Chapter 4

Multi-Objective Optimisation

Ant colony optimisation (ACO) algorithms were originally proposed for single-objective optimisation problems [51, 52, 53, 54, 56, 195]. For many real-world problems it is necessary to optimise more than one conflicting objective (multi-objective optimisation) simultaneously. In this respect, an ACO has to be modified in order for it to be applicable to multi-objective optimisation problems [11, 79, 134].

This chapter provides an overview of the different aspects of multi-objective optimisation. A definition of Pareto-optimality is provided and various multi-objective optimisation algorithm classes are discussed. The adaptation of ACO algorithms for multi-objective optimisation problems is discussed. In addition to the ACO algorithms for multi-objective optimisation, the elitist non-dominated sorting genetic algorithm (NSGA-II) is also described in detail. The NSGA-II is included in this chapter, as it is used to compare the results with those of the ACO algorithms proposed in this thesis. Performance metrics that can be used to compare the performance of multi-objective algorithms are also discussed.

4.1 Introduction

Many real-world problems require the simultaneous optimisation of a number of objective functions. This is referred to as multi-objective optimisation (MOO) [208] and presents a situation in which certain of the required objectives may be in conflict with one another. An example of a multi-objective problem (MOP) is compressor design where the major objectives are the maximisation of overall isentropic efficiency, the maximisation of mass flow rate, the maximisation of total pressure ratio, the minimisation of weight, and the maximisation of durability. Another example of a MOP is routing in data communication networks, where the objectives may include minimisation of routing cost, minimisation of route length, minimisation of congestion, and maximisation of the utilisation of physical infrastructure. There is an important trade-off between the last two objectives, as minimisation of congestion is achieved by reducing the utilisation of links. A reduction in utilisation, on the other hand, means that infrastructure, for which high installation and maintenance costs are incurred, is under-utilised. Solutions to such problems require a balance between conflicting objectives.

The remainder of this chapter is organised as follows: Section 4.2 provides a theoretical overview of the multi-objective problem (MOP). Section 4.3 discusses the concepts of Pareto-optimal set and Pareto-optimal front. Section 4.4 presents a summary of MOO algorithm classes. Section 4.5 demonstrates the way in which ACO can be adapted to solve multi-objective problems. Section 4.6 discusses evolutionary multi-objective optimisation (EMO) and the NSGA-II algorithm. Section 4.7 discusses performance metrics for comparing the performance of multi-objective algorithms.

4.2 Multi-Objective Optimisation Problem

Let $S \subseteq \mathbb{R}^{n_x}$ denote the n_x-dimensional search space defined by a finite set of decision variables. Let $\mathbf{x} = (x_1, x_2, ..., x_{n_x}) \in S$ refer to a *decision vector*. A single objective function, $f_k(\mathbf{x})$, is defined as $f_k : \mathbb{R}^{n_x} \rightarrow \mathbb{R}$. Let $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_{n_o}(\mathbf{x})) \in \mathcal{O} \subseteq \mathbb{R}^{n_o}$ be an objective vector containing n_o objective function evaluations; $\mathcal{O}$ is referred to as the *objective space*. The search space, S, is also referred to as the *decision space*. Let $\mathcal{F} \subseteq S$ denote the feasible space which is constrained by n_g-inequality and n_h-equality constraints, i.e.

$$\mathcal{F} = \{\mathbf{x} : g_m(\mathbf{x}) \leq 0,\ h_l(\mathbf{x}) = 0,\ m = 1, ..., n_g;\ l = 1, ..., n_h\} \tag{4.1}$$

where g_m and h_l are the inequality and equality constraints respectively. With no constraints the feasible space is the same as the search space, S.

Using the notation above, a multi-objective optimisation problem is defined as:

$$\begin{aligned} \text{minimise} \quad & \mathbf{f}(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{F} \\ & \mathbf{x} \in [x_{min},\ x_{max}]^{n_x} \end{aligned} \tag{4.2}$$

Solutions, $\mathbf{x}^*$, to the MOP are in the feasible space, $\mathcal{F} \subseteq S$. In order for $\mathbf{x}^*$ to be in the feasible space, $\mathcal{F}$, both the inequality and equality constraints have to be satisfied. The main issue in MOO is the presence of conflicting objectives, where improvement in one objective may result in deterioration in another objective. Trade-offs do exist between such conflicting objectives, however, and the task is to find solutions which balance these trade-offs. Such a balance may be achieved when a solution is unable to effect an improvement in any of the objectives without degrading one or more of the other objectives. These solutions are referred to as non-dominated solutions of which many may exist.

Therefore, the objective when solving a MOP is to produce a set of acceptable compromises rather than a single solution. This set of solutions is referred to as the non-dominated set or the Pareto-optimal set. The plot of the objective functions whose non-dominated solutions are in the Pareto-optimal set is called the Pareto front.

4.3 Pareto-Optimality

This section presents a number of definitions which pertain to MOO.

Definition 4.3.1. Domination: For two decision vectors, $\mathbf{x}$ and $\mathbf{z}$, $\mathbf{x}$ dominates $\mathbf{z}$, noted $\mathbf{x} \prec \mathbf{z}$, if and only if $\mathbf{x}$ is equally good or better than $\mathbf{z}$ for each of the objectives to optimise, i.e.

$$f_k(\mathbf{x}) \leq f_k(\mathbf{z}),\ \forall k \in \{1,2,..,n_o\} \wedge \exists k \in\ \{1,2,..,n_o\} | f_k(\mathbf{x}) < f_k(\mathbf{z}) \tag{4.3}$$

The concept of dominance is illustrated in Figure 4.1 for a two-objective function, $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}))$. The shaded area denotes the area of the objective vectors which are dominated by $\mathbf{f}$.

Definition 4.3.2. Weak domination: A decision vector, $\mathbf{x}$, weakly dominates a decision vector, $\mathbf{z}$, noted $\mathbf{x} \preceq \mathbf{z}$, if and only if $\mathbf{x}$ is not worse than $\mathbf{z}$ for each of the objectives to optimise, i.e.

$$f_k(\mathbf{x}) \leq f_k(\mathbf{z}),\ \forall k \in \{1,2,..,n_o\} \tag{4.4}$$

Definition 4.3.3. Pareto-optimal: A decision vector $\mathbf{x}^* \in \mathcal{F}$ is termed a Pareto-optimal solution for the MOP (refer to equation (4.2)) if there does not exist a decision

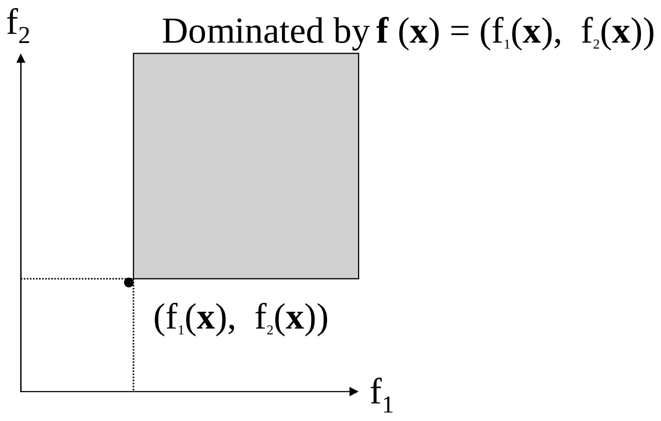

Figure 4.1: The concept of dominance

vector, $\mathbf{x} \neq \mathbf{x}^* \in \mathcal{F}$ that dominates $\mathbf{x}^*$, i.e. $\nexists \mathbf{x} : f_k(\mathbf{x}) < f_k(\mathbf{x}^*) \; \forall k \in \{1, 2, .., n_o\}$. An objective vector $\mathbf{f}^*(\mathbf{x})$ is Pareto-optimal if $\mathbf{x}$ is Pareto-optimal.

In words, a decision vector, $\mathbf{x}^*$, is Pareto optimal if there exists no feasible vector of decision variables $\mathbf{x} \in \mathcal{F}$ which would decrease some criterion without causing a simultaneous increase in at least one other criterion. The presence of multiple objective functions, usually conflicting among them, give rise to a set of optimal solutions called the *Pareto-optimal set*.

Definition 4.3.4. Pareto-optimal set: The set of all Pareto-optimal decision vectors form the Pareto-optimal set, $\mathcal{P}^*$. That is,

$$\mathcal{P}^* = \{\mathbf{x}^* \in \mathcal{F} | \; \nexists \mathbf{x} \in \mathcal{F} : \mathbf{x} \prec \mathbf{x}^*\} \tag{4.5}$$

Therefore, the Pareto-optimal set contains the set of solutions, or balanced trade-offs, for the MOP. The corresponding objective vectors are referred to as the Pareto-optimal front $\mathcal{PF}^*$.

