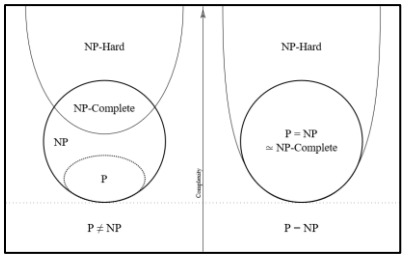
UiO : Matematisk institutt
Det matematisk-naturvitenskapelige fakultet

STK-4051/9051 Computational Statistics Spring 2022
Tabu algorithm & Examples of Combinatorial optimization

Instructor: Odd Kolbjørnsen, oddkol@math.uio.no

Last time

- Iterative reweighted least squares (IRLS)
- Method of moments (constrained optimization)
- Alternating Direction Method of Multipliers (ADMM)
- Decision vs Optimization
- Solving vs checking
- P, NP, NP-Hard, NP-Complete
- Heuristics:
  - Algorithms that find a good local optima within tolerable time
  - Genetic algorithm population
  - Local search = Neighborhood
    - Greedy
    - Simulated annealing
    - Tabu algorithm (today)



Neighborhood

- We can't solve the full problem, i.e. not P
- We look for refinement of our current best guess
- Limiting the search to a **local neighborhood** $\mathcal{N}(\boldsymbol{\theta}^{(t)})$ at any iteration
  - Example model selection : (change only one component)

$$\mathcal{N}\left(\boldsymbol{\theta}^{(t)}\right) = \{\boldsymbol{\theta}\colon \exists\, l \text{ such that } \theta_j = \theta_j^{(t)} \text{ for } j \neq l\,\}$$

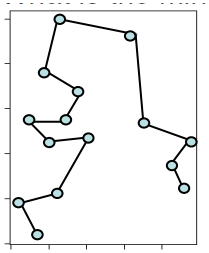
$$"\boldsymbol{\theta} = \boldsymbol{\gamma}"$$

$$Y_i = \beta_0 + \sum_{j=1}^{p} \gamma_j \beta_j x_{ij} + \varepsilon_i$$

  - Example traveling salesperson

$$\mathcal{N}\left(\boldsymbol{\theta}^{(t)}\right) = \{\boldsymbol{\theta}\colon \exists\, (k,l) \text{ such that } \theta_l = \theta_k^{(t)}, \theta_k = \theta_l^{(t)} \text{ and } \theta_j = \theta_j^{(t)} \text{ for } j \neq l,k\}$$

$$"\boldsymbol{\theta} = \boldsymbol{ordering}"$$



Tabu algorithms

- Greedy gets stuck in local minima
- Simulated annealing
  - Next move may reverse previous move
- Is there a way to escape local minima in a controlled way?

- Tabu idea:
  - Allow downhill move when no uphill move is possible (i.e. local minima)
  - Make some moves temporarily forbidden or tabu
    (to avoid returning to local minima in next step)
  - Early form: steepest ascent /mildest decent
    - Move to least unfavorable when there is no uphill move

Traveling salesperson Tabu

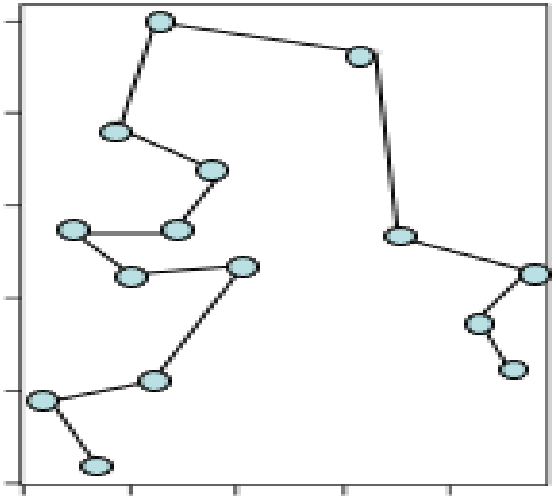
- Neighborhood: Swap the order of two components
- Move: To the best state in the neighborhood even if it is worse
- Tabu: Do not allow to pick two components that have been selected in the last k iterations
- Implementation:
  - Make a table of all possible pairs that can be picked, a $p(p-1) \times 2$ table
  - Make a list H containing the last k pairs that have been picked (references to the rows in the table above)
  - When searching within neighborhood, do not consider those pairs contained in H
  - When found the best pair, remove the first element of H and add the new pair to the end of H
- `Travel_salesman_tabu.R`

