Modeling Transportation Problems Using Concepts of Swarm Intelligence and Soft Computing

by

Panta Lučić

Dissertation submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Civil Engineering

Dušan Teodorović, Chair

Hugh VanLandingham

Antonio Trani

Konstantinos Triantis

Paul Schonfeld

February 2002

Falls Church, Virginia

Keywords: Transportation Engineering, Swarm Intelligence, Fuzzy Sets Theory

Modeling Transportation Problems Using Concepts of Swarm Intelligence and Soft Computing

Panta Lučić
(ABSTRACT)

Many real-world problems could be formulated in a way to fit the necessary form for discrete optimization. Discrete optimization problems can be solved by numerous different techniques that have developed over time. Some of the techniques provide optimal solution(s) to the problem and some of them give "good enough" solution(s). The fundamental reason for developing techniques capable of producing solutions that are not necessarily optimal is the fact that many discrete optimization problems are NP-complete. Metaheuristic algorithms are a common name for a set of general-purpose techniques developed to provide solution(s) to the problems associated with discrete optimization. Mostly the techniques are based on natural metaphors. Discrete optimization could be applied to countless problems in transportation engineering.

Recently, researchers started studying the behavior of social insects (ants) in an attempt to use the swarm intelligence concept to develop artificial systems with the ability to search a problem's solution space in a way that is similar to the foraging search by a colony of social insects. The development of artificial systems does not entail the complete imitation of natural systems, but explores them in search of ideas for modeling.

This research is partially devoted to the development of a new system based on the foraging behavior of bee colonies – *Bee System*. The Bee System was tested through many instances of the Traveling Salesman Problem.

Many transportation-engineering problems, besides being of combinatorial nature, are characterized by uncertainty. In order to address these problems, the second part of the research is devoted to development of the algorithms that combine the existing results in the area of swarm intelligence (The Ant System) and approximate reasoning. The proposed approach – *Fuzzy Ant System* is tested on the following two examples: Stochastic Vehicle Routing Problem and Schedule Synchronization in Public Transit.

Acknowledgements

I was fortunate to work with a research group of professors and graduate students in Transportation Infrastructure and Systems Engineering in the Charles Edward Via, Jr. Department of Civil and Environmental Engineering at Virginia Tech since August 1999. I have spent the entire period of the time at both campuses; the main campus in Blacksburg and at the Northern Virginia Graduate Center.

I am greatly indebted to Dr. Dušan Teodorović, Dr. Hugh VanLandingham, Dr. Antonio Trani, Dr. Konstantinos Triantis and Dr. Paul Schonfeld for being members of my dissertation advisory committee, and for their support and assistance.

I would like to extend a very special thank you to my committee chairman, Dr. Dušan Teodorović. I have known him for more than a decade. We have collaborated successfully from the last phase of my undergraduate work at the University of Belgrade. He has been my academic advisor for my pursuit of an M.S. degree at the same institution. He was always available to help me – to give his time and guidance.

I would like to thank Dr. Antonio Trani for giving me an opportunity to work with him and for the financial support provided during one phase of my study.

I would like to express my appreciation to all my friends who have stood by me during my stay at Virginia Tech. Without them my time here would have been completely different.

Finally, I am grateful to my parents for so many things, and to my wife Ivana for her love, understanding and support, without which I would have never gone through the dissertation adventure.

Contents

List of Figures

List of Tables

Chapter 1. Introduction

1.1 Motivation

The possibility of modeling many real-world problems as discrete optimization problems creates a need to develop tools capable of solving such problems quickly and efficiently. Discrete optimization problems are characterized by a countably finite solution space (feasible region) as well as a value of performance index (objective function) assigned to each feasible solution. Discrete optimization finds the best (optimal) solution(s) based on objective function values, in the whole of global feasible region of the problem. Many discrete optimization problems are NP-hard, meaning that there is no algorithm that can solve such problems in polynomial time with respect to problem size.

There are a number of techniques available to solve discrete optimization problems optimally. It could be seen that a great number of practical real-world problems were formulated and solved using optimization techniques during the last four decades. However, it is important to note that the majority of real-world problems solved by some of the optimization techniques were of small dimensionality.

Many traffic and transportation engineering problems are discrete optimization problems. Most of them are difficult to solve either because of the large dimensionality or because it is very difficult to decompose them into smaller sub-problems. Typical examples of these problems are vehicle fleet planning, static and dynamic routing and scheduling of vehicles and crews for airlines, railroads, truck operations and public transportation services, designing transportation networks, optimizing alignments for highways and public transportation routes through complex geographic spaces, different location problems, etc.

Many times in real life we only need a "good" solution. In other words, very often the decision makers are satisfied with *a suboptimal solution* obtained when *heuristic* algorithm is applied.

In recent years, metaheuristic algorithms have increasingly been used in solving difficult combinatorial optimization problems. In the beginning, metaheuristics included the

following technique: *simulated annealing* (Metropolis et al. (1953), Kirkpatrick et al. (1983), Cerny (1985)), *genetic algorithms* (Holland (1975), Goldberg (1989), and *tabu search* (Glover (1986), Glover and Laguna (1993)). The successful application of emergent techniques based on natural metaphors, such as simulated annealing, genetic algorithms, and neural networks, to complex engineering problems is certainly encouraging; it most definitely points to the natural systems as a source of ideas and models for the development of various artificial systems. In recent years, many different techniques have been created to perform better than others with respect to produced criteria value (to be as close as possible to global extreme) as well as required computer time to obtain a solution.

Recently developed techniques with the general purpose of solving optimization problems in various areas are rooted in nature. A subset of the techniques has been developed based on concepts that are taken from results of studying the behavior of social insects. Among the various behaviors, it has been shown that foraging behavior is very important for developing a variety of artificial systems that could be used to solve optimization problems. One part of this article is devoted to the idea of applying some concepts of natural swarm intelligence to develop artificial ones. It is important to underline the fact that researchers have used just ideas (required major points) to develop artificial systems based on natural ones. It is not important and in some cases could be useless to simply copy all discovered properties of natural systems to artificial ones.

A wide range of traffic and transportation engineering parameters are characterized by uncertainty, subjectivity, imprecision, and ambiguity (Teodorović (1994, 1999), Teodorović and Vukadinović (1998)). Human operators, dispatchers, drivers, and passengers use subjective knowledge or linguistic information on a daily basis while they make decisions. Drivers, passengers, or dispatchers make decisions about route choice, mode of transportation or/and most suitable departure time. In each case the decision maker is a human. The environment in which a human expert (human controller) makes decisions is often complex, making it difficult to formulate a suitable mathematical model. Thus, the development of fuzzy logic systems seems justified in such situations (Zadeh (1965), Zimmermann (1991), Kosko (1992, 1993), Mendel (1995)). Developing

models for solving difficult combinatorial optimization problems characterized by *uncertainty* is a very important and challenging research task. Metaheuristic techniques need to be combined with Fuzzy Sets Theory Techniques for solving complex traffic and transportation engineering problems characterized by uncertainty.

When using the ideas taken from natural swarm intelligence, taking into account foraging behavior of the individual agents, how to model the behavior is still an open question. Many other methods such as binary logic have to be explored. The final decision on where to place food search among available fields will be made based on some probability functions. One part of the dissertation is contributed to this area.

1.2 Research Goals

The ways in which natural systems process information are still unmatched by current computers. They learn to recognize relevant patterns, they remember patterns that have been seen previously, they construct internal models, and finally, their data and control structures are extremely parallel and highly distributed. Examples of such systems are common in biology (including neuroscience, vision, immunology and ecology), social systems, physics and chemistry.

The qualities that a social system should necessarily possess to have biological intelligence are as follows:

- perception (what is happening in the surrounding area),
- memory (not necessarily but most frequently to memorize perceptions),
- usage of gained knowledge (abstraction of essential characteristics that input data can posses) to plan appropriate behavior,
- communication, and
- learning.

One important direction that has been explored widely in last decade is the attempt to use the concept of swarm intelligence to develop various artificial systems. Researchers have started studying the behavior of social insects to take the most promising concepts from

their natural behavior (individual and/or collective), with the purpose of developing artificial systems that perform well. The development of artificial systems does not entail the complete imitation of natural systems, but it explores natural systems for various ideas and models.

It should be noted that a large number of traditional engineering models and algorithms are based on control and centralization. On the other hand, bee or ant swarm behavior in nature is primarily characterized by autonomy, distributed functioning and self-organizing. In natural systems, it is found that very simple individual organisms are capable of performing highly complex tasks by dynamically interacting with each other. It is of course of great importance to investigate both advantages and disadvantages of autonomy, distributed functioning and self-organizing with respect to traditional engineering methods that rely on control and centralization. The basic question about the above-mentioned characteristics of social insects that should be answered is:

Can we use some principles of natural swarm intelligence in the development of artificial systems aimed at solving complex problems in traffic and transportation?

The first goal of this dissertation is to explore, and to illustrate with examples in the transportation area, the possibilities of usage of bees' behavior for developing artificial systems with the purpose of providing a "good" solution to difficult combinatorial optimization problems in general.

One of the most important results of the Artificial Systems development, based on Swarm Intelligence, was the creation of the Ant System. Artificial ants, proposed by Colorni et al. (1991, 1992), search the solution space, simulating real ants looking for food in the environment. The objective function values correspond to the quality of food sources. Existence of adaptive memory characterizes the searching process. In this case, the adaptive memory corresponds to the pheromone trails. Colorni et al. (1991, 1992) demonstrated an Ant System through its application to the Traveling Salesman Problem. They showed that natural ants foraging behavior could be utilized with certain modifications in an optimization field. This could be done as follows: Artificial ants would explore the feasible region of the considered problem to find the best possible

solution, as the natural ants would explore their surrounding environment to find the best reachable food source. Many successful implementations of the Ant System through the last decade have proven the possibility of usage of this technique in solving discrete optimization problems. The main idea was modeling of an individual ant's behavior in the environment of foraging ant colony. To model individual decisions, Colorni et al. (1991, 1992) have proposed that ants will make individual decisions based on certain probabilities. The proposed way of calculating those probabilities did not allow treating problems characterized by uncertainty.

Many transportation-engineering problems that belong to the discrete optimization area are also characterized by uncertainty. Is it possible to expand existing Ant System (Colorni et al., 1991, 1992) to address the problems characterized by uncertainty?

The second goal of this dissertation is to explore the possibilities of solving discrete optimization problems characterized by uncertainty using Ant systems enriched with Fuzzy Logic. Each artificial ant will use an approximate reasoning algorithm in order to make individual decisions. The Fuzzy-Ant System should be produced and tested on examples taken from the transportation area.

1.3 Organization of the Dissertation

The remainder of this dissertation is organized as follows. Chapter 2 provides a review of literature relevant to this research. Practically, the literature review contains an overview of discrete optimization problems as well as a brief description of techniques developed based on metaphors taken from nature.

Chapter 3, through description of real bees behavior, will introduce the basic ideas that will be used for future development of artificial bees' behavior. An account of the usage of artificial bees' behavior for developing the Bee System – tool that could be used in solving discrete optimization problems is provided later. An approach will be illustrated on the Traveling Salesman Problem and the Stochastic Vehicle Routing Problem. The

last problem will be solved by sequential application of the Bee System and Fuzzy Logic system where a fuzzy rule base is developed from numerical examples. Additionally, basic information related to Fuzzy Logic will be provided.

Chapter 4 will provide an exhaustive review of natural ants' behavior as well as Ant System development. The Fuzzy-Ant System has been developed as a combination of the "classical" Ant System and Fuzzy Logic to be a tool for solving discrete optimization problems characterized by uncertainty. This approach has been tested on the following two examples: The Vehicle Routing Problem and Scheduling Synchronization in Public Transit.

Finally, Chapter 5 provides a summary and conclusions, along with recommendations for future research.

Chapter 2. Literature Review

2.1 Introduction

Developing algorithms that utilize some analogies with natural and social systems to derive non-deterministic heuristics methods capable of obtaining "very good" results in hard combinatorial optimization problems could prove to be a promising field of research. This chapter presents an overview of existing algorithms in this area of interest as well as a brief discussion of the literature devoted to the development and applications of these algorithms. Algorithms that use natural metaphors derived from physics, biology and social science are the focus of interest.

Entire population of algorithms are obtained based on the following (Colorni et al., 1996):

- repeated trials,
- agents (particles, chromosomes, neurons, ants, bees, etc.),
- in case of multiple agents operating mechanism: competition – cooperation,
- introduced procedure for modification of the heuristic's parameters or of the problem representation.

The basic characteristic of heuristics from nature could be summarized as follows (Colorni et al., 1996):

- they model a phenomenon existing in nature,
- they are stochastic,
- in case of multiple agents, they often have parallel structure,
- they use feedback information for modifying their own parameters – they are adaptive.

Let us introduce *the landscape of the problem* $\mathcal{L}$. Let ε, ω and θ *be as* follows:

- ε - the set of all points of the search space,
- ω - the operator that is utilized by search algorithm,
- $\theta = (\varepsilon, E)$ – the graph whose nodes (vertices) are the points in the search space

and edges connect point x with point x_1 in case where point x_1 could be obtained by applying operator ω on x,

The geometrical object that could be obtained by assigning the performance index as an altitude of nodes (vertices) of θ is landscape of the problem $\mathcal{L}$.

In cases where some solution of the problem is known (such as one point in the landscape) it is possible to walk/jump through the field from one solution to another in order to find a better solution to the problem. Steps through the solution space should be well organized, in order to find a "good enough" or optimal solution in the least possible number of moves.

It is possible to obtain different agent based heuristics established on the following propositions (Colorni et al., 1996):

- In case of one or several agents (located in different positions in solution space) it is possible to use a greedy technique to choose each move.
- In case some solution exists, it is possible to improve it by making "small perturbations" using local search techniques.
- Make "small perturbations" in a random manner and accept only the ones with improvement.
- In case of a non-improving perturbation, it is possible to give the solution a chance to be accepted using some non-deterministic rule for accepting.
- After every "small perturbation", improve system memory if it exists, to direct the search process into regions that are not yet explored.

Algorithms can be classified based on a variety of characteristics. Some of them are as follows:

- Algorithms could be used to produce a solution or just improve an existing solution.
- Algorithms dealing with one solution or population of solutions.
- Searching process has employed memory or memory is not employed.

This chapter presents an overview of the literature devoted to development and application of the algorithms that use natural metaphors derived from physics, biology and social science, designed to find a "good" solution to discrete optimization problems (not necessarily optimal) in a reasonable amount of CPU time.

2.2 Discrete Optimization Problems

Discrete optimization is the process of analyzing and finding a solution for problems mathematically modeled as the minimization or maximization of a measure, over a feasible space involving mutually exclusive logical constraints (Parker and Rardin, 1988).

In their most abstract mathematical form, discrete optimization problems can be presented in the following way:

$$\begin{aligned} &\min \text{ (or max)} \quad \varphi(S) \\ &\text{subject to} \quad S \in F \end{aligned}$$

where S is the solution (arrangement), F is the collection of feasible solutions (arrangements), and $\varphi(S)$ measures the value of members of F.

Discrete optimization is the selection of the arrangement (solution) with the best performance index (measure) among mutually exclusive alternatives.

Discrete optimization problems can also be given in the following form:

$$\begin{aligned} &\min \text{ (or max)} \ c^t x \\ &\text{subject to:} \\ &\qquad A x \geq b \\ &\qquad x \geq 0 \\ &\qquad x \text{ - integer} \end{aligned}$$

This is the Integer Programming formulation of the problem. Any Discrete Optimization problem can be given in this way.

Solving discrete optimization problems, i.e. finding an optimal solution to such problems can be a difficult task. Difficulty arises from the fact that the feasible region is not necessarily (like in linear programming problems) a convex set.

There is a significant number of possible approaches to the problem. Some of them will give optimal solutions (they could be: enumerative techniques, relaxation and decomposition techniques or cutting planes approaches based on polyhedral combinatorics) and some of them will direct the search process in a way that finds a good enough or near-optimal solution in a reasonable amount of CPU time.

2.3 Simulated Annealing

Simulated annealing (SA) as a technique was mentioned for the first time in the literature written by Metropolis et al. (1953). The authors are physicists, and they simulated the cooling material in a heat bath to obtain the lowest possible energy states of the particles of material. This process was called annealing.

At the very beginning of the annealing process (solid) material is melted. During the annealing process its temperature is slowly reduced. It takes very long time for the material's temperature to become close to the freezing point. The cooling of the material happened under a particular cooling schedule until convergence to steady state (a near-frozen condition) occurs.

It is possible, at any temperature, to change the total energy of the material with small displacements of particles. At one temperature, this process could be called *perturbation*. Several perturbations may occur at one temperature, and each of them gives different total energy level.

The question is, what perturbation will give the lowest total energy level at one temperature?

The following procedure provides a possibility for how to solve the above mentioned problem. After perturbation, calculate the magnitude of energy change (ΔE). In cases where $\Delta E < 0$, the new position of the particles has the lowest energy level and it will become the new initial position of particles for making another perturbation. In cases

where $\Delta E > 0$, the new position of the particles has a higher level of energy. There is a probability (probability increases with temperature) that after perturbation of the (worst) configuration of these particles, significant energy decrement takes place. Metropolis et al. (1953) used the Boltzmann distribution to calculate the probability of acceptance in energy sense worst perturbation:

$$p_{acc} = e^{-\frac{\Delta E}{k T}}$$

where:

ΔE – magnitude of energy change,

k – a constant,

T – current temperature.

At each temperature it is possible to make a large number of perturbations, but after some number of perturbations, the magnitude of change in the total material's energy is very small. At that point the material achieves *thermal equilibrium* for that fixed temperature. Thermal equilibrium means that a material has achieved the lowest possible energy level at that fixed temperature. The next step is to decrease the temperature and make perturbations again in order to reach thermal equilibrium at that temperature. The procedure is finished when the searching process for the last temperature (the lowest value specified) is done.

Before applying this algorithm, the following values should be defined:

- the total temperature number (if change between temperatures is step function),
- initial temperature value, and
- amount of change in energy that is acceptable as very low. It is necessary to define when the thermal equilibrium occurs.

The following authors: Kirkpatrick, Gellat and Vecchi (1983), and independently, Cherny (1985) were the first to start with implementation of some kind of modification of this technique in order to solve problems that belongs to the discrete optimization area.

The basic idea is to develop some kind of random searching strategy that starts from one possible solution (variable and criteria values) and jumps through the neighborhood of the solution (in the feasible region) in order to obtain a new starting point (for a new jump) using modification of the previously described procedure. It is possible that old

and new starting points are the same. In general, through these jumps, the search process will arrive into area(s) with better solutions.

There is an analogy between the Physical Cooling Process and the methodology for solving discrete optimization problems:

Physical Process	Meta-heuristic
Energy	Criteria value
Temperature	Control parameter
Particle configuration	Feasible solution

Eglese (1990) offers a very simple pseudo-code for Simulated Annealing algorithm:

```
Select an initial state i ∈ S;
Select an initial temperature T > 0;
Set temperature change counter t := 0;
Repeat
        Set repetition counter n := 0;
        Repeat
                Generate state j, a neighbor of i;
                Calculate δ := f(j) – f(i)
                if δ < 0 then i := j
                else if random (0, 1) < exp (-δ/T) then i:= j;
                Inc(n);
        Until n = N(t);
        Inc(t);
        T:= T(t);
Until stopping criterion true.
```

where:

S – finite solution set,
i – previous solution,
j – next solution,
f(*x*) – criteria value for solution *x*, and
N(*t*) – number of perturbation at the same temperature.

2.4 Evolutionary Algorithms

The first work related to this area was done in the late 1950's. However, development and utilization of this area has only become significant during the last decade (Back, et al., 1997).

Evolutionary Algorithms (EA) contains set of procedures-techniques used for solving

difficult combinatorial optimization problems. All of them are developed based on natural evolution processes.

The following are the most common EAs:

- Genetic Algorithms,
- Scatter Search,
- Ant Systems,
- Adaptive Memory Algorithms.

The basic idea of EA can be described using the following pseudo-code (Hertz and Kobler, 2000):

```
Generate an initial population of individuals;
While no stopping condition is met do
        Co-operation;
        Self-adaptation;
End While
```

Individuals can be solutions of a particular problem or they can be pieces of solutions (with the idea being to gather the pieces in order to obtain one feasible solution) or even sets of solutions (as found in parallel implementation, for example: island-based genetic algorithms).

Co-operation is the part of the algorithm where two or more individuals in the population are identified to exchange information.
Self-adaptation is the part of the algorithm where every individual or some of them (usually randomly selected) is modified independently.
Each of the techniques of EA, mentioned above, will be discussed briefly later.

2.4.1 Genetic Algorithms

Genetic Algorithms (GA) is a technique inspired by biological processes that allow populations of organisms to adapt to their surrounding environment.
The earliest papers published in this area are by Holland (1962), Rechenberg (1965), Fogel et al. (1966) and Holland (1975).

Basically, GA works with three operators:

Selection operator determines which individuals (solutions in optimization problem) will survive in the population (set of solutions). That means only the subset of individuals produced in the previous generation will be used to produce the next generation of individuals. There are several methods for applying the Selection operator, and of those, the most efficient are the methods based on some kind of random choice. The probability of choosing one individual depends on how good the individual is (usually shown through objective function value).

Crossover is an operator that combines two individuals in order to make two new (different) individuals. Crossover can be based on one or several identical cuts over both individuals and exchanges of "material" (information) between individuals.

Mutation operator introduces a little *noise* in the population with the idea to prevent fast convergence that may end up finding the local optimum. Operator mutation can be applied in a very small number of individuals. To apply the operator, any of the individuals can be chosen, but usually with a very small probability (range 10^{-3}).

These operators are used sequentially through many iteration in order to make the population evolve.

Hertz and Kobler (2000) offer a very simple pseudo-code for genetic algorithms:

Chose an even integer $p \geq 2$ and generate an initial population P_0 of p individuals;
Set i:= 0; (iteration counter);
While no stopping criteria is met **do**[1]
    Inc (i) and initialize P_i to the empty set;
    While P_i has less than p individuals **do**[2]
        Select two individuals I_1 and I_2 in P_{i-1};
        Apply the crossover operator to I_1 and I_2 for creating offspring O_1 and O_2;
        Add O_1 and O_2 to P_i;
    End While
    Apply the mutation operator to each individual in P_i;
End While

Before implementation of GA, it is necessary to code a set of feasible solutions in order to make sure that new generation individuals created are always in this set after applying

[1] Stopping criteria can be satisfied objective value and/or number of generation.
[2] Through all generation developed models keep constant *p*. This value can be function of number of generation.

Crossover and Mutation operators. This part can be the hardest in the area of applying GA for solving discrete optimization problems.

2.4.1.1 Island-Based Genetic Algorithms

The paper written by Gordon and Whitley (1993) is the first paper published in this area. Island-Based GA is an improvement of GA. In GA applications, it usual that individuals are solutions of the considered problem. In that light, the first difference between Island-based GA and GA is the fact that Island-based GA works with sets of solutions as individuals. The basic idea is to distribute set of solutions (population in GA) into subpopulations that evolve independently (Calegari et al., 1997). Existence of these subpopulations (islands or demes) increases the chances for the algorithm to find good solutions by having several GA operating on small populations (subpopulations) simultaneously. In order to allow some islands to benefit from the information that has been founded by others, some solutions can migrate from one island to another. In this way we have introduced a new operator in GA called *migration*.

There are several ways to define the migration operator. One of them is as follows:

Let us imagine that islands are virtually positioned on an oriented ring and migrations are allowed only along the ring. Every time the new generation is computed (CPU time can be subdivided onto several workstations – one island on each), a copy of the best solution (defined based on criteria value) ever found by each island is sent to the next island on the ring. In this way every island will receive a new solution, the best from the previous island on the ring that would replace the randomly chosen solution (local individual).

Pseudo-code for Island-based GA (Hertz and Kobler, 2000):

Choose the number k of islands and the size p of each island;
Generate a set of $k \cdot p$ initial feasible solutions and partition this set into k islands $P_1, P_2, \ldots P_k$;
Set i:= 0;
While no stopping criteria is met **do**
    Set i:= i + 1;
    For each island P_j **do**
        Select two solutions I_1 and I_2 in P_j;
        Apply the crossover operator to I_1 and I_2 for creating offspring O_1 and O_2;
        Apply the mutation operator and improvement algorithm (if exist) on O_1 and O_2;

Decide whether or not O_1 and O_2 should enter P_j for replacing older solutions;
End For
If *i* is multiple of a given integer *n* **then**
Move the best solution of each island P_j to island $P_{(jmodk)+1}$;
End While

Calegari et al. (1997) have obtained much better results with Island-based GA than with GA. They have tested both on the same example with initial populations of the same size. The question was whether it is necessary to subdivide the initial population into set of subpopulations. They have discovered that the number of islands has a significant effect on the quality of the results obtained by Island-based GA.

For implementing Island-based GA, the following questions need to be answered in addition to the common questions considered in GA (Cantu-Paz and Goldberg, 2000):

- the size and number of islands (demes),
- the topology of the connections between the islands and
- how many individuals migrate each time.

2.4.2 Scatter Search

Proposed by Glover (1994), scatter search is a search strategy based on two sets of points in a solution space. The first is a set of *reference points* from which iteration would generate set of *dispersed points* that may act as reference points along with some of the previous reference points for the next iteration. Every point in the set of dispersed points is defined through several steps:

1. make linear combination of subset of the current reference points (*trial point* will be defined),
2. apply repair procedure if necessary (trial point can be out of feasible region and repair procedure should make that point feasible),
3. apply improvement algorithm in order to get point with higher quality.

In order to generate a linear combination of the reference points, it is even possible to apply negative weights (more general case).

Pseudo code for the Scatter search technique (Hertz and Kobler, 2000):

Generate an initial set R_0 of reference points;
Set i:= 0;
While no stopping criteria is met **do**
    Inc(i);
    Determine a set T_i of trial points by making linear combinations of points in R_{i-1};
    Transforms the trial points in T_i into set F_i of feasible points;
    Improve the points in F_i in order to obtain a set D_i of dispersed points;
    Select points in$R_{i-1} \cup D_i$ to be put in the new set R_i of reference points;
End While

2.4.3 Ant Systems

The algorithms that are defined by Colorni et al. (1991, 1992) are models derived from the study of real ant colonies.

Ant Systems (AS) contain artificial ants that have some major differences with real (natural) ones:

- artificial ants will have some memory,
- they will not be completely blind,
- they will live in an environment where time is discrete.

When applying AS, search in the feasible region will happen at discrete time points. Artificial ants are able to define one part of a solution per iteration through the searching process. After finishing the set of iterations, every ant would have finished one solution of the considered problem. The next step is to exchange some information that will allow all the ants to benefit from the solutions developed by all other ants through the searching process. Thus, one cycle is finished through the searching process. The algorithm ends after a certain number of cycles.
Ants will make movements along iterations in a random manner. Probabilities for possible steps in the next iteration are calculated based on *visibility* (available local information) and *pheromone trail* (exchange of information among ants).

Simple pseudo code for AS technique (Hertz and Kobler, 2000):

Initialize pheromone trails and define number of ants;
While stopping criteria is not met **do**
    Build solution based on visibility and pheromone trail for every ant;
    Update pheromone trail;

Apply improvement algorithm on each new solution if such algorithm exist;
End While

More detailed information about development of the AS algorithms would be provided at the beginning of chapter 4.

2.4.4 Adaptive Memory Algorithms

Adaptive memory algorithms were recently developed (Rochat and Taillard (1995), Golden et al. (1997)) based on the Tabu Search technique. This technique works with central memory that keeps track of the best components of the solutions visited during the search process. These components are combined in order to create a new solution. When a new solution is not feasible, it is changed using the repair procedure. Also, on each new feasible solution, some improvement procedure like the tabu search or some other technique is applied. Then some components from all new solutions are selected to be the candidates to replace old components in the central memory.

Pseudo code for Adaptive memory algorithms (Hertz and Kobler, 2000):

Generate a set of solutions and introduce their components in the central memory;
While no stopping criteria is met **do**
Combine components of the central memory in order to create new solutions;
Use repair procedure on each infeasible new solution;
Apply an improvement algorithm on each new solution;
Update the central memory by removing some components and introducing new ones originating from the new feasible solutions;
End While

2.5 Tabu Search

Tabu Search (TS) is a general heuristic procedure for guiding a search to obtain a good solution in a complex solution space (Glover, 1993). This technique is able to escape from local extreme points and to search areas beyond local extreme points. TS is developed based on some concepts taken from artificial intelligence. TS was introduced

by Glover (1986) and independently, Hansen (1986).

TS, like all other metaheuristics, is a good tool for solving the following kind of problem:

$$\min_{x \in X} \quad F(x)$$

The basic idea is to obtain an initial solution ($x_{in} \in X$) – it can be obtained in various ways – and reach a solution that is good enough through a certain number of iterations. From one iteration to another, two solutions usually exist: current and the best found. In that light, there are some similarities between SA and TS.

If the current solution at the beginning of iteration *i* is x_i then it is possible to obtain a new solution that belongs to X with one modification (*move*) of x_i. This solution is the neighbor solution to the x_i. Let us introduce the set of all neighbor solutions of solution x_i as $N(x_i)$. For set $N(x_i)$ there are conditions: $N(x_i) \subset X$ and $x_i \notin N(x_i)$. Sometimes it is very hard (time consuming) to generate all solutions that belong to $N(x_i)$. Let's introduce $N'(x_i) \subseteq N(x_i)$. Sometimes $\left|N'(x_i)\right|$ can be equal to 1.

In one iteration (*i*), there is a local optimization problem to find a solution x_{i+1} that yields $\min \left\{ F(x_{i+1}) \,\middle|\, x_{i+1} \in N'(x_i) \subseteq N(x_i) \right\}$. This new solution will become the current solution and the same actions are repeated, even if a new solution is worse than previous (based on criteria value). In order to avoid cycling (repetition), TS contains a *history list* and a *tabu list*. Both are parts of the adaptive memory structure that *penalizes* or *forbids* certain move(s) that lead the searching process to recently visited solutions.

Dammeyer and Voβ (1993) offer a very simple pseudo-code for TS. One modification of the code is as follows:

Set iteration counter $i := 1$;
Select an initial state $x_i \in X$ and initial objective value $F(x_i)$;
Let $x^* := x_i$ and $F^* := F(x_i)$;
While stopping criterion is not fulfilled **do**
    Select best admissible move (based on history and tabu lists entities) that transforms x_i into x_{i+1} with objective function value $F(x_{i+1})$;
    Perform tabu list management: update history and tabu lists;
    Replace solution x_i with x_{i+1} and objective value $F(x_i)$ with $F(x_{i+1})$;
    If $F(x_{i+1}) > F^*$ then $x^* := x_{i+1}$ and $F^* := F(x_{i+1})$;
    Inc (*i*);
End While

where:

x^* - the best solution found through the entire searching process.

Pham and Karaboga (2000) describes strategies of TS algorithms:
The forbidding strategy is employed to avoid stopping the searching process in some area by forbidding certain moves (put them as tabu). A tabu list can contain just the few last moves (even only one) or a large number of moves (extreme is all of them). The number of iterations where one move is forbidden is called the *length of tabu list* (T_s). The probability of cycling among the same solutions is highly dependent on T_s.
Making moves that are tabu is allowed when the aspiration criteria is satisfied. The criteria can be time independent (earlier applications) or dependent (new applications). While the tabu list has a role in constraining the search space, the aspiration criterion has a role in guiding the search process.

2.6 Current Research

The main differences between traditional mathematical methods to solve some kind of problems and metaheuristics can be determined based on solution optimality and CPU time. The beauty of traditional mathematical methods is that they provide an optimal solution to the stated problem. However, when large size instances are treated, the CPU time required to achieve an optimal solution becomes an issue. In addition to that, traditional approaches most often require simplifications of the original problem formulation. Some times due to oversimplifications, computed solution does not solve the original problem. Furthermore, one of the fundamental advantages of the metaheuristics approach over traditional mathematical methods is its capability to adapt to the considered problem (Back et al., 1997). All metaheuristics should not be considered as ready to use algorithms but rather as a general concept that can be applied to most of the real world applications.
The possibility to adapt to the problem as well as lower required CPU time to obtain a "good enough" solution encourages researchers to explore the potential to make the

existing techniques available in a variety of areas (expand the capabilities of these techniques to make them capable of solving a variety of problems) and develop new ones.

From the literature, it is possible to separate the following few directions of the current research:

- application of metaheuristics and comparison among them,
- expansion in the development of the existing algorithms,
- development of new techniques, and
- convergence analysis.

2.6.1 Application of metaheuristics and comparison among them

Application oriented research in this area has been quite successful. Only a few application domains could be identified, if any, where these algorithms have not been tested so far (Back et al., 1997).

Some authors in their articles, in addition to presenting the developed applications, have tried to compare several techniques to find the most suitable one.

Youssef et al. (2001) give a comparison of SA, GA and TS based on the floorplanning problem. The authors discovered that the best performance was given by the TS algorithm with respect to obtained solutions. Furthermore, with respect to the complexity of implementation and tuning of algorithm parameters, they have found that GA requires the highest effort.

Hasan et al. (2000) have compared SA, GA and TS in terms of solution quality and CPU time for their application to the unconstrained 0-1 Pseudo-Boolean quadratic problems. They have found that general performance of the TS algorithm compared to the performances of GA and SA was unsatisfactory with respect to both criteria. Furthermore, they conclude about the superiority of the GA method over SA and TS in Quadratic Programming (QP) applications.

Wolpert and Macready (1997) proved a number of theorems stating that the average performance of any pair of iterative (deterministic or non deterministic) algorithms across

all problems is identical.

Taking into account any metaheuristic algorithm, it is clear that there are several different parameters that should be adjusted. For some parameters there is an adjusting procedure and for some there is not. Second ones are assigned subjectively estimated values. In addition, it is important to say once again, all metaheuristics are general concepts. In that light there is some part in implementation that could be called the "art of modeling".

It seems that there is no need for this kind of comparison. There is a possibility that someone will present, on the same problems, even on the same instances, different recommendations later.

2.6.2 Expansion in development of the existing algorithms

One of the most important reasons for the existence of metaheuristics is the fact that the time that is necessary to achieve a "good enough" problem solution is significantly lower than the time for traditional mathematical methods. However, for some larger instances of any problem, CPU time is still "high". In literature, one can find a lot of papers related to metaheuristics where authors tried to develop procedures that would allow parallel searching process. In other words, if CPU time is high, let us define the random searching algorithm in a way to allow simultaneous searching processes at several computers.

In the case of population based algorithms, it is not so hard to imagine how the search process can be subdivided into several parallel processes. In case of SA and TS it is not the case because they always operate with one solution (single point search techniques).

Onbasoglu and Ozdamar (2001) have presented the basic concepts of Parallel SA. They have used two different approaches to develop various categories of SA algorithms as follows:

- the asynchronous approach where no information is exchanged among parallel runs,
- the synchronous approaches where solutions are
    - exchanged using genetic operators,

- transmitted occasionally,
- highly coupled (synchronization is achieved at every iteration).

First work in this field has been summarized in Aarts and Korst (1989).

Gordon and Whitley (1993) have subdivided all modification of GA into the following two groups:

- "traditional" GA,
- parallel GA
  - Global GA,
  - Island GA, and
  - Cellular GA (in literature known as massively parallel GA or fine grain GA).

The second part presents research directions devoted to speeding up the searching process.

There have been several successful attempts to develop a Parallel TS algorithm. Fiechter (1994) has proposed a method based on the Parallel TS algorithm to solve large TSP instances.

Another direction in this area is the development of algorithms that will give "better" solutions that usually take a larger amount of CPU time. Those algorithms were produced based on a combination of existing algorithms, and they are known as *hybrid* algorithms in literature. One of the most important types of hybrid algorithms is *memetic* algorithms (MA). MA are population based heuristic search algorithms for optimization problems similar to GA. In contrast to GA, MA mimic cultural evolution. While, in nature, *genes* are usually not modified during an individual's lifetime, *memes* are. MAs are developed when individuals in GA (solutions to the problem) are improved by the local search technique such as SA, TS or any other. In addition, improvement is usually done at the very beginning through the entire first generation, and every time during the mutation phase.

Burke and Smith (2000) provide one application of this technique to a maintenance

scheduling problem. They have tested several improvement techniques (local optimizers) and for the considered problem they have found that MA with TS as a local optimizer produced the best results.
Hybrid algorithms could be developed as a combination of metaheuristics and optimization techniques. Budenbender et al. (2000) have successfully developed a hybrid of tabu search and branch and bound algorithm with application to the direct flight network design problem.

There are a lot of papers that tried to offer improvements for all metaheuristics through applications. Those improvements are related to, for example, visiting distribution in SA offering different expressions for acceptance probability, etc.
From literature, we can find a strong relationship between metaheuristics (specially evolutionary algorithms) and some other techniques like fuzzy logic and neural networks (Back et al., 1997).

2.6.3 Development of New Metaheuristics

The number of publications in the area of metaheuristics is still high. "We can observe a remarkable and steady (still exponential) increase in the number of publications" in the EA area (Back et al., 1997). With the introduction of memetic algorithms, and even without, in other areas of metaheuristics we can find a high amount of publications. Furthermore, it is important to say that new stochastic search procedures are being created. Researchers are trying to find better algorithms, more suitable for some kinds of problems. Successful implementation of some techniques highly depends on the problem that is being considered (comparison of the techniques on different problems will give different rank of techniques).
Here are some recent results laid out:
Hansen and Mladenović (2001) have described, in detail, a *Variable neighborhood search* (VNS) technique that has been introduced by the same authors in the year 1996.