Definition 4.3.5. Pareto-optimal front: Given the objective vector, $\mathbf{f}(\mathbf{x})$, and the Pareto-optimal solution set, $\mathcal{P}^*$, then the Pareto-optimal front, $\mathcal{PF}^*$ is defined as

$$\mathcal{PF}^* = \{\mathbf{f}(\mathbf{x}^*) = (f_1(\mathbf{x}^*), f_2(\mathbf{x}^*), ..., f_{n_o}(\mathbf{x}^*)) | \mathbf{x}^* \in \mathcal{P}^*\} \tag{4.6}$$

An example of a Pareto front is illustrated in Figure 4.2. Figure 4.3 illustrates the assignment of an *objective vector*, **f**, to the *decision vector*, **x**.

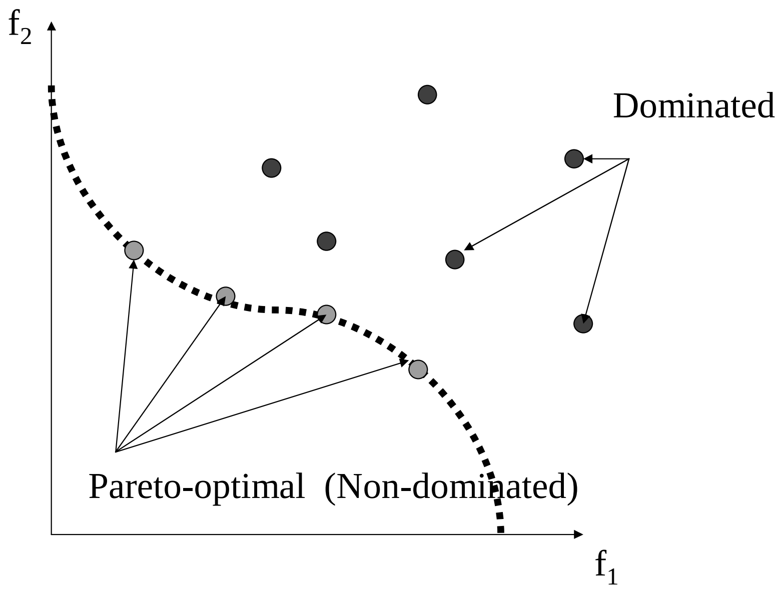

Figure 4.2: Pareto-optimal front for objectives f_1 and f_2

4.4 Multi-Objective Optimisation Algorithm Classes

It may be computationally expensive to generate the Pareto front [224]. Owing to the computational complexity, exact methods to find all non-dominated solutions are not feasible.

For this reason, a number of stochastic search strategies such as particle swarm optimisation (PSO) [60, 63], evolutionary algorithms (EAs) [15], tabu search [81], simulated annealing [118], and ant colony optimisation (ACO) [54, 56] have been developed. These strategies endeavour to find a set of solutions for which the objective vectors are not too far removed from the optimal objective vectors.

MOO involves guiding the search towards the true Pareto front while maintaining a diverse set of non-dominated solutions. The task of MOO is thus reduced to finding an

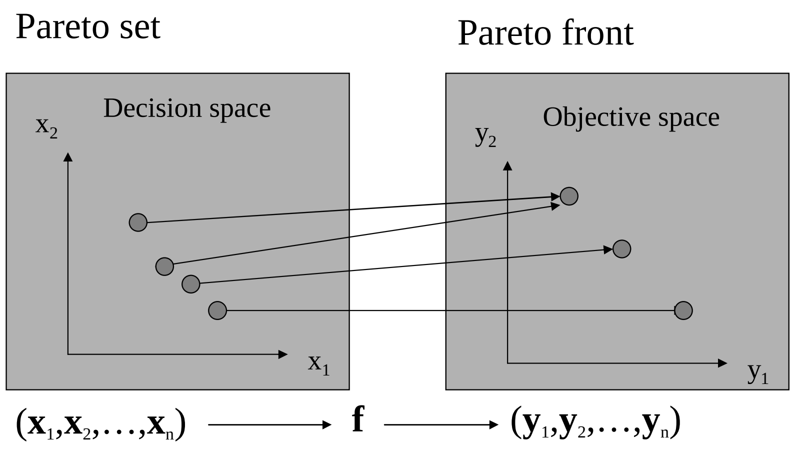
Pareto set
Pareto front
Decision space
Objective space
x_2
x_1
y_2
y_1
$(\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_n) \longrightarrow \mathbf{f} \longrightarrow (\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_n)$

Figure 4.3: Mapping between decision space and objective space

approximation to the true Pareto front such that

- the distance to the true Pareto front is minimised,
- the set of non-dominated solutions, i.e. the Pareto-optimal set, is as diverse as possible, and
- non-dominated solutions which have already been found are maintained.

The first objective is addressed by assigning an appropriate fitness function in order to quantify the quality of a solution in the presence of multiple optimisation criteria. In terms of the second objective, methods are used which preserve the diversity of non-dominated solutions [42, 226]. The third objective which, in essence, addresses both the first two objectives is achieved by using archives of previously found non-dominated solutions [120]. The use of archives is a type of elitist strategy, where the best solutions are maintained in a repository.

Depending on the fitness assignment strategy, classes of MOO algorithms are grouped into:

- **Aggregation-based methods**. The objectives are aggregated into a single parameterised objective function. The parameters of this function are systematically varied during the optimisation run in order to find a set of non-dominated solutions instead of a single trade-off solution. The aggregation-based method is referred to as the weighted sum method [143]. Another aggregation-based method is the epsilon-constraint method [46, 213], which involves optimising a primary objective and expressing the other objectives in the form of inequality constraints.

Although aggregation methods have been successfully applied to solve MOPs, the following disadvantages of these methods should be noted:

  – Aggregation methods can find only one solution with a single application of the algorithm. To find more than one solution repetitive applications of the algorithm are required. However, repetitive applications do not guarantee that distinct Pareto-optimal solutions will be found.
  – Optimal values for the weights are problem dependent. Care must be taken when choosing the values of the weights in order to ensure that an acceptable solution is found.

- **Criterion-based methods**. Criterion-based methods [76, 142] switch between the objectives during the optimisation process, i.e. different stages of the optimisation process use different objectives.

- **Dominance-based methods**. One of the most important issues in MOO is that of determining when one solution is better than another with respect to all objectives. To address this issue the notion of dominance is used (refer to Section 4.3).

  Dominance-based MOO algorithms [104, 177], using an archive, provide an efficient way in which to find multiple Pareto-optimal solutions simultaneously in a single simulation run.

The following paragraphs briefly describe MOO algorithms that make use of Pareto dominance to find a set of non-dominated solutions.

Evolutionary algorithms (EA) refers to a class of stochastic optimisation methods that simulate the process of natural evolution [224]. As a result of the population-based nature of EAs, EAs have been widely used in multi-objective optimisation as the population-based nature of EAs allows the generation of several elements of the Pareto-optimal set with a single run [16, 40, 167].

Particle swarm optimisation (PSO) [60, 61] refers to a population based stochastic optimisation technique which was inspired by the social behaviour of bird flocking. PSO algorithms have been adapted to maintain a set of non-dominated solutions using the Pareto dominance [102, 158, 171].

Ant colony optimisation (ACO) algorithms have been adapted to solve MOPs [30, 49, 76, 85, 104, 177]. Most of the ACO algorithms adapt the way that pheromone and heuristic information is used to increase diversity. Diversity is improved by ensuring better exploration by artificial ants.

The rest of the chapter focuses on describing these MOO ACO algorithms, as MOO ACO algorithms are used in this thesis to solve the multi-objective power-aware routing problem.

4.5 Ant Colony Optimisation for Multi-Objective Optimisation

Ant colonies are increasingly used to solve various optimisation problems [51, 52, 53, 54, 56, 195]. Most of these applications of ACO algorithms are for single-objective optimisation. For many real-world problems it is necessary to optimise more than one conflicting objective simultaneously. In this respect, ACO algorithms have been adapted to find a set of acceptable non-dominated solutions that cover, in the best way possible, the various regions of the true Pareto front.

