A Hybrid Genetic Algorithm for the Periodic Vehicle Routing Problem with Time Windows

Michel Toulouse [1,2] **Teodor Gabriel Crainic[2]** **Phuong Nguyen[2]**

[1]Oklahoma State University

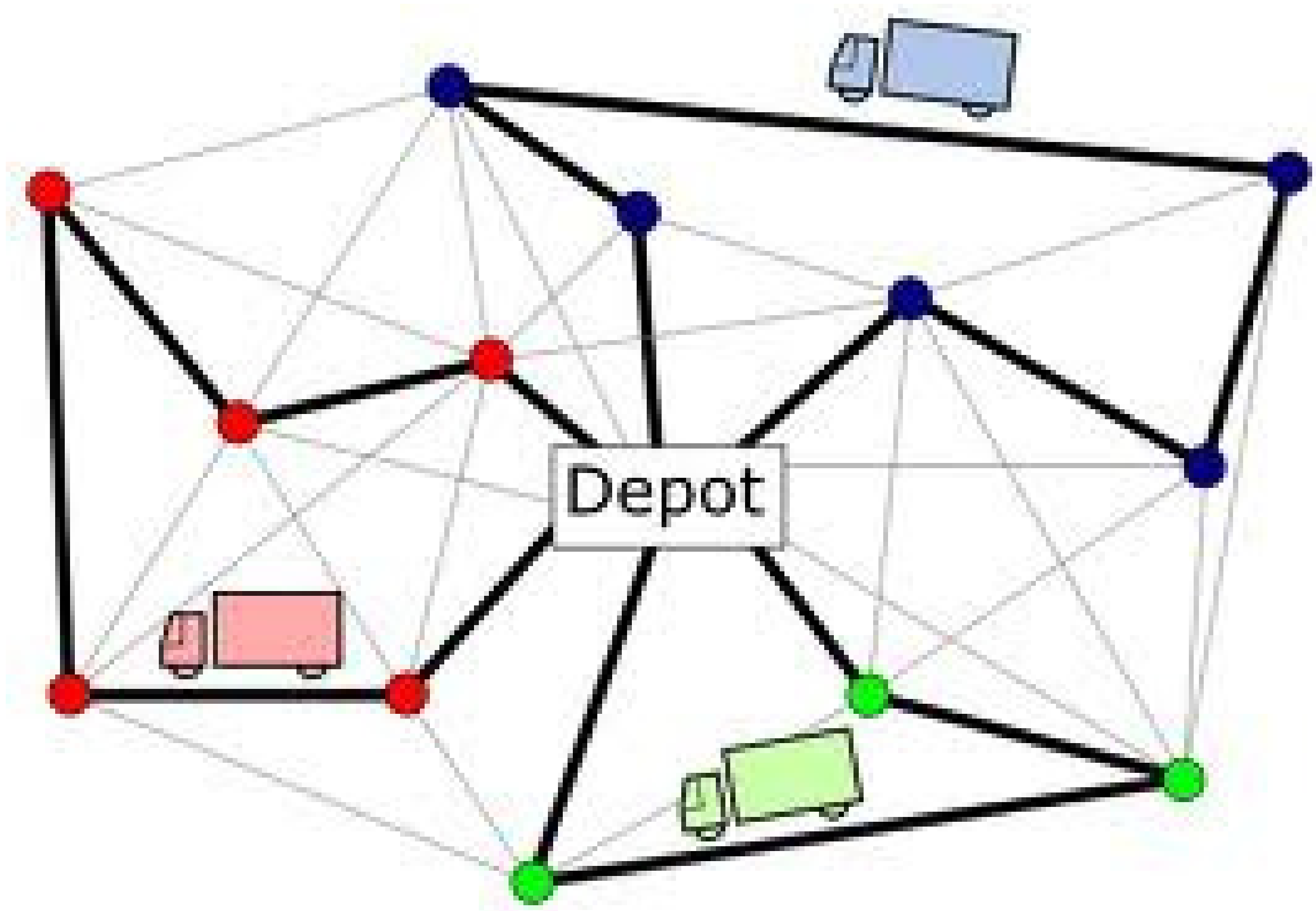
[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

