Assignment I
TSP with Simulated Annealing

- **Well known example : Traveling Sales-Person Problem (TSP)**

  "Given N cities, and distances, among them, find a shortest tour of N cities by visiting each city exactly once and returning to the starting city."

- **Algorithm of exponential complexity, NP complete.**
- **Many algorithms use greedy approach - local minimas.**

Example: *Objective Function*

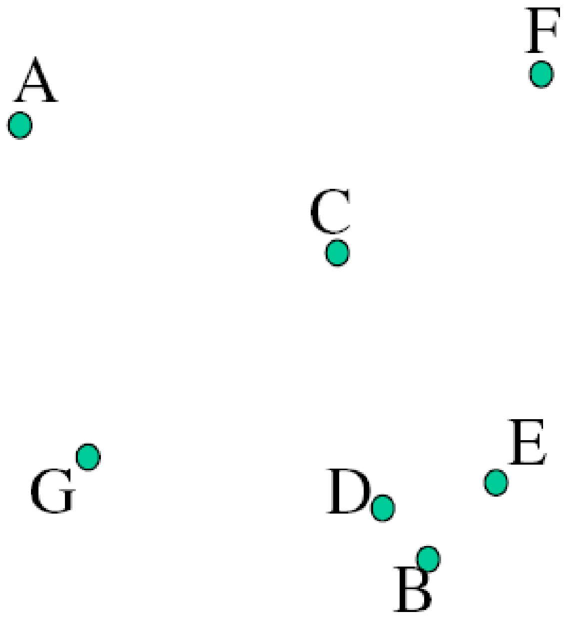
could be the sum of the distance between each pair of cities i.e. :

F = sqrt Σ $((x_i - x_{i-1})^2 + (y_i - y_{i-1})^2)$

Could be a more complex function taking into consideration traveling costs, traveling time, etc.

In our example, cost function will be given in form of a matrix.

Example : TSP



Cost of traveling :

	A	B	C	D	E	F	G
A	0	9	4	7	7	6	5
B	9	0	6	12	2	8	6
C	4	6	0	3	4	3	4
D	7	12	3	0	1	8	5
E	7	2	4	1	0	6	6
F	6	8	3	8	6	0	8
G	5	6	4	5	6	8	0

Let us choose randomly an initial path :

AGCFEBDA : cost = 5+4+3+6+2+12+7 = 39

Data in Assignment I:

You have to solve three TSP problem with 10, 30 and 50 cities.

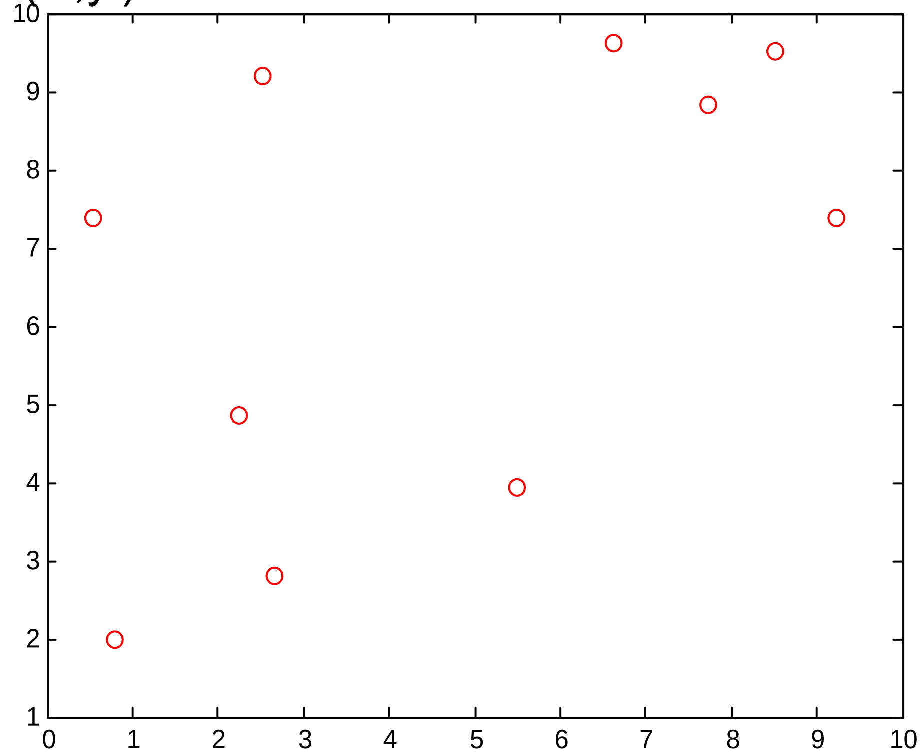
Coordina_10 : coordinates for 10 cities (xi,yi)

Coordina_30: coordinates for 30 cities (xi,yi)

Coordina_50: coordinates for 50 cities (xi,yi)

For example **<u>Coordina_10</u>**:

2.5346	9.1670
0.5659	7.3454
2.2720	4.8172
0.8226	1.9721
7.7530	8.8111
5.5220	3.8949
2.6775	2.7794
9.2573	7.3441
6.6485	9.5929
8.5379	9.4993



Create Cost Matrix:

0	2.6822	4.3577	7.3958	5.2305	6.0597	6.3892	6.9655	4.1359	6.0125
2.6822	0	3.0500	5.3794	7.3350	6.0389	5.0306	8.6914	6.4845	8.2578
4.3577	3.0500	0	3.1930	6.7818	3.3783	2.0778	7.4283	6.4777	7.8220
7.3958	5.3794	3.1930	0	9.7367	5.0776	2.0230	10.0001	9.5926	10.7789
5.2305	7.3350	6.7818	9.7367	0	5.3987	7.8830	2.1012	1.3532	1.0439
6.0597	6.0389	3.3783	5.0776	5.3987	0	3.0554	5.0842	5.8083	6.3644
6.3892	5.0306	2.0778	2.0230	7.8830	3.0554	0	8.0081	7.8862	8.9164
6.9655	8.6914	7.4283	10.0001	2.1012	5.0842	8.0081	0	3.4443	2.2721
4.1359	6.4845	6.4777	9.5926	1.3532	5.8083	7.8862	3.4443	0	1.8917
6.0125	8.2578	7.8220	10.7789	1.0439	6.3644	8.9164	2.2721	1.8917	0

