LISTING 1:

OVERALL ALGORITHM FOR THE PROGRAM

|  |  |
| --- | --- |
| 1: | START PROGRAM [input: *pop\_size*, *n\_iter*) |
| 2: | Instance←getUserInput() |
| 3: | If Instance==random: |
| 4: | generateRandomInstance() |
| 5: | Else: |
| 6: | readInstanceFromFile() |
| 7: | Population = createInitialPopulation(*pop\_size*=*N*) |
| 8: | assignCrowdingDistance(Population) |
| 9: | Parents = tournamentSelection(Population) |
| 10: | Children = crossover(pairs of Parents) |
| 11: | mutate(Children, mutation\_probability) |
| 12: | For *n\_iter* iterations, do: |
| 13: | Combined\_population = Parents U Children |
| 14: | Fronts = nondominatedSort(Combined\_population) |
| 15: | Next\_gen\_parents = None, i = 1 |
| 16: | While len(Next\_generation)+len(Fronts[i])<N, do: |
| 17: | assignCrowdingDistance(Fronts[i]) |
| 18: | Next\_gen\_parents = Next\_gen\_parents U Fronts[i] |
| 19: | i = i +1  End While. |
| 20: | crowdedSort(Fronts[i]) |
| 21: | Remnant = Fronts[i][1:*N*-len(Next\_gen\_parents)] |
| 22: | Next\_gen\_parents = Next\_gen\_parents U Remnant |
| 23: | Parents = tournamentSelection(Next\_gen\_parents) |
| 24: | Children = crossover(pairs of Parents) |
| 25: | mutate(Children, mutation\_probability) |
| 26: | End For. |
| 27: | END PROGRAM |

Chart, scatter chart

Description automatically generated

Fig. 2: “Optimal” pareto fronts obtained by varying crossover operator and mutation probability for random generated instance (with 51 cities, 7 salesmen) of the MinMax SD-MTSP. The HX operator performs significantly better than the other, with OX better than PMX, all much better than CX. Mutation probability does not seem to have much influence on HX performance, while it does for others. Although 0.08 seems to be slightly better, for other instances, 0.05 was similarly slightly better. Hence, mutation probability seems inconsequential for NSGA-II with HX

Chart, scatter chart

Description automatically generated

Fig. : Optimal pareto fronts obtained by varying population size for a different randomly generated instance (again with 51 cities, 7 salesmen) of the MinMax SD-MTSP, keeping mutation probability = 0.05 and the HX operator. Pop=100 performs somewhat better at balancing the tours, while pop=200 is better at minimizing total cost (distance). However, pop=300 has better performance at optimizing both objectives together. Since there is not much difference, the slight differences in performances can be ignored. Indeed, in other runs, the situation was slightly different.

Chart, line chart

Description automatically generated

Fig. 1: Plot showing two convex pareto fronts that could be obtained while optimizing an arbitrary bi-objective problem with objective function obj fn 1 and obj fn 2. The blue front dominates the red front. Other pareto fronts could be concave or irregularly shaped (neither concave or convex).

Chart, line chart

Description automatically generated

Fig. 1: Plot showing two convex pareto fronts that could be obtained while optimizing an arbitrary bi-objective problem with objective functions obj fn 1 and obj fn 2. The blue front dominates the red front. Other pareto fronts could be concave or irregularly shaped (neither concave nor convex).