiveness of GAs
  - To improve robustness of GLS



Guided Genetic Algorithm

- Initialize population
- Repeat
  - Run GA till best fitness remains unchanged for *n* generations (*n* is parameter)
  - Pick the best chromosome *X*
  - Penalize features of *X* according to GLS
  - Augment cost function
- Until Termination Condition

Using GLS Penalties in GGA



- High value in fitness template ⇒ instability

GGA Applications

- Royal Road Function
  - More effective than both GA and GLS
- Processors Configuration
- Radio Length Frequency Assignment
  - Gained robustness over GLS
- General Assignment Problem