Sample solution:

- route =

6 3 7 8 5 1 2 4 9 10

Objective function Value:

- **long =**
- **40.3418**

Simulated Annealing

1. Select an initial (feasible) solution s_0 6 3 7 8 5 1 2 4 9 10
2. Select an initial temperature $t_0 > 0$
3. Select a cooling schedule CS
4. Repeat

   Repeat

   6 10 9 4 2 1 5 8 7 3

   Randomly select $s \in N(s_0)$ (N = neighborhood structure)

   $\delta = f(s) - f(s_0)$ (f = objective function)

   If $\delta < 0$ then $s_0 \leftarrow s$

   Else

   Generate random x (uniform distribution in the range $(0, 1)$)

   If $x < \exp(-\delta/t)$ then $s_0 \leftarrow s$

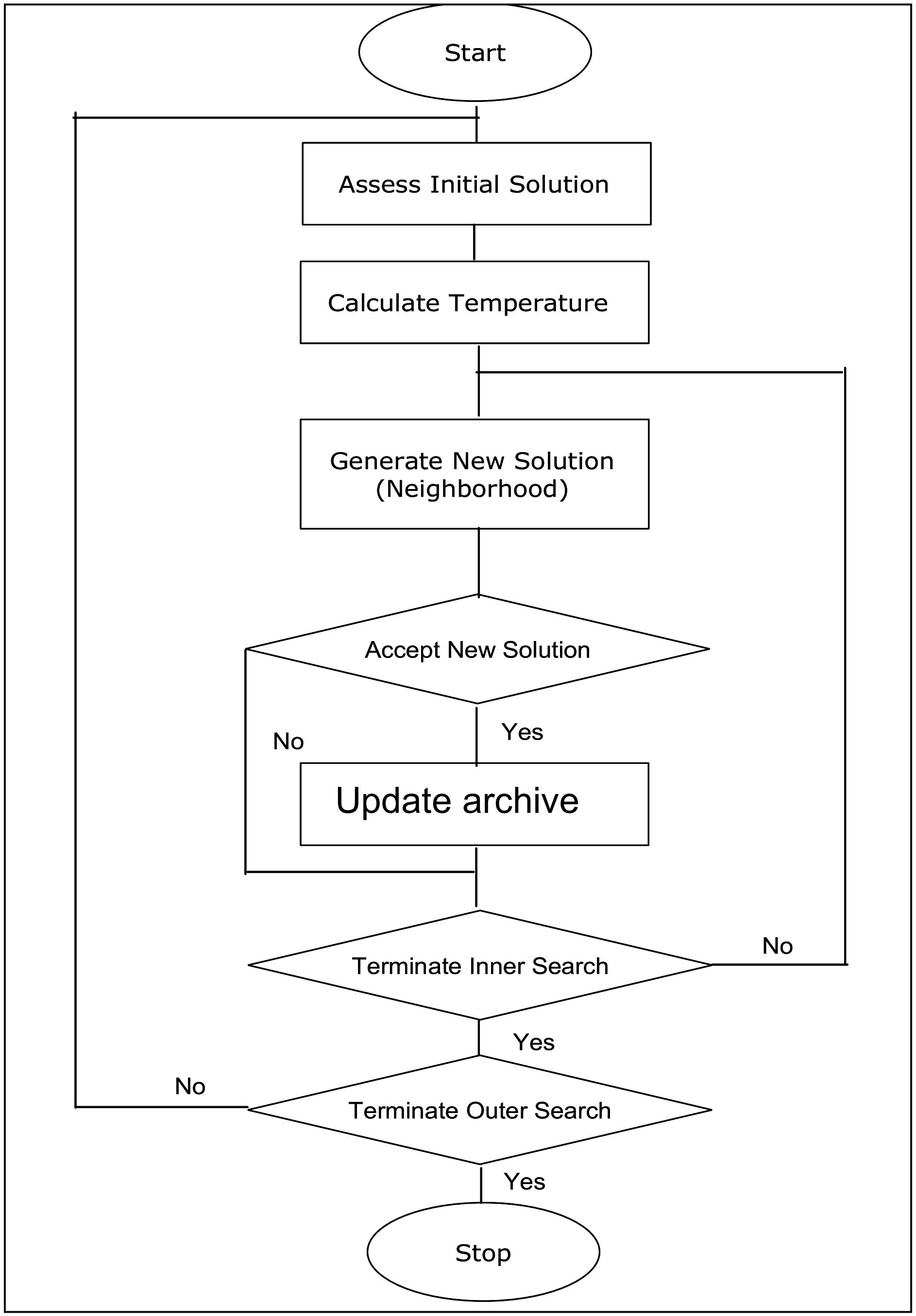
   Until max. number of iterations $ITER$ reached

