Decision Making and Optimization

Master in Data Analytics for Business

Lisbon School of Economics & Management
Universidade de Lisboa

2025-2026

Heuristics

Heuristics:

- Find good approximate solutions to difficult problems
- Advantage - usually get good solutions quickly
- Disadvantage - quality is usually unknown

There are several types of heuristics:

- Constructive
  - Greedy
- Improvement
  - Greedy
  - Local Search
- Matheuristics
  - Local Search
  - Branch & Cut
  - Branch & Price
- Metaheuristics
  - Tabu Search
  - Iterated local search, variable neighborhood search,
  - Genetic/Evolutionary Algorithm
  - Ant colony optimization
  - Particle swarm optimization.
  - Simulated annealing

Heuristics

Specific to each problem:

- Constructive/Greedy: the Knapsack, the Set Covering problem
- Constructive: the TSP
- Improvement: the TSP, the Location problem

General strategy

- Local Search
- Metaheuristics
- Matheuristics

but also tailored to each specific problem

Local Search Heuristic

Local Search Heuristic

Optimization problem in the solution space $\mathcal{S}$

$$\min_{x \in \mathcal{S}} z = F(x),$$

Define a **neighborhood** $\mathcal{N}(x_k) \subseteq \mathcal{S}$ of x_k

Local Search:

Initial step

Start in a random feasible point, $x_0 \in \mathcal{S}$

Iteration k

Move to x_{k+1} a better point in the **neighborhood** $\mathcal{N}(x_k)$ of x_k, $\mathcal{N}(x_k) \subseteq \mathcal{S}$

Stopping criteria

Example

Consider the following discrete function

$$F(x) = \begin{cases} 90, & x = 1, \\ 60, & x = 2, \\ 50, & x = 3, \\ 80, & x = 4, \\ 100, & x = 5, \\ 40, & x = 6, \\ 20, & x = 7, \qquad \text{global minimum at } x = 7 \\ 70, & x = 8 \end{cases}$$

and the problem

$$\min F(x), \;\; x \in S = \{1, 2, 3, 4, 5, 6, 7, 8\}$$

Example 1

neighborhood 1: $\mathcal{N}(x_k) = \{x_k - 1, x_k + 1\}$,

$$F(x) = \begin{cases} 90, & x = 1, \\ 60, & x = 2, \\ 50, & x = 3, \\ 80, & x = 4, \\ 100, & x = 5, \\ 40, & x = 6, \\ 20, & x = 7, \\ 70, & x = 8 \end{cases}$$

start at a random selected solution, for example, $x_0 = 1$

iteration	k	x_k	$N(x_k)$	$F(x_{k-1})$	$F(x_{k+1})$	x^*	$F(x^*)$	x_{k+1}
Start	0	1				1	90	1
	1	1	{-,2}	-	60	2	60	2
	2	2	{1,3}	90	50	3	50	3
End	3	3	{2,4}	60	80	3	50	**stop**

stop, there is no improvement
a local minimum was found: $x = 3$, $F(x) = 50$

Example 2

neighborhood 2: $\mathcal{N}(x_k) = \{1, \ldots, x_k - 1, x_k + 1, \ldots, 8\}$,

$$F(x) = \begin{cases} 90, & x = 1, \\ 60, & x = 2, \\ 50, & x = 3, \\ 80, & x = 4, \\ 100, & x = 5, \\ 40, & x = 6, \\ 20, & x = 7, \\ 70, & x = 8 \end{cases}$$

at each iteration randomly select an element of the neighborhood to be evaluated
r_k is a $(0, 1)$ random number, limiting to 5 iterations

iteration	k	$\mathbf{x_k}$	$\mathbf{F(x_k)}$	$\mathbf{N(x_k)}$	$\mathbf{r_k}$	$\mathbf{x'_k}$	$\mathbf{F(x'_k)}$	$\mathbf{(x^*, F(x^*))}$
Start	0	1	90					(1, 90)
	1	1	90	{2,3,4,5,6,7,8}	.4128	4	80	(4, 80)
	2	4	80	{1,2,3,5,6,7,8}	.2039	2	60	(2, 60)
	3	2	60	{1,3,4,5,6,7,8}	.0861	1	90	-
	4	2	60	{1,3,4,5,6,7,8}	.5839	6	40	(6, 40)
End	5	6	40	{1,2,3,4,5,7,8}	.5712	4	80	-