Stillwater, November 2010

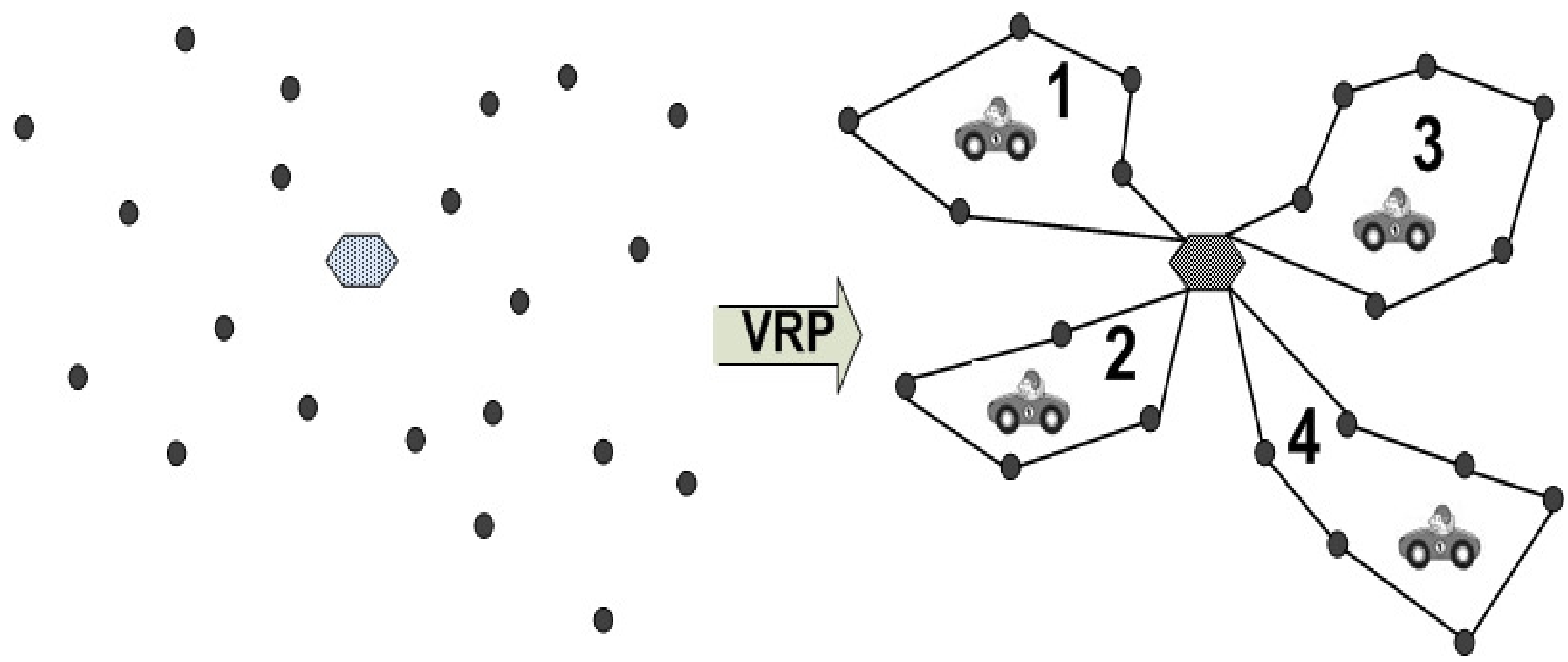
Presentation Outline

Introduction

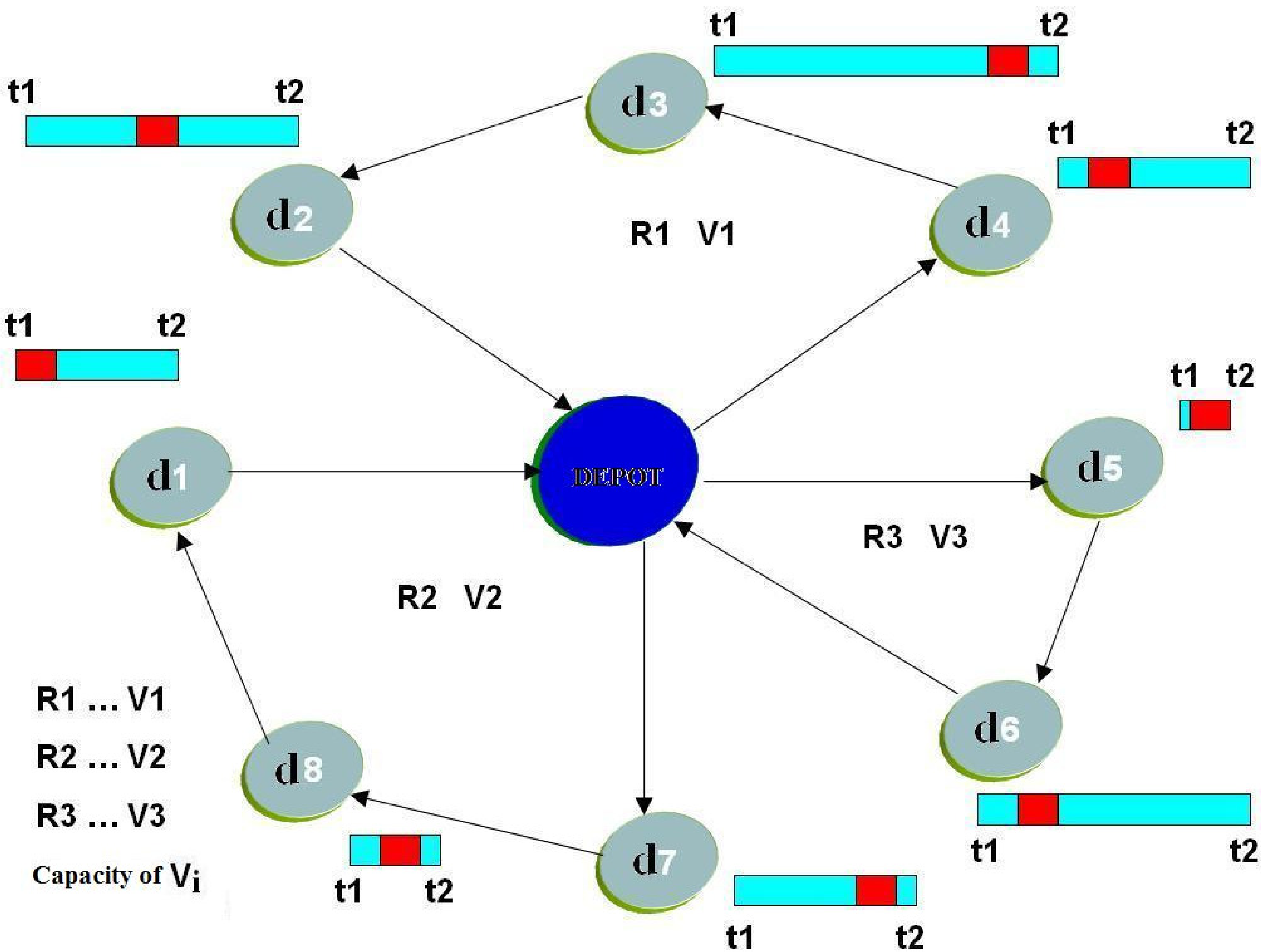
Introduction



Vehicle Routing Problem (VRP)



VRP with Time Windows (VRPTW)



Description of PVRPTW

- **The Periodic Vehicle Routing Problem with Time Windows (PVRPTW) is defined as having:**
  - A planning horizon of t days,
  - n customers having a demand $q_i > 0$, a service duration $d_i > 0$, a time window $[e_i, l_i]$, a service frequency f_i and a set R_i of allowable patterns of visit days,
  - a single depot with time window $[e_0, l_0]$, at which is based a fleet of m vehicles with limited on capacity and duration,
  - a cost (or travel time) $c_{ij} > 0$ between the locations.

Description of PVRPTW

- **The PVRPTW aims to select a single visit day pattern per customer and design at most *m* vehicle routes on each day of the planning horizon such that:**
  - each route starts and ends at the depot in the interval $[e_0, l_0]$,
  - each customer i belongs to exactly f_i routes over the horizon and is serviced in the interval $[e_i, l_i]$,
  - the total capacity and duration of route k do not exceed Q_k and D_k , respectively,
  - the total cost (or travel time) of all vehicles is minimized.

PVRPTW's mathematical formulation

$$\text{minimize} \quad \sum_{t \in T} \sum_{(i,j) \in E} \sum_{k \in K} c_{ij} x_{ijk}^{t}$$

subject to

$$\sum_{r \in R_i} y_{ir} = 1 \qquad \forall i \in V_C \qquad (1)$$

$$\sum_{j \in V} x_{ijk}^{t} = \sum_{j \in V} x_{jik}^{t} \qquad \forall i \in V, k \in K, t \in T \qquad (2)$$

$$\sum_{k \in K} \sum_{j \in V} x_{ijk}^{t} = \sum_{r \in R_i} y_{ir} a_{rt} \qquad \forall i \in V_C, t \in T \qquad (3)$$

$$\sum_{k \in K} \sum_{j \in V} x_{0jk}^{t} \le m \qquad \forall t \in T \qquad (4)$$

$$\sum_{i,j \in S} x_{ijk}^{t} \le |S| - 1 \qquad \forall S \subseteq V_C, k \in K, t \in T \qquad (5)$$

$$\sum_{j \in V_C} x_{0jk}^{t} \le 1 \qquad \forall k \in K, t \in T \qquad (6)$$

$$\sum_{i \in V_C} q_i \sum_{j \in V} x_{ijk}^{t} \le Q \qquad \forall k \in K, t \in T \qquad (7)$$

$$w_{jk}^{t} \ge w_{ik}^{t} + s_i + c_{ij} - M(1 - x_{ijk}^{t}) \qquad \forall (i,j) \in E, k \in K, t \in T \qquad (8)$$

$$e_i \le w_{ik}^{t} \le l_i \qquad \forall i \in V_C, k \in K, t \in T \qquad (9)$$

$$\sum_{i \in V_C} x_{i0k}^{t} (w_{ik}^{t} + s_i + c_{i0}) \le D \qquad \forall k \in K, t \in T \qquad (10)$$

Genetic Algorithms: Engineering view

Two horses and a groom (Han Gan [Tang Dynasty])

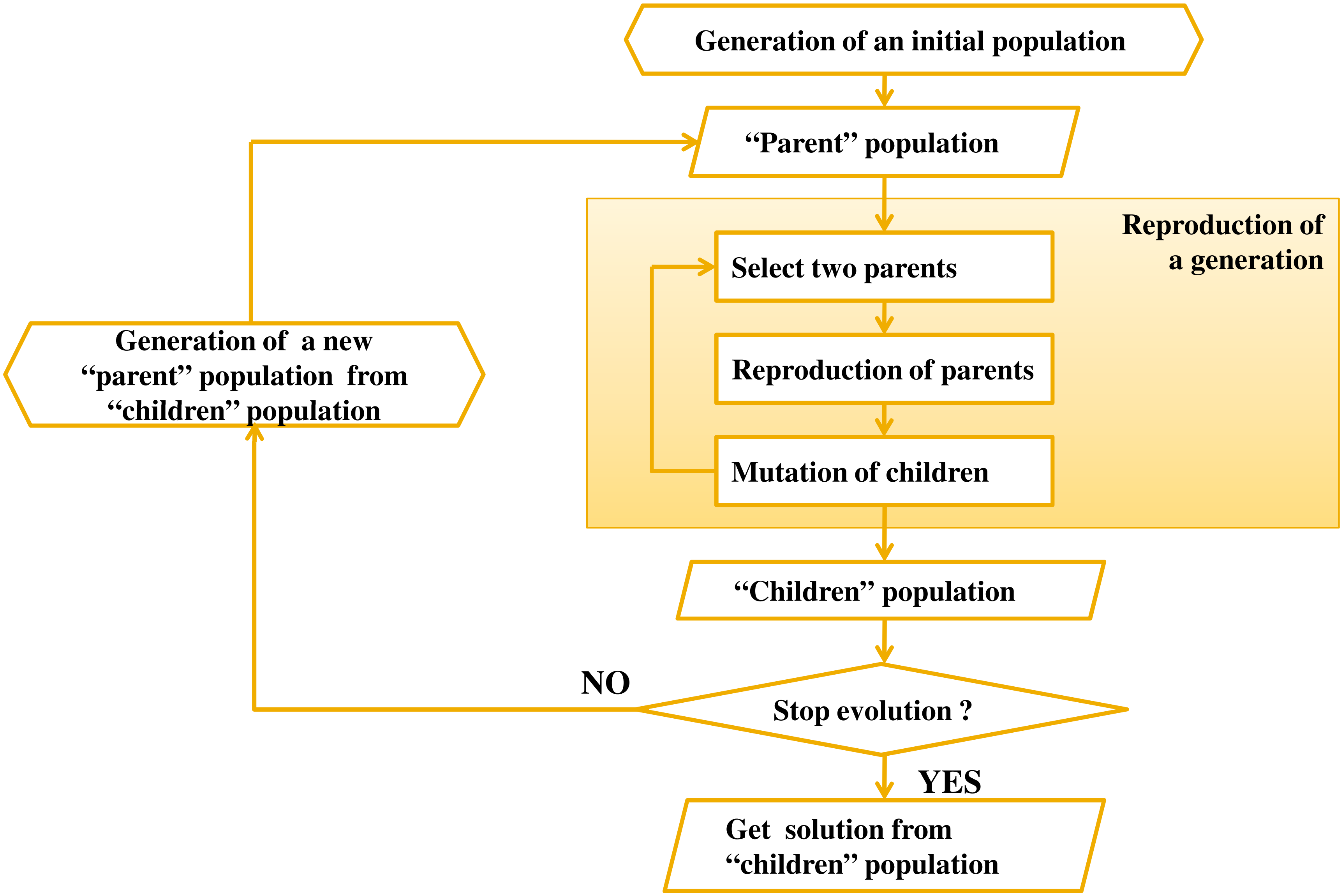
Genetic algorithms: Evolutionary view

- Species constantly have to adapt to changes in their environment.
- Fittest individuals of a specie live long enough to breed (natural selection).
- They pass their genetic adaptive features to their offspring.
- Through several generations, the specie get the upper hand on environment changes.