   $t \leftarrow CS(t)$
5. Until stopping condition is met

Simulated Annealing

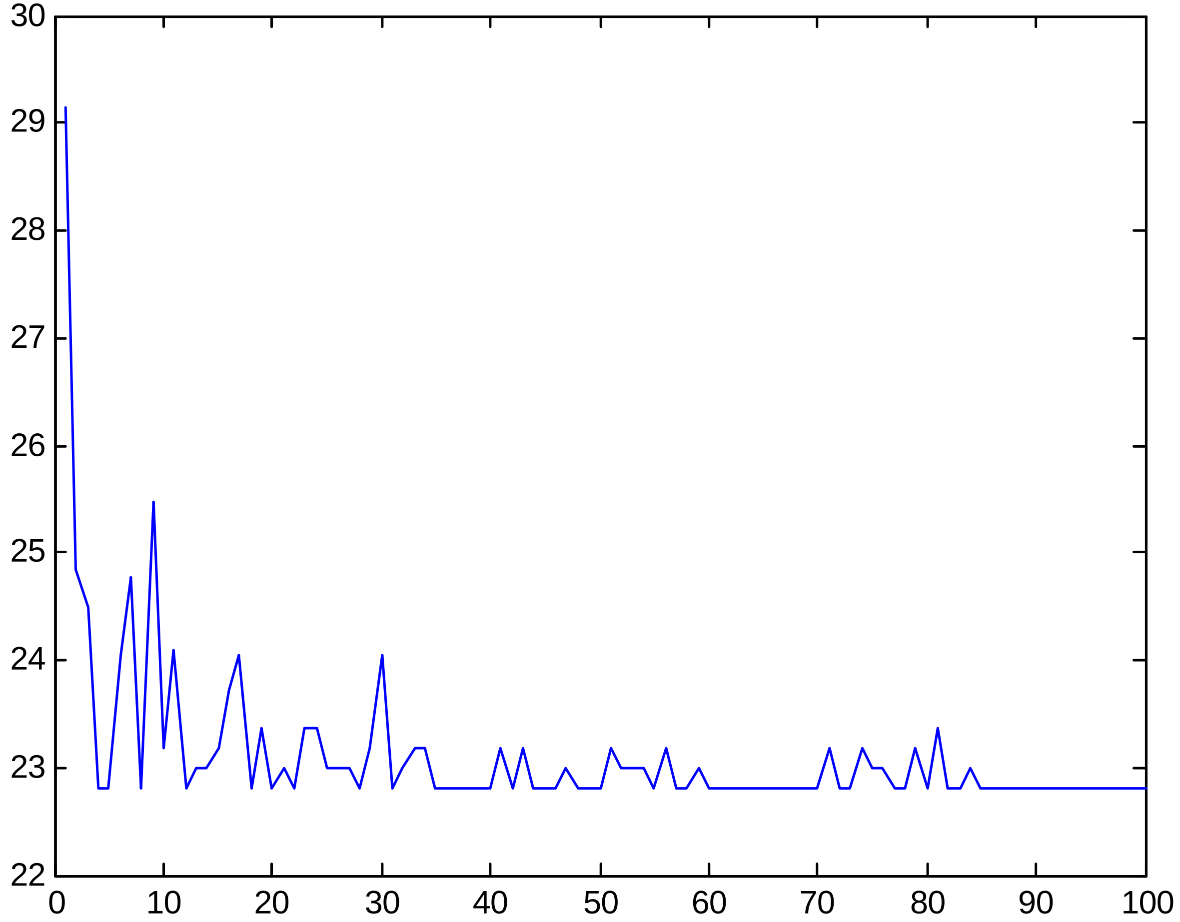
SA generates local movements in the neighborhood of the current state, and accepts a new state based on a function depending on the current "temperature" t. The two main parameters of the algorithm are $ITER$ (the number of iterations to apply the algorithm) and CS (the cooling schedule), since they have the most serious impact on the algorithm's performance.

Despite the fact that it was originally intended for combinatorial optimization, other variations of simulated annealing have been proposed to deal with continuous search spaces.

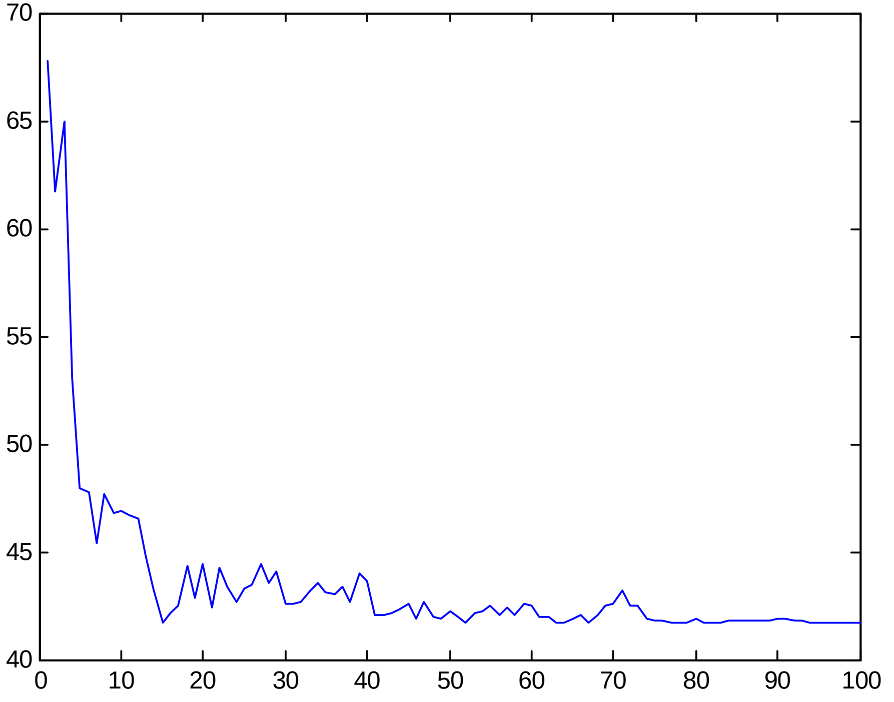


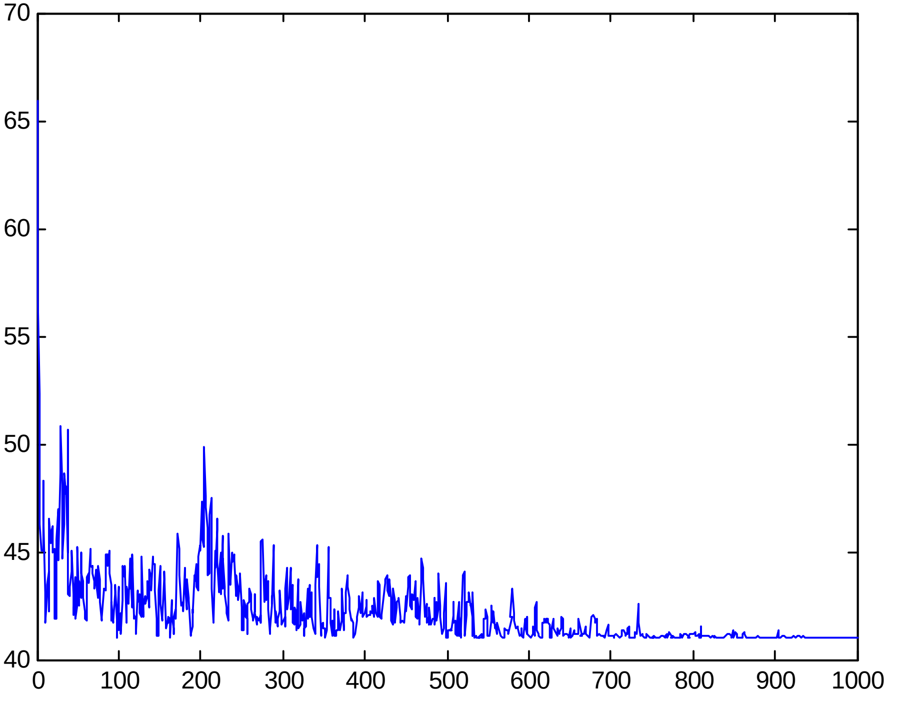
Start
Assess Initial Solution
Calculate Temperature
Generate New Solution (Neighborhood)
Accept New Solution
Yes
No
Update archive
Terminate Inner Search
No
Yes
Terminate Outer Search
No
Yes
Stop

