Chapter 5

ACO in Dynamic Optimisation Problems

Ant colony optimisation has proved suitable to solve static optimisation problems, that is problems where the objective does not change with time [52, 54]. However, many real-world problems are defined for dynamic environments, where a previously optimum solution may become sub-optimal and new optima may appear.

Dynamic optimisation problems form a class of difficult optimisation problems. Optimisation algorithms applied to dynamic environments must be able to find and track solutions as the environment changes. In this regard, it should be possible to track both the position and value of optima as changes occur, and it should be possible to detect any new optima that appear and those that disappear. Changes in environments may take various forms, such as changes in the objective functions and/or problem constraints. An acceptable solution at a particular point in time may not be acceptable after a change in the environment has occurred.

The power-aware routing problem considered in this thesis is a dynamic optimisation problem: A realistic mobility model is used within MANET, the position of the nodes changes, and both the optimal decision variables (Pareto set) and the optimal objective values (Pareto front) change repeatedly during the optimisation process.

This chapter is organised as follows: Section 5.1 provides a mathematical definition of dynamic optimisation problems. Section 5.2 discusses how ACO algorithms can be adapted for dynamic optimisation problems. Section 5.3 discusses performance metrics for DOP. Section 5.4 describes dynamic multi-objective optimisation (DMOO), while Section 5.5 discusses performance metrics for dynamic multi-objective optimisation problems (DMOP).

5.1 Definition of Dynamic Optimisation Problems

A single-objective dynamic optimisation problem is formally defined as

Definition 5.1.1. Dynamic optimisation problem:

$$\begin{aligned} \text{minimise} \quad & f(\mathbf{x}, \delta(t)), \quad \mathbf{x} = (x_1, ..., x_{n_x}), \delta(t) = (\delta_1(t), ..., \delta_{n_\delta}(t)) \\ \text{subject to} \quad & g_m(\mathbf{x}, \delta(t)) \leq 0, \quad m = 1, ..., n_g \\ & h_m(\mathbf{x}, \delta(t)) = 0, \quad m = 1, ..., n_h \\ & \mathbf{x} \in \mathbb{R}^{n_x} \end{aligned} \tag{5.1}$$

where $\delta(t)$ is a vector of time-dependent objective function control parameters, n_δ is the number of objective function control parameters, and g_m and h_m denote the inequality and equality constraints respectively. The objective is to find and track

$$\mathbf{x}^*(t) = \min_{\mathbf{x}} f(\mathbf{x}, \delta(t)) \tag{5.2}$$

where the solution $\mathbf{x}^*(t)$ is the optimum found at time step t.

Therefore, the task of a dynamic optimisation algorithm is to locate the optimum and track its trajectory as closely as possible and also to search for new optima that may appear. Algorithms should have the ability to track changes in both the position of $\mathbf{x}^*(t)$ and the value of the optimum, $f(\mathbf{x}^*(t))$.

Dynamic environments can be classified into the following classes of problems [61, 64, 68]:

- the location of the optimum changes,
- the location of the optimum remains the same, but its value changes, and
- both the location and value of the optimum change simultaneously.

The difficulty of a dynamic optimisation problem is determined by the frequency, the severity and the predictability of environment change [23]:

- **The frequency of change** determines how often the environment changes, usually in terms of the number of the iterations between each change. As frequency of change increases, the time available for adaptation becomes shorter and the optimisation task becomes more difficult.

- **The severity of change** determines the amount of displacement of the current location of the optimum. Large displacements make the problem more difficult.

- **The predictability of change** defines the pattern of the change, which can be linear, random or cyclic.

5.2 ACO Algorithms and Dynamic Environments

ACO was developed for static environments in which ants, as a result of the autocatalytic feedback process, converge on a single solution. This characteristic of ACO metaheuristics limits their application to static environments. In order for ACO algorithms to be applied to DOPs, mechanisms should be employed that maintain diversity. These mechanisms should find a trade-off between the opposing goals of preserving pheromone information and sufficient resetting of pheromone information to allow the ants to continuously explore the search space.

ACO algorithms for dynamic environments can be classified into the following approaches:

- **Re-initialisation methods.** The simplest way to enforce exploration is to restart the ant algorithm after each environment change has occurred. However, a simple restart of the ant algorithm will discard all old information about best paths. As a result, it will take longer to find a solution which may prevent re-convergence to a new optimum under frequent changes.

- **New pheromone update methods.** Changes in the search space render pheromone information inaccurate and inconsistent. Therefore, an alternative approach to restarting the ACO algorithm is to develop new approaches to pheromone updates.

  Under the assumption that the change in the environment is relatively small, it is likely that the new optimum will, in some sense, be related to the old one, and it would probably be beneficial to transfer knowledge in the form of pheromone information from the old optimisation run to the new run. On the other hand, if too much information is transferred and the severity of change is high, the next iteration of the algorithm after the change has occurred may start near a local optimum, and the algorithm may become detained at this local optimum. Thus, a reasonable compromise between these two opposing approaches has to be found.

The rest of this section focuses on such methods.

5.2.1 Re-initialisation Methods

Gambardella *et al.* [77] proposed a complete re-initialisation of all pheromone concentrations when no improvement in the quality of solutions is observed. All pheromone concentrations, τ_{ij}, are re-initialised to $\tau_0 = \frac{1}{QTL^{best}(t)}$, where Q is a parameter and $TL^{best}(t)$ is the tour length of the best solution found so far. While all pheromones are re-initialised, information about the best solution found so far is retained and used to initialise new pheromones on each link. This technique can be applied in dynamic environments when a change in the environment occurs. This diversification mechanism increases exploration while retaining some knowledge from previous environments. Limiting the amount of stored history to the adapted to change elitist solution assists the algorithm to track the optimal solution.

5.2.2 New Pheromone Updates Methods

Stützle [194] proposed that, when stagnation occurs, pheromone values be increased proportionally to the difference between the pheromone value and the largest pheromone value. The increase of pheromone intensity for all links will increase the selection probability for all links. The same increase in pheromone intensity can be applied in dynamic environments when a change in the environment occurs. The relative difference of the pheromone trails will not be very large and, as a consequence, exploration of new paths is increased.

Guntsch and Middendorf [90] proposed three pheromone update rules (strategies) for dynamic environments, namely, the restart strategy, the η-strategy, and the τ-strategy. The strategies distribute a reset-value, $\gamma_i \in [0, 1]$, to each node i. These reset values are used to reinitialise the pheromone values on all links incident to i, as follows:

$$\tau_{ij}(t+1) = (1-\gamma_i)\tau_{ij} + \gamma_i \frac{1}{n_G - 1} \tag{5.3}$$

where n_G is the number of nodes in the representation graph.

The reset-values for each strategy are calculated as follows:

- **Restart strategy:** This strategy is a global pheromone modification strategy which reinitialises all the pheromone values by the same degree. For each node, i,

$$\gamma_i = \lambda_R \tag{5.4}$$

where $\lambda_R \in [0, 1]$ is a strategy-specific parameter.

This strategy acts globally without considering the position of the environment change. Consequently the ant algorithm may need more time to find the optimum. The most extensive resetting of pheromone values should generally be performed in the close vicinity of the changed node. With lower values of λ_R, a trade-off between exploration and exploitation would be achieved. With higher values of λ_R virtually all pheromone information is reset and the ant algorithm needs more time to rediscover a good solution.

- **η-strategy:** The η-strategy uses heuristic-based information to decide to what degree pheromone values are to be equalised on all links incident to a node i. The equalization of the pheromone values to some degree resets the pheromone values and effectively reduces the influence of experience on the decisions an ant makes to build a solution, thus improving diversity. Each node, i, is given a value, γ_i, proportionate to the nearest changed node j (a node which is inserted or deleted), and equalisation is effected on all links incident to node i. The node, i, receives the reset value

$$\gamma_i = \max\{0, d_{ij}^{\eta}\} \tag{5.5}$$

where

$$d_{ij}^{\eta} = 1 - \frac{\bar{\eta}}{\lambda_E . \eta_{ij}} \tag{5.6}$$

with

$$\bar{\eta} = \frac{1}{n_G * (n_G - 1)} \sum_{i=1}^{n_G} \sum_{\substack{k=1 \\ k \neq i}}^{n_G} \eta_{ik} \tag{5.7}$$

and $\lambda_E \in [0, \infty)$ is a strategy-specific parameter.

- **τ-strategy:** This strategy uses a distance measure based on pheromone information to equalise those links which are closer to the changed node to a greater extent than links that are further from the changed node. The reset value is

  $$\gamma_i = \min\{1, \lambda_\tau d_{ij}^\tau\}, \lambda_\tau \in [0, \infty) \tag{5.8}$$

  where

  $$d_{ij}^\tau = \max_{T(i,j)} \left\{ \prod_{(x,y) \in T(i,j)} \frac{\tau_{xy}}{\tau_{max}} \right\} \tag{5.9}$$

  where $T(i,j)$ is the set of all paths from i to j, and τ_{xy} is the pheromone associated with link (x,y).

All three strategies adapt the pheromone information such that exploration is increased, assisting the algorithm in the detection of new optima. Also, there is adequate transfer of knowledge from previous iterations.

Guntsch *et al.* [93] developed a novel ACO algorithm for dynamic environments, where extensive resetting of pheromone values is performed in the vicinity of change (i.e. local pheromone resetting) and an elitist strategy is proposed for use in dynamic environments. This ACO algorithm combines the three strategies discussed above in order to make ant algorithms more suitable for optimisation in dynamic environments.

In a situation where strong local resetting of pheromones in the area of change is necessary, a combination of the global restart strategy with one of the two more locally acting strategies, i.e. with the $\eta-$strategy or $\tau-$strategy, is applied. In the areas where no change occurs, a lower global resetting of the pheromone values is needed to be able to change the best solution found. This combination can be realised by having each of the two combined strategies distribute reset-values, and then choosing the maximum of the two reset-values for each node.

A standard elitist strategy for ant algorithms is that an elitist ant, which represents the best solution found so far, updates the pheromone values in every generation. However, this best solution may no longer represent a feasible solution after an environment change. Instead of forgetting the old best solution, Guntsch *et al.* [93] adapted the old best solution so that it becomes a reasonably good solution after the environment changes. Two steps are followed in order to adapt the old best solution: i) all nodes that were deleted after the environment change are also deleted from the old best solution, effectively connecting the predecessors and successors of the deleted nodes, and ii) the nodes that were added after the environment change are inserted individually into the old best solution at the place where they cause the minimum increase in cost. The solution derived from this process is the new solution of the elitist ant.

Using a combination of the three strategies and the heuristic for keeping a modified elitist ant, better solutions are found for different strategy parameters than is the case with the pure strategies alone. The authors proved empirically that using the above strategies and resetting the information only in the area of change performed best when problem changes occur frequently. The novel ACO algorithm was applied successfully to a combinatorial dynamic TSP.

Eyckelhof and Snoek [67] modified the AS to apply it to DOPs by introducing global shaking: All pheromone values are squashed into a pre-defined range, while preserving the relative pheromone rankings. That is, if $\tau_{ij}(t) > \tau_{i'j'}(t)$ holds before shaking, it also holds after shaking.

Shaking is implemented as follows

$$\tau_{ij} = \tau_0 \left(1 + \log(\frac{\tau_{ij}}{\tau_0}) \right) \tag{5.10}$$

Shaking is done when an environment change has been detected. The shaking procedure changes the ratio between exploitation and exploration and diversifies search when change occurs.

There is a high probability that paths only have to change in the vicinity of the environment change. Therefore, in addition to global shaking, a local shaking operator was developed which operates in the same way as global shaking, but only within a defined radius around the area of change. Local shaking showed good performance under increased change frequencies. The higher the change frequency, the more important it is to preserve some of the pheromone information in order to exploit the solutions in the vicinity of the environment change. The local shake algorithm, by smoothing the part of the pheromone matrix which is close to the environment change, combines exploitation of the current pheromone matrix and biased exploration within the area of change.

Guntsch and Middendorf [91] proposed P-ACO, an ACS-based algorithm which uses a population of previously best solutions to update the pheromone matrix. In the usual implementations of the ACO, pheromone values are the incremental result of all the updates since the start of algorithm execution. In P-ACO, on the other hand, the pheromone patterns at time t are those precisely induced by the current ant population $P(t)$ and not the result of all past updates. That is, after one ant has completed its solution, the solution is either added or included by replacement into the population set according to both deterministic and stochastic criteria (population update strategies) based on quality, population size, or age.

If a solution T_k enters the population, then the values of the pheromone variables associated with T_k are correspondingly increased according to an AS-like rule using equations (3.4)-(3.6). On the other hand, if a solution T_k leaves the population, then the corresponding pheromone is decreased by the same amount it was increased when T_k entered the population. In this way, the pheromone values precisely reflect the solutions belonging to the current population $P(t)$ at iteration t. In addition to this specific mechanism, P-ACO makes use of the ACS's transition rule using equations (3.7) and (3.8) for component selection; however, it does not make any use of online step-by-step pheromone updates, since pheromone updates are subject to the fact that solutions are either entering or leaving the population.

Guntsch and Middendorf [92] applied the P-ACO algorithm for DOPs. When a change occurs, heuristic repair of the solutions of the population is applied: the solutions maintained in the population are modified after a change, such that they can also become good solutions after the environment change. Since pheromone values depend only on the solutions in the current population, using the solutions in the current population to update pheromone information automatically adapts the pheromone information to reflect the environment change. To maximise search efficacy the P-ACO algorithm only inserts the best solution from each iteration into the population, thereby maintaining a small population of elite solutions. The purpose of maintaining a population of solutions is to provide the P-ACO algorithm with a quick way to adjust the pheromone mapping if a change occurs. The P-ACO algorithm has been shown to be more efficient than most ACO algorithms for dynamic combinatorial optimisation [92].

Ramos *et al.* [169] developed distributed pheromone laying over the dynamic environment itself in order to track different optima. The authors show that the self-organised algorithm is able to cope with, and to adapt quickly to unforeseen situations.

5.3 Performance Metrics for Dynamic Optimisation Algorithms

It is far more difficult to quantify the performance of an algorithm on dynamic optimisation problems than on static optimisation problems [19]. The difficulty in quantifying performance in dynamic environments stems from the fact that the global optimum changes over time, resulting in multiple solutions. The ability of the algorithm to respond to these changes over time has to be quantified.

Even though techniques exist to detect environment changes, these techniques are infeasible owing to the additional computational complexity. Morrison [149] provides a summary of performance measures for dynamic environments with respect to evolutionary algorithms:

- Accuracy (Acc): At each environment change, the absolute difference between the value of the best solution of the iteration found just before a change has taken place and the value of the true global optimum before the change is calculated. The average of these differences for all environment changes is then calculated as a measure of performance [207], denoted by Acc, and defined as

$$Acc = \frac{1}{n_c}\sum_{i=1}^{n_c} |f(L^{opt}(t^c)) - f(L^{best}(t^c))| \tag{5.11}$$

where t^c is the iteration just before a change has taken place, n_c is the number of changes of the fitness landscape (environment) during the run, $L^{opt}(t^c)$ is the optimum solution, $L^{best}(t^c)$ is the best solution found in the environment after iteration t^c (just before the environment change), and f is the fitness function. If n_I is the number of iterations between changes then $t^c = n_I - 1$.

The smaller the measured value for Acc the better the result. In particular, a value of 0 for Acc means that the algorithm found the optimum every time before the landscape was changed (i.e. n_I iterations were sufficient to track the optimum).

This measure requires knowledge of the iteration when the environment changed and the true optimum for each change. In real problems, the position of the true global optimum is not always available, and when the environment changes is also not usually known.