Tabu additional rules

- Aspiration criterion:
  - Allow a tabu move if it is better than the best found state so far
  - Allow a tabu move if it gives a large change
- Diversification
  - Penalize moves to a worse state if such a move has happened many times before
- Intensification
  - Reward moves that retain features that have shown to be important earlier
    - Variable selection: If inclusion of component j correspond to many good solutions, reward moves including this component

R-Examples

- Baseball model selection
  - Genetic
- Travelling salesperson
  - Greedy
  - Simulated annealing
  - Tabu



- Question: Why do you think I do not show Traveling salesperson example for the genetic algorithm?

Genetic algorithm baseball salaries

- Salaries for $n = 337$ baseball players
- $p = 27$ possible covariates, $2^{27} = 134\,217\,728$ possible models

Covariates are statistics collected during a season

- # runs scored
- batting average
- on pace percentage
- …

- Genetic algorithm (for model selection)
  - Starting with $P = 100$ models selected randomly
  - Choose two parents with probabilities proportional to $\exp(-AIC)$
  - For each component choose the state from one of the parents randomly
  - Allow mutation (change) with probability $\mu = 0.01$
  - `Baseball_genetic.R`

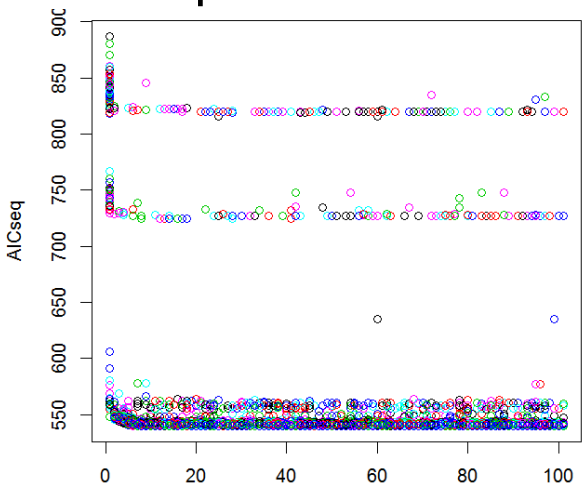
Genetic algorithm, Baseball $

- `Baseball_genetic.R`
- Maximize: $-AIC$ (model selection criteria)
- P=100 (population size)
- Chromosome length C=p=27
- Random initialization
- Select individuals with probability $\propto \exp(-AIC)$
- Mutation 1% (per locus and individual)
- 100 generations

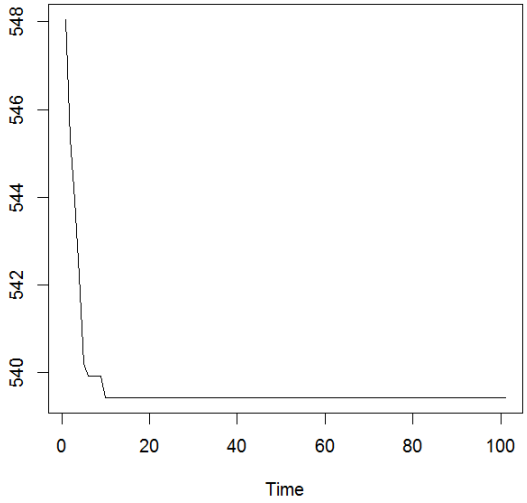
- Best achieved (first run)
  - $\theta^* =$ [2 3 6 8 10 13 14 15 16 24 25 26]
  - $f(\theta^*) = 539.4174$

