

COOPERATIVE AND ADAPTIVE ALGORITHMS

Simulated
Annealing (SA)

OUTLINE

Physical Annealing,
Simulated Annealing (SA),
Examples
SA for TSP,
Adaptive SA,
Cooperative SA,
References.

HEURISTICS

- A **Heuristic** is simply a **rule of thumb** that hopefully will find a good answer.
- **Why** use a Heuristic?
 - Heuristics are typically used to solve **complex** (large, nonlinear, non-convex (i.e. contain local minima)) **multivariate combinatorial optimization problems** that are difficult to solve to optimality.
- Unlike gradient-based methods in a convex design space, heuristics are **NOT guaranteed** to find the true **global optimal solution** in a single objective problem, but should find many good solutions (the **mathematician's** answer vs. the **engineer's** answer)
- Heuristics are **good at dealing with local optima** without getting stuck in them while searching for the global optimum.

TYPES OF HEURISTICS

- **Two Special Cases of Heuristics**

- **Construction Methods**

- Must first find a feasible solution and then improve it.

- **Improvement Methods**

- Start with a feasible solution and just try to improve it.

- **EXAMPLES:**

- **Simulated Annealing**

- Genetic Algorithms

- Tabu Search

- New Methods: Particle Swarm Optimization, etc...

SIMULATED ANNEALING ANALOGY

- **Statistical Mechanics:** The behavior of systems with many degrees of freedom in thermal equilibrium at a finite temperature.
- **Combinatorial Optimization:** Finding the minimum of a given function depending on many variables.
- **Analogy:** If a liquid material **cools and anneals too quickly**, then the material will solidify into a **sub-optimal** configuration. If the liquid material **cools slowly**, the crystals within the material will solidify **optimally into a state of minimum energy** (i.e. ground state).
 - This ground state corresponds to the minimum of the cost function in an optimization problem.

PHYSICAL ANNEALING

Annealing is the thermal process of achieving low-energy state in a solid,

Physical annealing has been used since 5000 B.C.

A process used to make the material less fragile and improve its working properties,

Mainly used with glass and crystals.

PHYSICAL ANNEALING

At high temperatures, molecules move freely

At low temperatures, molecules are "stuck"

The structural properties of the cooled material depend on the rate of cooling.

PHYSICAL ANNEALING

The annealing process involves:

- Heating the material, till it reaches the *annealing temperature*,
- Holding the material at that temperature until it is even throughout,
- Cooling the material slowly (usually to room temperature)

PHYSICAL ANNEALING

The sequence of annealing times and temperatures is referred to as the *annealing* or *cooling schedule*,

The annealing schedule is very critical:

- Annealing times should be long enough for the material to undergo the required transformation,
- If the difference in the temperatures rate of change between the outside and inside of a material is too big, this may cause defects and cracks.

SIMULATED ANNEALING

Mimics the physical annealing process,

First procedure to simulate the annealing behaviour was developed by Metropolis in 1953 [1],

His algorithm simulates the change in energy of the material when it subjected to cooling process until it reaches a steady “frozen” state.

He developed a formula for the probability of an increase in energy.

SIMULATED ANNEALING

In thermodynamics, a state at a temperature t has a probability of an increase in the energy magnitude ΔE given by:

$$P(\Delta E) = e^{-\Delta E / k \times t}$$

where k is the Boltzman constant.

SA FOR OPTIMIZATION PROBLEMS

Applied to optimization problems independently by Kirkpatrick et. al in 1983 [2] and Černý in 1985 [3],

Used for locating a good approximation of the global optimum in large search spaces,

SIMULATED ANNEALING

- The approach is part of neighborhood search approaches to move from current solution to new solutions.
- It also tries to escape from local optimum by allowing a non-improving solution but in a controlled manner.

SIMULATED ANNEALING

Physical Annealing

State of a system

Energy of a state

Temperature



Simulated Annealing

Solution of a problem

Cost of a solution

Control parameter
(Temperature)



SIMULATED ANNEALING

Physical Annealing

Molecules move freely

Molecules are stuck

Simulated Annealing

Explore parameter space

Restrict exploration



SIMULATED ANNEALING

The *acceptance probability* is defined as:

$$P = \begin{cases} 1 & \text{if } \Delta c \leq 0 \\ e^{-\Delta c / t} & \text{if } \Delta c > 0 \end{cases}$$

where:

Δc is the change in the solution cost,

t is the current temperature (control parameter, no physical role).

Notice that we dropped Boltzmann's constant

SA BASIC ALGORITHM

Set current solution to an initial solution $s = s_0$

Set current temperature to an initial temperature $t = t_0$

Select a temperature reduction function α

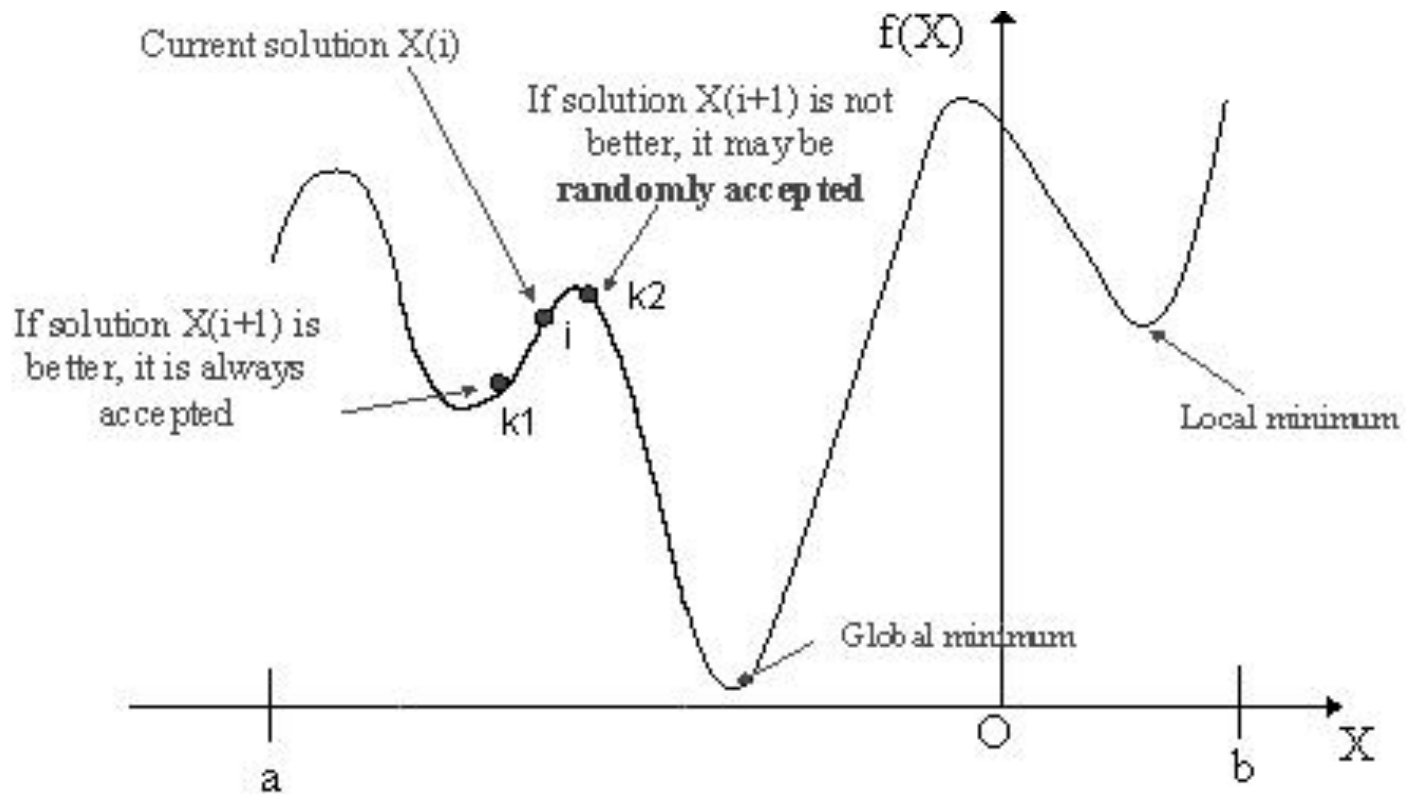
Repeat

- Repeat
 - Select a solution s_i from the neighborhood $N(s)$
 - Calculate change in cost according to the new solution ΔC
 - If $\Delta C < 0$ then accept new solution (it is improving) $s = s_i$
 - Else generate a random number x in the range $(0,1)$
 - If $x < e^{-\Delta C/t}$ then accept new solution $s = s_i$
- Until max no. of iterations for the temperature
- Decrease t using α