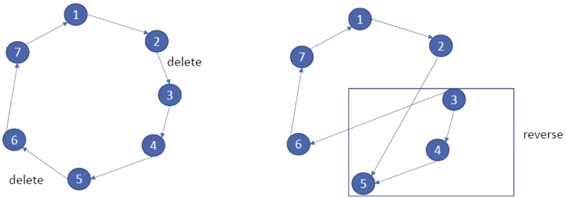
a local minimum was found: $x = 6$, $F(x) = 40$ at iteration 4

the size of the neighborhood is 7, $\frac{1}{7} = 0{,}145$

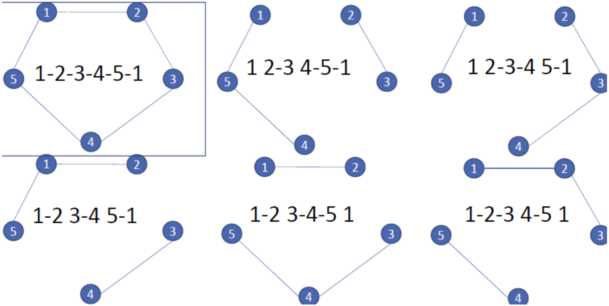
r_k	$\leq 0{,}14$	$\leq 0{,}29$	$\leq 0{,}435$	$\leq 0{,}58$	$\leq 0{,}725$	$\leq 0{,}87$	≤ 1
solution order	1	2	3	4	5	6	7

Example: the 2-opt neighborhood of a circuit

how to construct the 2-opt **neighborhood** of a circuit:
delete 2 non consecutive arcs, reverse arcs direction of a piece, include the new arcs to rebuild the circuit



Example: the 2-opt neigborhood of a circuit



used in the Nearest Neighbour algorithm for the TSP

Metaheuristics

- designed to avoid trapping at local optima
- allow inferior moves, in the hope that the added search flexibility will lead to a better solution

Stopping criteria:

- The number of search iterations exceeds a specified number.
- The number of iterations since the last best solution exceeds a specified number.
- The neighborhood associated with the current search point is either empty or cannot lead to a new viable search move.
- The quality of the current best solution is acceptable.

Tabu Search Heuristic

Tabu Search Heuristic

When the search is trapped at a local optimum, Tabu Search (TS) selects the next (possibly inferior) search move in a way that temporarily prohibits re-examining previous solutions.

The main tool for achieving this result is a tabu list, which "remembers" previous search moves and disallows them for a certain period of time. When a tabu move expires, it is removed from the tabu list and becomes available for future moves.

Let L be the tabu list and τ be the tabu period expressed in terms of the number of successive iterations

Example

Consider, again, the following discrete function

$$F(x) = \begin{cases} 90, & x = 1, \\ 60, & x = 2, \\ 50, & x = 3, \\ 80, & x = 4, \\ 100, & x = 5, \\ 40, & x = 6, \\ 20, & x = 7, \\ 70, & x = 8 \end{cases}$$

and the problem $\min F(x),\ x \in S = \{1, 2, 3, 4, 5, 6, 7, 8\}$

For the current solution x_k define:

. the tabu list L_k of examined solutions

. the **neighborhood** $\mathcal{N}(x_k) = \{x_{k-4}, \ldots, x_{k-1}, x_{k+1}, \ldots, x_{k+4}\} - L_k$