Other seeds 539.4174, 541.7527

Population fit



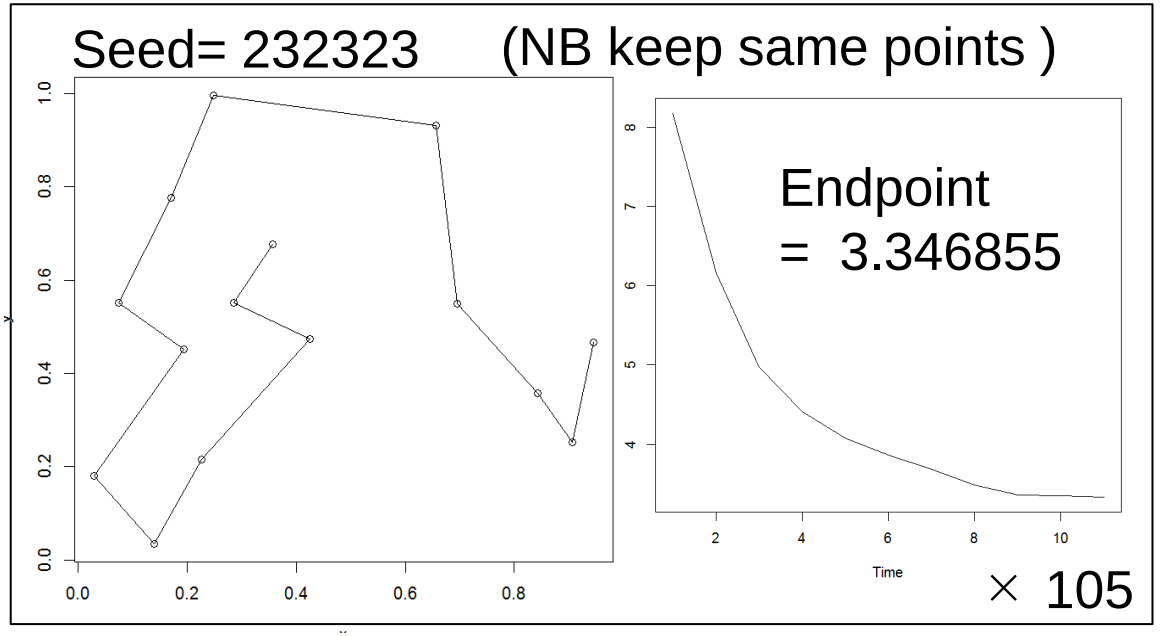
Best in generation

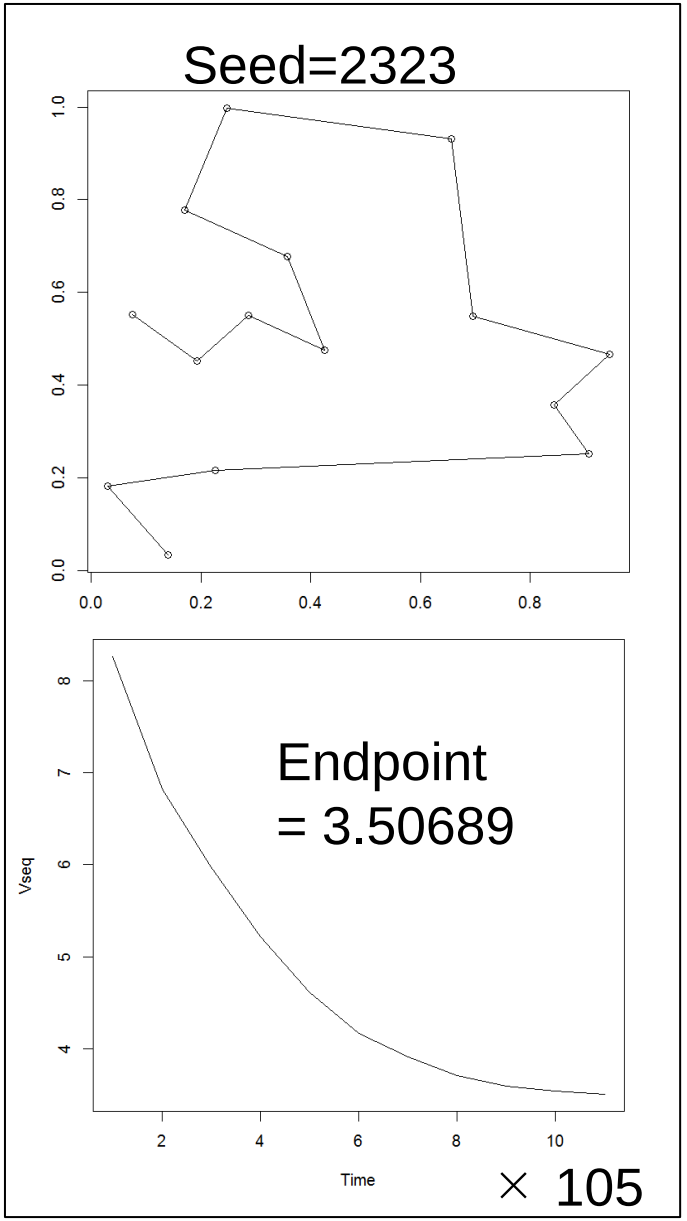


```
AICseq = AICfit
more = TRUE
Numit=100
pop.new = pop
AICfit.new = AICfit
mu = 0.01   #Probability for mutation
#Start iteration on updating populations
for(i in 1:Numit)
{
 for(k in 1:P)
  {
    #Selecting parents with probability proportional to exp(-AIC)
    phi1 = exp(-AICfit)
    phi2 = exp(-AICfit)

    #Selecting parents
    parent1 = sample(1:P,1,prob=phi1)
    parent2 = sample(1:P,1,prob=phi2)

    #Sampling independently which parent to inherit from
    bred = sample(1:2,p,replace=T)
    pop.new[k,bred==1] = pop[parent1,bred==1]
    pop.new[k,bred==2] = pop[parent2,bred==2]

    #Mutation
    ind2 = sample(0:1,p,replace=T,prob=c(1-mu,mu))
    if(sum(ind2)>0)
      pop.new[k,ind2==1] = 1-pop.new[k,ind2==1]

    #Extract only those components that are selected
    ind = c(1:p)[pop.new[k,]==1]
    base2 = baseball[,c(1,1+ind)]
    #Fit the new model
    AICfit.new[k] = AIC(lm(log(salary)~.,data=base2))
  }
  pop = pop.new
  AICfit = AICfit.new
  AICseq = rbind(AICseq,AICfit)
}
```