Search for 10 Cities

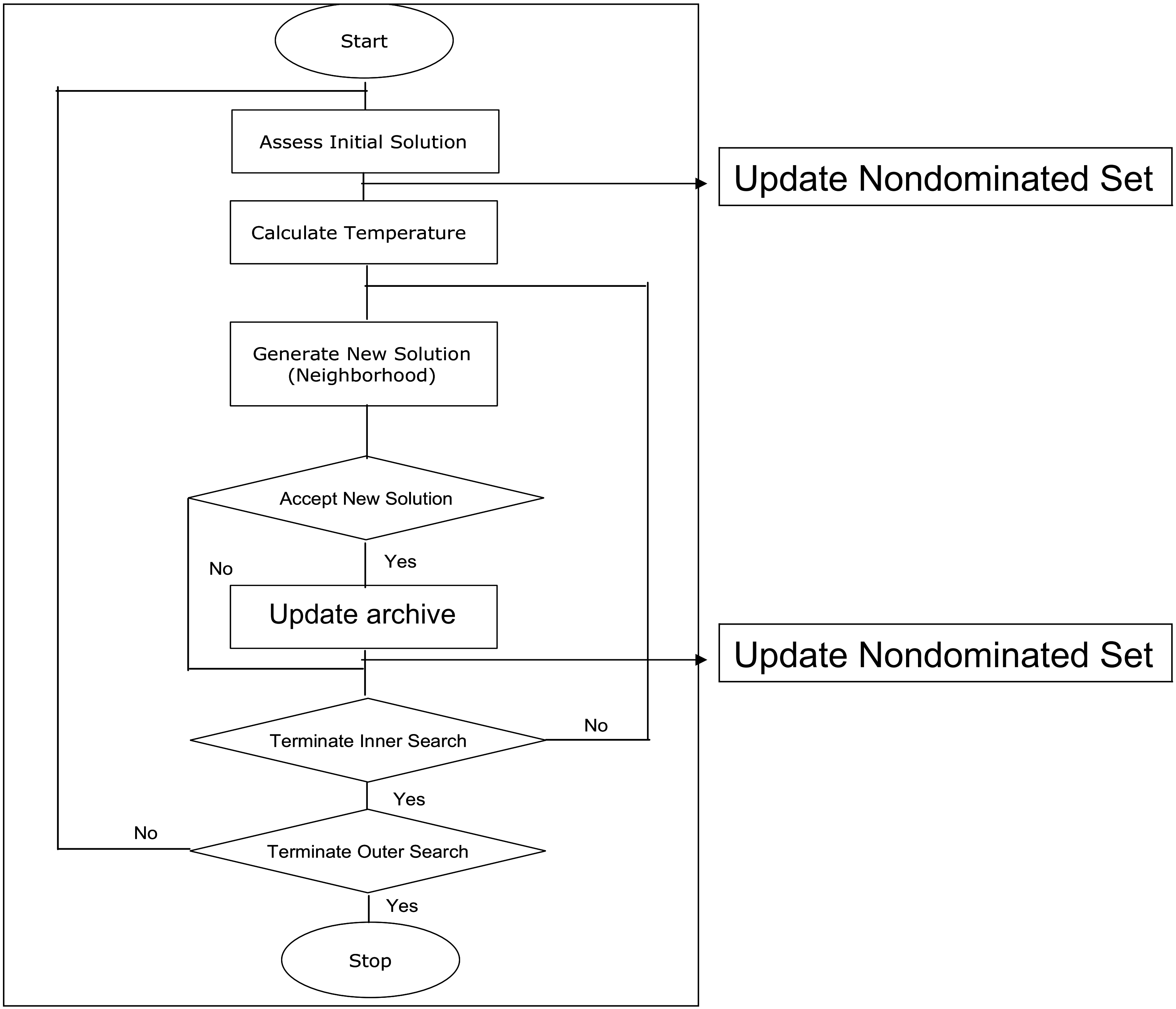


Search for 30 Cities





Multi-Objective Simulated Annealing


Start
Assess Initial Solution
Update Nondominated Set
Calculate Temperature
Generate New Solution (Neighborhood)
Accept New Solution
No
Yes
Update archive
Update Nondominated Set
Terminate Inner Search
No
Yes
No
Terminate Outer Search
Yes
Stop

Multi-Objective

Simulated Annealing

The key in extending simulated annealing to handle multiple objectives lies in determining how to compute the probability of accepting an individual $\vec{x}'$ where $f(\vec{x}')$ is dominated with respect to $f(\vec{x})$.

Sanghamitra Bandyopadhyay, Sriparna Saha, Ujjwal Maulik and Kalyanmoy Deb.
A Simulated Annealing Based Multi-objective Optimization Algorithm: AMOSA
, IEEE Transactions on Evolutionary Computation, Volume 12, No. 3, JUNE 2008, Pages 269-283.

Case 1:

