Nature Inspired Optimisation for Delivery Problems
Chapter 4: Solving Problems that have Dual Solution Characteristics

Neil Urquhart

May 27, 2022

These slides are designed to accompany the book "Nature Inspired Optimisation for Delivery Problems : From Theory to the Real World".

https://link.springer.com/book/10.1007/978-3-030-98108-2

Dealing with Solutions with Twin-Characteristics

Introduction

- In the previous chapter we optimised our CVRP problem based upon a single objective (distance)
- In practice our CVRP problem has two potential objectives vehicles/routes and distance
- When solving our problem is would be useful to take into account **both** objectives

Objectives and Characteristics

- We will use the term *solution characteristic* to describe vehicles and distance
- From an algorithmic perspective we face the question, which is more important distance o routes?
- We adopt the approach of finding a set of solutions with differing characteristics and allowing the user to make the final choice of solution.

The important aspect is to present a set of options to the user :

- that displays the range of options available to them
- is small enough for them to comprehend and make a meaningful choice

A Bi-Objective Example

Let's define a vehicle routing problem:

- We have two objectives
  - emissions (CO_2) e
  - average delivery time t (the average time taken to reach each customer)
- We have the option of making use of diesel powered vans or cargo bikes.
  - Diesel powered vans are quick, but produce CO_2
  - Cargo bikes are far slower but produce no emissions.

If we wish to decrease t we use more vans as they are quicker, but this increases e, if we use more cargo bikes then we reduce e, but increase t. We can describe e and t as being *conflicting*.

Conflicting Objectives

When two characteristics are conflicting then an attempt to reduce one, generally results in an increase in the other.
There would appear to be three choices of solution available to us:

1. Find a solution that optimises e at the expense of t
2. Find a solution that optimises t at the expense of e
3. Find a solution that optimises t and e such that neither is optimal, but both are acceptable to the user

The third type of solution is often referred to as a *trade-off*. The decision as to whether to give priority to e or t is not really a decision for us to make, ultimately that decision lies in the hands of the domain expert.

Pareto Dominance

Ranking Solutions

- We need a means of ranking two solutions and establishing which is the more fit.
- , For a single characteristic this becomes trivial, we simply compare the numerical fitness values.

One means of dealing with bi-objective problems is to combine the objectives into a single fitness value f thus:

$$f = (xa) + (yb)$$

Where a and b are weights applied to the objectives.
This has the immediate drawback that values for a and b must be set, which involves giving one a higher weighting, and therefore, implied importance importance over the other.

Pareto Dominance

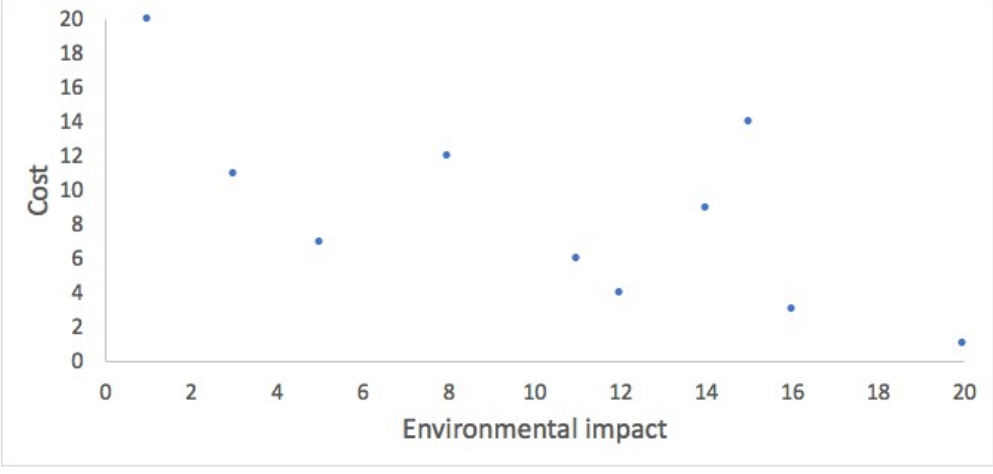
We can use the technique of *Pareto Dominance* to compare solutions. The Italian economist and engineer Vilfredo Pareto established the concept of *Pareto Dominance*, allowing us to compare two solutions and examine if one *dominates* the other.
Assuming that we have two solutions s_x and s_y:
s_x is said to *dominate* s_y if:

- s_x is no worse than s_y in all characteristics
- s_x is better than s_y in at least one characteristic

We can use the concept of dominance to rank solutions when undertaking selection within an EA, allowing us to take into account both characteristics and avoids the need to combine them into a fitness value.

Pareto Dominance Example

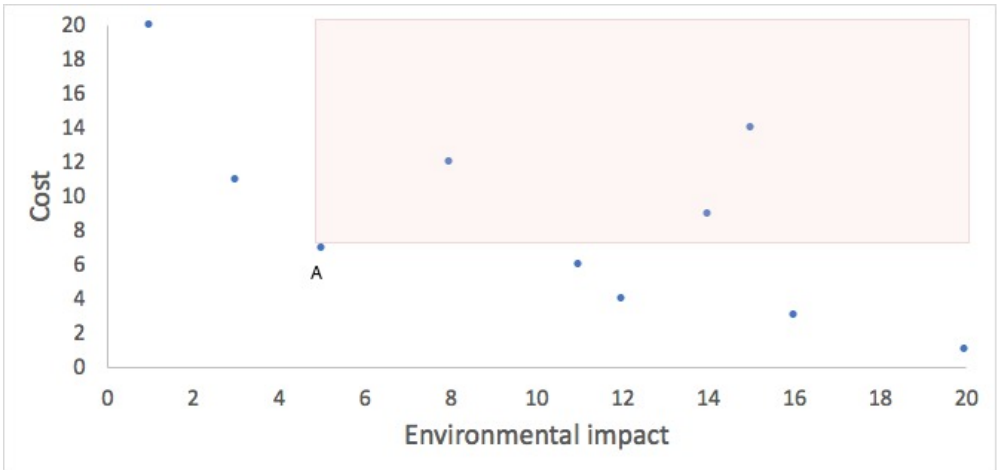
We can plot a set of solutions by their characteristics :



We would like a solution that minimises both characteristics (1,1) - but this probably does not exist!
Note that the population includes two solutions (20,1) and (1,20) which optimise one of the characteristics at the expense on the other.