. the tabu period $\tau = 3$

. the stopping criteria: 5 iterations

Example

limit to 5 iterations,

$r_0 = 0.0935 \in [0, \frac{1}{8}] = [0, 0.125]$ thus take $x_0 = 1$,

$$F(x) = \begin{cases} 90, & x = 1, \\ 60, & x = 2, \\ 50, & x = 3, \\ 80, & x = 4, \\ 100, & x = 5, \\ 40, & x = 6, \\ 20, & x = 7, \\ 70, & x = 8 \end{cases}$$

iteration	k	r_k	x_k	$F(x_k)$	L_k	$N(x_k)$
						$\{\mathbf{1},2,3,4,5,6,7,8\}$
Start	0	.0935	1	90	-	$\{2, 3, 4, 5\}$
	1	.4128	3	50	$\{1\}$	$\{2, 4, 5, 6, 7\}$
	2	.2039	4	80	$\{1, 3\}$	$\{2, 5, 6, 7, 8\}$
	3	.0861	2	60	$\{1, 3, 4\}$	$\{5, 6\}$
	4	.5839	6	40	$\{3, 4, 2\}$	$\{5, 7, 8\}$
End	5	.5712	7	20	$\{4, 2, 6\}$	$\{3, 5, 8\}$

best solution found: $x = 7$, $F(x) = 20$,

if the size of the neighborhood is 5, $\frac{1}{5} = 0{,}145$

	$0 < r_k \leq 0{,}2$	$0{,}2 < r_k \leq 0{,}4$	$0{,}4 < r_k \leq 0{,}6$	$0{,}6 < r_k \leq 0{,}8$	$0{,}8 < r_k < 1$
solution order	1	2	3	4	5

Example of a Job Sequencing Problem

Jobco uses a single machine to process three jobs. For each job, both the processing time and the due date (in days) are given in the following table. As well the holding costs per day and the penalty costs per day. The due dates are measured from zero, the assumed start time of the first job.

Job j	Processing time T_j	Due date d_j	Holding cost h_j	Penalty cost p_j
1	10	15	3	10
2	8	20	2	22
3	6	10	5	10
4	7	30	4	8

The objective of the problem is to determine the job sequence that minimizes the total cost (holding and penalty cost).

Example of a Job Sequencing Problem

j	T_j	d_j	h_j	p_j
1	10	15	3	10
2	8	20	2	22
3	6	10	5	10
4	7	30	4	8

Consider the sequence (3-1-2-4)

Job	3	1	2	4
Processing time	6	10	8	7
Due date	10	15	20	30
Completion date	6	16	24	31
Holding time	4	0	0	0
Delay time	0	1	4	1
Holding cost	20	0	0	0
Delay cost	0	10	88	8

Total cost = 126

Example of a Job Sequencing Problem

Consider for the current sequence x_k at iteration k:

- z_k the total cost of the sequence
- the tabu list L_k of examined solutions
- the tabu period $\tau = 2$
- x^* the best solution available during the search
- z^* the total cost of x^*
- the **neighborhood** $\mathcal{N}(x_k)$
  exchange position of consecutive pairs of jobs
  $\mathcal{N}(1,2,3,4) = \{(2,1,3,4),\ (1,3,2,4),\ (1,2,4,3)\}$
- the stopping criteria: 5 iterations

Example of a Job Sequencing Problem

the size of the neighborhood is ≤ 3, $\frac{1}{3} = 0{,}33(3)$,

$0 < r_k \leq 0{,}33(3)$	$0{,}33(3) < r_k \leq 0{,}66(6)$	$0{,}66(6) < r_k < 1$
1	2	3

Iteration	k	**Sequence** s_k	**cost**	z^*	$L(s_k)$	$N(s_k)$	r_k
Start	0	(1-2-3-4)	167	167	-	(2-1-3-4) (1-3-2-4) (1-2-4-3)	.5124
	1	(1-3-2-4)	171		{3-2}	(3-1-2-4) (~~1-2-3-4~~) (1-3-4-2)	.3241
	2	(3-1-2-4)	126	126	{3-2,3-1}	(~~1-3-2-4~~) (3-2-1-4) (3-1-4-2)	.2953
	3	(3-2-1-4)	130		{3-1,2-1}	(2-3-1-4) (~~3-1-2-4~~) (3-2-4-1)	.4241
	4	(2-3-1-4)	162		{2-1,2-3}	(~~2-3-1-4~~) (2-1-3-4) (2-3-4-1)	.8912
End	5	(2-3-4-1)	260		{2-3,4-1}	(~~3-2-4-1~~) (2-4-3-1) (~~2-3-1-4~~)	.0992

best sequence: (3-1-2-4) with total cost $= 126$ (iteration 2)

the tabu list $L(s_k)$ contains the "out of order" sequences from the selected solution/sequence:

- for sequence (1-3-2-4) $\in N(s_0)$ it is $L(s_1) = \{3-2\}$ as 1 and 4 are in the same position as in sequence s_0=(1-2-3-4);
- for sequence (3-1-2-4) $\in N(s_1)$ we compare with s_1=(1-3-2-4) and $(3-1)$ enter the list as 2 and 4 have the same position in both sequences;
- for sequence (3-2-1-4) $\in N(s_2)$ it is $(2-1)$ as 3 and 4 are in the same position as in sequence s_2=(3-1-2-4);

Improvements / Fine-tuning

- **Aspiration Criterion.** The design of TS search disallows moves that are on the tabu list. An exception occurs when a disallowed move leads to an improved solution.

  For example the crossed-out tabu sequences in iterations 1, 2, 3, and 4 should be examined for the possibility of producing better search moves. If they do, they should be accepted as search moves.

- **Intensification** calls for a more thorough examination of nearby solution points
- **Diversification** attempts to move the search to unexplored solution regions.

One way to implement these strategies is to control the size of the tabu list.
A shorter tabu list increases the size of the allowed neighbourhood set, thereby intensifying the search for points close to the best solution.
A longer tabu list does the opposite, allowing escape from a local optimum by allowing exploration of "distant" regions.

Tabu Search

algorithm

- Step 0: Select a starting solution $x_0 \in S$. Initialize the tabu list $L_0 = \emptyset$, and choose a schedule for specifying the size of the tabu list. Set $k = 0$.
- Step 1: Determine the feasible neighborhood $\mathcal{N}(x_k)$ that excludes (inferior) members of the tabu list L_k.
- Step 2: Select the next move x_{k+1} from $\mathcal{N}(x_k)$ (or from L_k if it provides a better solution), and update the tabu list L_{k+1}.
- Step 3: If a termination condition is reached, stop. Otherwise, set $k = k + 1$ and go to Step 1.

Genetic Heuristic

Genetic Heuristic

J.H. Holland (1975), *Adaptation in Natural and Artificial Systems*, The University of Michigan Press.

Genetic or evolutionary algorithm is an optimization technique inspired by the process of natural selection in biology. It is used to find approximate solutions to complex problems where traditional methods may be ineffective.

Genetic or evolutionary algorithm is a metaheuristic that generates a set of solutions and in the course of the evolution process the solutions interact through the action of genetic operators such as selection, crossover, mutation and evolve towards "optimization".

Genetic algorithms are widely used in areas such as optimization, machine learning and artificial intelligence.

(it is not a local search algorithm)

Genetic Heuristic

In the context of optimization, the vocabulary of natural genetics is used with simplifications:

- **individual** - solution (feasible or not)
- **population** - subset of solutions
- **population size** - cardinality of the subset of solutions
- **generation** - iteration
- **time** - number of iterations
- **chromosome(s)** characterising the individual - mathematical structure(s) representing the solution
- **gene** - element of the representation of the solution
- **loci** - position of the element (position of the gene on the chromosome)
- **allele** - value of the element (value of the gene)
- **individual's fitness** - quality/value of the solution

In each generation, the selection operator and the genetic operator work to modify the population to produce individuals for the next generation.

Genetic Heuristic: algorithm

Initialization

- Determine the initial population P_1
- Evaluate P_1: calculate fitness for each individual x in P_1
- Let x_1^* be the individual (solution) with best fitness in P_1
- Set $k = 1$ (generation counter)

Iteration

- Select individuals from P_k with the selection operator
- Generate new individuals from those selected by the crossover operators
- Evaluate the fitness of the new individuals
- Optional: Perform elitism; Perform mutation
- Update the population, P_{k+1}
- If P_{k+1} contains and individual better than x_k^* then update x_{k+1}^*
- Check the stopping criterion

Genetic Heuristic: characteristics

Population: set of individuals/solutions

- updated each generation having
- fixed dimension or
- variable from generation to generation

Individuals: each solution x (feasible or not) is associated with an individual to which it corresponds, in general, a single chromosome encoded by a mathematical structure:

- Vector or matrix
- Permutation...