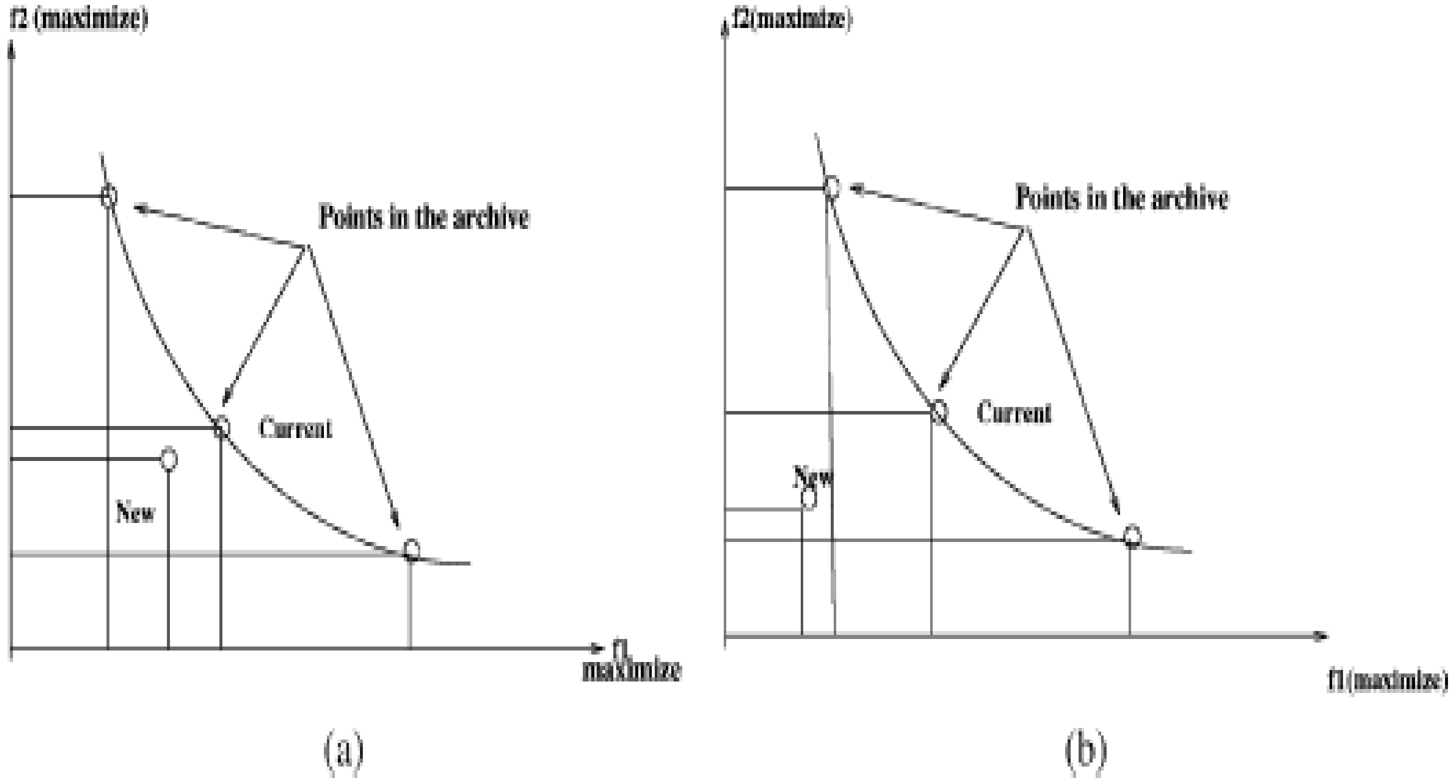


Fig. 3. Different cases when *New* is dominated by *Current*. (a) *New* is nondominating with respect to the solutions of *Archive* except *Current*, if it is in the *Archive*. (b) Some solutions in the *Archive* dominate *New*.

Case 2:

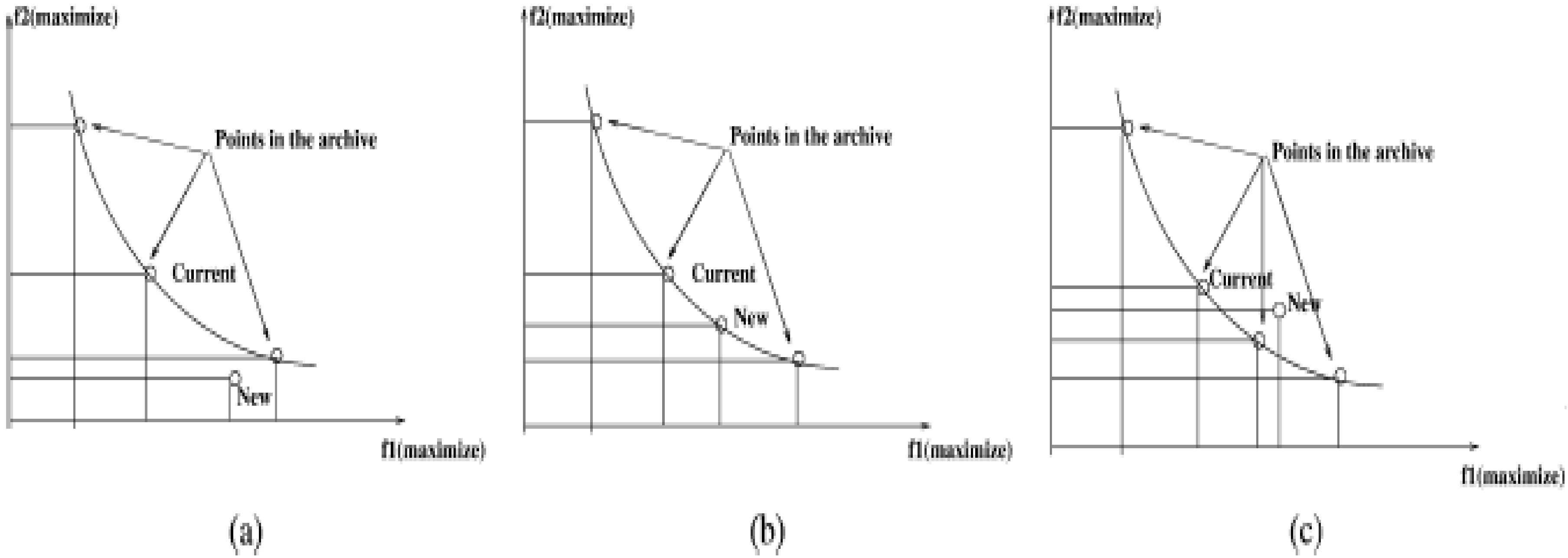


Fig. 4. Different cases when *New* and *Current* are nondominating. (a) Some solutions in *Archive* dominates *New*. (b) *New* is nondominating with respect to all the solutions of *Archive*. (c) *New* dominates $k(k \geq 1)$ solutions in the *Archive*.