Until stopping conditions are satisfied

s is the final solution

SIMULATED ANNEALING



SIMULATED ANNEALING

SA always accepts better solutions,

SA can accept worse states according to some probability, the *acceptance probability*,

The acceptance probability is chosen so as to ultimately move to a better solution in the search space,

The acceptance probability is dependent on:

- The current temperature,
- The cost difference.

STRATEGY

At high temperatures, explore parameter space

At lower temperatures, restrict exploration

ACCEPTANCE PROBABILITY OF WORSE SLN.

Change	Temperature	Acceptance Probability
0.2	0.9	0.8007
0.4	0.9	0.6412
0.6	0.9	0.5134
0.8	0.9	0.4111

ACCEPTANCE PROBABILITY

Change	Temperature	Acceptance Probability
0.2	0.1	0.1353
0.4	0.1	0.0183
0.6	0.1	0.0025
0.8	0.1	0.0003

ACCEPTANCE PROBABILITY

As the temperature decreases, the probability of accepting a worse state decreases,

At zero temperature, no worse states are accepted,

ANNEALING SCHEDULE

The value of the temperature is adjusted according to the *annealing schedule*,

The annealing schedule is a mapping from time to temperature,

The annealing schedule is determined by:

- The initial temperature,
- The final temperature,
- The temperature decrement rule,
- Number of iterations at each temperature.

TEMPERATURE

The initial temperature:

- Must be high enough to allow a move to possibly any state in the search space,
- Must not be too hot, SA would behave as random search for a number of iterations,
- The maximum change of the cost function could be used as a guide to set this value,
- A good initial temperature is a one that accepts about 60% of the worse moves.

TEMPERATURE

The final temperature:

- Doesn't have to reach zero,
- Might take a very long time to reach zero when some decrement approaches are used,
- To stop the algorithm:
 - A reasonably low temperature,
 - The system is frozen, no better moves are generated neither any worse moves are getting accepted.

TEMPERATURE

Temperature decrement rules

- Linear

$$t = t - \alpha$$

- Geometric

$$t = t \times \alpha$$

TEMPERATURE

- In the geometric update rule
 - t should be between 0.8 and 0.99,
 - The higher the value of t , the longer it will take to reach the final (low) temperature
- Another rule is to reduce the temperature by $t = t / (1 + \beta t)$, β is small value which decreases the temperature very slowly and in this case use one iteration per temperature.

TEMPERATURE-ITERATIONS

The number of iterations at each temperature:

- Enough iterations should be allowed at every temperature for the system to be stable at that temperature,
- The required number of iterations might be large for large size problems
- Usually, a constant value is used.

TEMPERATURE -ITERATIONS

- We could dynamically change this value:
 - Allowing a small number of iterations at high temperatures,
 - Allowing a large number of iterations at low temperature to fully explore the local optimum.

CONVERGENCE OF SA

The behavior of SA Algorithm can be modeled using Markov chains.

It has been shown that under constant temperature and as long as it is possible to find a sequence of exchanges that will transform any solution to another with non-zero-probability, the process converges independent of the starting solution.

CONVERGENCE OF SA

In SA the temperature is not constant.

The process is still a Markov chain but non-homogeneous. It converges if the temperature is reduced to zero slowly (with a specific logarithmic form) and the number of iterations at each temperature is large (exponential in terms of the problem size).

CONVERGENCE

The convergence theory dictates a certain form of cooling schedule that is not practical to use.

However, the results give SA a theoretical backing not always possible with heuristic methods.

SIMULATED ANNEALING

Advantages:

- Ease of use,
- Providing good solutions for a wide range of problems.

Disadvantages:

- A lot of computer time for many runs,
- A lot of tuneable parameters.

SIMULATED ANNEALING

Applied successfully to many applications:

- Non-linear function optimization,
- Travelling Salesman Problem (TSP),
- **Scheduling Problems,**
- FPGA placement,
- Task allocation,
- Network design,
- Graph colouring and partitioning,
- Many more ...

SCHEDULING PROBLEMS

[HTTP://RIOT.IEOR.BERKELEY.EDU/RIOT/APPLICATIONS/SCHEDULING/OBJECTIVES.HTML](http://RIOT.IEOR.BERKELEY.EDU/RIOT/APPLICATIONS/SCHEDULING/OBJECTIVES.HTML)

Scheduling jobs to be processed on machine(s)

Objective functions

Makespan

Let C_i denote the completion time of the i^{th} job in a batch of n jobs given. The makespan, C_{\max} , is defined as,

$$C_{\max} = \max (C_1, C_2, \dots, C_n)$$

The makespan is the completion time of the last job.

A common problem of interest is to minimize C_{\max} ,

This criterion is usually used to measure the level of utilization of the machine.

OBJECTIVE FUNCTIONS

Total (Weighted) Completion Time

Let C_i denote the completion time of the i^{th} job in a batch of n jobs given. The total completion time is defined as,

$$\sum_{i=1}^n C_i$$

The total completion time is the sum of all the completion times of the jobs. It is commonly referred to as the **flow time**.

Let w_i denote the **weight assigned to the i^{th} job** in the a batch of n jobs given. The total weighted completion time is defined as,

$$\sum_{i=1}^n w_i C_i$$

A common problem is in minimizing the total (weighted) completion time. This problem allows one to find an indication to the total holding or inventory caused by the schedule.

OBJECTIVE FUNCTIONS

Lateness

Difference between completion time and the due time. $L_i = C_i - d_i$, where C_i is the completion of job i and d_i is the due date of job i .

Some of the problems involving lateness are:

1. Minimizing the total lateness. The total lateness is defined as,

$$\sum_{i=1}^n L_i$$

2. Minimizing the total weighted lateness. Let w_i denote the weight assigned to the i^{th} job. The total weighted lateness is defined as,

$$\sum_{i=1}^n w_i L_i$$

3. Minimizing the maximum lateness, L_{\max} . L_{\max} is defined as,

$$L_{\max} = \max (L_1, L_2, \dots, L_n)$$

OBJECTIVE FUNCTIONS

Tardiness

The tardiness of job i , T_i , is defined as $T_i = \text{Max}(0, C_i - d_i)$, where C_i is the completion of job i and d_i is the due time of job i .

Some of the problems involving tardiness are:

1. Minimizing the maximum tardiness

$$\text{Min}(\max(T_i))$$

2. Minimizing the number of tardy jobs. The number of tardy jobs is defined as,

$$N_T = \sum_{i=1}^n q(T_i)$$

where $q(x) = 1$ if $x > 0$

$q(x) = 0$ otherwise

TARDINESS (CONT'D)

Minimizing the total weighted tardiness. Let w_i denote the weight assigned to the i^{th} job. The total weighted tardiness is defined as,

n

$$\sum_{i=1} w_i T_i$$

$i=1$

MACHINE ENVIRONMENTS

Single Machine: Only one machine is available to process jobs. Each job has a single task. Every job is performed on the same machine.