GA as an optimization method

- A stochastic search method.
- The environment is the cost function.
- A specie is a set of solutions
- Individuals are solutions, the cost of a solution measure it fitness. Furthermore, solutions are encoded under a suitable representation.
- Individuals (solutions) are selected to breed based on a random procedure biased by the fitness of the solutions.
- Breeding consists to brake solutions into components and to reassemble components to create offspring.

GA as optimization method

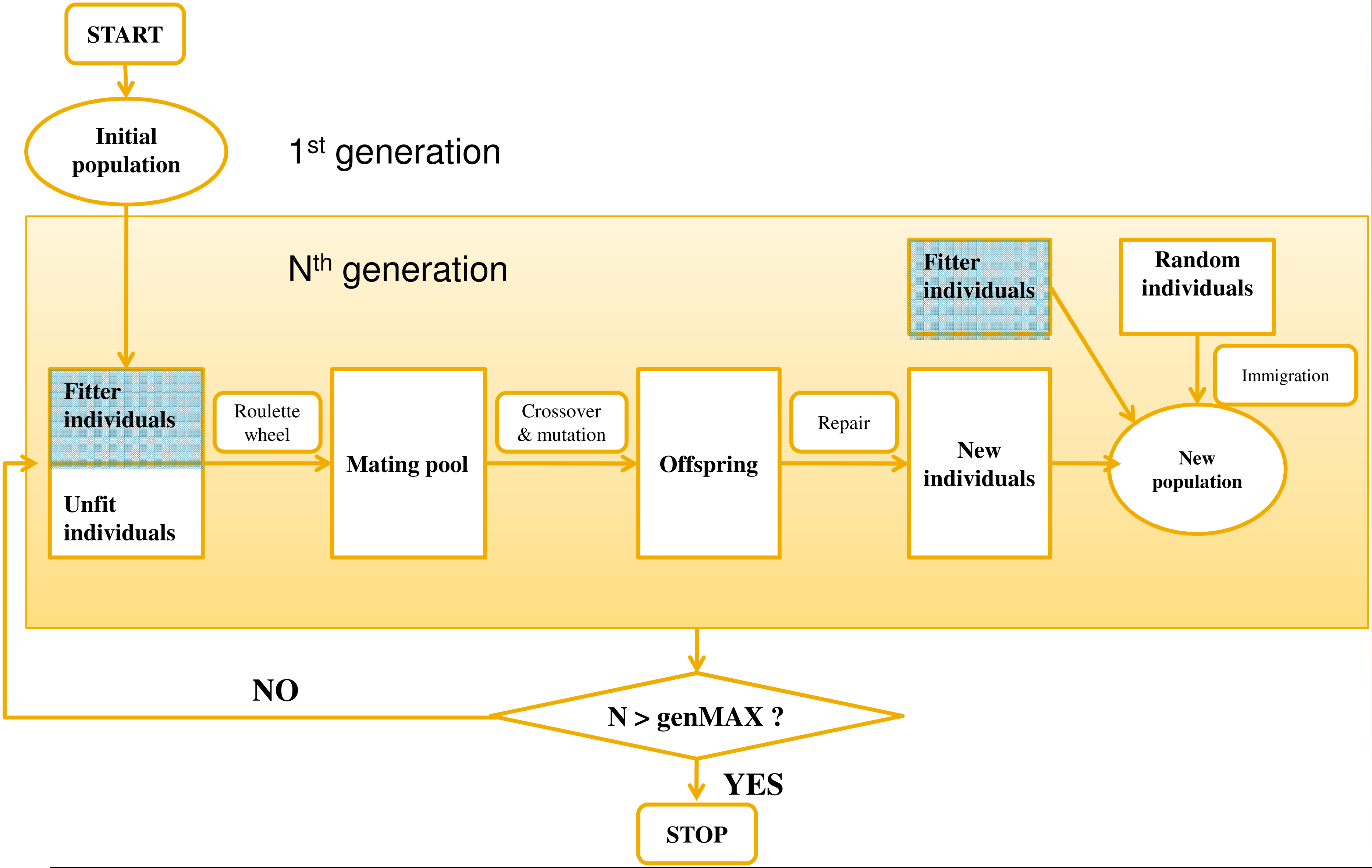


Pseudo code for generational GAs

Algorithm A simple generational genetic algorithm

$t = 0$
Generate the initial population of individuals $P(t)$
repeat
    $t = t + 1$
    Create a "parent population" $M(t)$ from $P(t-1)$
    Set "children" population $C(t) = \emptyset$
    while $C(t)$ is not full **do**
        Select 2 individuals P_1, P_2 from $M(t)$
        Apply crossover operator on P_1, P_2 to create children O
        Apply mutation operator on children O
        $C(t) = C(t) \cup O$
    end while
    Replacement: create a new population $P(t)$ from $P(t-1)$ and $C(t)$
until stop condition is satisfied
return the current best solution in $P(t)$

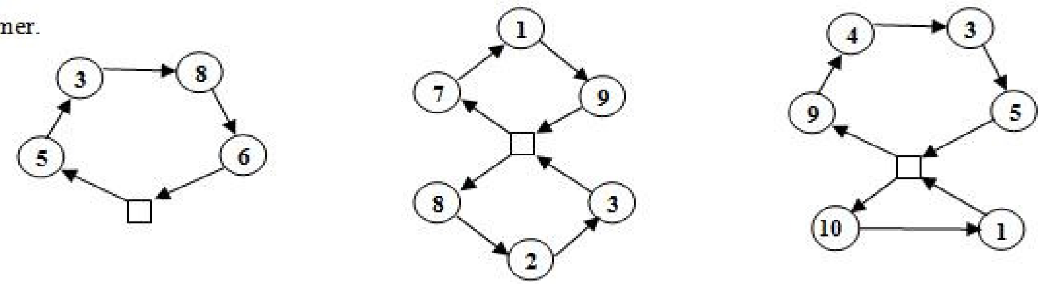
Overview of our GA for PVRPTW



Representation of individuals

UsedPattern	3	2	7	1	5	4	2	6	3	1
Combination in binary form	011	010	111	001	101	100	010	110	011	001

UsedPattern: Assign pattern of visit days to each customer.

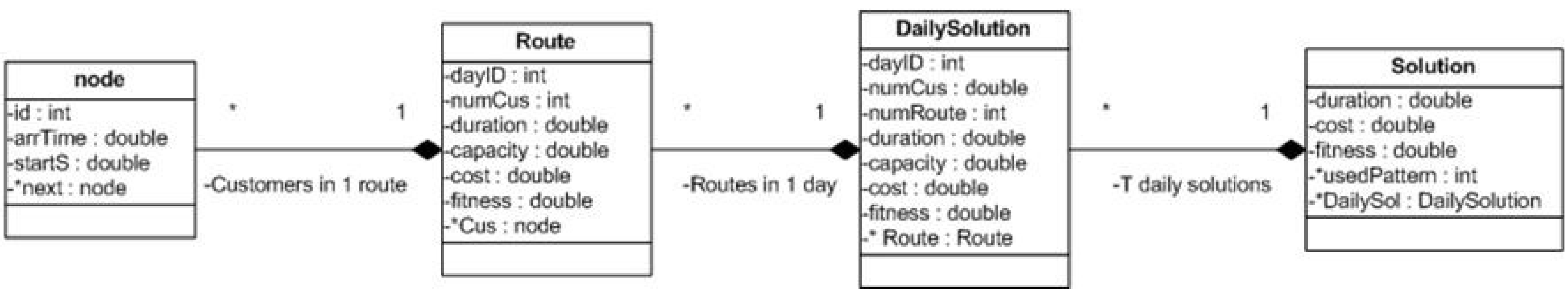


Day 1:
Route 1: 0, 5, 3, 8, 6, 0

Day 2:
Route 1: 0, 7, 1, 9, 0
Route 2: 0, 8, 2, 3, 0

Day 3:
Route 1: 0, 10, 1, 0
Route 2: 0, 9, 4, 3, 5, 0

Route: Represent routes of each daily solution.



The classes used in the proposed algorithm.