A simple pseudo code of a VNS algorithm can be presented in the following way (Hansen and Mladenović, 2001):

Select the set of neighborhood structures N_k, k= 1, 2, …, k_{max}, that will be used in the search; find initial solution x; choose a stopping condition;
While stopping condition is not met **do**
    Set $k = 1$;
    While $k <> k_{max}$ **do**
        Shaking. Generate a point x' at random from the k-th neighborhood of x ($x' \in N_k(x)$);
        Local search. Apply some local search method with x' as initial solution; denote with x" obtained local optimum;
        Move or not. If this local optimum is better than the incumbent, move there (x = x"), and continue search with $N_1(k=1)$; otherwise, set Inc(k)
    End While
End While

Boettcher and Percus (2000) have introduced a new method called Extremal Optimization. This procedure successfully eliminates extremely undesirable components of sub-optimal solutions. They have tested the algorithm on TSP instances up to 256 nodes, and they have concluded that they arrived at better solutions than the ones they had gotten using SA or GA on the same instances.

This material would describe some outcomes obtained as a result of our work in the development of new metaheuristics. Chapter 3 is devoted to the development of the random search technique based on bees' behavior; Bee System.

An artificial immune system could potentially be a new approach to the hard combinatorial optimization problems. King et al. (2001) provide a biological basis for this system. They have applied the artificial immune system in the area of computers stems (control parallel programs during their execution and monitoring their performance).
In addition, it is important to mention that most probably there are some other works that have been done, and it is not stated here. There is a tremendous number of publications in this area and it is hard to cover everyone's work precisely.

2.6.4 Convergence Analysis

In contrast to practical results, the theoretical foundations are to some extent still weak. More precisely, "We know that they work, but we do not know why." Back et al. (1997). Some theoretical results about convergence fortunately exist, and here some of them are briefly pointed out.

- Simulated Annealing

  Theoretical results exist for the SA algorithm modeled as a homogenous Markov chain or an inhomogeneous Markov chain (Sullivan, 1999). Aarts and Laarhoven (1985) have produced the first theoretical result related to the convergence of SA.

  Locatelli (2000) proved that convergence exists in the case of SA application on continuous global optimization problems. The author states several assumptions to prove that convergence exists.

  Stainhofel et al. (2000) applied a logarithmic cooling schedule of inhomogeneous Markov chains to the flow shop scheduling problem. They prove a lower bound for the number of steps, which are sufficient to approach an optimum solution with certain probability.

- Genetic Algorithms

  Holland (1975) has presented the Schema Theorem stating that individual solutions with good, low order schema (similar beneficial parts among solutions) should be evaluated and allowed to crossover in an exponentially increasing number of successive populations.

  Eiben et al. (1991) use Markov chain analysis to obtain a unifying theory for SA and GA, such that any SA or GA application at hand is an instance of developed abstract algorithm.

  Agapie (1997) has used homogenous Markov chain modeling to provide a set of minimal sufficient conditions for convergence to the global optimum of Elitist GA (EGA).

- Tabu Search

  The least number of papers in this area were published with TS consideration. However, some theoretical results exist; Faigle and Kern (1992) provide a convergence result for probabilistic versions of TS. Probabilistic TS incorporates the acceptance function of SA into the TS framework. The authors prove that this kind of TS converges asymptotically.

2.7 Artificial Neural Networks

One of the areas based on natural metaphor that has been studied widely is Artificial Neural Networks (ANN). The human brain performs some complex tasks relatively easily compared to traditional algorithms and computational techniques. The architecture of the brain is different from common serial computers. Researchers interested in ANN seek possible ways to produce machines with abilities as close as possible to the human brain. ANN are inspired by biology; they are composed of elements with functionality similar to a biological neurons. Even the most optimistic supporters do not state that ANN will soon mimic the entire functionality of the human brain. However, today's level of development proves promising for the future of the technique.
Basically, ANN are considered as approximators because of their ability to approximate unknown functions with a certain degree of accuracy.

2.7.1 The Biological Neuron

The most basic element of the human brain is a specific type of cell – neuron. The human brain contains approximately 10^{11} neurons each having the ability to receive, process and transmit electrochemical signals. Neurons are connected; the process of receiving and transmitting of signals is allowed between connected neurons. A simplified scheme of the biological neuron structure is shown on figure 2.1.

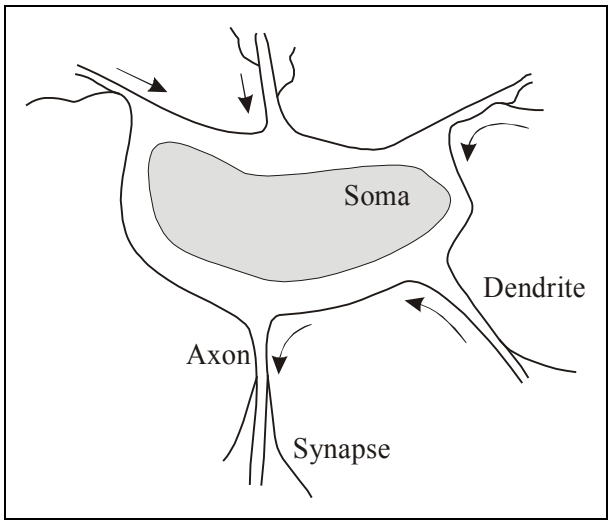

Figure 2.1 – Biological neuron – basic structures

A neuron contains a cell body (*soma*) and several branches that accept input information to the cell (*dendrites*), or transmit information out of the cell (*axons*). Interaction among the neurons takes place at the connection points – *synapses*. In cases where stimulus exceeds a certain threshold, the neuron will transmit a signal through the axon to other neurons. Transmitted signals highly depend on received signals from all other neurons connected to the current neuron, as well as the strength of their connectivity. The transmitted signal is considered to be a weighted summation of the received inputs.

Basic characteristics of the human brain are: ability to learn and generalization of knowledge that has been obtained. Generalization refers to generation of similar response on similar inputs.

2.7.2 The Artificial Neuron

The basic unit of ANN is the artificial neuron. Artificial neurons simulate properties of the biological neurons. That means artificial neurons will receive some input signals (outputs of adjacent artificial neurons) and transmit some output signals. The output signals of connected neurons are denoted by x_1, x_2, …, x_n. Considering that an output signal is a weighted summation of the input signals, the output signal (*NET*) could be defined as follows:

$$NET = w_1x_1 + w_2x_2 + \quad + w_nx_n$$

where w_i denotes the strength of connection between artificial neuron *i* and the neuron under consideration.

A graphical representation of the simple structure of an artificial neuron is presented in figure 2.2.

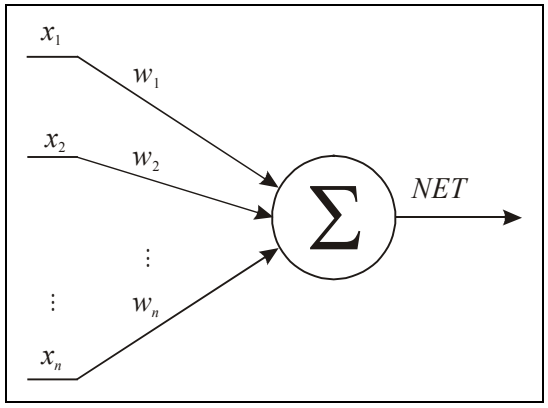

Figure 2.2 – Artificial neuron structure

The first artificial neuron was proposed by McCulloch and Pitts (1943). The developed artificial neuron had binary input, binary output and a fixed activation threshold.

In order to process signals of different strength, usually an activation function is applied on the output signal. Activation functions most commonly used are (Teodorović and Vukadinović, 1998): nonlinear, continuous, monotonously increasing, bounded, differentiable logistic function.

2.7.3 ANN Design

The design of ANN is an iterative process containing, most frequently, a trial and error procedure before coming up with a satisfactory design. For the design of ANN, one should determine the following elements:

- set of artificial neurons and their arrangement in various layers,

- type of connections among neurons for different layers as well as among the neurons within the same layer,
- rule of signal propagation through the network,
- activation functions,
- training algorithm.

ANN contains three types of nodes: input, output and hidden. The nodes are collected into layers. Input nodes are collected into the input layer, output nodes in the output layer, and the others into one or several layers in between. Input nodes receive signals from a source that is outside of the ANN. Output nodes transmit signals outside of the ANN that could be used for an input for some other algorithm or making a decision. Signals that occur on input and output of any other nodes will be the internal property of the ANN and will not be "visible" from the outside. Based on the input signal obtained at the input layer, all the other nodes will receive and transmit signals of different strengths to adjacent nodes (adjacent to one node are all nodes connected to it). At the very end, signal progression will come to nodes in the output layer and they will provide an output signal.

Connections among nodes are commonly unidirectional. One node will receive output signals from one group of adjacent nodes and transmit its own signal as an input to the other adjacent nodes. With respect to connectivity among nodes, in some ANN, there is a possibility that only inter-layer and some intra-layer connectivity could exist.

ANNs could be classified as follows (Teodorović and Vukadinović, 1998):

Classification according to the network structure:

- autoassocciative – input nodes are simultaneously output nodes,
- heteroassociative – there is a set of output nodes different than the set of input nodes.

Classification according to feedback presence:

- ANN where there are no connections between output and input nodes,
- ANN with feedback (*recurrent neural networks*) where the current output signal is determined by the current input signal and former output signal.

Classification according to the network training:

- Supervised training is training performed until the ANN is able to produce the desired output vector y for each input vector x. Training will be performed based on the set of input-output data pairs (x, y).
- Unsupervised training is the kind of training where a sample of input vectors is involved in the learning process. During the training, statistical properties of the samples are estimated and similar input vectors are grouped into classes. The idea is to obtain consistent output vectors for similar inputs.
- Reinforcement learning is provided as a combination of previous training algorithms in the following way: ANN will give output vector y for each input vector x. If the obtained output vector is valued as "good" then the ANN is "rewarded" by increasing the existing weights of the branches. Otherwise, the ANN is "punished" decreasing weights on the branches.

The literature shows a wide range of possible applications of ANN. It is seen that there has been a significant growth in the number of publications in recent years. The most successful applications of ANN are in categorization and pattern recognition. In the transportation area ANN was implemented on a large number of different problems such as dispatching, incident detection, drivers behavior modeling, origin-destination matrices estimation, etc.

Chapter 3. Bee System Approach to the Modeling of Combinatorial Optimization Problems

3.1 Behavior of real bees

The honey bee colony chooses sections of a field that are most profitable among different nectar sources available. Previous studies have shown that the colony quickly and precisely adjusts its searching pattern in time and space following the environment's changing of nectar sources. Self-organization of the bees is based on a few relatively simple rules of individual insect's behavior (Gould (1987), Hill et al. (1997), Banschbach and Waddington (1994), Dukas and Real (1991), Kadmoon and Shmida (1992), Peleg et al. (1992), Seeley (1992), Dukas and Visscher (1994), Keasar et al. (1996), Chittka et al. (1997), Chittka and Thompson (1997), Collevatti et al. (1997), Waddington et al. (1998), Williams and Thompson (1998)). It is described here how the activities of a large number of individual bees can emerge into an organized pattern of collective foraging.

It is natural to consider a colony as a system of interacting individuals – foraging bees (Camazine and Sneyd, 1991). In that light, it is possible to first examine the relevant behavior of the individuals and then the information shared among the individuals in order to achieve common knowledge. "Collective – Swarm intelligence" is the emergent property of the colony of individuals requiring only limited and local knowledge that every contributor should possess. The exchange of information among individuals is the most important occurrence in the formation of collective knowledge. While examining the entire hive it is possible to distinguish some parts that commonly exist in all hives. The most important part of the hive with respect to exchanging information is the dancing area. Communication among bees related to the quality of food sources occurs in the dancing area. The related dance is called the *waggle* dance.

Generally, in a social insect colony individuals usually do not perform all tasks. They specialize in a set of tasks according to their morphology, age or chance (Bonabeau et al., 1999). However, a significant part of the entire bee colony will be foragers.

The basic characteristics of behavioral cycle of honey bee foraging for nectar could be seen in figure 3.1.

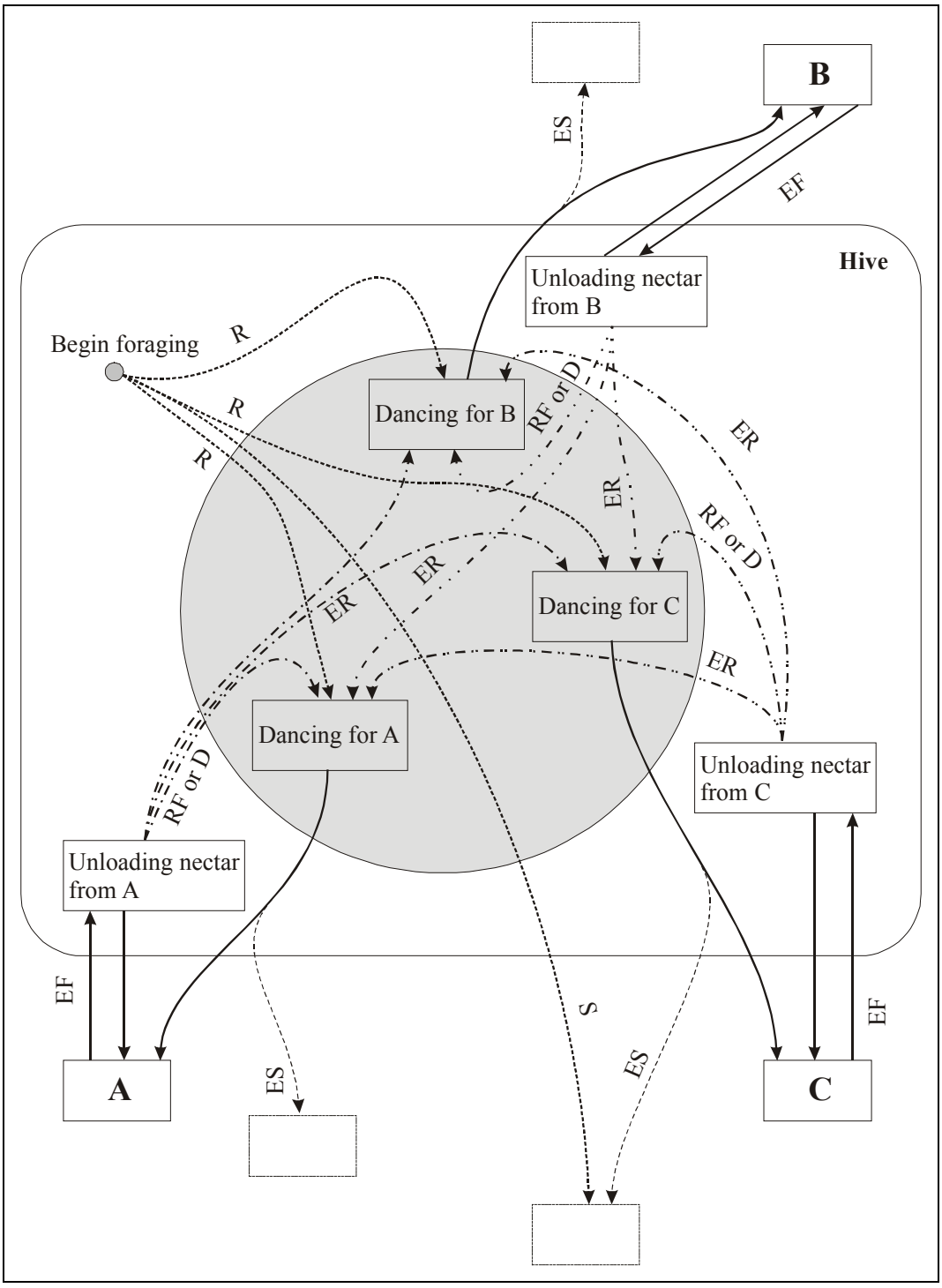

Figure 3.1 – Typical behavioral cycle of honey bee foraging for nectar in case when three discovered food sources exist

At the very beginning, a potential forager will start as an unemployed forager. That bee will have no knowledge about the food sources in the field. There are two possible options for such a bee:

- Due to some internal motivation or possibly some external clue, the bee will start searching spontaneously and in that way becomes a scout.
- As a response to the attendance of the waggle dance done by some other bee, the bee will start searching for a food source and in that way becomes a recruit.

After finding the food source, the bee utilizes its own capability to memorize the location and then immediately starts exploiting it. In this way the bee will become an "employed forager".

The foraging bee takes a load of nectar from the field and returns to the hive, relinquishing the nectar to a food store. After it relinquishes the food, the bee has the following options:

(a) abandon the food source and become an uncommitted follower,
(b) continue to forage at the food source without recruiting the nestmates, or
(c) dance and thus recruit the nestmates before returning to the same food source.

The bee opts for one of the above alternatives with a certain probability. The probabilities highly depend on the quality of the food source that has been visited.

Within the dancing area, the bee dancers "advertise" different food areas. The mechanisms by which the bee decides to follow a specific dancer are not well understood, but it is considered that "the recruitment among bees is always a function of the quality of the food source" (Camazine and Sneyd, 1991).

Through the passage of time, food sources could become exhausted and it could happen that some employed foraging bees could become unemployed – stop exploiting the source.

Bees have some memory and as long as a bee keeps information about this food source in memory, the bee is called an experienced unemployed bee. Sometimes experienced unemployed bees will make examinational flights to the food source and thus the bee is called an inspector. A reactivated forager is an experienced unemployed bee that has attended a dance containing information about a food source that is already known and

based on that has made a decision to go and explore the known and at the same time the advertised food source. In cases where an experienced unemployed bee attends a dance advertising an unknown food source, is possible that the bee wanted to begin exploring this unknown source, and become a recruit. In cases of experienced unemployed bees not finding a waggle dance to follow on the dance floor, the bee might start to search for some unknown (new) food source spontaneously and become a scout again.

It can be seen that there are several categories of foragers (de Vries and Biesmeijer, 1998):

- Employed forager – knows and uses a profitable food source. Just flying from food source to unloading zone in hive and vice versa (EF on figure 3.1).
- Unemployed forager could be the following (begin foraging – figure 3.1):
    - *Scout*, starts searching spontaneously without any knowledge of the food sources (S on figure 3.1).
    - *Recruit*, starts searching upon attending dance area in the hive – knows approximately the position of food source without any knowledge about quality (R on figure 3.1).
- Experienced forager that has some knowledge about the position and profitability of a food source could have the following tasks:
    - *Inspector* inspects the profitability status of a food source that has been already discovered.
    - *Reactivated forager* explore the same food source but upon attending dancing area – after receiving confirmation about source from other nestmates (RF on figure 3.1).
    - *Scout* starts to search for a new source after the previous source has been deteriorated (ES on figure 3.1).
    - *Recruit* unsatisfied with the currently visited food source, will start searching for a new source that has been advertised in dancing area (ER on figure 3.1).

It is important to note that not all bees start foraging simultaneously. The experiments confirmed that new bees begin foraging at a rate proportional to the difference between the eventual total number of bees and the number presently foraging.

At any moment, each foraging bee could be in one of the following places:

- not active,
- unloading nectar from food source,
- dancing for food source,
- feeding at food source,
- following a dancer, and
- scouting.

3.2 Artificial Bees

The successful applications of the Ant System to the complex engineering and management problems are certainly encouraging. At the same time, these successes act as a great inspiration to attempt to explore bees' behavior as a source of ideas and models for development of various artificial systems.

A large portion of social insects' activities is tied to food foraging. It is known that honeybees "normally spend the last part of their life collecting food" (Biesmeijer et al, 1998). Also, they "spend a considerable portion of their life span learning and improving their foraging skills" (Dukas and Visscher, 1994). Every bee colony has scouts that are the colony's explorers (Seeley and Visscher, 1988). The explorers do not have any guidance while looking for food. They are primarily concerned with finding any kind of food source. As a result of such behavior, the scouts are characterized by low search costs and a low average in food source quality. Occasionally, the scouts can accidentally discover rich, entirely unknown food sources. In the case of artificial bees, the artificial scouts attempting to solve difficult combinatorial optimization problems could have the fast discovery of the group of feasible solutions as a task. Some of those feasible solutions to the difficult combinatorial optimization problems could then prove to be solutions of very good quality.

In the case of honey bees, the recruitment rate represents a "measure" of how quickly the bee colony finds and exploits a newly discovered food source. Artificial recruiting could similarly represent the "measurement" of the speed with which the feasible solutions or

the "good quality" solutions are found.

The cooperation between the insects decreases foragers' costs in finding new food sources. This suggests that cooperation between artificial bees would also allow the fast discovery of the feasible solution.

It is also known that cooperation increases the quality of the food sources located by foragers. This implies that cooperation could also help us find the best solutions of the difficult combinatorial optimization problems.

The survival and progress of the bee colony is dependent upon the rapid discovery and efficient utilization of the best food resources. In other words, the successful solution of difficult engineering problems (especially those that need to be solved in real time) is connected to the relatively fast discovery of "good solutions".

3.3 Solving The Traveling Salesman Problem with The Bee System

The primary goal of the research is to explore the possible applications of swarm intelligence (and especially, in this part, collective bee intelligence) in solving complex traffic and transportation engineering problems. The development of the new heuristic algorithm for the Traveling Salesman Problem using the Bee System will serve as an illustrative example for such applications and will show the characteristics of the proposed concept.

The Traveling Salesman Problem (TSP) is chosen for the following characteristics:

- Very difficult (NP-hard) problem.
- There are plenty of benchmark problems (TSP has been studied a lot and through that process with purpose of technique comparison a lot of instances appears).
- Easy to understand.
- If one assumes that the closest food source is more attractive for honey bees, foraging bees behavior could be applied in order to find the best solution.

TSP could be formulated in the following way:

Let G = (N, A) be a graph where N is a set of nodes and A is a set of arcs or edges. Let us also introduce $C = (c_{ij})$ as the distance or cost matrix associated with a set of arcs. In TSP, the minimum distance circuit that passes through each node once and only once should be determined. In literature, this circuit is known as the Hamiltonian circuit (or cycle). There are a lot of different modalities to the problem. Most important are symmetric TSP where $c_{ij} = c_{ji}$ for all $(i, j) \in$ A and asymmetric TSP that occur otherwise.

There are several ways to formulate TSP mathematically. One of the earliest formulations, given by Dantzig, Fulkerson and Johnson (1954) will be presented here. Let x_{ij} be the following binary variable:

$$x_{ij} = \begin{cases} 1, & \text{if } (i, j) \text{ is used in optimal solution} \\ 0, & \text{otherwise} \end{cases}$$

Integer linear programming formulation could be as follows:

$$\min \quad \sum_{i} \sum_{j \neq i} c_{ij} x_{ij} \tag{3.1}$$

subject to

$$\sum_{j} x_{ij} = 1, \qquad i = 1, 2, ..., |N| \tag{3.2}$$

$$\sum_{i} x_{ij} = 1, \qquad j = 1, 2, ..., |N| \tag{3.3}$$

$$\sum_{(i, j) \in S} x_{ij} \leq |S| - 1 \qquad S \subset N, \; 2 \leq |S| \leq |N| - 2 \tag{3.4}$$

$$i, j = 1, 2, ..., |N| \tag{3.5}$$

The objective is to find the least costly tour. Constraints 3.2 and 3.3 should restrict the number of times that one node is visited on value 1. Constraint 3.4 represent subtour elimination constraints – they prohibit formulation of subtours (tours with less than $|\mathrm{N}|$ nodes).

Here, the focus will not be on TSP and its solutions developed in the past; more detailed survey can be found in Laporte (1992).

Let G = (N, A) be the graph in which the bees are collecting nectar (the graph in which the traveling salesman route should be discovered). Let us also randomly locate the hive in one of the nodes. When foraging, the bees are trying to collect as much nectar as possible. Let us also assume that the nectar quantity that is possible to collect flying along a certain link is inversely proportional to the link length. In other words, the shorter the link, the higher the nectar quantity collected along that link. This means that the greatest possible nectar quantity could be collected when flying along the shortest traveling salesman route. Our artificial bees will collect the nectar during a certain prescribed time interval. After that, we will randomly change the hive position. The bees will start to collect the nectar from the new location. We will then again randomly change the hive location, etc. The iteration in the searching process represents one change of the hive's position. Our artificial bees live in an environment characterized by discrete time. Each iteration is composed of a certain number of stages. A stage is an elementary time unit in the bees' environment. During one stage the bee will visit s nodes, create a partial traveling salesman tour, and after that return to the hive (the number of nodes s to be visited within one stage is prescribed by the analyst at the beginning of the search process). In the hive the bee will participate in a decision making process. The bee will decide whether to abandon the food source and become again uncommitted follower, to continue to forage at the food source without recruiting the nestmates, or to dance and thus recruit the nestmates before returning to the food source. Let us denote by B the total number of bees in the hive and by $B(u, z)$ the total number of bees collecting nectar during stage u $(u = 0, 1, 2, \ldots, \left\lceil \frac{|N|-1}{s} \right\rceil)$ in iteration z.

We will assume that at the beginning of every iteration z all bees are in the hive, i.e.:

$$B(0, z) = 0 \tag{3.6}$$

It is noted that not all bees start foraging simultaneously in nature. In the case of artificial bees, we increase the number of foraging bees in every subsequent stage in the following

way:

Let us introduce the binary variables $b_k(u, z)$, defined as:

$$b_k(u,z) = \begin{cases} 1, & \text{if } k\text{ - th bee participates in foraging during stage } u \text{ in iteration } z \\ 0, & \text{otherwise} \end{cases} \tag{3.7}$$

$$k = 1,2,\ldots, B$$

$$u = 1, 2, \ldots, \left\lceil \frac{|N|-1}{s} \right\rceil$$

$$z = 1,2,\ldots, M$$

where:

M – maximum number of iterations.

Some bees will start foraging in the first stage. The remaining bees will join the nestmates in the second stage, or in the third stage, or in the fourth stage, etc. Once a bee starts foraging, it will remain "active" until the end of the considered iteration. The foraging activity of every bee can be described by the array composed of 0's and 1's. The array [1, 1, 1,…, 1] describes foraging activity of the bee that has been foraging from the first stage; the array [0, 0, 0, 1, 1, 1,…, 1] describes foraging activity of the bee that has been foraging from the fourth stage, and so forth.

Let us also introduce the binary variables $h_k(u, z)$, defined in the following way:

$$h_k(u,z) = \begin{cases} 1, & \text{if } w > r_k(u,z) \text{ and } b_k(u-1,z) = 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.8}$$

where:

w – the parameter given by the analyst ($0 \le w \le 1$)

$r_k(u, z)$ – the random number taken from the unit interval [0, 1]

The binary variables $h_k(u, z)$ indicate the stage in which a particular bee starts foraging. For every bee k that has not been participating in the foraging process in the stage $(u – 1)$, we have chosen the random number $r_k(u, z)$. The k-th bee will join her nestmates in foraging during stage u, if the following relation has been satisfied:

$$w > r_k(u, z) \tag{3.9}$$

The higher the value of the parameter w given by the analyst, the higher the chance for any bee to become foraging. That is, practically all bees from the hive will start the foraging process very quickly. A smaller value for the parameter w implies a slower increase of the number of foraging bees.

The binary variable $b_k(u, z)$ equals:

$$b_k(u, z) = b_k(u\text{-}1, z)+h_k(u, z) \tag{3.10}$$

The total number of foraging bees during u-th stage in the z-th iteration equals:

$$B(u, z) = \sum_{k=1}^{B} b_k(u, z) \tag{3.11}$$

During any stage, bees are choosing nodes to be visited in a random manner. The Logit model is one of the most successful and widely accepted discrete choice models. Inspired by the Logit model, we have assumed that the probability of choosing node j by the k-th bee, located in the node i (during stage u +1 and iteration z) equals:

$$p_{ij}^{k}(u+1, z) = \begin{cases} \dfrac{e^{-ad_{ij}\frac{z}{\sum_{r=\max(z-b,1)}^{z-1} n_{ij}(r)}}}{\sum_{l\in N_k(u,z)} e^{-ad_{il}\frac{z}{\sum_{r=\max(z-b,1)}^{z-1} n_{il}(r)}}}, & i = g_k(u,z),\ j \in N_k(u,z), \forall k,u,z \\ 0, & \textit{otherwise} \end{cases} \tag{3.12}$$

where:

- i, j – node indexes (i, j = 1, 2, …, $|N|$),
- d_{ij} – length of link (i, j),
- k – bee index (k = 1,2,…, B),
- z – iteration index (z = 1, 2,…, M),
- $g_k(u, z)$ – last node that bee k visits at the end of stage u in iteration z,
- $N_k(u, z)$ – set of unvisited nodes for bee k at stage u in iteration z (in one stage bee will visits s nodes; we have $|N_k(u, z)| = |N| - us$),

b – "memory length"*,
$n_{il}(r)$ – total number of bees that visited link (i, l) in r-th iteration,
a – input parameter.

Let us discuss relation (3.12) in more detail. The greater the distance between node i and node j, the lower the probability that the k-th bee located in the node i will choose node j during stage u and iteration z. The distance d_{ij} is obviously a very important factor influencing the bee's choice of the next node. The influence of the distance is lower at the beginning of the search process. The greater the number of iterations z, the higher the influence of the distance. This is expressed by the term z in the nominator of the exponent (relation 3.12). In other words, at the beginning of the search process bees have "more freedom of flight". They have more freedom to search the solution space. The more iterations we make, the less freedom the bees have to explore the solution space. The closer we come to the end of the search process, the more focused the bees are on the flowers (nodes) in the neighborhood. Like their natural counterparts, artificial bees have memory; they can receive and remember information about how many bees visited a certain link during the last b iterations. The greater the total number of bees that visited a certain link in the past, the higher the probability is of choosing that link in the future. This represents the interaction between individual bees in the colony.

For every bee, we now know the nectar quantity collected by the bee (the length of the partial traveling salesman tour). After returning to the hive, bees relinquish the nectar to a food storer bee. After relinquishing the food, the bee then makes the decision about abandoning the food source or continuing the foraging at the food source. The basic assumption is that every bee can obtain the information about nectar quantity collected by every other bee. The probability that the bee k, at the beginning of the stage $u + 1$, will use the same partial tour that is defined in stage u in iteration z equals:

$$p_k(u+1, z) = e^{-\frac{L_k(u,z) - \min\limits_{r \in w(u,z)} (L_r(u,z))}{u\, z}} \tag{3.13}$$

* While foraging in stage u, every artificial bee has the ability to notice the total number of bees in every link. Every artificial bee has the same ability for previous stages as well. That is, artificial bees have the capacity to remember former bee assignments in the network. However, bee recollection is limited. The maximum number of stages that a bee can recall represents memory length.

where $L_k(u, z)$ is the length of partial route that is discovered by bee k in stage u in iteration z.

We can see from relation (3.13) that the bee will fly along the same partial tour with the probability equal to one when the bee has discovered the shortest partial traveling salesman tour in stage u in iteration z. The longer the tour that the bee has discovered, the smaller is the probability that the bee will fly again along the same tour. When bee decides not to abandon the food source it can:

(a) continue to forage at the same food source without recruiting the nestmates;
(b) fly to the dance floor area and start dancing, thus recruiting the nestmates before the return to the food source.

The bee opts for each one of the above alternatives with a certain probability. Within the dance area the bee dancers "advertise" different food areas. It has been mentioned before that the mechanisms by which the bee in nature decides to follow a specific dancer are not well understood, but it is considered that the recruitment among bees is a function of the quality of the advertised food source. Since the bees are, before all, social insects (the interaction between individual bees in the colony has been well documented), it is assumed in this paper that the probability p^* of an event in which the bee will continue foraging at the food source without recruiting the nestmates is very low:

$$p^* << 1 \tag{3.14}$$

After relinquishing the food, and after making the decision to continue foraging at the food source, the bee flies to the dance floor and starts dancing with the probability equal to $(1- p^*)$. Bee dancing represents the interaction between individual bees in the colony. This kind of communication between individual bees contributes to the formation of the "collective intelligence" of the bee colony.

At the beginning of stage $u + 1$, if a bee does not use the same partial traveling salesman tour, the bee will go to the dancing area and will follow another bee(s). Every partial traveling salesman tour ξ that is being advertised in the dance area has two main attributes:

(a) the total length, and

(b) the number of bees that are advertising the partial route.

We introduce the normalized value of the total length of the partial traveling salesman tour and the normalized value of the number of bees advertising the partial tour. Both the normalized values are defined in the following way: (a) They can take any value between 0 and 1; (b) The smaller the normalized value of the total length, the better is the partial tour; (c) The bigger the normalized value of the number of bees, the better is the partial tour.

Let us denote by $Y(u, z)$, the set of partial tours that were visited by at least one bee and by $B_\xi(u, z)$ – the number of bees that discovered partial route ξ. The normalized value of the partial route length equals:

$$\alpha_\xi(u,z)=\begin{cases}\dfrac{L_\xi(u,z)-\min\limits_{r\in Y(u,z)}(L_r(u,z))}{\max\limits_{r\in Y(u,z)}(L_r(u,z))-\min\limits_{r\in Y(u,z)}(L_r(u,z))}, & for\ \max\limits_{r\in Y(u,z)}(L_r(u,z))\neq\min\limits_{r\in Y(u,z)}(L_r(u,z)), \xi\in Y(u,z), \forall u,z\\ 0, & otherwise\end{cases} \tag{3.15}$$

The normalized value of the number of bees advertising the partial tour equals:

$$\beta_\xi(u,z)=\begin{cases}\dfrac{B_\xi(u,z)-\min\limits_{r\in Y(u,z)}(B_r(u,z))}{\max\limits_{r\in Y(u,z)}(B_r(u,z))-\min\limits_{r\in Y(u,z)}(B_r(u,z))}, & for\ \max\limits_{r\in Y(u,z)}(B_r(u,z))\neq\min\limits_{r\in Y(u,z)}(B_r(u,z)), \xi\in Y(u,z), \forall u,z\\ 0, & otherwise\end{cases} \tag{3.16}$$

Inspired by the Logit model, we have assumed that the probability that the partial route ξ will be chosen by any bee that decided to choose the new route equals:

$$p_\xi(u,z)=\frac{e^{\rho\beta_\xi(u,z)-\theta\alpha_\xi(u,z)}}{\sum\limits_{\tau\in Y(u,z)}e^{\rho\beta_\tau(u,z)-\theta\alpha_\tau(u,z)}} \qquad \xi\in Y(u,z) \quad ,\forall u,z \tag{3.17}$$

where:

ρ , θ - parameters given by the analyst.

Before relocating the hive, we tried to improve the solution obtained by the bees in the current iteration by applying the well-known 2-opt and 3-opt heuristic algorithms.

Creating the partial Traveling Salesman Tours, where in every stage every bee will visit just one node (s = 1) is shown in figure 3.2. It could be seen that the bee (– – –) has abandoned the food source it discovered in stage 1 and follows bee (——) through stage 2.

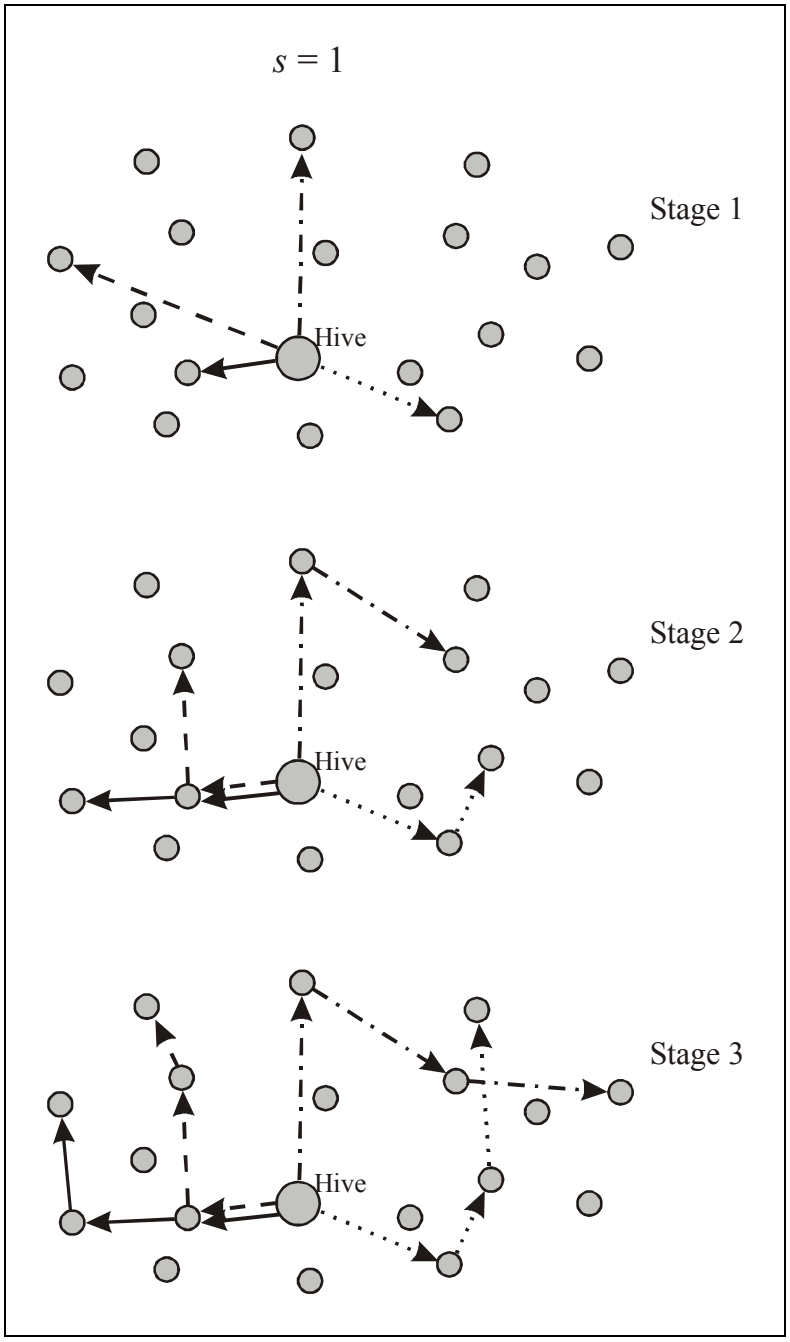

Figure 3.2 – Creating partial Traveling Salesmen Tours (for stage length s = 1)

3.3.1 Tour improving algorithms

The most frequently used tour improvement algorithms are developed based on the *k*-opt procedure. The basic idea is to replace the subset of *k* arcs in the previously defined tour with the subset of new arcs with same cardinality and smaller total length such that a new, shorter tour will result. The complexity of the *k*-opt algorithm is $O(n^k)$ where *n* is the number of nodes considered in the TSP instance. In the literature, researchers have used 2-opt and 3-opt algorithms most frequently.