Greedy TS

- Random initialization
- Compare all pairs of swap ($\frac{p\cdot(p-1)}{2}$)
- Select the best
- Continue until no improvement





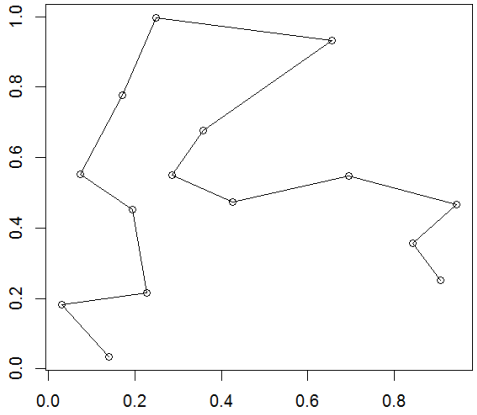
Greedy TS (Local search)

```r
#Simulate positions for n=15 cities
set.seed(2323)
p = 15
pos = data.frame(x=runif(p),y=runif(p)

par(mfrow=c(1,1))
plot(pos)
#dev.copy2pdf(file="../doc/example_tra

#par(mfrow=c(3,1))
plot(pos)

#Calculate pairwise distances between
d = as.matrix((dist(pos,diag=TRUE,uppe
#image(d)

#Convert to vector in order to access
d = as.vector(d)

#Random order of visits
theta = sample(1:p,p)
#Convert sequential pairs into index
ind = (theta[-p]-1)*p+theta[-1]  # loc
#Calculate total distance of order
v = sum(d[ind])
vseq = v
```