Individual fitness

The fitness function: $f(s) = c(s) + \alpha q(s) + \beta d(s) + \gamma w(s)$

where

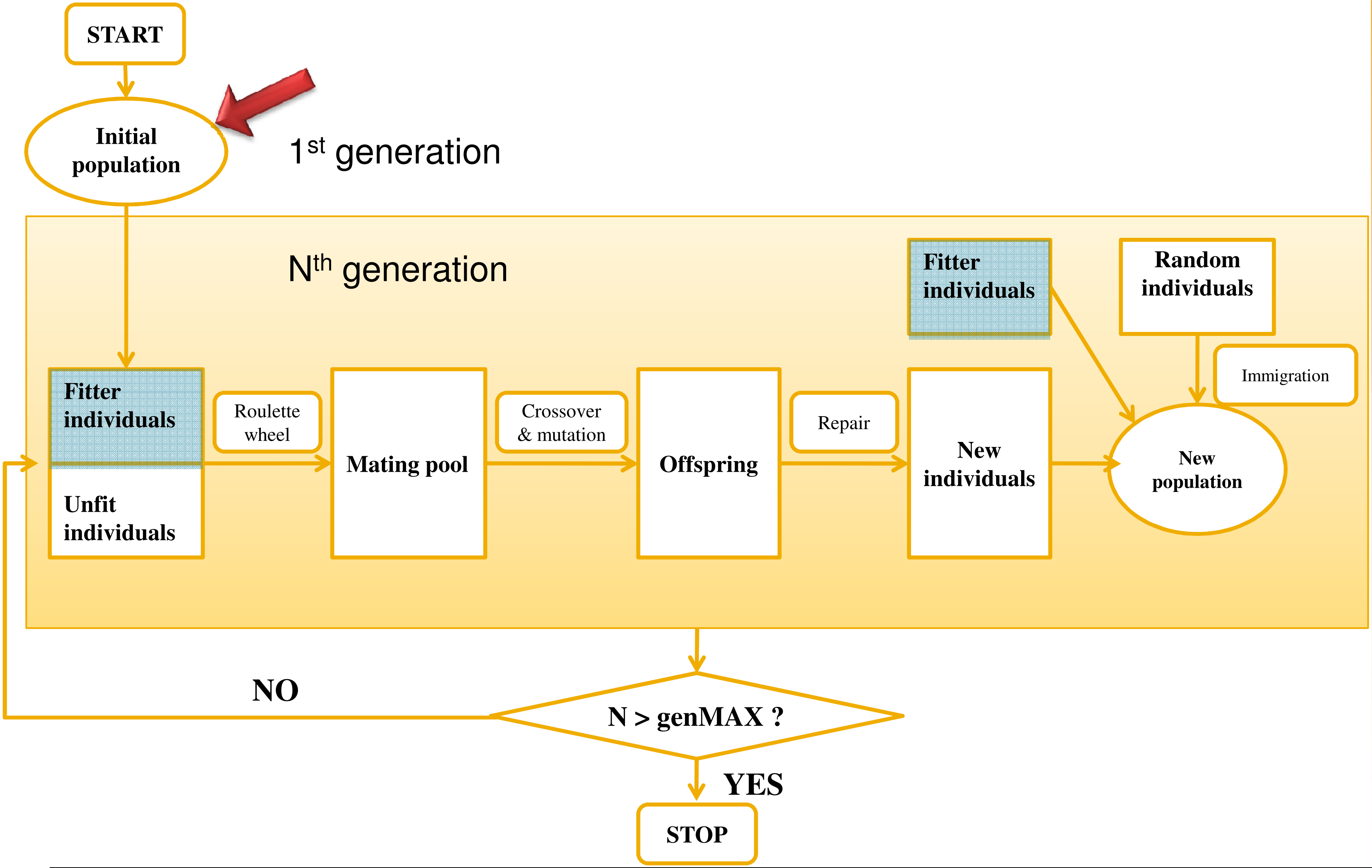
$q(s)$, $d(s)$, $w(s)$: the total violation of capacity, duration and time windows, respectively,

$$\alpha = h \frac{\bar{q}}{\bar{q}^2 + \bar{d}^2 + \bar{w}^2}, \quad \beta = h \frac{\bar{d}}{\bar{q}^2 + \bar{d}^2 + \bar{w}^2}, \quad \gamma = h \frac{\bar{w}}{\bar{q}^2 + \bar{d}^2 + \bar{w}^2}$$

$$h = \begin{cases} c(s_{worst}) & \text{if no feasible solution} \\ c(s_{bestfeasible}) & \text{otherwise} \end{cases}$$

$\bar{q}, \bar{d}, \bar{w}$ are the violation of capacity, duration and time windows respectively averaged over the current population.