Case 3:

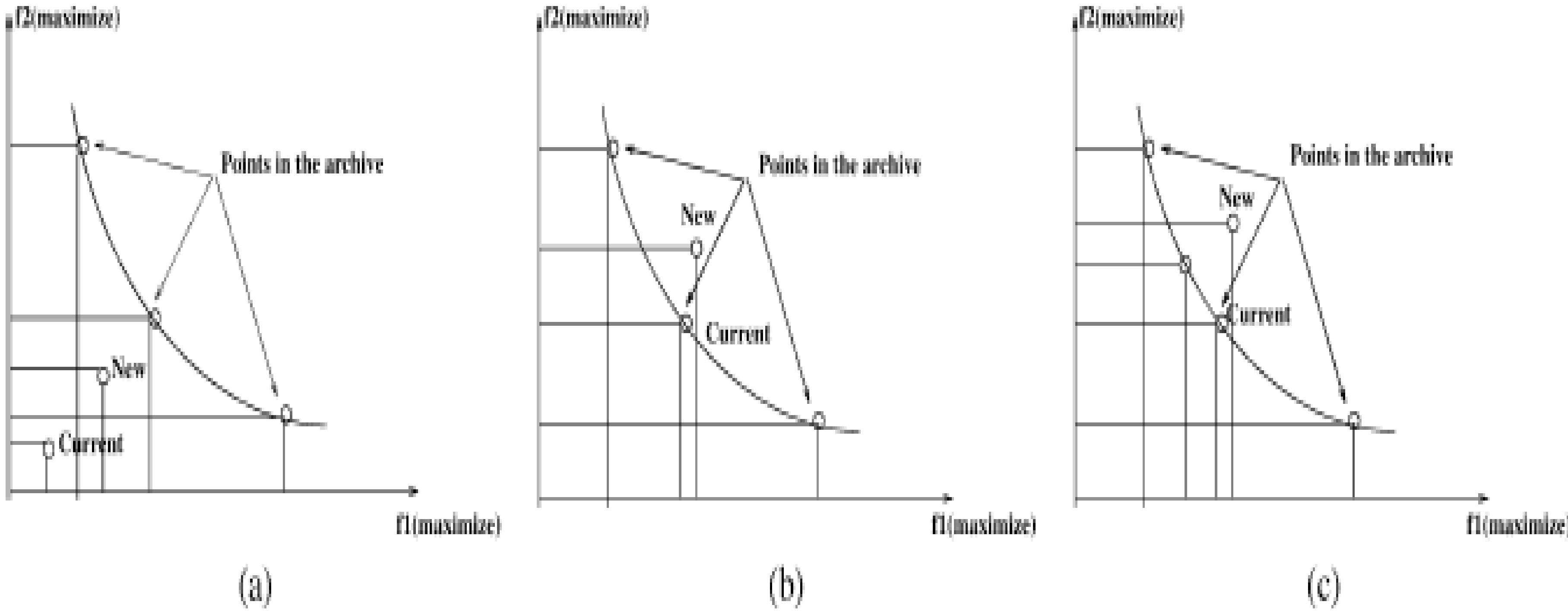


Fig. 5. Different cases when *New* dominates the *Current*. (a) *New* is dominated by some solutions in *Archive*. (b) *New* is nondominating with respect to the solutions in the *Archive* except *Current*, if it is in the *Archive*. (c) *New* dominates some solutions of *Archive* other than *Current*.

Some multiobjective versions of SA are the following:

- Serafini (1994): Uses a target-vector approach to solve a bi-objective optimization problem (several possible transition rules are proposed).
- Ulungu (1993): Uses an L_∞-Tchebycheff norm and a weighted sum for the acceptance probability.
- Czyzak & Jaszkiewicz (1997,1998): Population-based approach that also uses a weighted sum.
- Ruiz-Torres et al. (1997): Uses Pareto dominance as the selection criterion.
- Suppapitnarm et al. (1999,2000): Uses Pareto dominance plus a secondary population.

