Order Crossover(P1, P2, C1, C2):

m <- Random(n); M <- Random(n)

if m > M

then swap m & M

S1 <- P1[from m to M]

S2 <- P2[from m to M]

R1 <- (P1 - S1)

R2 <- (P2 - S2)

s <- 0

r <- 0

for (from i <- 0 to n-1)

if (m <= i and i < M)

C1[i] <-S1[s]

C2[i] <-S2[s]

s <- s+1

else

C1[i] <-R2[r]

C2[i] <-R1[r]

r <- r+1

Position Base Crossover (P1, P2, C1, C2):

N <- Random(n)

P <- [ Random(n) ]\* N

S1 <- P1[P]

S2 <- P2[P]

R1 <- (P1 - S1)

R2 <- (P2 - S2)

s <- 0

r <- 0

for (from i <- 0 to n-1)

if (P contains i)

C1[i] <-S1[s]

C2[i] <-S2[s]

s <- s+1

else

C1[i] <-R2[r]

C2[i] <-R1[r]

r <- r+1

Order Base Crossover (P1, P2, C1, C2):

N <- Random(n)

Int [] P <- [ Random(n) ]\* N

Int []P’ <- P2.indexOf(P1[P])

S1 <- P1[P]

S2 <- P2[P’]

R1 <- (P1 - S1)

R2 <- (P2 - S2)

s <- 0

r <- 0

for (from i <- 0 to n-1)

if (P contains i)

C1[i] <- S1[s]

s <- s+1

else

C1[i] <-R2[r]

r <- r+1

s <- 0

r <- 0

for (from i <- 0 to n-1)

if (P’ contains i)

C2[i] <-S2[s]

s <- s+1

else

C2[i] <-R1[r]

r <- r+1

Cycle Crossover (P1, P2, C1, C2):

p <- Random(n)

e = P1[p]

E.insert(e)

while (e is not equal to -1)

e <- P1[P2.indexOf(e)]

if(E contains e)

e <- -1

E.insert(e)

P <- E

P’ <- P2.indexOf(E)

S1 <- P[P]

S2 <- P2[P’]

R1 <- (P1 - S1)

R2 <- (P2 - S2)

s <- 0

r <- 0

for (from i <- 0 to n-1)

if (P contains i)

C1[i] = S1[s]

s <- s+1

else

C1[i] = R2[r]

r <- r+1

s <- 0

r <- 0

for (from i <- 0 to n-1)

if (P’ contains i)

C2[i] = S2[s]

s <- s+1

else

C2[i] = R1[r]

r <- r+1

Inversion Mutation (B, A):

m <- Random(n); M <- Random(n)

if m > M

then swap m & M

for (from i <- M to m)

R.insert(i)

r <- 0

for (from i <- 0 to n)

if (i < M and i >m)

A[i] <- R[r]

r <- r+1

else

A[i] <- B[i]

Insertion Mutation (B, A):

p <- Random(n)

S <- B[p]

R <- (B-S)

r <- 0

t <- Random(R.length)

for (from i <- 0 to N)

if (i equals to t)

A[i] = S

Else

A[i] = R[r]

r <- r+1

Displacement Mutation (B, A):

m <- Random(n); M <- Random(n)

if m > M

then swap m & M

l <- M-m

for (from i <- 0 to l)

D[i] = B[m+i]

R <- (B-D)

r <- 0

d <-0

t <- Random(R.length)

for (from i <- 0 to N)

if (i equals to t)

for (from j <- i to D.lengh)

A[j] = D[d]

d <- d+1

i <- M+1

Else

A[i] = R[r]

r <- r+1

Reciprocal Exchange Mutation (B, A):

p <- Random(n); p’ <- Random(n)

for (from i <-0 to N)

A[i] <- B[i]

A[p] = B[p’]; A[p’] = B[p]