Fitness: the fitness of an individual x can be given by

- the objective function value $z = f(x)$
- a difference to a "target" value
- the fitness of an individual that corresponds to a non feasible solution may be penalized

Genetic Heuristic: genetic operators

Selection operator: operation that chooses the individuals on whom the crossover will act

- roulette
- tournament - wins the best of several randomly chosen

Crossover: operation that acts on two or more individuals - the parents - to produce 1, 2 or more offspring that should be better than the parents, but not always (example, simple crossover)

Mutation (optional): operation that acts on an individual, modifying it to create genetic diversity

Genetic Heuristic

Elitism (optional): to introduce into the population the best solution found so far to replace the worst of the current generation

Population update: the individuals produced can be introduced into the population continuously or only at the end of each generation - block replacement
offspring can always be kept or not

Stopping criteria:

- the diversity of individuals in the current generation is low
- the number of generations passed reaches a certain value
- the number of generations since the last update of x^* reaches a previously fixed value

Simple crossover example

Crossover is the mating of two individuals that combines the characteristics of both and passes this information on to the offspring. Thus, pairs of selected individuals are crossed with a certain probability.

suppose that the parents A and B are

$$A = [10011100] \text{ and } B = [10001111]$$

and fixing a crossing point equal to 5

$$A = [10011\ 100] \qquad B = [10001\ 111]$$

descendants C and D would be obtained as follow

$$C = [10011111] \qquad D = [10001100]$$

Simple mutation example

Mutation is the random change in the value of a gene in each of the offspring with a certain probability.

A simple mutation in a binary chromosome is equivalent to changing a 1 to a 0 or a 0 to a 1.
Given the chromosome

$$C = [10011111]$$

and assuming that the gene at position 4 has been selected for mutation, the resulting chromosome is

$$C = [10001111]$$

Genetic Heuristic

Key features:

- simple and easy coding
- adapted to the use of parallel processors
- high consumption of computational resources
- requires genetic coding and operators appropriate to each problem
- does not aim at obtaining local optima

Example of a Job Sequencing Problem

Job j	Processing time T_j	Due date d_j	Holding cost h_j	Penalty cost p_j
1	10	15	3	10
2	8	20	2	22
3	6	10	5	10
4	7	30	4	8
5	4	12	6	15

The objective of the problem is to determine the job sequence that minimizes the total cost (holding and penalty cost).

Genetic Heuristic

- individual: x job sequence
- generation: k iteration
- population P_k: set of job sequences at iteration k, dimension 4
- fitness z of individual x: total cost (holding + penalty)
- x^*: best job sequence available so far with total cost z^*
- selection operator: 2 parents: the best and a random
- crossover: at position 3 or 4 (keep first 2 or 3, resp., genes from the job sequence of one parent, complete with the sequence from the other parent)
- mutation: exchange jobs in two positions

Example of a Job Sequencing Problem

j	T_j	d_j	h_j	p_j
1	10	15	3	10
2	8	20	2	22
3	6	10	5	10
4	7	30	4	8
5	4	12	6	15

Consider the sequence (1-2-3-4)

Job	1	2	3	4	5
Processing time	10	8	6	7	4
Due date	15	20	10	30	12
Completion date	10	18	24	31	35
Holding time	5	2	0	0	0
Delay time	0	0	14	1	23
Holding cost	15	4	0	0	0
Delay cost	0	0	140	8	345

Total cost $=$ 512

Example of a Job Sequencing Problem

- initial iteration: $k = 0$
- Initial random population: $P_0 = \{I1, I2, I3, I4\}$ with size 4

Individual	Job Sequence x	Fitness = Cost z
I1	1-2-3-4-5	512
I2	2-3-4-1-5	605
I3	4-1-5-3-2	695
I4	3-2-1-4-5	475

- parents **selection**:
  the best $I4 = (3 - 2 - 1 - 4 - 5)$ and
  a random $I3 = (4 - 1 - 5 - 3 - 2)$