Pareto Dominance Example

We select a solution at random (A) and apply the Pareto dominance rules, every solution in the shaded area is dominated:

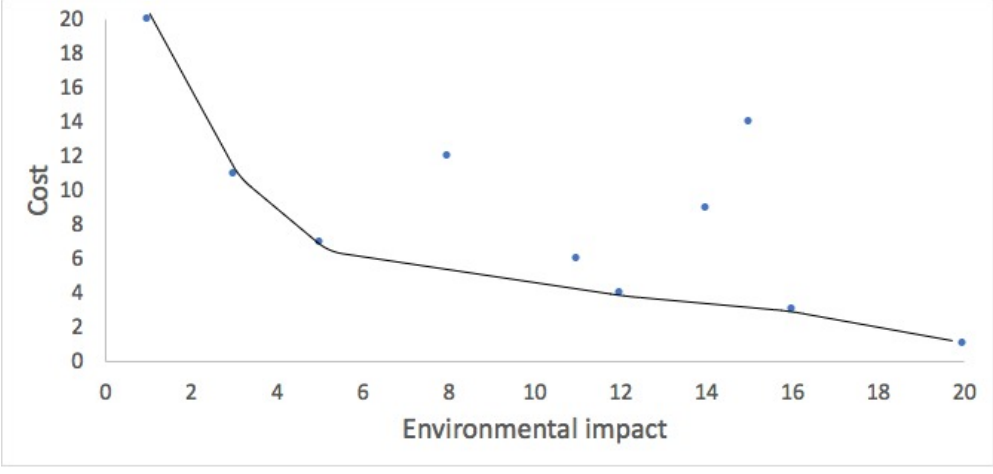


	e	c
	8	12
	14	9
	15	14
A	5	7

A is not dominated by any other solution, therefore we term it *non-dominated*.

Non-Dominated Fronts

The complete set of non-dominated solutions is known as the *non-dominated front*:



c	e	
20	1	minimises e
11	3	trade-off
7	5	trade-off
5	12	trade-off
3	16	trade-off
1	20	minimises c

An Evolutionary Algorithm to Find Non-Dominated Fronts

Introduction

- A non-dominated front of solutions is a useful set to present to the user, to allow them to make their final choice of solution.
- As a population-based heuristic, the Evolutionary Algorithm is easily adaptable to finding Pareto fronts.
- The evolutionary mechanism continues to be used to evolve a population, when the algorithm has completed a non-dominated front can be extracted.
- Our modified version of the EA we will carry out selection based on dominance, rather than fitness.

Selection Based on Pareto Dominance

Procedure DominatedTournament(population[])
x = population.random()
y = population.random()
if *x.dominates(y)* **then**
  return x
end
else
  return y
end

Case Study: Routes Versus Customer Service

Introduction

A company have a simple CVRP based delivery problem to solve, they have two solution characteristics that they wish to take into account:

- **Number of routes**: The number of vehicles and drivers required.
- **Customer service**: A metric based on the time taken for each customer to receive their order, each customer wants their order as quickly as possible.

It is useful to consider two extreme examples of solutions based on the above characteristics

- Minimise the customer service metric by having each customer served by a dedicated route
- Minimise the number of routes by placing the maximum number of customers into each route

A Revised Representation

In order to be able to generate the extreme solutions we need to be able to control how the grand tour is divided into routes.

- Our previous representation for solving the CVRP used a simple Grand Tour representation.
- The decoder used determined how the tour would be split based on vehicle capacity
- We now need to be able to specify additional routes as and not just split on the capacity constraint

A Revised Representation and Decoder

We modify our representation so that each element of the grand tour also contains a Boolean flag to denote whether the customer should be placed at the start of a new route.
The decoder still commences a new route if the capacity constraint is about to be broken.
Assuming that we have customer deliveries A to E with demands as follows:

Cust	Demand
A	2
B	3
C	2
D	1
E	2

A Revised Representation and Decoder ... cont

An example of a valid genotype would be:

| A,False | B,True | C,False | D,True | E,True |

Assuming that we have a capacity of 5, the phenotype decodes to the following genotype under the current decoder:

Route	**Customers**
1	A B
2	C D E

Using the modified decoder (taking into account the New Route flags) the following phenotype would be created:

Route	**Customers**
1	A
2	B C
3	D
4	E

Implementing Our EA

We can now implement a modified EA to solve our bi-objective problem. To summarise the changes we need to make are:

- Revised representation to include the new route flag
- Revised decoder to split on the new route flag and on the capacity constraint
- Selection operator based on dominance
- Identification of the non-dominated front from the population at the end of the run

Summary

Summary

- Where multiple conflicting solution characteristics exist a human expert may be best placed to prioritise them
- The approach outlined in this chapter allows a human expert to make final choice of solution
- As our problem is NP-Hard. we commence with a vast number of solutions (too many for an expert to consider equally), using dominance we *filter* the solutions to the very much smaller non-dominated front, from which the choice of final solution may be made.