To explain the 2-opt procedure, let us consider two arcs that are not adjacent (for example (t_i, t_{i+1}) and (t_j, t_{j+1}) from figure 3.3). If we were to remove them from the tour, the remainder of the tour will contain two paths with end nodes $t_{i+1} - t_j$ and $t_i - t_{j+1}$. There are two ways to reconnect those two paths into a tour again: by inserting already removed arcs, which would yield the old tour or by inserting the edges (t_i, t_j) and (t_{i+1}, t_{j+1}). The original tour will be changed in case the total length of the new arc pair is shorter than the total length of previously removed arc pair (i.e. $d_{t_i,t_j} + d_{t_{i+1},t_{j+1}} - d_{t_i,t_{i+1}} - d_{t_j,t_{j+1}}$ is negative).

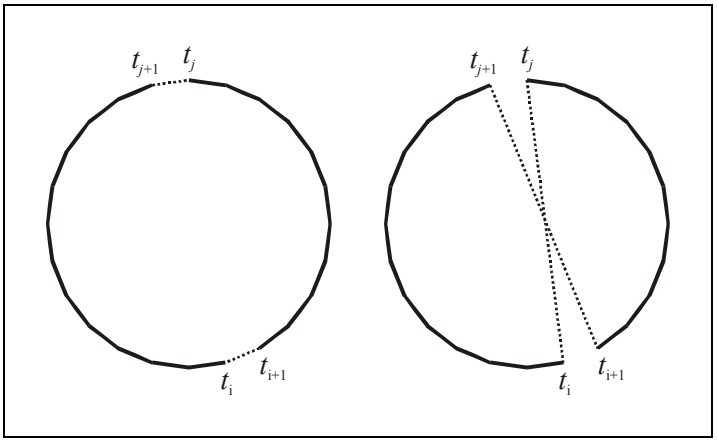

Figure 3.3 – A 2-opt change (original tour on the left and resulting tour on the right)

A pseudo code for a 2-opt algorithm could be formulated as follows (Smith, 1982):

Consider the tour $(t_1, t_2, \ldots, t_n, t_1)$ with total length *L*;
Set *i* := 1;
Set *j* := *i* + 2;
While *i* <= $\lfloor N \rfloor$ - 2 **do begin**
    Consider the tour $(t_1, t_2, \ldots, t_i, t_j, t_{j-1}, \ldots, t_{i+1}, t_{j+1}, t_{j+2}, \ldots, t_1)$ with total length L_1;
    if $L_1 < L$ **then begin**
        $(t_1, t_2, \ldots, t_n, t_1) := (t_1, t_2, \ldots, t_i, t_j, t_{j-1}, \ldots, t_{i+1}, t_{j+1}, t_{j+2}, \ldots, t_1)$;

```
        L := L1;
        i := 1;
        j := i + 2;
    end else begin
        Inc (j);
        if j > |N| then begin
            Inc (i);
            j := i + 2;
        end;
    end;
End While;
```

The 3-opt algorithm utilizes the same idea as the 2-opt. The only difference is the fact that now, initially three arcs are removed and then three new arcs are added. The algorithm used to make the tour improvement is Lin's algorithm (Lin, 1965).

The pseudo code for 3-opt algorithm could be formulated as follows:

```
for i := 1 to n do begin
    for k := 1 to n-3 do begin
        for j:= k+1 to n-1 do begin
            if d_{t_k,t_{j+1}} + d_{t_1,t_j} ≤ d_{t_1,t_{j+1}} + d_{t_k,t_j} then begin
                d := d_{t_k,t_{j+1}} + d_{t_1,t_j} ;
                α := true;
            end else begin
                d := d_{t_1,t_{j+1}} + d_{t_k,t_j} ;
                α := false;
            end;
            if d + d_{t_{k+1},t_n} < d_{t_1,t_n} + d_{t_k,t_{k+1}} + d_{t_j,t_{j+1}} then begin
                if α then
                    (t1, t2, …., tn) = (tj+2, …., tn, tk+1, …., tj, t1, …., tk, tj+1)
                else
                    (t1, t2, …., tn) = (tj+2, …., tn, tk+1, …., tj, tk, …., t1, tj+1);
            end;
        end;
    end;
    (t1, t2, …., tn) = (tn, t1, t2, …., tn-1);
end;
```

The procedure for the 3-opt algorithm described by the pseudo code is time consuming because the algorithm will try all possible combinations of three arcs replacement. There is the possibility to run this algorithm much faster with a little chance of loss in solution quality. One simple way to do that is to reduce the size of searching space. Nodes t_1, t_k, and t_j should be "close enough" to each other to be considered as starting nodes of

arcs to be removed. Algorithm will start with node t_1 at the beginning of each improving trial. The number of closest nodes to node t_1 that will be considered for possible improvement is given in advance.

3.3.2 Experimental study of the bee system

The proposed Bee System was tested on a large number of numerical examples. In all the cases one of the tour improving algorithms was employed. There are three sets of results that correspond to 2opt, 3-opt and 3-opt "short version" tour improving algorithm respectively. The benchmark problems were taken from the following Internet address: **http://www.iwr.uni-heidelberg.de/iwr/comopt/software/TSPLIB95/tsp/**. The following 10 problem instances were considered: Eil51.tsp, Berlin52.tsp, St70.tsp, Pr76.tsp, Kroa100.tsp, Eil101.tsp, Tsp225.tsp, A280.tsp, Pcb442.tsp and Pr1002.tsp. Problem instances Pcb442.tsp and Pr1002.tsp were considered just with 3opt "short version" tour improving algorithm. All tests were run on an IBM compatible PC with PIII processor (533MHz). Tables 3.1, 3.2 and 3.3 present the results obtained by Bee System when search is limited to 100 cycles.

Table 3.1 – The results obtained by the Bee System enriched with 2-opt heuristic

Problem	Number of nodes	Optimal Value (O)	The best value obtained by the Bee System (B)	$\frac{(B-O)}{O}$ (%)	Time required to find the best solution (seconds)	Average value obtained by the Bee System over 20 runs (A)	St. Dev. (SD)	$\frac{(A-O)}{O}$ (%)
Eil51	51	428.87	431.121	0.53%	44	433.758	1.37	1.14%
Berlin52	52	7544.366	7544.366	0%	18	7634.37	78.2	1.19%
St70	70	677.11	678.621	0.22%	238	684.275	3.53	1.06%
Pr76	76	108159	108790	0.58%	127	109444.6	461	1.19%
Kroa100	100	21285.4	21441.5	0.73%	58	21575.7	138.83	1.36%
Eil101	101	640.21	642.45	0.35%	146	665.62	7.94	3.97%
Tsp225	225	3859	4065.56	5.35%	2076	4113.71	27.3	6.6%
A280	280	2586.77	2740.63	5.95%	1855	2784.81	19.56	7.66%

From table 3.1 it could be seen that the Bee System, reinforced with 2-opt tour improvement heuristic will provide excellent results for a small size of TSP instances. However, if the size of instances is increased, the quality of the solutions is reduced (last column table 3.1).

Table 3.2 – The results obtained by the Bee System enriched with 3-opt heuristic

Problem	Number of nodes	Optimal Value (O)	The best value obtained by the Bee System (B)	$\frac{(B-O)}{O}$ (%)	Time required to find the best solution (seconds)	Average value obtained by the Bee System over 20 runs (A)	St. Dev. (SD)	$\frac{(A-O)}{O}$ (%)
Eil51	51	428.87	428.87	0	37	428.87	0	0
Berlin52	52	7544.366	7544.366	0	1	7544.366	0	0
St70	70	677.11	677.11	0	22	677.11	0	0
Pr76	76	108159	108159	0	11	108159	0	0
Kroa100	100	21285.4	21285.4	0	10	21285.4	0	0
Eil101	101	640.21	640.21	0	1741	643.05	1.7	0.44%
Tsp225	225	3859	3876.05	0.44%	5153	3905.32	18.9	1.2%
A280	280	2586.77	2600.34	0.53%	13465	2627.45	12.31	1.57%

Table 3.3 – The results obtained by the Bee System enriched with 3-opt "short version" heuristic

Problem	Number of nodes	Optimal Value (O)	The best value obtained by the Bee System (B)	$\frac{(B-O)}{O}$ (%)	Time required to find the best solution (seconds)	Average value obtained by the Bee System over 20 runs (A)	St. Dev. (SD)	$\frac{(A-O)}{O}$ (%)
Eil51	51	428.87	428.87	0	29	428.87	0	0
Berlin52	52	7544.366	7544.366	0	0	7544.366	0	0
St70	70	677.11	677.11	0	7	677.11	0	0
Pr76	76	108159	108159	0	2	108159	0	0
Kroa100	100	21285.4	21285.4	0	10	21285.4	0	0
Eil101	101	640.21	640.21	0	61	643.07	1.84	0.45%
Tsp225	225	3859	3899.9	1.06%	11651	3909.69	9.19	1.31%
A280	280	2586.77	2608.33	0.83%	6270	2632.42	14.89	1.76%
Pcb442	442	50783.55	51366.04	1.15%	4384	51756.89	195.3	1.92%
Pr1002	1002	259066.6	267340.7	3.19%	28101	268965.6	1182.22	3.82%

We can see from tables 3.1, 3.2 and 3.3 that the proposed Bee System produced results of a very high quality. The Bee System was able to obtain the objective function values that are very close to the optimal values of the objective function. In all instances with less than 100 nodes, the Bee System produced the optimal solution (table 3.2 and table 3.3). The times required to find the best solutions by the Bee System are low. In other words, the Bee System was able to produce "very good" solutions in a "reasonable amount" of computer time.

The best solutions discovered by Bee System are presented in the following figures.

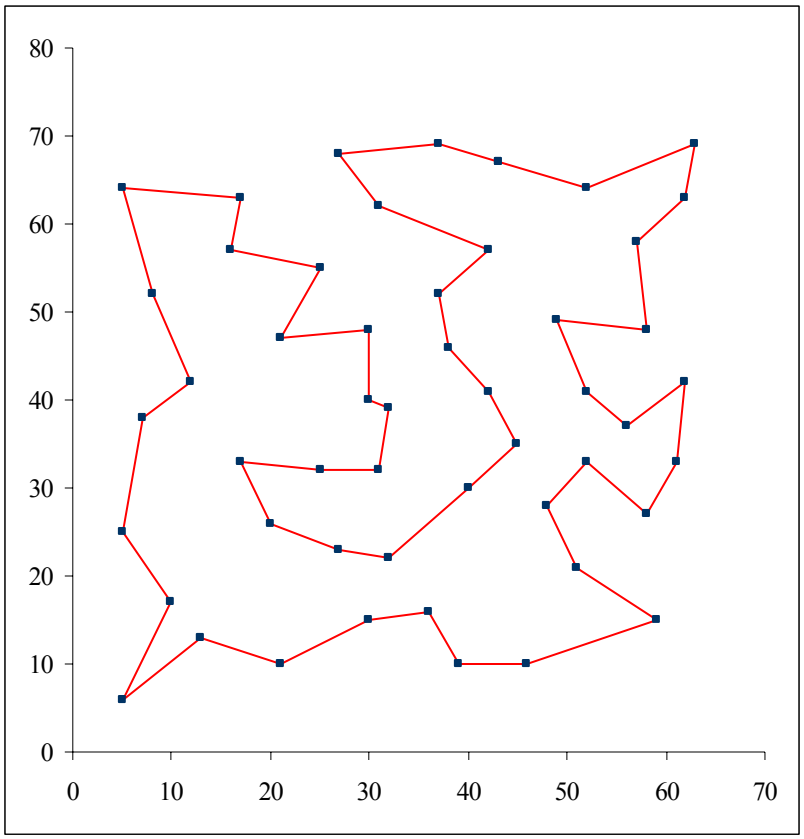

Figure 3.4 – Bee System solution to the TSP problem instance Eil51.tsp (limited to 100 cycles)

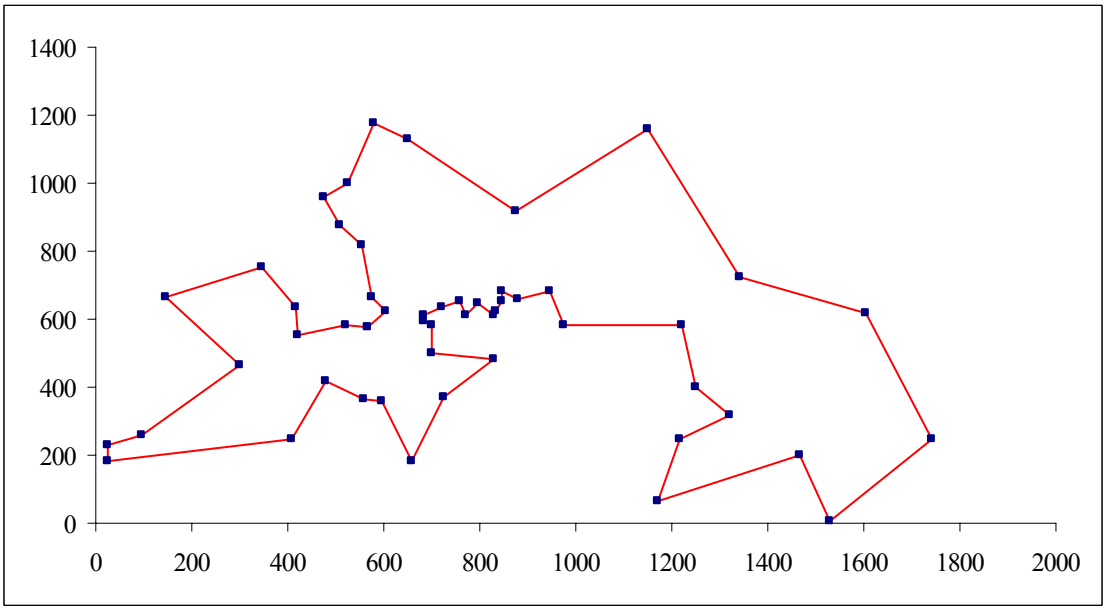

Figure 3.5 – Bee System solution to the TSP problem instance Berlin52.tsp (limited to 100 cycles)

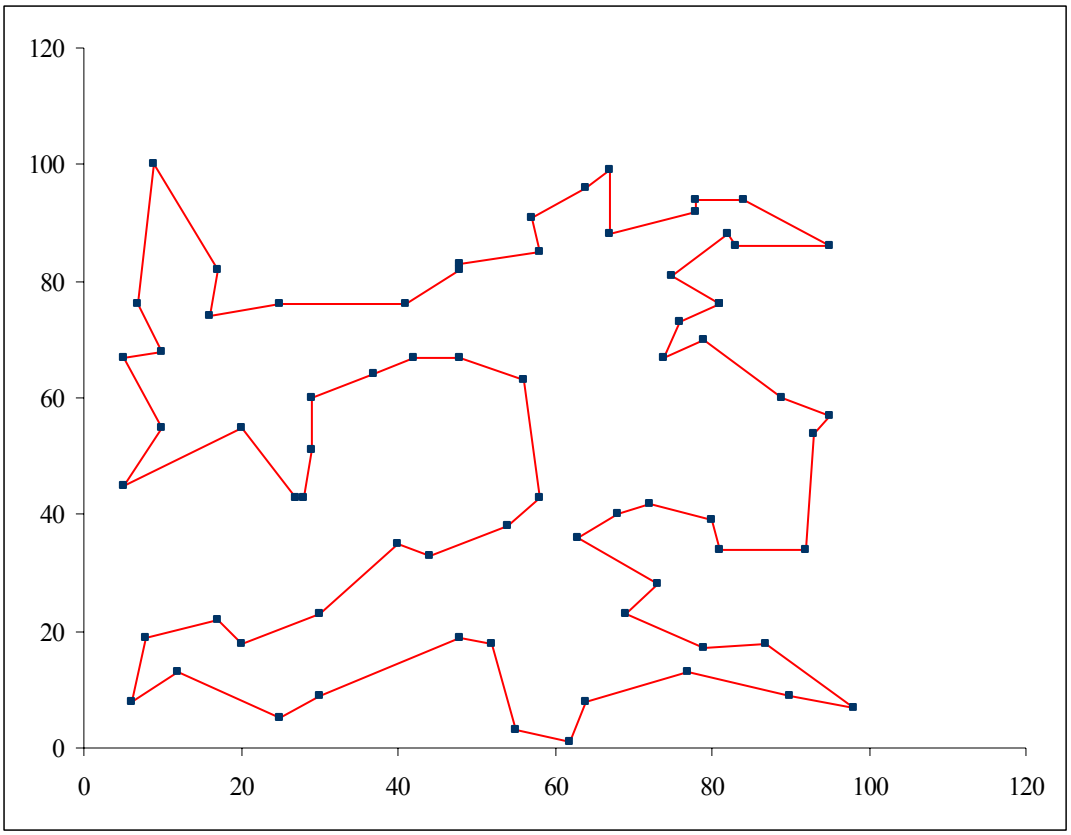

Figure 3.6 – Bee System solution to the TSP problem instance St70.tsp (limited to 100 cycles)

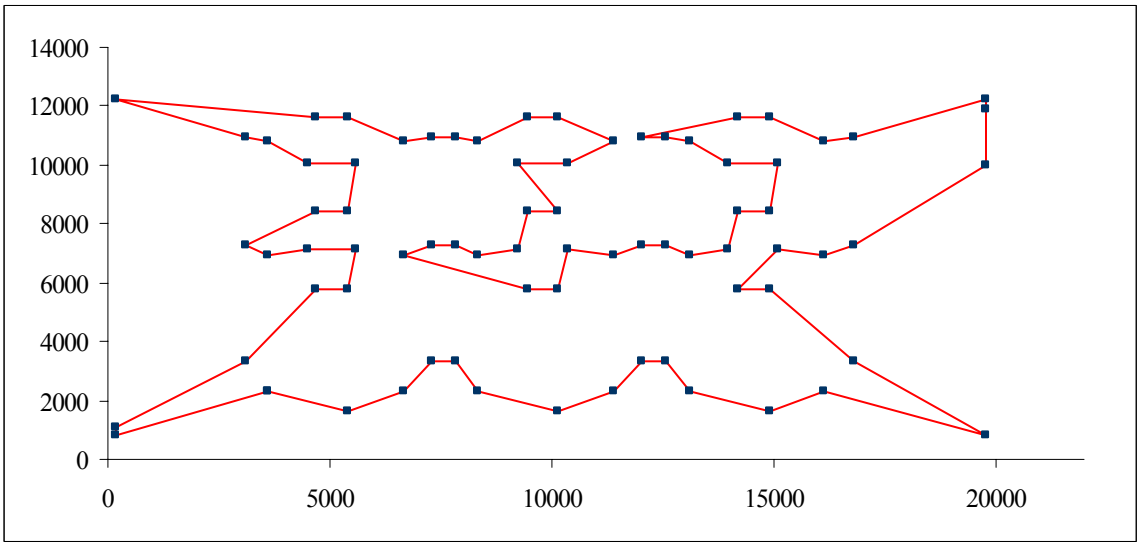

Figure 3.7 – Bee System solution to the TSP problem instance Pr76.tsp (limited to 100 cycles)

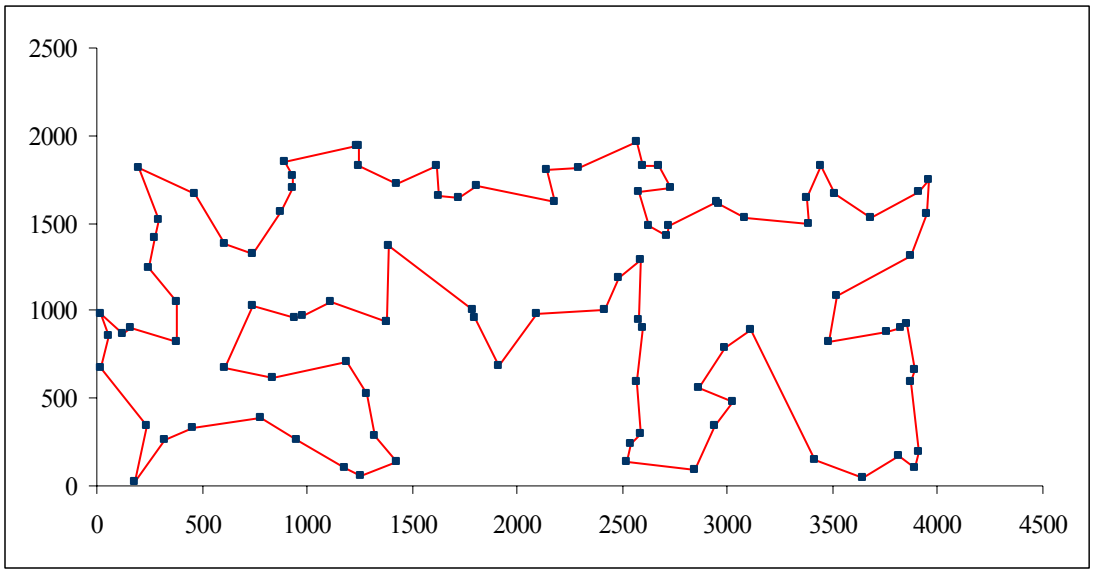

Figure 3.8 – Bee System solution to the TSP problem instance Kroa100.tsp (limited to 100 cycles)

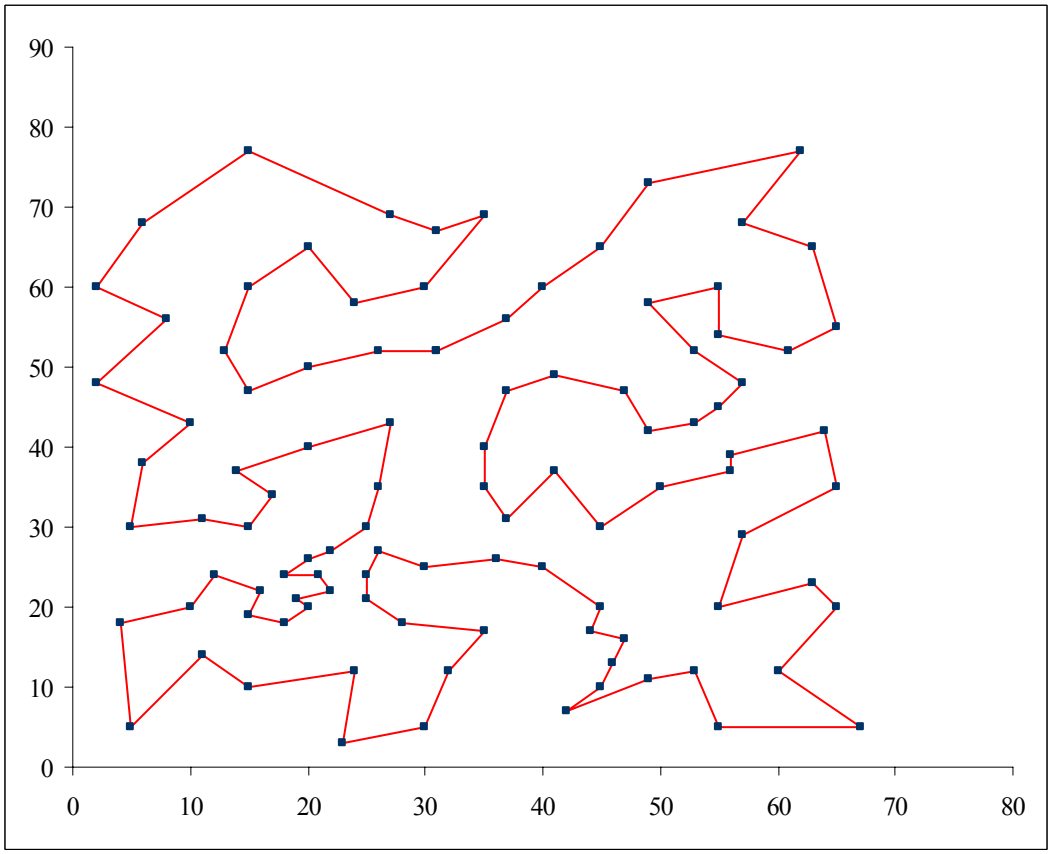

Figure 3.9 – Bee System solution to the TSP problem instance Eil101.tsp (limited to 100 cycles)

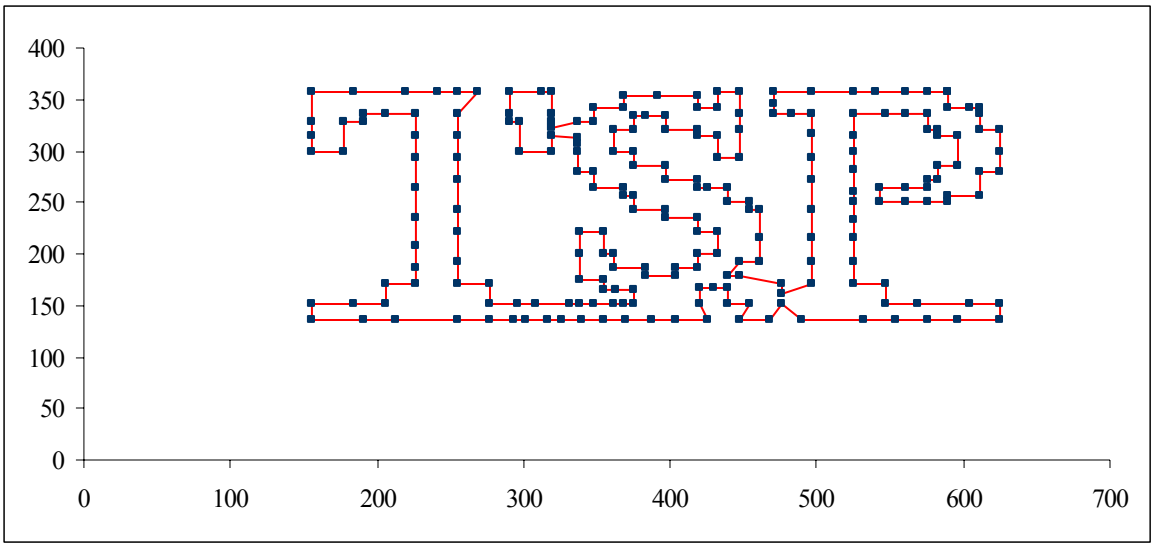

Figure 3.10 – Bee System solution to the TSP problem instance Tsp225.tsp (limited to 100 cycles)

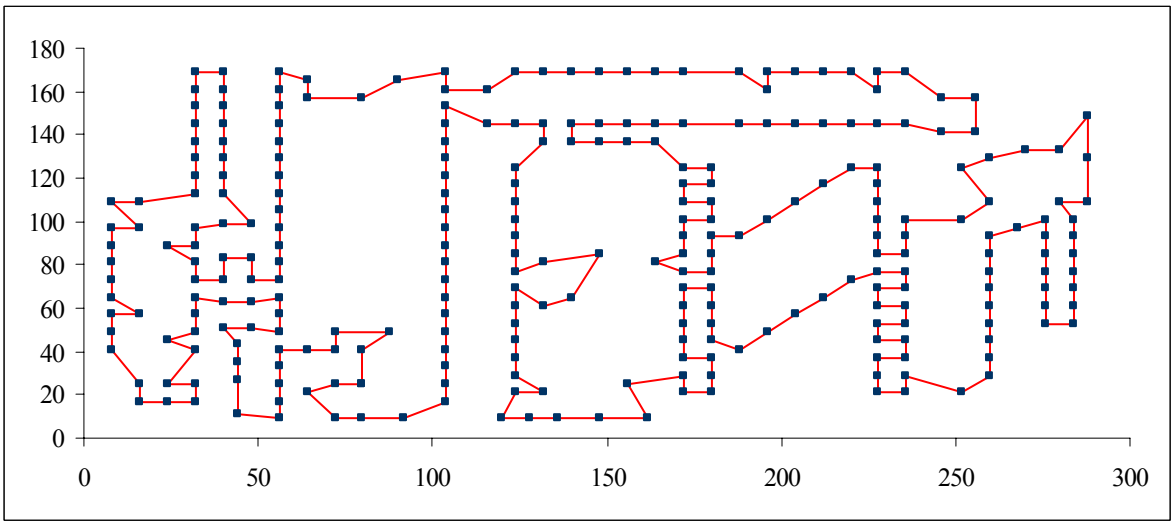

Figure 3.11 – Bee System solution to the TSP problem instance A280.tsp (limited to 100 cycles)

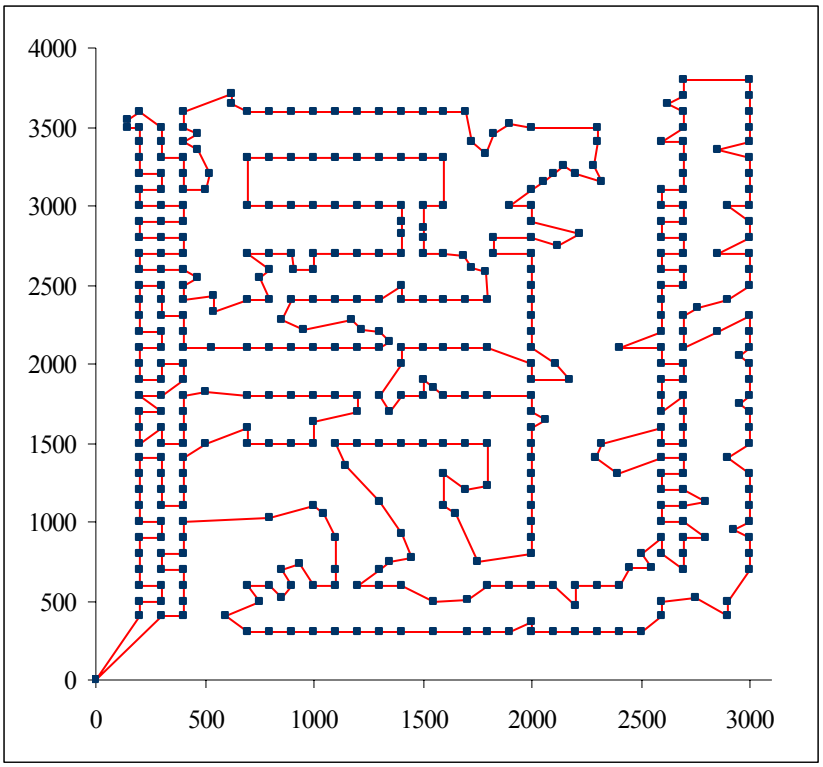

Figure 3.12 – Bee System solution to the TSP problem instance Pcb442.tsp (limited to 100 cycles)

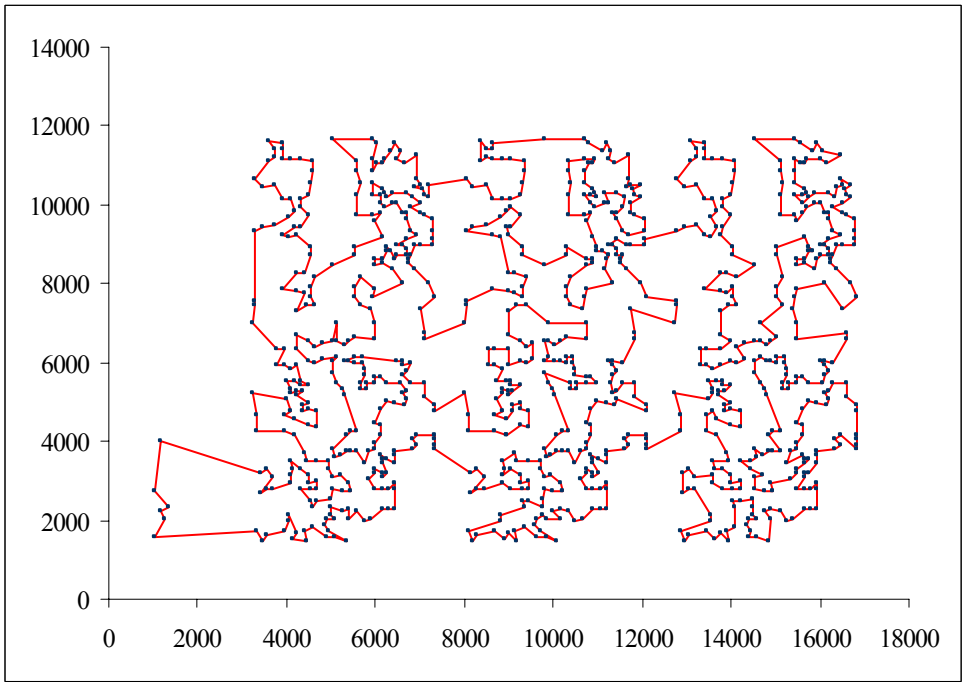

Figure 3.13 – Bee System solution to the TSP problem instance Pr1002.tsp (limited to 100 cycles)

3.4 Bee System and Fuzzy Logic approach to the Stochastic Vehicle Routing Problem

3.4.1 Introduction

The classical vehicle routing problem, which appears in various transportation activities, consists of finding a set of routes that would minimize transport costs. Vehicles leave the depot, serve nodes that request service, and on completion of their routes, return to the depot. Every node is characterized by a certain *demand* (the amount to be delivered or the amount to be picked up from the node). Other known values include the *coordinates* of the depot and nodes, the *distances* between all pairs of nodes, and the *capacity* of the vehicles providing service.

During the last two decades, papers that deal with uncertain demand at nodes have appeared (Dror and Trudeau (1986), Dror et al. (1989), Teodorović and Pavković (1992), Lambert et al. (1993), Dror (1993), Dror et al. (1993), Gendreau et al. (1996), Teodorović and Pavković (1996), Yang et al. (2000), Teodorović and Lučić (2000)). In all of them, stochastic demand at nodes was represented by random variables.

Vehicle routing problems with uncertain demand at nodes appears in the delivery of home heating, trash collection, beer and soft drink distribution, provision of the bank automates with cash and collection of cash from bank branches. The basic characteristic of the vehicle routing problem with uncertain demand at nodes is that the real value of demand at a node becomes known only after the vehicle has reached the node. Because of the uncertainty of demand at the nodes, a vehicle might not be able to service a node once it arrives there due to an insufficient capacity. Such a situation is known as a "route failure". In the case of "route failure" different actions need to be applied.

A new algorithm for handling the vehicle routing problem when there is uncertain demand at nodes will be described. The goal is to produce an "intelligent" vehicle routing system capable of providing real time decisions necessary to build a set of "high quality routes" for the vehicles servicing the nodes.

The entire approach presentation is organized in the following way: Statement of the problem is given in section 3.4.2. A proposed solution to the problem is given in section

3.4.3. Numerical experiments are shown in section 3.4.4.

3.4.2 Statement of the problem

Let us assume that Stochastic Vehicle Routing Problem is assigned as follows:

- single depot and *n* nodes to be serviced (figure 3.14),
- homogenous feet (vehicle capacity is denoted by C) and
- demand at each node is a random variable (probability density function is known).

Vehicles set out from depot *D*, serve a number of nodes, and on completion of their service, return to the depot.

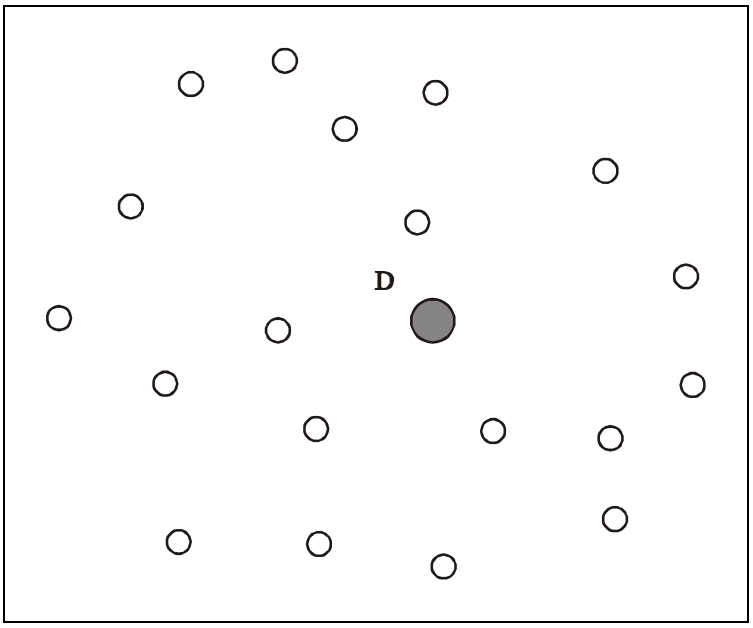

Figure 3.14 – Depot *D* and nodes requiring service

Increasing the number of nodes served along a route decreases the available (remaining) capacity of the vehicle. After completing service at one node, it is simple to calculate whether the vehicle is able to serve the next node or not, when demand at nodes is deterministic. On the other hand, when the demand at the nodes is characterized by uncertainty and treated as a random value, it is not a simple task to decide whether the

vehicle should serve the next node or return to the depot.