Example of a Job Sequencing Problem

- **crossover**: at position 3, keep the first 2 genes from the job sequence of one parent, complete with the order of the other parent
  - C1$= (3-2-x-x-x)$ from I4, complete with I3$-\{2,3\} = (4-1-5-3-2)-\{2,3\} = (4-1-5)$
  - C2$= (4-1-x-x-x)$ from I3, compete with I4$-\{1,4\} = (3-2-1-4-5)-\{1,4\} = (3-2-5)$

$$C1 = (3-2-4-1-5) \qquad C2 = (4-1-3-2-5)$$

- **mutation**: for $C1$ exchange positions 2 and 5; for $C2$ exchange positions 1 and 5

$$C1 = (3-2-4-1-5) \qquad C2 = (4-1-3-2-5)$$

will be

$$C1 = (3-5-4-1-2) \qquad C2 = (5-1-3-2-4)$$

Example of a Job Sequencing Problem

iteration $k = 0$

Individual	Sequence s	Cost z	
I1	1-2-3-4-5	512	Initial **random** population I1,I2,I3,I4
I2	2-3-4-1-5	605	Parents selection: the best I4 and I3 (random)
I3	4-1-5-3-2	695	Crossover I4 and I3 start at position 3
I4	3-2-1-4-5	475	
C1	3-2-4-1-5	573	Mutate C1 by exchanging positions 2 and 5
C2	4-1-3-2-5	829	Mutate C2 by exchanging positions 1 and 5
mC1	3-5-4-1-2	534	
mC2	5-1-3-2-4	367	

Example of a Job Sequencing Problem

iteration $k = 1$

Individual	Sequence s	Cost z	
I1	1-2-3-4-5	512	New population I1,I2=mC1,I3=mC2,I4
I2	3-5-4-1-2	534	Parents selection: the best I3 and I1 (random)
I3	5-1-3-2-4	367	Crossover I1 and I3 start at position 4
I4	3-2-1-4-5	475	
C1			Mutate C1 by exchanging positions 2 and 3
C2			Mutate C2 by exchanging positions 2 and 4
mC1			
mC2			

Example of a Job Sequencing Problem

iteration $k = 1$

Individual	Sequence s	Cost z	
I1	1-2-3-4-5	512	New population I1,I2=mC1,I3=mC2,I4
I2	3-5-4-1-2	534	Parents selection: the best I3 and I1 (random)
I3	5-1-3-2-4	367	Crossover I1 and I3 start at position 4
I4	3-2-1-4-5	475	
C1	5-1-3-2-4	367	Mutate C1 by exchanging positions 2 and 3
C2	1-2-3-5-4	439	Mutate C2 by exchanging positions 2 and 4
mC1	5-3-1-2-4	314	
mC2	1-5-3-2-4	361	

Example of a Job Sequencing Problem

iteration $k = 2$

Individual	Sequence s	Cost z	
I1	5-3-1-2-4	314	New population I1=mC1,I2=mC2,I3,I4
I2	1-5-3-2-4	361	Parents selection: the best I1 and I4 (random)
I3	5-1-3-2-4	367	Crossover I1 and I4 start at position 3
I4	3-2-1-4-5	475	
C1			Mutate C1 by exchanging positions 1 and 2
C2			No Mutation in C2
mC1			
mC2			

Example of a Job Sequencing Problem

iteration $k = 2$

Individual	Sequence s	Cost z	
I1	5-3-1-2-4	314	New population I1=mC1,I2=mC2,I3,I4
I2	1-5-3-2-4	361	Parents selection: the best I1 and I4 (random)
I3	5-1-3-2-4	367	Crossover I1 and I4 start at position 3
I4	3-2-1-4-5	475	
C1	3-2-5-1-4	292	Mutate C1 by exchanging positions 1 and 2
C2	5-3-2-1-4	222	No Mutation in C2
mC1	2-3-5-1-4	324	
mC2	5-3-2-1-4	222	