This section describes the different ways in which ACO has been adapted to solve MOPs in general and is organised as follows: Subsection 4.5.1 discusses the different issues with MOACO algorithms, while Subsections 4.5.2 to 4.5.6 respectively discuss the single colony, single-pheromone, single-heuristic matrix methods, the single colony, single-pheromone, multi-heuristic matrix methods, the single colony, multi-pheromone, single-heuristic matrix methods, the single colony, multi-pheromone, multi-heuristic matrix methods, and the multi-colony MOO algorithms.

4.5.1 Introduction

Very few studies have dealt with MOO using multi-objective ACO algorithms (MOACOs). The design of a MOACO algorithm should address the following issues [79, 134]:

- **The management of the pheromone information in MOO.** ACO algorithms for single objective problems represent the pheromone information in terms of a single pheromone matrix (or vector) in which each entry in the matrix corresponds to the desirability of the move from node i to node j. One pheromone matrix represents one objective, because pheromone updates are proportional to some objective function expressing the quality of a solution (trail) or partial solution.

  So, the issue is to change the way in which the pheromone matrix is used to account for multiple objectives. This can be achieved either by keeping a single pheromone matrix [142], where pheromone updates are proportional to a weighted sum of updates, where each update corresponds to an objective, or using multiple pheromone matrices, one for each objective [104].

  The essential difference between single and multi-pheromone matrix methods is that those algorithms that use multiple pheromone matrices are able to keep objective specific history information completely partitioned, whereas those that use a single pheromone matrix must combine this information.

  When a MOACO algorithm uses multiple pheromone information and a one-to-one pheromone to objective mapping is used then ideally the pheromone information contained in a single matrix will reflect which solution components advantage a particular objective.

  The choice of the pheromone model depends on design issues such as how the solution construction process uses pheromone information and how pheromone matrices are updated and decayed.

  The reduction in memory associated with using a single pheromone matrix versus multiple pheromone matrices is also a motivating factor when choosing a pheromone model [10]. To make positive use of the extra memory required by multiple pheromone matrices, pheromone matrices must contain different information by being updated or decayed non-uniformly or by being mapped differently during the pheromone update and solution construction processes.

- **The management of heuristic information in MOO.** Single-objective ACO algorithms use one heuristic information matrix to represent the attractiveness of each edge with reference to a single objective. In a multi-objective optimisation problem with n_o objectives, n_o different cost factors (heuristic information functions) are defined between each pair of nodes, one for each objective. The heuristic information must then be changed to account for multiple objectives. This can be achieved by using two different strategies. The first strategy is to consider an aggregation of the different objectives into a single heuristic information, where each entry is an aggregated value of the attractiveness of each edge [27, 49, 72]. A second strategy is to consider each different objective separately. In this case separate heuristic information matrices are maintained, one for each objective function [56, 76, 177].

- **Pareto archival.** Single-objective ACO algorithms have all ants converging to a single solution. MOACO has to have the ability to find multiple solutions, which can be achieved using a Pareto archive. Pareto archival is the method by which multiple Pareto optimal solutions are stored for post algorithm run-time analysis or use [30, 104, 148, 161]. A storage repository (called an archive) is used to store the non-dominated solutions that are found.

- **Balancing exploration against exploitation.** The balance between exploration and exploitation is guided by the ants' memory, the pheromone matrices with pheromone information accumulated by all the ants from the beginning of the search process, the problem-specific heuristic information, the pheromone evaporation, and the pheromone update.

  Balancing exploration against exploitation is necessary in order to meet the two MOACO goals:

  – to find a number of solutions that are close to the Pareto front (quality enhancing behaviour), and

  – to maintain a coverage of solutions along the entire Pareto front (diversity preserving behaviour).

The use of several colonies can serve to achieve a balance between exploration and exploitation. Communication between colonies emphasises exploitation by recruiting colonies to work in the same region of the search space. Less communication has an explorative effect, since each colony is more likely to be searching in a different part of the search space.

- **Pheromone update.** In single objective optimisation problems, the best performing ACO algorithms often use only the best solutions of each iteration (iteration-best strategy) or, since the start of the algorithm (best-so-far strategy), for updating the pheromones. However, in the multi-objective case it is more difficult to determine which are the best solutions to be chosen for the pheromone update.

  Two different ways of implementing the pheromone update are possible: the *selection by dominance* strategy [104, 177] and the *selection by objective* strategy [76]. The selection by dominance strategy allows only the non-dominated solutions to update pheromone concentrations. An iteration-best strategy would consider the non-dominated solutions among those generated in the current iteration; a best-so-far strategy would be obtained by choosing only solutions of an archive of the non-dominated solutions found since the start of the algorithm.

  The selection by objective strategy allows solutions that find the best values for each objective within the current cycle or since the start of the algorithm to update pheromone concentrations.

  If the selection by objective strategy is used and multiple pheromone information is considered, each pheromone matrix associated with each objective will be updated by the solution with the best objective value for the respective objective. Selection by objective has two benefits:

  - As in MMAS and ACS, one solution per pheromone matrix only will be allowed to deposit pheromones, which leads to improved performance compared to the original AS (see Sections 3.5.2 and 3.5.3). As a result the advanced techniques used in the MMAS and ACS algorithms may easily be adapted to multi-objective problems.
  - Each pheromone matrix focuses on one objective only, and thus the aggregation of all the pheromone matrices by means of a weight vector truly regulates the relative importance of each objective.

Recently, different ACO algorithms for multi-objective problems have been developed which address the issues mentioned above. Gravel *et al.* [85] proposed a MOACO based on the AS algorithm using a single heuristic information matrix and a single pheromone matrix. Schaerer and Barán [177] adapted the ACS to use two heuristic matrices and a single pheromone matrix, while Pinto *et al.* [161] adapted the MMAS to use multiple heuristic matrices and a single pheromone matrix. Cardoso *et al.* [30] and Doerner *et al.* [49] modified the AS and ACS respectively to employ a pheromone matrix for each of the objectives using a single heuristic matrix information. Iredi *et al.* [104] adapted the AS for two objectives by including two pheromone matrices and two heuristic matrices – one for each objective. Mora *et al.* [148] adapted the ACS to use two heuristic matrices and two pheromone matrices.

With respect to the number of colonies used, either one colony can be used [177], or one colony for each objective function [142]. The above algorithms all use only one colony. Gambardella *et al.* [76] and Iredi *et al.* [104] have proposed the use of multiple colonies, where each colony focus on the optimisation of one of the objectives. Using several colonies can serve different goals. The usual aim is to have colonies that specialise to find good solutions in different regions of the Pareto front, but it could also be used to let each colony specialise on a given objective.

MOACO algorithms can be classified according to different criteria. One of them could be whether the algorithm returns a set of non-dominated solutions, i.e. if it looks for a set of Pareto solutions during its run, or it just gives a single solution as output. Another interesting criterion is the way the pheromone information is updated.

For the purpose of this thesis, MOACO algorithms are examined and classified into:

- single colony, single-pheromone matrix, single-heuristic matrix algorithms,
- single colony, single-pheromone matrix, multi-heuristic matrix algorithms,
- single colony, multi-pheromone matrix, single-heuristic matrix algorithms,
- single colony, multi-pheromone matrix, multi-heuristic matrix algorithms, and
- multi-colony algorithms.

The following subsections provide a short overview of these classes of algorithms.

4.5.2 Single Colony, Single-Pheromone, Single-Heuristic Matrix Methods

Gravel *et al.* [85] proposed a MOACO called a multiple objective ACO metaheuristic (MOACOM). MOACOM is based on the AS algorithm and uses a single heuristic information matrix and a single pheromone matrix. Each value of the heuristic information matrix and each value of the pheromone matrix is the result of the aggregation of information associated with every objective. Considering these two matrices, the AS transition rule is applied to build the ants' solutions.

MOACOM deals with the multiple objectives in a lexicographic order, a priori established by the decision maker. At the end of each iteration, only the first solution according to the lexicographic order considered is taken into account and the pheromone matrix is updated on the basis of the evaluation of the primary objective. Each ant k deposits $\Delta\tau_{ij}^k$ pheromone on each link (i, j) used by the ant as follows:

$$\Delta\tau_{ij}^k = \frac{Q}{f^0(\mathbf{T}_k)} \tag{4.7}$$

where Q is a constant related to the amount of pheromone laid by the ants, f^0 is the primary objective function, and $\mathbf{T}_k$ is the solution built by the ant k.

MOACOM uses elite solution storage, and hence does not maintain populations of non-dominated solutions. As such, MOACOM saves on the computation costs associated with maintaining a non-dominated solution set.