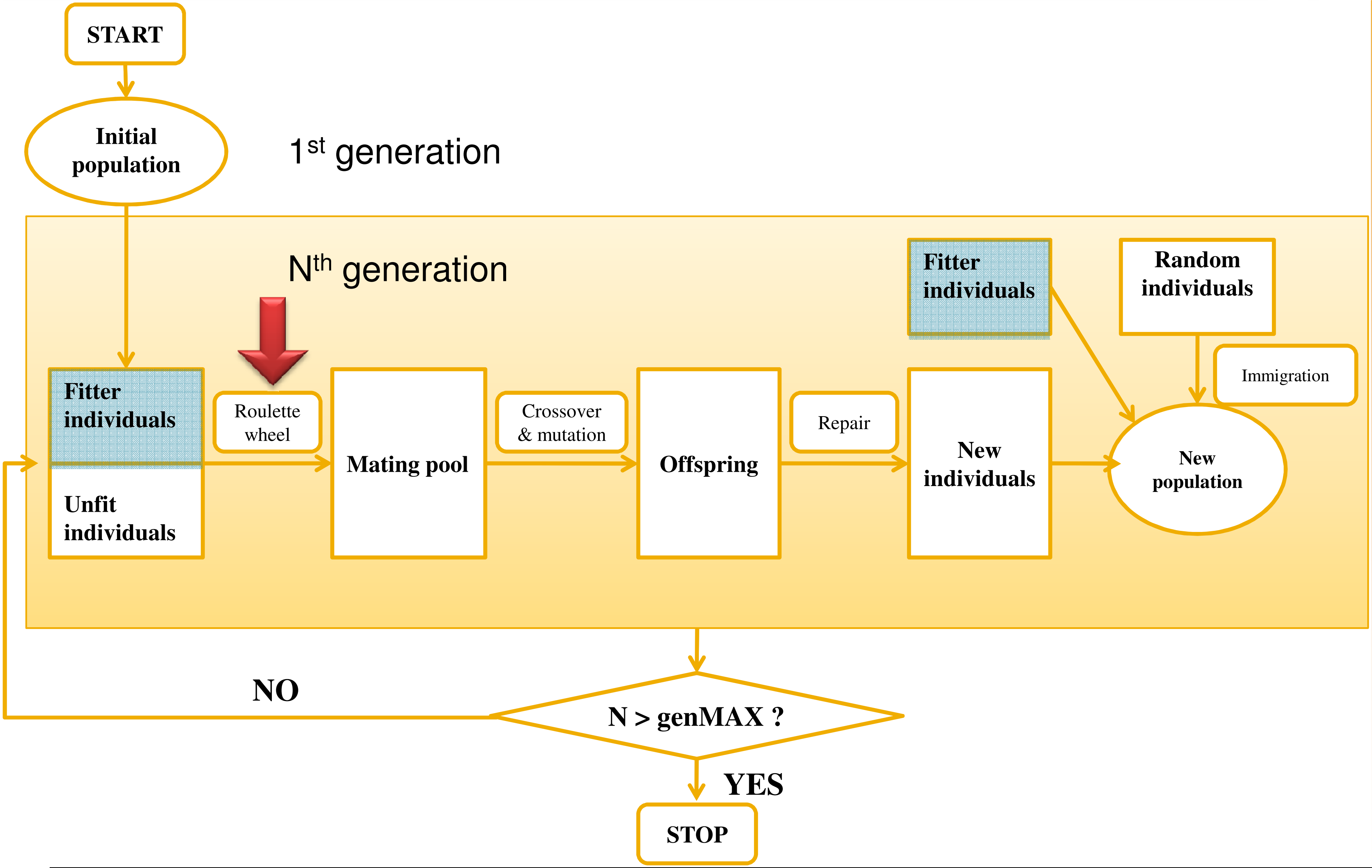
Algorithmic elements



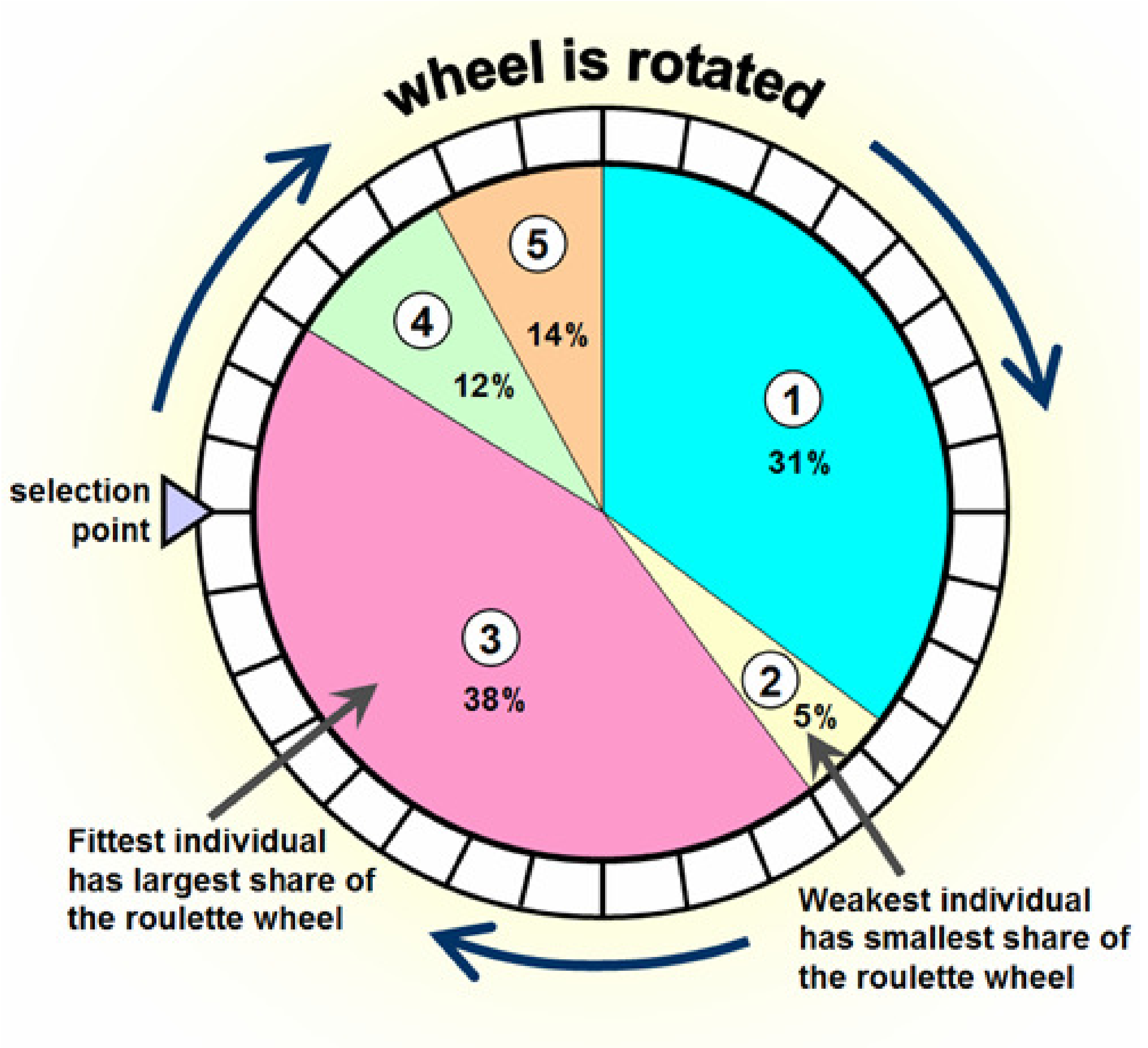
The initial population

- **Each customer is assigned a feasible pattern of visit days randomly.**
- **Solve VRPTW by applying:**
  1. Time-Oriented, Sweep Heuristic by Solomon.
  2. Parallel route building by Potvin and Rousseau.
  3. Our route construction method.