It is quite clear that the greater the vehicle's remaining capacity and the lesser the demand at the next node, the greater the vehicle's "chances" of being able to serve the next node. In other words, the greater the difference between the remaining capacity and demand at the next node, the greater is our preference to send the vehicle to serve that subsequent node and thus, the greater the number of nodes in the route. Also, due to the uncertainty of demand at nodes, a vehicle might not be able to service a node once it arrives there due to insufficient capacity. It will be assumed that in such situations, the vehicle returns to the depot, empties what it has picked up thus far, returns to the node where it had a "failure," and continues service along the predefined path (or along the rest of the planned route in cases where the set of routes is defined in advance) as it has been shown on figure 3.15.

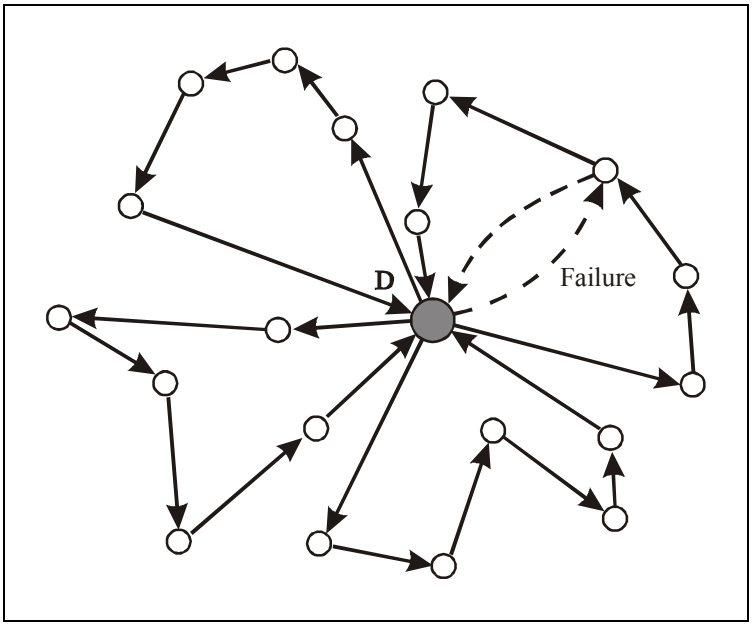

Figure 3.15 - "Failure" at a node of the planned route

Our wish to use the vehicle capacity to its fullest potential will produce planned routes with shorter total distances. However, this will also increase the number of cases in which vehicles arrive at a node and are unable to service it. In other words, the total distance will be increased due to the route "failure". Smaller utilization of vehicle capacity, on the other hand, will produce longer routes and less additional distance to cover due to the

failures.

The goal is to develop a decision support system that will make real time decisions on whether one particular vehicle located in the just serviced node will go to another node in the sequence or return to the depot.

3.4.3 Proposed solution to the problem

This part of the dissertation will describe an attempt to develop an "intelligent" vehicle routing system capable of making real time decisions of high quality. The system behaves "intelligently" if it "emits" similar output results for similar input variables. Artificial neural networks and fuzzy systems are "intelligent" systems since they have the ability to "learn from experience". Recognition without definition is a characteristic of intelligent behavior. The initial assumption is that it is possible to develop a vehicle routing system that will recognize different situations. As in other intelligent systems, the "intelligent" vehicle routing system should be able to generalize, adapt and learn based on new knowledge and new information.

Let us introduce the following assumption: *the future could be predicted without a mistake*. In this case, the assumption means that we are able to exactly predict demand values at all nodes in the transportation network. In the case of *perfect prediction* we should be able to make *optimal decisions*. In other words, once we know the random demand values at all nodes, we can try to develop the best vehicle routes.

For a known "scenario" (the random demand values at all nodes are known) the problem of producing the best set of vehicle routes can be solved using a particular optimization technique or heuristic algorithm.

The approach applied here contains the following two steps:

- Using the developed Bee System, solve the Vehicle Routing Problem as a Traveling Salesman Problem and create a "Giant route" (most frequently unfeasible solution to the original problem).
- By "walking" along the created giant route it is possible to decide when to finish one vehicle's route and when to start with the next vehicle's route. These decisions could be made easily since knowledge about demand at every node and

vehicle capacity is assumed.

The above-considered problem can be solved *many times for different scenarios*. If one can precisely predict the future then he/she can get the optimal or close to the optimal solution to the problem. Instead of predicting it though, one can simulate future events. In other words, one can simulate the demand at nodes represented by random variables. In the next step the optimal, or close to the optimal solution will be obtained for the set of demands simulated, by solving the problem. One can repeat this procedure a large number of times. In this way one can get the "best" solution for every simulated "scenario". This "statistical material" enables the generation of a fuzzy rule base. The Wang and Mendel (1992) procedure could be used to generate a fuzzy rule base from numerical data. Before introducing the procedure for generating a fuzzy rule base from numerical data, basic elements of fuzzy systems are to be presented.

3.4.4 Fuzzy Systems

It could happen that an experienced decision maker could achieve better results than a classical automatic control system when managing a complex system (transportation, industrial, etc.). The entire control strategy of an experienced decision maker could be formulated into a huge set of descriptive rules (number of rules highly depends on the complexity of the system that should be managed). The set of rules could be easily processed manually, but it could be complicated to reformulate those rules and apply one of the classical algorithms. Complications arise from the fact that extracting knowledge from human beings could bring qualitative expressions instead of quantitative. For human beings this, at times, is more preferred. The qualitative, or *fuzzy* nature of the human process of thinking and decision making has encouraged researchers to attempt to develop artificial systems that will behave in a similar way, while making decisions to control some processes.

Development of the fuzzy set theory started in the 1960s (Zadeh, 1965) and up to now, fuzzy logic has been successfully used in the management of processes in many different areas. Fuzzy logic has been used in solving transportation and traffic problems. The first

implementation of fuzzy logic theory was in the control system for isolated signalized intersections of two one-way streets (Pappis and Mamdani, 1977).

The fuzzy logic system can be described as mapping of an input data into a scalar output (Mendel, 1995). Most often mapping is crisp input into crisp output.

Fuzzy rules could be defined from the knowledge of experienced operators used in control systems. The rules could be collected – produced by verbalizing the operator's expertise or by conducting a carefully composed survey. In both cases, rules are formed by observing the decisions being made. Decisions are composed based on some input data and finally input-output data pairs could be seen.

When it can be positively claimed that the operator is consciously or unconsciously using the rules in managing, the rules can be recorded by observing the operator's behavior. In other words, the rules can be formulated by using the observed decisions (input/output numerical data) of the operator.

Fuzzy rules include descriptive expressions such as small, medium, or large used to categorize the linguistic (fuzzy) input and output variables. A set of fuzzy rules, describing the control strategy, forms a fuzzy control algorithm, that is, an approximate reasoning algorithm, where the linguistic expressions are represented and quantified by fuzzy sets. The main advantage of this approach is the possibility of introducing and using rules from experience, intuition, heuristics, and the fact that a model of the process is not required. Fuzzy reasoning (approximate reasoning) involves the transformation of a group of fuzzy rules into fuzzy relations in order to achieve a result. Fuzzy reasoning is an inference procedure, i.e., the method of generating the conclusion from the premises when the linguistic expressions are quantified by fuzzy sets. The inference engine of the fuzzy logic system maps fuzzy sets into fuzzy sets. The inference engine "handles the way in which rules are combined" (Mendel, 1995). There are a number of various inferential procedures in literature.

Fuzzy rule (fuzzy implication) could be presented in the following form:

If x is **A**, then y is **B**

where **A** and **B** represent linguistic values quantified by fuzzy sets defined over universes

of discourse X and Y. The first part of the rule "*x* is **A**" is *premise* or *hypothesis* or *antecedent*, the second part of the rule "*y* is **B**" forming *conclusion* or *consequent*.

The entire expression describes an existing relation between the variables *x* and *y*; a fuzzy rule is defined as a binary fuzzy relation **R** in the Cartesian space X × Y = {(*x*, *y*)}. Each element (*x*, *y*) of the fuzzy relation is associated with the corresponding membership grade $\mu_{\mathbf{R}}(x, y)$. The binary fuzzy relation **R** can be treated as the fuzzy set defined in space X × Y characterized by a two-dimensional membership function $\mu_{\mathbf{R}}(x, y)$.

In cases where "x is $\mathbf{A}^*$" is a fact, after application of fuzzy implication fuzzy set $\mathbf{B}^*$ could be expressed in the following way:

$$\mu_{B*}(y) = \max_x \min\{\mu_{A*}(x), \mu_R(x, y)\}$$
$$\mu_{B*}(y) = q \wedge \mu_B(y)$$

In order to determine fuzzy set $\mathbf{B}^*$, the degree of match of sets **A** and $\mathbf{A}^*$, represented by *q*, is first defined as the maximal value of the intersection of the sets; as it is shown on figure 3.16. Fuzzy set $\mathbf{B}^*$ is determined by the intersection of fuzzy sets **B** and *q*.

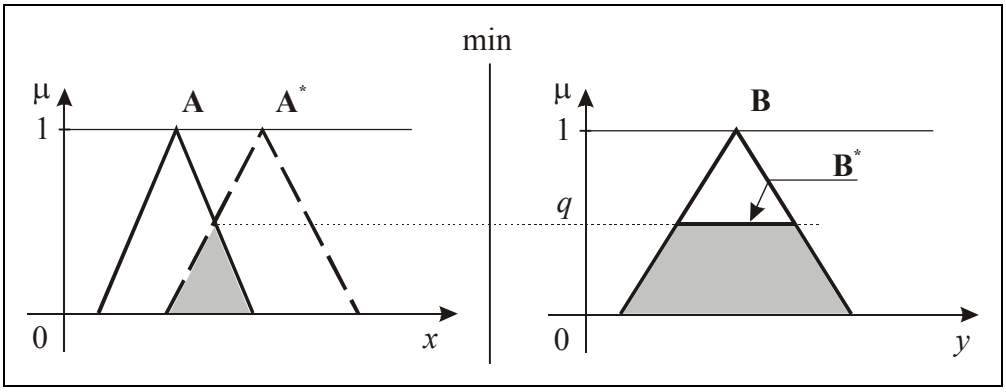

Figure 3.16 – Approximate reasoning for a single rule

Let us introduce the following rule for instance:

If x_1 is $\mathbf{A_1}$ and x_2 is $\mathbf{A_2}$, then *y* is **B**

At the same time let us assume the following facts:

x_1 is $\mathbf{A}^*_\mathbf{1}$ and x_2 is $\mathbf{A}^*_\mathbf{2}$

First, the degree of match of sets **A** and $\mathbf{A}^*_\mathbf{1}$, q_1, should be determined, then the degree of

match of sets $\mathbf{A_2}$ and $\mathbf{A^*_2}$, q_2, and in final, the resulting fuzzy set $\mathbf{B^*}$ is obtained by the intersection of fuzzy set **B** and q = min (q_1, q_2); as it shown in figure 3.17. In this procedure the operator *min* represents operation "and."

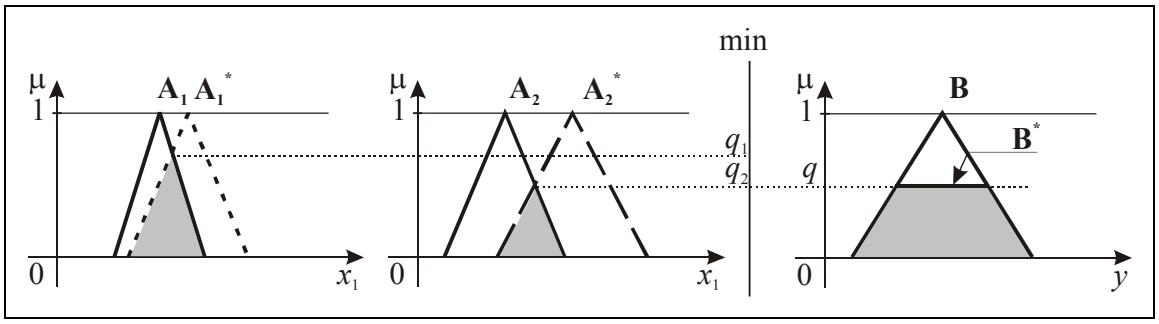

Figure 3.17 – Approximate reasoning for a single rule with two elements in premise

A set of rules presents union of fuzzy relations. Let us consider the following example:

If x_1 is $\mathbf{A_1}$ and x_2 is $\mathbf{B_1}$, then y is $\mathbf{C_1}$
If x_1 is $\mathbf{A_2}$ and x_2 is $\mathbf{B_2}$, then y is $\mathbf{C_2}$

Operator max represents operation "or," and the composition of inferences could be found under name "max-min composition" in literature. Mamdani and Assilian (1975) proposed the first model of the fuzzy system where the approximate reasoning is performed by max-min composition. Figure 3.18 shows the approximate reasoning performed by max-min composition when the algorithm is composed of two rules – given example.

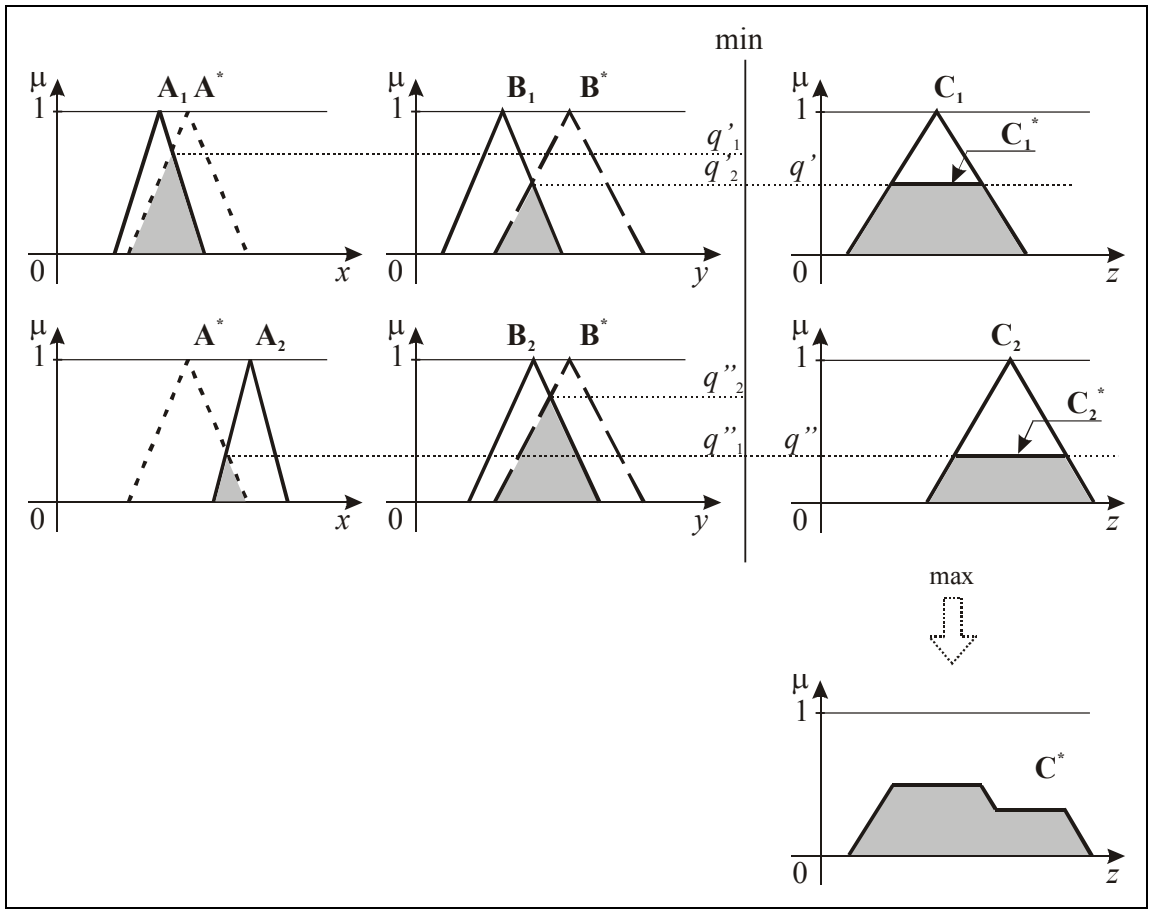

Figure 3.18 – Approximate reasoning algorithm when two rules are present

Practically, the last step in the approximate reasoning algorithm is defuzzification; choosing one value for the output variable. Using the fuzzy reasoning algorithm, a fuzzy set is obtained as the output result of max-min composition. By defuzzification, the fuzzy information is compressed and given out by representative numerical information. Mamdani and Assilian (1975) used the center of gravity of the resulting fuzzy set as a representative output numerical value. In most applications an analyst or decision maker will decide about representative value. Besides choosing the center of gravity, "the smallest maximal value," "the largest maximal value," "mean of the range of maximal values," and so on (Figure 3.19) also could be chosen.

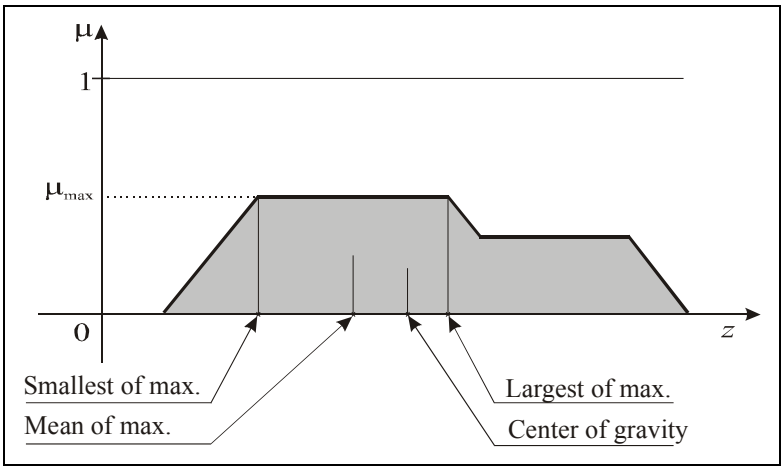

Figure 3.19 – Approximate reasoning using max-min composition

3.4.5 Generating fuzzy rule base from numerical examples

In order to generate fuzzy rule bases from numerical examples, the procedure proposed by Wang and Mendel (1992) was used. We could establish fuzzy sets for all the antecedents and the consequences. We will do it in such a way that, at the very beginning, we will establish the domain intervals for all input and output variables (Figure 3.20).

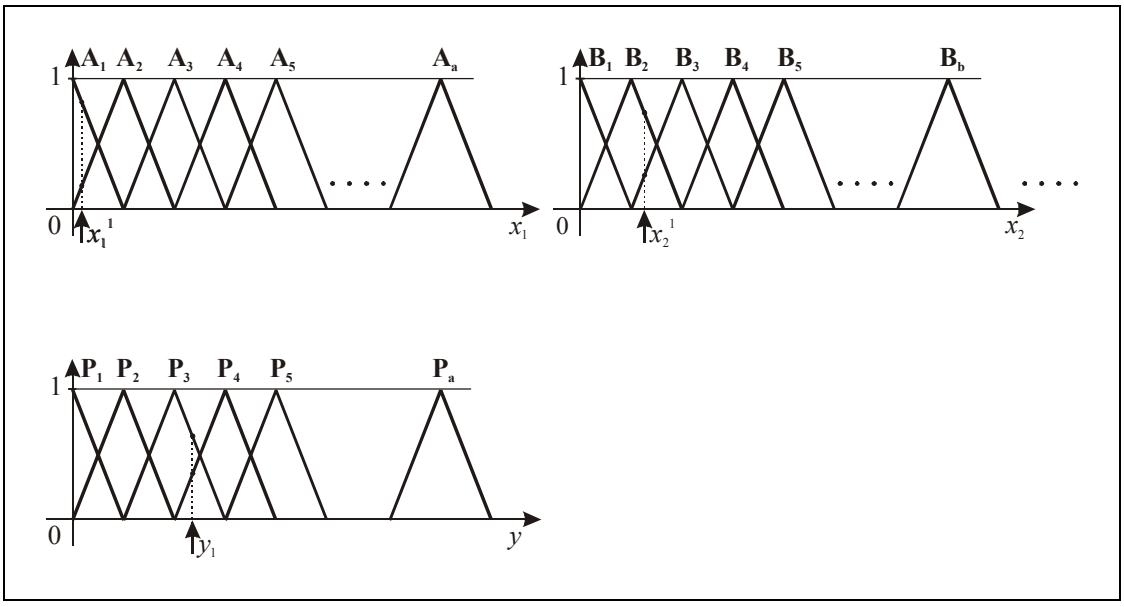

Figure 3.20 – Division of the domain interval into a prespecified number of overlapping regions

As Wang and Mendel (1992) suggested, it is possible to divide each domain interval into a prespecified number of overlapping regions (Figure 3.20). The number of overlapping regions is not the same for each variable. The lengths of these overlapping regions are usually equal, but not necessarily. In the next step each overlapping region is labeled and one membership function is assigned to it. Furthermore, different types of membership functions for different variables can be used. According to figure 3.20, let us assume that triangular membership functions for all variables will be used. Let us assume that the following set of the input-output data pairs is available:

$$(x_1^1, x_2^1, \ldots y^1), (x_1^2, x_2^2, \ldots y^2), (x_1^3, x_2^3, \ldots y^3) \tag{3.18}$$

where x_1, x_2,… are input and y is output. We will generate the fuzzy rule base from this set of input-output data pairs. The values of x_1, x_2,… and y belong to the domain intervals (x_{1min}, x_{1max}), (x_{2min}, x_{2max}), … (y_{min}, y_{max}) respectively. From every input-output data pair we can eventually generate one fuzzy rule. The next step is to explain how to generate fuzzy rule from the first input-output data pair $(x_1^1, x_2^1, \ldots y^1)$.

We have to determine the membership function values of the elements. From figure 3.20 can be seen that x_1^1 has degree 0.2 in $\mathbf{A}_2$, and 0.8 in $\mathbf{A}_1$, also, x_2^1 has degree 0.7 in $\mathbf{B}_2$ and 0.3 in $\mathbf{B}_3$, … and y^1 has degree 0.4 in $\mathbf{P}_4$, and 0.6 in $\mathbf{P}_3$. The next step is to assign each variable to the region with maximum degree. This means that x_1^1 is considered to be $\mathbf{A}_1$, for x_2^1 is $\mathbf{B}_2$, …, and y^1 is considered to be $\mathbf{P}_3$. The rule, which we obtained from the first pair of the input-output data, is:

If x_1 is $\mathbf{A}_1$ and x_2 is $\mathbf{B}_2$ and …

Then y is $\mathbf{P}_3$

This is the way to generate the rules from the input-output data pairs. Because many input-output data pairs are generated, some conflicting rules can be produced. The conflicting rules have the same antecedents but different consequence. Wang and Mendel (1992) resolved this “by assigning a degree to each rule and accept only the rule from a conflict group that has maximum degree”. The degree of a rule 1, which we got from the first input-output data pair, equals:

$$D(Rule\ 1) = \mu_{A_1}(x_1^1) \cdot \mu_{B_2}(x_2^1) \cdot \ldots \mu_{P_3}(y^1) = 0.8 \cdot 0.7 \ldots \cdot 0.6 \tag{3.19}$$

Finally, following this procedure it is possible to generate a fuzzy rule base containing a minimum of one rule (all data pairs are similar and produce the same rule or all produced rules are conflicting each other) and a maximum number of rules equal to the number of input-output data pairs.

To apply Wang and Mendel's procedure it is necessary first of all to define all variables x_i (i = 1, 2, 3, …) that would appear in premise and variable y (consequence). It is important to simultaneously consider appropriate domain when defining any of these variables.

A developed decision support system should provide judgment about visiting the next node (t_{i+1}) in the "giant" route, once the vehicle has visited a previous node (t_i) in the route. Important data that should be considered when making decision would be the following:

- C_r – remaining of the vehicle capacity,
- C – vehicle capacity,
- $\mu_{t_{i+1}}$ – mean of the demand in the following node, and
- $\sigma_{t_{i+1}}$ – standard deviation of the demand in the following node.

When considering the Stochastic Vehicle Routing Problem we have introduced the following three variables for the antecedent part:

- $x_1 = \frac{C_r}{C}$,
- $x_2 = \frac{\mu_{t_{i+1}}}{C}$, and
- $x_3 = \frac{\sigma_{t_{i+1}}}{C}$.

Consequent (y) is number of nodes to be visited after node t_i.

All introduced variables are bounded; x_1, x_2 and x_3 are ranging from 0 to 1 and y is ranging from 0 to the total number of nodes.

3.4.6 Numerical example

We have considered the same set of ten TSP examples introduced in section 3.3.2. To transfer original TSP problem into Vehicle Routing Problem, in all the examples, the first node was considered to be depot. Originally, examples just contain node coordinates (they were prepared to fit TSP needs).

The following procedure was used to define/generate random demand associated to each of the nodes:

1. Define vehicle capacity and the ranges for mean and standard deviation of demand at nodes.
2. For each particular node define values for mean and standard deviation as random number taken from uniform distribution within predefined range. This procedure is not completely random because it is necessary to avoid the possibility to have "actual" demand as negative value.
3. Based on values for mean and standard deviation for every node define "actual" demand as a random number taken from the Normal distribution with predefined mean and standard deviation.

In all examples we have vehicle capacity and the ranges for mean and standard deviation defined as follows:

- Vehicle capacity is 1000,
- Range for mean is (0, 200) and
- Range for standard deviation is (0, 60).

The basic assumption was that demand at a node is distributed according to Normal probability density function. Through simulation we have assigned values for mean and standard deviation and then "actual demand" (based on previously assigned parameters of the distribution) to every node. The entire set of cases (experiments) is subdivided into two parts: a training set (based on training set fuzzy rule base is produced) and a control set (the control set is used to check the quality of the developed fuzzy rule base).

When producing the fuzzy rule base in all the cases we have used triangular fuzzy numbers, uniformly distributed over the entire domain interval for variables that appear in the antecedent and consequent parts. The number of uniformly distributed fuzzy

numbers associated to each of variables x_1, x_2 and x_3 was nine and we have associated 17 fuzzy numbers to y (consequent part).

Figures 3.21 and 3.22 show the error distribution in all 10 considered examples. When producing the error distribution we consider just results obtained on the test set.

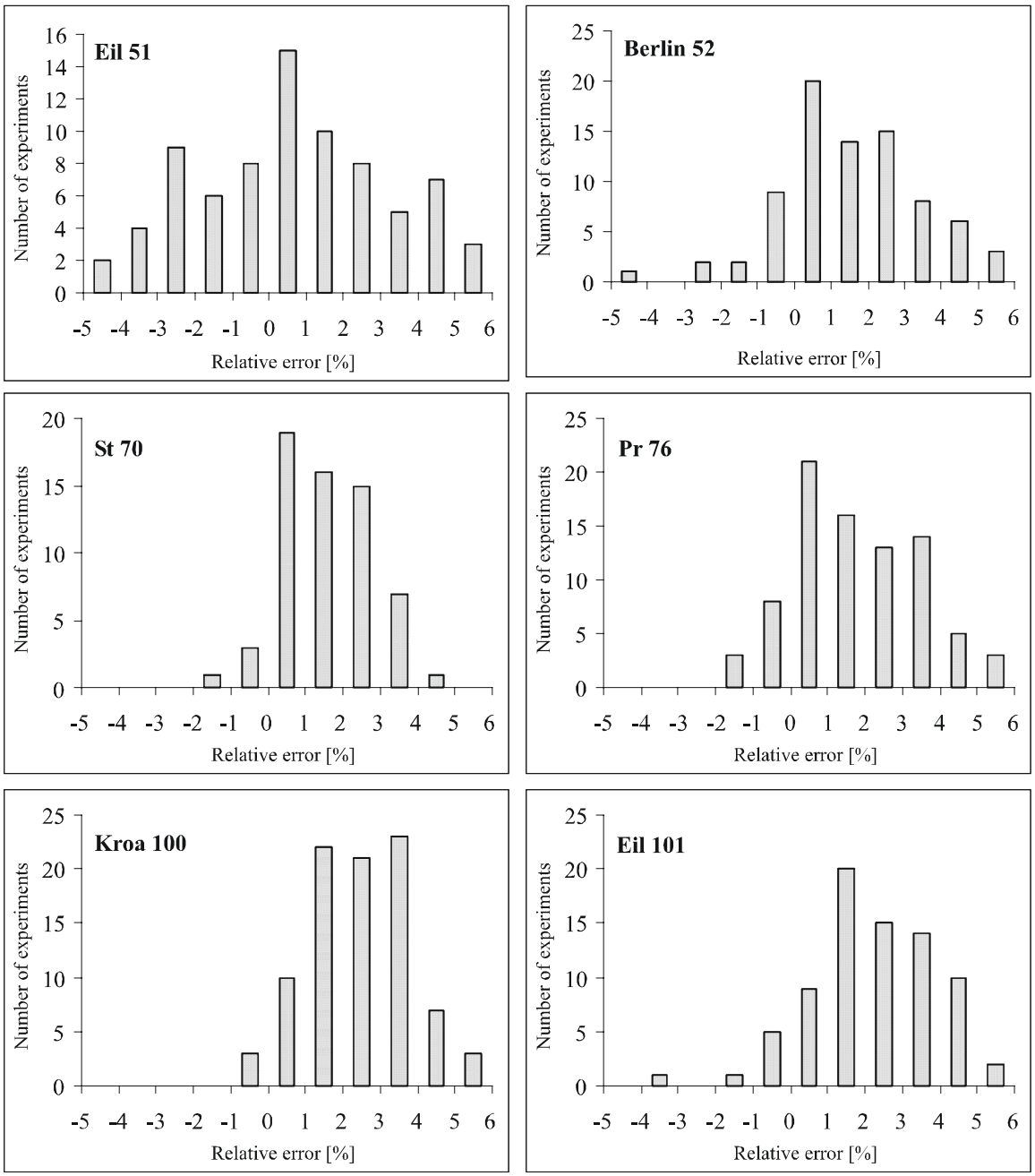

Figure 3.21 – Error distribution for the following examples: Eil51, Berlin52, St70, Pr76, Kroa100 and Eil101

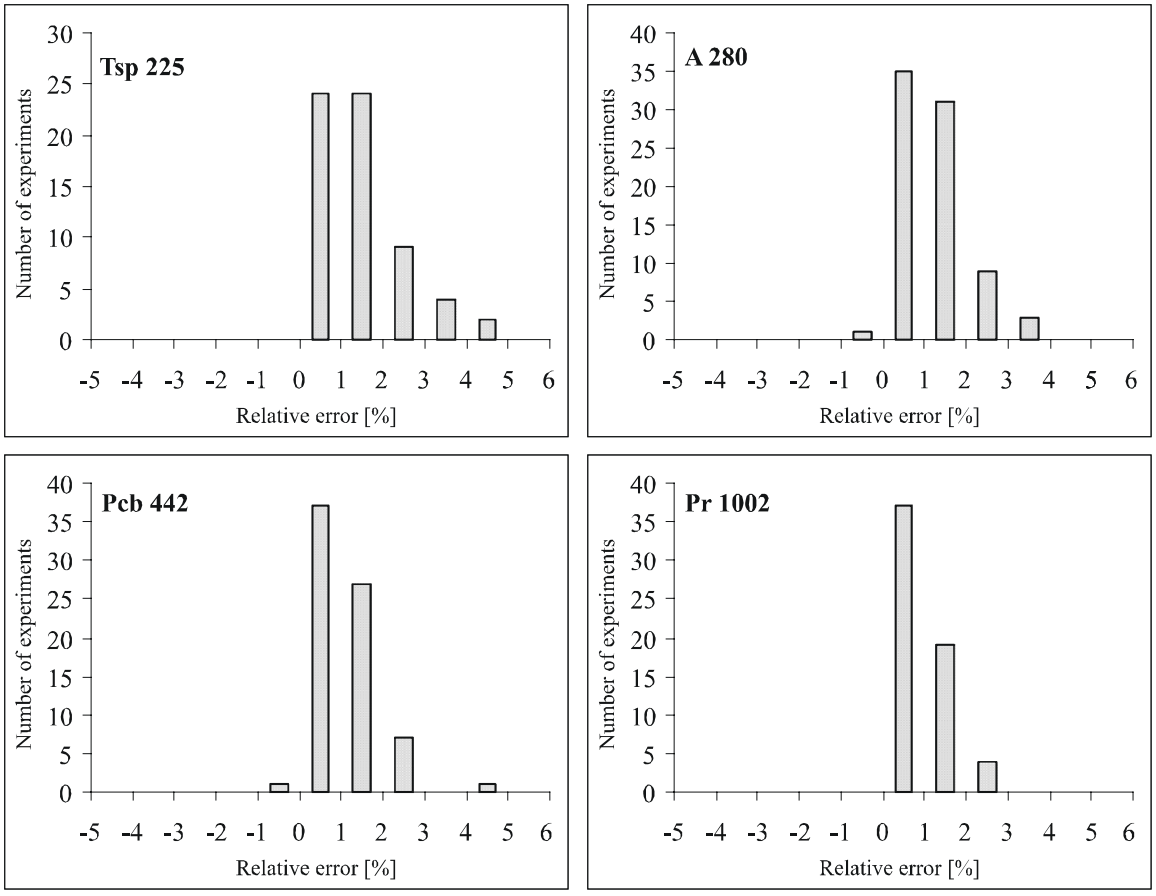

Figure 3.22 – Error distribution for the following examples: Tsp225, A280, Pcb442 and Pr1002

Each experiment has two solutions associated with it. One is the solution obtained under conditions where future demand is known in advance (a priory) and the other one is solution obtained by the developed Fuzzy Logic based system. Those two solutions are associated with each particular experiment. This could be referred to as the solution pair. Graphical representation of the solution pair obtained as result for one random demand assignment to the nodes (one experiment) is shown for the following examples: Berlin52, Pr76, Eil101 and A280.