Example of a Job Sequencing Problem

iteration $k = 3$

Individual	Sequence s	Cost z	
I1	5-3-1-2-4	314	New population I1,I2,I3=mC1,I4=mC2
I2	1-5-3-2-4	361	Parents selection: the best I4 and I2 (random)
I3	2-3-5-1-4	324	Crossover: I2 and I4 start position 3
I4	5-3-2-1-4	222	
C1			No Mutation in C1
C2			No Mutation in C2
mC1			
mC2			

Example of a Job Sequencing Problem

iteration $k = 3$

Individual	Sequence s	Cost z	
I1	5-3-1-2-4	314	New population I1,I2,I3=mC1,I4=mC2
I2	1-5-3-2-4	361	Parents selection: the best I4 and I2 (random)
I3	2-3-5-1-4	324	Crossover: I2 and I4 start position 3
I4	5-3-2-1-4	222	
C1	5-3-1-2-4	314	No Mutation in C1
C2	1-5-3-2-4	361	No Mutation in C2

Algorithm

Step 0:

- Generate a random population P of feasible chromosomes.
- For each chromosome x in the selected population, evaluate its associated fitness. Record x^* as the best solution so far available.
- Encode each chromosome using binary or numeric representation

Step 1:

- Select two parent chromosomes from population P
- Crossover the parents' genes to create two children.
- Mutate the children' genes randomly.
- If resulting solutions are infeasible, repeat Step 1 until feasibility is achieved. Else , replace the weakest two parents with the new children to form a new population P and update x^*. Go to Step 2.

Step 2: If a termination condition is reached, stop; x^* is the best available solution. Otherwise, repeat Step 1.

Excel Solver - Evolutionary

Job Sequencing

	A	B	C	D	E	F	G
1							
2							
3		job	Tj	dj	hj $/day	pj $/day	
4		1	10	15	3	10	
5		2	8	20	2	22	
6		3	6	10	5	10	
7		4	7	30	4	8	
8		5	4	12	6	15	
9							
10	sequence	3	5	2	1	4	
11	Tj	6	4	8	10	7	
12	dj	10	12	20	15	30	
13	Completiton time	6	10	18	28	35	
14	Holding time	4	2	2	0	0	
15	hj	5	6	2	3	4	
16	Delay time	0	0	0	13	5	
17	pj	10	15	22	10	8	
18							
19		cost	206				

Excel Solver - Evolutionary

Job Sequencing

	A	B	C	D	E	F	G
1							
2							
3		job	Tj	dj	hj $/day	pj $/day	
4		1	10	15	3	10	
5		2	8	20	2	22	
6		3	6	10	5	10	
7		4	7	30	4	8	
8		5	4	12	6	15	
9							
10	sequence	3	5	2	1	4	
11	Tj	6	4	8	10	7	
12	dj	10	12	20	15	30	
13	Completiton time	6	10	18	28	35	
14	Holding time	4	2	2	0	0	
15	hj	5	6	2	3	4	
16	Delay time	0	0	0	13	5	
17	pj	10	15	22	10	8	
18							
19		cost	206				
20							

3

=SUMIF($B$4:$B$8;B10;$C$4:$C$8)

=SUMIF($B$4:$B$8;B10;$D$4:$D$8)

=B11

=MAX(0;B12-B13)

=SUMIF($B$4:$B$8;B10;$E$4:$E$8)

=MAX(0;B13-B12)

=SUMIF($B$4:$B$8;B10;$F$4:$F$8)

cost

Excel Solver - Evolutionary

Job Sequencing

	A	B	C	D	E	F	G
1							
2							
3		job	Tj	dj	hj $/day	pj $/day	
4		1	10	15	3	10	
5		2	8	20	2	22	
6		3	6	10	5	10	
7		4	7	30	4	8	
8		5	4	12	6	15	
9							
10	sequence	3	5	2	1	4	
11	Tj	6	4	8	10	7	
12	dj	10	12	20	15	30	
13	Completiton time	6	10	18	28	35	
14	Holding time	4	2	2	0	0	
15	hj	5	6	2	3	4	
16	Delay time	0	0	0	13	5	
17	pj	10	15	22	10	8	
18							
19		cost	206				