4.5.3 Single Colony, Single-Pheromone, Multi-Heuristic Matrix Methods

Single colony, single-pheromone, multi-heuristic matrix methods use one pheromone matrix and multiple heuristic matrices, one for each of the sub-objectives.

Schaerer and Barán [177] adapted the ACS to use two heuristic matrices. The algorithm, referred to as multi-objective ant colony system (MOACS), changes the ACS transition rule to

$$j = \begin{cases} Arg\ Max_{u \in N_i^k(t)}\{\tau_{iu}(t)(\eta_{iu}^1)^{\beta\lambda_k}(t)(\eta_{iu}^2)^{\beta(1-\lambda_k)}(t)\} & \text{if } r \leq r_0, \\ J & \text{otherwise} \end{cases} \tag{4.8}$$

where β weights the relative importance of the heuristic matrices of the different objectives with respect to the pheromone matrix, λ_k is computed for each ant k as $\lambda_k = \frac{k}{n_k}$, where n_k is the number of ants, and $J \in N_i^k(t)$ is a node that is randomly selected according to probability,

$$p_{iJ}^k(t) = \frac{\tau_{iJ}(t)(\eta_{iJ}^1)^{\beta\lambda_k}(t)(\eta_{iJ}^2)^{\beta(1-\lambda_k)}(t)}{\sum_{u \in N_i^k(t)} \tau_{iu}(t)(\eta_{iu}^1)^{\beta\lambda_k}(t)(\eta_{iu}^2)^{\beta(1-\lambda_k)}(t)} \tag{4.9}$$

The local pheromone update is the same as for the original ACS, with τ_0 initially calculated as follows:

$$\tau_0 = \frac{1}{\bar{f}^1 \bar{f}^2} \tag{4.10}$$

where $\bar{f}^1$ and $\bar{f}^2$ are the average objective values over a set of heuristically obtained solutions (prior the execution of the ant algorithm) for the objective functions, f^1 and f^2, respectively. The value of τ_0 is not fixed during the algorithm run, but undergoes adaptation taking a new value $\tau_0^{'}$. Every time an ant k builds a complete solution, $\mathbf{T}_k$, this solution is compared to the current set of non-dominated solutions, $\mathcal{P}$, to check whether $\mathbf{T}_k$ is a non-dominated solution. If $\mathbf{T}_k$ is a non-dominated solution it is included in $\mathcal{P}$ while the solutions dominated by $\mathbf{T}_k$ are deleted from $\mathcal{P}$. At the end of each iteration, $\tau_0^{'}$ is calculated by applying equation (4.10) with the average values of each objective function taken from the solutions currently included in $\mathcal{P}$. If $\tau_0^{'} > \tau_0$, then $\tau_0 = \tau_0^{'}$, and the pheromone matrices are reinitialised to the new value of τ_0; otherwise, the global update is performed for each solution, $\mathbf{T}_p$, in $\mathcal{P}$ by applying the following rule:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \frac{\rho}{f^1(\mathbf{T}_p)f^2(\mathbf{T}_p)}, \ \forall (i,j) \in \mathbf{T}_p \tag{4.11}$$

Instead of relying on multiple pheromone matrices to guide objective specific solution construction, MOACS uses multiple heuristic matrices (one per objective) which are weighted in a specific way for each ant to bias solution construction toward different objective trade-offs. In other words, MOACS achieves diversity across the Pareto front through the use of heuristics rather than pheromones.

Pinto *et al.* [161] presented a multi-objective algorithm based on MMAS referred to as M-MMAS. M-MMAS simultaneously optimises four objectives (f^1, f^2, f^3, f^4), using a single pheromone matrix, τ, and three heuristic matrices (M-MMAS uses the same heuristic matrix for two of the objectives since both objectives are functions of the same heuristic information). The MMAS transition rule is changed to

$$p_{ij}^k(t) = \begin{cases} \frac{\tau_{ij}^{\alpha}(t)\prod_{l=1}^{3}(\eta_{ij}^l)^{\beta_l}(t)}{\sum_{h\in N_i^k(t)} \tau_{ih}^{\alpha}(t)\prod_{l=1}^{3}(\eta_{ih}^l)^{\beta_l}(t)} & \text{if} \quad j \in N_i^k(t) \\ 0 & \text{otherwise} \end{cases} \tag{4.12}$$

where β_1, β_2, and β_3 determine the relative influence among heuristic information.

The global update is performed for each solution, $\mathbf{T}_p$, of the current set of non-dominated solutions by applying the following rule:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \Delta\tau^p, \quad \forall(i,j) \in \mathbf{T}_p \tag{4.13}$$

where

$$\Delta\tau^p = \frac{1}{f^1(\mathbf{T}_p) + f^2(\mathbf{T}_p) + f^3(\mathbf{T}_p) + f^4(\mathbf{T}_p)} \tag{4.14}$$

The upper limit, τ_{max}, for the pheromone matrix is

$$\tau_{max} = \frac{\Delta\tau^p}{(1-\rho)} \tag{4.15}$$

The lower limit, τ_{min}, for the pheromone matrix is

$$\tau_{min} = \frac{\Delta\tau^p}{2n_k(1-\rho)} \tag{4.16}$$

Other MOACO approaches that use single pheromone and multiple heuristic matrices can be found in [72, 177].

4.5.4 Single Colony, Multi-Pheromone, Single-Heuristic Matrix Methods

This class of MOACO addresses the management of the pheromone information (refer to Section 4.5.1) using one pheromone matrix for each sub-objective. Assuming that n_o sub-objectives need to be optimised, n_o pheromone matrices are used.

Cardoso *et al.* [30] extended the AS to maintain multiple pheromone matrices. The AS transition rule (refer to equation (3.3)) is changed to

$$p_{ij}^k(t) = \frac{\eta_{ij}^{\beta}(t) \prod_{l=1}^{n_o} (\tau_{ij}^l)^{\alpha_l}(t)}{\sum_{h \in N_i^k(t)} \eta_{ih}^{\beta}(t) \prod_{l=1}^{n_o} (\tau_{ih}^l)^{\alpha_l}(t)} \tag{4.17}$$

where τ_{ij}^l represents the pheromone information for the l-th objective, η_{ij} is the heuristic information, $N_i^k(t)$ is the feasible neighbourhood of ant k at node i, β determines the relative importance of the heuristic information, and each α_l controls the influence of the corresponding objective.

After each iteration, evaporation is applied separately for the pheromone of each objective as follows:

$$\tau_{ij}^l = (1 - \rho_l)\tau_{ij}^l \tag{4.18}$$

where ρ_l is the pheromone evaporation rate for the l-th objective (a different evaporation rate is considered for each pheromone matrix).

At every iteration and for each objective, each ant k deposits $\Delta\tau_{ij}^l$ pheromone on each link (i, j) used by the ant, where

$$\Delta\tau_{ij}^l = \frac{Q}{f^l(\mathbf{T}_k)} \tag{4.19}$$

and Q is a constant related to the amount of pheromone laid by the ants, and f^l is the l-th objective function. Non-dominated solutions generated in each iteration are stored in an archive.

Similar to Cardoso *et al.*, Doerner *et al.* [49] modified the ACS to employ a pheromone matrix for each of the objectives. The ACS transition rule (refer to equation (3.7)) is changed to

$$j = \begin{cases} Arg\ Max_{u \in N_i^k(t)} \{ (\sum_{l=1}^{n_o} w_l \tau_{iu}^l(t)) \eta_{iu}^{\beta}(t) \} & \text{if } r \leq r_0, \\ J & \text{otherwise} \end{cases} \tag{4.20}$$

where w_l is the weight assigned to the pheromone matrix of each objective function, and $J \in N_i^k(t)$ is a node that is randomly selected according to probability,

$$p_{iJ}^k(t) = \frac{(\sum_{l=1}^{n_o} w_l \tau_{iJ}^l(t)) \eta_{iJ}^{\beta}(t)}{\sum_{u \in N_i^k(t)} (\sum_{l=1}^{n_o} w_l \tau_{iu}^l(t)) \eta_{iu}^{\beta}(t)} \tag{4.21}$$

Pheromone update was carried out by using two different ants which had discovered the best and the second-best solution generated in the current iteration for each l-th objective. The global pheromone information is updated for each l-th objective according to equation (3.10), as follows:

$$\tau^l_{ij}(t+1) = (1-\rho)\tau^l_{ij}(t) + \rho\Delta\tau^l_{ij}(t) \tag{4.22}$$

where $\Delta\tau^l_{ij}(t)$ has the following values:

$$\Delta\tau^l_{ij}(t) = \begin{cases} 15 & \text{if edge } (i,j) \in \text{ best and second-best solutions,} \\ 10 & \text{if edge } (i,j) \in \text{ best solution,} \\ 5 & \text{if edge } (i,j) \in \text{ second-best solution,} \\ 0 & \text{otherwise} \end{cases} \tag{4.23}$$