- Adaptability (Ada): At each environment change, the average of the absolute difference between the value of the best solution found for each iteration and the value of the optimum before the change is calculated. The average of these differences for all environment changes (over the entire run) is then calculated as a measure of performance [207], denoted by Ada, and defined as

$$Ada = \frac{1}{n_c}\sum_{i=1}^{n_c}\frac{1}{n_I}\sum_{t=0}^{n_I-1}|f(L^{opt}(t^c)) - f(L_i^{best}(t))| \qquad (5.12)$$

  where $L_i^{best}(t)$ is the best solution found in the environment for iteration t for the fitness landscape after the i-th environment change ($i \in [0, n_c - 1]$).

  The smaller the measured value for Ada the better the result. A value of 0 for Ada means that the best solution in the population is the same as the optimum for all iterations, i.e. the optimum is never lost by the algorithm.

  Similar to the first measure, Ada requires knowledge of the iteration when the environment changed and the true optimum for each change.

  Combining the Acc and the Ada measures, the quality of the search process performed by the algorithm can be evaluated. For example, results with low values for Acc and larger values for Ada indicate that the algorithm loses the optimum after a change is made, but the time interval between changes is long enough to recover.

- At each iteration, the average of the Euclidean distance between each solution of the iteration and the global optimum before the change is calculated. The average over these distances for all iterations before a change and for all environment changes is then calculated as a measure of performance [212], denoted by AED, and defined as

  $$AED = \frac{1}{n_c}\sum_{i=1}^{n_c}\frac{1}{n_I}\sum_{t=0}^{n_I-1}\frac{1}{n_{PF}}\sum_{k=1}^{n_{PF}}||L^{opt}(t) - L_k^i(t)|| \tag{5.13}$$

  where n_{PF} is the number of solutions at iteration t, $L_k^i(t)$ is the k-th solution at iteration t for the fitness landscape after the i-th environment change, and $||L^{opt}(t) - L_k^i(t)||$ denotes the Euclidean distance between the solutions $L^{opt}(t)$ and $L_k^i(t)$.

  The smaller the measured value for AED the better the result. A value of 0 for AED means that the algorithm has found the optimum at each iteration.

  Again, the problem with the AED measure is the fact that the position of the global optimum in the search space is usually not available, except in test problems. Another problem is that the Euclidian distance does not apply to all problem spaces.

- Best-of-generation average (BOGA): This is the average of the best solution for each iteration over several executions of the algorithm on the same problem [88]. $BOGA$ for algorithm A is defined as

  $$BOGA_A(t) = \frac{1}{n_r}\sum_{r=1}^{n_r} L_r^{best}(t) \tag{5.14}$$

  where n_r is the number of runs and $L_r^{best}(t)$ is the best solution for iteration t and run r.

  To compare the performance of one algorithm against another, the BOGA metric is calculated for each iteration before a change in the environment occurs (refers to Equation (5.14)).

  Figure 5.1 illustrates the BOGA metric for algorithms A and B. The run of each algorithm consists of 20 iterations and the frequency of change is 5. Therefore, during each run the function changes every 5 iterations, resulting in 4 changes per run.

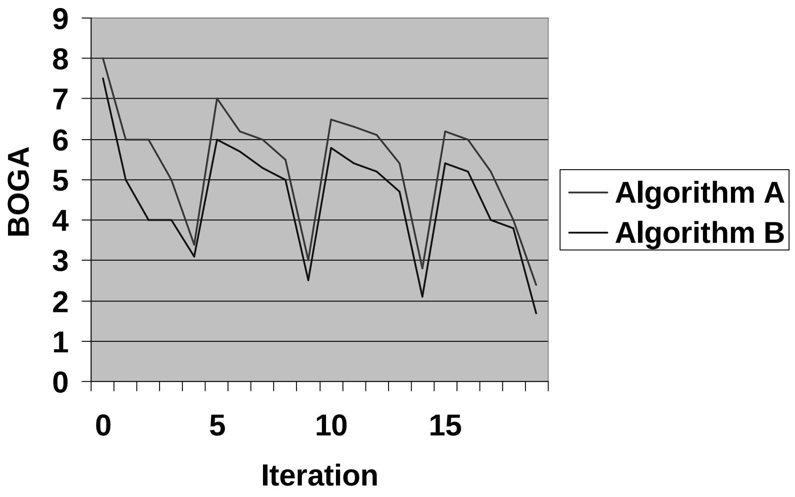

Figure 5.1: Best of generation averages

This method is the most commonly used method to compare the performance of one algorithm against another. However, it does not provide a convenient method for comparing performance across the full range of changes in the environment: Many experiments are required for an accurate measure over all possible changes and using this metric requires the determination of the number of iterations to be used for a representative sample of all the environment changes. In addition, this metric does not provide a convenient method for measuring the statistical significance of the results: it is difficult to determine whether any differences in performance are statistically significant.

- At each iteration, the difference between the value of the best solution of the iteration minus the value of the worst solution within a small window, W, of recent iterations, compared to the value of the best solution within the window minus the value of the worst solution within the window, is calculated as a measure of performance [112], denoted by WA. WA is defined as

$$WA_A(t) = \frac{f(L^{best}(t)) - f(L_W^{worst}(t))}{f(L_W^{best}(t)) - f(L_W^{worst}(t))} \tag{5.15}$$

where $L_W^{best}(t)$ is the best solution within the window $[t - W, ..., t]$, and $L_W^{worst}(t)$ is the worst solution within the window $[t - W, ..., t]$.

This measure is based on the assumption that the best fitness value will not change much over a small number of iterations, which may not be true. This measure also does not provide a convenient method for comparing performance across the full range of changes in the environment.

Since the position of the global optimum in the search space is not available for the DMOP proposed in this thesis, the $BOGA_A(t)$ measure is used to compare the performance of the developed algorithms.

5.4 Dynamic Multi-objective Optimisation

A dynamic multi-objective optimisation problem (DMOP) is a dynamic optimisation problem (DOP) where at least one of the sub-objectives changes over time.

Solving a DMOP consists of tracking changes in the Pareto-optimal front. New dominated solutions may be found and should be added. It may happen that current solutions in the Pareto-front become dominated after a change and these should be removed.

Very little research has been done on DMOO. Most research in DMOO has been done on EAs [43, 68, 89, 106, 132] and PSO [87].

This section will only discuss a dynamic multi-objective optimisation evolutionary algorithm (DMOEA) proposed by Liu and Wang [132], since this thesis will borrow some of the aspects of this method. Liu and Wang proposed a new DMOEA denoted by DMEA. The DMEA algorithm divides the simulation time of the DMOP into several equal time sub-periods. In each sub-period, the DMOP is approximated by a static multi-objective optimisation problem. As a result, the original DMOP is approximately transformed into several static multi-objective optimisation problems. The comparative study in [132] showed that DMEA is more effective than the compared algorithms with respect to convergence, diversity, and the distribution of the obtained Pareto optimal solutions.

To the author's knowledge no studies exist of the application of ACO algorithms to DMOPs.

5.5 Performance Metrics for Dynamic Multi-Objective Optimisation Problems

Performance measures for DMOPs need to quantify the ability of the algorithm to adapt the Pareto-front under changes in the environment. As indicated in Section 5.3, performance measures for single-objective optimisation problems can be divided into measures that make use of the global optimum and measures independent of the global optimum. In addition, performance measures for MOPs can be divided into measures where the true Pareto front is known and performance metrics where the true Pareto front is unknown (refer to Section 4.7.2).

Since the true Pareto-front is not known for the DMOP proposed in this thesis, the focus of this thesis is on measures that do not require knowledge of the true Pareto-front.

The following performance metrics are used to compare the algorithms of this thesis:

- Number of non-dominated solutions found (ND) [119, 208]

  The number of non-dominated solutions found (refer to Section 4.7.2) is calculated for each iteration before a change to the environment occurs. The average over n_r runs is then calculated for each of these iterations as follows:

$$\overline{ND}^i = \frac{1}{n_r}\sum_{r=1}^{n_r} ND_r^i \tag{5.16}$$

  where ND^i is the number of non-dominated solutions found for iteration i and ND_r^i is the number of non-dominated solutions found for run r at iteration i, which is an iteration before a change occurs in the environment.

  The performance of the algorithm over time is expressed as the average $\overline{ND}^i$ over all iterations, that is,

$$\overline{ND} = \frac{1}{n_c}\sum_{i=1}^{n_c} \overline{ND}^i \tag{5.17}$$

  where n_c is the total number of recorded iterations (or change periods). When comparing the performance of two algorithms, A and B, the total number of times that $\overline{ND}_A^i$ is better than $\overline{ND}_B^i$ is calculated as the performance measure.

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

- Size of the dominated space or hypervolume measure (S_d) [17, 218]

  To compare the performance of one algorithm against another, the hypervolume, S_d (refer to Section 4.7.2), is calculated for each iteration before a change to the environment occurs. The average over n_r runs is then calculated for each of these iterations as follows:

$$\overline{S_d}^i = \frac{1}{n_r} \sum_{r=1}^{n_r} S_{d_r}^i \tag{5.18}$$

  where S_d^i is the hypervolume calculated for iteration i and $S_{d_r}^i$ is the hypervolume calculated for run r at iteration i, which is an iteration before a change occurs in the environment.

  The performance of the algorithm over time is expressed as the average $\overline{S_d}^i$ over all iterations, that is,

$$\overline{S_d} = \frac{1}{n_c} \sum_{i=1}^{n_c} \overline{S_d}^i \tag{5.19}$$

  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 (refer to Section 4.7.2). The hypervolume metric measures how well the algorithms performed in identifying solutions along the full extent of the Pareto front. Higher values of $\overline{S_d}$ indicate more closeness to the true Pareto front and better performance.

  When comparing the performance of two algorithms, A and B, the total number of times that $\overline{S_{dA}}^i$ is better than $\overline{S_{dB}}^i$ is calculated as the performance measure.

- Spacing (SP) or spread metric [179, 208]

To compare the performance of one algorithm against another, the spacing metric, SP^i (refer to Section 4.7.2), is calculated for each iteration before a change to the environment occurs (refer to Equation (4.34)). The average over n_r runs is then calculated for each of these iterations as follows:

$$\overline{SP}^i = \frac{1}{n_r} \sum_{r=1}^{n_r} SP_r^i \tag{5.20}$$

where SP_r^i is the spacing metric value of run r at iteration i, which is an iteration before a change occurs in the environment.

The performance of the algorithm over time is expressed as the average $\overline{SP}^i$ over all iterations, i.e.

$$\overline{SP} = \frac{1}{n_c} \sum_{i=1}^{n_c} \overline{SP}^i \tag{5.21}$$

The smaller the value of $\overline{SP}$, 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.

When comparing the performance of two algorithms, A and B, the total number of times that $\overline{SP}_A^i$ is better than $\overline{SP}_B^i$ is calculated as the performance measure.

5.6 Summary

This section provided an overview of the main characteristics of dynamic problems and the main goals of an optimisation algorithm for dynamic environments. The use of ant algorithms to handle changes in dynamic environments has been described, and DMOO and performance metrics for DMOO have been discussed.

The following chapter presents the multi-objective optimisation algorithms for power-aware routing metrics.

Chapter 6

Multi-Objective Optimisation Algorithms for Power-Aware Routing Metrics

This chapter formally introduces the multi-objective power-aware routing problem. Five multi-objective ant colony optimisation algorithms are then developed to solve the multi-objective power-aware routing problem.

6.1 Introduction

The mobile ad hoc network routing problem is rendered difficult due to node mobility, time-varying capacity of wireless links, and limited resources. Physically available routes become invalid as a result of topology changes brought about by node movement or link failure (i.e. routes may not be found by the routing algorithm) thus causing packets to be dropped and leading both to throughput degradation and increased control overhead. Control packet overhead (e.g. resource reservation, routing and scheduling) is an expensive operation in mobile ad hoc wireless networks in terms of energy consumption and should be kept to a minimum.

Routing algorithms for mobile networks that attempt to optimise routes while attempting to keep message overhead small have been discussed in Chapter 2. Different routing protocols use one or more of a small set of metrics to determine optimal paths. However, some of these metrics have a negative impact on node and network life by inadvertently overusing the energy resources of a small set of nodes in favour of others (refer to Section 2.5.10).

Conservation of power and careful sharing of the cost of routing packets will ensure that node and network life be increased. The simultaneous optimisation of several power-aware metrics will result in energy efficient routes and power saving.

This chapter presents new adaptations of the ant colony system (ACS), the max-min ant system (MMAS), and the multiple colony ACO algorithm for solving the MOP power-aware routing problem. This MOP consists of the following five objectives: 1) minimise energy consumed per packet, 2) maximise time to network partition, 3) minimise variance in node power levels, 4) minimise cost per packet, and 5) minimise maximum node cost while taking into consideration a realistic mobility model. This thesis proposes five algorithms for solving this MOP. The first two algorithms, namely, the energy efficiency for mobile networks using multi-objective ant colony optimisation, multi-pheromone (EEMACOMP) algorithm and the energy efficiency for mobile networks using multi-objective ant colony optimisation, multi-heuristic (EEMACOMH) algorithm are adaptations of multi-objective ant colony optimisation algorithms (MOACO) based on the ant colony system (ACS) algorithm. The next two algorithms, namely, the energy efficiency for mobile networks using multi-objective MAX-MIN ant system optimisation, multi-pheromone (EEMMASMP) algorithm and the energy efficiency for mobile network using multi-objective MAX-MIN ant system optimisation, multi-heuristic (EEMMASMH) algorithm solve the above multi-objective problem using an adaptation of the MAX-MIN ant system optimisation algorithm. The last algorithm, namely, the energy efficiency for mobile networks using multi-objective ant colony optimisation, multi-colony (EEMACOMC) algorithm uses a multiple colony ACO algorithm.

One of the objectives of this thesis is to explore different ways of adapting ACO algorithms for the power aware routing problem and to identify which algorithms have a better performance in terms of the optimisation criteria. The management of the pheromone information in MOO is an important factor for the design of a MOACO algorithm. 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 (EEMACOMH and EEMMASMH), where pheromone updates are proportional to a weighted sum of updates, each update corresponding to an objective, or using multiple pheromone matrices, one for each objective (EEMACOMP and EEMMASMP). EEMACOMC uses multiple colonies, where each colony focuses 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. Finally, EEMACOMP, EEMACOMH and EEMMASMP, EEMMASMH are chosen because they transfer knowledge of the best performing ACO algorithms for single objective optimization, respectively ACS and MMAS, into the multi-objective context for the power aware routing problem.

This chapter is organised as follows. Section 6.2 discusses the suitability of ACO algorithms for the power-aware routing problem. Section 6.3 describes in detail the five metrics for power-aware routing and formulates them mathematically. Section 6.4 formulates the multi-objective optimisation problem. Section 6.5 discusses the mobility model used, namely, the reference point group mobility model. All the changes to the power-aware routing problem formulation resulting from this mobility model are discussed. Section 6.6 presents the five multi-objective ant colony optimisation algorithms proposed in this thesis for simultaneously optimising the five power-aware routing metrics. Section 6.7 describes in detail the elitist non-dominated sorting genetic algorithm for multi-objective power-aware routing (NSGA-II-MPA). The five algorithms proposed will be compared with the NSGA-II-MPA algorithm.

6.2 Suitability of Ant Algorithms for the Power-Aware Routing Problem

The ant algorithms discussed in the previous chapters illustrate the various reasons why it is possible that these types of algorithm could perform well in mobile multi-hop ad hoc networks. Some of these reasons will now be discussed in terms of their relevance to important properties of the power-aware routing problem.

- **Dynamic topology:** The power-aware routing problem is a dynamic optimisation problem because, after applying the mobility model to the MANET, the position of the nodes changes and as a consequence the objective functions change repeatedly. The changes in the environment are responsible for the poor performance of many "classical" routing algorithms in mobile multi-hop ad hoc networks. The ability of ACO algorithms to adapt from the optimum solution for one set of circumstances to the optimal solution to another set of circumstances makes ACO suitable for DOP.