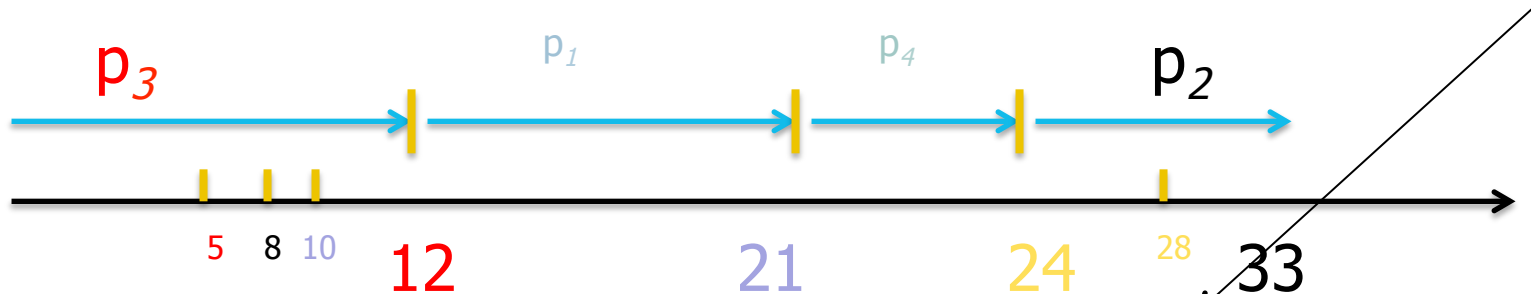
Parallel Machines: Multiple machines are available to process jobs. The machines can be identical, of different speeds, or specialized to only processing specific jobs. Each job has a single task.

EXAMPLE

single machine, minimize total weighted tardiness

	jobs	1	2	3	4
processing	p_j	9	9	12	3
due	d_j	10	8	5	28
weight	w_j	14	12	1	12

Let an initial solution be 3, 1, 4, 2



$$\sum w_j T_j = 1 \cdot \max(0, (12-5)) + 14 \cdot \max(0, (21-10)) + 12 \cdot \max(0, (24-28)) + 12 \cdot \max(0, (33-8)) = 461$$

SA

jobs	1	2	3	4
p_j	9	9	12	3
d_j	10	8	5	28
w_j	14	12	1	12

Neighborhood by swapping the order of jobs. Select one at random

Use geometric temperature reduction

$$\alpha = 0.9, t_0 = 0.9$$

Set no of iterations at the same temperature=2

Iteration 1 Consider the following neighbor (initial 3, 1, 4, 2 @461)

1, 3, 4, 2

$$\sum w_i T_i = 14 \cdot \max(0, (9-10)) + 1 \cdot \max(0, (21-5)) +$$

$$12 \cdot \max(0, (24-28)) + 12 \cdot \max(0, (33-8)) = 316$$

$\Delta C = -145$ then we accept the new solution as it is improving

jobs	1	2	3	4
p_j	9	9	12	3
d_j	10	8	5	28
w_j	14	12	1	12

1, 3, 4, 2@316

Generate a neighbor (say) 1, 4, 3, 2

$$\sum w_i T_i = 14 \cdot \max(0, (9-10)) + 12 \cdot \max(0, (12-28)) +$$

Iteration 2

$$1 \cdot \max(0, (24-5)) + 12 \cdot \max(0, (33-8)) = 319$$

$\Delta C = +3$ i.e not-improving, apply the acceptance criteria.

Calculate $e^{-\Delta C/t} = e^{-(319-316)/.9} = 0.035$

Generate a random number x between (0,1), say it was $x=0.01$ which is less than the acceptance probability, then we accept the solution.

So we accept a non-improving solution and we exceeded the max number of iterations at this temperature, so decrease the temperature

$$t = t \times \alpha, \quad t = 0.9 \times 0.9 = 0.81$$

MULTIPLE MACHINES MINIMUM MAKESPAN PROBLEM

Given processing times for n jobs, p_1, p_2, \dots, p_n , and an integer m , the Minimum Makespan Scheduling problem is to find an assignment of the jobs to m machines, M_1, M_2, \dots, M_m so that the final completion time (the makespan) is minimized.

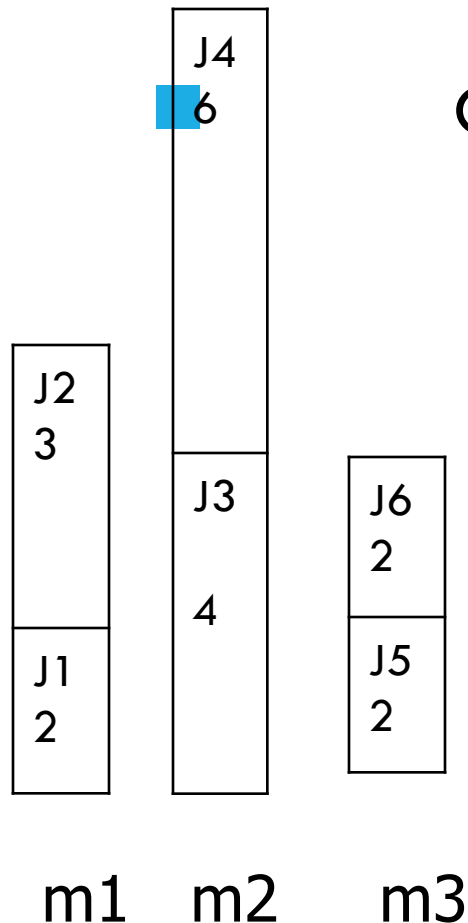
EXAMPLE

6 jobs with processing times: $p_1, p_2, p_3, p_4, p_5, p_6 = (2, 3, 4, 6, 2, 2)$

3 machines m_1, m_2, m_3 .

All machines should be assigned jobs.

$$P=(2,3,4,6,2,2)$$



Consider the following schedule

we can code this schedule as

Jobs 1 and 2 \rightarrow Machine m_1 Jobs 3 and 4 \rightarrow Machine m_2

$(1,1,2,2,3,3)$ indicating

machine assigned to

the corresponding jobs

Makespan=10

GENERATING NEIGHBORS

We can consider permutation of the machines numbers for the 6 jobs, e.g

$(1,1,1,1,2,3), (1,1,1,2,1,3), \dots$ there are a bit less than 3^6 as we exclude unassigned machines.

Notice that swap operation from the initial solution doesn't cover feasible solutions

SA

Initial soln (1,1,2,2,3,3)

- Select a permutation at random
- Use geometric temperature reduction

$$\alpha = .9, t_0 = 0.9$$

- Set no. of iterations at the same temperature=2

Consider the following neighbor

- Iteration 1
- (1,1,1,1,2,3) for p(2,3,4,6,2,2) will produce a makespan of 15 which is worse than the initial solution
 - Calculate $e^{-\Delta c/t} = e^{-5/0.9} = 0.0004$
 - Generate a random number x between (0,1), say it was $x=0.2$ which is larger than the acceptance probability so we reject the solution

SA

Iteration 2

Generate another neighbor, say $(1,2,3,1,2,3)$ with makespan of 8

This is an improvement of the current solution so we accept it.

And so on.