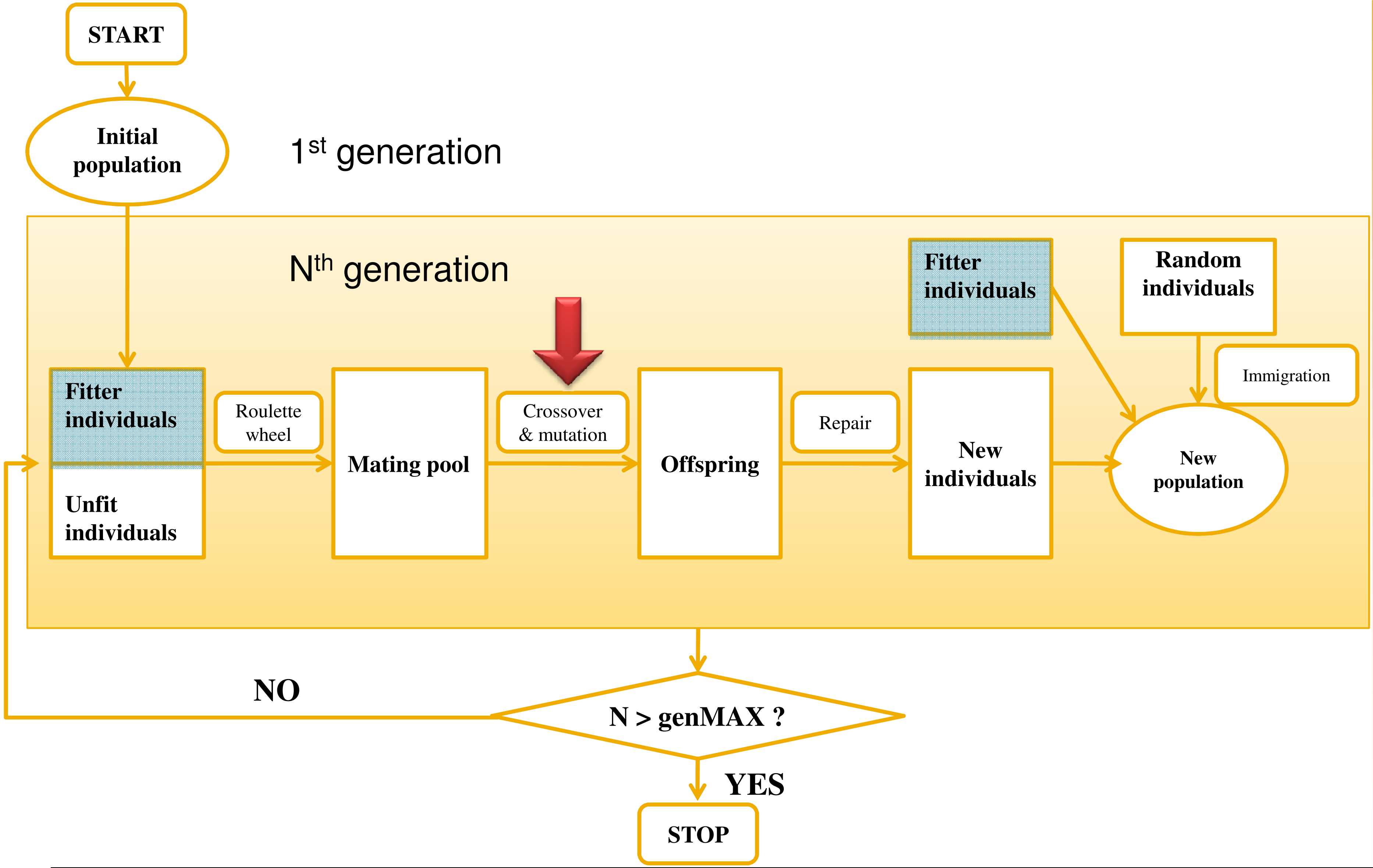
Algorithmic elements



Roulette Wheel selection operator



Algorithmic elements



The first crossover operator

P_1

3	2	7	1	5	4	2	6	3	1

011	010	111	001	101	100	010	110	011	001

5	3	8	6	7	1	9	8	2	3	10	1	9	4	3	5

P_2

3	4	7	4	6	4	1	6	5	2

011	100	011	100	110	100	001	110	101	010

9	8	2	1	8	3	10	5	7	1	3

Off_1

The first crossover operator

P_1

3	2	7	1	5	4	2	6	3	1

011	010	111	001	101	100	010	110	011	001

5	3	8	6	7	1	9	8	2	3	10	1	9	4	3	5

P_2

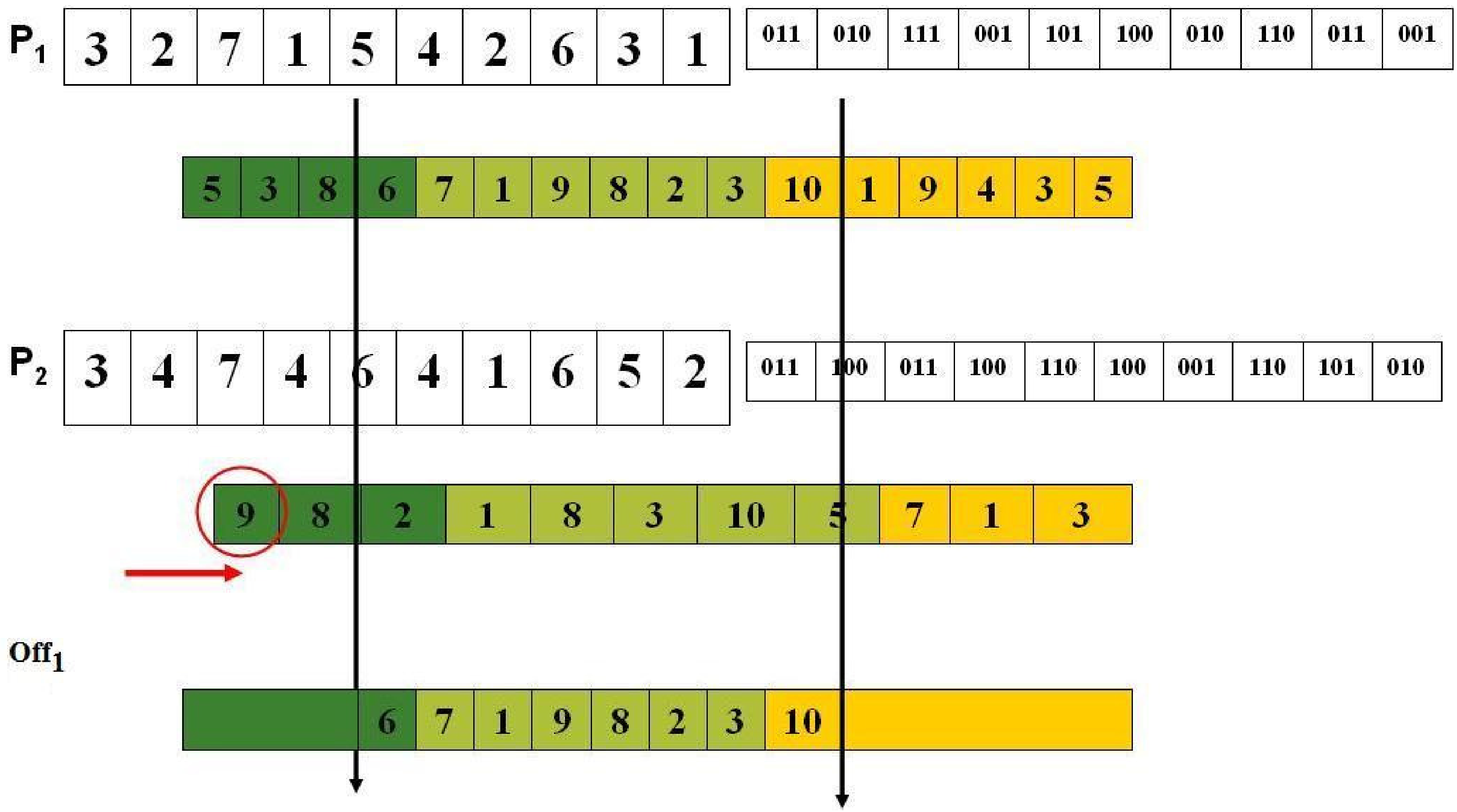
3	4	7	4	6	4	1	6	5	2

011	100	011	100	110	100	001	110	101	010

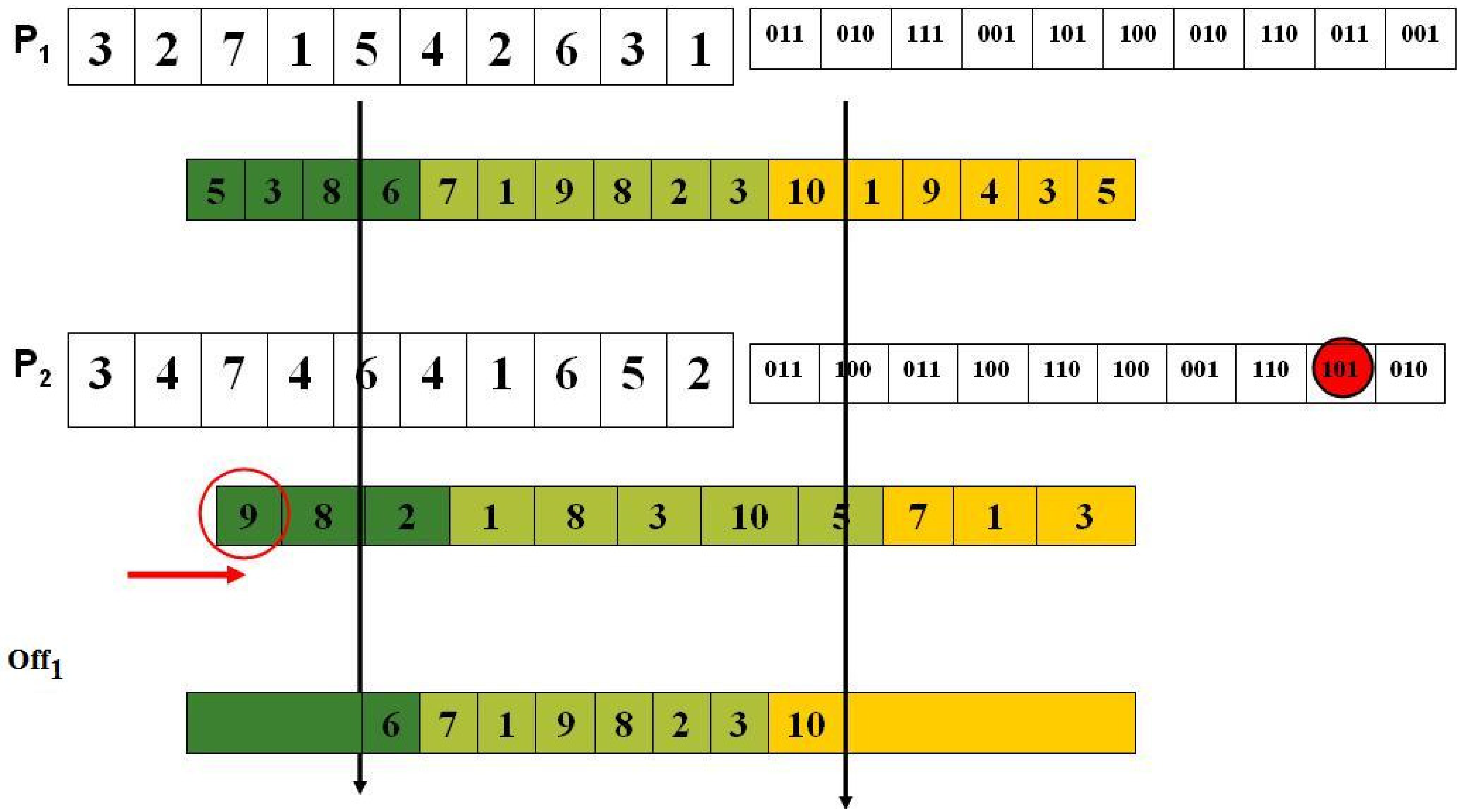
9	8	2	1	8	3	10	5	7	1	3

Off_1

The first crossover operator

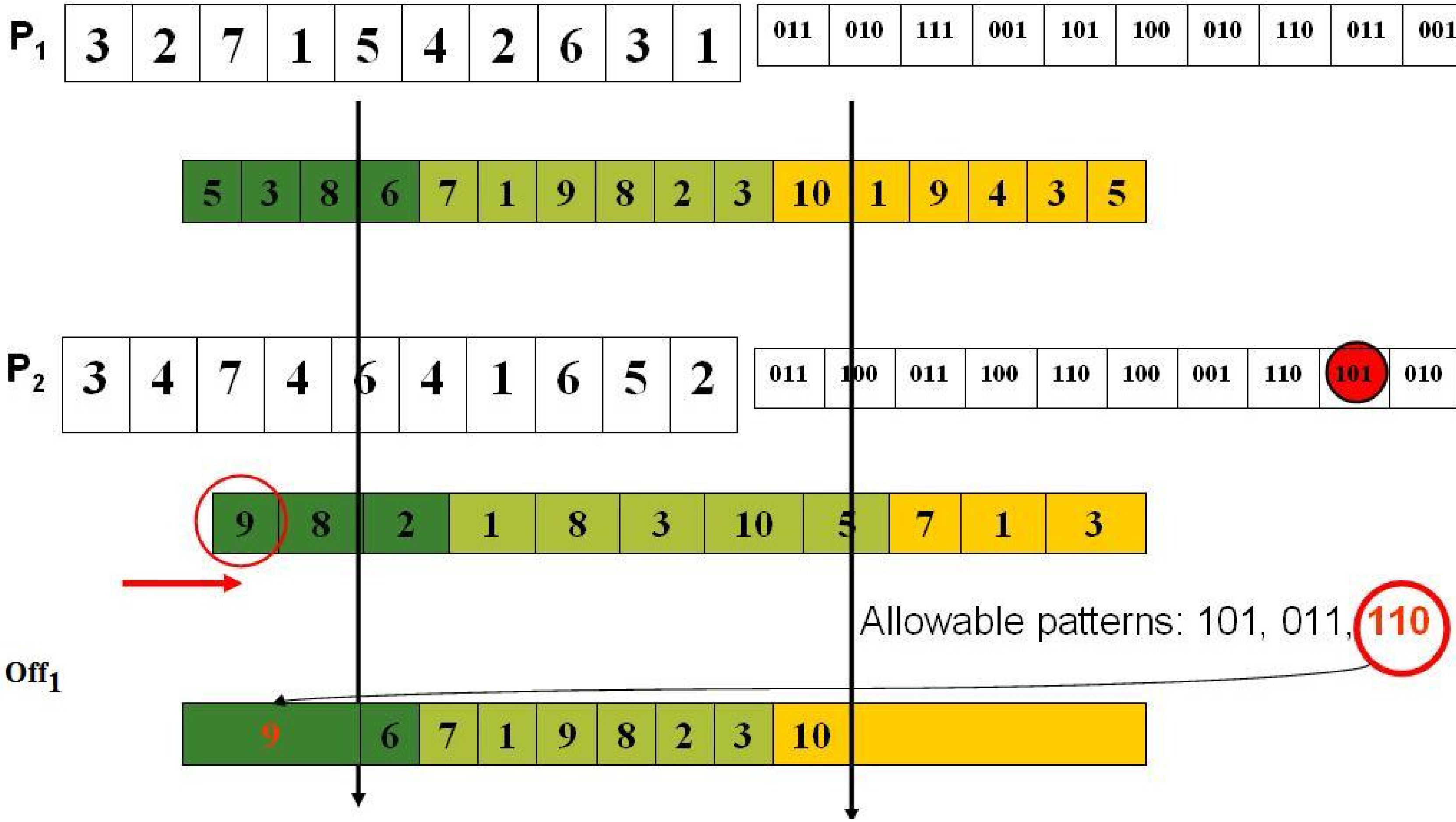


The first crossover operator

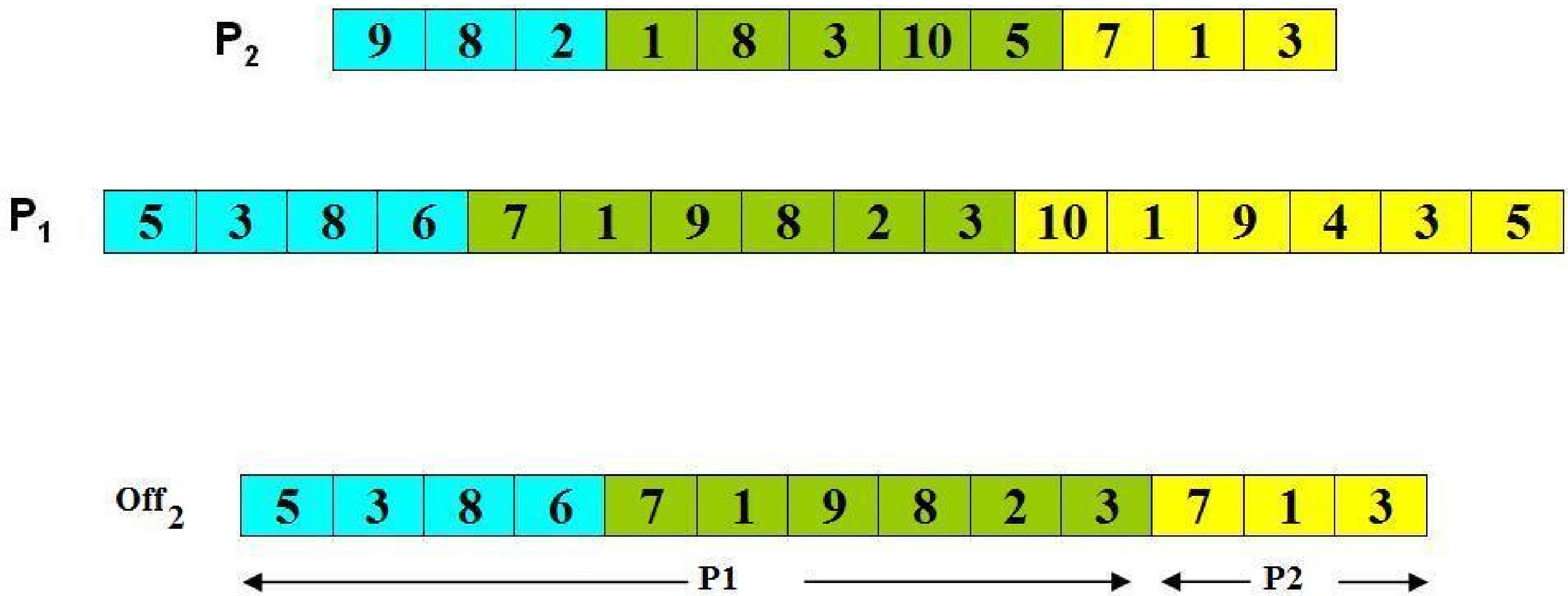


The first crossover operator