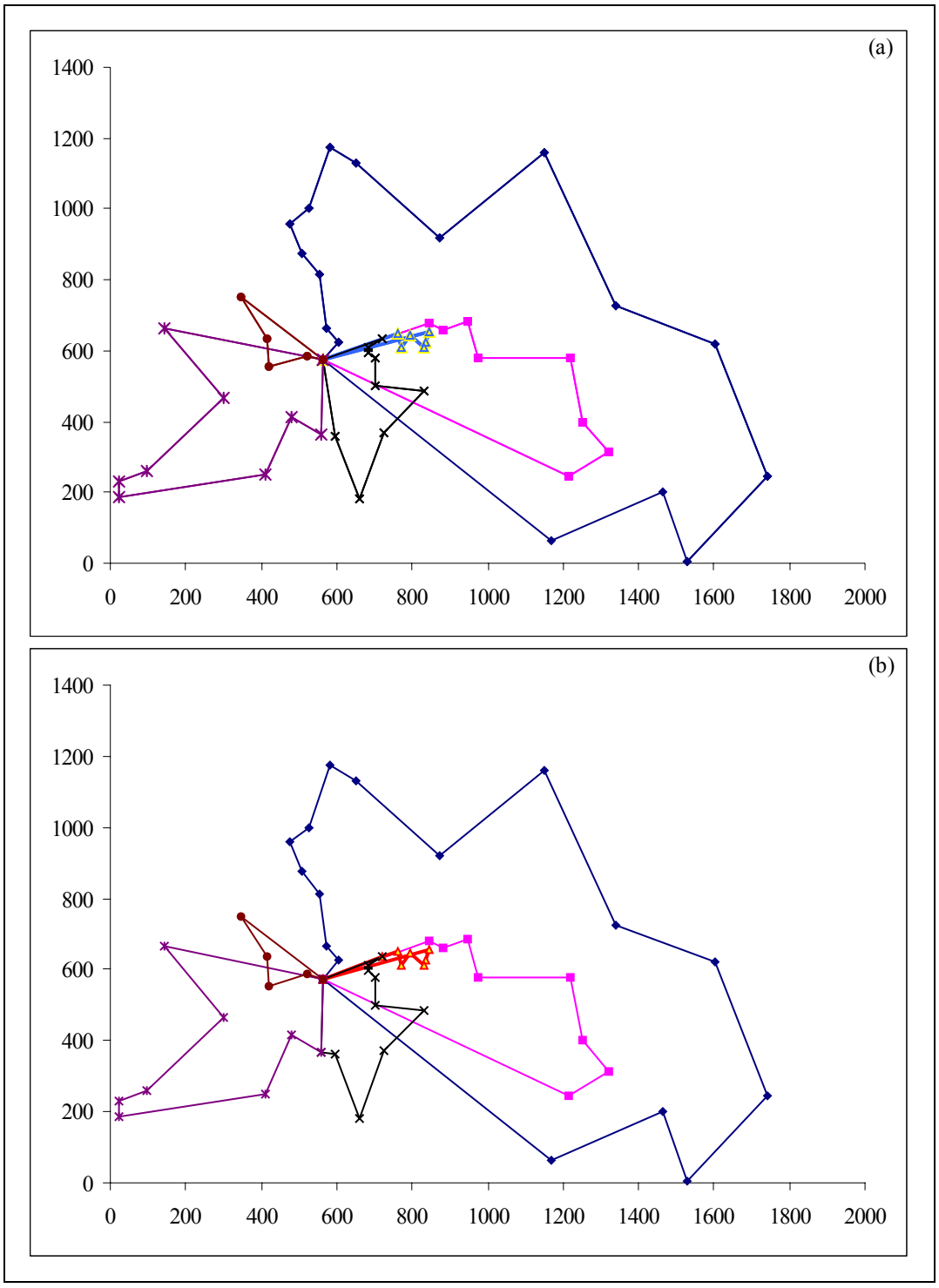

Figure 3.23 – One solution pair for Berlin52 example ((a) perfect future knowledge and (b) fuzzy system solution)

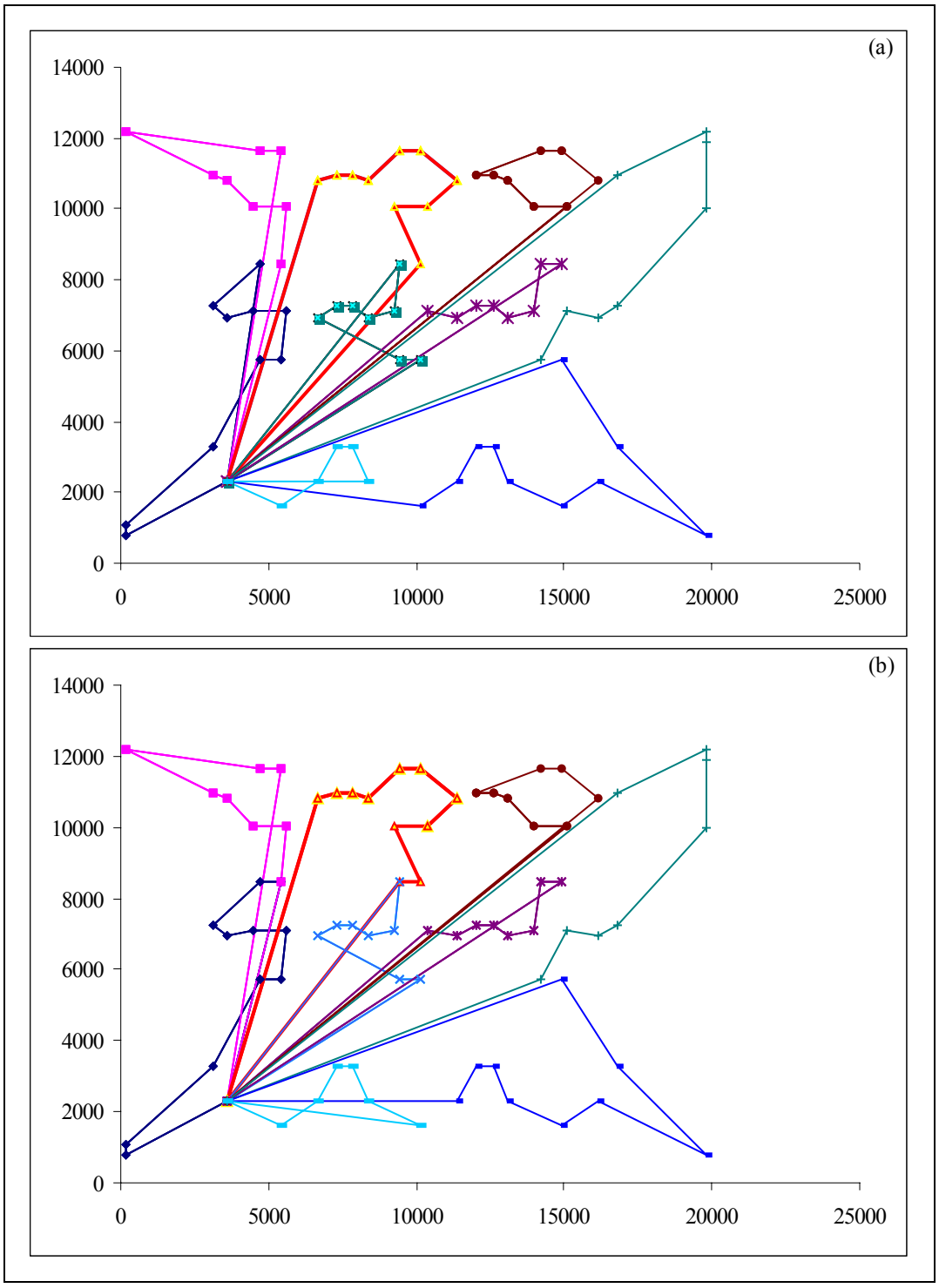

Figure 3.24 – One solution pair for Pr76 example ((a) perfect future knowledge and (b) fuzzy system solution)

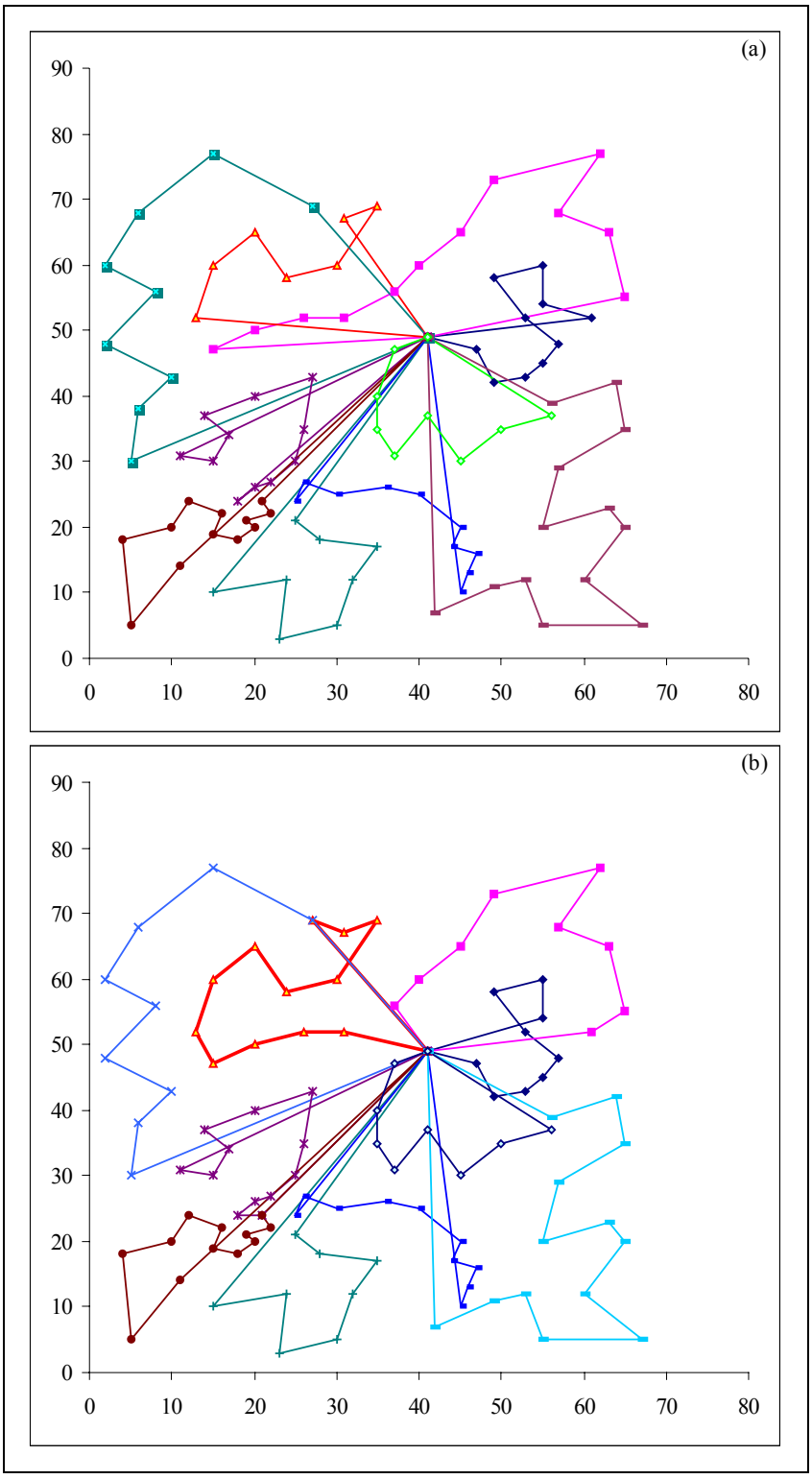

Figure 3.25 – One solution pair for Eil101 example ((a) perfect future knowledge and (b) fuzzy system solution)

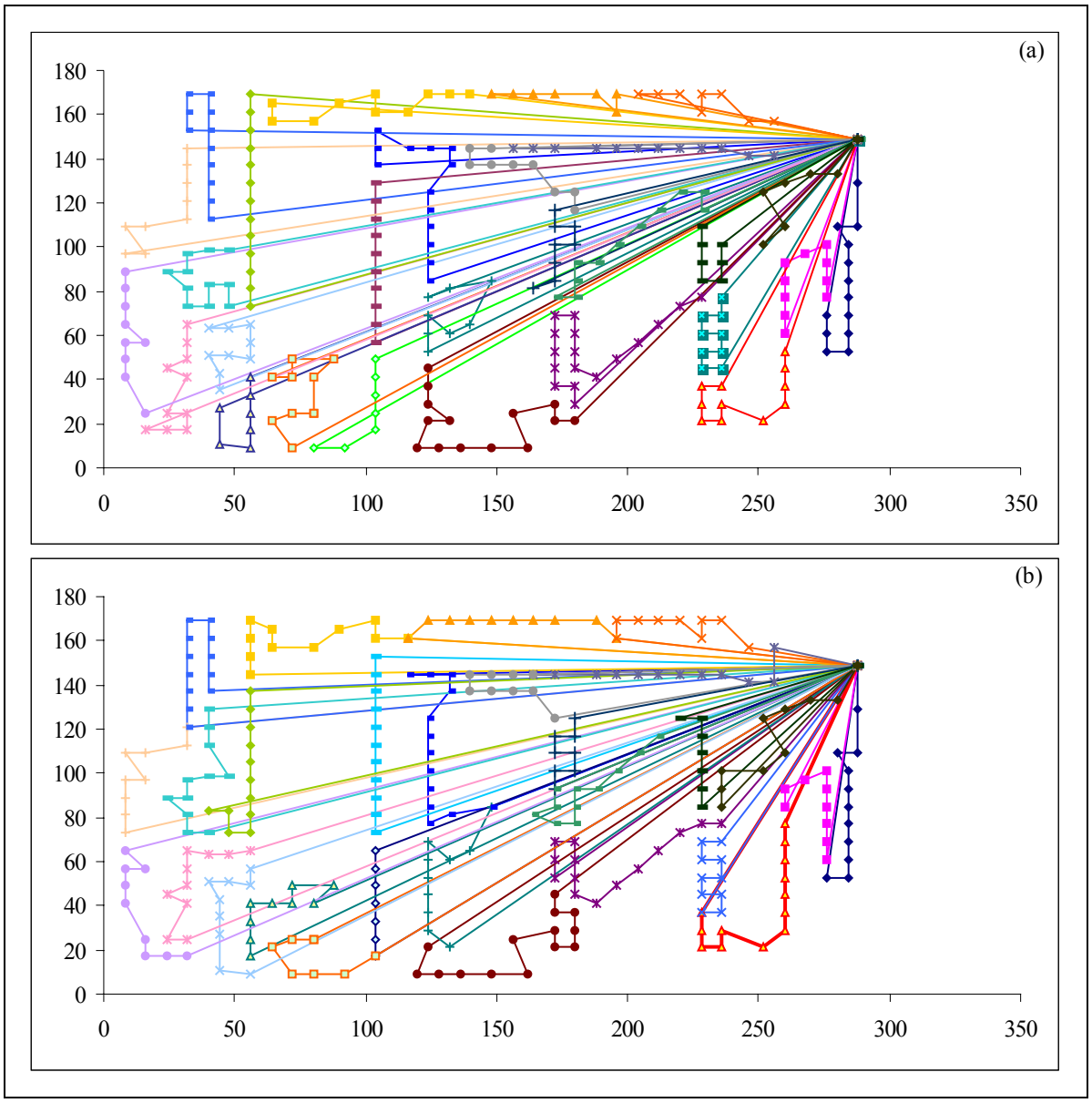

Figure 3.26 – One solution pair for A280 example ((a) perfect future knowledge and (b) fuzzy system solution)

Maximum and average values of relative error for all considered examples are provided in table 3.4.

Table 3.4 – Maximum and average values of relative error

	Maximum of relative error [%]	Average relative error [%]
Eil51	5.83	0.607
Berlin52	5.98	1.44
St70	4.05	1.63
Pr76	5.83	1.75
Kroa100	5.75	2.42
Eil101	5.44	2.18
Tsp225	4.70	1.49
A280	3.48	1.26
Pcb442	4.32	1.15
Pr1002	2.38	0.93

Chapter 4. Combined Ant System and Fuzzy Logic Modeling Approach

4.1 Behavior of real ants

As social insects, ants live in colonies. Ants' behavior is directed towards the survival of the colony as a whole rather then achieving individual goals. From an optimization point of view, one of the most important behaviors of ant colonies is foraging behavior. More precisely, how ants can find the shortest path between their nest and food source. Ethnologists tried to solve the same problem. The question was how almost blind animals like ants could establish and manage the shortest route from their nest to a feeding source and back.

When walking from the nest to the food source and vice versa, ants will deposit on the ground a substance called pheromone. When one or several ants use the same path, a pheromone trail will be formed. Ants are able to smell pheromone trails. Under the presumption that ants choose their future direction of walk based on some probability, it is obvious that a stronger pheromone concentration will cause a higher probability for that path to be chosen. Using pheromone trails, ants are able to find the way back to the nest or food source. In the same manner, other ants could use pheromone trails in order to find food sources discovered by their nestmates.

In case of the existence of several possible paths, it has been shown that pheromone trail following behavior employed by a colony of ants will emerge into choosing the one with the shortest path between the nest and food source. These conclusions come from experimental study of ants' behavior. One experiment that will be briefly described here is the binary bridge experiment that has been set up by Deneubourg et al. (1990). The nest of ant colony is separated from the food source by a double bridge (figure 4.1) where each branch has the same length.

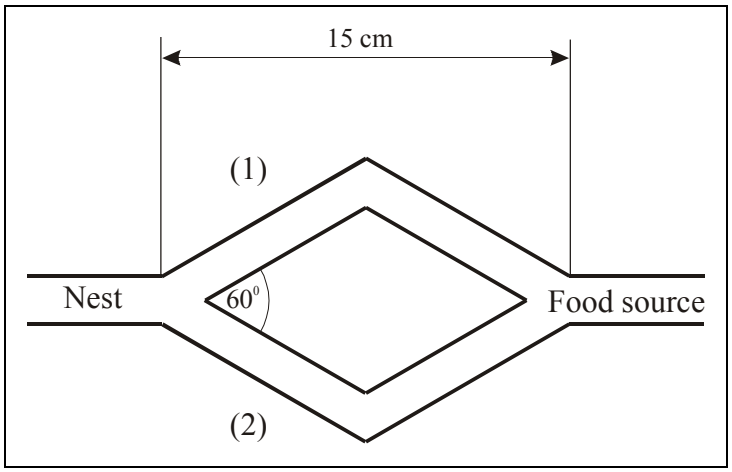

Figure 4.1 – Initial condition of the "double bridge" experiment

Ants are then left to move freely from the nest to the food source and vice versa. The percentage of ants that choose one or the other of the two branches is observed over time. The result shows that after an initial transition phase where some oscillations appear, ants converge to one (the same) path. At the beginning of the experiment there is no pheromone trail on the two branches and ants select branches with the same probability. Randomness after initial phase would cause that a few more ants will select one branch over the other. While walking, ants deposit pheromone and a greater number of ants on one branch will finally give a greater amount of pheromone on it. The produced pheromone trail will stimulate more ants to choose the same branch and so on. Goss et al. (1989) have presented the following probabilistic model describing this phenomenon:

$$P_1(m) = \frac{(N_{1,m} + k)^h}{(N_{1,m} + k)^h + (N_{2,m} + k)^h} \tag{4.1}$$

where:

$N_{1,m}$ and $N_{2,m}$ – number of ants that have used branches denoted by "1" and "2" after m ants have crossed bridge ($N_{1,m} + N_{2,m} = m$),

$P_1(m)$ – probability that m+1-st ant will choose "1" branch,

k, h – tuning parameters.

In order to test for correspondence between model and real (experimental) data, the Monte Carlo simulation was performed. To determine which branch an ant will use, after

calculating corresponding probabilities, a random number ψ was chosen. In cases where $\psi \leq P_1(m)$ is fulfilled $N_{1,m+1} = N_{1,m} + 1$; $N_{2,m+1} = N_{2,m}$; otherwise, $N_{1,m+1} = N_{1,m}$; $N_{2,m+1} = N_{2,m} + 1$. Random variable ψ has uniform distribution over the interval [0, 1]. Simulations were performed with the following values of the parameters: $k \approx 20$ and $h \approx 2$ (Pasteels et al., 1987).

The authors built the model with the following assumptions:

- the amount of pheromone on the branch is proportional to the number of ants that used that branch in the past (basic assumptions are: every ant will deposit pheromone with the same rate and ants are walking with approximately the same speed),
- evaporation of the pheromone is not considered (applicable in case of providing experiment for a short period of time).

The model closely matches the experimental observations.

In cases where bridge branches have different lengths, the difference in the amount of time that ants spend to cross the bridge (using different branches) causes the shorter branch to be passed more times than longer branch. Initially, transition period in this case will be much shorter. Finally, the shorter branch would have more pheromone that will, in turn, stimulate a significant number of ants to use the shorter branch. Occasionally, some ants will use the other route.

In principle, a single ant is able to find the route from the nest to the food source and vice versa (build some solution to the problem). However, only the ant colony is capable of finding the "shortest" path (shortest path finding behavior is a property of ant colony only). The ant colony is able to perform this specific behavior using a simple form of indirect communication (based of pheromone deposit) known as *stigmergy*.

The main characteristics that separate stigmetry from other means of communications are the following:

- information is released by the changing of the physical environment of the site visited by the ant, and
- information can be accessed by an ant that is visiting the same site (or neighboring site) – information has a local nature.

Additionally, experiments show that ants are capable of reconnecting the developed "shortest" path (Figure 4.2a) when an obstacle appears suddenly. When an obstacle appears, ants located in front of the obstacle are not able to continue to follow the pheromone trail and they have to choose to turn right or left (Figure 4.2b). In both directions a pheromone trail does not exist and expectation is that half of the ants will turn left and another half to turn right (Figure 4.2c). The same situation could be found on the other side of the obstacle, for those ants traveling in the opposite direction.

The difference in traveling time will cause those ants which choose the shorter path around the obstacle by chance, to more rapidly form a pheromone trail. This trail will be stronger than the trail created by the ants which choose the longer path. The higher amount of pheromone will cause more ants to choose the shorter path. Due to this positive feedback process, in a short period of time, all of the ants will use the shorter path (Figure 4.2d).

It has been observed that in ants foraging behavior it is not essential to have the following properties (Pasteels et al., 1987):

- visual clues, and
- individual memory (including left-right memory).

The same results were obtained with and without possibility of their use.
The experimental study also showed that the ant colony is incapable of switching to the short path, due to the irreversible nature of the positive feedback process involved, when shorter path is only present after the trail on long path has been established (Pasteels et al., 1987).

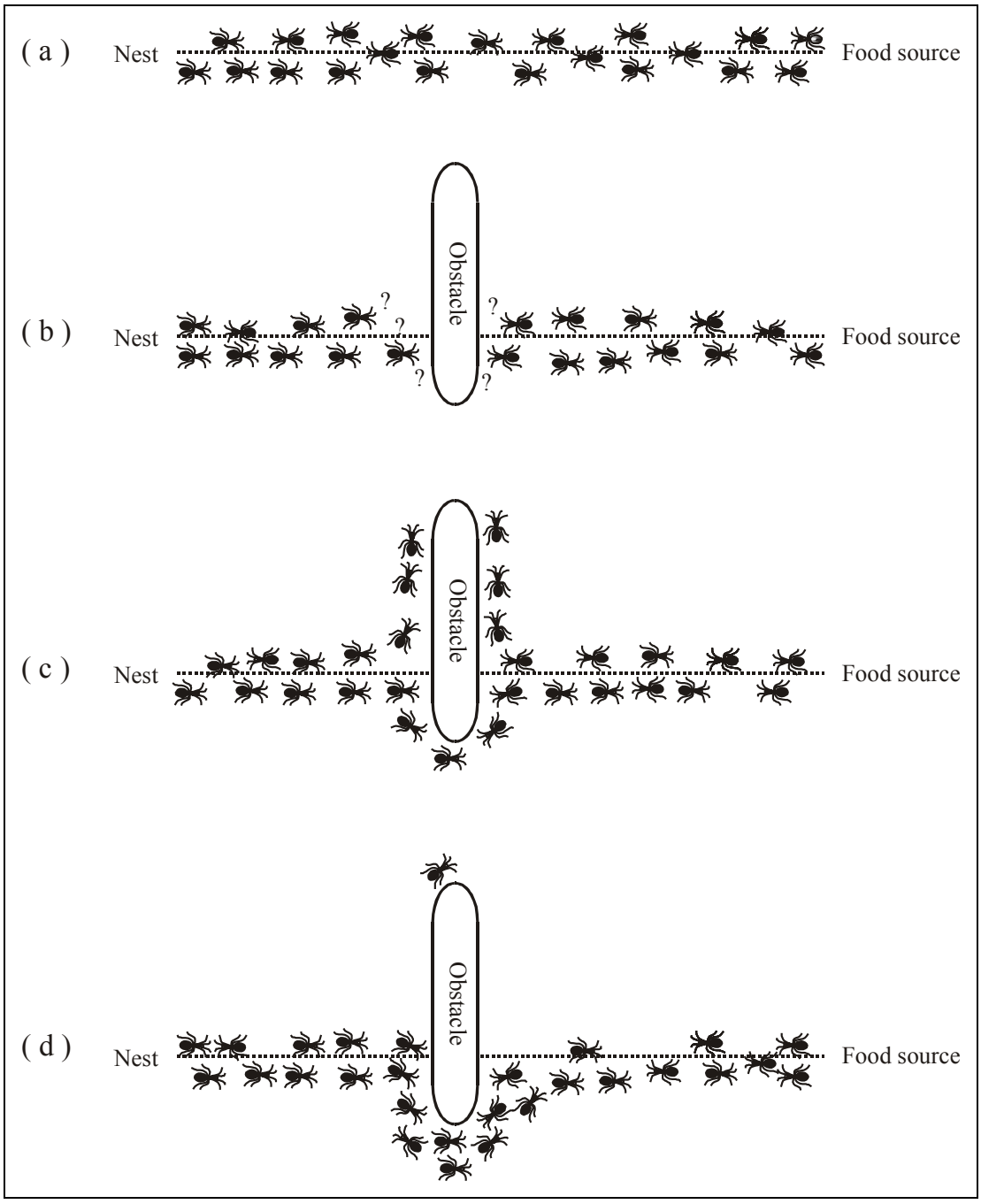

Figure 4.2 – Example of real ant behavior. (a) ants following the shortest route between the nest and food source, (b) suddenly an obstacle appears – ants have to choose to turn left or right, (c) approximately half of the ants would use either direction, (d) because of a stronger pheromone trail, ants will choose the new shorter path.

4.2 Ant System

The Ant System (AS) is a general-purpose heuristic algorithm that could be used to solve various combinatorial optimization problems. This algorithm has been developed based on an analogy with the foraging of real ant colonies. The transition from real ants to artificial is described based on the first presented examples Colorni et al. (1991, 1992) and Dorigo et al. (1996). The authors introduced AS by applying it to solve the Symmetric Traveling Salesman problem without time windows (TSP) using AS.

TSP is the problem of finding a shortest closed tour such that the tour will visit all the cities in a given set exactly once. TSP could be formulated in the following way:

Consider a sequence of n towns, and for each town pair (i, j) consider a distance d_{ij}. Aim is to find permutation π of n towns that will minimize the quantity:

$$\sum_{i=1}^{n-1} d_{\pi(i),\pi(i+1)} + d_{\pi(n),\pi(1)} \tag{4.2}$$

In this example attention will be restricted to TSP in which cities are on a plane and a path (edge, arc) exists between each pair of cities (i.e., the TSP graph is completely connected). All distances are Euclidean.

Let us introduce graph G = (N, A, d), and denote by:

$n = |N|$ - number of cities that ant should visit, and

m – total number of ants.

Formally, ants will make tours through steps; they will travel over the introduced graph by walking from one city to another. When all ants have produced routes (TSP solutions), they will do that again, so on.

Let us introduce the following indexes:

t – cycle index (will give information how many times ants make set of tours),

s – iteration index (it will give information how many steps all ants have made in the current cycle – every cycle contains n iterations).

Colorni et al. (1991, 1992) and Dorigo et al. (1996), have considered each ant as a simple agent that chooses the town to go with a probability that is a function of the town distance and of the amount of pheromone trail present on the connecting edge, to explain the transition from real ants to artificial ones.

The transition probability from town *i* to town *j* for the *k*-th ant is:

$$p_{ij}^{k}(t,s) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in J_i^k(t,s)} [\tau_{il}(t)]^{\alpha} \cdot [\eta_{il}]^{\beta}} & \text{if } j \in J_i^k(t,s) \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

where:

$\tau_{ij}(t)$ – pheromone trail on arc (i, j) in cycle t,

$\eta_{ij} = 1/d_{ij}$ – "visibility",

$J^k_i(t, s) = \{N – tabu_k (t, s)\}$ – set of nodes where ant k can go from node i,

$tabu_k (t, s)$ – *tabu list* for ant k in cycle t and iteration s,

α, β – parameters given in advance.

To force the ant to make legal tours, transitions to already visited towns are disallowed until a tour is completed (this is controlled by a *tabu list*).

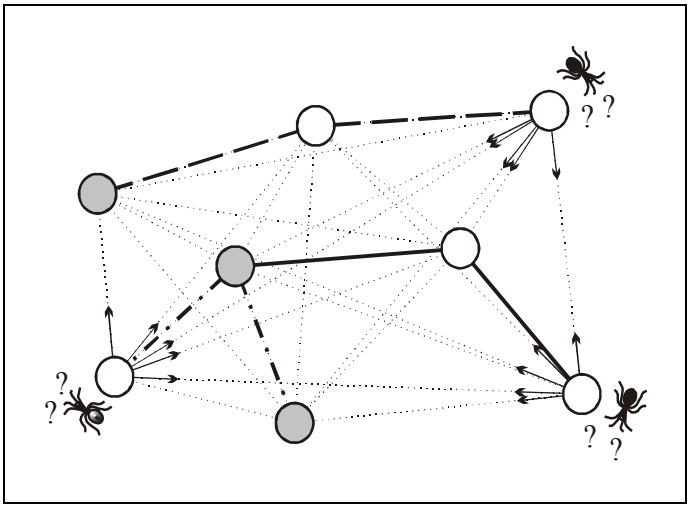

Figure 4.3 – Graph and search process that could happen in any cycle after $s = 2$ iterations where search process is carried out by $m = 3$ ants. Dark nodes are the ones where the ants are initially randomly placed.

Placed at the towns at the beginning randomly, ants explore the given graph simultaneously. Each ant in iteration *s* will choose the next town, where it will be in iteration *s*+1. The progression through iterations (figure 4.3) would exist until all ants have completed their tours. After the completion of their tours, ants will lay an *artificial pheromone trail* on each visited edge (i, j). Thereafter, a new cycle could start.

Again, an *iteration* of the AS algorithm contains the *m* moves carried out by the *m* ants. Every *n* iterations of the algorithm, each ant has completed a tour, and the cycle of the algorithm is also completed. At this point, the trail intensity is updated according to the following equation:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij}(t, t+1) \tag{4.4}$$

where :

ρ – coefficient such that $(1-\rho)$ represents the evaporation of trail between cycle *t* and $t + 1$. This coefficient is given in advance with value in interval $(0, 1]$.

$\Delta\tau_{ij}(t, t+1)$ – "additional" pheromone on arc (i, j) produced in cycle *t*. This amount of pheromone will be available for all ants through entire iteration $t + 1$.

Additional pheromone could be calculated using the following equation:

$$\Delta\tau_{ij}(t, t+1) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \tag{4.5}$$

$\Delta\tau^{k}_{ij}(t)$ – the quantity per unit of length of trail substance laid on edge (i, j) by the *k*-th ant at the end of cycle *t*. There are several ways to calculate this value – one common method is:

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \dfrac{Q}{L_k(t)} & \text{if } k-\text{th ant use arc } (i, j) \text{ in its tour defined in cycle } t \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

Q – constant given in advance,

$L_k(t)$ – produced tour length of the *k*-th ant in cycle *t*.

In order to initialize the process, some initial pheromone $\tau_{ij}(0) = c$ will be assigned to

every arc (c is small positive value).

Equation 4.3 contains two parameters α and β that have considerable role in the final probability values. If $\alpha = 0$, the closest city is more likely to be selected; it corresponds to the classical stochastic greedy algorithm that has multiple starting points (initially ants are located in cities randomly). On the other hand, if $\beta = 0$, pheromone trail intensity on arcs will influence the probabilities for stochastic node choice. This will lead the search process to rapid convergence – stagnation (generating tours that are similar). It is clear that trade-off among consideration of visibility and the pheromone trail should exist. Only values of α and β other than 0 will make it possible for ants to explore the feasible region in the way that would increase the chances to get a good enough or optimal solution to the problem.

The following algorithm presents the main steps that this developed AS algorithm contains.

For every arc (i, j) **do** $\tau_{ij}(0) = c$;
For $k = 1$ to m **do** Place ant k on a randomly chosen city;
Let T^+ be the shortest tour found from beginning and L^+ its length;
For $t = 1$ to t_{max} **do**
    For $k = 1$ to m **do**
        Build tour $T^k(t)$ by applying n - 1 times transition probability (4.3). Always updating tabu list ($tabu_k(t, s)$);
        Complete $T^k(t)$ – add returning link to the starting city of the tour $T^k(t)$; i.e. provide last (n-th) step.
    End For
    For $k = 1$ to m **do**
        Compute the length $L^k(t)$ of the tour $T^k(t)$ produced by ant k;
    End For
    If an improved tour is found **then** update T^+ and L^+;
    For every arc (i, j) **do**
        Update pheromone trails by applying the rule (4.4 – 4.6);
    End For
End For
Print T^+ and L^+;

In contrast to the real ants, it could be seen from the algorithm that artificial ants do not deposit pheromone when traveling between nodes. Pheromone trail will be updated at the

end of every cycle. In the literature, it could be found that the name of this algorithm is the Ant-Cycle algorithm. Among others (Ant-density and Ant-quantity), Ant-cycle algorithm is the most successful.

All these algorithms have similar structure, and the main difference is seen in the pheromone updating rule. Ant-density and Ant-quantity algorithms presume that pheromone will be updated when every iteration is completed. In every iteration of both algorithms, all ants will make one step (move from one city to another) and then pheromone trail will be updated on all visited arcs according to the following rules:

Ant-density:

$$\Delta\tau_{ij}^{k}(s+1)=\begin{cases}\dfrac{Q}{d_{ij}} & \text{if } k-\text{th ant goes from } i \text{ to } j \text{ in iteration } s\\ 0 & \text{otherwise}\end{cases} \tag{4.7}$$

Ant-quantity:

$$\Delta\tau_{ij}^{k}(s+1)=\begin{cases}Q & \text{if } k-\text{th ant goes from } i \text{ to } j \text{ in iteration } s\\ 0 & \text{otherwise}\end{cases} \tag{4.8}$$

4.3 Similarities and differences between real and artificial ants

It has been mentioned earlier that in order to build AS, authors have used mostly ideas from real ants behavior. However, some differences exist. Similarities could be stated as follows (Dorigo et al., 1999):

- *Colony of cooperating individuals.* Both real ant colonies and AS contain a population, or colony, of independent individuals – agents. They globally cooperate in order to find a "good solution" to the task under consideration. In both cases one agent is able to find "feasible solution" independently of others. However, a region of good solutions could be discovered as a result of the cooperation among agents.
- *Stigmergy through pheromone trail.* Like real ants, artificial ants, when "walking", change some aspects of their environment. While walking, real ants

deposit a chemical substance (*pheromone*) on the visited state. Artificial ants will change some numerical information of the problem state, locally stored, when that state is visited. Based on analogy, these information and changes could be called an *artificial pheromone trail*. Ant System algorithms presume that a local pheromone trail is the single way of communication among artificial ants. In reality, pheromone evaporation exists. AS algorithms also include artificial pheromone evaporation in the form of reduction in the artificial pheromone trail over time. Pheromone evaporation in nature and in AS algorithms are important because it will allow ant colony to slowly forget the history and direct the searching process in new directions. Artificial pheromone evaporation could be helpful to move the searching process towards new regions and to avoid stacking in local extremes.

- *Local moves and the shortest path searching*. In contrast with real ants that are walking through adjacent terrain's states, artificial ants are "jumping" from one to another "adjacent state" of the considered problem. The definition of states and their adjacency is problem specific, and should be different for different problems. Either walking or jumping these steps have the same purpose, which is finding the shortest (minimum cost) path between the origin and the destination.
- *Transition policy*. Both real ants and artificial ones will build solutions by applying decision making procedures to move through adjacent states. Decision making procedures could be based on some probabilistic rules (introduced by Colorni et al. (1991, 1992), and Dorigo et al. (1996)) or probabilities could be calculated based on approximate reasoning rules. In both cases, the transition policy will use local information only – it should be local in the space and time sense. The transition policy is a function of local – state information represented by problem specifications (this could be equivalent to the terrain's structure that surrounds the real ants) and the local modification of the environment (existing pheromone trails) introduced by ants that have visited the same location.

Main differences could be stated as follows:

- Artificial ants live in a discrete world. All their moves are jumps from one discrete state to another.

- Artificial ants have memory; they could remember states that have been visited already (tabu lists in the model).
- Pheromone deposit methodology is significantly different between real and artificial ants. Timing in pheromone laying is problem dependent and often does not have similarities with the real ants pheromone deposit methodology.
- Amount of pheromone that an artificial ant will deposit is mostly a function of the quality of the discovered solution. In reality, some ants behave in a similar way; the deposited amount of pheromone is highly dependent on the quality of the discovered food source.
- To improve overall performances, AS algorithms could be enriched with some additional capabilities that cannot be found in real ant colonies. Mostly AS contains some local optimization technique to improve solutions developed by ants.

4.4 Similarities and differences between artificial ants and artificial bees

Similarities could be stated as follows:

- Both approaches present artificial colonies of cooperating individuals.
- Basic structure is taken from natural swarms. In both cases the system was developed based on foraging behavior of the corresponding natural swarms.
- General behavior contains individual decisions based on simplified rules taken from nature and collaboration among individuals through exchanging of collected information.
- Artificial ants and artificial bees live in a discrete world.

Main differences could be stated as follows:

- An exchange of information among agents occurs at the end of iteration (Bee System) or at the end of cycle (Ant System).
- Type of the approach to the problem could be simultaneous (Bee System) or sequential (Ant System).

- Amount and type of information exchanged through one exchange of information trial. Agents will receive just local information (Ant system) or information about any relatively promising food source in the covered region.

4.5 Other AS approaches

4.5.1 Ant Colony System

It has been shown experimentally that the previously described AS algorithm is capable of producing good solutions within a reasonable amount of time for only small problem instances. In order to improve capabilities of AS, Gambardella and Dorigo (1996), Dorigo and Gambardella (1997a, 1997b) introduce the Ant Colony System (ACS).

ACS was developed based on the Ant – cycle algorithm. A description of dissimilarity will be presented on the same example (TSP) as it was for the AS introduction. Main differences could be stated as follows:
Change in transition rule:
Ant k that is located in city i (cycle t and iteration s) will make choice about city j (where to move) based on the following rule:

$$j = \begin{cases} \arg\max_{u \in J_i^k(t,s)} \left\{ p_{iu}^k(t,s) \right\}, & \text{if } q \le q_0 \\ J, & \text{if } q > q_0 \end{cases} \tag{4.9}$$

where:

q_0 – tunable parameter ($q_0 \in [0, 1]$),

q – value taken from uniform distribution over [0, 1],

$p_{iu}^k(t,s)$ - probabilities calculated based on equation 4.3.

The basic idea is to allow ants to choose the "closest" city based on criteria $[\tau_{ij}(t)] \cdot [\eta_{ij}]^{\beta}$ and one random value q. In case $q > q_0$, city J that the ant will go to will be chosen based on the same procedure like in AS. In other words, a second random number will be taken

and based on it as well as probabilities $p_{ij}^{k}(t,s)$ the ant will make the choice of where to go among its allowed cities and for its current position.

Tuning parameter q_0 will take value from the [0, 1] interval. In cases where q_0 is close to 0, the search process will explore the feasible region widely. If q_0 is closed to 1, the search process will be concentrated on the best partial solution (greedy).

Change in pheromone updating rule:

In contrast to previous algorithms where pheromone is updated at the end of each cycle – the global pheromone update, ACS algorithm contains pheromone updates at the end of each cycle and pheromone updates at the end of each iteration – local pheromone update. To further explain how pheromone trails will be updated; let τ_{ij} (t, s) be the amount of pheromone that could be found on arc (i, j) at iteration s that belongs to cycle t.

- The global pheromone updating rule (will be provided offline like in AS);
  At the end of every cycle t pheromone will be updated according the following equation:

$$\tau_{ij}(t+1, 1) = \rho\, \tau_{ij}(t, n) + (1 - \rho)\, \Delta\tau_{ij}(t, t+1) \tag{4.10}$$

$$\Delta\tau_{ij}(t, t+1) = \begin{cases} \dfrac{1}{L^{+}}, & \text{if } (i, j) \text{ belongs to } T^{+} \\ 0, & \text{otherwise} \end{cases} \tag{4.11}$$

  Only arcs that belong to the best tour T^{+} (since beginning of search process) would have a pheromone update at the global level. The strategy will increase the importance of searching around the best route ever found.