4.5.5 Single Colony, Multi-Pheromone, Multi-Heuristic Matrix Methods

Iredi *et al.* [104] adapted the AS for two objectives by including two pheromone matrices (τ and $\tau^{'}$) and two heuristic matrices (η and $\eta^{'}$) – one for each objective. The AS transition rule (refer to equation (3.3)) is changed to

$$p^k_{ij}(t) = \begin{cases} \dfrac{\tau_{ij}^{\alpha\lambda_k}(t)\tau_{ij}^{'\alpha(1-\lambda_k)}(t)\eta_{ij}^{\beta\lambda_k}(t)\eta_{ij}^{'\beta(1-\lambda_k)}(t)}{\sum_{h\in N^k_i(t)}\tau_{ih}^{\alpha\lambda_k}(t)\tau_{ih}^{'\alpha(1-\lambda_k)}(t)\eta_{ih}^{\beta\lambda_k}(t)\eta_{ih}^{'\beta(1-\lambda_k)}(t)} & \text{if} \quad j \in N^k_i(t) \\ 0 & \text{otherwise} \end{cases} \tag{4.24}$$

where λ_k is different for each ant k, in order to force the ants to search in different regions of the Pareto front; λ_k is calculated as the ratio of the ant index to the total number of ants.

Every ant that generates a solution in the non-dominated front for the current iteration is allowed to update both pheromone matrices, τ and $\tau^{'}$, by depositing an amount equal to $\frac{1}{n_p}$, where n_p is the number of ants that constructed a non-dominated solution. Then, all non-dominated solutions for the current iteration are added to an external archive and this archive is sorted to remove any dominated solutions. At the end of the algorithm execution, the external archive is returned as the final set of solutions.

Mora *et al.* [148] adapted the ACS to use two heuristic matrices and two pheromone matrices. The algorithm referred to as hCHAC changes the ACS transition rule to

$$j = \begin{cases} Arg\ Max_{u \in N_i^k(t)} \{\tau_{iu}^{\alpha\lambda_k}(t)\tau_{iu}^{'\alpha(1-\lambda_k)}(t)\eta_{iu}^{\beta\lambda_k}(t)\eta_{iu}^{'\beta(1-\lambda_k)}(t)\} & \text{if } r \leq r_0, \\ J & \text{otherwise} \end{cases} \tag{4.25}$$

where $J \in N_i^k(t)$ is a node that is randomly selected according to probability,

$$p_{ij}^k(t) = \begin{cases} \frac{\tau_{ij}^{\alpha\lambda_k}(t)\tau_{ij}^{'\alpha(1-\lambda_k)}(t)\eta_{ij}^{\beta\lambda_k}(t)\eta_{ij}^{'\beta(1-\lambda_k)}(t)}{\sum_{h \in N_i^k(t)} \tau_{ih}^{\alpha\lambda_k}(t)\tau_{ih}^{'\alpha(1-\lambda_k)}(t)\eta_{ih}^{\beta\lambda_k}(t)\eta_{ih}^{'\beta(1-\lambda_k)}(t)} & \text{if} \quad j \in N_i^k(t) \\ 0 & \text{otherwise} \end{cases} \tag{4.26}$$

where λ_k is calculated as the ratio of the ant index to the total number of ants.

The local pheromone update is the same as for the original ACS, with different initial values τ_0 and $\tau_0^{'}$ for each objective, initially calculated as follows:

$$\tau_0 = \frac{1}{N_G f^1(\mathbf{T}^{worst})} \tag{4.27}$$

$$\tau_0^{'} = \frac{1}{N_G f^2(\mathbf{T}^{worst})} \tag{4.28}$$

where $\mathbf{T}^{worst}$ is the worst solution heuristically obtained prior to the execution of the ant algorithm and f^1 and f^2 are the objective functions.

Every ant k that generates a solution, T_k, in the non-dominated front for the current iteration is allowed to update both pheromone matrices, τ and $\tau^{'}$, by depositing an amount equal to $\frac{1}{f^1(\mathbf{T}_k)}$ and $\frac{1}{f^2(\mathbf{T}_k)}$ respectively. Then, all non-dominated solutions for the current iteration are added to an external archive and this archive is sorted to remove any dominated solutions. At the end of the algorithm execution, the external archive is returned as the final solutions.

4.5.6 Multi-Colony MOACO Algorithms

The first implementations of ACO algorithms made use of only one colony of ants to construct solutions. These algorithms have been adapted to use multiple colonies [76, 104, 142, 146]. One of the first applications of multiple colony ACO algorithms was to solve MOP. This section discusses such algorithms.

MOPs are solved by assigning to each colony the responsibility for optimising one of the objectives. Each colony is independent of the other colonies in the sense that it has its own ants and its own pheromone information to the extent that, when an ant from a certain colony constructs the ant's solution, the ant is guided by the pheromone information from the ant's own colony only. After every iteration of the ant algorithm for each colony, the colony computes the new pheromone information.

Three aspects define the behaviour of the multiple colonies:

- The set of weight vectors which are used to aggregate multiple pheromone information.

  The image of the optimal Pareto set in the objective space is a trade-off surface between the different objectives. Over this surface, two solutions can be said to belong to different regions with respect to the differences in the objective vectors of the two solutions, for example, if the distance between their respective objective vectors is more than a given value.

  The use of single pheromone information does not in itself force each colony to focus on a certain region. However, when multiple pheromone information is used, the set of weight vectors that each colony uses in order to aggregate its multiple pheromone information defines in some way a region in the objective space on which the colony focuses the search. Choosing the set of weights forces each colony to approximate a different region of the optimal Pareto set.

  The infinite set of weights defines all the possible directions that can be taken to approximate the optimal Pareto set. Any finite subset Γ of maximally dispersed weight vectors defines a region for the entire optimal Pareto set. A partition of

Γ defines regions in the optimal Pareto set that can be either disjoint or overlapping regions depending on whether disjoint partitions of Γ are considered or not. Then, the multiple colonies can use i) the same partition of Γ, ii) disjoint, or iii) overlapping partitions of Γ.

As for the single colony algorithm, in multi-colony algorithm the ants in a colony use different λ-values. That is, when making decisions ants weight the relative importance of each optimisation objective differently. Given the number of colonies n_c and the number of ants per colony, n_k/n_c, Iredi *et al.* [104] proposed the following possibilities to define each weight value λ_k for each ant k, where $k \in [1, n_k/n_c]$:

– Single region: for all colonies the values of λ_k are in the interval $[0, 1]$, computed as
$$\lambda_k = \frac{k-1}{n_k/n_c - 1} \tag{4.29}$$
An alternative could be to use different λ_k-values in the colonies so that λ_k-values of the ants in the colonies are in different subintervals of $[0, 1]$. Thus, the colonies weight the optimisation sub-objectives differently.

– Disjoint regions: each colony, c, have distinct λ_k-values, computed as
$$\lambda_k = (c-1)n_k/n_c + k \tag{4.30}$$

– 50% overlapping regions: the interval of values of λ_k for colony c overlaps by 50% with the interval for colony $c-1$ and colony $c+1$. Colony c has ants with
$$\lambda_k \in \left[\frac{c-1}{n_c+1}, \frac{c+1}{n_c+1}\right] \tag{4.31}$$

• The pheromone update strategy.

In order to enforce specialisation of the colonies, each ant deposits pheromone on one colony only. The pheromone update strategies described for the single colony approach may also be applied to multiple colonies. The selection by dominance method is adapted straightforwardly to the multi-colony approach. That is, the ants belonging to the Pareto set of the candidate set are distributed between colonies and are allowed to deposit pheromone. In selection by objective, the Pareto set of the candidate set is, somehow, distributed among the colonies. Then, for each colony, the best solution in respect of each objective is allowed to deposit pheromone.

The pheromone update strategy must select the colony from which each solution updates the pheromone information.