Optimal makespan for this problem is 7

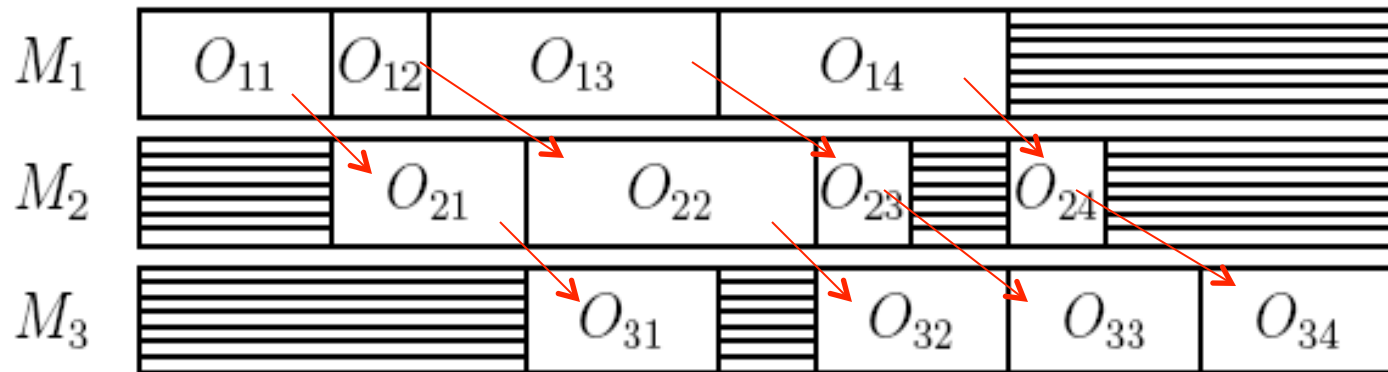
FLOW SHOP SCHEDULING

In the general flow shop model, there are a series of machines numbered $1, 2, 3 \dots m$. Each job has exactly m tasks. The first task of every job is done on machine 1, second task on machine 2 and so on. Every job goes through all m machines in a unidirectional order. However, the processing time each task spends on a machine varies depending on the job that the task belongs to. The precedence constraint in this model requires that for each job, task $i-1$ on machine $i-1$ must be completed before the i^{th} task can begin on machine i .

EXAMPLE

O_{kj} = Task k of job i

job	O_{1i}	O_{2i}	O_{3i}
1	2	2	2
2	1	3	2
3	3	1	2
4	3	1	2



JOB SHOP

In the general job shop model, there are a set of m machines. Jobs have tasks (or operations) not necessarily m as in the flow shop.

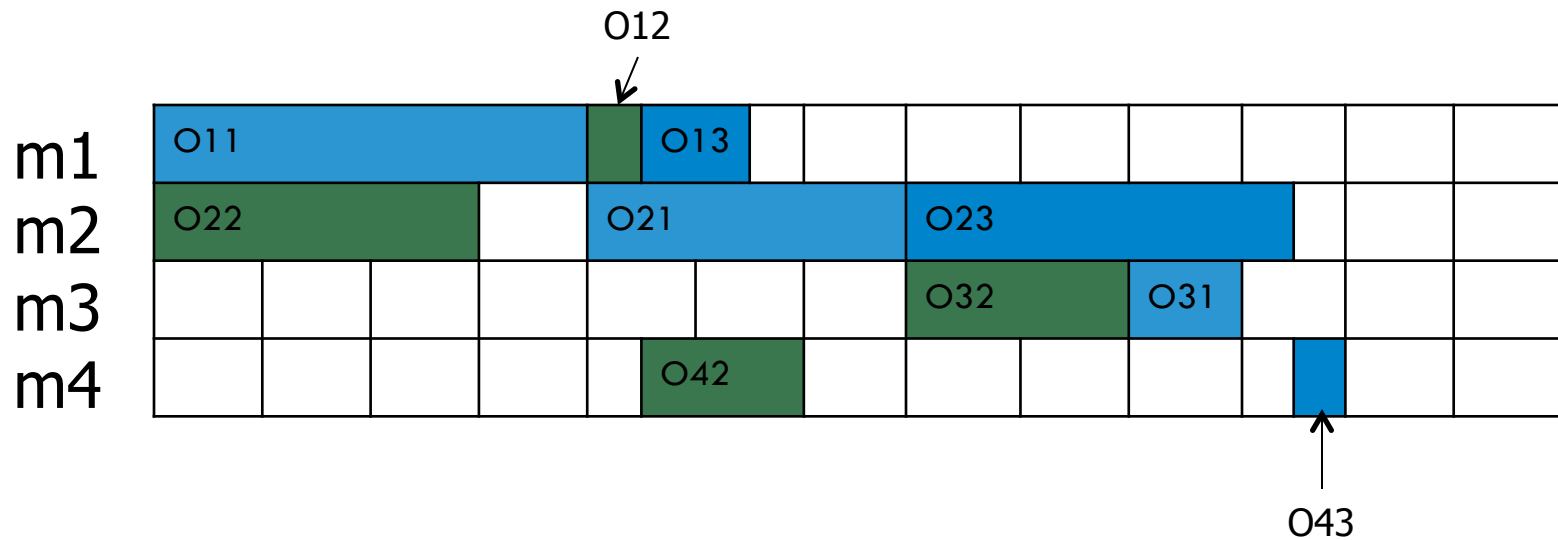
Each job uses the machines (or subset of them) in a specified sequence

The flow of the tasks in a job does not have to be unidirectional. Each job may also use a machine more than once.

EXAMPLE

$O_{\text{machine job}}$

Jobs	Machine Sequence	Processing Times
1	1,2,3	$O_{11}=8, O_{21}=6, O_{31}=2$
2	2,1,4,3	$O_{22}=6, O_{12}=1, O_{42}=3, O_{32}=4$
3	1,2,4	$O_{13}=2, O_{23}=7, O_{43}=1$



SA FOR TSP

Problem:

- Given n cities,
- Find a complete tour with minimal length.

Search space is **BIG**: for 30 cities there are $30! \approx 10^{32}$ possible tours.



SA FOR TSP

The solution would be a permutation of n cities:

$$Sol = [1, 2, 3, .., n]$$

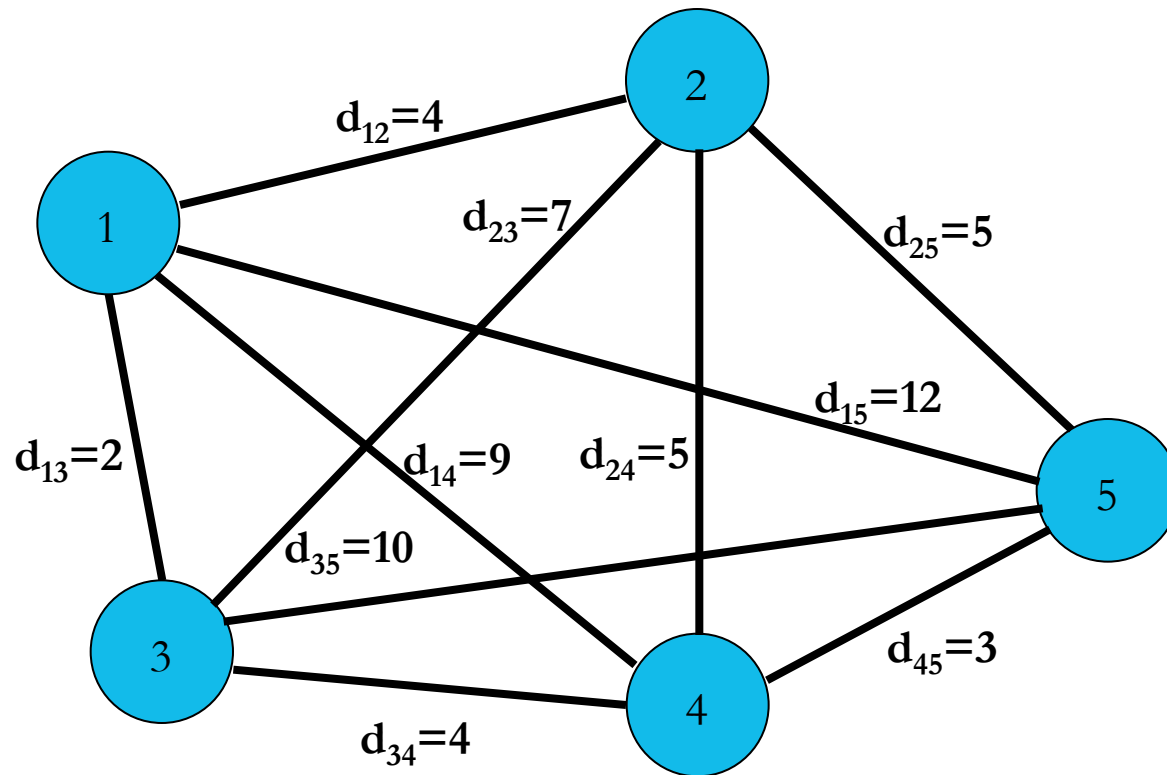
We should define a neighbourhood for the solution in order to generate the next one.