- Local updates of the pheromone trail (will go on-line – after each iteration):
  When ant k located in city i selects city $j \in J^{k}_{i}(t, s)$, the pheromone concentration on arc (i, j) will be changed according to:

$$\tau_{ij}(t, s+1) = \varphi\, \tau_{ij}(t, s) + (1 - \varphi)\, \tau_0, \qquad s < n \tag{4.12}$$

$$\tau_{ij}(t+1, 1) = \varphi\, \tau_{ij}(t, n) + (1 - \varphi)\, \tau_0, \qquad s = n \tag{4.13}$$

where:

φ - value (taken from interval (0, 1]),

τ_0 – initial pheromone trail; it was experimentally found that the best results are obtained when τ_0 is calculated according the following equation:

$$\tau_0 = \frac{1}{n\, L_{nn}} \tag{4.14}$$

n – number of cities,

L_{nn} – tour length produced by the nearest neighborhood heuristic.

Local pheromone updating is necessary to avoid stagnation of the searching process. Without local pheromone updating, the searching process could be stacked because of the global pheromone update. Additionally, the local update rule should decrease attraction of a recently visited arc. The idea is to make the visited arcs less and less attractive as ants visit them. In that way the ants' route will disperse – a common path would not be formed.

"Candidate list" data structure. A candidate list could be defined as a list of preferred cities to be visited from the given city. In order to avoid examination of all unvisited neighbors of the current city (which for large instances could be time consuming) the ant would choose to examine only those cities in the candidate list. The candidate list contains a certain number *f* (given in advance) of neighboring cities in ascending order of distances. If the ant has visited all cities in the candidate list, other neighboring cities could be considered. If the candidate list is not empty then the next city that ant will visit will be chosen according to the formulas (4.9 and 4.3) otherwise, the ant will go to the first closest city from the rest of the unvisited cities.

4.5.2 Ant-Q

Gambardella and Dorigo (1995) Dorigo and Gambardella (1996a) developed the Ant–Q system. The system is similar to ACS and in fact ACS was developed based on an experimental study related to the Ant–Q system. The main difference is in the local

pheromone updating rule that could be represented as follows:

$$\tau_{ij}(t, s+1) = \varphi\, \tau_{ij}(t, s) + (1 - \varphi)\, \gamma \max_{u \in J_i^k(t,s)} \tau_{iu}, \qquad s < n \tag{4.15}$$

$$\tau_{ij}(t+1, 1) = \varphi\, \tau_{ij}(t, n) + (1 - \varphi)\, \gamma \max_{u \in J_i^k(t,s)} \tau_{iu}, \qquad s = n \tag{4.16}$$

Difference exists in the last term of the summation. Through experimental study it was discovered that setting the last term in the summation to a small constant value would result in the reduction of time required to solve the problem without significant performance deterioration of the algorithm. The next step in Ant based algorithm development was practically the introduction of ACS.

4.5.3 Concept of "Elitist ants":

The elitist ant concept is almost the same as the Ant-cycle approach. Dorigo et al. (1996) have introduced the concept by adding one element into summation (in the equation 4.4).

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij}(t, t+1) + e\, \Delta\tau^{e}_{ij} \tag{4.17}$$

where:

e – constant (given in advance),

$$\Delta\tau_{ij}^{e} = \begin{cases} \dfrac{Q}{L^{+}} & \text{if arc } (i, j) \text{ in its tour } T^{+} \\ 0 & \text{otherwise} \end{cases} \tag{4.18}$$

The main idea is to increase the attraction of arcs that belong to the best tour.

4.5.4 Max-Min AS

Stützle and Hoos (1997) introduced the Max-Min AS developed based on the concept of elitist ants in an AS algorithm with the following differences:

- The pheromone trail could be updated only at the end of every cycle for arcs that belong to the best route discovered in the cycle.

- Pheromone trail values associated to each arc are restricted to an interval [τ_{min}, τ_{max}].
- At the very beginning of the search process, pheromone trails should be initialized to the maximum pheromone trail value (τ_{max}).

Restriction of pheromone intensity to a predefined interval will cause all the probabilities of choosing arcs (according to the equation 4.3) to also be restricted. This will help in avoiding possible stagnation, which was the main problem in the implementation of the elitist ants concept.

4.5.5 AS_{rank} algorithm

Bullnheimer et al. (1999) proposed one modification of AS called the AS_{rank} algorithm. As it was proposed in AS, the pheromone trail will be updated at the end of every cycle. The main difference is the fact that in the proposed algorithm, developed solutions would be ranked in a descending order based on their quality at the end of each cycle, and then just the arcs that belong to the first ω - 1 (given in advance) solutions will receive an amount of pheromone proportional to the solution rank. Additionally, arcs that belong to the best tour ever found will receive an extra amount of pheromone (equivalent to the elitist ants strategy). Pheromone updates will be provided according the following equation:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \omega\,\Delta\tau_{ij}(t, t+1) + \Delta\tau^{r}_{ij}(t, t+1) \qquad (4.19)$$

where:

$\Delta\tau_{ij}(t, t+1)$ – additional pheromone on the best route ever found (calculation based on equation 4.11),

$\Delta\tau^{r}_{ij}(t, t+1)$ – additional pheromone that each arc will receive because it belongs to one or more tours in first ω - 1 tours (ranked on the bases of their length).

$$\Delta\tau^{r}_{ij}(t, t+1) = \sum_{\chi=1}^{\omega-1} \Delta\tau^{\chi}_{ij}(t) \qquad (4.20)$$

$$\Delta\tau_{ij}^{\chi}(t)=\begin{cases}\dfrac{(\omega-\chi)}{L^{\chi}(t)}, & \text{if the ant with rank } \chi \text{ has used arc } (i,j) \text{ in its tour}\\ 0, & \text{otherwise}\end{cases} \tag{4.21}$$

Where $L^{\chi}(t)$ is the χ ranked tour length.

In the literature, it could be found that this improvement has made a significant contribution to the quality of solutions produced by AS.

4.6 Parallel AS

Many models that have been developed with the purpose of defining parallel algorithms for other population-based algorithms could be used for building parallel AS. Parallelization of the metaheuristics algorithms is desirable when attempting to solve problems huge in dimensionality. There are several papers describing the various possibilities of implementation of parallel AS algorithms to solve large combinatorial optimization problems.

The time taken for the ants to form feasible solutions to the considered problem (the time spent within a cycle) is the major component in the total time needed by AS based algorithms to be completed. In each cycle, each ant will form a solution through a certain number of steps (iterations), making decisions independently of other ants' decisions. Precisely, those decisions are independent of decisions that other ants are making just in the current cycle. That is the reason why the first natural method of parallelization of the AS based algorithms could be provided in the way of subdividing an ant colony (set of ants) into η subsets. Every subset could perform a search process within a cycle on one different processor. After completion of the search process in the cycle, produced solutions from all processors involved will be sent to the place where pheromone updating will be performed. That will change information about the pheromone trails on all links and then a new cycle can start. Solutions on all processors are built based on the same pheromone trails on links (all subsets of ants will build solutions based on the same history information).

Another approach is to allow an ant colony to change information (through pheromone

updating) occasionally. In cases where parallel AS is applied, on large instances of some problems there is a possibility that after every cycle ants exchange information only within the subset. After a predefined number of cycles, information about discovered solutions will be exchanged between subsets of ants.

Both approaches could be found in Bullnheimer et al. (1997) with the names Synchronous Parallel Implementation of AS for the first approach and Partially Asynchronous Parallel Implementation for the second one. Authors suggest that for the second approach ratio local /global information exchange should be five to one.

Stützle (1998) presents one simple approach to parallelization of AS through the execution of parallel independent runs. This approach is reasonable only in cases where the considered algorithm is randomized – decisions in the search process contain randomness. This approach is suitable when the solution to the problem should be found in a short period of time. It has been shown experimentally that the highest chance of finding an optimal solution has multiple runs of the algorithm as opposed to single runs in cases where all independent runs have the same amount of time as a single run. Surely, for every independent run it is necessary to have more time than the certain initial amount (t_{init}). This initial amount of time (t_{init}) could be defined for any metaheuristic algorithm and any considered problem instance. It is necessary to have the search process at least a little longer than t_{init} to have an acceptable chance for producing a solution that is good enough.

In many cases, metaheuristic algorithms are supported with some local search technique to produce better solutions and make the search time shorter. The amount of time that the local search technique needs to improve the current solution highly depends on heuristic's complexity and the size of the considered instance. One possible approach to parallelization is separation of the searching process among processors in such a way that the local search technique will be provided on one processor and the entire algorithm on the other one.

4.7 Fuzzy Ant System

The modifications in the development of later AS have primarily been in modeling the

methods of communication among ants. Through many publications, the methodology of how ants make the choice where to go in the next iteration (transition rule) has stayed practically unmodified for an entire decade. AS shows good performance in solving problems that are combinatorial in nature. However, some of the real-life problems are simultaneously characterized by uncertainty and by combinatorial nature. It could be seen that combinatorial problems characterized by uncertainty are not covered by any of the modifications of AS that are found in literature.

In this part of the article, research is focused on the attempt to modify the "classical" Ant system by proposing a new one – *The Fuzzy Ant System* (*FAS*). The basic modification would be in the way of calculating transition probabilities, where fuzzy logic is used. It is possible to deal with the uncertainty that could exist in complex combinatorial optimization problems by using fuzzy logic as a separate module within the Ant system. The control strategies of the ant can also be formulated in terms of numerous descriptive rules.

When making a decision about next node to be visited (in the case of the Traveling salesman problem), the ant takes both "visibility" and pheromone trail intensity, into the consideration. It is possible to assume that an ant can perceive a particular distance between nodes as fuzzy values (example: *"small"*, *"medium"* or *"large"*), and the trail intensity as fuzzy values such as *"weak"*, *"medium"* or *"strong"*.

When choosing the next link, the ant will have greater or lesser perceived utility towards it, depending on the distance from the next node, as well as the trail intensity. These utilities can be described by appropriate fuzzy sets.

The approximate reasoning algorithm for calculating the utility of choosing the next link could consist of the rules of the following type:

If distance is **SMALL** and trail intensity is **STRONG**
Then utility is **VERY HIGH**

The result of the application of approximate reasoning algorithms is finally a crisp number. In this example it should be the perceived ant's utility to go to a certain node. Based on all obtained utilities it is possible to calculate probabilities associated with each node where an ant could move from its current node position, taking into account all

nodes that the ant has visited as well as the arcs that are connecting the current node with others. At the end, final node choice would be obtained in a random manner as in "classic" AS. That means, the approximate reasoning algorithm could replace the original relation for calculating transition probabilities (4.3). Therefore, it is possible to calculate transition probabilities even if some of the input data were only approximately known. It seems that the approaches that have combined the Ant System and Fuzzy Logic (Fuzzy Ant System) could be a very powerful tool in solving different problems that are simultaneously characterized by combinatorial nature and uncertainty.

This approach will be illustrated with the following two examples:

- *Vehicle routing problem when demand at nodes is uncertain*. Demand at the nodes is treated as a fuzzy number and the actual demand value is known only after the visit to the node. Vehicle routing problems with all modifications are well known in literature and one new approach to the problem will be shown here.
- *Schedule synchronization in public transit*. Trips between two nodes in a public transit network may be made with or without making transfers. Transfers usually represent inconvenience to the passengers. Since badly coordinated transfers can significantly increase waiting times, it is especially important to carefully synchronize schedules in the cases of bigger headways. At the same time, badly coordinated transfers can decrease the total number of passengers in public transit and result in their switching to the competitive modes. When synchronizing schedules, it is necessary to try to minimize the total waiting times of all passengers at transfer nodes in a transit network. Very often only approximate numbers of transfer passengers are known. The model for schedule synchronization that is applicable while constructing timetables will be presented. The basic assumption is that the number of transfer passengers is only approximately known. The model is based on the Ant System and Fuzzy Logic (Fuzzy Ant System).

4.8 The combined Ant System - Fuzzy Logic approach to the vehicle routing problem when demand at nodes is uncertain

4.8.1 Introduction

The Stochastic Vehicle Routing problem has been introduced in chapter 3.4. In a high percentage of papers devoted to the vehicle routing problem, uncertain demand at nodes was treated as a random variable. The exception is the model proposed by Teodorović and Pavković (1996) in which appropriate fuzzy numbers represent demand at the nodes.
In this section, we will describe a new algorithm for handling the vehicle routing problem when there is uncertain demand at nodes. In the problem considered, as before, locations of the depot, location of the nodes to be served and vehicle capacity are known. Demands at the nodes are represented using triangular fuzzy numbers. The goal is to produce a set of "high quality routes" for the vehicles in advance (planning purpose).
The entire approach presentation is organized in the following way. Statement of the problem is given in section 4.8.2. The proposed solution to the problem is given in section 4.8.3. Numerical experiments are shown in section 4.8.4.

4.8.2 Statement of the problem

Let us assume that there are n nodes to be serviced (Figure 3.14). It is also assumed that vehicles of uniform size provide the service. We will denote vehicle capacity by C. Vehicles set out from depot D serve a number of nodes and after completing their service, return to the depot. Demand at each node is only approximately known and presented by an appropriate fuzzy number.

Let us assume that demand $\mathbf{D}_j$ at any node j, can be represented by the triangular fuzzy number, i.e.:

$$\mathbf{D}_j = (d_{1j}, d_{2j}, d_{3j}) \tag{4.22}$$

where d_{1j} is the left boundary of fuzzy number $\mathbf{D}_j$, d_{2j} is the value of fuzzy number $\mathbf{D}_j$ corresponding to a grade of membership of 1, and d_{3j} is the right boundary of fuzzy number $\mathbf{D}_j$.

As in previous example, "route failure" can occur due to uncertainty that characterizes demand at nodes. When evaluating the planned route, the additional distance that the vehicle travels due to the possible "route failure" at any node along the route, must be taken into consideration. The problem is to design such a set of vehicle routes that will result in the least total sum of planned route lengths and additional distance covered by vehicles due to "route failure".

4.8.3 Proposed solution to the problem

Let us denote the total number of ants by *m* and let us locate all of them in the depot. We will also assume that instead of vehicles, ants are servicing the nodes. Ants located in the depot of the considered transportation network must visit at least one node before coming back to the depot. Let us denote the capacity of every ant by *C*. In other words, every ant has the same capacity *C* and cannot have a load greater than *C* at any moment. The ants have the choice to pick up the next link. The number of choices is equal to the number of links incident to the current ant's position. At the beginning of the search process, we will assume that the pheromone trail is very low on every link and that is equal to some small positive constant.

Let us consider *k*-th ant located in node *i* at time *t*. Let us denote by $J_i^k(t)$ the set of nodes that ant *k* has not visited by the time *t*. Set of visited nodes, obviously, is $\overline{J_i^k(t)}$. This set could be subdivided into two subsets, one contains all nodes that are already included in routes that the ant has developed ($\overline{J_i^{'k}(t)}$) and another contains nodes that the ant includes in its currently building route ($\overline{J_i^{"k}(t)}$). After serving nodes from the set $\overline{J_i^{"k}(t)}$, the available ant's capacity $A_{k,\overline{J_i^{"k}(t)}}$ will equal:

$$A_{k,\overline{J_i^{"k}(t)}} = C - \sum_{j \in \overline{J_i^{"k}(t)}} D_j \tag{4.23}$$

Demand at every node $\mathbf{D_j} = (d_{1j}, d_{2j}, d_{3j})$ is represented by a corresponding triangular fuzzy number. Using fuzzy arithmetic rules, we obtain that the quantities $\sum_{j \in \overline{J_i^{"k}(t)}} D_j$ and

$(C - \sum_{j \in \overline{J_i^{"k}(t)}} D_j)$ are also triangular fuzzy numbers, i.e.:

$$\sum_{j \in \overline{J_i^{"k}(t)}} D_j = (\sum_{j \in \overline{J_i^{"k}(t)}} d_{1j}, \sum_{j \in \overline{J_i^{"k}(t)}} d_{2j}, \sum_{j \in \overline{J_i^{"k}(t)}} d_{3j}) \tag{4.24}$$

$$C - \sum_{j \in \overline{J_i^{"k}(t)}} D_j = (C, C, C) \text{ (-) } (\sum_{j \in \overline{J_i^{"k}(t)}} d_{1j}, \sum_{j \in \overline{J_i^{"k}(t)}} d_{2j}, \sum_{j \in \overline{J_i^{"k}(t)}} d_{3j}) =$$

$$(C - \sum_{j \in \overline{J_i^{"k}(t)}} d_{1j}, C - \sum_{j \in \overline{J_i^{"k}(t)}} d_{2j}, C - \sum_{j \in \overline{J_i^{"k}(t)}} d_{3j}) \tag{4.25}$$

The available capacity $A_{k, \overline{J_i^{"k}(t)}}$ is also triangular fuzzy number:

$$A_{k, \overline{J_i^{"k}(t)}} = (a_{1, k, \overline{J_i^{"k}(t)}}, a_{2, k, \overline{J_i^{"k}(t)}}, a_{3, k, \overline{J_i^{"k}(t)}}) = (C - \sum_{j \in \overline{J_i^{"k}(t)}} d_{1j}, C - \sum_{j \in \overline{J_i^{"k}(t)}} d_{2j}, C - \sum_{j \in \overline{J_i^{"k}(t)}} d_{3j}) \tag{4.26}$$

Membership functions of the fuzzy set $A_{k, \overline{J_i^{"k}(t)}}$ representing available capacity, and fuzzy set $\geq A_{k, \overline{J_i^{"k}(t)}}$ whose name is "demand greater than available capacity" are shown in figure 4.4.

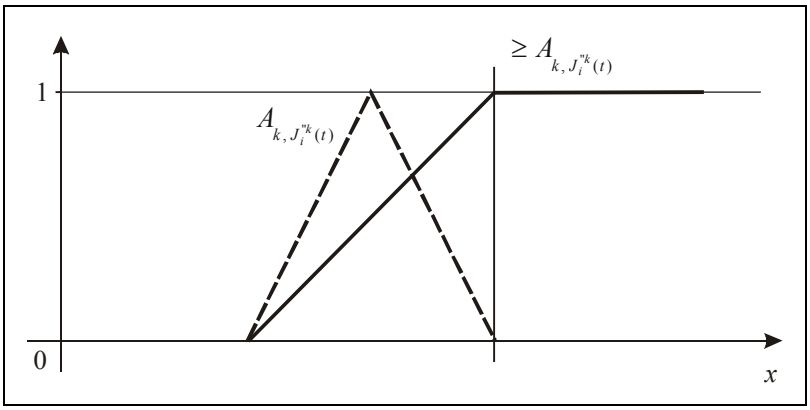

Figure 4.4 – Membership functions of fuzzy sets $A_{k, \overline{J_i^{"k}(t)}}$ and $\geq A_{k, \overline{J_i^{"k}(t)}}$

"In the theory of fuzzy subsets the law of possibility plays a role similar to that played by the law of probability in measurement theory for ordinary sets" (Kaufmann and Gupta, 1985).

If

$$\forall x \in \mathrm{R}: \quad h(x) \in [0,1] \qquad \text{and} \qquad \bigvee_{x \in \mathrm{R}} h(x) = 1 \tag{4.27}$$

then $h(x)$ is called the possibility law on R. If **A** is a fuzzy subset of R, then the possibility of **A** for the law $h(x)$ is defined as

$$\mathrm{poss}_{\mathrm{H}}\mathbf{A} = \bigvee_{x \in \mathrm{R}} \left(\mu_{\mathbf{A}}(x) \wedge h(x) \right) \tag{4.28}$$

where $\mu_{\mathbf{A}}(x)$ is a membership function of fuzzy set **A**, $\vee$ is maximum symbol, and $\wedge$ is minimum symbol.

The $\mathrm{poss}_{\mathrm{H}}\mathbf{A}$ presents the possibility of **A** (left) and $\mathrm{poss}_{\mathrm{H}}\mathbf{B}$ presents the possibility of **B** (right) for the law $h(x)$, as shown in figure 4.5.

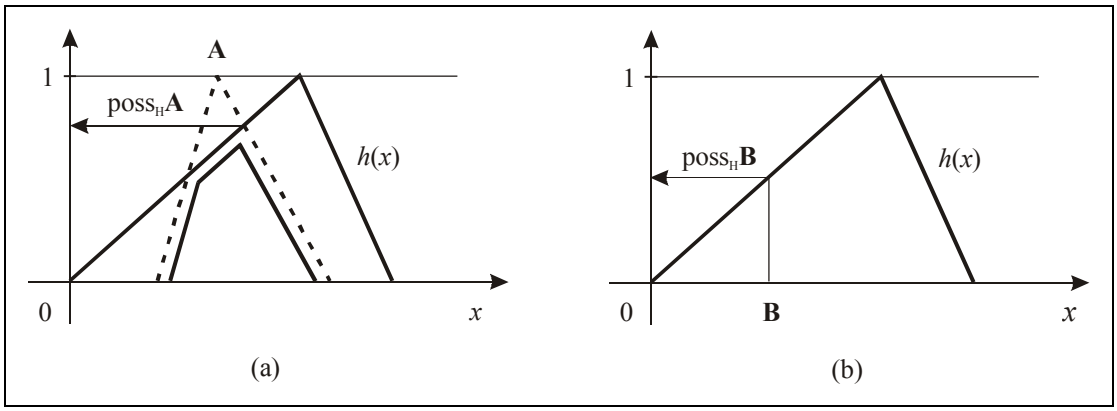

Figure 4.5 – The possibility of **A** and the possibility of **B** for the law $h(x)$

Let possibility $h(x)$ refer to "demand greater than available capacity". In figure 4.6, we denoted the possibility that demand in the next node to be visited $\mathbf{D_j}$ is equal to the "demand greater than available capacity". This possibility practically represents the possibility that the "route failure" will occur in the next node.

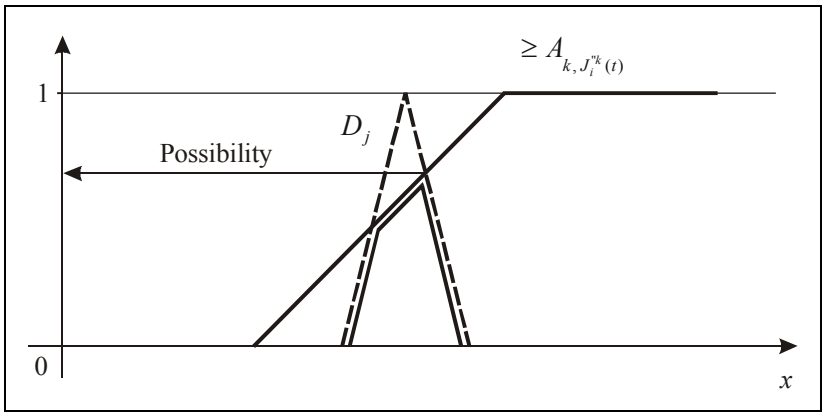

Figure 4.6 – The possibility that demand (D_j) in the next node to be visited is equal to the "demand greater than available capacity" ($\geq A_{k,\overline{J_i^{"k}(t)}}$).

It is clear that the "strength" of the ant's preference to serve some node after it has served $\left|\overline{J_i^{"k}(t)}\right|$ nodes depends on the available capacity $A_{k,\overline{J_i^{"k}(t)}}$, as well as on expected demand in the considered node ($\mathbf{D_j}$). The bigger the available capacity and the smaller the expected demand in the potential node to be visited, the higher the ant's expectation that it can serve the next node without the failure and the higher the ant's wish to go to that node. It is assumed that the artificial ants use the approximate reasoning to reach the conclusion about the next node to be visited. The ant's perceived utility of visiting the next node could be, for example, "low", "medium" or "high". Let u_j be the ant's utility of visiting node j (j being the ($\left|\overline{J_i^{"k}(t)}\right|$+1)-st node in the route), after it has already served $\left|\overline{J_i^{"k}(t)}\right|$ nodes.

Let the perceived utility index be between 0 and 1, that is,

$$0 \leq u_j \leq 1 \qquad j \in J_i^k(\mathrm{t}) \tag{4.29}$$

When u_j = 1, the ant is absolutely certain that he wants to serve j-th node. When u_j = 0, ant is completely sure that he should return to the depot or serve another node. The linguistic expressions "very very low utility," "very low utility," "low utility," "medium utility," "high utility," "very high utility" and "very very high utility" can be represented by corresponding fuzzy sets. The membership functions of fuzzy sets "very very low,"

"very low," "low," "medium," "high," "very high" and "very very high" ant's perceived utility are shown in figure 4.7.

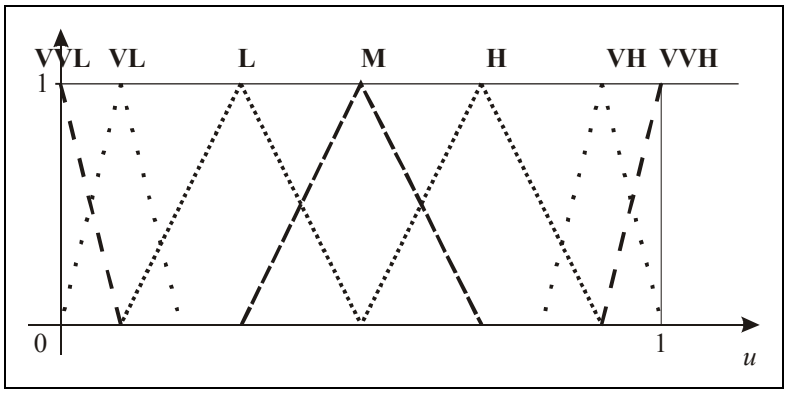

Figure 4.7 – Fuzzy sets describing ant's utility strength

The estimation of additional distance that ant will pass by introducing node *j* into route could be also described using fuzzy sets. The membership function of fuzzy sets "small," "medium" and "large" are shown in figure 4.8.

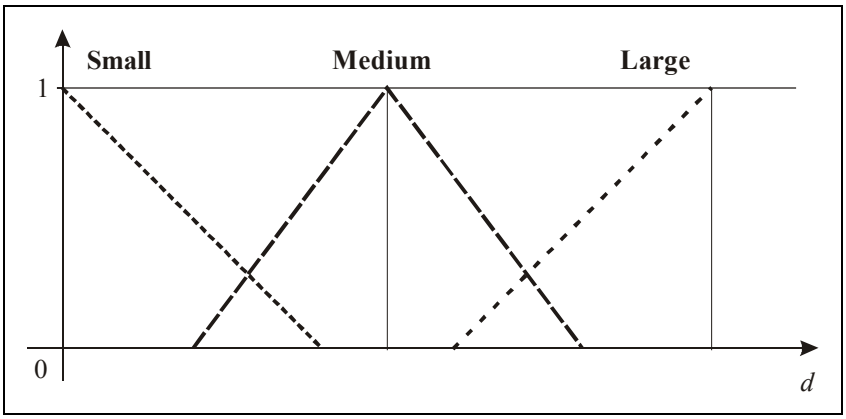

Figure 4.8 – Fuzzy sets describing expected additional distance

Furthermore, let us introduce fuzzy sets "weak," "medium" and "strong" denoting weak, medium and strong feelings by the ant about pheromone trail intensity (Figure 4.9).

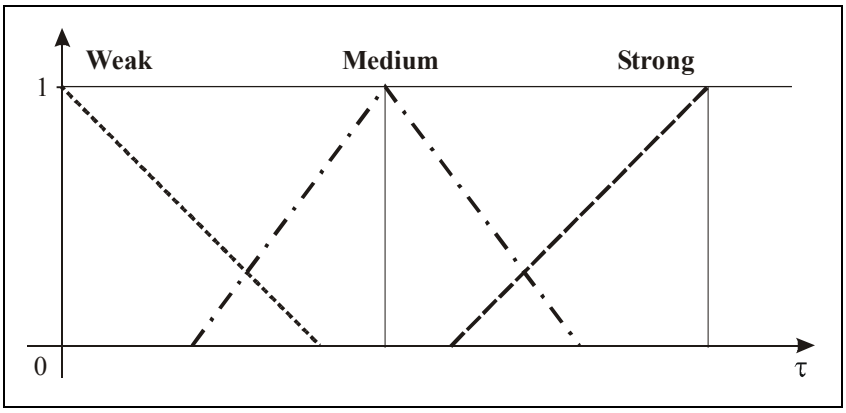

Figure 4.9 – Fuzzy sets describing pheromone trail intensity

Let us assume that at the time point *t*, ant *k* is located at node *i*. It is already mentioned that $J^k_i(t)$ denotes the set of nodes that ant *k* has not visited by the time *t* (the set of unvisited nodes). In the cases of bigger instances of this problem (especially in the beginning of search process) the number of nodes $|J^k_i(t)|$ in the set of unvisited nodes $J^k_i(t)$ could be large. We have assumed that our artificial ant will consider only *q* nodes from set $J^k_i(t)$ that are closest to node *i* as a potential candidates for visit. Let us denote this subset of *q* closest node to node *i*, by ${}_qJ^k_i(t)$. In this way, the number of potential nodes to be visited is decreased. Knowing available capacity and demand at every node $j \in {}_qJ^k_i(t)$ and available vehicle capacity, we can calculate the possibility $\lambda^k_{ij}(t)$ of route failure in node *j* if artificial ant *k*, being located at node *i*, decides to visit node *j*.

If $\lambda^k_{ij}(t) > \lambda$, ant *k* will exclude node *j* from future consideration. This will further decrease the number of potential nodes to be visited. The set ${}_qJ'^k_i(t)$ contains all nodes *j* that satisfy the following relation: $\lambda^k_{ij}(t) \leq \lambda$. The value of parameter λ is given in advance. Let us assume that, after the exclusion of a certain number of nodes from further consideration, set ${}_qJ'^k_i(t)$ is not empty, i.e. $|{}_qJ'^k_i(t)| > 0$. This means that there are still some unvisited nodes that are relatively "good" candidates to be visited by the ant. Here, it will be assumed that the ant has the following two options regarding next node to be visited: (a) The ant will visit the closest unvisited node; (b) Artificial ants use the approximate reasoning to reach the conclusion about the next node to be visited. Random number $r \in [0,1]$ is chosen. In the case when $r < r^*$, ant will go to the closest node. When $r \geq r^*$, ant will choose approximate reasoning algorithm to reach the conclusion

about next node to be visited. Value r^* is constant given in advance. Let us explain the input and output variables that participate in this approximate reasoning algorithm. Let us denote the following variables by d^k_{ij}, τ_{ij} and u^k_{ij} respectively:

d^k_{ij} - the "expected" distance that the *k*-th ant will travel if he decides to go from node *i* to node *j*;
τ_{ij} - the pheromone trail intensity that the *k*-th ant can smell when traveling between node *i* and node *j*;
u^k_{ij} - utility of *k*-th ant being located in node *i* to visit node *j*;

Due to the uncertainty of demand at the nodes, an ant might not be able to service a node when it arrives there because of insufficient capacity. In such situations, the ant returns to the depot, empties what it has picked up thus far, returns to the node where it had a "failure," and continues service along the planed route. This means that the "expected" distance d^k_{ij} could be calculated as:

$$d^k_{ij} = d_{ij} + 2d_{D,j} \cdot \lambda^k_{ij}(t) \tag{4.30}$$

where:

$d_{D,j}$ - distance between depot D and node *j*

$\lambda^k_{ij}(t)$ – as we have mentioned, possibility that the demand in node *j*, $\mathbf{D_j}$ is greater than the available capacity $A_{k, \overline{J_i^{\pi k}(t)}}$.

Usually, an ant's visibility is expressed as inverse of the distance between the current and considered ant's position. The better the visibility (the smaller the distance), the higher the chances are that the ant will visit the considered node. In our case, when calculating the distance between the current and potential (new) ant's location, it was necessary to take into account the potential additional distance that the ant could travel due to insufficient capacity. The *k*-th ant is located in node *i*. The variable d^k_{ij} measures the "goodness" of visiting the *j*-th node by the *k*-th ant, taking into account the available capacity, as well as the demand characteristics in the *j*-th node. The lower the value of the variable d^k_{ij}, the better for the ant to go to the *j*-th node. The variable τ_{ij} represents the pheromone trail intensity that the *k*-th ant can smell when traveling between node *i* and node *j*. The higher the value of the variable τ_{ij}, the better it is for the ant to go to the *j*-th

node. The typical rule in the approximate reasoning algorithm for determining the perceived utility of visiting the next node can be the following one:

If d^k_{ij} is **Small** and τ_{ij} is **Strong**
Then Ant's utility u^k_{ij} of visiting the *j*-th node is **Very High.**

We can see that the antecedent of the rules contains the expected travel distance and the pheromone trail intensity. The approximate reasoning algorithm for calculating the ant's perceived utility of visiting some of the neighboring nodes consists of the following 9 rules:

Rule 1:
If d^k_{ij} is **Small** and τ_{ij} is **Weak**
Then Ant's utility u^k_{ij} of visiting the j-th node is **High.**
else
Rule 2:
If d^k_{ij} is **Small** and τ_{ij} is **Medium**
Then Ant's utility u^k_{ij} of visiting the j-th node is **Very High.**
else
Rule 3:
If d^k_{ij} is **Small** and τ_{ij} is **Strong**
Then Ant's utility u^k_{ij} of visiting the j-th node is **Very Very High.**
else
Rule 4:
If d^k_{ij} is **Medium** and τ_{ij} is **Weak**
Then Ant's utility u^k_{ij} of visiting the j-th node is **Low.**
else
Rule 5:
If d^k_{ij} is **Medium** and τ_{ij} is **Medium**
Then Ant's utility u^k_{ij} of visiting the j-th node is **Medium.**
else
Rule 6:
If d^k_{ij} is **Medium** and τ_{ij} is **Strong**
Then Ant's utility u^k_{ij} of visiting the j-th node is **High.**
else
Rule 7:

If d^k_{ij} is **Large** and τ_{ij} is **Weak**
Then Ant's utility u^k_{ij} of visiting the j-th node is **Very Very Low.**
else
Rule 8:
If d^k_{ij} is **Large** and τ_{ij} is **Medium**
Then Ant's utility u^k_{ij} of visiting the j-th node is **Very Low.**
else
Rule 9:
If d^k_{ij} is **Large** and τ_{ij} is **Strong**
Then Ant's utility u^k_{ij} of visiting the j-th node is **Low.**

Graphical representation of the fuzzy rule base is shown in figure 4.10.

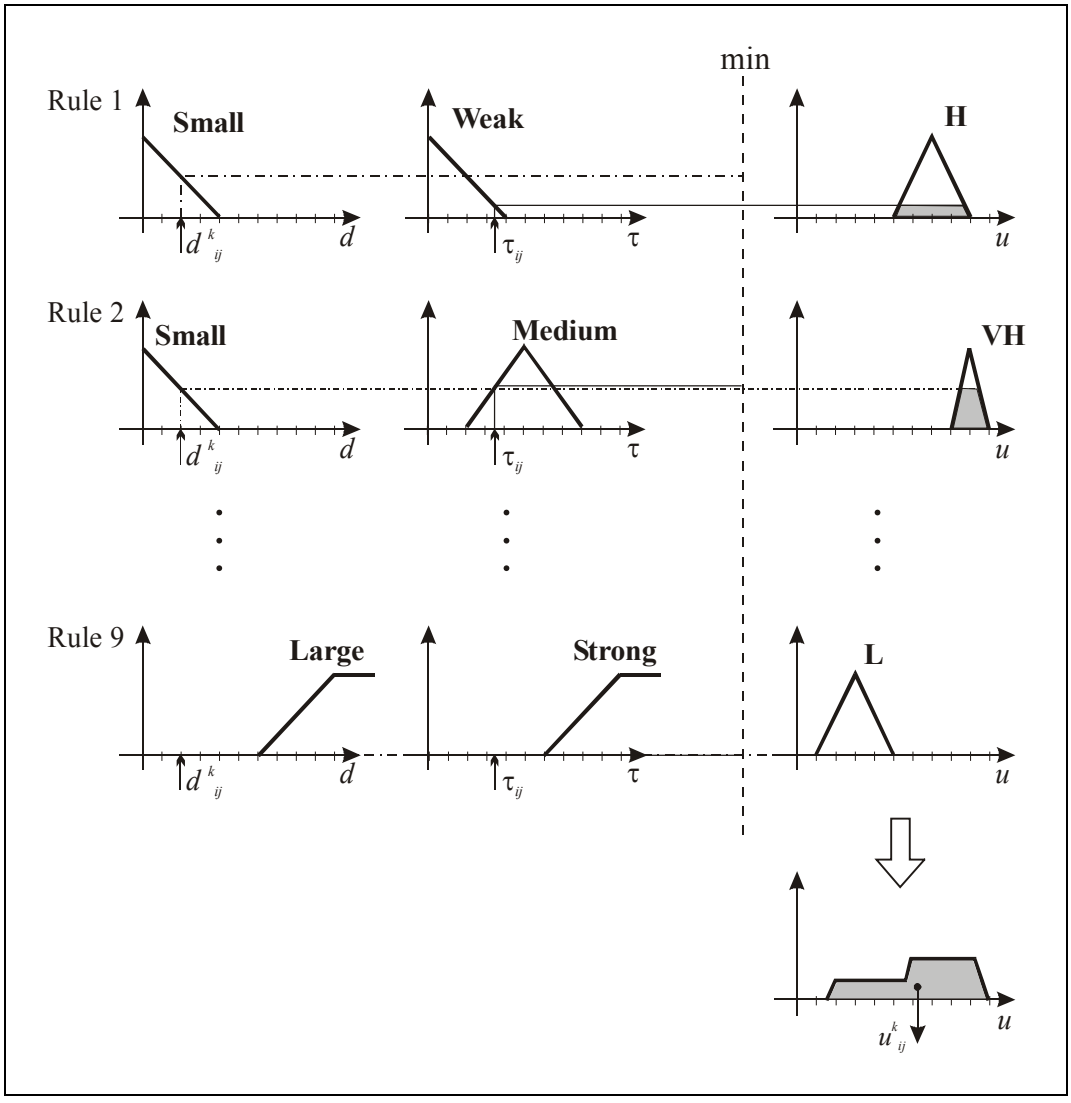

Figure 4.10 – Graphical representation of the fuzzy rule base

The nodes with better perceived utility values are more likely to be selected by the ant. The probability p^k_{ij} for node *j* to be selected by the ant *k* is equal to the ratio of u^k_{ij} to the sum of utility values of all nodes in the group of considered nodes:

$$p_{ij}^{k} = \frac{u_{ij}^{k}}{\sum_{h \in {}_{q}J_{i}^{"k}(t)} u_{ih}^{k}} \tag{4.31}$$

This type of selection represents a proportional selection, known as the "roulette wheel selection" in Evolutionary programming (The sections of roulette are in proportion to the probabilities p^k_{ij}).