=SUMIF($B$4:$B$8;C10;$C$4:$C$8)
=SUMIF($B$4:$B$8;C10;$D$4:$D$8)
=B13+C11
=MAX(0;C12-C13)
=SUMIF($B$4:$B$8;C10;$E$4:$E$8)
=MAX(0;C13-C12)
=SUMIF($B$4:$B$8;C10;$F$4:$F$8)

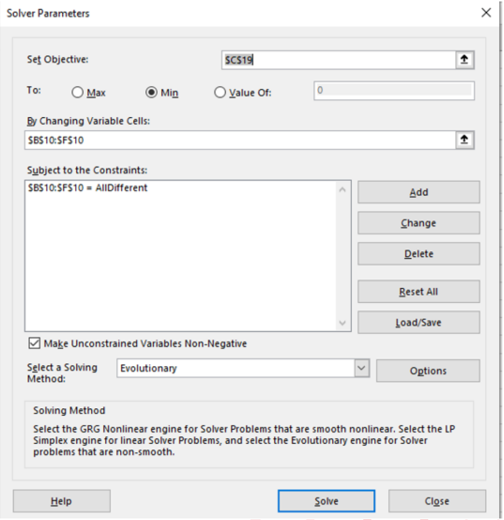
=SUMPRODUCT(B14:F14;B15:F15)+SUMPRODUCT(B16:F16;B17:F17)

Excel Solver - Evolutionary

Job Sequencing

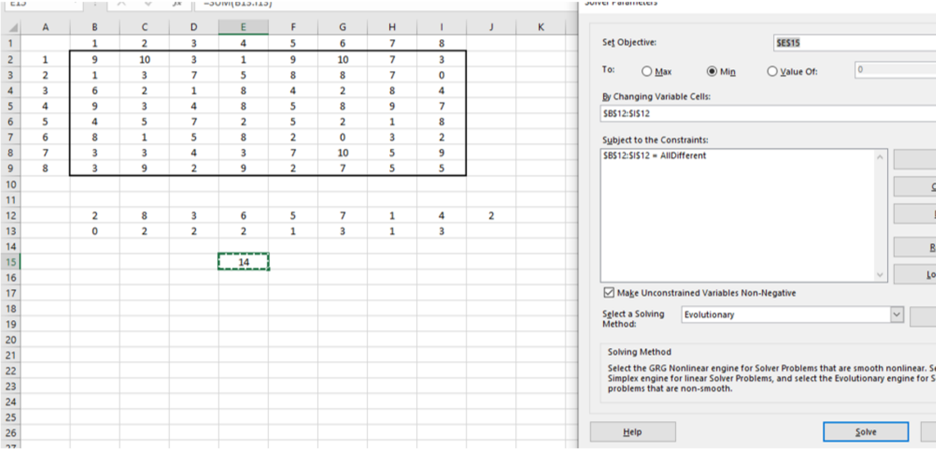
C19 | =SUMPRODUCT(B14:F14;B15:F15)+SU

	A	B	C	D	E	F
3		job	Tj	dj	hj $/day	pj $/day
4		1	10	15	3	10
5		2	8	20	2	22
6		3	6	10	5	10
7		4	7	30	4	8
8		5	4	12	6	15
9						
10	sequence	3	5	2	1	4
11	Tj	6	4	8	10	7
12	dj	10	12	20	15	30
13	Completiton time	6	10	18	28	35
14	Holding time	4	2	2	0	0
15	hj	5	6	2	3	4
16	Delay time	0	0	0	13	5
17	pj	10	15	22	10	8
18						
19		cost	206			



Excel Solver - Evolutionary

TSP



	A	B	C	D	E	F	G	H	I	J	K
1		1	2	3	4	5	6	7	8		
2	1	9	10	3	1	9	10	7	3		
3	2	1	3	7	5	8	8	7	0		
4	3	6	2	1	8	4	2	8	4		
5	4	9	3	4	8	5	8	9	7		
6	5	4	5	7	2	5	2	1	8		
7	6	8	1	5	8	2	0	3	2		
8	7	3	3	4	3	7	10	5	9		
9	8	3	9	2	9	2	7	5	5		
12		2	8	3	6	5	7	1	4	2	
13		0	2	2	2	1	3	1	3		
15					14						