SA FOR TSP

The new solution could be generated by performing a predetermined number of swaps on the current solution,

Some adaptive implementations decrease the number of cities to swap during the run. Thus reducing the neighbourhood size as the search progresses.

SA FOR TSP - EXAMPLE




SA FOR TSP - EXAMPLE

Start with a random solution :

$$Sol = [1, 3, 4, 2, 5]$$

The total distance would be:

$$total_dist = \sum_{i=1}^{n-1} d_{Sol_i Sol_{i+1}} + d_{Sol_n Sol_1}$$


For a closed tour

SA FOR TSP - EXAMPLE

The tour length of the initial solution is:

$$total_dist = 2 + 4 + 5 + 5 + 12 = 28$$

To generate a candidate solution, select two random cities and swap them:

$$Sol = [1, 2, 4, 3, 5]$$

The tour length of the candidate solution is:

$$total_dist = 4 + 5 + 4 + 10 + 12 = 35$$

SA FOR TSP - EXAMPLE

Since the new solution has a longer tour length, it will be conditionally accepted according to a probability of (at higher temperatures, there's a higher probability of acceptance):

$$P = e^{-7 / t}$$

Assuming the new solution was not accepted, we generate a different one starting from the initial solution:

$$Sol = [1, 3, 4, 5, 2]$$

SA FOR TSP - EXAMPLE

The tour length of the candidate solution is:

$$total_dist = 2 + 4 + 3 + 5 + 4 = 18$$

Since this solution has a shorter tour length, it will get accepted,

The search continues ...

SA FOR TSP

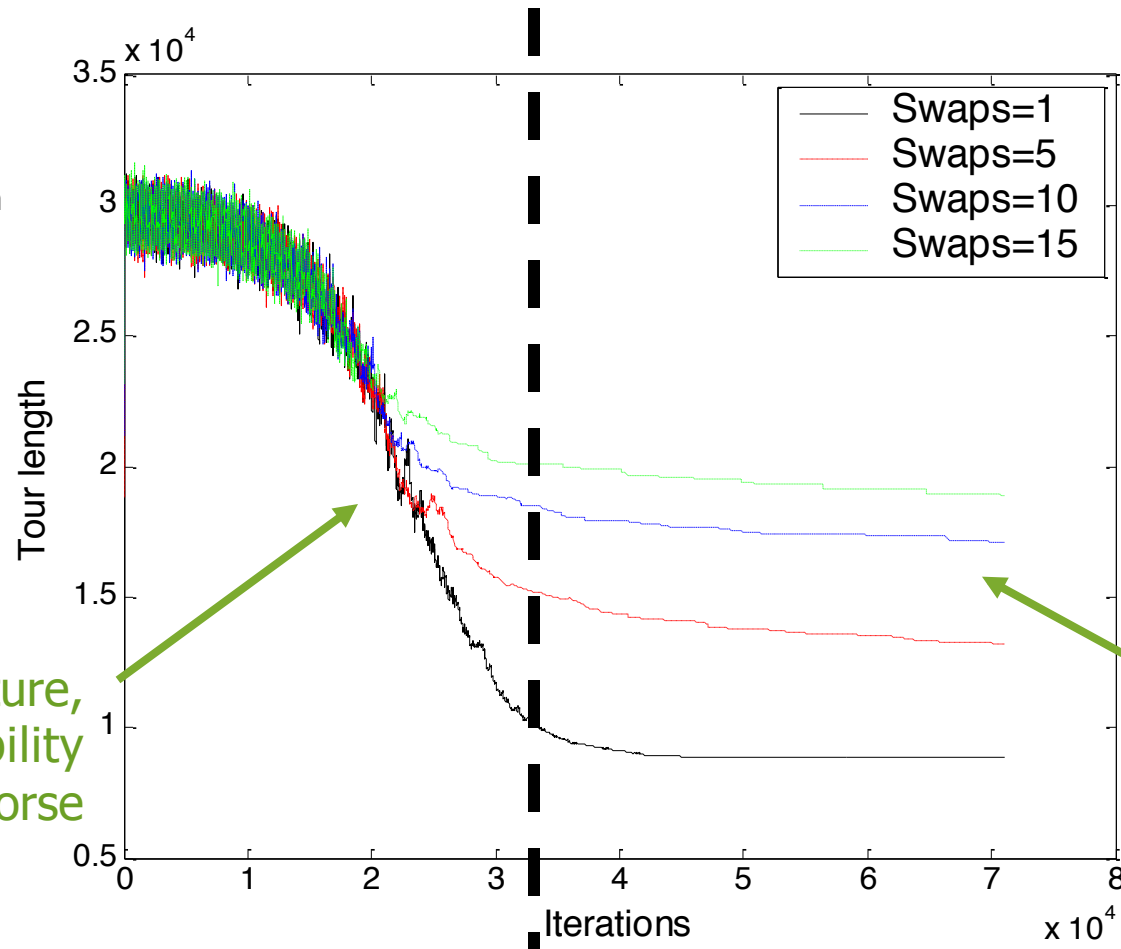
The SA was applied for the berlin52 TSP instance by having:

- $T_{\text{initial}} = 10000$,
- $T_{\text{Final}} = 0.1$,
- $\alpha = 0.85$,
- Using 1000 iteration for each T
- Using different number of swaps for obtaining the neighbouring solution (1, 5, 10 and 15),