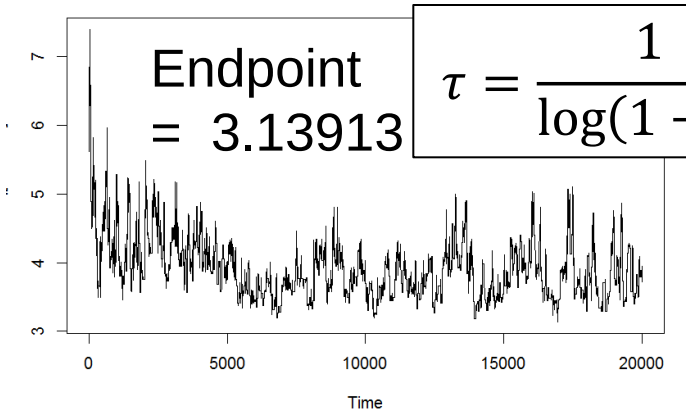
```r
#Perform neighbor search, changing best two components
more = TRUE
while(more)
{
  v2opt = v
  i1opt = NA
  # loop below determines the best pair to swap
  for(i1 in 1:(p-1))
    for(i2 in (i1+1):p)
    {
      theta2 = theta
      theta2[i1] = theta[i2]
      theta2[i2] = theta[i1]
      ind2 = (theta2[-p]-1)*p+theta2[-1]
      v2 = sum(d[ind2])
      if(v2<v2opt)
      {
        v2opt = v2
        i1opt = i1
        i2opt = i2
      }
    }
    more = FALSE
    if(v2opt<v) ## if the best swap is better than current optimum  update and continiue
    {
      theta2 = theta
      theta2[i1opt] = theta[i2opt]
      theta2[i2opt] = theta[i1opt]
      theta = theta2
      v = v2opt
      vseq = c(vseq,v)
      more = TRUE
  }
}
```

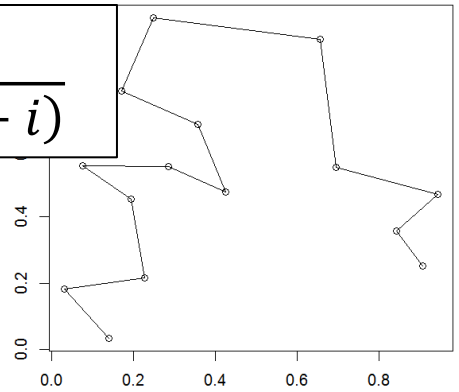
Simulated annealing TS

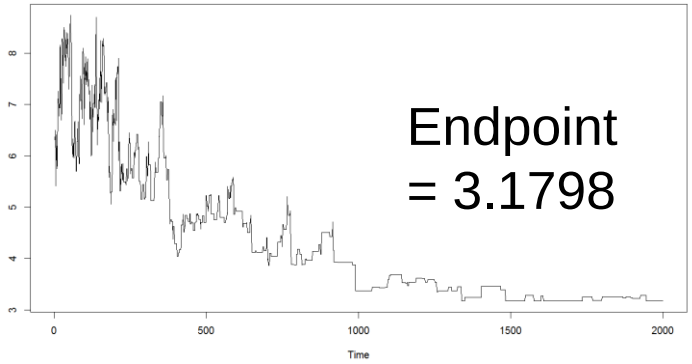
- Random initialization , evaluate $V^{(0)}$
- For each iteration draw a random pairs among ($\frac{p\cdot(p-1)}{2}$) possible
- Evaluate proposal gives value V^p
- Temperature $100/i$ or $\frac{1}{\log(1+i)}$, $m = 1$
- Accept if improvement or with probability $\exp((V^{(t)} - V^p)/\tau)$
- Iterate a fixed number of times (50000) NB
  do not loop all pairs in one update