- **Local information:** The power-aware routing problem has certain characteristics, including distributed information, non-stationary stochastic dynamics, and asynchronous evolution of the network status. These characteristics match some of the properties of ACO algorithms, such as the use of local information to generate solutions, indirect communication via the pheromone trails and stochastic state transitions. In contrast to other routing approaches, the ant algorithms use only local information to make stochastic decisions, that is, there is no need to transmit routing tables or other information blocks to other nodes of the network. In addition, ACO algorithms are characterised by the fact that they are multi-agent systems interacting with each other via a form of indirect communication (stigmergy).

- **Link quality:** It is possible to integrate the connection/link quality into the computation of the pheromone concentration, especially into the evaporation process. Link quality is inversely proportional to the cost between nodes and that may easily reflect at the pheromone matrices. Evaporation helps to remove old or poor links from the collective memory of the system. The association of link quality with pheromone information will improve the decision process with respect to the link quality. It is important to note that the link quality approach may be modified so that nodes may also manipulate the pheromone concentration independent of the ants, for example, if a node detects a change in the link quality.

- **Support for multi-path:** Each node has a routing table with entries for all its neighbours. This routing table also contains the pheromone concentration. The decision rule for selecting the next node is based on the pheromone concentration at the current node which is provided for each possible link. Since this decision is stochastic, a set of alternative valid paths can be discovered (multi-path) in order to disperse the data. A multi-path data transfer provides reliable network operations, while considering the energy levels of the nodes.

6.3 Metrics for Power-Aware Routing

This thesis hypothesises that conserving power and carefully sharing the cost of routing packets will ensure that node and network life are increased. This section, therefore, describes five power-aware metrics that result in energy-efficient routes [186]:

1. **Minimise energy consumed per packet (EP):** This is one of the most obvious metrics that reflects the hypothesis of this thesis in respect of conserving energy. Assume that a certain packet, p, traverses the route $T(s, D)$ which consists of nodes $n_1, ..., n_{n_T}$ where n_1 is the source, s, and n_{n_T} the destination, D. Let E_{ij} denote the energy consumed in transmitting (and receiving) one packet over one hop from node n_i to node n_j and T_p the route which the packet, p, traverses. The energy, $EP(T_p)$, consumed for one packet, p, is denoted by

$$EP(T_p) = \sum_{i=1}^{n_T - 1} E_{i(i+1)} \tag{6.1}$$

where n_T is the number of nodes in the route, T_p. The energy, $EP(T)$, consumed for all packets is denoted by

$$EP(T) = \sum_{p=1}^{n_p^T} EP(T_p) \tag{6.2}$$

where n_p^T is the total number of packets from s to D and T is the set of routes T_p, one for each packet p.

Thus, the goal of this metric is to minimise $EP(T)$.

Discussion:

EP facilitates finding the *min-power* path which minimises the overall energy consumption for delivering a packet. Each wireless link is annotated with its transmission energy. The min-power path is that path that minimises the sum of the link costs along the path. In fact, it is interesting to observe that, under light loads, the routes selected when using this metric are identical to the routes selected by shortest-hop routing. This is not a surprising observation because, if the assumption is made that $E_{ij} = E_c, \forall (i, j) \in L$, where E_c is a constant and L is the set of all links, then the power consumed is $(n_T - 1)E_c$. In order to minimise this value n_T needs to be minimised and this is equivalent to finding the shortest-hop path.

In situations where one or more nodes on the shortest-hop path are heavily loaded, the selected route may differ from the route selected by shortest-hop routing. This is as a result of the fact that the amount of energy expended in transmitting one packet over one hop will not be a constant since variable amounts of energy (per hop) may be expended on network contention. Thus, the EP metric will tend to route packets around congested areas (possibly increasing hop-count).

Shortest hop algorithms, while resulting in minimum delay, often result in the early death of some mobile nodes. When mobile nodes are unfairly burdened to support several packet-relaying functions these nodes consume more battery energy and run out of energy earlier than other nodes, thereby creating partitions and disrupting the overall functionality of the ad hoc network. Consider the network illustrated in Figure 6.1 [186]. Here, node 6 will be selected as the route for packets going from 0 to 3, 1 to 4 and 2 to 5, thus providing the shortest hop. As a result, node 6 will expend its battery resources at a faster rate than the other nodes in the network and will be the first node to die. Thus, the EP metric alone does not really meet the goal of increasing node and network life.

2. **Maximise time to network partition (TNP):** The objective of the TNP metric is to divide the load among mobile nodes so that the network will partition in such a way that nodes drain their energy at equal rates. The TNP metric is very important in mission-critical applications such as battle site networks. Optimisation of TNP is very difficult if it has to simultaneously maintain low delay and high throughput. The goal of the TNP metric is to obtain a balanced ad hoc network in order to achieve better performance in terms of execution time and throughput.

   A common approach to balancing network load is to minimise the utilisation of the link with the least capacity:

$$TNP(T_p) = Max_{i,j \,\in\, T_p} \left\{ \frac{1}{c^a(i,j)} \right\} \tag{6.3}$$

   with the capacity of link (i,j) defined as

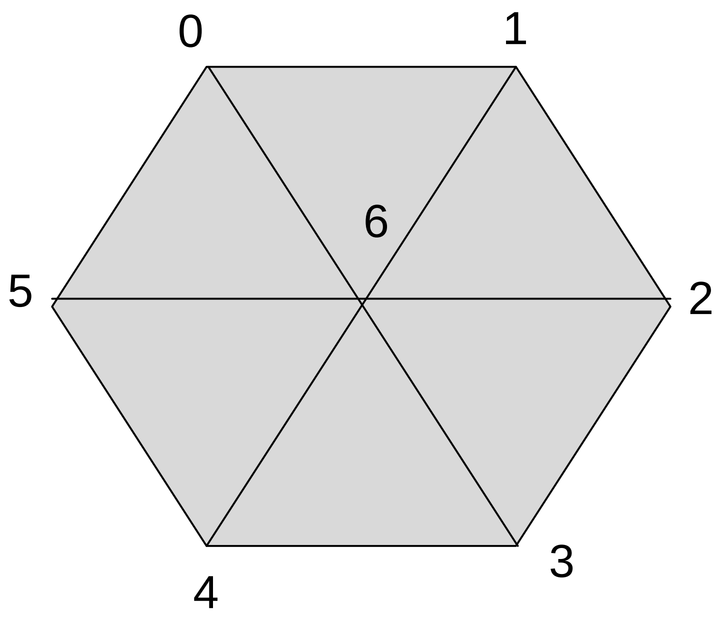

Figure 6.1: A network illustrating the problem with energy per packet as a metric.

$$c^a(i,j) = \frac{e_i^c}{E_{ij}} \tag{6.4}$$

where e_i^c is the current energy of node i, and E_{ij} is the energy expenditure for unit flow transmission over the link (i,j); $c^a(i,j)$ is the capacity of the link (i,j) defined as the number of unit-length messages that may be transmitted along (i,j) before node i runs out of energy.

Discussion: Given a network topology, the maxflow-min-cut theorem [130] may be used to find a minimal set of nodes (i.e. the cut-set nodes) whose removal will cause the network to partition. The routes between these two partitions must pass through one of these cut-set nodes, which are called critical nodes. Therefore, a routing procedure must divide the work among the critical nodes in order to maximise the lifetime of the network. Dividing the work among the critical nodes is similar to the "load balancing" problem, where tasks need to be sent to one of the many servers available so that response time is minimised – this is an NP-complete problem. If care is not taken that the critical nodes drain their power at an equal rate, there will be delay increases as soon as one of these nodes die as a result of network partition. Achieving equal power drain rate among these nodes requires careful routing, and is similar to the load balancing problem described above. Since nodes in different partitions determine routes independently, it is not possible to achieve the global balance required to maximise the network partition time whilst minimising the average delay. Maximising the network lifetime using the TNP metric is a more fundamental goal of an energy efficient routing algorithm: Given alternative routing paths, TNP selects that path which will result in the longest network operation time.

3. **Minimise variance in node power levels (VNP):** In order to measure quantitatively how well the nodes share the load, the variation factor, VF, is introduced. The VF is defined as the variance of the capacity of the nodes:

$$VF(T_p) = \frac{\sum_{i,j \,\in\, T_p} |c^a(i,j) - \mu_T|}{n_T - 1} \tag{6.5}$$

   where μ_T is the average capacity for solution T_p, which is computed as

$$\mu_T = \frac{1}{(n_T - 1)} \sum_{i,j \,\in\, T_p} c^a(i,j) \tag{6.6}$$

   A lower value of VF indicates both a better load distribution and minimum variance in node power and tends to zero for a perfectly balanced load sharing system.

   The VNP metric is based on the assumption that all nodes in the network are equally important and that no single node must be penalised more than any other node. The VNP metric ensures that all the nodes in the network remain up and running for as long as possible.

   Discussion: A problem with minimising variance in node power levels is similar to "load sharing" in distributed systems, where the objective is to minimise response time while keeping the amount of unfinished work in all nodes the same. This goal may be achieved by using a routing procedure, where each node sends traffic through the neighbour with the least amount of data waiting to be transmitted, thus avoiding overloading a node.

4. **Minimise cost per packet (CP):** Let $f_i(e_i^e(t))$ denote the node cost or weight of node n_i where $e_i^e(t)$ represents the total energy expended by node n_i thus far.

   The total cost of sending a packet along path $T(s, D) = (s = n_1, n_2, ...D = n_{n_T})$ is defined as the sum of the node costs of all the nodes that lie along that path. The cost CP of sending a packet p from n_1 to n_{n_T} via intermediate nodes $n_2, ..., n_{n_T-1}$ is,

$$CP(T_p) = \sum_{i=1}^{n_T-1} f_i(e_i^e(t)) \tag{6.7}$$

   The goal of the CP metric is to minimise the total cost over all the packets. The paths selected when using the CP metric should be such that those nodes with depleted energy reserves do not lie on many paths. In this way the network partition is extended.

   Discussion: Since f_i represents the reluctance of a node to forward packets, f_i is chosen as [186]

$$f_i(e_i^e(t)) = \frac{1}{E_i - (e_i^e(t))} \tag{6.8}$$

   where E_i is the initial energy of node n_i when the network is deployed.

   Function f_i is the reciprocal of the residual energy of node n_i. Therefore, as the energy of a node decreases, the cost of using that node increases.

   Using equation (6.8), equation (6.7) becomes

$$CP(T_p) = \sum_{i \in T_p} \frac{1}{E_i - (e_i^e(t))} \tag{6.9}$$

   To summarise, the benefits of the CP metric are the following:

- It is possible to incorporate the residual energy characteristics directly into the routing protocol.
- As a side-effect, the CP metric increases the time to network partition and reduces variation in node costs.
- The effects of network congestion are incorporated (as an increase in node cost due to contention).

5. **Minimise maximum node cost (MNC):** The node cost, $C_{n_i}(t)$, is defined as the ratio of the total energy consumed up to time, t, to the initial energy, E_i [65]:

$$C_{n_i}(t) = \frac{E_i - e_i^c(t)}{E_i} \tag{6.10}$$

The MNC metric is then defined as

$$MNC(T_p) = \max_{i \in T_p} C_{n_i}(t), \forall t > 0 \tag{6.11}$$

The objective of this metric is to minimise MNC. Minimising the cost per node significantly reduces the maximum node cost. MNC delays node failure and reduces variance in remaining battery lives because links with high energy cost are avoided.

Since future network lifetime is difficult to estimate, the last three metrics have been included in order to increase network lifetime indirectly. Variance of residual battery energy of mobile nodes is a simple indication of energy balance and may be used to extend network lifetime. The cost-per-packet metric is similar to the energy-per-packet metric but cost-per-packet includes the residual energy life of each node in addition to the transmission energy. The corresponding energy-aware routing protocol prefers wireless links which require low transmission energy, but, at the same time, avoids nodes with low residual energy with high node cost. The outcome of the MNC metric is that each candidate path is annotated with the maximum node cost among the intermediate nodes (equivalently, the minimal residual battery life), and the path with the minimum path cost, is selected. Maximum node cost is also referred to as the max-min path in online max-min (OMM) protocol [165] because this protocol uses the residual battery life of nodes rather than their node cost. It is clear from the above that, in order to maximise network lifetime, it is necessary to achieve a measure of balance between the energy consumed by a route and the minimum residual energy at the nodes along the chosen route.

The five metrics discussed in this section express, in different ways, the hypothesis of this thesis about conserving energy in the network by selecting routes carefully.

The next section formulates the power-aware routing problem.

6.4 Multi-Objective Optimisation Problem for Power-Aware Routing Metrics Using a Mobility Model

Based on the five metrics defined in Section 6.3, this section defines the dynamic MOP for power-aware routing. The problem is formulated in Section 6.4.1, while Section 6.4.2 discusses the heuristic information used to solve the dynamic MOP.

6.4.1 Problem Formulation

For this thesis, a network is modelled as a directed graph, $G = (V, L)$, where V represents the set of nodes and L is the set of links. The definitions in appendix C are used for the formulation of the power-aware routing problem.

The power-aware routing problem is defined as a dynamic MOP, with the objective to find a path, $T(s, D) = \{s = n_1, n_2, ..., D = n_{n_T}\}$, such that the energy consumed per packet, $EP(T_p)$, utilisation of the most heavily used link, $TNP(T_p)$, variance in node power levels, $VF(T_p)$, cost per packet, $CP(T_p)$, and maximum node cost, $MNC(T_p)$, are minimised (refer to equations (6.1), (6.3), (6.5), (6.9) and (6.11)). More formally, the problem is defined as

$$\text{minimise } \mathbf{f}(T_p) = (EP(T_p),\ TNP(T_p),\ VF(T_p),\ CP(T_p),\ MNC(T_p)) \qquad (6.12)$$

An example of a networking routing problem is discussed in the remainder of this section in order to illustrate the multi-objective power-aware problem stated above: Given the network topology of a directed graph G as illustrated in Figure 6.2, the number associated with each link (i, j) denotes the energy, E_{ij}, consumed in transmitting one

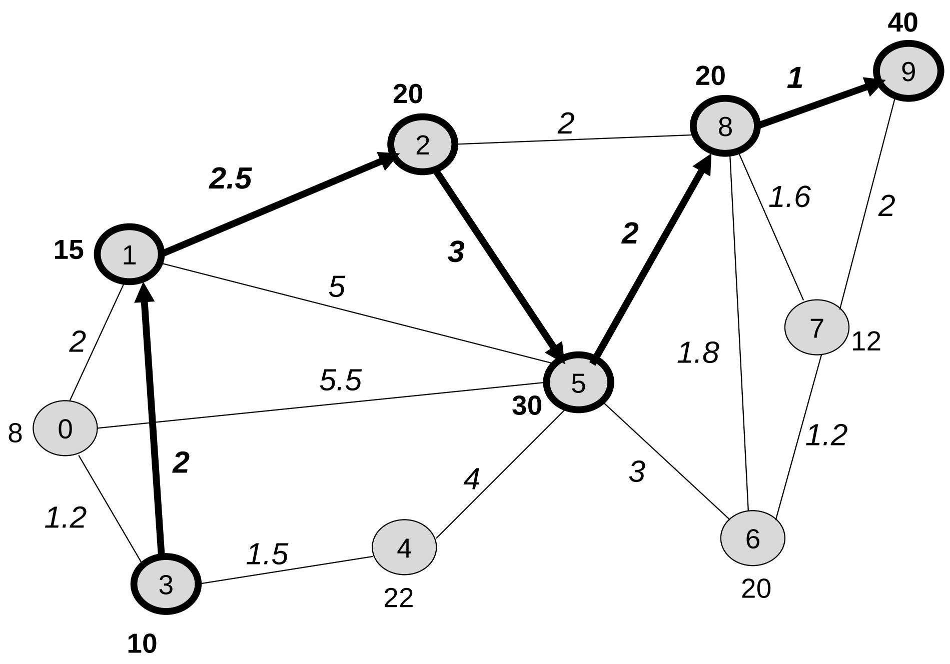

Figure 6.2: A network illustrating the multi-objective optimisation problem

packet over one hop from node i to node j. With each node is associated a number, e_i^c, where e_i^c is the residual power of node i. The initial energy for each node is 50Joule, $n_{pT} = 10$, $s = 3$, and $D = 9$. Table 6.1 presents the values of the objective functions for the random solution, $T_p(3, 9) = (3, 1, 2, 5, 8, 9)$. After finding the set of candidate paths (solutions) from source to destination, and using the concept of non-dominance, the Pareto set is calculated. The objective of the algorithms of this thesis is to calculate the Pareto set in one run and evaluate it just before a change in the environment occurs, using the performance metrics discussed in Section 5.5.