SA FOR TSP

Number of swaps	Tour length
1	8861
5	13180
10	17124
15	18900

SA FOR TSP

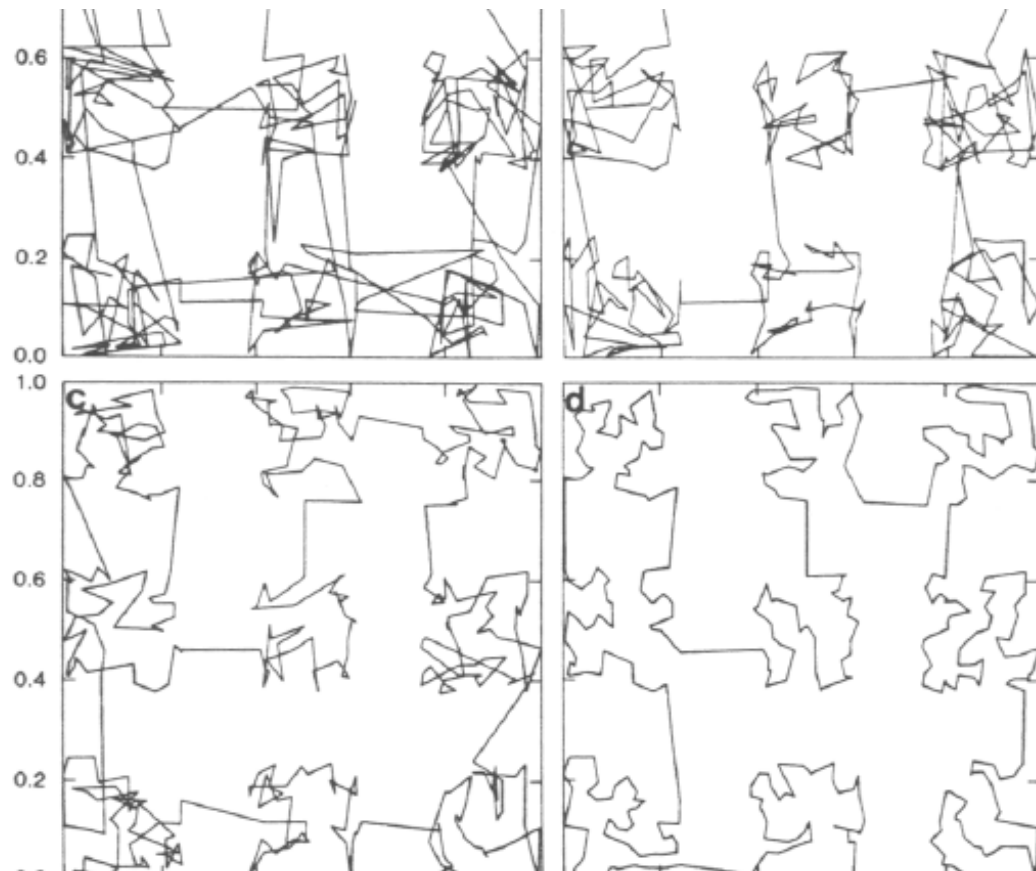


Green blotch here is an artifact of the plotting

High temperature, higher probability of selecting worse solutions

Low temperature, lower probability of selecting worse solutions

How
simulated
annealing
proceeds ...



ADAPTIVE SA

ADAPTATION

Adaptation in SA is in the parameters used by the algorithm:

- The *initial temperature*, the *cooling schedule* and *no. of iterations* per temperature being the most critical.
- Other components such the *cost function* and method of generating *neighborhood* solutions and *acceptance probability* affect the computational costs.

INITIAL TEMPERATURE

Finding the right temperature depends on the type of problem and sometime the instance of the problem.

One can try different values and see which leads to better solutions.

Some researchers suggested doing this adaptively using a search method such as Genetic Algorithm [7].

PROBABILITY OF ACCEPTANCE

The Boltzmann-based acceptance probability takes significant computational time ($\sim 1/3$ of the SA computations).

The idea of using a lookup table where the exponential calculation are done once for a range of values for change in c and t has been suggested.

Other non-exponential probability formulas have been suggested, for example Johnson et.al.[8] suggested $P(\delta c) = 1 - \delta c / t$

COST FUNCTION

Avoid cost functions that return the same value for many states (e.g number of colors in graph coloring problem). This type of functions doesn't lead the search.

Many problems have some constraints that can be represented in the cost function using penalty terms.

One way to make the algorithm more adaptive is to have a dynamically changing weighting of the penalty terms. So in the initial phase the constraints can be relaxed more than in advanced phases.

To reduce the computational time involved, update functions have been suggested.

ADAPTIVE SA

There have been some attempts at making the selection and control of SA totally adaptive.

One such attempt proposed by Ingber in 1993 [4]. ASA automatically adjusts the algorithm parameters that control the temperature schedule.

It requires the user to only specify the cooling rate α . All the other parameter are automatically determined.

ADAPTIVE SA

The method uses linear random combination of previously accepted steps and parameters to estimate new steps and parameters.

More information including source code available at:

<http://www.ingber.com/>

COOPERATION AND SA

COOPERATIVE SA

Cooperative SA (COSA) was proposed by Wendt et al. in 1997 [5],

COSA implements *concurrent* and *synchronous* runs of multiple SA processes,

The concurrent processes are coupled through the *cooperative transitions*,

The cooperative transition replaces the uniform distribution used to select the neighbours.

COOPERATIVE SA

The algorithm manipulates multiple solutions (a *population* of solutions) at once,

Assuming we have five solutions, we start by a randomly chosen population:

$$Pop_0 = [s_1, s_2, s_3, s_4, s_5]$$

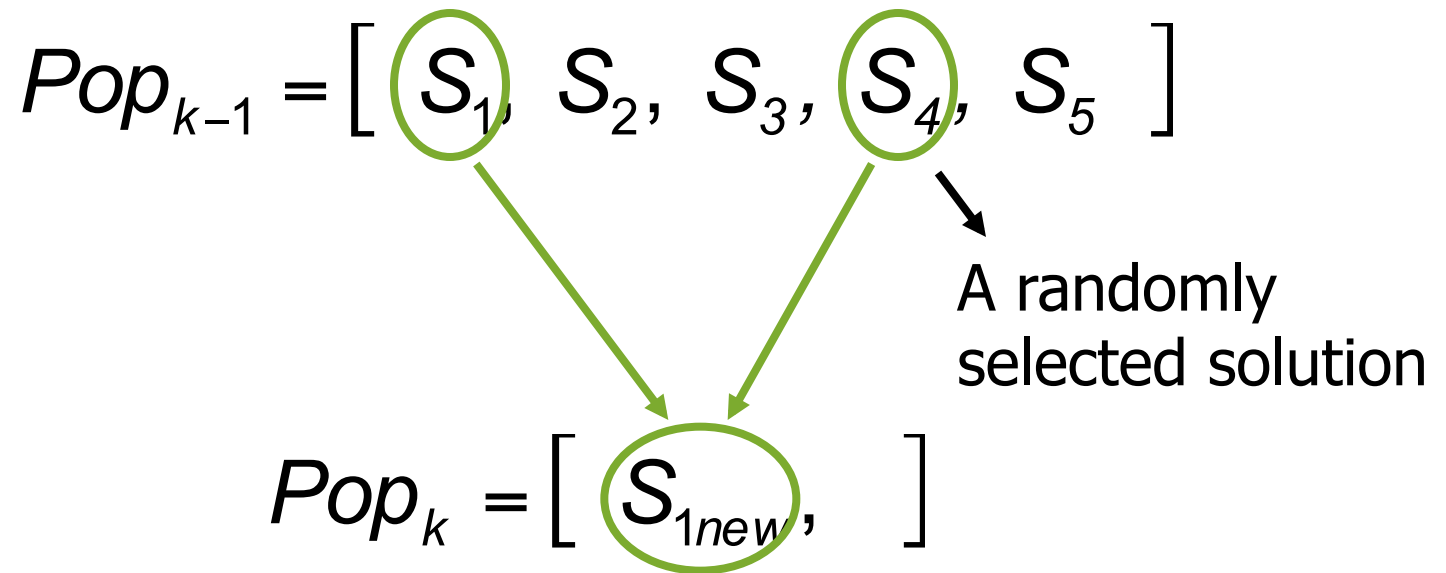
COOPERATIVE SA

The new population is iteratively produced,

Any new solution is cooperatively produced by:

- The previous value of that solution,
- The previous value of a randomly selected solution,

COOPERATIVE SA



COOPERATIVE SA

$$Pop_{k-1} = [s_1, s_2, s_3, s_4, s_5]$$


$$Pop_k = [s_{1new}, s_{2new},]$$

COOPERATIVE SA

How do we find $S_{1_{new}}$ from S_1 and S_4 ?

We find the neighbours of S_1 that are closer to S_4 than S_1 itself (using some distance measure),

These neighbours constitute what is known as the **CLOSER** set.

COOPERATIVE SA

The *CLOSER* set is defined as:

$$CLOSER = \{s_k \in N(S_1) \mid d(s_k, S_4) < d(s_k, S_1)\}$$

If the *CLOSER* set is not empty, the new solution is randomly selected from it,

Otherwise, the new solution is randomly selected from the neighbourhood of S_1 .