Iredi *et al.* [104] presented two different methods to determine in which colony an ant should update the pheromone matrix:

1. Method 1 – Update by origin: an ant only updates the pheromone matrices in its own colony. Using this method other colonies help to detect which of the solutions in the local non-dominated front of a colony might be dominated. The update by origin method enforces both colonies to search in different regions of the non-dominated front.
2. Method 2 – Update by region: the sequence of solutions along the non-dominated front is split into n_c parts of equal size. Ants that have found solutions in the c-th part update in the colony c, $c \in [1, n_c]$. This method may be used with bi-objective problems only, because for more than two objectives, a set of non-dominated objective vectors may just be partially sorted.

- Cooperation between colonies.

  Cooperation is achieved by colonies exchanging solutions so that the pheromone updates of one colony are influenced by solutions from other colonies. Another alternative is to form the candidate set – from which the best solutions have been selected in order to update the pheromone information – with solutions from all colonies.

  If no cooperation takes place between colonies, then a reasonable way for pheromone update in the multi colony algorithm is that only those ants that found a solution which is in the local non-dominated front of the colony, update the colony's pheromone information. Therefore the results are the same as with a multi-start approach where a single colony ant algorithm is run several times and the global non-dominated front at the end is determined from the non-dominated fronts of all runs. This approach significantly increases the processing speed and decreases the number of evaluations for each iteration.

> Middendorf *et al.* [146] demonstrated that if a high solution quality is required or the overall performance of the algorithm needs to be increased, information exchange between the colonies is important. Information exchange allows the colonies to profit from the good solutions found by other colonies. Information exchange also allows colonies to search in different regions of the search space by using different pheromone matrices, thus, improving diversity.
>
> To balance exploration against exploitation, the colonies cooperate with a given communication policy specifying the details of what kind of information to exchange, when to exchange it, and among which colonies.

In the remainder of this section multi-colony methods for multi-objective optimisation are described.

Mariano and Morales [142] proposed a multi-colony ACO approach where one colony of ants exists for each objective. Mariano and Morales studied a problem in which every objective was influenced by parts of a solution only, so that an ant from colony c received a (partial) solution from an ant from colony $c-1$ and then tried either to improve or to extend this solution with respect to the c-th sub-objective. A final solution that had passed through all the colonies was allowed to update the pheromone information when it formed part of the non-dominated front.

Gambardella *et al.* [76] adapted the ACS for two objectives by defining two ant colonies each dedicated to the optimisation of a different objective function. Each colony maintains its own pheromone matrix, initialised to have a bias towards an initial solution. A local heuristic is first used to obtain the initial solution TL^{gb}, which is then improved by the two colonies, each with respect to a different objective: TL^{gb} is updated each time one of the colonies computes an improved feasible solution and represents the best path globally that has been found from the beginning of the trial. The colonies cooperate by exchanging TL^{gb} which is used for global pheromone updating. The pheromone global update is performed with TL^{gb} using equation (3.10).

Both previous approaches used a lexicographical order to decide the order of importance of each objective. That is, no two objectives could be assigned the same importance.

Iredi *et al.* [104] proposed an approach for bi-criteria optimisation based on multiple ant colonies without considering a lexicographical order. The ant algorithm for each colony adapted the AS for two objectives by including two pheromone matrices (τ and

$\tau^{'}$) and two heuristic matrices (η and $\eta^{'}$) – one for each objective (refer to Section 4.5.5). Each colony specialised in finding satisfactory non-dominated solutions in different parts of the Pareto front.

In order to achieve collaboration between the colonies, the ants within an iteration place their solutions into a global solution pool that is shared by the other colonies. The pool is used to determine the non-dominated front of all the solutions pertaining to that iteration. Subsequently, only those ants that had found a solution which was in the global, non-dominated front are allowed to update the pheromone information. Cooperation was activated after each iteration at which point all colonies had found a solution.

Iredi *et al.* [104] showed that cooperation between the colonies permits the finding of good solutions along the whole Pareto front. Heterogeneous colonies were used in which the ants have different preferences as regards the sub-objectives when constructing a solution. This choice of heterogeneous colonies has a considerable impact on the performance of the algorithms.

Alaya *et al.* [6] proposed a multi-colony ant algorithm to solve a multi-objective optimisation problem with any number n_o of objectives. The proposed algorithm uses $n_o + 1$ ant colonies and n_o pheromone matrices. Each of the l-th colonies ($l \in [1, ..., n_o]$) aims at optimising the l-th objective function and uses one pheromone matrix, τ^l, and one heuristic information function, η^l, defined with respect to the l-th objective. The $(n_o + 1)$-th colony considers, at each construction step, a randomly chosen objective to optimise. The pheromone matrix, τ^{n_o+1}, considered by the (n_o+1)-th colony is the same as the pheromone matrix of the l-th objective function, where $l \in [1, ..., n_o]$ is randomly chosen. The heuristic information η^{n_o+1} considered by the $(n_o + 1)$-th colony is the sum of heuristic informations associated with all objectives, i.e. $\eta^{n_o+1} = \sum_{l=1}^{n_o} \eta^l$.

The ant algorithm for each colony is based on the MAX-MIN ant algorithm. The pheromone update is as follows: For each of the first n_o colonies, pheromone is laid on the components of the best solution, $\mathbf{T}_l^{ib}$, found by the l-th colony during the current iteration, where the quality of solutions is evaluated with respect to the l-th objective, f^l, only.

The quantity, $\Delta\tau_{ij}^l$, of pheromone deposited on each link (i, j) used by the $\mathbf{T}_l^{ib}$ solution, for the l-th pheromone matrix is defined as follows

$$\Delta\tau_{ij}^{l} = \frac{1}{(1 + f^{l}(\mathbf{T}_{l}^{ib}) - f^{l}(\mathbf{T}_{l}^{gb}))} \tag{4.32}$$

where $\mathbf{T}_{l}^{gb}$ is the global best solution since the beginning of the run considering the l-th objective function.

The (n_o+1)-th colony maintains a set of solutions: a best solution for each objective. The $(n_o + 1)$-th colony lays pheromone on each pheromone structure relative to the correspondent objective using equation (4.32).

4.5.7 Summary

This section examined several approaches in which ACO algorithms have been adapted to solve MOPs. Five different approaches have been discussed, namely, the single colony, single-pheromone, single-heuristic matrix, the single colony, single-pheromone, multi-heuristic matrix, the single colony, multi-pheromone, single-heuristic matrix, the single colony, multi-pheromone, multi-heuristic matrix, and the multi-colony approach.

4.6 Evolutionary Multi-Objective Optimisation

Over the past decade, a number of multi-objective evolutionary algorithms (MOEAs) have been suggested [16, 40, 167, 224]. This section briefly describes evolutionary algorithms (EAs) in Subsection 4.6.1 and the elitist non-dominated sorting genetic algorithm (NSGA-II) in Subsection 4.6.2.

4.6.1 Evolutionary Algorithms

EAs are a class of stochastic optimisation methods that simulate the process of natural evolution [224]. Several classes of EAs have been developed, including genetic algorithms, evolutionary programming, and evolution strategies [16]. All of these approaches maintain a population of candidate solutions. During the optimisation process, these candidate solutions are changed through application of selection and variation operators. While selection mimics competition for reproduction and resources among living beings, variation imitates the natural capability of creating “new” living beings by means of recombination and mutation. Although the underlying mechanisms are simple, EAs have proved to be a general, robust and powerful search mechanism [16]. In particular, EAs have been successful in solving optimisation problems with multiple conflicting objectives [41] and intractable, large and complex search spaces [223].

In EA terminology, candidate solutions are referred to as individuals or chromosomes. The set of candidate solutions is referred to as a population.

Algorithm 7 General Scheme of an Evolutionary Algorithm

```
Initialise population $P$ with random candidate solutions;
while not terminating condition do
    Evaluate each candidate from $P$;
    Select parents;
    Recombine pair of parents;
    Mutate the resulting offspring;
    Select individuals for the next generation and insert them into $P'$;
    $P \leftarrow P'$;
end while
Return $P$;
```

A generic EA is summarised in Algorithm 7. An EA consists of the following steps: Firstly, an initial population is created at random, and this constitutes the starting point of the evolution process. A loop consisting of the following steps – evaluation (fitness assignment), selection, recombination, and/or mutation – is then executed a certain number of times. Each loop iteration is termed a generation, and a predefined maximum number of generations often serves as the termination criterion of the loop. However, other conditions, for example stagnation in the population or the existence of an individual with sufficient quality, may also be used to terminate the simulation. In the end the best individuals in the final population represent the outcome of the EA. The different steps of the loop are discussed next:

- Evaluation function (fitness function). The evaluation function is a function or procedure that assigns a quality measure to individuals.