The following subsection presents the heuristic information associated with each sub-objective.

Table 6.1: Objective functions calculated for the route of Figure 6.2

Solution $T_p = (3,1,2,5,8,9)$						
(i,j)		$(3,1)$	$(1,2)$	$(2,5)$	$(5,8)$	$(8,9)$
E_{ij}		2	2.5	3.0	2	1
i	3	1	2	5	8	9
e_i^c	10	15	20	30	20	40
$\frac{e_i^c}{E_{ij}}$		5.00	6.00	6.66	15.00	20.00
$\frac{1}{e_i^c}$	0.10	0.066	0.050	0.033	0.050	0.025
$EP(T_p)$	10(2+2.5+3+2+1)=105					
$TNP(T_p)$	0.2					
μ_{T_p}	10.53					
$VF(T_p)$	5.57					
$CP(T_p)$	0.1+0.066+0.05+0.033+0.05+0.025=0.324					
$MNC(T_p)$	max(0.8, 0.7, 0.6, 0.4, 0.6, 0.2) =0.8					

6.4.2 Heuristic Information

For each of the five sub-objectives, the heuristic information needed to guide the search to an optimal solution is defined as follows:

- Energy consumed per packet heuristic: Let $\eta_{\nu_{ij}}$ denote the heuristic desirability or the attractiveness of the move from node i to node j which is associated with the objective EP. Then,

$$\eta_{\nu_{ij}} = \frac{1}{E_{ij}} \tag{6.13}$$

  The heuristic information, $\eta_{\nu_{ij}}$, guides the ants to construct good solutions with low energy consumed per packet. For this purpose, the function η_ν creates a matrix that associates a row to each node and a column for each feasible neighbour node. Each entry is calculated according to the energy, E_{ij}, consumed in transmitting one packet over one hop from i to j (refer to equations (6.1) and (6.2)). The motivation for using the energy consumed per packet heuristic information is that, intuitively, good solutions will choose a link with low transmission energy.

- Time to network partition heuristic: Let $\eta_{\xi_{ij}}$ denote the heuristic information associated with the objective TNP. Then,

$$\eta_{\xi_{ij}} = \frac{e_i^c}{E_{ij}} \tag{6.14}$$

The heuristic information, $\eta_{\xi_{ij}}$, guides the ants to construct good solutions with minimal utilisation of the most heavily used link in the network. The assignment of each entry in the matrix created by η_ξ is made according to the current energy, e_i^c, of node i and the energy, E_{ij}, which is consumed in transmitting one packet over one hop from i to j (refer to equation (6.3)). The motivation for using the time to network partition heuristic information is that the use of the least heavily used link in the network delays first node failure (due to node energy depletion), and thus delays network partition.

- Variance in node power levels heuristic: Let $\eta_{\pi_{ij}}$ denote the heuristic information associated with the objective VNP. Then,

$$\eta_{\pi_{ij}} = \frac{1}{|c^a(i,j) - \mu_T|} \tag{6.15}$$

where $c^a(i,j)$ denotes the capacity of link (i,j) and μ_T is the average capacity for all links of T which is computed as in equation (6.6).

The heuristic information, $\eta_{\pi_{ij}}$, guides the ants to construct good solutions with minimum variance in node power level and to send traffic through the nodes with less capacity variation. The assignment of each entry in the matrix created by η_π is made according to the current capacity variation of link (i,j) (refer to equation (6.5)). The motivation for using the variance in node power levels heuristic information is that the performance potential from an energy environment depends on the variation in the energy available across the network. While the amount of energy available is definitely relevant, it is the distribution of this energy in space and time which significantly affects network performance. For instance, if large amounts of energy are available, but concentrated in a small region of the network, then the nodes in those regions without energy supply will limit the total useful lifetime of the network. Energy available in other regions of the network may not be able to meet the performance requirements of the system as a whole.

- Cost per packet heuristic: Let $\eta_{\varrho_{ij}}$ denote the heuristic information associated with the objective CP. Then,

$$\eta_{\varrho_{ij}} = \frac{e_j^c}{E_j} \tag{6.16}$$

where e_j^c is the residual energy of node j and E_j is the initial energy of node j.

The heuristic information, $\eta_{\varrho_{ij}}$, guides the ants to construct good solutions with low cost per packet and to send traffic through nodes with more residual energy (refer to equation (6.9)). The motivation for using the cost per packet heuristic information is that, if routes that pass through nodes with high residual energy are used, network lifetime will be maximised [8].

- Maximum node cost heuristic: Let $\eta_{\varsigma_{ij}}$ denote the heuristic information associated with the objective MNC. Then,

$$\eta_{\varsigma_{ij}} = \frac{E_j}{E_j - e_j^c(t)} \tag{6.17}$$

  The heuristic information, $\eta_{\varsigma_{ij}}$, guides the ants to construct good solutions with minimum node cost and to send traffic through links with high residual energy (refer to equation (6.11)). The motivation for using the maximum node cost heuristic information is that network lifetime will be maximised if routes that pass through links with high node cost are avoided [18].

The next section deals with the mobility model used for the multi-objective power-aware routing problem.

6.5 Reference Point Group Mobility Model

This thesis makes use of the reference point group mobility (RPGM) model [29] (refer to Section 2.4). Nodes move in a random fashion using the random way point mobility (RWP) model [29] around a global centre from which the nodes are not able to move farther than a radius of $R_g/2$. This global centre is also mobile, and its motion may follow an arbitrary motion pattern. The RWP model is used to create the motion pattern of the global centre. The effect is that each node will follow the mobility pattern.

RPGM was selected because group motion occurs frequently in ad hoc networks, and RPGM may be readily applied to many existing applications [101]. Moreover, with a proper choice of parameters, RPGM may be used to model several mobility models which have previously been proposed (refer to Section 2.4). RPGM has the advantage of providing a general and flexible framework for describing mobility patterns which are task oriented and time restricted, in addition to being easy to implement and verify. RWP is one of the widely used models for ad hoc and infrastructure wireless simulations [163] and it has been implemented in several network simulations [110, 183].

The dynamic nature of the power-aware routing problem is determined by the mobility model. The mobility model, in part, makes the problem a dynamic optimisation problem. The R_g determines the severity of change and the pause time, T_{sm}, before applying the mobility model, determines the frequency of change (refer to Section 5.1).

The remainder of this section presents the actions to be taken just before applying the RPGM mobility model and all the updates after the change occurs.

Every T_{sm} seconds, a packet is sent from a source node to a destination node using a random route, T_s, from the Pareto set. The choice of a random route is motivated by the fact that none of the Pareto set solutions is absolutely better than the other non-dominated solutions and all of them are equally acceptable with regards to the satisfaction of all the objectives.

The energy required to do a single-hop transmission from node i to node j is E_{ij}. Therefore, the current energy, e_i^c, for each node $i \in T_s$ is updated according to

$$e_i^c = e_i^c - E_{ij} \tag{6.18}$$

where link $(i, j) \in T_s$.

The RPGM mobility model is applied (refer to Section 6.5) and the distances, d_{ij}, between all nodes i and j belonging to L are recalculated. Since the energy required for transmitting a message along a link (i, j) is proportionate to the distance, d_{ij}, the energy, E_{ij}, for all links $(i, j) \in L$ is recalculated using

$$E_{ij} = max\left\{\frac{1}{(T_r)^2}, \frac{1}{(T_r)^2} * d_{ij}^2\right\} \tag{6.19}$$

where T_r is the transmission range of the node and d_{ij} is the Euclidean distance between nodes i and j (refer to Section 2.5.8). The Pareto set is updated by deleting the invalid routes (routes with broken links), i.e. routes for which $E_{ij} > e_i^c$ or $d_{ij} > T_r$. Since the objective functions have been changed, the sub-objectives for all solutions of the current Pareto set are evaluated using equations (6.2), (6.3), (6.5), (6.9), (6.11) and all the dominated solutions are discarded from the Pareto set.

Finally, the heuristic matrices are recalculated (refer to equations (6.13)-(6.17)).

The above steps are summarised in Algorithm 12.

Algorithm 12 General Procedure of ApplyMobilityChanges

```
Choose a random route, $T_s$, from $P_f$;
Send one packet from source to destination using $T_s$;
for all nodes $i \in T_s$ do
    Update $e_i^c$ according to equation (6.18);
end for
Apply the RPGM mobility model (refer to Section 6.5);
for all links $(i, j) \in L$ do
    Recalculate $d_{ij}$;
    Recalculate $E_{ij}$ according to equation (6.19);
end for
Delete invalid solutions from $P_f$;
for all $T \in P_f$ do
    Re-evaluate the sub-objectives using equations (6.2), (6.3), (6.5), (6.9), (6.11);
end for
Remove dominated solutions from $P_f$;
for all links $(i, j) \in L$ do
    Recalculate all the heuristic matrices according to equations (6.13)-(6.17);
end for
```

6.6 Multi-Objective Ant Colony Optimisation

This section presents the five multi-objective ant colony optimisation algorithms proposed in this thesis for simultaneously optimising the five power-aware routing metrics given in Section 6.3.

Subsection 6.6.1 discusses the general framework of the ACO algorithms developed for the power-aware routing problem. Subsections 6.6.2 to 6.6.6 respectively present the EEMACOMP, EEMACOMH, EEMMASMP, EEMMASMH and EEMACOMC multi-objective ACO algorithms.

6.6.1 General Framework of ACO Algorithms for the Power-Aware Routing Problem

An overview of the ACO algorithms for the power-aware routing problem is given below in Figure 6.3.

The remainder of this subsection discusses the solution construction process for each algorithm, the management of the archive, P_f (Pareto set), and the global pheromone update.

Solution construction process. At every iteration of the developed algorithms each ant constructs a complete solution as follows: Starting from the source node, s,

```
1. Initialise control parameters, and pheromone matrices.
2. Calculate the heuristic matrices.
3. Place all ants at source node $s$.
4. Repeat for the duration of the simulation time
    (a) Repeat for the duration of pause time {Optimise static problem for power-aware routing }
        i. For each ant
            A. Construct a solution.
            B. Evaluate the solution.
            C. If solution is non-dominated insert it into the archive.
            D. Keep the size of the archive to a predefined limit.
        ii. Apply global pheromone update using all non-dominated solutions of the archive.
    (b) Dynamic aspect for power-aware routing
        i. Send a packet using a solution from the archive.
        ii. Apply RPGM mobility model.
        iii. ApplyMobilityChanges {apply updates due to the mobility of the nodes}
            A. Update energy levels.
            B. Update node distances.
            C. Eliminate invalid solutions from the archive.
            D. Re-evaluate the sub-objectives.
            E. Eliminate dominated solutions from the archive.
            F. Recalculate heuristic information.
        iv. Apply pheromone conservation.
5. Return solutions of the archive.
```

Figure 6.3: Overview of ACO algorithms for the power-aware routing problem

a non-visited node is selected at each step using an adaptation of the ACS transition rule for the EEMACOMP, EEMACOMH, and EEMACOMC algorithms (refer to Section 3.5.2) and an adaptation of the MAX-MIN transition rule for the EEMMASMP and EEMMASMH algorithms (refer to Section 3.5.3). This process continues until the destination node, D, is reached. When a new node has been added to a candidate solution, a local pheromone update is performed for the EEMACOMP, EEMACOMH, and EEMACOMC algorithms (refer to Section 3.5.2).

Archive management. At the conclusion of each iteration, the Pareto set, P_f, is updated including the non-dominated solutions that have been found thus far. Dominated solutions are removed from P_f. If the number of non-dominated solutions exceeds the predefined archive size, P_{as}, then a truncation operator is used to remove solutions with a lower value of the crowding distance from the archive (refer to Section 4.6.2). The use of the crowding distance guides the selection process at the various stages of the algorithm towards a uniformly spread-out Pareto-optimal front.

Global pheromone update. In the case of multiple pheromone algorithms (EEMACOMP and EEMMASMP) each pheromone matrix associated with each objective is updated using all the solutions of the archive, P_f. That is, every ant that generated a solution in the non-dominated front, P_f, from the beginning of the run is allowed to update all pheromone matrices (refer to Section 4.5.4). In the case of a single pheromone algorithm (EEMACOMH, EEMMASMH) every ant that generated a solution in the non-dominated front from the beginning of the run is allowed to update the pheromone matrix (refer to Section 4.5.3).

The next sections elaborate in detail on the different steps for each of the algorithms.

6.6.2 Energy Efficiency Using Multi-Objective Ant Colony Optimisation, Multi-Pheromone Algorithm

In accordance with the ant colony system (ACS) (refer to Section 3.5.2) the energy efficiency for a mobile network using a multi-objective ant colony optimisation, multi-pheromone (EEMACOMP) algorithm makes use of a colony of ants and several pheromone matrices. This concept has been borrowed from Iredi *et al.* [104] (refer to Section 4.5.4). EEMACOMP calculates five pheromone matrices – one for each optimisation criterion, which, together with a heuristic matrix for each optimisation criterion, are used to calculate transition probabilities. The association of different pheromone matrices for each objective may be useful if the weight of each objective is different and the solution components for each objective must be defined differently (refer to Section 4.5.4).