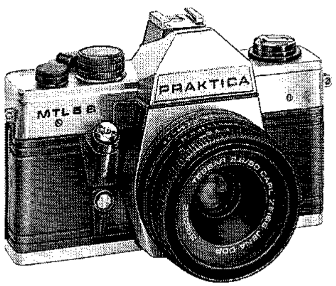
The 0-1 Knapsack Problem

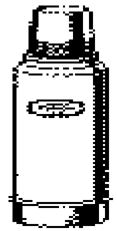
weight = 750g
profit = 5

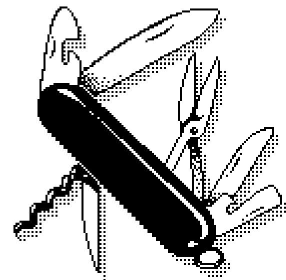
weight = 1500g
profit = 8

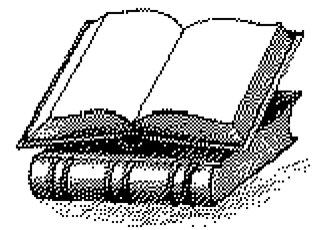
weight = 300g
profit = 7

weight = 1000g
profit = 3





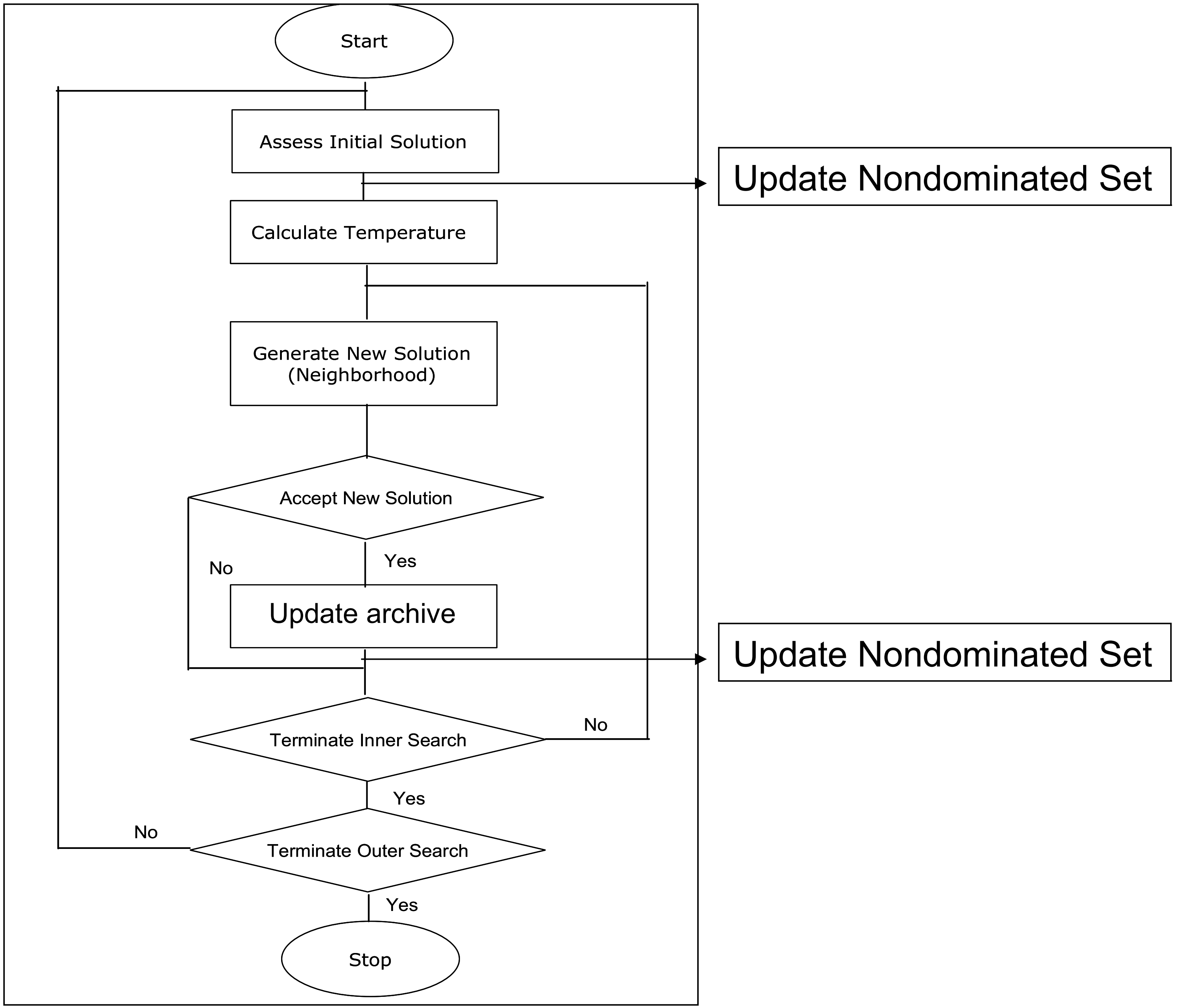




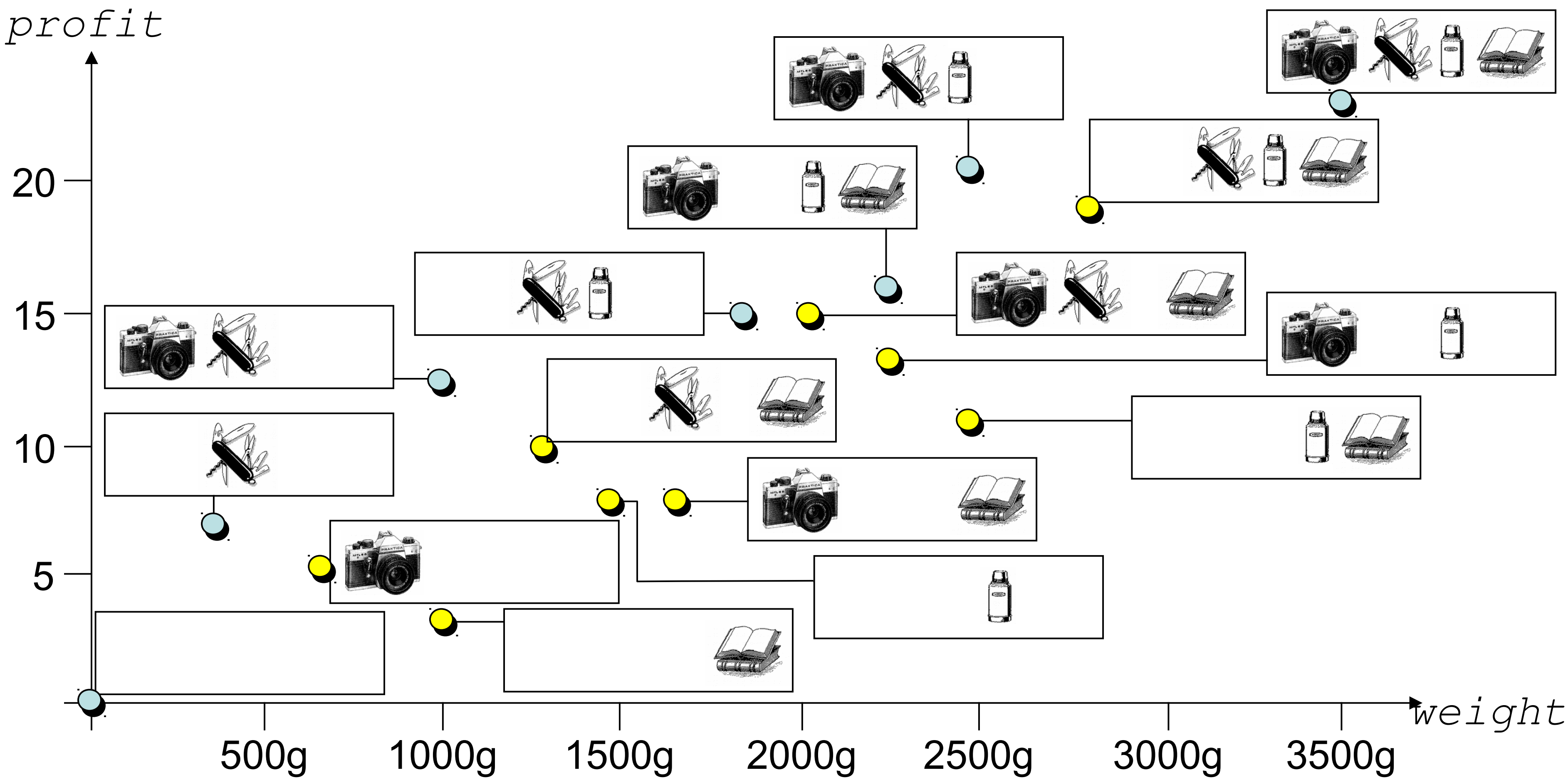
?