The first crossover operator

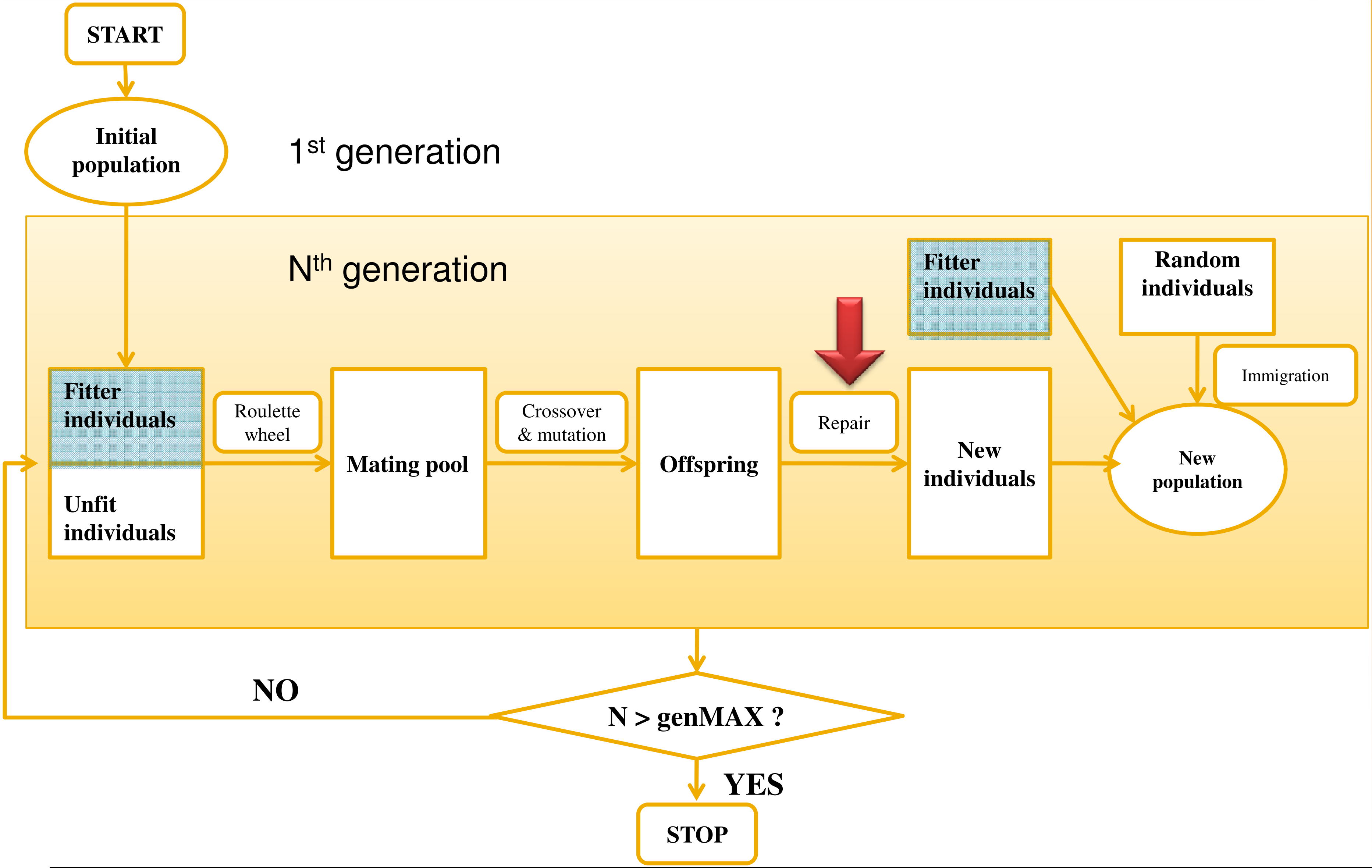


The second crossover operator



- For each day t, one parent among $\{P_1, P_2\}$ is selected randomly, from which all routes in day t are copied into the offspring Off_2.
- Remove/insert customers from/into days such that the pattern of visit days of all customers are satisfied.

Algorithmic elements



At issues

- GA not well adapted to constrained optimization, crossover create infeasible solutions.
- PVRPTW is a heavily constrained optimization problem.
- Usually repair strategies aim at regaining feasibility, but for PVRPTW this often leads to very poor solutions.
- Not only solutions are poor, but also their genetic make-up (building blocks hypothesis).
- Need to repair not only feasibility but also the building block features of the population.
- Make use of metaheuristics.