From equation (3.7), the transition rule for EEMACOMP becomes

$$j = \begin{cases} arg\ \max_{u\,\in\,N_i^k(t)} \Big\{ \tau_{\nu_{iu}}^{\lambda_\nu}(t)\eta_{\nu_{iu}}^{\beta_\nu}(t) + \tau_{\xi_{iu}}^{\lambda_\xi}(t)\eta_{\xi_{iu}}^{\beta_\xi}(t) + \tau_{\pi_{iu}}^{\lambda_\pi}(t)\eta_{\pi_{iu}}^{\beta_\pi}(t) + \tau_{\varrho_{iu}}^{\lambda_\varrho}(t)\eta_{\varrho_{iu}}^{\beta_\varrho}(t) + \\ \qquad \tau_{\varsigma_{iu}}^{\lambda_\varsigma}(t)\eta_{\varsigma_{iu}}^{\beta_\varsigma}(t) \Big\} \quad \text{if } r \leq r_0 \\ J \qquad\qquad \text{if } r > r_0 \end{cases} \tag{6.20}$$

where $\lambda_\nu, \lambda_\xi, \lambda_\pi, \lambda_\varrho, \lambda_\varsigma \in (0,1)$ are user-defined parameters which establish the importance of the objectives; $\tau_{\nu_{ij}}(t)$, $\tau_{\xi_{ij}}(t)$, $\tau_{\pi_{ij}}(t)$, $\tau_{\varrho_{ij}}(t)$ and $\tau_{\varsigma_{ij}}(t)$ are the pheromone matrices, one per objective; $\eta_{\nu_{ij}}(t)$, $\eta_{\xi_{ij}}(t)$, $\eta_{\pi_{ij}}(t)$, $\eta_{\varrho_{ij}}(t)$ and $\eta_{\varsigma_{ij}}(t)$ represent the heuristic information. The parameters β_ν, β_ξ, β_π, β_ϱ, β_ς define the relative influence among heuristic information. The other symbols are as defined in Section 3.5.2. $J \in N_i^k(t)$ is a node randomly selected based on the probability given by equation (3.8) [104]. According to this equation the probability for EEMACOMP becomes

$$p_{iJ}^{k}(t) = \begin{cases} \dfrac{\left(\tau_{\nu_{iJ}}^{\lambda_\nu}(t)\eta_{\nu_{iJ}}^{\beta_\nu}(t)+\tau_{\xi_{iJ}}^{\lambda_\xi}(t)\eta_{\xi_{iJ}}^{\beta_\xi}(t)+\tau_{\pi_{iJ}}^{\lambda_\pi}(t)\eta_{\pi_{iJ}}^{\beta_\pi}(t)+\tau_{\varrho_{iJ}}^{\lambda_\varrho}(t)\eta_{\varrho_{iJ}}^{\beta_\varrho}(t)+\tau_{\varsigma_{iJ}}^{\lambda_\varsigma}(t)\eta_{\varsigma_{iJ}}^{\beta_\varsigma}(t)\right)}{\sum_{\forall u\in N_i^k(t)}\left(\tau_{\nu_{iu}}^{\lambda_\nu}(t)\eta_{\nu_{iu}}^{\beta_\nu}(t)+\tau_{\xi_{iu}}^{\lambda_\xi}(t)\eta_{\xi_{iu}}^{\beta_\xi}(t)+\tau_{\pi_{iu}}^{\lambda_\pi}(t)\eta_{\pi_{iu}}^{\beta_\pi}(t)+\tau_{\varrho_{iu}}^{\lambda_\varrho}(t)\eta_{\varrho_{iu}}^{\beta_\varrho}(t)+\tau_{\varsigma_{iu}}^{\lambda_\varsigma}(t)\eta_{\varsigma_{iu}}^{\beta_\varsigma}(t)\right)} \\ \qquad \text{if } J \in N_i^k(t) \\ 0 \quad \text{otherwise} \end{cases} \tag{6.21}$$

where $N_i^k(t)$ is a list containing a set of valid nodes to visit from node i.

All objective functions are normalized and bounded by an upper limit value that gives approximately the same magnitude to each objective function [103]. To further reduce any problems which may arise due to difference in magnitude of the objective functions, the above rule uses the sum of the products of the different pheromone and heuristic information matrices instead of the product as used by Iredi *et al.* [104].

The transition rule in equation (6.21) creates a bias towards those nodes which are connected by links with a large amount of pheromone. The same transition rule will bias the search toward minimising energy consumed per packet, utilisation of the link with the least capacity, variance in node power levels, cost per packet, and maximum node cost. Such nodes have a higher probability of being selected.

Heuristic information matrices are calculated at the beginning of the simulation and every time there is a change in the environment using equations (6.13)-(6.17). Pheromone matrix initialisation, and global and local updates are discussed next.

Pheromone matrix initialisation

The five pheromone matrices are initialised as follows:

$$\tau_{\psi_{ij}}(t) = \tau_{0\psi} \tag{6.22}$$

for all $\psi \in \{\nu, \xi, \pi, \varrho, \varsigma\}$, and for all $(i, j) \in L$, where $\tau_{0\psi s}$ are small positive values calculated for each objective, as follows

$$\tau_{0\psi} = \frac{1}{N_G B_\psi} \tag{6.23}$$

where N_G is the number of nodes; $B_\nu = EP(T_\nu)$, where T_ν is the route from source s to destination D with minimal energy consumed per packet; B_ξ is the maximum capacity of all links belonging to T for all paths T from s to D; B_π is the minimal variance of the capacity of all nodes for all paths T from s to D; $B_\varrho = CP(T_\varrho)$, where T_ϱ is the route with minimal cost consumed per packet; B_ς denotes the maximum energy consumed in transmitting one packet over one hop for all links belonging to T and for all paths T from s to D. Routes T, T_ν, and T_ϱ are calculated using greedy heuristics as follows: Starting from the source node, s, a non-visited node is selected at each step according to the heuristic information associated with the corresponding objective. This process continues until the destination node, D, is reached and the greedy algorithm is completed.

Global pheromone update

Once all n_k ants have constructed solutions (refer to Section 6.6.1), the archive (Pareto set), P_f, is updated including the non-dominated solutions that have been found thus far. All the solutions from P_f are then selected to modify the pheromone trails globally. Each pheromone trail associated with each objective is updated using the following global update rules (refer to Section 3.5.2):

$$\tau_{\psi_{ij}}(t+1) = (1-\rho_g)\tau_{\psi_{ij}}(t) + \rho_g \Delta^\psi \tau_{ij}(t) \tag{6.24}$$

where $\rho_g \in [0,1]$ is the global evaporation factor, ψ_{ij} represents either ν_{ij}, ξ_{ij}, π_{ij}, ϱ_{ij}, or ς_{ij} depending on the sub-objective, and

$$\Delta^\psi \tau_{ij}(t) = \begin{cases} \frac{1}{C(T_k)} & \text{if } i,j \in T_k \\ 0 & \text{otherwise} \end{cases} \tag{6.25}$$

for all $T_k \in P_f$, where $C(T_k)$ represents the corresponding objective function.

If ρ_g is small, this strategy favours exploitation by encouraging ants to search in the vicinity of the solutions of the current Pareto front.

Local pheromone update

When a new node is added to a solution being constructed by an ant, a local pheromone update is performed using

$$\tau_{\psi_{ij}}(t+1) = (1-\rho_l)\tau_{\psi_{ij}}(t) + \rho_l \tau_{0\psi} \tag{6.26}$$

where $\tau_{0\psi}$ is the initial amount of pheromone on every link, ψ represents either ν, ξ, π, ϱ, or ς depending on the sub-objective, and $\rho_l \in [0,1]$ is the local evaporation factor.

The EEMACOMP discussed above has been developed for static objective functions. The following subsection discusses an adaptation of EEMACOMP for dynamic objective functions.

Dynamic EEMACOMP

The problem considered in this thesis is a dynamic optimisation problem. Therefore, the static EEMACOMP above has to be adapted to dynamic environments. This section presents an adaptation of EEMACOMP to dynamic environments.

The dynamic ACO algorithm is based on the principle of dividing the simulation time, $S_{T_{tot}}$, into n_{ts} time slices with equal length, $S_{T_{tot}}/T_{sm}$, where T_{sm} is the length of the pause time for the mobility model. T_{sm} is an indication of frequency of change (refer to Section 5.1). During each time slice a problem very similar to a static EEMACOMP is created, and optimisation is carried out. For each of these static problems the aim is to simultaneously minimise the five objectives and to create a Pareto set, P_f, with the best non-dominated solutions that have been found within the time slice. Every T_{sm} seconds the mobility model is applied, the position of the nodes is changed with severity of change, R_g (refer to Section 5.1), and a new static EEMACOMP is created. The concept of a time slice has been introduced to bound the time dedicated to each static problem. A different strategy could be to stop and restart the algorithm each time a new event occurs (i.e. the mobility model is applied). The disadvantage of such an approach is that the time to be dedicated to each static problem would not be known in advance, and, consequently, optimisation may be interrupted before a good local minimum is reached, thus producing unsatisfactory results.

If T_{sm} is small enough, the change in the problem is frequent but a lesser number of iterations are allowed to track new optimal solutions. There is a lower limit to T_{sm} below which, albeit a small change in the problem, the number of iterations are not enough for an algorithm to track the new optimal solutions adequately. Such a limiting T_{sm} will depend on the chosen algorithm, but importantly allows the best scenario which the chosen algorithm can achieve. The next chapter investigates this aspect and finds such a limiting T_{sm} for the different developed algorithms.

Once a time slice is completed and the respective static problem has been solved by the ACS based EEMACOMP algorithm, each pheromone matrix will contain information on the characteristics of good solutions for this specific problem. In particular, good paths will manifest high values in the corresponding entries of the pheromone matrix for each objective. Pheromone information, together with the current Pareto set, may be passed on to the static problem corresponding to the following time slice since the two problems would potentially be very similar. This operation would prevent optimisation having to restart each time from scratch and would contribute greatly to the good performance of the EEMACOMP algorithm. In case there is no similarity between the two static problems above, the performance of EEMACOMP will be the same as with restarting the optimisation from scratch.

As part of the dynamic EEMACOMP, the ApplyMobilityChanges procedure is called (refer to Algorithm 12) and pheromone conservation rules are applied.

The **ApplyMobilityChanges** procedure sends a message from source to destination, applies the RPGM mobility model and executes all the necessary updates after the change has occured. The P_f archive is re-evaluated and dominated solutions are removed from the archive.

Pheromone conservation rules are applied in order to promote the efficient passing on of information regarding the properties of good solutions from a static problem in one time slice to the static problem in the following time slice. At the end of each time slice the mobility procedure is applied and the environment changes. The distances between nodes are modified, the energy required for transmitting a message along a link (i, j) may be changed and nodes may move out of transmission range or run out of energy. Pheromone trails must be adapted in order to reflect these changes. In order to correct the amount of pheromones, this thesis uses the pheromone update rules for dynamic environments which were proposed by Guntsch and Middendorf [90] (refer to Section 5.2). The choice of these update rules is motivated by the fact that pheromone values are not completely reinitialised but a trace of old values containing characteristics of good solutions remain. These rules distribute reset-values $\gamma_{\nu_i}, \gamma_{\xi_i}, \gamma_{\pi_i}, \gamma_{\varrho_i}, \gamma_{\varsigma_i} \in [0, 1]$ to each node i. These values determine the amount of re-initialisation of the pheromone values on links j incident to i according to

$$\tau_{\psi_{ij}}(t+1) = (1-\gamma_{\psi_i})\tau_{\psi_{ij}}(t) + \gamma_{\psi_i}\frac{1}{n_G - 1} \tag{6.27}$$

where ψ_i represents either ν_i, ξ_i, π_i, ϱ_i, or ς_i depending on the sub-objective.

In order to calculate the reset-values, the η-strategy (refer to Section 5.2.2) is used, where heuristic information is applied to decide to what degree pheromone values are equalised. According to the η-strategy, the closer a link is to a modified node, the greater the amount of pheromone which will be removed from this link. The motivation behind the choice of η-strategy is that this specific strategy helps the ants to find new paths by increasing the influence of new links. The measurement of closeness is based on the link costs in terms of the different objectives.

Each node i is assigned the values $\gamma_{\nu_i}, \gamma_{\xi_i}, \gamma_{\pi_i}, \gamma_{\varrho_i}, \gamma_{\varsigma_i}$ proportionate to the distances $d^{\eta}_{\nu_{ij}}$, $d^{\eta}_{\xi_{ij}}$, $d^{\eta}_{\pi_{ij}}$, $d^{\eta}_{\varrho_{ij}}$, and $d^{\eta}_{\varsigma_{ij}}$ from the changed component, j, and equalisation is carried out on all links incident to the changed component. These distances are derived from the heuristics $\eta_{\nu_{ij}}$, $\eta_{\xi_{ij}}$, $\eta_{\pi_{ij}}$, $\eta_{\varrho_{ij}}$, and $\eta_{\varsigma_{ij}}$ such that a high heuristic implies a large distance.

For each pheromone matrix, the $\gamma_{\nu_i}, \gamma_{\xi_i}, \gamma_{\pi_i}, \gamma_{\varrho_i}, \gamma_{\varsigma_i}$ values are calculated using

$$\gamma_{\psi_i} = \max\{0, d^{\eta}_{\psi_{ij}}\} \tag{6.28}$$

with

$$d^{\eta}_{\psi_{ij}} = 1 - \frac{\bar{\eta}_{\psi}}{\lambda_E . \eta_{\psi_{ij}}} \tag{6.29}$$

where ψ represents either ν, ξ, π, ϱ, or ς depending on the sub-objective, and

$$\bar{\eta}_{\psi} = \frac{1}{n_G * (n_G - 1)} \sum_{i=1}^{n_G} \sum_{\substack{j=1 \\ j \neq i}}^{n_G} \eta_{\psi_{ij}} \tag{6.30}$$

where $\lambda_E \in [0, \infty)$ is a strategy-specific parameter.

Algorithm 13 summarises EEMACOMP. In summary, if the environment has changed, then pheromone conservation is applied to allow the ants to explore new, relevant areas of the search space in later iterations.

Algorithm 13 General Procedure of EEMACOMP

```
$t = 0$; $P_f = \emptyset$; Set timer $T_{sm}$;
Initialise $s$, $D$, $r_0$, $\beta_\nu$, $\beta_\xi$, $\beta_\pi$, $\beta_\varrho$, $\beta_\varsigma$, $\lambda_\nu$, $\lambda_\xi$, $\lambda_\pi$, $\lambda_\varrho$, $\lambda_\varsigma$;
Initialize $n_k$; {number of ants}
Initialize $P_{as}$; {Maximum archive size}
Initialize $\tau_{0\nu}$, $\tau_{0\xi}$, $\tau_{0\pi}$, $\tau_{0\varrho}$, $\tau_{0\varsigma}$;
for each link $(i, j)$ do
  Initialize pheromone matrices $\tau_{\nu_{ij}}(t), \tau_{\xi_{ij}}(t), \tau_{\pi_{ij}}(t), \tau_{\varrho_{ij}}(t), \tau_{\varsigma_{ij}}(t)$ using equation (6.22);
end for
for each link $(i, j)$ do
  Calculate $\eta_{\nu_{ij}}, \eta_{\xi_{ij}}, \eta_{\pi_{ij}}, \eta_{\varrho_{ij}}, \eta_{\varsigma_{ij}}$;
end for
Place all ants, $k = 1, ..., n_k$ at source node $s$;
while $t <= S_{T_{tot}}$ do
  {begin resolve static EEMACOMP}
  while $T_{sm}$ seconds not elapsed do
    for $k = 1$ to $n_k$ do
      $T = \emptyset$;
      $i = s$; {where $s$ the source node and $i$ the current node}
      while $(i \neq D)$ do
        Build set $N_i^k(t)$ for node $i$;
        Assign probability $p_{ij}^k$ to each node of $N_i^k$ according to equation (6.21);
        Select node $j$ of $N_i^k$ using equations (6.20) and (6.21);
        $T = T \cup j$ ;
        Apply local update pheromone using equation (6.26);
        $i = j$;
      end while
      Evaluate the sub-objectives for solution T using equations (6.2), (6.3), (6.5), (6.9), (6.11) ;
      if $T$ is non-dominated by any $T_x \in P_f$ then
        $P_f = P_f \cup T - \{T_y \mid T \prec T_y\}, \forall T_y \in P_f$;
        if size of $P_f > P_{as}$ then
          Truncate $P_f$
        end if
      end if
    end for
    for all $T_k \in P_f$ do
      Update_global_pheromone $\forall(i, j) \in T_k$ using equations (6.24)-(6.25);
    end for
  end while
  {end resolve static EEMACOMP}
  Call Procedure ApplyMobilityChanges() (refer to Algorithm 12);
  Apply pheromone conservation $\forall(i, j) \in L$ using equations (6.27)-(6.30);
  $t = t + T_{sm}$;
  Reset timer $T_{sm}$;
end while
Return $P_f$
```

6.6.3 Energy Efficiency Using Multi-Objective Ant Colony Optimisation, Multi-Heuristic Algorithm

In accordance with the ant colony system (ACS) (refer to Section 3.5.2), the energy efficiency for mobile networks using a multi-objective ant colony optimisation, multi-heuristic (EEMACOMH) algorithm makes use of a colony of ants, one pheromone matrix and a different heuristic matrix for each objective.