- Parent selection mechanism. The role of parent selection or mating selection is to distinguish between individuals based on their quality, in particular, to allow the better individuals to become parents of the next generation. An individual is a parent if it has been selected to undergo variation in order to create offspring.

- Recombination. A binary variation operator is called recombination or crossover. A recombination operator merges information from two parent individuals into one or two offspring individuals.

- Mutation. A unary variation operator is commonly called mutation. A mutation operator is applied to one individual and delivers a (slightly) modified mutant, its child or offspring. A mutation operator is always stochastic: its output - the child - depends on the outcomes of a series of random choices.

- Survivor selection mechanism. The role of survivor selection or environmental selection is to distinguish among individuals from the offspring based on their quality. The fittest will be allowed in the next generation.

4.6.2 Elitist Non-Dominated Sorting Genetic Algorithm

Since this thesis will compare the performance of MOACO algorithms to the performance of the elitist non-dominated sorting genetic algorithm (NSGA-II) on the multi-objective power aware routing problem, this section provides an overview of NSGA-II.

The NSGA-II [42] is one of the most efficient multi-objective evolutionary algorithms using an elitist approach. The fitness assignment of NSGA-II consists of sorting the population on different fronts using the non-domination order relation. Initially, a random parent population P of size N_p is created. The population is sorted on the basis of non-domination. Each solution is assigned a fitness equal to its non-domination level (1 is the best level). Thus, minimisation of fitness is assumed. Binary tournament selection, recombination, and mutation operators are used to create a child population of size N_p. From the first generation onward, the procedure differs, consisting of the main loop:

The parent and child population are combined into the population R_t. R_t is sorted into non-dominated fronts. A new population $P^{'}$ is generated starting from the first non-dominated front until N_p individuals are found. The **crowded comparison operator** is used for the selection process: Between two solutions with differing non-domination fronts, the solution from the lower non-dominated front (with lower rank) is selected. Otherwise, if both the solutions belong to the same non-dominated front then the solution which is located in a region with the lesser number of solutions is included. From $P^{'}$, the child (offspring) population is generated with the standard bimodal crossover and polynomial operators.

At the end of the execution of the algorithm, the best individuals in terms of non-dominance and diversity are chosen.

The NSGA-II consists of the following different modules:

1. **Finding the non-dominated front.** This module does a quick sorting on the solution space obtained after combining the parent and child population, and extracts the set of non-dominated solutions (non-dominated front).

   Algorithm 8 summarises the steps to find the set of non-dominated solutions from a set of solutions.

Algorithm 8 Procedure to Find the Set of Non-Dominated Solutions (Find-Non-Dominated-Front)

```
Input parameter $P$; {Population from which to extract the non-dominated front}
$P' = \emptyset$;
for all $p \in P \wedge p \notin P'$ do
  $P' = P' \cup \{p\}$; {include $p$ in $P'$ temporarily }
  {compare $p$ with other members of $P'$}
  for all $q \in P' \wedge q \neq p$ do
    if $p \prec q$ then
      $P' = P' \setminus \{q\}$; {if $p$ dominates a member $q$ of $P'$, delete $q$ }
    else
      if $q \prec p$ then
        $P' = P' \setminus \{p\}$; {if $p$ is dominated by other members of $P'$, do not include $p$
        in $P'$}
      end if
    end if
  end for
end for
Return $P'$;
```

2. **A fast non-dominated sorting approach.** This module strips out the non-dominated fronts one by one from the solution space and then ranks these non-dominated fronts. The solutions belonging to the first non-dominated front are given a rank of 1, and those belonging to the next non-dominated front a rank of 2, and so on. This way the set of non-dominated fronts is extracted from the population to be sorted.

   Algorithm 9 summarises the steps for finding the set of non-dominated fronts.

Algorithm 9 Procedure to Find the Set of Non-Dominated Fronts (Non-Dominated-Sort)

```
Input parameter $P$; {Population for which to find the set of non-dominated fronts}
$\mathcal{Z} = \emptyset$; {$\mathcal{Z}$ is the set of non-dominated fronts}
$c = 1$; {$c$ is the non-dominated front counter and is initialized to one}
while $P \neq \emptyset$ do
  $\mathcal{Z}_c$ =Find-non-dominated-front($P$); {Find the $c$-th non-dominated front}
  $P = P \setminus \mathcal{Z}_c$; {remove non-dominated solutions from $P$};
  $\mathcal{Z} = \mathcal{Z} \cup \mathcal{Z}_c$;
  $c = c + 1$;
end while
Return $\mathcal{Z}$;
```

3. **Density estimation.** It is very important to keep the solution points well spread out. Therefore, efficient measures are required for controlling the crowding (density) in one region. In order to obtain an estimate of the density of the solutions surrounding a particular solution in the population, the average distance of two points on either side of this point along each of the objectives is calculated. This distance is termed the crowding distance.

   The crowding distance computation requires that the population be sorted according to each objective function value in ascending order of magnitude. Thereafter, for each objective function, the boundary solutions (solutions with the smallest and the largest function values) are assigned an infinite distance value. All other intermediate solutions are assigned a distance value equal to the absolute difference in the function values of two adjacent solutions. This calculation is continued in terms of other objective functions. The overall crowding distance value is calculated as the sum of the individual distance values corresponding to each objective. The following crowding-distance-assignment procedure (Algorithm 10) outlines the crowding distance computation procedure for all solutions in a non-dominated set $\mathcal{I}$.

   After all population members in set $\mathcal{I}$ have been assigned a distance metric then any two solutions are compared for the extent of their proximity to other solutions. A solution with a smaller value of this distance measure is, in some sense, more crowded than other solutions. This is exactly what is compared in the crowded comparison operator which is described below.

Algorithm 10 General Procedure of Crowding-distance-assignment

```
Input parameter $\mathcal{I}$; {A non-dominated set}
$n_{si} = |\mathcal{I}|$; {number of solutions in $\mathcal{I}$}
**for all** $i \in 1, ..., n_{si}$ **do**
    $\mathcal{I}[i].distance = 0$; {initialise distance}
**end for**
{$n_o$ number of objectives}
**for all** $j \in 1, ..., n_o$ **do**
    $\mathcal{I} = sort(\mathcal{I}, j)$; {sort using each $j$-th objective value}
    $\mathcal{I}[1].distance = \infty$;
    $\mathcal{I}[n_{si}].distance = \infty$;
    **for** $i = 2\ To\ n_{si} - 1$ **do**
        $\mathcal{I}[i].distance = \mathcal{I}[i].distance + (\mathcal{I}[i+1].j - \mathcal{I}[i-1].j)$; {$\mathcal{I}[i].j$ refers to the $j$-th
        objective value of the $i$-individual in the set $\mathcal{I}$}
    **end for**
**end for**
Return $\mathcal{I}$;
```

4. **Crowded comparison operator.** The crowded comparison operator $\prec_n$ guides the selection process at the various stages of the algorithm towards a uniformly spread-out Pareto-optimal front. The assumption is made that every individual, $\mathcal{I}_i$, in the population possesses two attributes: 1) non-domination rank $\mathcal{I}_{i_{rank}}$ which is based on the non-domination front, and 2) crowding distance $\mathcal{I}_{i_{distance}}$.

   The crowded comparison operator $\prec_n$ is defined as:

$$\mathcal{I}_i \prec_n \mathcal{I}_j \text{ if } (\mathcal{I}_{i_{rank}} < \mathcal{I}_{j_{rank}}) \text{ or } ((\mathcal{I}_{i_{rank}} = \mathcal{I}_{j_{rank}}) \text{ and } (\mathcal{I}_{i_{distance}} > \mathcal{I}_{j_{distance}}) \quad (4.33)$$

   In other words, in terms of two solutions with differing non-domination ranks the solution with the lower (better) rank is preferred. Otherwise, if both solutions belong to the same front then the solution which is located in a lesser crowded region is preferred.

Using the above procedures – a fast, non-dominated sorting procedure, a fast, crowded distance estimation procedure and a simple crowded comparison operator – the NSGA-II algorithm is summarised in Algorithm 11.

The NSGA-II is a genetic algorithm with $O(n_o N_p^2)$ computational complexity (where n_o denotes the number of objectives and N_p the population size). As a result of the low computational requirements of NSGA-II, its elitist approach and its parameterless sharing scheme, the NSGA-II was selected as the algorithm to which the five algorithms presented in this thesis will be compared.