$$\tau = \frac{100}{i}$$







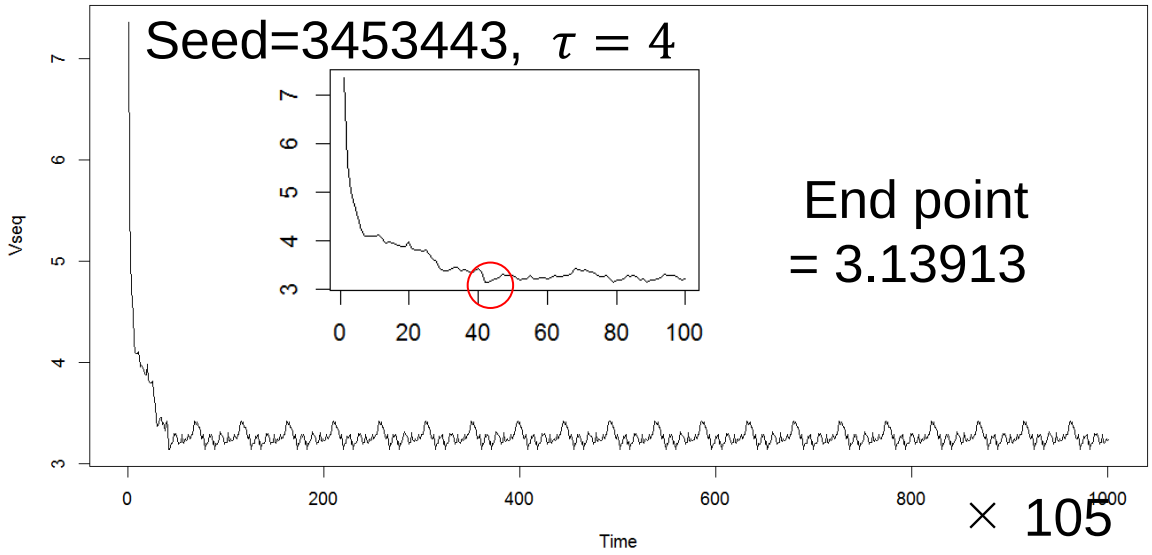
Simulated annealing TS

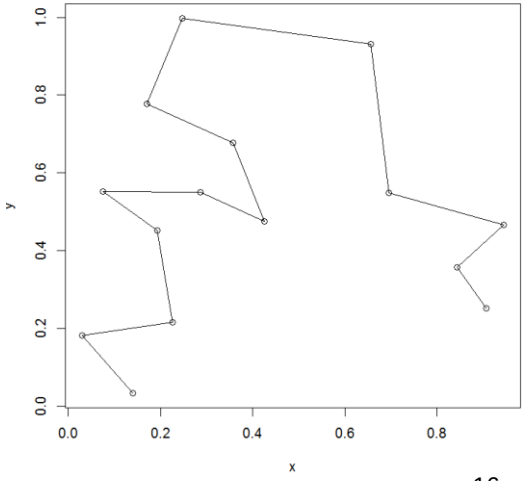
```
28  Numit= 20000
29  for(i in 1:Numit)
30  {
31    tau = 100/i
32    #tau = 1/log(i+1)
33    ind2 = sample(1:p,2,replace=F)
34    theta2 = theta
35    theta2[ind2[1]] = theta[ind2[2]]
36    theta2[ind2[2]] = theta[ind2[1]]
37    ind2 = (theta2[-p]-1)*p+theta2[-1]
38    V2 = sum(d[ind2])
39    prob = exp((V-V2)/tau)
40    u = runif(1)
41    if(u<prob)
42    {
43      theta = theta2
44      V = V2
45    }
46    Vseq = c(Vseq,V)
47  }
```

TABU TS

- Random initialization
- Compare all pairs of swap ($\frac{p\cdot(p-1)}{2}$), except the four last $\tau = 4$
- Build TABU list gradually to max size ($\tau = 4$).
- Remove FIFO when exceeding max size
- Select the best on the list
- Store the best so far
- Iterate a fixed number of times (1000)

Seed	tau	Value	First occur.
2323	4	3.5068	11
2323	10	3.1391	359
232323	4	3.2979	10
232323	10	3.1391	915
3453443	4	3.1391	42
3453443	10	3.1391	22





```
#Perform neighbor search, changing best two components
more = TRUE
tabu = NULL
H = NULL
tau = 10
#while(more)
for(it in 1:10000)
{
  V2opt = V+1000  #Just to get some initial value to beat
  i1opt = NA
  for(i in 1:num)
  {
    if(is.na(pmatch(i,H)))
    {
      #Find indices to swap
     i1 = searchtab[i,1]
     i2 = searchtab[i,2]
     #Swap components, put into theta2
     theta2 = theta
     theta2[i1] = theta[i2]
     theta2[i2] = theta[i1]
     #Calculate value for new configuration
     ind2 = (theta2[-p]-1)*p+theta2[-1]
     V2 = sum(d[ind2])
     #If best so far, store it
     if(V2<V2opt)
     {
       V2opt = V2
       iopt = i
       i1opt = i1
       i2opt = i2
     }
    }
  }
  #Change to best configuration found
  theta2 = theta
  theta2[i1opt] = theta[i2opt]
  theta2[i2opt] = theta[i1opt]
  theta = theta2
  V = V2opt
  Vseq = c(Vseq,V)
  #Include the swap in TABU table
  H = c(H,iopt)
  #If table is too large, remove first element (oldest swap)
  if(length(H)>tau)
    H = H[-1]
  #Check if better than best so far
  if(V < Vopt)
  {
    theta.opt = theta
    Vopt = V
  }
}
```