This concept has been borrowed from Iredi *et al.* [104] (refer to Section 4.5.3). The EEMACOMH algorithm is similar to EEMACOMP with the only difference being that EEMACOMH uses a single pheromone matrix instead of five pheromone matrices. This single pheromone matrix represents the desirability of the solution components with regard to all the objectives.

From equation (3.7), the transition rule for EEMACOMH becomes

$$j = \begin{cases} arg\ \max_{u\,\in\,N_i^k(t)} \left\{\tau_{iu}(t) \times (\eta_{\nu_{iu}}^{\beta_\nu}(t) + \eta_{\xi_{iu}}^{\beta_\xi}(t) + \eta_{\pi_{iu}}^{\beta_\pi}(t) + \eta_{\varrho_{iu}}^{\beta_\varrho}(t) + \eta_{\varsigma_{iu}}^{\beta_\varsigma}(t))\right\} \\ \qquad\qquad \text{if } \ r \leq r_0 \\ J \qquad\quad \text{if } r > r_0 \end{cases} \tag{6.31}$$

where $\tau_{ij}(t)$ is the pheromone matrix – the same for all objectives – and $\eta_{\nu_{ij}}(t)$, $\eta_{\xi_{ij}}(t)$, $\eta_{\pi_{ij}}(t)$, $\eta_{\varrho_{ij}}(t)$ and $\eta_{\varsigma_{ij}}(t)$ represent the heuristics. The parameters β_ν, β_ξ, β_π, β_ϱ, and β_ς define the relative influence among heuristic information. The rest of the symbols are as defined in Section 3.5.2. $J \in N_i^k(t)$ is a node randomly selected based on the probability given by equation (3.8). According to this equation the probability for EEMACOMH becomes

$$p_{iJ}^k(t) = \begin{cases} \frac{\left(\tau_{iJ}(t)\times(\eta_{\nu_{iJ}}^{\beta_\nu}(t)+\eta_{\xi_{iJ}}^{\beta_\xi}(t)+\eta_{\pi_{iJ}}^{\beta_\pi}(t)+\eta_{\varrho_{iJ}}^{\beta_\varrho}(t)+\eta_{\varsigma_{iJ}}^{\beta_\varsigma}(t))\right)}{\sum_{\forall u \in N_i^k(t)}\left(\tau_{iu}(t)\times(\eta_{\nu_{iu}}^{\beta_\nu}(t)+\eta_{\xi_{iu}}^{\beta_\xi}(t)+\eta_{\pi_{iu}}^{\beta_\pi}(t)+\eta_{\varrho_{iu}}^{\beta_\varrho}(t)+\eta_{\varsigma_{iu}}^{\beta_\varsigma}(t))\right)} \\ \qquad\qquad \text{if } J \ \in \ N_i^k(t) \\ 0 \quad \text{otherwise} \end{cases} \tag{6.32}$$

The solution construction process is as given in Section 6.6.1. Heuristic information matrices are calculated at the beginning of the simulation and every time there is a change in the environment using equations (6.13)-(6.17). Pheromone matrix initialisation, and global and local updates are discussed next.

Pheromone matrix initialisation

The pheromone matrix is initialised as follows:

$$\tau_{ij}(t) = \tau_0 \tag{6.33}$$

for all $(i, j) \in L$, where τ_0 is a small positive value calculated as

$$\tau_0 = \frac{1}{N_G \times B_a} \tag{6.34}$$

where N_G is the number of nodes, and B_a is the length of the initial solution produced by the nearest neighbour heuristic [70].

Global pheromone update

Once each of the n_k ants has completed its solution, if the ant's solution, T_k, is non-dominated by the solutions of P_f, the solution is stored in the archive, P_f, and solutions of P_f newly dominated by T_k are removed from the archive. All the solutions from P_f are then selected to modify the pheromone trail globally. The pheromone trail is updated using the following global update rule (refer to Section 3.5.2) [161]:

$$\tau_{ij}(t+1) = (1-\rho_g)\tau_{ij}(t) + \rho_g \Delta\tau^k(t),\ \forall\ (i,j) \in T_k \tag{6.35}$$

where

$$\Delta\tau^k = \frac{1}{EP(T_k) + TNP(T_k) + VF(T_k) + CP(T_k) + MNC(T_k)} \tag{6.36}$$

for all $T_k \in P_f$, where each objective value is normalised to ensure that all objectives have approximately the same magnitude [103].

Local pheromone update

When a new node is added to a solution being constructed by an ant, a local pheromone update is performed using

$$\tau_{ij}(t+1) = (1-\rho_l)\tau_{ij}(t) + \rho_l \tau_0 \tag{6.37}$$

where τ_0 is the initial amount of pheromone on every link of L.

The EEMACOMH discussed above has been developed for static objective functions. The following section discusses an adaptation of EEMACOMH for dynamic objective functions.

Dynamic EEMACOMH

As with EEMACOMP (refer to Section 6.6.2) the dynamic EEMACOMH is addressed as a series of static problems with a new problem being defined each time the mobility model is applied. The ApplyMobilityChanges procedure (refer to Algorithm 12) is applied at the end of each time slice.

The pheromone conservation rules are applied as for EEMACOMP, but only one pheromone matrix is updated as follows:

$$\tau_{ij}(t+1) = (1-\gamma_i)\tau_{ij}(t) + \gamma_i \frac{1}{n_G - 1} \tag{6.38}$$

where

$$\gamma_i = \max\{0, d_{ij}^{\eta}\} \tag{6.39}$$

with

$$d_{ij}^{\eta} = 1 - \frac{\bar{\eta}}{\lambda_E.(\eta_{\nu_{ij}} + \eta_{\xi_{ij}} + \eta_{\pi_{ij}} + \eta_{\varrho_{ij}} + \eta_{\varsigma_{ij}})} \tag{6.40}$$

where

$$\bar{\eta} = \frac{1}{n_G * (n_G - 1)} \sum_{i=1}^{n_G} \sum_{\substack{j=1 \\ j \neq i}}^{n_G} (\eta_{\nu_{ij}} + \eta_{\xi_{ij}} + \eta_{\pi_{ij}} + \eta_{\varrho_{ij}} + \eta_{\varsigma_{ij}}) \tag{6.41}$$

where $\lambda_E \in [0, \infty)$ is a strategy-specific parameter.

The EEMACOMH algorithm is very similar to the EEMACOMP algorithm (refer to Algorithm 13) with the following differences in respect of EEMACOMH: a) there is only one pheromone matrix for the EEMACOMH algorithm, b) if the environment is modified, equations (6.38)-(6.41) are used for pheromone conservation, c) equations (6.35) and (6.36) are used for the global pheromone updating, d) the local pheromone update is performed using equation (6.37), and e) the transition rule is applied according to equations (6.31) and (6.32).

In the interests of completeness, Algorithm 14 presents the general procedure of the proposed EEMACOMH algorithm.

6.6.4 Energy Efficiency Using Multi-Objective MAX-MIN Ant System Optimisation, Multi-Pheromone Algorithm

Energy efficiency for mobile networks using the multi-objective MAX-MIN ant system optimisation, multi-pheromone (EEMMASMP) algorithm is based on the MAX-MIN ant system (MMAS) (refer to Section 3.5.3). The EEMMASMP algorithm modifies MMAS with the following changes in order to solve the dynamic multi-objective problem:

- Similar to the previous two algorithms, EEMMASMP finds a set of Pareto optimal solutions termed P_f, instead of finding a single optimal solution as is done by MMAS.

- EEMMASMP solves the dynamic aspect of the multi-objective power-aware routing problem.

In accordance with MMAS, the EEMMASMP algorithm uses a colony of ants and several pheromone matrices. EEMMASMP calculates five pheromone matrices – one for each optimisation criterion – which, together with a heuristic matrix for each optimisation criterion, are used to calculate transition probabilities. This concept has been borrowed from Iredi *et al.* [104] (refer to Section 4.5.4).

The EEMACOMP and EEMMASMP algorithms are very similar. The only difference between EEMACOMP and EEMMASMP is that the ACS (adapted to develop EEMA-

Algorithm 14 General Procedure of EEMACOMH

```
$t = 0$; $P_f = \emptyset$; Set timer $T_{sm}$;
Initialize $s, D$, $r_0$, $\rho$, $\beta_\nu, \beta_\xi, \beta_\pi, \beta_\varrho, \beta_\varsigma$, $n_k$;
Initialize $P_{as}$; {Maximum archive size}
Calculate $\tau_0$ using equation (6.34);
for each link $(i, j)$ do
    Initialize pheromone matrix $\tau_{ij}(t) = \tau_0$;
    Calculate $\eta_{\nu_{ij}}, \eta_{\xi_{ij}}, \eta_{\pi_{ij}}, \eta_{\varrho_{ij}}, \eta_{\varsigma_{ij}}$;
end for
Place all ants, $k = 1, ..., n_k$ at source node $s$;
while $t <= S_{T_{tot}}$ do
    {begin resolve static EEMACOMH}
    while $T_{sm}$ seconds not elapsed do
        for $k = 1$ to $n_k$ do
            $T = \emptyset$;
            $i = s$; {where $s$ the source node and $i$ the current node}
            while $(i \neq D)$ do
                Build set $N_i^k(t)$ for $i$;
                Assign probability $p_{ij}^k$ to each node of $N_i^k$ according to equation (6.32);
                Select node $j$ of $N_i^k$ using equations (6.31) and (6.32);
                $T = T \cup j$ ;
                Apply local update pheromone using equation (6.37);
                $i = j$;
            end while
            Evaluate the sub-objectives for solution T using equations (6.2), (6.3), (6.5), (6.9), (6.11);
            if $T$ is not dominated by any $T_x \in P_f$ then
                $P_f = P_f \cup T - \{T_y \mid T \prec T_y\}, \forall T_y \in P_f$;
                if size of $P_f > P_{as}$ then
                    Truncate $P_f$
                end if
            end if
        end for
        for all $T_k \in P_f$ do
            Update_global_pheromone $\forall (i, j) \in T_k$ using equations (6.35) and (6.36);
        end for
    end while
    {end resolve static EEMACOMH}
    Call Procedure ApplyMobilityChanges() (refer to Algorithm 12);
    Apply pheromone conservation $\forall (i, j) \in L$ using equations (6.38)-(6.41);
    $t = t + T_{sm}$
    Reset timer $T_{sm}$;
end while
Return $P_f$
```

COMP) is replaced with MMAS. The differences between EEMMASMP and EEMA-COMP with reference to MMAS are presented next.

From equation (3.3), the transition rule for EEMMASMP becomes

$$p_{ij}^k(t) = \begin{cases} \dfrac{\left(\tau_{\nu_{ij}}^{\alpha\lambda_\nu}(t)\eta_{\nu_{ij}}^{\beta_\nu}(t)+\tau_{\xi_{ij}}^{\alpha\lambda_\xi}(t)\eta_{\xi_{ij}}^{\beta_\xi}(t)+\tau_{\pi_{ij}}^{\alpha\lambda_\pi}(t)\eta_{\pi_{ij}}^{\beta_\pi}(t)+\tau_{\varrho_{ij}}^{\alpha\lambda_\varrho}(t)\eta_{\varrho_{ij}}^{\beta_\varrho}(t)+\tau_{\varsigma_{ij}}^{\alpha\lambda_\varsigma}(t)\eta_{\varsigma_{ij}}^{\beta_\varsigma}(t)\right)}{\sum_{\forall u\in N_i^k(t)}\left(\tau_{\nu_{iu}}^{\alpha\lambda_\nu}(t)\eta_{\nu_{iu}}^{\beta_\nu}(t)+\tau_{\xi_{iu}}^{\alpha\lambda_\xi}(t)\eta_{\xi_{iu}}^{\beta_\xi}(t)+\tau_{\pi_{iu}}^{\alpha\lambda_\pi}(t)\eta_{\pi_{iu}}^{\beta_\pi}(t)+\tau_{\varrho_{iu}}^{\alpha\lambda_\varrho}(t)\eta_{\varrho_{iu}}^{\beta_\varrho}(t)+\tau_{\varsigma_{iu}}^{\alpha\lambda_\varsigma}(t)\eta_{\varsigma_{iu}}^{\beta_\varsigma}(t)\right)} \\ \qquad\qquad \text{if } j \in N_i^k(t) \\ 0 \quad \text{otherwise} \end{cases} \tag{6.42}$$

where α defines the relative influence between the heuristic information and the pheromone levels. The other symbols are as defined in Section 6.6.2.

The solution construction process is as given in Section 6.6.1. Heuristic information matrices are calculated at the beginning of the simulation, and every time there is a change in the environment, using equations (6.13)-(6.17). Pheromone matrix initialisation and global updates are discussed next.

Pheromone matrix initialisation

The five pheromone matrices are initialised using

$$\tau_{\psi_{ij}}(t) = \tau_0 \tag{6.43}$$

for all $(i, j) \in L$, where ψ_{ij} represents either ν_{ij}, ξ_{ij}, π_{ij}, ϱ_{ij}, or ς_{ij} depending on the sub-objective; τ_0 is the upper bound set to some arbitrarily high value in order to achieve a high degree of exploration at the start of the algorithm.

Global pheromone update

Once all n_k ants have constructed solutions (refer to Section 6.6.1), the archive, P_f, is updated including the non-dominated solutions that have been found thus far. All the solutions from P_f are then selected to modify the pheromone trails. Each pheromone trail associated with each objective is updated using equations (6.24)-(6.25) (refer to Section 6.6.2).

If, after application of the global update rule, the pheromone value of a link becomes greater than the maximum value allowed, then the pheromone value of a link is explicitly set as equal to the maximum allowed value (refer to Section 3.5.3) for all pheromone matrices according to the following rules:

$$\text{if } \tau_{\psi_{ij}}(t+1) > \tau_{max_\psi} \text{ then } \tau_{\psi_{ij}}(t+1) = \tau_{max_\psi} \tag{6.44}$$

for all $(i,j) \in T_k$ and for all $T_k \in P_f$, and τ_{max_ψ} is calculated as proposed in Pinto and Barán [161]:

$$\tau_{max_\psi} = \sum_{T_k \in P_f} \frac{1}{(1-\rho_g)C(T_k)} \tag{6.45}$$

where $C(T_k)$ represents the corresponding objective function.

On the other hand, if the pheromone value of a link becomes less than the lower limit, then the pheromone value of a link is explicitly set equal to the lower limit (refer to Section 3.5.3) for all pheromone matrices according to the following rules:

$$\text{if } \tau_{\psi_{ij}}(t+1) < \tau_{min_\psi} \text{ then } \tau_{\psi_{ij}}(t+1) = \tau_{min_\psi} \tag{6.46}$$

for all $(i,j) \in T_k$ and for all $T_k \in P_f$, and τ_{min_ψ} is calculated as proposed in Pinto and Barán [161]:

$$\tau_{min_\psi} = \frac{\tau_{max_\psi}}{2n_k} \tag{6.47}$$

The EEMMASMP discussed above has been developed for static objective functions. EEMMASMP is adapted for dynamic objective functions in the same way as with EEMACOMP (refer to Section 6.6.2).

Algorithm 15 summarises the EEMMASMP algorithm.