Algorithm 11 General Procedure of NSGA-II

```
Create a random population $P_0$;
$\mathcal{Z}$ = non-dominated-sort($P_0$);
Use binary tournament selection, recombination, and mutation operators to create a child population $Q_0$ of size $N_p$.
$t = 0$;
while $t < N_{maxgen}$ do
    $R_t = P_t \cup Q_t$; {combine parent and children population}
    $\mathcal{Z}$ = non-dominated-sort($R_t$); {$\mathcal{Z} = (\mathcal{Z}_1, \mathcal{Z}_2, ...)$, all non-dominated fronts of $R_t$}
    $P_{t+1} = \emptyset$;
    $c = 1$;
    {till the parent population is filled}
    while $|P_{t+1}| + |\mathcal{Z}_c| <= N_p$ do
        crowding-distance-assignment($\mathcal{Z}_c$); {calculate crowding distance in $\mathcal{Z}_c$};
        $P_{t+1} = P_{t+1} \cup \mathcal{Z}_c$; {include $c$-th non-dominated front in the parent population}
        $c = c + 1$; {check the next front for inclusion}
    end while
    Sort($\mathcal{Z}_c, \prec_n$); {sort in descending order using $\prec_n$}
    $P_{t+1} = P_{t+1} \cup \mathcal{Z}_c[1 : (N_p - |P_{t+1}|)]$; {Choose the first $(N_p - |P_{t+1}|)$ elements of $\mathcal{Z}_c$ }
    $Q_{t+1}$ = make-new-pop($P_{t+1}$); {Use selection, recombination, and mutation operators to create a child population $Q_{t+1}$}
    $t = t + 1$; {increment the generation counter}
end while
Return $P_t$;
```

4.7 Performance Metrics for Multi-Objective Optimisation

It is not an easy task to compare the performance of multi-objective algorithms [22]. In multi-objective optimisation the performance metric must assess a number of solutions, each having a vector of objective values. Performance metrics are hard to define and more than one metric is necessary to evaluate the performance of multi-objective algorithms.

There are several criteria for quantifying the quality of the approximation to the

Pareto front. This section examines a number of performance criteria for MOO. The rest of the section is organised as follows: Section 4.7.1 provides an overview of the goals of multi-objective optimisation, while Section 4.7.2 discusses the perfomance metrics used in this thesis.

4.7.1 Multi-Objective Optimisation Goals

It is relatively simple to define performance metrics for single objective optimisation. However, MOPs do not have a single global optimum solution, but rather a number of optimum solutions that represent a trade-off between the various sub-objectives. The overall aim in MOO is to produce a set of solutions that represent a good approximation to the trade-off surface. A good approximation set should be as close as possible to the true Pareto front and should also provide a good coverage of the true Pareto front. The goal of achieving a good coverage of the trade-off surface, i.e. to maintain diversity and spread of solutions, is of particular interest in multi-objective optimisation.

It is difficult to define a single measure that can be used to quantify the quality of solutions obtained by a multi-objective algorithm (MOA). Instead, MOA performance is characterised using a number of different aspects [37, 208, 227]. This thesis focuses on two aspects of MOA performance:

- Closeness to the true Pareto front, as the distance between the non-dominated set obtained and the true Pareto front.
- Diversity of solutions in the Pareto front.

4.7.2 Performance Metrics

Several criteria have been developed to assess the quality of a MOO algorithm and to compare such algorithms [95, 120, 188]. Some of the criteria require knowledge of the true Pareto-optimal solutions, which are unknown in the optimisation problem presented in this thesis. Taking this limitation into account, the following three criteria are selected: 1) size of the dominated space, 2) spread metric, and 3) the size of the approximated Pareto front (number of non-dominated solutions). The first metric measures the closeness of the obtained non-dominated set to the true Pareto front. The second metric measures the spread of solutions along the non-dominated set obtained, while the last metric measures the size of the non-dominated set. These three metrics, do not require knowledge of the true Pareto front.

The above three metrics are described below:

1. **Size of the dominated space or hypervolume measure** (S_d):

   The size of the dominated space is a measure of how much of the objective space is weakly dominated by a given non-dominated set [17, 126, 218]. Consider the non-dominated set $P = \{\mathbf{p_1}, \mathbf{p_2}, ..., \mathbf{p_l}\}$. The size of the space dominated by the set P, denoted by $S_d(P)$, is defined as the volume of the union of hypercubes $\{C_1, ..., C_l\}$, where C_i is a hybercube whose two opposite vertices are $\mathbf{p_i}$ and the origin of the objective space. Since the optimisation problem in this thesis involves the minimisation of five objectives, a reasonable maximum value for each objective is selected for the origin of the objective space. The values of 100.0, 0.1, 500.0, 0.5, and 30.0, corresponding to a maximum value for each of the objectives EP, TNP, VF, CP, and MNC have been selected as the origin of the objective space. These values lead to a maximum hypervolume of 75000. The hypervolume metric measures how well the algorithms performed in identifying solutions along the full extent of the Pareto front. Higher values of $S_d(P)$ indicate more closeness to the true Pareto front and better performance.

2. **Spacing** (SP) or spread metric [179, 208]: The SP metric is used to measure the spread (distribution) of vectors in the current non-dominated set. Schott [179] proposed an SP metric to measure the range (distance) variance of neighbouring vectors in the non-dominated set. The SP is defined as

   $$SP = \sqrt{\frac{1}{n_{\mathcal{PF}}} \sum_{i=1}^{n_{\mathcal{PF}}} (\bar{r} - r_i)^2} \tag{4.34}$$

   where $r_i = \min_{j=1,...,n_{\mathcal{PF}}} \sum_{m=1}^{n_o} |f_m^i - f_m^j|$; f_m is the m-th objective function, $\bar{r}$ is the mean of all r_i, $n_{\mathcal{PF}}$ is the number of non-dominated solutions, and n_o is the number of objectives. The smaller the value of SP, the better the distribution in the current non-dominated set. A value of zero indicates that all members of the current Pareto front are equidistantly spaced.

If an MOP has a $\mathcal{PF}$ which is composed of two or more Pareto curves then the distance between the end-points of two successive curves may skew this metric. Therefore, for this kind of Pareto sets, the distance corresponding to the breaks should be removed from the spacing computation.

It should be noted that the objective values should be normalised before calculating the distance. Various normalisation schemes have been proposed in the literature [103, 121].

3. **Number of non-dominated solutions found (ND)** [119, 208]:

   This performance measure quantifies the size of the approximated Pareto front, that is, the number of non-dominated solutions. The ND metric is defined as

$$ND(\mathcal{PF}) = |\mathcal{PF}| \tag{4.35}$$

   The size of the approximated Pareto front may also be calculated with respect to the solution vectors in the decision space.

   The $\bar{n}_{alg}$ metric measures how well the algorithms performed in identifying solutions along the Pareto front. Larger values for $\bar{n}_{alg}$ are preferred as it indicates that many efficient solutions were found which is preferred by the decision maker. The maximum value for $\bar{n}_{alg}$ is 100 which is the size of the archive.

   Although counting the number of non-dominated solutions does provide an indication of the effectiveness of the MOO algorithm in generating desired solutions, ND does not reflect on the distance of $\mathcal{PF}^*$ from these non-dominated solutions. Also, it is not possible to draw any conclusions about any dominance relation between two approximation sets [228].

The three performance metrics are chosen because they address the main functional goals of MOO algorithms (closeness to the true Pareto front and diversity of solution in the $\mathcal{PF}$). This set of three metrics will enable two or more non-dominated solution sets to be compared among each other in terms of their functional achievements. Also, these three metrics are unary and do not require knowledge of the true Pareto front which is uknown in the power aware optimization problem presented in this thesis.

Moreover, the hypervolume indicator is the only unary quality measure that is known to be strictly monotonic with regard to Pareto dominance: whenever a Pareto set approximation entirely dominates another one, then the indicator value of the former will also be better. This property is of high interest and relevance for the problem examined in this thesis which involves a large number of objective functions. The spacing metric has a low computational overhead, and can be used with more than two objectives. A high number of non-dominated solutions, i.e. the cardinality of a non-dominated set, and therefore more route choices is also desired for power aware dynamic optimization problems where it is difficult to obtain many different non-dominated solutions.

4.8 Summary

The main objective of this chapter was to review multi-objective optimisation theory and algorithms used in this thesis, with specific reference to ACO algorithms and the NSGA-II. Different adaptations of ACO algorithms to solve MOPs have been discussed in detail, and a compact description of the NSGA-II was given. Performance metrics for MOO used in this thesis were discussed.

The next chapter discusses ACO optimisation methods for dynamic environments.