Goal: choose subset that

- maximizes overall profit
- minimizes total weight


Start
Assess Initial Solution
Update Nondominated Set
Calculate Temperature
Generate New Solution
(Neighborhood)
Accept New Solution
No
Yes
Update archive
Update Nondominated Set
No
Terminate Inner Search
Yes
No
Terminate Outer Search
Yes
Stop

The Solution Space



The 0-1 Multiple Knapsack Problem

The 0-1 multiple knapsack problem (0-1 MKP) is a maximization problem. It is a generalization of the 0-1 simple knapsack problem, and is a well-known member of the NP-hard class of problems. In the simple knapsack problem, a set of objects $O = \{o_1, o_2, o_3, ..., o_n\}$ and a knapsack of capacity C are given. Each object o_i has an associated profit p_i and weight w_i. The objective is to find a subset $S \subseteq O$ such that the weight sum over the objects in S does not exceed the knapsack capacity and yields a maximum profit. The 0-1 MKP involves m knapsacks of capacities $c_1, c_2, c_3, ..., c_m$. Every selected object must be placed in all m knapsacks, although neither the weight of an object o_i nor its profit is fixed, and will probably have different values in each knapsack (see Figures 4.2 and 4.3). A small problem with 10 objects and two knapsacks is defined in Table 4.1.

Table 4.1. A sample problem with ten objects and two knapsacks.

Object number	Knapsack 1 Capacity = 38		Knapsack 2 Capacity = 35	
	Weight	Profit	Weight	Profit
1	9	2	3	3
2	8	7	4	9
3	2	4	2	1
4	7	5	4	5
5	3	6	9	3
6	6	2	5	8
7	1	7	4	2
8	3	3	8	6
9	9	7	3	1
10	3	1	7	3

Table 4.2. The Pareto set for the sample problem with ten objects and two knapsacks.

Knapsack 1 Profit	knapsack 2 Profit	Objects in Knapsacks
39	27	{2, 3, 4, 5, 7, 8, 9}
38	29	{2, 3, 4, 5, 6, 7, 9}
36	30	{2, 3, 5, 6, 7, 8, 9}
35	32	{2, 3, 4, 6, 7, 8, 9}
34	33	{2, 3, 4, 5, 6, 8, 9}
32	34	{2, 4, 6, 7, 8, 9, 10}
29	35	{1, 2, 3, 4, 5, 6, 8}
27	36	{1, 2, 4, 6, 7, 8, 10}

Domination

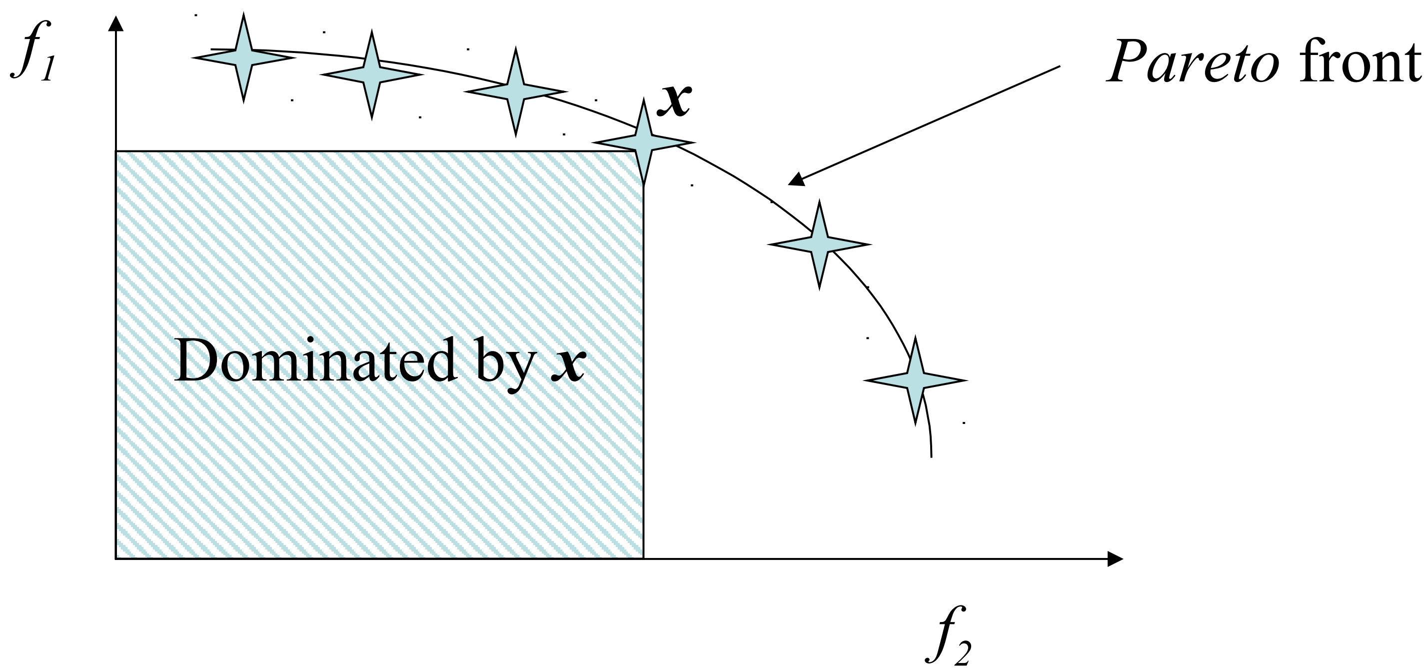
- For any two solutions x^1 and x^2
- x^1 is said to **dominate** x^2 if these conditions hold:
  - – x^1 is **not worse** than x^2 in all objectives.
  - – x^1 is **strictly better** than x^2 in at least one objective.
- If one of the above conditions does not hold x^1 does not dominate x^2.

Examples for minimization

Candidate	f_1	f_2	f_3	f_4
1 (dominated by: 2,4,5) (dominated by: 5)	5	6	3	10
2	4	6	3	10
3 (non-dominated)	5	5	2	11
4 (non-dominated)	5	6	2	10
5 (non-dominated)	4	5	3	9

Dominance

- we say *x* dominates *y* if it is at least as good on all criteria and ***better*** on at least one



Pareto Optimal Set

- **Non-dominated set**: the set of all solutions which are not dominated by any other solution in the **sampled search space**.

- **Global Pareto optimal set**: there exist no other solution in the **entire search space** which dominates any member of the set.