Algorithm 15 General Procedure of EEMMASMP

```
$t = 0$; $P_f = \emptyset$; Set timer $T_{sm}$;
Initialize $s$, $D$, $\tau_0$, $\alpha$, $\beta_\nu$, $\beta_\xi$, $\beta_\pi$, $\beta_\varrho$, $\beta_\varsigma$, $\lambda_\nu$, $\lambda_\xi$, $\lambda_\pi$, $\lambda_\varrho$, $\lambda_\varsigma$;
Initialise $n_k$; {number of ants}
Initialise $P_{as}$; {Maximum archive size}
for each link $(i, j)$ do
    Initialise pheromone matrices $\tau_{\nu_{ij}}(t)$, $\tau_{\xi_{ij}}(t)$, $\tau_{\pi_{ij}}(t)$, $\tau_{\varrho_{ij}}(t)$, $\tau_{\varsigma_{ij}}(t)$ using equation (6.43);
    Calculate $\eta_{\nu_{ij}}$, $\eta_{\xi_{ij}}$, $\eta_{\pi_{ij}}$, $\eta_{\varrho_{ij}}$, $\eta_{\varsigma_{ij}}$;
end for
Place all ants, $k = 1, ..., n_k$ at source node $s$;
while $t <= S_{T_{tot}}$ do
    {begin resolve static EEMMASMP}
    while $T_{sm}$ seconds not elapsed do
        for $k = 1$ to $n_k$ do
            $T = \emptyset$; $i = s$; {where $s$ the source node and $i$ the current node}
            while $(i \neq D)$ do
                Build set $N_i^k(t)$ for $i$;
                Assign probability $p_{ij}^k$ to each node of $N_i^k(t)$ according to equation (6.42);
                Select node $j$ of $N_i^k(t)$ using equation (6.42);
                $T = T \cup j$ ;
                $i = j$;
            end while
            Evaluate the sub-objectives for solution T using equations (6.2), (6.3), (6.5), (6.9), (6.11);
            if $T$ is not dominated by any $T_x \in P_f$ then
                $P_f = P_f \cup T - \{T_y \mid T \prec T_y\}, \forall T_y \in P_f$;
                if size of $P_f > P_{as}$ then
                    Truncate $P_f$
                end if
            end if
        end for
        for all $T_k \in P_f$ do
            update_global_pheromone $\forall(i, j) \in T_k$ using equations (6.24)-(6.25);
            Restrict the pheromone intensities within the lower and upper limits, $\forall(i, j) \in T_k$, using
            equations (6.44)-(6.47);
        end for
    end while
    {end resolve static EEMMASMP}
    Call Procedure ApplyMobilityChanges() (refer to Algorithm 12);
    Apply pheromone conservation $\forall(i, j) \in L$ using equations (6.27)-(6.30);
    Restrict the pheromone intensities within the lower and upper limits, $\forall(i, j), \in L$ using equations
    (6.44)-(6.47);
    $t = t + T_{sm}$;
    Reset timer $T_{sm}$;
end while
Return $P_f$
```

6.6.5 Energy Efficiency Using Multi-objective MAX-MIN Ant System Optimisation, Multi-heuristic Algorithm

In accordance with the max-min ant system (MMAS) (refer to Section 3.5.3) the energy efficiency for mobile networks using a multi-objective MAX-MIN ant system optimisation, multi-heuristic (EEMMASMH) algorithm uses a colony of ants, a single pheromone matrix and a different heuristic matrix for each objective. EEMMASMH is very similar to the EEMMASMP algorithm, the sole difference being that EEMMASMH uses only one pheromone matrix, $\tau_{ij}(t)$. This single pheromone matrix represents the desirability of the solution components with regard to all objectives.

From equation (3.3), the transition rule for EEMMASMH becomes

$$p_{ij}^{k}(t) = \begin{cases} \frac{\tau_{ij}^{\alpha}(t)\left(\eta_{\nu_{ij}}^{\beta_\nu}(t)+\eta_{\xi_{ij}}^{\beta_\xi}(t)+\eta_{\pi_{ij}}^{\beta_\pi}(t)+\eta_{\varrho_{ij}}^{\beta_\varrho}(t)+\eta_{\varsigma_{ij}}^{\beta_\varsigma}(t)\right)}{\sum_{\forall u \in N_i^k(t)} \tau_{iu}^{\alpha}(t)\left(\eta_{\nu_{iu}}^{\beta_\nu}(t)+\eta_{\xi_{iu}}^{\beta_\xi}(t)+\eta_{\pi_{iu}}^{\beta_\pi}(t)+\eta_{\varrho_{iu}}^{\beta_\varrho}(t)+\eta_{\varsigma_{iu}}^{\beta_\varsigma}(t)\right)} \\ \quad \text{if } j \in N_i^k(t) \\ 0 \quad \text{otherwise} \end{cases} \tag{6.48}$$

where the symbols are as defined in Section 6.6.2.

The pheromone matrix is initialised as

$$\tau_{ij}(t) = \tau_0 \tag{6.49}$$

for all $(i, j) \in L$, where τ_0 is the upper bound set to some arbitrarily high value in order to achieve a high degree of exploration at the start of the algorithm.

The equations for global pheromone updating are the same as those of EEMACOMH (refer to equations (6.35) and (6.36)).

If, after applying the global update rule, $\tau_{ij}(t+1)$ becomes greater than τ_{max}, then $\tau_{ij}(t+1)$ is explicitly set equal to τ_{max} according to the following rule:

$$\text{if } \tau_{ij}(t+1) > \tau_{max} \text{ then } \tau_{ij}(t+1) = \tau_{max} \tag{6.50}$$

for all $(i, j) \in T_k$ and for all $T_k \in P_f$, where

$$\tau_{max} = \sum_{T_k \in P_f} \frac{1}{(1-\rho_g)(EP(T_k) + TNP(T_k) + VF(T_k) + CP(T_k) + MNC(T_k))} \tag{6.51}$$

On the other hand, if $\tau_{ij}(t+1) < \tau_{min}$ then $\tau_{ij}(t+1)$ is explicitly set equal to τ_{min} according to the following rule:

$$\text{if } \tau_{ij}(t+1) < \tau_{min} \text{ then } \tau_{ij}(t+1) = \tau_{min} \tag{6.52}$$

for all $(i, j) \in T_k$ and for all $T_k \in P_f$, where

$$\tau_{min} = \frac{\tau_{max}}{2n_k} \tag{6.53}$$

The EEMMASMH discussed above has been developed for static objective functions. EEMMASMH is adapted for dynamic objective functions in the same way as with EEMACOMH (refer to Section 6.6.3).

Algorithm 16 summarises the EEMMASMH algorithm.

6.6.6 Energy Efficiency Using Multi-Objective Ant Colony Optimisation, Multi-Colony Algorithm

The multiple colony ACO developed by Gambardella *et al.* [76] and described in Section 4.5.6 was used to solve MOPs by assigning a colony to each objective. This section proposes the energy efficiency for mobile networks using a multi-objective ant colony optimisation, multi-colony (EEMACOMC) algorithm which applies the multi-colony concept to the power-aware routing problem.

Since five objectives are defined for the power-aware routing problem, five colonies are created – one for each of the objectives. These colonies each have the same number of ants, $n_{k_c} = n_k/5$, with $c = 1, ..., 5$.

Each colony implements an ACS, where the transition rule for each sub-objective is

$$j = \begin{cases} arg\ \max_{u \in N_i^k(t)} \{\tau_{\psi_{iu}}(t) \eta_{\psi_{iu}}^{\beta_\psi}(t)\} & \text{if } r \leq r_0, \\ J & \text{otherwise} \end{cases} \tag{6.54}$$

where β_ψ represents either β_ν, β_ξ, β_π, β_ϱ, or β_ς depending on the sub-objective, and

Algorithm 16 General Procedure of EEMMASMH

```
$t = 0$; $P_f = \emptyset$; Set timer $T_{sm}$;
Initialise $s$, $D$, $\tau_0$, $\alpha$, $\beta_\nu$, $\beta_\xi$, $\beta_\pi$, $\beta_\varrho$, $\beta_\varsigma$;
Initialize $n_k$; {number of ants}
Initialize $P_{as}$; {Maximum archive size}
for each link $(i, j)$ do
    Initialise pheromone matrix $\tau_{ij}(t)$, using equation (6.49);
    Calculate $\eta_{\nu_{ij}}$, $\eta_{\xi_{ij}}$, $\eta_{\pi_{ij}}$, $\eta_{\varrho_{ij}}$, $\eta_{\varsigma_{ij}}$;
end for
Place all ants, $k = 1, ..., n_k$ at source node $s$;
while $t <= S_{T_{tot}}$ do
    {begin resolve static EEMMASMH}
    while $T_{sm}$ seconds not elapsed do
        for $k = 1$ to $n_k$ do
            $T = \emptyset$; $i = s$; {where $s$ the source node and $i$ the current node}
            while $(i \neq D)$ do
                Build set $N_i^k(t)$ for $i$;
                Assign probability $p_{ij}^k$ to each node of $N_i^k(t)$ according to equation (6.48);
                Select node $j$ of $N_i^k(t)$ using equation (6.48);
                $T = T \cup j$ ;
                $i = j$;
            end while
            Evaluate the sub-objectives for solution T using equations (6.2), (6.3), (6.5), (6.9), (6.11);
            if $T$ is not dominated by any $T_x \in P_f$ then
                $P_f = P_f \cup T - \{T_y \mid T \prec T_y\}, \forall T_y \in P_f$;
                if size of $P_f > P_{as}$ then
                    Truncate $P_f$
                end if
            end if
        end for
        for all $T_k \in P_f$ do
            update_global_pheromone $\forall(i, j) \in T_k$ using equations (6.35) and (6.36);
            Restrict the pheromone intensities within the lower and upper limits, $\forall(i, j) \in T_k$, using
            equations (6.50)-(6.53);
        end for
    end while
    {end resolve static EEMMASMH}
    Call Procedure ApplyMobilityChanges() (refer to Algorithm 12);
    Apply pheromone conservation $\forall(i, j) \in L$ using equations (6.38)-(6.41);
    Restrict the pheromone intensities within the lower and upper limits, $\forall(i, j) \in L$ using equations
    (6.50)-(6.53);
    $t = t + T_{sm}$;
    Reset timer $T_{sm}$;
end while
Return $P_f$
```

defines the importance of $\eta_{\psi_{iu}}(t)$. The rest of the symbols are as defined in Section 3.5.2. $J \in N_i^k(t)$ is a node that is randomly selected according to probability,

$$p_{iJ}^k(t) = \frac{\tau_{\psi_{iJ}}(t)\eta_{\psi_{iJ}}^{\beta_\psi}(t)}{\sum_{u \in N_i^k(t)} \tau_{\psi_{iu}}(t)\eta_{\psi_{iu}}^{\beta_\psi}(t)} \tag{6.55}$$

At every iteration of the ACS algorithm each ant of a colony constructs a complete solution (optimising the objective of the colony) as follows (refer to Algorithm 17): Starting from the source node, s, a non-visited node is pseudo-randomly selected at each step as in equation (6.54), while equation (6.55) provides the probability of selecting a link. This process continues until the destination node, D, is reached. When a new node has been added to a candidate solution, a local pheromone update is performed using equation (6.26) (refer to Section 3.5.2). The solution created by the ant is inserted into the global archive P_G without being evaluated. The archive, P_G, contains all the candidate solutions.

Algorithm 18 summarises the general procedure for constructing $n_k/5$ solutions for all the ants of a colony.

Algorithm 17 General Procedure of BuildPathMultiColony

```
input parameters $s$, $D$, $r_0$, $\beta_\psi$, $\tau_\psi$, $\eta_\psi$;
$T = \emptyset$;
$i = s$; {where $s$ the source node and $i$ the current node}
**while** $(i \neq D)$ **do**
    Build set $N_i^k(t)$ for $i$;
    Assign probability $p_{ij}^k$ to each node of $N_i^k(t)$ according to equation (6.54);
    Select node $j$ of $N_i^k(t)$ using equations (6.54) and (6.55);
    $T = T \cup j$;
    Apply local pheromone update using equation (6.26);
    $i = j$;
**end while**
Return T;
```

The five pheromone matrices are initialised using equations (6.22)-(6.23) (refer to Section 6.6.2). The cooperation between colonies and the global pheromone updates are discussed next.

Algorithm 18 General Procedure of BuildAllPathsMultiColony

```
input parameters $s$, $D$, $r_0$, $\beta_\psi$, $\tau_\psi$, $\eta_\psi$, $n_k$;
for $k = 1$ to $n_k/5$ do
    T = BuildPathMultiColony($s$, $D$, $r_0$, $\beta_\psi$, $\tau_\psi$, $\eta_\psi$);
    Insert T in to global archive, $P_G$;
end for
Return $P_G$
```

Cooperation between colonies

The five colonies of ants cooperate and specialise in order to find good solutions in different regions of the Pareto front. In every iteration, the ants in each colony deposit their solutions into a global solution pool (called P_G) that is shared by all colonies. This pool of solutions, P_G, is used to determine the non-dominated front of all solutions from all the colonies in that iteration. Once all the ants from all colonies have constructed their solutions and deposited them in P_G, all the solutions of P_G are evaluated against the sub-objectives using equations (6.2), (6.3), (6.5), (6.9), (6.11). Every solution, T, of P_G is compared with the solutions of P_f and, if T is non-dominated by any solution of P_f, then T is inserted into P_f. All the solutions of P_f which are dominated by T are removed from P_f. If the number of non-dominated solutions exceeds the predefined archive size, P_{as}, a truncation operator is used to remove solutions with a lower value of the crowding distance from the archive (refer to Section 4.6.2). The use of the crowding distance guides the selection process during the various stages of the algorithm towards a uniformly spread-out Pareto-optimal front.

Global pheromone update

Only those ants that found a solution which is in the global non-dominated front are allowed to update pheromones. Each ant uses the update by origin method (refer to Section 4.5.6) to determine in which colony the ant should update the pheromone matrix, that is, the ant updates only in its own colony [104]. The update by origin method imposes a stronger selection pressure on those ants that are allowed to update. The update by origin method might also force the colonies to search in different regions of the non-dominated front.

When only a few solutions from other colonies are in the same region, it is more likely that a solution from the local non-dominated front of a colony might also appear in the global non-dominated front. Hence, it is more likely that an ant with solutions in less dense areas of the non-dominated front will be allowed to update and, thereby, influence the ensuing search process [104].

The ants of each colony that found a solution which is in the global non-dominated front, P_f, update only the pheromone matrix which is associated with the objective of the colony (refer to Section 4.5.6).

The pheromone trail of each colony is updated using the solutions of P_f as follows:

$$\tau_{\psi_{ij}}(t+1) = (1-\rho_g)\tau_{\psi_{ij}}(t) + \rho_g \Delta^{\psi}\tau_{ij}(t) \tag{6.56}$$

where ψ_{ij} represents either ν_{ij}, ξ_{ij}, π_{ij}, ϱ_{ij}, or ς_{ij} depending on the sub-objective, and

$$\Delta^{\psi}\tau_{ij}(t) = \begin{cases} \frac{1}{C(T_k)} & \text{if } i,j \in T_k \text{ and } T_k \in \text{colony } n_c \\ 0 & \text{otherwise} \end{cases} \tag{6.57}$$

for all $T_k \in P_f$, where $C(T_k)$ represents the corresponding objective function, and $n_c \in \{1, ..., 5\}$ represents the index of the colony associated with each objective.

The EEMACOMC algorithm discussed above has been developed for static objective functions. EEMACOMC is adapted for dynamic objective functions in the same way as for EEMACOMP (refer to Section 6.6.2).

Algorithm 19 summarises the dynamic EEMACOMC.