Repair strategies

Phase 1: Simultaneously tackle routing and pattern improvements

- **Unified Tabu** of Cordeau et al.
- **Random VNS** of Pirkwieser and Raidl: order of neighborhood structures are chosen randomly
- **Pattern improvement**: explore all feasible patterns of all customers where each customer is reassigned to new pattern, one by one

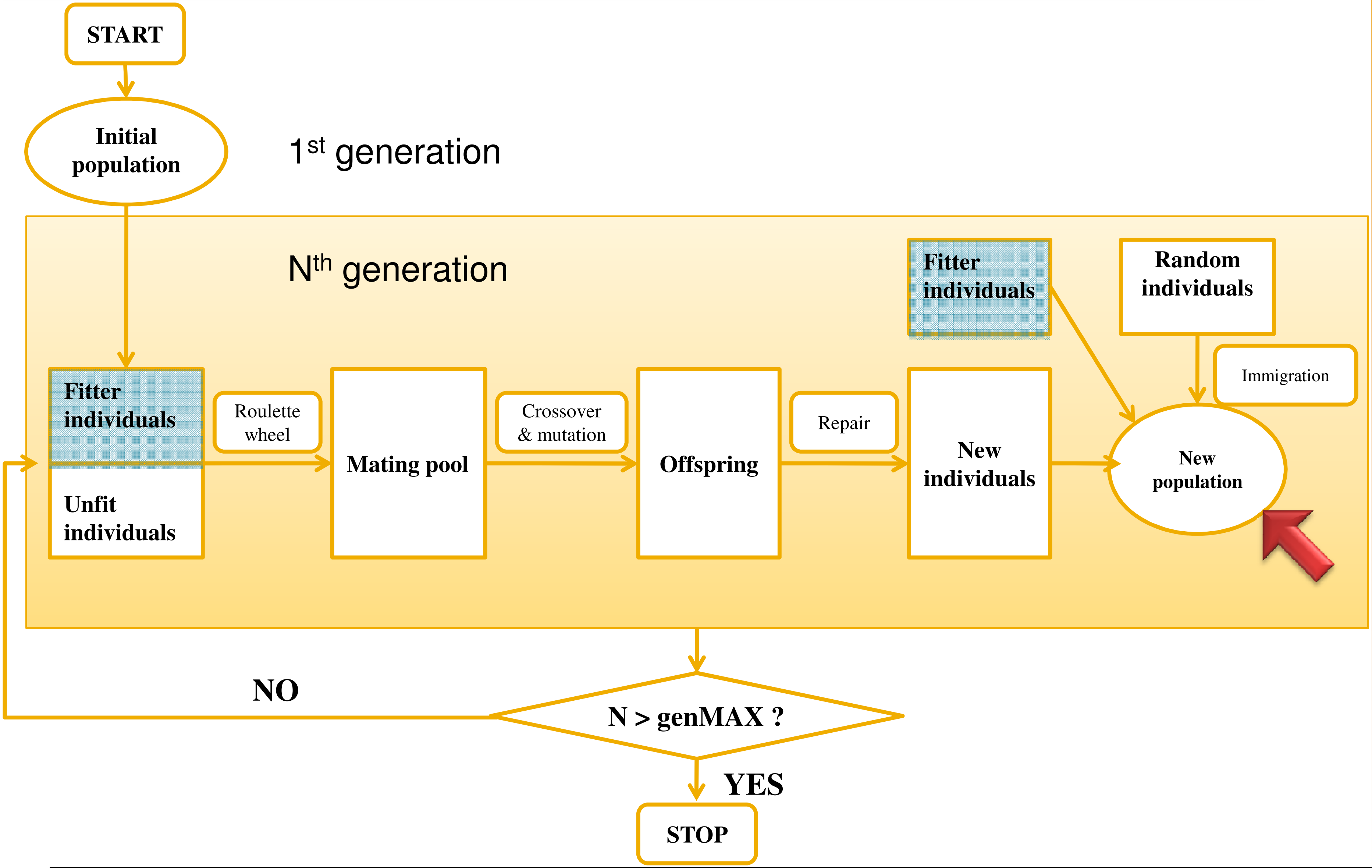
Phase 2: Routing improvements

- locally re-optimize the routes in which hybridized neighborhood structures with a set different route improvement techniques are used

Algorithm 1 RepairProcess(solution s, $curGEN$)
Note: $curGEN$ is the current number of generations

```
{The first phase:}
if (curGEN % 2) then
    UTS(s)
else
    RVNS(s)
end if
Pattern_improvement(s)
{The second phase:}
Route_improvement(s)
return s
```

Algorithmic elements



Replacement

- *nPop* parents are selected from the current population using Roulette wheel to build the mating pool and *nPop* offspring are then created.
- The next generation is composed of the *nKeep* best individuals among the pool of chromosomes in the current population (*nKeep* < *nPop*) and *nPop* new offspring.

Previous works

20 instances generated by Cordeau et al.

1. Unified Tabu Search of Cordeau et al. (2001)
   - (1) move a customer, (2) change pattern of a customer,
   - accept infeasible solutions.
2. Variable Neighborhood Search of Pirkwieser and Raidl (2008)
   - change pattern of a customer, (2) move a segment, (3) exchange segments,
   - accept worse solutions.

45 instances generated by Pirkwieser and Raidl

1. Hybrid scheme between VNS and ILP-based column generation approach (2009)
2. Multiple cooperating VNS (2010)
3. Hybrid scheme between multiple-VNS and ILP-based column generation approach (2010)

Instances

Benchmark	#instances	#customers	#vehicles	Planning period
Cordeau et al.	**20**	**[48, 288]**	**[3, 20]**	**4 or 6 days**
S.Pirkwieser & Raidl	**45**	**100**	**[7, 14]**	**4, 6 or 8 days**

Experiment parameters

Parameters	Setting	Final values	
		Small instance	Large instance
Population size (*nPop*)	[50, 400]	100	350
Number of elite (*nKeep*)	[25, 300]	60	200
Number of iterations applying UTB	[20, 120]	60	100
Number of iterations applying VNS	[100, 800]	100	200

Numerical Results

- Compare with currently best published results:
  - **For 20 instances generated by Cordeau et al.**: produces 19 new best known solutions, with improved quality of 0.75% on average in term of best solutions cost.
  - **For 45 instances generated by S.Pirkwieser and Raidl**: produces solutions with improve quality of 0.88% on average in term of average solutions cost.

Numerical Results

Instances						UTB	VNS	HGA	%GAP to BKS
No	n	T	m	D	Q				
1a	48	4	3	500	200	3007.84	2989.58	**2989.58**	0
2a	96	4	6	480	195	5328.33	5127.98	**5107.51**	-0.40
3a	144	4	9	460	190	7397.10	7260.37	**7158.77**	-1.40
4a	192	4	12	440	185	8376.95	8089.15	**7981.85**	-1.33
5a	240	4	15	420	180	8967.90	8723.63	**8666.59**	-0.65
6a	288	4	18	400	175	11686.91	11063.00	**10999.90**	-0.57
7a	72	6	5	500	200	6991.54	6917.71	**6892.71**	-0.36
8a	144	6	10	475	190	10045.05	9854.36	**9751.66**	-1.04
9a	216	6	15	450	180	14294.97	13891.03	**13707.30**	-1.32
10a	288	6	20	425	170	18609.72	18023.62	**17754.20**	-1.49
1b	48	4	3	500	200	2318.37	2289.17	**2284.83**	-0.19
2b	96	4	6	480	195	4276.13	4149.96	**4141.15**	-0.21
3b	144	4	9	460	190	5702.07	5608.67	**5567.15**	-0.74
4b	192	4	12	440	185	6789.73	6534.12	**6471.74**	-0.95
5b	240	4	15	420	180	7102.36	6995.87	**6963.11**	-0.47
6b	288	4	18	400	175	9180.15	8895.31	**8855.97**	-0.44
7b	72	6	5	500	200	5606.08	5517.71	**5509.08**	-0.16
8b	144	6	10	475	190	7987.64	7712.40	**7677.68**	-0.45
9b	216	6	15	450	180	11089.91	10944.59	**10874.80**	-0.64
10b	288	6	20	425	170	14207.64	14065.16	**13851.40**	-1.52

Conclusion

- This algorithm outperforms the best existing methods for solving PVRPTW.
- It is part of a larger project to develop a cooperative system for VRP:
  - A set $S = \{1, 2, \ldots, n\}$ of n different search agents with dynamics $x_i(t+1) = h_i(x_i(t-1)),\ i \in S$
  - Agents are part of a network which can be represented by an oriented graph $G = (V, E)$, $V = \{1, 2, \ldots, n\}$ and $E \subseteq V \times V$
  - $N_i = \{j \in V \mid (i, j) \in E\}$ are the neighbors of search agent i
  - Cooperation protocol: $x_i(t+1) = f(\sum_{j \in N_i} (x_j(t))) + h(x_i(t))$