COOPERATIVE SA

The temperature is updated based on the difference of the mean fitness of the new and old populations,

$$\Delta E = E(Pop_k) - E(Pop_{k-1})$$

$$T = \begin{cases} T & , \text{if } \Delta E < 0 \\ \alpha T & , \text{otherwise} \end{cases}$$

COOPERATIVE SA

Set population size Pop ,

Initialize Pop_0 ,

For a determined number of transitions K

- For $i = 1$ to Pop
 - Select a random cooperator $S_j \in Pop_{k-1}$,
 - Generate $S_{inew} = cotrans(S_i, S_j)$,
 - Accept the new solution if it's better,
 - Otherwise, probabilistically determine if a move to be taken to the new solution and update the solution accordingly.

end

Update Temperature,

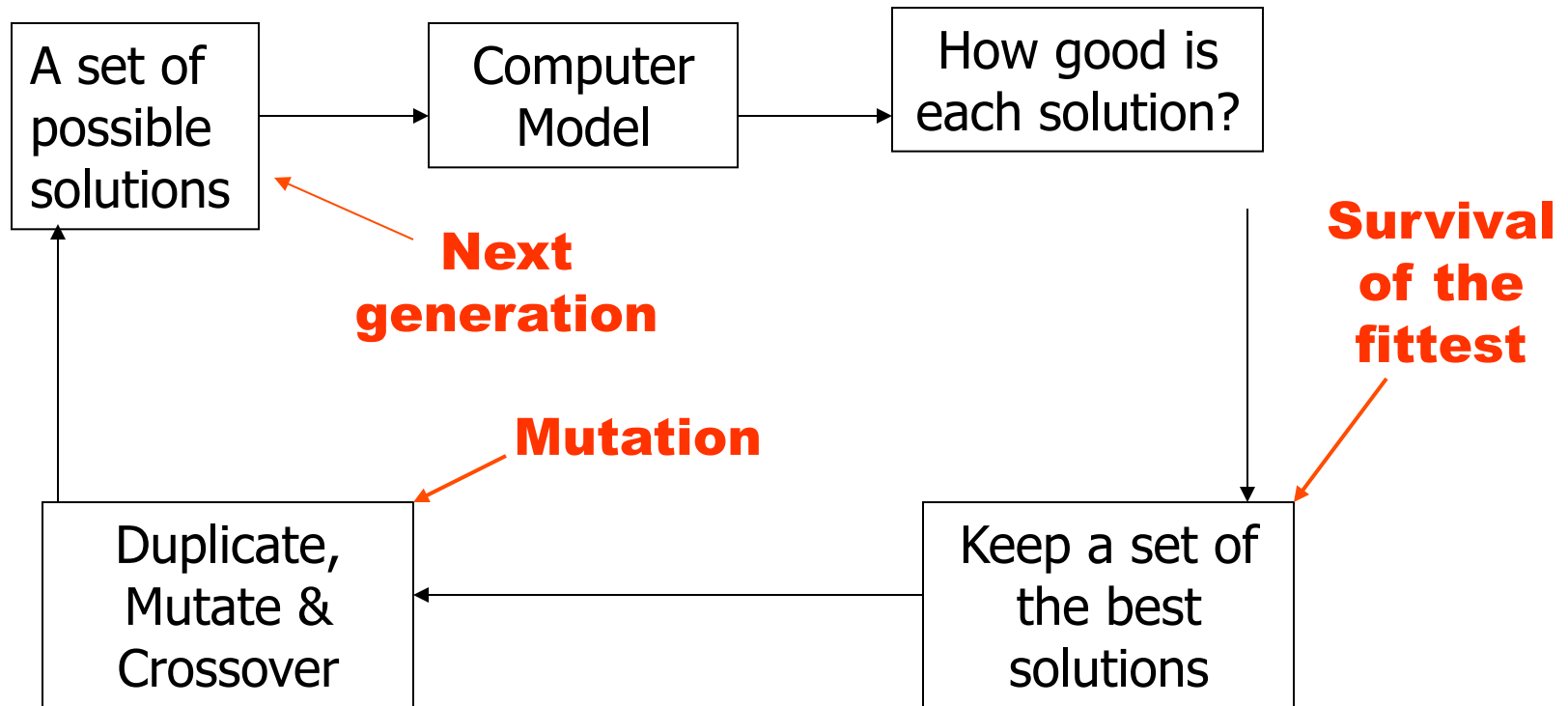
end


COOPERATIVE SA

COSA inherits the idea of population and information exchange from Genetic Algorithms.

It uses *cooperative transitions* instead of GA crossover.

In [5] they showed that using increased cooperation had positive effect on getting better solutions (TSP, Job Shop Scheduling, select of comm. Protocols).






Conservation of information theorems indicate that any search algorithm performs on average as well as random search without replacement unless it takes advantage of problem-specific information about the search target or the search-space structure.

Combinatorics shows that even a moderately sized search requires problem-specific information to be successful.

Three measures to characterize the information required for successful search are (1) endogenous information, which measures the difficulty of finding a target using random search;

(2) exogenous information, which measures the difficulty that remains in finding a target once a search takes advantage of problem-specific information; and

(3) active information, which, as the difference between endogenous and exogenous information, measures the contribution of problem-specific information for successfully finding a target.



A methodology is developed based on these information measures to gauge the effectiveness with which problem-specific information facilitates successful search. It then applies this methodology to various search tools widely used in evolutionary search.

INFORMATION TRANSFORMATION

“The [computing] machine does not create any new information, but it performs a very valuable transformation of known information.”

--Leon Brillouin, *Science and Information Theory* (Academic Press, New York, 1956).

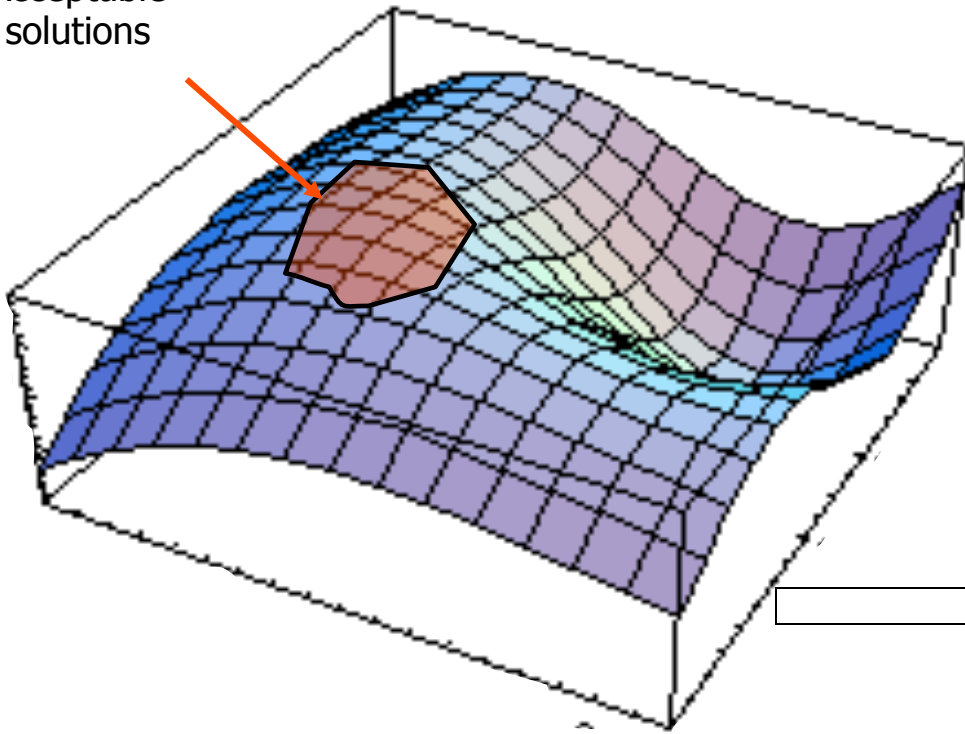
BERNOULLI'S PRINCIPLE OF INSUFFICIENT REASON

“in the absence of any prior knowledge, we must assume that the events have equal probability

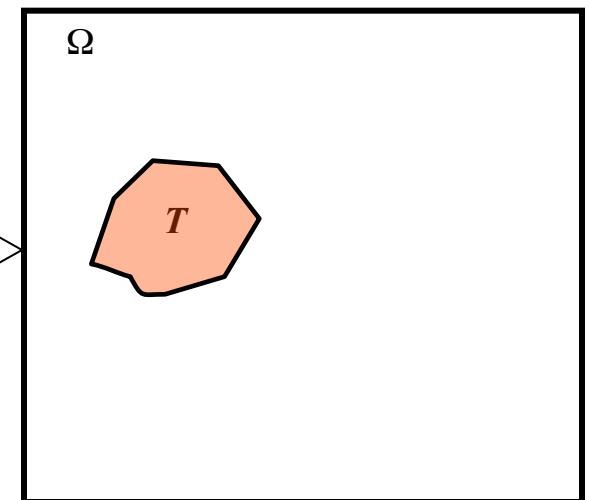
Jakob Bernoulli, “Ars Conjectandi” (“The Art of Conjecturing”), (1713).