Algorithm 19 General Procedure of EEMACOMC

```
$t = 0$; $P_f = \emptyset$; Set timer $T_{sm}$;
Initialise $s$, $D$, $r_0$, $\beta_\nu$, $\beta_\xi$, $\beta_\pi$, $\beta_\varrho$, $\beta_\varsigma$;
Initialise $n_k$; {number of ants}
Initialise $\tau_{0\nu}$, $\tau_{0\xi}$, $\tau_{0\pi}$, $\tau_{0\varrho}$, $\tau_{0\varsigma}$;
Initialise $P_{as}$; {Maximum archive size}
for each link $(i, j)$ do
  Initialise pheromone matrices $\tau_{\nu_{ij}}(t), \tau_{\xi_{ij}}(t), \tau_{\pi_{ij}}(t), \tau_{\varrho_{ij}}(t), \tau_{\varsigma_{ij}}(t)$ using equations (6.22)-(6.23);
end for
for each link $(i, j)$ do
  calculate $\eta_{\nu_{ij}}, \eta_{\xi_{ij}}, \eta_{\pi_{ij}}, \eta_{\varrho_{ij}}, \eta_{\varsigma_{ij}}$;
end for
Place all ants, $k = 1, ..., n_k$ at source node $s$;
while $t <= S_{T_{tot}}$ do
  {begin resolve static EEMACOMC}
  while $T_{sm}$ seconds not elapsed do
    $P_G = \emptyset$;
    $P_G = P_G \cup BuildAllPathsMultiColony(s, D, r_0, \beta_\nu, \tau_\nu, \eta_\nu, n_k)$;
    $P_G = P_G \cup BuildAllPathsMultiColony(s, D, r_0, \beta_\xi, \tau_\xi, \eta_\xi, n_k)$;
    $P_G = P_G \cup BuildAllPathsMultiColony(s, D, r_0, \beta_\pi, \tau_\pi, \eta_\pi, n_k)$;
    $P_G = P_G \cup BuildAllPathsMultiColony(s, D, r_0, \beta_\varrho, \tau_\varrho, \eta_\varrho, n_k)$;
    $P_G = P_G \cup BuildAllPathsMultiColony(s, D, r_0, \beta_\varsigma, \tau_\varsigma, \eta_\varsigma, n_k)$;
    for all $T \in P_G$ do
      Evaluate the sub-objectives for solution T using equations (6.2), (6.3), (6.5), (6.9), (6.11);
      if $T$ is not dominated by any $T_x \in P_f$ then
        $P_f = P_f \cup T - \{T_y \mid T \prec T_y\}, \forall T_y \in P_f$;
        if size of $P_f > P_{as}$ then
          Truncate $P_f$;
        end if
      end if
    end for
    for all $T_k \in P_f$ do
      Update_global_pheromone $\forall(i, j) \in T_k$ using equations (6.56) and (6.57);
    end for
  end while
  {end resolve static EEMACOMC}
  Call Procedure ApplyMobilityChanges() (refer to Algorithm 12);
  Apply pheromone conservation $\forall(i, j) \in L$ using equations (6.27)-(6.30);
  $t = t + T_{sm}$; Reset timer $T_{sm}$;
end while
Return $P_f$
```

6.7 Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Power-Aware Routing

NSGA-II has not yet been applied to the multi-objective power-aware routing problem. This section proposes for the first time to solve the multi-objective power-aware routing problem using an adaptation of the NSGA-II algorithm called NSGA-II multi-objective power-aware algorithm (NSGA-II-MPA).

NSGA-II-MPA modifies the NSGA-II procedure in tracking a new Pareto-optimal front as soon as there is a change in the multi-objective power-aware routing problem. The change in the problem is introduced with the application of the mobility model. Hence, the position of the nodes changes and, as a result, there is a change in objective functions.

As with the five ant algorithms proposed in this thesis, the dynamic multi-objective power-aware routing problem is based on the principle of dividing the simulation time, $S_{T_{tot}}$, into n_{ts} time slices of equal length, $S_{T_{tot}}/T_{sm}$. T_{sm} is the length of the pause time for the mobility model. T_{sm} is an indication of frequency of change (refer to Section 5.1). During each time slice a static problem is created, and optimisation is carried out using the standard NSGA-II. For each of these static problems the aim is to simultaneously minimise the five objectives and to create a population, P_t, with several non-domination fronts that have been found within the time slice. The population, P_t, may be passed on to the static problem corresponding to the following time slice since the two problems would potentially be very similar. This operation would prevent optimisation from having to restart each time from scratch and would contribute greatly to the good performance of the NSGA-II-MPA algorithm. To introduce diversity into the non-dominated solutions obtained by the NSGA-II-MPA algorithm, a number of random solutions are added whenever there is a change in the problem. The number of random solutions is equal to a percentage of the population, P_t. When this percentage of random solutions increases, the performance of NSGA-II-MPA deteriorates. More generations are needed to track the new optimal front. At each change, the P_t archive is re-evaluated and non-dominated sorting is applied.

Considering the power-aware routing problem formulation as given in Section 6.4.1, the NSGA-II-PMA is described in more detail below.

The first iteration consists of the following three steps:

1. **Initialise all parameters**.

2. **Build routing tables**. A procedure for building the routing tables is executed to build possible paths from the source, s, to the destination, D, of the network. This procedure selects the R paths with minimum energy consumed per packet and with minimum cost per packet where R is a parameter of the algorithm. The k shortest path algorithm is used to select the R paths [222].

3. **Calculate initial population,** P_0.

   From the routing tables, $|P_0|$ different random chromosomes are generated. This set of chromosomes is termed the chromosome pool, P_0 (or population), and it forms the first generation. Duplicated solutions in the population are replaced with new randomly generated solutions. A chromosome is represented by a binary string of size $\log_2(R)$, in order to represent R different paths. Each chromosome represents a possible route (path) between the source node, s, and the destination node, D.

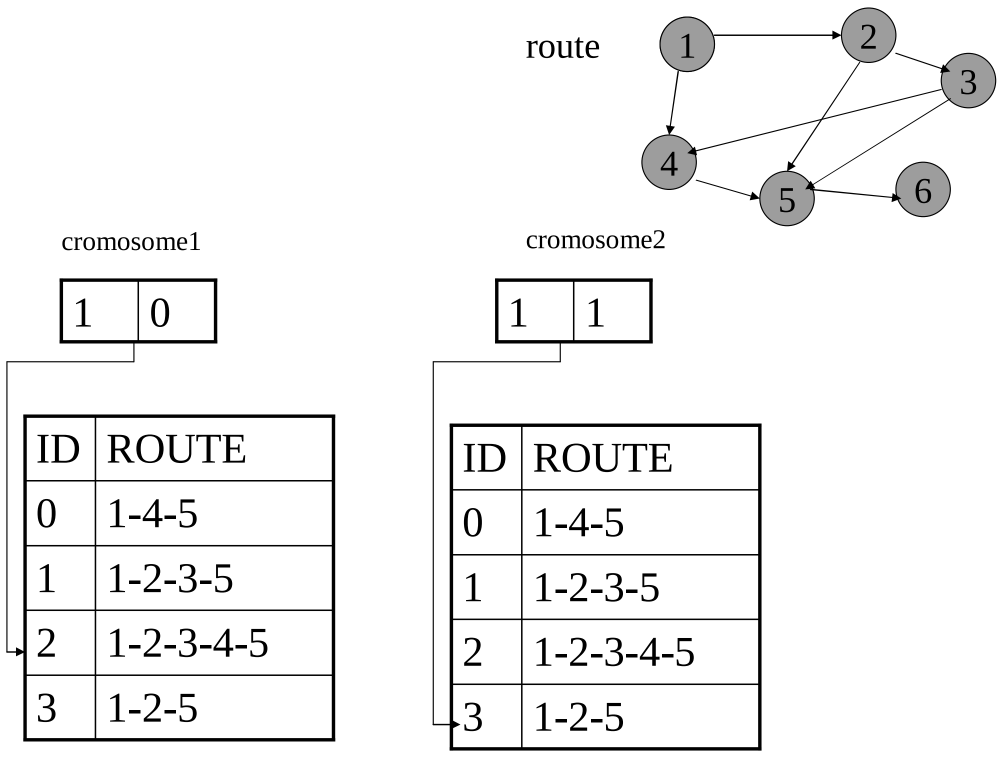

Figure 6.4: Routing tables and chromosomes

In the example of Figure 6.4 the routing table contains 4 paths ($R = 4$) with minimum energy consumed per packet and minimum cost per packet from $s = 1$ to $D = 5$. Each chromosome consists of a binary string of size 2 ($\log_2(4)$) and points to a specific path of the routing table. For example, the chromosome with a binary value of 10 has a decimal value of 2 and points to the path with $id = 2$ (path $1-2-3-4-5$).

Parents are selected from the population by using binary tournament selection based on the rank and crowding distance. Out of two individuals the individual with the lowest rank is selected, or, if both individuals have the same rank, the individual with the greater crowding distance is selected (refer to Section 4.6.2). The population selected generates offspring using crossover and mutation operators. Using the genetic operators, a child population Q_0 of size N_p is created.

The simulation (main loop) is run for $S_{T_{tot}}$ seconds with the following instructions: Within the time slice, T_{sm}, the maximum number of allowed generations are used to find the optimal population.

At each iteration, t, the following steps, as for the standard NSGA-II (refer to Section 4.6.2) are applied: The population, P_t, is evaluated against the five sub-objectives. Using binary tournament selection on parent population, P_t, of size N_p, based on the rank and crowding distance and applying crossover and mutation operators, Q_t offspring, of size N_p, are created. The combined population, $R_t = P_t \cup Q_t$, of size $2N_p$, is sorted into different non-domination levels. Individuals are then selected from this combined population to be inserted into the new population on the basis of their non-domination level. If there are more individuals in the last front than there are slots remaining in the new population of size N_p, a diversity preserving mechanism based on rank and the crowding distance on the last front is employed. Individuals from this last front are placed in the new population, P_{t+1}, on the basis of their contribution to diversity in the population.

At the end of time slice, T_{sm}, the ApplyMobilityChangesNSGA procedure, as given in Algorithm 20, and RebuildRoutesUpdatePopulation as given in Algorithm 21, handle the dynamic aspect of the power-aware routing problem.

The ApplyMobilityChangesNSGA procedure sends one packet from the source to the destination. In order to send the packet, a route T_{best} from P_{t+1} is used. T_{best} is the individual with rank 1 and the largest crowding distance. The RPGM mobility model (refer to Section 6.5) is applied. Energy levels and distances are recalculated (refer to Section 6.5).

The RebuildRoutesUpdatePopulation procedure applies the following modifications to the original NSGA-II procedure, whenever a change in the multi-objective power-aware routing problem occurs:

- All parent solutions are re-evaluated before merging the parent and child populations into a bigger pool. This process allows both offspring and parent solutions to be evaluated using the changed sub-objectives.

- New random solutions are introduced. If N_e represents the first N_e solutions of the population, P_t, a percentage, $\varpi = \frac{N_p - N_e}{N_p}\%$, of the new population is replaced with randomly created solutions which helps to improve exploration. The first N_e solutions of P_t are kept in the archive to maintain elitism. The population, P_t, is completed up to the maximum size N_p, using $N_p - N_e$ new chromosomes generated based on routing tables which are created using the k shortest path algorithm. The final P_t is passed onto the next static problem.

Algorithm 20 General Procedure of ApplyMobilityChangesNSGA

```
Choose the best route, $T_{best}$, from $P_t$;
Send one packet from source to destination using $T_{best}$;
for all nodes $i \in T_{best}$ do
    Update $e_i^c$ according to equation (6.18);
end for
Apply the RPGM mobility model (refer to Section 6.5);
for all links $(i, j) \in L$ do
    Recalculate $d_{ij}$;
    Recalculate $E_{ij}$ according to equation (6.19);
end for
```

The algorithm moves to the next time slice optimising the new static problem until the end of the simulation.

The NSGA-II algorithm for multi-objective power-aware routing problem is summarised in Algorithm 22.

Algorithm 21 General Procedure for RebuildRoutesUpdatePopulation

```
Delete invalid routes associated with $P_t$, i.e. routes for which $E_{ij} > e_i^c$ or $d_{ij} > T_r$;
for all $T \in P_t$ do
    Re-evaluate the sub-objectives using equations (6.2), (6.3), (6.5), (6.9), (6.11);
end for
Apply non-dominated-sort on $P_t$; {refer to Section 4.6.2, Algorithm 9}
Find the first $N_e$ individuals from $P_t$ based on the rank and crowding distance; {An individual is selected if the rank is lesser than the other, or if the rank is the same and the crowding distance is greater than that of the other individuals}
The population $P_e$ with $|P_e| = N_e$ is created;
The corresponding paths, $T_{N_e}$, are selected; {$P_e$ represent the elitist solutions}
Build routing table $T_{N_l}$ with $N_l = N_p - N_e$ paths from $s$ to $D$; {Using the $k$ shortest path algorithm}
Complete the routing table, $T_{N_p} = T_{N_l} \cup T_{N_e}$;
From $T_{N_p}$ generate $N_p$ different chromosomes, which form the new generation $P_t$;
Using the genetic operators on $P_t$, a child population $Q_t$ of size $N_p$ is created;
```

6.8 Summary

This chapter presented five new ant-based algorithms to solve the power-aware routing problem. Versions of each algorithm have been developed assuming that the optimisation problem is static. Each algorithm is then adapted to also solve the power-aware routing problem in changing environments under the RPGM mobility model. The chapter also presented an adaptation to the NSGA-II to solve the power-aware routing problem.

The next chapter empirically analyses the five algorithms, and compares their performance to that of the NSGA-II.

Algorithm 22 General Procedure of NSGA-II for the Multi-objective Power-Aware Routing Problem

```
Initialise $E_i$, $e_i^c$, $\forall i \in V$;
Calculate $E_{ij}$, $d_{ij}$, $\forall (i,j) \in L$;
Create initial routing tables; {Using the $k$ shortest path algorithm}
Create a random population, $P_0$, from the routing tables;
Evaluate the sub-objectives for all $T \in P_0$ using equations (6.2), (6.3), (6.5), (6.9), (6.11);
$\mathcal{Z}$ = non-dominated-sort($P_0$); {refer to Section 4.6.2, Algorithm 9}
Use binary tournament selection, recombination, and mutation operators to create a child
population $Q_0$ of size $N_p$;
$simulation\_time = 0$; Set timer $T_{sm}$;
while $simulation\_time <= S_{T_{tot}}$ do
    $t = 0$;
    {begin resolving static NSGA-II}
    while $T_{sm}$ seconds not elapsed do
        for all $T \in P_t$ do
            Re-evaluate the sub-objectives using equations (6.2), (6.3), (6.5), (6.9), (6.11);
        end for
        $R_t = P_t \cup Q_t$; {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 $R_t$}
        $P_{t+1} = \emptyset$;
        $i = 1$;
        {untill the parent population is filled}
        while $|P_{t+1}| + |\mathcal{Z}_i| <= N_p$ do
            crowding-distance-assignment($\mathcal{Z}_i$); {calculate crowding distance in $\mathcal{Z}_i$ using Algo-
            rithm 10};
            $P_{t+1} = P_{t+1} \cup \mathcal{Z}_i$; {include $i$-th non-dominated front in the parent population}
            $i = i + 1$; {check the next front for inclusion}
        end while
        Sort($\mathcal{Z}_i, \prec_n$); {sort in descending order using the crowded comparison operator, $\prec_n$}
        $P_{t+1} = P_{t+1} \cup \mathcal{Z}_i[1 : (N_p - |P_{t+1}|)]$; {Choose the first $(N_p - |P_{t+1}|)$ elements of $\mathcal{Z}_i$}
        $Q_{t+1}$ = 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
    {end resolving static NSGA-II}
    Call Procedure ApplyMobilityChangesNSGA() (refer to Algorithm 20);
    Call procedure RebuildRoutesUpdatePopulation() (refer to Algorithm 21);
    $simulation\_time = simulation\_time + T_{sm}$;
    $P_0 = P_t$;
    $Q_0 = Q_t$;
    Reset $T_{sm}$;
end while
Return $P_t$;
```