In the situation where, $|\ {}_{q}J'^{k}_{i}(t)\ | = 0$ and $|\ J^{k}_{i}(t)\ | > 0$ the ant will go to the depot and start a new route. Since $|\ {}_{q}J'^{k}_{i}(t)\ | = 0$ the route failure possibility is very high in all unvisited nodes, so the ant decides to go back to the depot and start with a new route, because $|\ J^{k}_{i}(t)\ | > 0$, which means there are still nodes not visited by the ant *k*.

After returning to the depot, the same ant will continue to design the routes, taking care exclusive of only the unvisited nodes. In this way one ant can design one set of routes. Since there are *m* ants, the total of *m* sets of routes will be created. When all *m* ants create routes one cycle will be finished. As it is mentioned earlier artificial ants live in discrete time. After finishing the cycle, we will update the pheromone trails. We have determined the number of cycles in advance. After the assigned number of cycles is finished, the best set of routes is chosen.

The following strategy has been used for updating pheromone trail:

The best set of routes is chosen based on the value of performance index (*F*) of that set. Performance index (*F*) is the total sum of planned routes lengths and additional distance covered by vehicles due to route failure. For the pheromone updating rule equations 4.4, 4.5 and one modification of equation 4.6 were used. Additional rewarding was applied on the arcs that belong to the solutions that have performance indexes (*F*) that is the best ever found or γ percent higher than the best. The value of γ is given in advance and will be proportionally reduced as the cycle progresses. Expression that has been used for selection

of the solutions whose arcs will receive additional amount of pheromone is as follows:

$$F_k(t) \le F^+ \ \frac{t_{\max} - t + 1}{t_{\max}}(1+\gamma), \quad \text{for } k = 1, 2, ..., m. \tag{4.32}$$

where:

F^+ – the best performance index value ever found,
$t_{\max}$ – total number of cycles (given in advance),
t – current cycle number.

The equation 4.6 has been modified as follows:

$$\Delta\tau_{ij}^k(t) = \begin{cases} \dfrac{Q}{F_k(t)}(1 - \phi_k(t) + e\,\phi_k(t)), \text{ if } k - \text{th ant uses arc } (i, j) \text{ in its solution in the cycle } t \\ 0, \quad \text{otherwise} \end{cases} \tag{4.33}$$

where:

$$\phi_k(t) = \begin{cases} 1, \text{ if the inequality 4.32 is True} \\ 0, \text{ otherwise} \end{cases}$$

Q, e – constants given in advance.

This pheromone updating strategy is just one variation of the "Elitist ants" concept introduced in section 4.5.3. The main idea is to increase the attraction of arcs that belong to the certain number of "good" sets of tours.

At the end of each cycle, routes that ants involved in the search process have developed are improved by 2-opt heuristic independently.

The vehicle routes are created in the following way:

Step 1: Describe the demand at the nodes by corresponding triangular fuzzy numbers. Set the counter of the cycles to one (t = 1).

Step 2: If the number of finished cycles t is equal to the assigned number of cycles $t_{\max}$, go to step 4. Otherwise, go to Step 3.

Step 3: Set the counter of ants to one (k = 1). Locate all m ants in the depot. Generate m sets of routes by m ants. Generate routes using sequential approach (one ant at the time). When all nodes are visited, ant k will finish with the route design.

Increase the ant counter by one after creating one set of the routes. If the ant counter is equal to m+1, increase the cycle counter by one, apply 2-opt heuristic, update pheromone trails and go to step 2. In the opposite case allow the next ant to create the set of routes within considered cycle.

Step4: Take the final solution as a set of routes, such that they have the least total sum of planned route lengths and additional distance covered by vehicles due to failure. End the algorithm.

4.8.4 Results obtained using the Fuzzy Ant Vehicle Routing System

The developed model was tested on a large number of different numerical examples. In the first step the location of the depot and n (up to 150) nodes were generated randomly. The characteristics of the node demand (symmetric triangular fuzzy numbers $\mathbf{D_i}$) were also randomly generated for every created node i. When randomly generating the demand in every node the following relations are fulfilled:

$$0 \le d_{2i} \le \frac{C}{\rho}; \quad \frac{d_{3i} - d_{1i}}{2} \le \frac{d_{2i}}{3}; \quad i = 1, 2, ..., n. \tag{4.34}$$

where value ρ is integer (ρ >1) given in advance.

The "real" demand in every node was also generated randomly.

Demand at each node is a deterministic amount, obtained by simulation. By moving along the route, designed by the approximate reasoning algorithm, and accumulating the amounts picked up at each node, it was easy to determine the nodes where failures occurred and to calculate the additional distance that the vehicles had to make.

All computer experiments were done on a PC computer (PII 450 processor). CPU time highly depends about input data (problem size, number of cycles and number of ants) and for example with 70 node, 500 iterations and 15 ants it takes about 2 hours. Achieved CPU time is acceptable for planning purpose.

Input data and final solution for one 70 node numerical example are shown in table 4.1 and figure 4.11. In the example we assumed vehicle capacity of 1500 units. Node 0 is depot.

Table 4.1 – Input data (70 nodes example)

Node No.	Node Coordinates		Demand at nodes		
	x_i	y_i	d_{1i}	d_{2i}	d_{3i}
0	0	0	/	/	/
1	748	-397	136	140	144
2	669	988	26	39	52
3	709	-325	49	71	93
4	654	397	32	48	64
5	874	636	52	54	56
6	-211	281	80	81	82
7	811	-197	137	165	193
8	463	-615	77	111	145
9	-291	-491	64.6667	97	129.3333
10	867	432	98	144	190
11	73	399	30.6667	46	61.3333
12	292	-637	52	52	52
13	761	-534	169	191	213
14	151	369	12.6667	19	25.3333
15	720	-140	1.3333	2	2.6667
16	-600	937	198	198	198
17	55	-558	113	154	195
18	793	315	167	185	203
19	647	-250	34	51	68
20	211	194	81	97	113
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
68	437	-392	10	10	10
69	119	-514	106	152	198
70	-161	-590	105	122	139

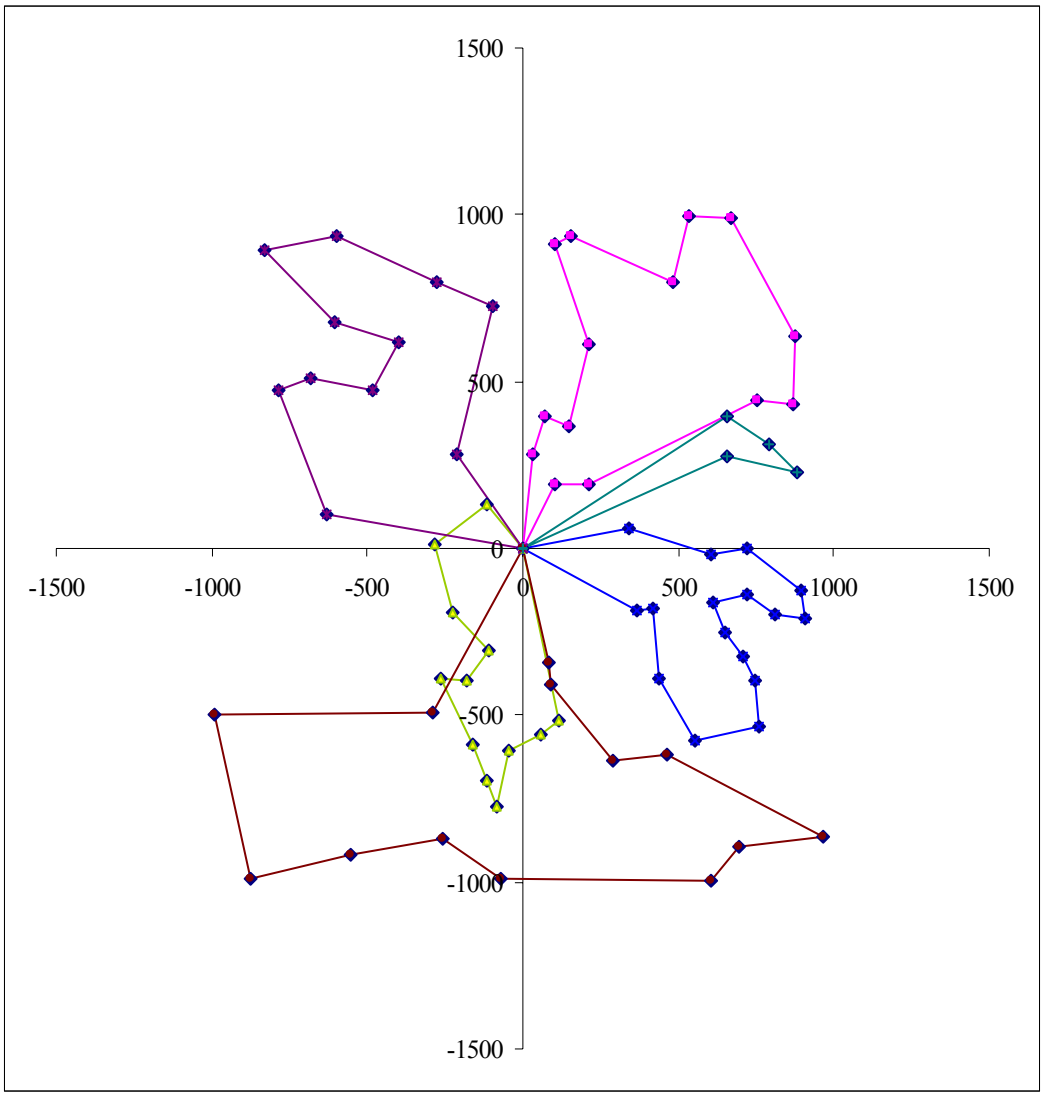

Figure 4.11 – Graphical representation of the best solution

The typical shape of the curve that represents the best performance index progression through cycles is shown on figure 4.12. This result was obtained for the same example (70 nodes).

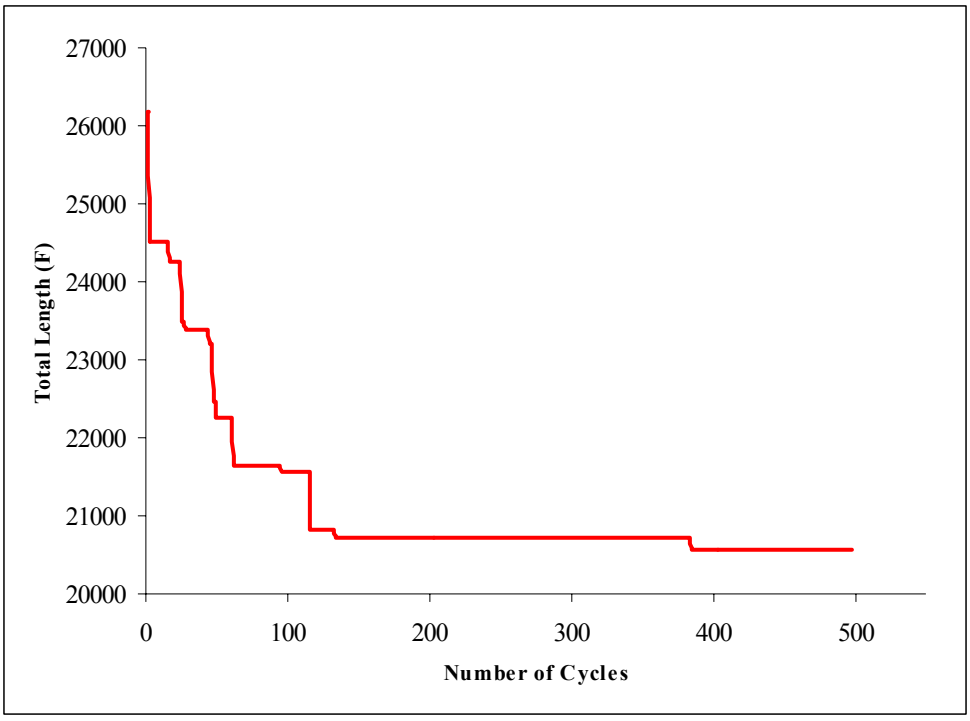

Figure 4.12 – The best criteria value progression through cycles

4.9 Schedule synchronization in public transit using Ant System and Fuzzy Logic

4.9.1 Introduction

Trips between two nodes in a public transit network may be made with or without making a transfer. A direct connection could be provided only to a certain number of passengers primarily due to the economic reasons. In addition, some passengers need to make trips by using different modes of transportation (bus, light rail, tram, metro, trolley bus, coordinated Dial-A-Ride). In other words, transfers are a necessity in public transit. At the same time, transfers represent inconvenience to the passengers. It is especially important to carefully synchronize schedules in the case of bigger headways, since badly coordinated transfers can significantly increase waiting times. Poorly coordinated transfers can also decrease the total number of passengers in public transit and encourage their switching to the competitive modes (private cars). The total number of transfers

when making a trip between any two nodes in the network is determined by the line alignments. The average waiting times while making transfers are a direct consequence of the schedule synchronization. While making schedule synchronization it is necessary to try to minimize the total waiting times of all passengers at transfer nodes in the transit network. The uncertainty surrounding the number of transfer passengers complicates the schedule synchronization. Very often only approximate numbers of transfer passengers are known. Sometimes it is very difficult and quite costly to collect relevant statistical data and to make an accurate prediction of the number of transfer passengers. In the case of a new or reconstructed transit network such data usually do not exist at all. In this section, the attempt to develop a model for schedule synchronization (while constructing timetables), when the number of transfer passengers is only approximately known, is presented. The presented model is based on the Ant System and Fuzzy Logic. This part of the document is organized in the following way. A statement of the problem is given in section 4.9.2. A proposed solution to the problem is described in section 4.9.3. The results obtained by using proposed model are given in section 4.9.4.

4.9.2 Statement of the problem

Timetabling and schedule synchronization are the planning phases that follow transit network design, detailed line alignment and determination of the frequencies and line headways.

The basic assumption is that the following input quantities and data are given:

(a) vehicle travel times between any two successive nodes along any transit line,

(b) vehicle stop times at any station,

(c) line headways,

(d) transfer times at any station, and

(e) the numbers of transfer passengers among particular transit lines are only approximately known.

An example of a public transit network is given in figure 4.13.

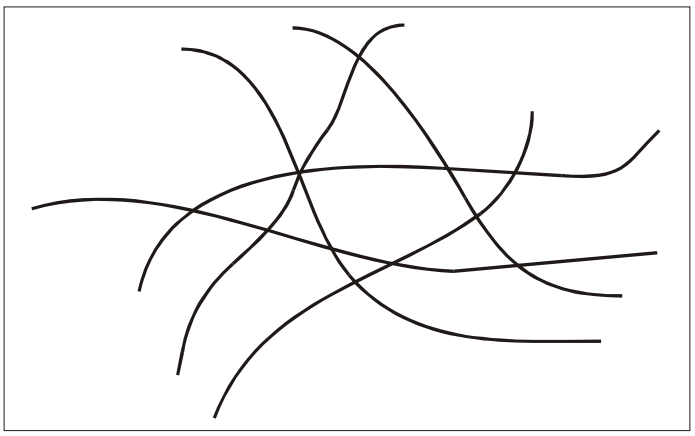

Figure 4.13 – Public transit network

Let us denote by m the total number of transit lines, by $L = \{l_1, l_2, l_3, \ldots, l_m\}$ the set of transit lines, and by H_i the headway along the i-th transit line ($i = 1, 2, 3,\ldots, m$). Let us also assume that the first departure from the first station of the transit line l_i can be performed only in certain time points (Figure 4.14).

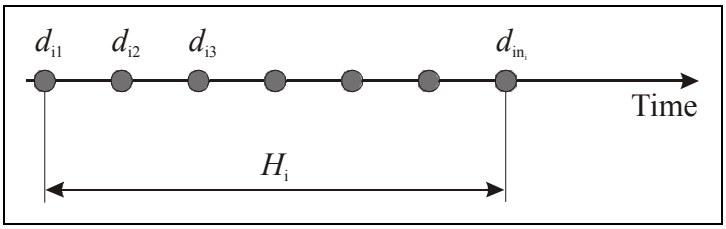

Figure 4.14 – Possible departure times from the first station of the transit line l_i

We can see from the figure 4.14 that there is the set $D_i = \{ d_{i1}, d_{i2}, \ldots, d_{in_i} \}$ composed of the n_i possible departure times within the headway H_i ($i = 1, 2, \ldots, m$).

Let us assume that public transit network is considered within a certain period of time T (Figure 4.15).

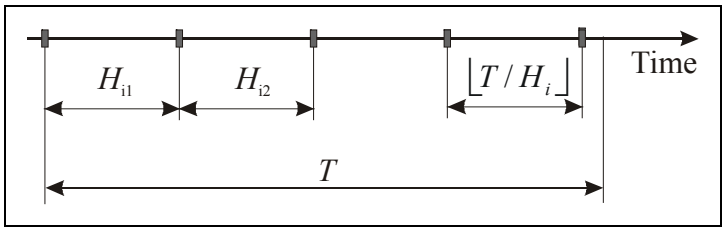

Figure 4.15 – Considered time interval T and possible headways

In order to take into account variations in the number of transfer passengers for every line pair, it is possible to subdivide entire interval T into ξ parts (Figure 4.16). The number ξ is given in advance.

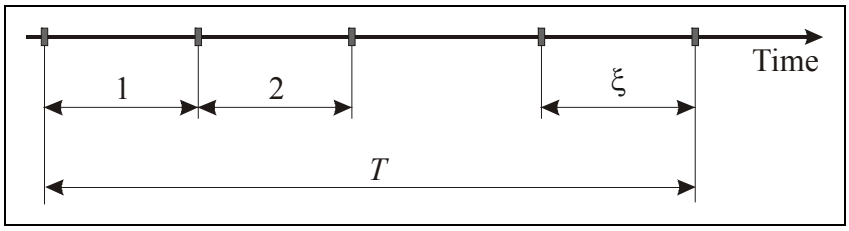

Figure 4.16 – Horizon T subdivided into intervals with approximately the same number of transfer passengers

Let $\mathbf{P}_{iju}$ be number of transfer passengers from transit line l_i to transit line l_j when transfers occurring within interval u. It has been already mentioned that the initial assumption is that the number of transfer passengers is only approximately known for every intersecting transit line pair. Furthermore, the number of transfer passengers will be expressed as triangular fuzzy number as follows:

$$\mathbf{P}_{iju} = (p_{iju1}, p_{iju2}, p_{iju3}), \quad i = 1, 2, \ldots, m. \qquad (4.35)$$

$$j = 1, 2, \ldots, m.$$

$$u = 1, 2, \ldots, \xi.$$

where, following the previous notation, p_{iju1} is the lower (left) boundary of the triangular fuzzy number, p_{iju2} is a number corresponding to the highest level of presumption, and p_{iju3} is the upper (right) boundary of the fuzzy number (Figure 4.17).

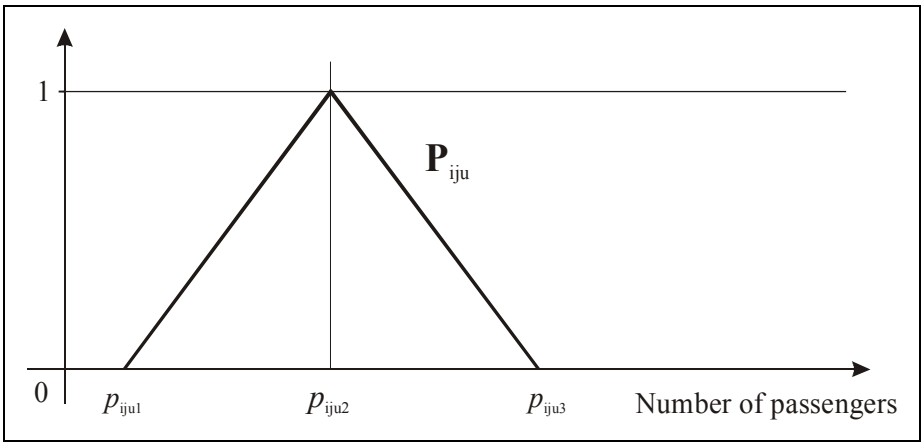

Figure 4.17 – Triangular fuzzy numbers representing the number of transfer passengers

We cannot expect to have fewer than p_{iju1} or more than p_{iju3} transfer passengers. The most expected value of the transfer passengers equals p_{iju2}. The values of p_{iju1}, p_{iju2}, and p_{iju3} can be determined using all available statistical data, as well as the experience and intuition of the transportation experts. The more precise statistical data we have, the smaller the uncertainty and the smaller the difference between p_{iju3} and p_{iju1}.

The total waiting time, $\boldsymbol{W}_{pq}^{rs}$, of all transfer passengers from transit line l_p to transit line l_q when the departure times from the first stations of these lines are d_{pr} and d_{qs}, equals:

$$\boldsymbol{W}_{pq}^{rs} = \sum_{a=0}^{\left\lfloor \frac{T}{H_p} \right\rfloor} \left(\sum_{b=0}^{\left\lfloor \frac{T}{H_q} \right\rfloor} \rho_{pq}^{rs}(a,b) \cdot \omega_{pq}^{rs}(a,b) \cdot (\tau_{qs} + bH_q) + \left(1 - \sum_{b=0}^{\left\lfloor \frac{T}{H_q} \right\rfloor} \rho_{pq}^{rs}(a,b) \cdot \omega_{pq}^{rs}(a,b) \right) T - (\tau_{pr} + aH_p) \right) \sum_{u=1}^{\xi} \xi_p^r(a,u) \boldsymbol{P}_{pqu} \tag{4.36}$$

where:

τ_{ir}, τ_{js} – the first arrival times at the intersection station of the lines l_i and l_j when departure times from the starting stations are respectively d_{ir} and d_{js}

ρ, ω, ξ - binary variables defined as follows:

$$\rho_{pq}^{rs}(a,b) = \begin{cases} 1, & \tau_{pr} + aH_p < \tau_{qs} + bH_q \\ 0, & \text{otherwise} \end{cases} \tag{4.37}$$

$$\omega_{pq}^{rs}(a,b)=\begin{cases}1, & \left|\tau_{pr}+aH_p-\tau_{qs}-bH_q\right|<H_q\\ 0, & \text{otherwise}\end{cases} \tag{4.38}$$

$$\xi_p^r(a,u)=\begin{cases}1, & (u-1)\dfrac{T}{\xi}\le\tau_{pr}+aH_p<u\dfrac{T}{\xi}\\ 0, & \text{otherwise}\end{cases} \tag{4.39}$$

A graphical representation of waiting times is given on figure 4.18.

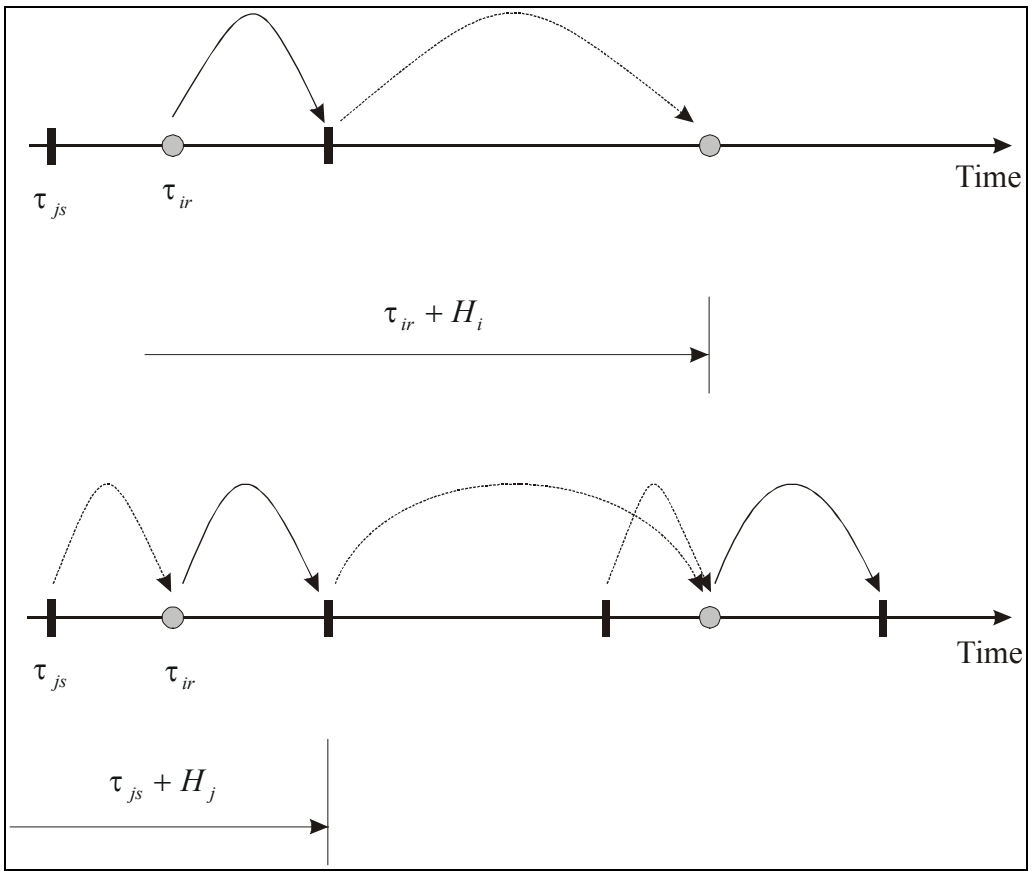

Figure 4.18 – Waiting times for intersecting transit line pair (i, j)

The total waiting time that occurs when d_{ir} is chosen as a starting time for line l_i and d_{js} is chosen as a starting time for line l_j, could be presented in the following way:

$$\boldsymbol{W}_{irjs} = \boldsymbol{W}_{ij}^{rs} + \boldsymbol{W}_{ji}^{sr} \tag{4.40}$$

The total waiting time $\boldsymbol{W}_{irjs}$ is a triangular fuzzy number that can be easily calculated using Fuzzy arithmetic rules (Kaufman and Gupta (1985), Teodorović and Vukadinović (1998)).

Let us introduce the following binary variables:

$$x_{ir} = \begin{cases} 1, & \text{if departure from the first station of the line } l_{\text{i}} \text{ is in time point } d_{\text{ir}} \\ 0, & \text{otherwise} \end{cases} \tag{4.41}$$

The total waiting time of all transfer passengers in the whole transit network equals:

$$\boldsymbol{W} = \sum_{i=1}^{m} \sum_{r=1}^{n_i} \sum_{\substack{j=1 \\ j \neq i}}^{m} \sum_{s=1}^{n_j} x_{ir} x_{js} \boldsymbol{W}_{irjs} \tag{4.42}$$

The departure from the first station of any transit line l_{i} must be in one of the time points d_{i1}, d_{i2}, …, d_{in_i}. In other words, the following relation must be satisfied:

$$\sum_{r=1}^{n_i} x_{ir} = 1, \qquad i = 1, 2, \ldots, m. \tag{4.43}$$

The problem of transit schedule synchronization could be defined in the following way: *for the known line headways for every transit line, determine the departure times from the first station so as to minimize the total waiting times of all passengers at transfer nodes in a whole transit network*. Mathematically, the problem of schedule synchronization could be defined as follows:

$$\text{Minimize} \quad \boldsymbol{W} = \sum_{i=1}^{m} \sum_{r=1}^{n_i} \sum_{\substack{j=1 \\ j \neq i}}^{m} \sum_{s=1}^{n_j} x_{ir} x_{js} \boldsymbol{W}_{irjs} \tag{4.44}$$

subject to:

$$\sum_{r=1}^{n_i} x_{ir} = 1, \quad i = 1, 2, \ldots, m \tag{4.45}$$

$$x_{\mathrm{ir}} \in \{0, 1\} \quad i = 1, 2, \ldots m, \quad r = 1, 2, \ldots, n_{\mathrm{i}}$$

4.9.3 Schedule synchronization in public transit using the Fuzzy Ant System

It is possible to try to determine the departure times simultaneously for all transit lines, or sequentially, line by line. Here the sequential approach is used.

Let us sort the lines in descending order of the number of transfer points. There are also some other options for this sorting (descending order of the number of transfer passengers, randomly, etc).

Let us denote the total number of ants by n. Let us also locate all ants in the starting terminal of the transit line with the greatest number of intersection points. Ants will walk along the transit lines one by one. The first ant located at the starting terminal of the transit line with the greatest number of intersections will walk first. An ant will randomly choose its departure time. An ant's departure time choice mechanism will be explained in more detail later. After randomly choosing the departure time from the first station, the first ant will walk to the end of the first line, and return (by walking in opposite direction) to the starting terminal. For every station along the line the ant's departure and arrival time are known. At this moment the total waiting times of all transfer passengers equals zero. The first ant will then go to the starting terminal of the transit line with the second greatest number of intersections. Then, it will again randomly choose the departure time associated with the second transit line. The first ant will also walk to the end of the second transit line, change direction and return to the starting point. In this way, the first ant's departure and arrival time for every station along the second line are determined. If the second transit line has the intersection with the first one, the total waiting time of all transfer passengers from the first transit line to the second transit line and from the second transit line to the first transit line will be calculated. In this calculation we will use relations (4.36)-(4.40). If these two transit lines do not intersect the total waiting time will

still equal zero. The first ant will start its walk along the third transit line. After that, calculation of the total waiting time of all transfer passengers from the first transit line to the third transit line, from the third transit line to the first transit line, from the second transit line to the third transit line, and from the third transit line to the second transit line is performed. After that, the first ant will start to walk along the fourth transit line, and so on. When the first ant finishes its walks along all transit lines, the first iteration will be finished. This means that the first alternative of the public transit schedule will be determined. After the first ant, the second ant will start to generate the second alternative of the public transit schedule. When n iterations are finished, n different public transit schedules will be generated. As in previous example, one cycle is composed of n iterations. The pheromone trail will be updated at the end of the first cycle. After that, all n ants will start the second cycle. At the end of the second cycle, the second set of public transit schedules will be created, and the pheromone trail will be updated again. Ants will then start the third cycle, etc. Ants will finish with the public transit schedule generation after t_{max} cycles, where t_{max} is a previously defined number of cycles. The best solution discovered during the search process will contain final departure times on all transit lines.

4.9.3.1 Departure time choice mechanism

Ants' movements could also be graphically represented in the following way. Let us note the network shown in figure 4.19. The network is composed of a few layers. The total number of layers in the network, m, is equal to the total number of transit lines. The number of nodes in every layer equals to the number of possible transit line departure times. There is a full connectivity between the two neighboring layers. Let us assume that all ants are located in origin O and that all of them travel to destination D. Every node in the network is described by the appropriate coordinates. For example, the coordinates (i, r) describe r-th node located in the i-th layer. This node represents r-th possible departure moment of the i-th transit line.

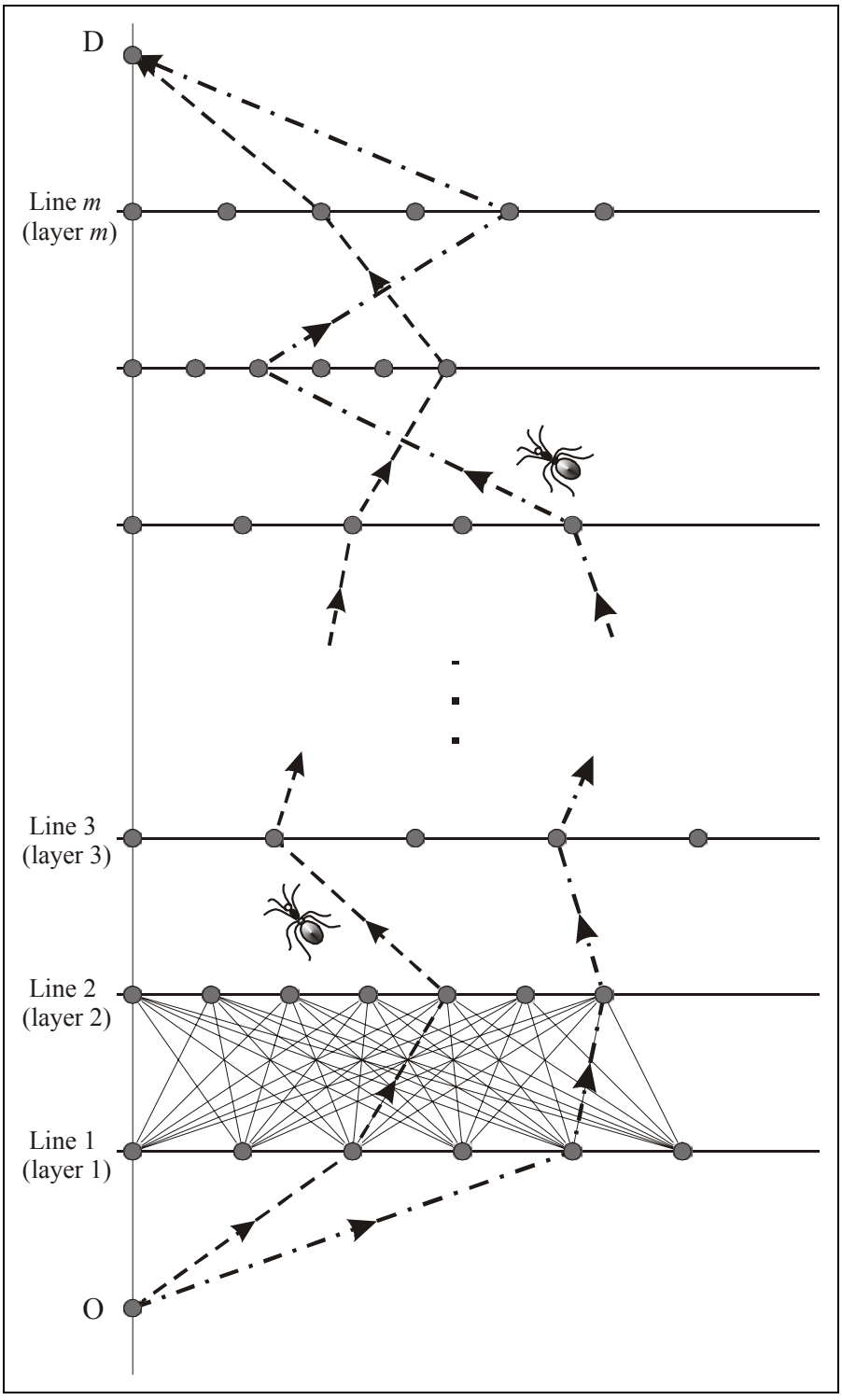

Figure 4.19 – Transit lines and possible departure times

Ants located in origin *O* go towards destination *D*. The ants have few options when choosing the first node in the first layer (the departure time of the first transit line). The

number of choices is equal to the number of possible departure times. A pheromone will be deposited at every *node* that was visited by at least one ant. At the beginning of the search process, like in other AS applications, it will be assumed that the pheromone trail is very low in every node and that it is equal to some small positive constant (c).

An ant starts its trip from the origin, chooses one node in the first layer, then moves to the second layer, chooses one node from the second layer, and so on. When considering a certain node (departure time) to be visited by an ant, we use the formulas (4.36) - (4.40) to calculate the potential waiting time of transfer passengers caused by an ant's possible choice. It should be underlined once again that passenger waiting time is a fuzzy number.

While deciding on the next node to be visited, the ant takes into account local information such as "visibility", as well as historical information such as pheromone trail intensity. It is assumed that the ant can perceive the particular waiting time as "small", "medium" or "long", and the trail intensity as "weak", "medium" or "strong". Possible membership functions for small, medium and long waiting time are shown in figure 4.20. In this example, it has been used the same set of membership functions like in previous example with respect to pheromone trail intensity. The set is shown in figure 4.9.