FITNESS

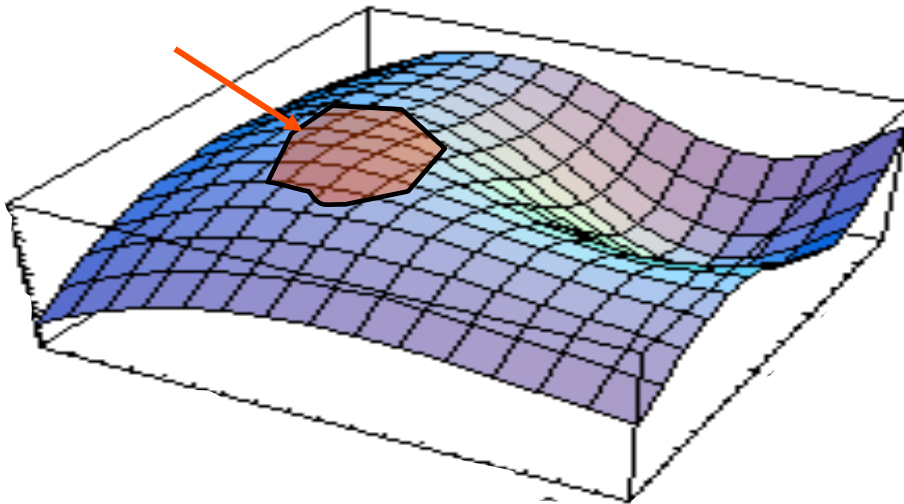
Acceptable
solutions



Each point in the parameter space has a fitness. The problem of the search is finding a good enough fitness.



SEARCH ALGORITHMS



Steepest Ascent
Exhaustive
Newton-Rapheson
Levenberg-Marquardt
Tabu Search
Simulated Annealing
Particle Swarm Search
Evolutionary Approaches

Problem: In order to work better than average, each algorithm implicitly assumes something about the search space and/or location of the target.

QUOTES ON THE NEED FOR ADDED INFORMATION FOR TARGETED SEARCH ...

1. "...unless you can make prior assumptions about the ... [problems] you are working on, then no search strategy, no matter how sophisticated, can be expected to perform better than any other" Yu-Chi Ho and D.L. Pepyne, (2001).
2. No free lunch theorems "indicate the importance of incorporating problem-specific knowledge into the behavior of the [optimization or search] algorithm." David Wolpert & William G. Macready (1997).

1. "Simple explanation of the No Free Lunch Theorem", Proceedings of the 40th IEEE Conference on Decision and Control, Orlando, Florida,
2. "No free lunch theorems for optimization", IEEE Trans. Evolutionary Computation 1(1): 67-82 (1997).

ACTIVE INFORMATION

Nothing works better, on the average, than random search.

For a search algorithm like evolutionary search to work, we require active information.



LAGRANGE MULTIPLIER OPTIMIZATION

The method of Lagrange multipliers gives a set of necessary conditions to identify optimal points of equality constrained optimization problems.

This is done by converting a constrained problem to an equivalent unconstrained problem with the help of certain unspecified parameters known as Lagrange multipliers.

The classical problem formulation

$$\begin{array}{ll} \text{minimize} & f(x_1, x_2, \dots, x_n) \\ \text{Subject to} & h_1(x_1, x_2, \dots, x_n) = 0 \end{array}$$

can be converted to

$$\text{minimize} \quad L(x, \lambda) = f(x) - \lambda h_1(x)$$

where

$L(x, \lambda)$ is the Lagrangian function

λ is an unspecified positive or negative constant called the Lagrangian Multiplier

LAGRANGE OPTIMIZATION

New problem is: minimize $L(x, \lambda) = f(x) - \lambda h_1(x)$

Suppose that we fix $\lambda = \lambda^*$ and the unconstrained minimum of $L(x; \lambda)$ occurs at $x = x^*$ and x^* satisfies $h_1(x^*) = 0$, then x^* minimizes $f(x)$ subject to $h_1(x) = 0$.

Trick is to find appropriate value for Lagrangian multiplier λ .

This can be done by treating λ as a variable, finding the unconstrained minimum of $L(x, \lambda)$ and adjusting λ so that $h_1(x) = 0$ is satisfied.

LM Approach: Example

$$1-2) \quad U = x_1 x_2 + 2x_1 \text{ s.t.} \quad B = 60 - 4x_1 - 2x_2 = 0$$

Form the Lagrangian function

$$3) \quad L = x_1 x_2 + 2x_1 + \lambda(60 - 4x_1 - 2x_2)$$

FOC

$$4) \quad L_\lambda = 60 - 4x_1 - 2x_2 = 0$$

$$5-6) \quad L_{x_1} = x_2 + 2 - \lambda 4 = 0; \quad \lambda = (1/4)x_2 + 1/2$$

$$7-8) \quad L_{x_2} = x_1 - \lambda 2 = 0; \quad \lambda = (1/2)x_1$$

$$9-10) \quad (1/4)x_2 + 1/2 = (1/2)x_1; \quad x_2 = 2x_1 - 2$$

$$11-12) \quad 60 = 4x_1 + 2(2x_1 - 2); \quad x_1^* = 8$$

$$13-14) \quad 60 = 4(8) - 2x_2; \quad x_2^* = 14$$

$$15-17) \quad U = (8)(14) + 2(8); \quad U^* = 128; \quad \lambda^* = 4$$

REFERENCES

1. N. Metropolis, A. W. Rosenbluth, M. Rosenbluth, A. H. Teller and E. Teller. "Equation of State Calculations by Fast Computing Machines". *J. Chem. Phys.* 21, pp. 1087-1092, 1953.
2. S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi. "Optimization by Simulated Annealing". *Science* 220, 671-680, 1983.
3. V. Cerny. "Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm". *J. Opt. Theory Appl.*, vol. 45, no. 1, pp. 41-51, 1985.
4. L. Ingber. "Adaptive Simulated Annealing: Lessons Learned". invited paper to a special issue of the Polish Journal *Control and Cybernetics* on "Simulated Annealing Applied to Combinatorial Optimization.", Vol.25, No.1, pp.33-54 1996.
5. O. Wendt and W. König. "Cooperative Simulated Annealing: How much Cooperation is Enough?". Research Report 97-19, Institute of Information Systems, GoetheUniversity Frankfurt, 1997.
6. Reeves, C., (ed) Modern Heuristic Techniques for Combinatorial Problems. Ch 2 by Kathryn Dowsland, 1993.

REFERENCES (CONT'D)

7. Miki, Hiroyasu, Wako and Yoshida Adaptive Temperature Schedule Determined by Genetic Algorithm for Parallel Simulated Annealing, CEC 2003.
8. David S. Johnson; Cecilia R. Aragon; Lyle A. McGeoch; Catherine Schevon, Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning, *Operations Research*, Vol. 39, No. 3. (May - Jun., 1991), pp. 378-406.