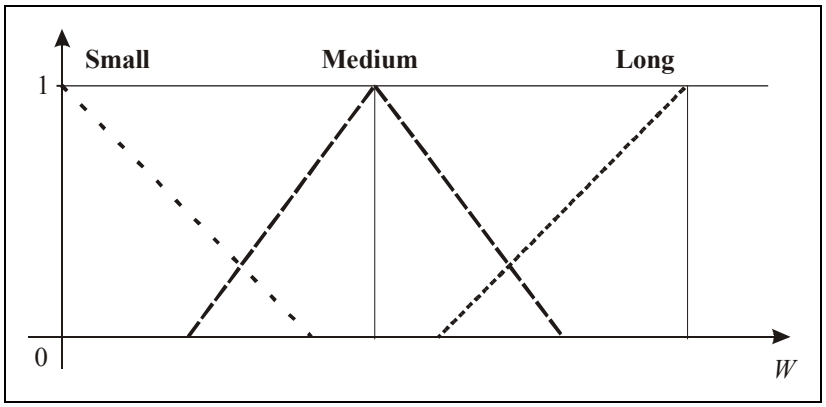

Figure 4.20 – Membership functions of the fuzzy sets describing expected waiting time

Depending on calculated total waiting time, as well as the trail intensity, the ant will have stronger or weaker utility to choose the considered node. These utilities can be described by appropriate fuzzy sets and here is used the same fuzzy sets as in the previous example (Figure 4.7).

Let W^{k}_{ir} be the total (cumulative) waiting time caused by decisions that ant k has made when walking up to the layer i; τ_{ir} be the pheromone intensity in node (i, r); and u^{k}_{ir} be the value of the utility that ant has where it chose the departure r of the transit line i respectively.

An approximate reasoning algorithm for calculating the utility of choosing the next node could be composed, for example, of the rules of the following type:

If W^{k}_{ir} is **SMALL** and τ_{ir} is **STRONG**
Then u^{k}_{ir} is **VERY HIGH**

The ant's utility for choosing the next node would be calculated using the following approximate reasoning algorithm:

Rule 1:
If W^{k}_{ir} is **SMALL** and τ_{ir} is **WEAK**
Then u^{k}_{ir} is **HIGH**
else
Rule 2:
If W^{k}_{ir} is **SMALL** and τ_{ir} is **MEDIUM**
Then u^{k}_{ir} is **VERY HIGH**
else
Rule 3:
If W^{k}_{ir} is **SMALL** and τ_{ir} is **STRONG**
Then u^{k}_{ir} is **VERY VERY HIGH**
else
Rule 4:
If W^{k}_{ir} is **MEDIUM** and τ_{ir} is **WEAK**
Then u^{k}_{ir} is **LOW**
else
Rule 5:
If W^{k}_{ir} is **MEDIUM** and τ_{ir} is **MEDIUM**
Then u^{k}_{ir} is **MEDIUM**
else
Rule 6:
If W^{k}_{ir} is **MEDIUM** and τ_{ir} is **STRONG**
Then u^{k}_{ir} is **HIGH**
else
Rule 7:
If W^{k}_{ir} is **BIG** and τ_{ir} is **WEAK**

Then u^{k}_{ir} is **VERY VERY LOW**

else

Rule 8:

If W^{k}_{ir} is **BIG** and τ_{ir} is **MEDIUM**

Then u^{k}_{ir} is **VERY LOW**

else

Rule 9:

If W^{k}_{ir} is **BIG** and τ_{ir} is **STRONG**

Then u^{k}_{ir} is **LOW**

Graphical representation of fuzzy set reasoning algorithm is given in figure 4.21.

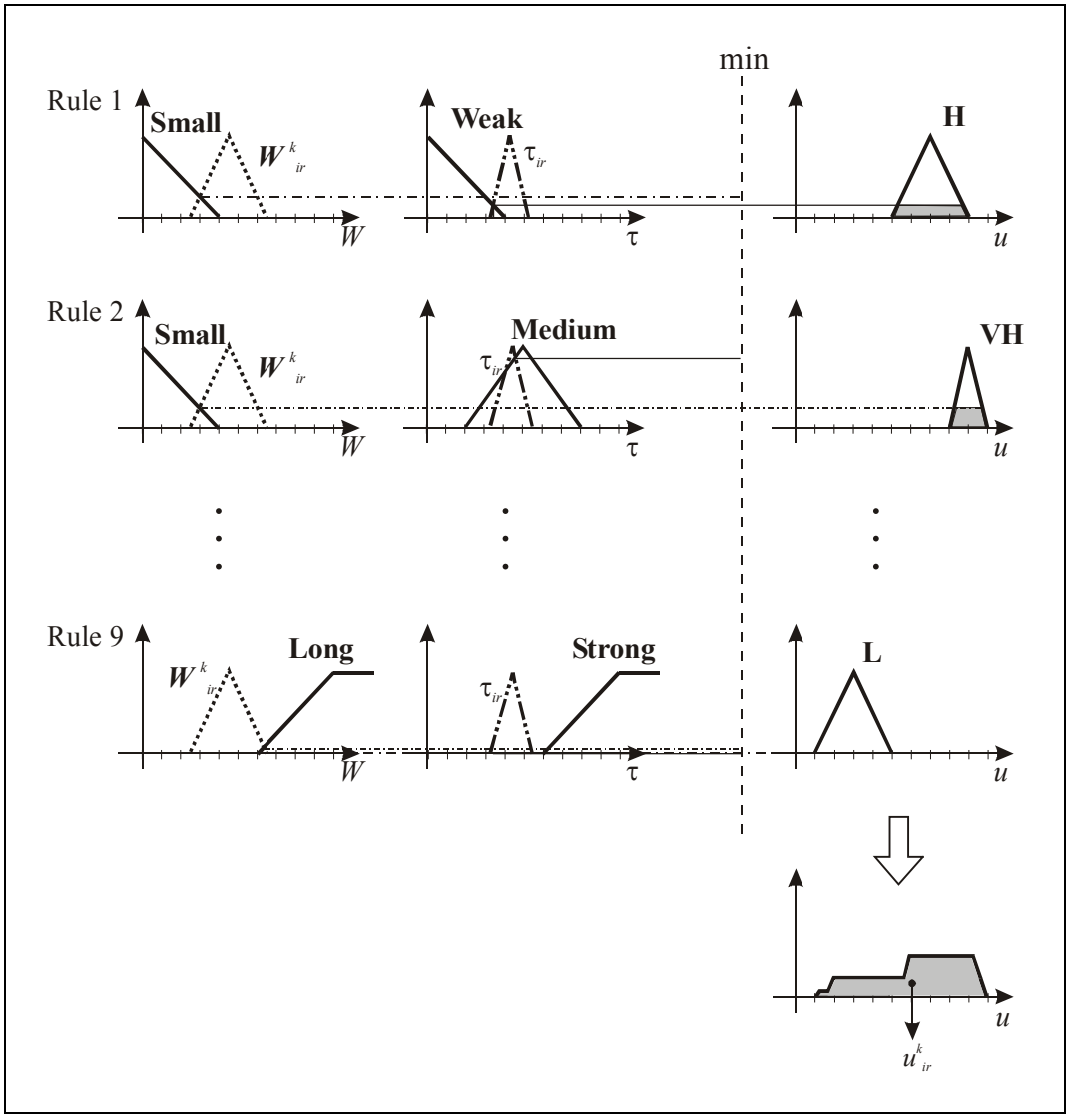

Figure 4.21 – Graphical representation of the approximate reasoning algorithm

The nodes with better utility values are more likely to be selected by an ant. The probability p^{k}_{ir} for node (i, r) to be selected by ant k will be equal, as before, to the ratio of u^{k}_{ir} to the sum of all nodes' utility values in the group of considered nodes:

$$p_{ir}^{k} = \frac{u_{ir}^{k}}{\sum_{h=1}^{n_i} u_{ih}^{k}} \tag{4.46}$$

This type of selection represents a proportional selection known in Evolutionary programming as the "roulette wheel selection." (The sections of roulette are proportional to probabilities p^{k}_{ir}).

4.9.3.2 Pheromone update

When n different transit schedules are generated, one cycle in the searching process is finished. Pheromone trail will be updated after every finished cycle. In contrast to previous example, pheromone will be given only to *nodes*. Let us denote by (i, r) the coordinates of the node representing r-th possible departure time of the i-th transit line. Amount of pheromone associated to node (i, r) would be updated in the following way:

$$\tau_{ir}(t+1) = \rho\,\tau_{ir}(t) + \Delta\tau_{ir}(t, t+1) \tag{4.47}$$

where:

ρ is the coefficient ($0 < \rho < 1$) such that ($1 - \rho$) represents evaporation of the trail in cycle.

The total increase in trail intensity in node (i, r) after one completed cycle is equal to:

$$\Delta\tau_{ir}(t, t+1) = \sum_{k=1}^{n} \Delta\tau_{ir}^{k}(t) \tag{4.48}$$

where:

$\Delta\tau_{ir}^{k}(t)$ is the quantity of pheromone laid in node (i, r) by the k-th ant at the end of cycle t.

The quantity $\Delta\tau_{ir}^{k}(t)$ is given by:

$$\Delta\tau_{ir}^{k}(t)=\begin{cases}(1+ab_k(t))\dfrac{Q}{W_k(t)}, & \text{if } k\text{ - th ant chose node } (i,r) \text{ in toure performed trough cycle } t\\ 0, & \text{otherwise}\end{cases} \quad (4.49)$$

where:

Q, a - the constants,

$W_k(t)$ - the total waiting time of all passengers in the case of the schedule generated by the k-th ant in the cycle t.

$$b_k(t)=\begin{cases}1, & \text{if } \quad W_k(t)\le\left(1+x\dfrac{t_{\max}-t+1}{t_{\max}}\right)\cdot W^{*}\\ 0, & \text{otherwise}\end{cases} \quad (4.50)$$

where:

W^{*} - the best criteria value discovered in all previous cycles,

x - parameter given in advance ($x\in(0, 1)$),

$t_{\max}$ - total number of cycles that will be performed by algorithm.

Let us explain relations (4.49) and (4.50) in more details. All nodes visited by the k-th ant will at least get the amount of pheromone equal to $\frac{Q}{W_k(t)}$. The "good" nodes will get even more pheromone. The amount of pheromone that will be given to the "good" nodes equals to $(1+ab_k(t))\frac{Q}{W_k(t)}$. The idea is to differentiate among nodes and extract some of them that are more promising for future search. Using relation (4.50) it is possible to discover those "good" nodes. The "good" nodes belong to the path that will lead to the objective function value that is "close enough" to the best objective function value discovered in all previous cycles. Parameter $b_k(t,t+1)$ that is additionally rewarding "good" nodes is time dependent. We can see from relation (4.50) that we treat more nodes as "good" nodes at the beginning of a search process. The more time passes during which the ants are searching for a solution, the stricter we are in declaring some nodes to be "good" nodes. In other words, as more time passes, the search process should be more

focused to search primarily the paths producing objective function values relatively close to the best objective function value discovered in all previous cycles.

4.9.3.3 Comparison of fuzzy numbers during search process

The total waiting time $\boldsymbol{W}_k(t)$ of all passengers in the case of the schedule generated by the *k*-th ant in the cycle *t* represents a triangular fuzzy number. Quantity $\Delta\boldsymbol{\tau}_{ir}^{k}(t)$ could be treated as a triangular fuzzy number. Parameter $b_k(t)$ that is additionally rewarding "good" nodes is time dependent. This parameter can take the value 0 or value 1. To reach the conclusion about the parameter $b_k(t)$ value we must compare fuzzy numbers $\boldsymbol{W}_k(t)$ and $\left(1+x\frac{t_{\max}-t+1}{t_{\max}}\right)\cdot\boldsymbol{W}^{*}$.

The problem of comparing fuzzy numbers has been the subject of many papers. Here, for comparison of fuzzy numbers Kaufmann and Gupta's (1988) method is used. This method is chosen primarily because of its simplicity. The method is based on the concept of "removal" of a fuzzy number.

Kaufmann and Gupta's (1988) method for comparing fuzzy numbers comprises the following steps:

Step 1: Compare the "removal" of the numbers. If a conclusion can be made based on this comparison, the algorithm is ended. Otherwise, go to Step 2.

Step 2: Compare the values that correspond to the highest grades of membership. If a number order can be determined after this comparison, the algorithm is ended. If a conclusion cannot be made after comparing the highest grades of membership, go to Step 3.

Step 3: Compare the length of the fuzzy numbers' bases.

Let us note figure 4.22. The "Left removal" $R_l(\mathbf{A}, a)$ of fuzzy number **A** compared to real number *a* consists of the surface between real number *a* and the left side of fuzzy number **A**. The "right removal" $R_r(\mathbf{A}, a)$ of fuzzy number **A** is defined as the surface between real number *a* and the right side of fuzzy number **A**.

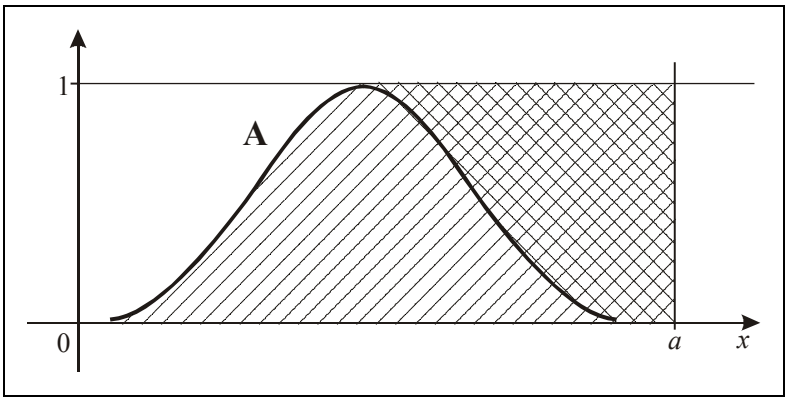

Figure 4.22 – Fuzzy number **A**'s "removal" compared to real number *a*

The "removal" of fuzzy number **A** compared to real number *k* is defined as

$$\mathrm{R}(\mathbf{A}, a) = \frac{\left(\mathrm{R}_l(\mathbf{A}, a) + \mathrm{R}_r(\mathbf{A}, a)\right)}{2} \tag{4.51}$$

Figure 4.22 shows the "left removal," "right removal," and "removal" of fuzzy number **A** compared to real number *a*. When fuzzy number **A** has a triangular shape and when $a = 0$, it can be easily shown that the "removal" of fuzzy number **A** equals:

$$\mathrm{R}(\mathbf{A}, 0) = \frac{\left(a_1 + 2a_2 + a_3\right)}{4} \tag{4.52}$$

Fuzzy number **A** is smaller than fuzzy number **B** if

$$\mathrm{R}(\mathbf{A}, a) < \mathrm{R}(\mathbf{B}, a) \tag{4.53}$$

When $\mathrm{R}(\mathbf{A}, a) = \mathrm{R}(\mathbf{B}, a)$, the second algorithmic step must be used and the values of the highest grades of membership must be compared. Let x^*_{A} and x^*_{B} denote the highest grades of membership in fuzzy sets **A** and **B**. Fuzzy number **A** is smaller than fuzzy number **B** if

$$x^*_{\mathrm{A}} < x^*_{\mathrm{B}} \tag{4.54}$$

When $\mathrm{R}(\mathbf{A}, a) = \mathrm{R}(\mathbf{B}, a)$ and $x^*_{\mathrm{A}} = x^*_{\mathrm{B}}$, the third step must be used, which compares the bases of the fuzzy numbers. Fuzzy number **A** is smaller than fuzzy number **B** if the base of fuzzy number **A** is smaller than the base of fuzzy number **B**.

4.9.4 Numerical example

The model developed is tested on a greater number of numerical examples. The results for a transit network containing 50 transit lines are presented. The configuration of the network was generated randomly. The example was treated under the following assumptions:

- T = 3 hours,
- $\xi = 1$,
- $n_i = 5, \ \forall \ i.$

Input data representing network configuration were organized in a way presented in tables 4.2 and 4.3.

Table 4.2 – Intersecting points data for public transit line pair (i, j)

Transit line pair		Travel time from the first station to the intersecting point – line l_i [h]	$\mathbf{P}_{ij}$			Travel time from the first station to the intersecting point – line l_j [h]	$\mathbf{P}_{ji}$		
l_i	l_i		p_{ij1}	p_{ij2}	p_{ij3}		p_{ji1}	p_{ji2}	p_{ji3}
1	2	0.4094	10	11	12	0.1492	4	4	4
1	3	0.7458	12	12	12	0.3655	3	4	5
1	6	0.5216	6	9	12	0.6677	4	5	6
1	10	0.2196	8	10	12	0.4071	8	13	18
1	14	0.3338	6	9	12	0.2158	12	12	12
1	17	0.3435	10	13	16	0.5414	12	13	14
1	22	0.774	3	4	5	0.1492	10	10	10
1	24	0.3104	4	5	6	0.337	6	7	8
1	25	0.779	10	10	10	0.7623	4	5	6
1	29	0.4387	6	7	8	0.5344	6	7	8
1	32	0.6509	3	4	5	0.2928	2	3	4
1	37	0.6612	9	11	13	0.2522	12	13	14
1	46	0.2768	8	11	14	0.2307	6	8	10
1	50	0.6414	7	10	13	0.1367	10	13	16
2	7	0.3703	6	9	12	0.309	3	5	7
2	8	0.5984	7	7	7	0.2288	6	7	8
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
43	47	0.2803	10	14	18	0.3281	8	10	12
47	48	0.1234	8	8	8	0.4141	5	6	7
47	49	0.3022	8	11	14	0.5387	6	9	12

Table 4.3 – Transit lines data

Line number (*i*)	H_i [h]	n_i	No intersecting points
1	0.5	5	15
2	0.6667	5	14
3	0.4167	5	13
4	0.5833	5	13
. .	. .	. .	. .
48	0.4167	5	5
49	0.75	5	4
50	0.5	5	4

Achieved final results are decisions for each transit route (l_i) with ordinary number of starting time point d_{ir}. Data pairs i (d_{ir}) are presented below:

1(3), 2(4), 3(3), 4(3), 5(4), 6(4), 7(5), 8(5), 9(4), 10(5), 11(4), 12(5), 13(4), 14(5), 15(5), 16(5), 17(1), 18(4), 19(4), 20(2), 21(5), 22(5), 23(5), 24(3), 25(3), 26(2), 27(3), 28(5), 29(1), 30(5), 31(5), 32(5), 33(5), 34(4), 35(3), 36(2), 37(3), 38(1), 39(5), 40(5), 41(3), 42(1), 43(1), 44(1), 45(3), 46(1), 47(2), 48(4), 49(1) and 50(2).

In the example, the highest number of intersecting points per line was 15 and the lowest was 4. The best criteria value changes through cycles are shown in figure 4.23.

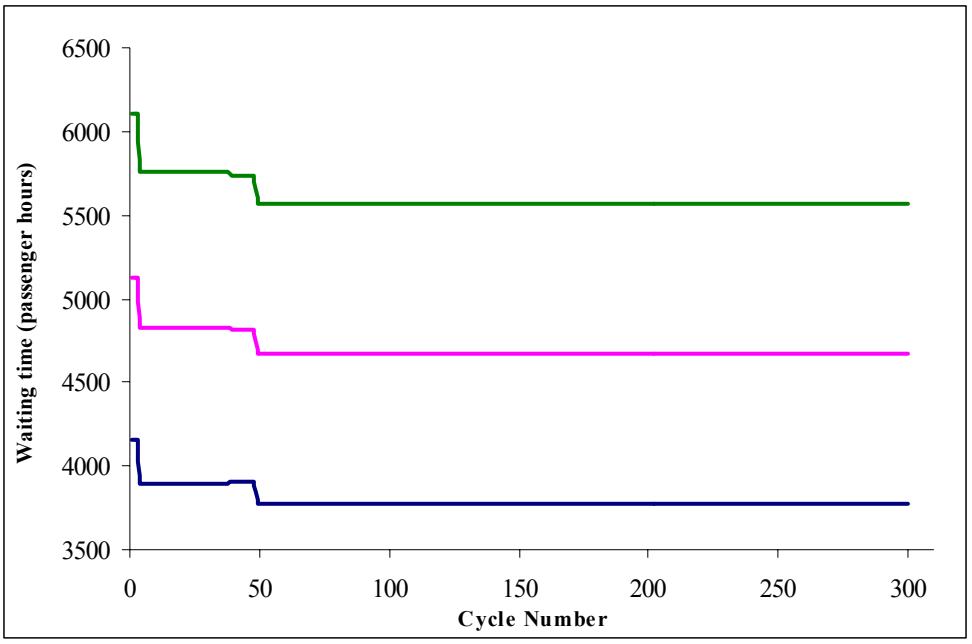

Figure 4.23 – The best criteria value changes through cycles

Three curves shown in figure 4.23 represent the lowest possible, expected and the maximum possible waiting time values W^*. Relatively fast discovery of a good solution can be seen in figure 4.23. The total number of ants was equal to 10. Computer experiments were performed using a 600MHz Pentium III. The total computer time necessary to produce one transit schedule was about one second. Relatively low values of computer time enable the analyst to explore different transit network configurations, and different headways values, and to choose the most desirable transit schedule.

Chapter 5. Summary, conclusions and recommendations for future research

5.1 Summary and conclusions

Many real-world problems could be modeled as discrete optimization problems; therefore a need to develop tools capable of solving such problems efficiently is significant. There are numerous techniques available to solve discrete optimization problems. Some of these techniques are capable of providing an optimal solution to the problem. These techniques could be used to discover optimal solutions to the problem in instances of small dimensionality. The remaining techniques are either heuristics (provide solutions of "acceptable" quality in reasonable amount of computer time for some specific problems) or metaheuristics (provide "good enough" solutions in reasonable amount of computer time for variety of problems). Real life applications are most frequently large in dimensionality and very often decision makers are satisfied with suboptimal solution(s) to their problem.

In the last couple of decades, metaheuristic algorithms have increasingly been used in solving discrete optimization problems. Early stages of metaheuristics development include the following techniques: simulated annealing, genetic algorithms and tabu search. The successful applications of these techniques in variety of complex engineering problems encourage researchers to try building new techniques having better performances than previous ones. They mostly identify the natural systems as a source of ideas and models for development of various artificial systems. During the last decade, researchers identified possibilities to use concepts obtained from studying the behavior of social insects for developing artificial systems that could be employed in optimization. Among the other behaviors, it has been shown that analysis of foraging behavior of different swarms is essential.

Many Transportation Engineering problems can be addressed by discrete optimization. Furthermore, numerous transportation engineering parameters are characterized by

uncertainty, subjectivity, and imprecision. Human operators, dispatchers, drivers, and passengers often use subjective knowledge or linguistic information on a daily basis while they make different decisions. The entire environment in which decisions should be made is often complex, and formulating a suitable mathematical model could be rather difficult.

This dissertation has two parts. The first part of the dissertation is devoted to developing a new technique based on the foraging behavior of natural bees – the Bee System. The artificial Bee System has been successfully applied to the classical *Traveling Salesman Problem*. This application of the Bee System to the difficult combinatorial optimization problem is very encouraging as it confirms the value of natural systems as a source of ideas and models for development of various useful artificial systems.

Additionally, in this research the *Stochastic Vehicle Routing* problem is considered on well-known set of nodes (statistical data about node demand is available). Demands at nodes are considered to be random variables. The problem was attacked by sequential usage of the following two techniques: the Bee System and Fuzzy Logic. The first step was to consider all the nodes and solve the problem as a Traveling Salesman Problem by the developed Bee System. When solving TSP node demand is ignored and one "giant" route is produced. Through the second step, a tool capable of subdividing "giant" route (usually unfeasible solution to the original problem) into set of routes (feasible solution) is produced. Based on the assumption that we have perfect knowledge about future demand at nodes, we first design a fuzzy logic tool (define variables that would appear in antecedent and consequent part, their ranges and appropriate fuzzy numbers that will cover the ranges) and then collect an appropriate set of input output data pairs. One set of input output data pairs corresponds to one simulation of future demands at nodes. Simulations of future demands at nodes and data collection are repeated numerous times. The set of simulations was subdivided into two parts, the first one was used to build a fuzzy rule base and the other one to test the quality of decisions that the fuzzy controller is producing once it is developed.

The entire tool would be used as decision support system that will provide judgment for a vehicle located at one node whether to go to the other – adjacent node on the "giant"

route or return to depot.

The second part of the dissertation is focused on an attempt to develop tool capable of solving discrete optimization problems characterized by uncertainty. To solve such problems, a metaheuristic approach is proposed based on a combination of the existing Ant System and Fuzzy Logic. It could be seen, through publications, that researchers mostly stay focused on improvement of the part of the Ant System algorithm related to collaboration among individuals with the intent to improve its efficiency in general. None of them had tried to change any part (even minor) of the transition probability – the way in which agents will make individual decisions. This research proposed the calculation of transition probabilities based on normalization of an agent's utilities to visit particular states in a solution space. It has been proposed to use Fuzzy Logic to obtain those utilities.

The approach was tested on the following two problems:

- Stochastic Vehicle Routing Problem and
- Scheduling Synchronization in Public Transit.

In the *Stochastic Vehicle Routing Problem*, the source of uncertainty was node demand. Very often only approximate values of demand at nodes are known. Sometimes it is quite difficult and very costly to collect relevant statistical data and to make accurate predictions of the demand values. In some cases, such data does not exist at all. These are the reasons why the demand values were treated as triangular fuzzy numbers. Using triangular fuzzy numbers could make communication between the analyst and practitioner easier. Practitioners usually do not have any difficulties accepting the concept of triangular fuzzy numbers. Using knowledge, experience and intuition, they can determine "least expected demand" (left boundary of triangular fuzzy number), or "greatest expected demand" (right boundary of triangular fuzzy number) relatively easily. The developed Fuzzy Ant System was successful in producing sets of routes for the problem instances considered.

The average waiting times while making transfers are the direct consequence of the

schedule synchronization. While making schedule synchronization, it is necessary to try to minimize the total waiting times of all passengers at transfer nodes in the transit network. In the previous research, the numbers of passengers making transfers were treated as deterministic quantities. However, very often only approximate numbers of transfer passengers are known. Sometimes it is very difficult and costly to collect relevant statistical data and to make accurate predictions of the number of transfer passengers. In the case of a new or reconstructed transit network, such data usually do not exist at all. These were the reasons for treating the numbers of transfer passengers as fuzzy numbers. All the other quantities that serve as input data for the problem are assumed to be crisp values. The Fuzzy Ant System was applied to the problem and obtained solutions were promising. Usually, when applying the Ant System, pheromone is deposited along the links in the network. The nature of the problem caused pheromone to be deposited just in network nodes. At the same time, due to calculations, the values of the deposited pheromone at nodes were only approximately known.

The Swarm intelligence approach has huge potential. The approach offers a new way of modeling different complex systems. Instead of building centralized control and extensive preprocessing, the system will relay on direct or indirect interaction among simple agents. The Swarm intelligence approach in discrete optimization should have a promising future. Belonging to the group of Evolutionary Algorithms, the Swarm intelligence approach is easy to parallelize. The main advantage of the Swarm Intelligence approach is the possibility to assign (create) different properties to each individual agent. Individual agents could be grouped with the idea to achieve a common goal. Individual agents could be in reality very simple or be enriched with a more complex algorithm. Development of simple, but at the same time robust hybrid algorithms would be a promising direction of future research. Swarm intelligence algorithms have the capability to be part of the future.

The main disadvantage of Swarm Intelligence models is their possible unpredictable behavior. This occurs because of the lack of detailed understanding of the rules that agents use to interact in nature. Better understanding of the rules in nature is crucial to avoid the appearance of unpredictable behavior of artificial swarms.

5.2 Dissertation contribution

The dissertation contributions are as follows:

Metaheuristic algorithms have increasingly been used in solving problems belonging to discrete optimization. Natural systems are mostly identified as a source of ideas and models for development of various artificial systems capable of providing "good" solution(s) to variety of engineering problems. During the last decade, researchers identified possibilities of using concepts obtained from studying the behavior of social insects for developing artificial systems that could be employed in optimization. One contribution of the dissertation to the idea of usage concepts taken from studying of real swarms in optimization is *development of a new artificial system based on foraging behavior of bee colonies – the Bee System*.

Many real life problems, besides being of combinatorial nature, are characterized by the uncertainty associated to one or more of their independent parameters. In order to treat these problems using the metaheuristic approach, *the Fuzzy Ant System is proposed*. The proposed algorithm presents a hybrid approach; it combines existing results in the area of swarm intelligence (existing Ant System) and approximate reasoning. The approximate reasoning algorithm is used to help every individual agent (artificial ant) make its decisions when some of the independent parameters are just approximately known.

The Stochastic Vehicle Routing Problem is considered in two its variations and *two new approaches to the problem are proposed* as follows:

- Sequential usage of the developed Bees System and Approximate Reasoning Algorithm developed from numerical example is proposed to treat the problem when demand at nodes is described by known probability density function.
- The Fuzzy Ant System is proposed to treat the problem when appropriate fuzzy numbers describe stochasticity associated to the demands at nodes.

The Fuzzy Ant System approach to the Scheduling Synchronization in Public Transit problem is proposed.

5.3 Recommendations for future research

In future research, one of the main goals should be the exploration of the different types of artificial bee organizations and interactions and their influence on the dynamics of the population. There are questions relating to the above mentioned characteristics of the social insects that need to be further answered in future research: (a) Should artificial bees (agents) be equal, or should there sometimes be several types of agents? (b) Are there further possibilities for hybrid combination of swarm intelligence and other artificial intelligence approaches and different heuristic algorithms? (c) What are the costs and benefits of the development of Artificial Systems based on natural Swarm Intelligence in engineering, computer science and management science?
Swarm Intelligence (Ant System) has already been applied in some other engineering areas such as robotics. The results obtained are also very good. In further research, models inspired by the developed Bee System could be created for different transportation engineering problems. The quality of the results obtained in the classical TSP indicates that the development of new models based on swarm intelligence principles could significantly contribute to the solution of complex transportation engineering problems.

The developed Bee System was employed as the first step in the process of solving the Stochastic Vehicle Routing Problem. The second step in the proposed solution was development (based on set of input output data pairs) and employment of a Fuzzy Logic controller with the purpose of making real time decisions, such as whether a vehicle being located at one particular node will go to an adjacent node or return to the depot. Instead of developing a solution in advance, a set of routes was developed based on real time decisions.
The Fuzzy Logic based controller had an antecedent part containing just variables associated with remaining vehicle capacity and demand characteristics of successive node. Future research in this area should explore the possibilities of incorporating distances from the node where the vehicle is currently located to the depot and from the node next to the current vehicle location to the depot into the decision process

(antecedent part).

A short description of some of the potential applications of the Bee System in development of Artificial Systems aimed to solve complex problems in transportation engineering is provided. The following examples were chosen among a great number of potential applications of the developed Bee System.

Planning and designing the transit, airline, or utility networks is an extremely complex planning task combinatorial by its nature. The chosen network shape and the vehicle frequency on individual links directly affect the business results of the operator and the level of service provided to passengers. Passengers in intercity transport and goods are very frequently routed from the origins to the destinations through one or more hubs. Instead of routing the passengers from each origin directly to their destination, the hub and spoke system transports passengers and goods through the hubs. Due to the highly competitive environment in air transportation, logistics and communication assumes the best possible hub and spoke architecture. It should also be underlined that when transportation is to be established among a large number of nodes, the dimensions of the problem become very large. The transportation network planning and design problem is the ideal problem for the application of the developed Bee System.

The classical vehicle routing problem consists of finding the set of routes that minimizes transport costs. Further variations of the classical vehicle routing problem include existing few depots in the network, doing service with a few types of different vehicles, uncertain demand at nodes, or existing time windows for doing service at certain nodes. Developing hybrid models (Bee System, Fuzzy Logic and Visualization) for solving complex vehicle routing and scheduling problems would be of a great benefit for both the distribution companies, and the wider public.

The highway alignment problem assumes selection of the “best” path to connect two points in the space. The “best” alignment minimizes total costs and satisfies the engineering design, operational constraints. Developing a hybrid model (Bee System and Fuzzy Logic) for solving highway alignment problem would be of a great benefit for transportation agencies, as well as to the wider public.

Different versions of the Dial-A-Ride problem are found in every day practice:

transportation of people in low-density areas, transportation of the disabled and elderly persons, and parcel pick-up and delivery service in urban areas are some of the examples. The dynamic Dial-A-Ride problem could be described as follows: all customers demanding fast service define pick-up and delivery location, as well as preferred beginning of the service. The problem is to assign every new passenger request to one of the vehicles already on the network, and to design a new route and schedule for this vehicle. This assignment has to be done in real time.

The Gate Assignment Problem represents the assignment of arriving aircraft to available gates. During the last decade, with the increase in the number of flights and number of passengers in air transportation, the airport gate assignment problem has become much more complex. This problem is a difficult combinatorial optimization problem. By its nature this problem is also dynamic. Originating passengers walk from the check-in to the departure gate. Transfer passengers walk from the arriving gate to the new departing gate, while terminating passengers walk from arriving gate to the baggage claim area. The total passenger walking distance depends on the passenger transfer volume between every pair of aircraft, as well as the distance between every pair of gates. It is very logical to try to minimize the total passenger walking distance. While solving this problem it is necessary to take into account different operational constraints. Developing Decision Support Systems based on the Bee System for the aircraft gate assignments would be also of great importance for the mitigation of airline schedule disturbances.

Berthing time of ships carrying containers accounts for a considerable portion of the journey. Decreasing the turnaround time at port would result in a reduction of total traveling time of ships and accordingly a reduction in the cost of container transportation. Berth allocation addresses the problem of determining berth assignment to ships in the public berth systems. The assignment should be done in a way to minimize summation of waiting and service times of all ships coming to the port within the planning horizon. Waiting time depends on previous assignment of the ships to the berth and service time depends on the distance between a ship (the berth where ship is assigned) and its container(s) location.

Currently there is a variety of metaheuristic algorithms and a direction of future research

would be comparison of the results obtained with the Bee System versus results obtained with the other techniques. Based on the comparison it is possible to make recommendations about a more promising approach to the problem.

One of directions for future research would be convergence analysis of Bee System algorithm.
Taking into account the Fuzzy Ant System and its application to the Stochastic Vehicle Routing Problem, it should be considered as a step in future research, application of tour improvement heuristic on the entire set of vehicle routes produced by each ant. In the case of applying *n*-opt algorithm it is necessary to introduce artificial nodes that represent a depot. The total number of nodes representing a depot would be equal to the total number of routes produced by the ant.

Additional sources of uncertainty when Scheduling Synchronization is considered could be travel time between successive nodes and/or dwelling time. Both values could be treated as approximate known values. A possible direction of future research regarding the problem is incorporation of these new sources of uncertainty into a model by using the Fuzzy Sets Theory.

The Fuzzy Ant System has been tested on two examples. In order to prove its suitability to provide solutions for combinatorial optimization problems where one or more of the independent variables are characterized by uncertainty, the approach should be tested on a variety of different problems.

Literature

1. Aarts, E., Korst, J., (1989) Simulated Annealing and Boltzmann Machines, John Wiley & Sons.
2. Aarts, E., Laarhoven, P. J. M., (1985) Statistical cooling: A general approach to combinatorial optimization problems, Philips Journal of Research, 40, 193-226.
3. Agapie, A., (1997) Genetic Algorithms: Minimal Conditions for Convergence, Lecture Notes in Computer Science, 1363, 183-193.
4. Back, T., Hammel, U., Schwefel, H-P., (1997) Evolutionary Computation: Comments on the History and Current State, *IEEE Transactions on Evolutionary Computation*, 1, 1, 3-17.
5. Banschbach, V.S., Waddington, K.D., (1994) Risk-sensitive Foraging in Honey Bees: No Consensus Among Individuals and No Effect of Colony Honey Stores, *Animal Behavior*, 47, 933-941.
6. Biesmeijer, J.C., van Nieuwstadt, M.G.L., Lukacs, S., Sommeijer, M.J., (1998) The Role of Internal and External Information in Foraging Decisions of *Melipona* Workers (Hymenoptera: Meliponinae), *Behavior Ecology Sociobiology*, 42, 107-116.
7. Boettcher, S., Percus, A., (2000) Nature's way of optimizing, *Artificial Intelligence*, 119, 275-286.
8. Bonabeau, E., Dorigo, M., Theraulaz, G., (1999) *Swarm Intelligence, From Natural to Artificial Systems*, Oxford University Press.
9. Budenbender, K., Grunert, T., Sebastian, H-J., (2000) A Hybrid Tabu Search/Branch-and-Bound Algorithm for the Direct Flight Network Design Problem, *Transportation Science*, 34, 4, 364-380.
10. Bullnheimer, B., Hartl, R. F., Strauss, C., (1999) A new rank-based version of the Ant System: A computational study, *Central European Journal for Operations Research and Economics*, 7, 1, 25-38.
11. Bullnheimer, B., Kotsis, G., Strauss, C., (1997) Parallelization strategies for the ant system, Technical report No. 8, October 1997, Institute of Management Science, University of Vienna, Austria.
12. Burke, E. K., Smith, A. J., (2000) Hybrid Evolutionary Techniques for the Maintenance Scheduling Problem, *IEEE Transactions on Power Systems*, 15, 1, 122-128
13. Calegari, P., Guidec, F., Kuonen, P., Kobler, D., (1997) Parallel Island-Based Genetic Algorithm for Radio Network Design, *Journal of Parallel and Distributed Computing*, 47, 86-90.

14. Camazine, S., Sneyd, J., (1991) A Model of Collective Nectar Source by Honey Bees: Self-organization Through Simple Rules, *Journal of Theoretical Biology*, 149, 547-571.

15. Cantu-Paz, E., Goldberg, D.E., (2000) Efficient Parallel Genetic Algorithms: Theory and Practice, Computer methods in Applied Mechanics and Engineering, 186, 221-238.

16. Cherny, V., (1985) Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm, *Journal of Optimization Theory and Applications*, 45, 41-51.

17. Chittka, L., Gumbert, A., Kunze, J., (1997) Foraging Dynamics of Bumble Bees: Correlates of Movements Within and Between Plant Species, *Behavioral Ecology*, 8, 239-249.

18. Chittka, L., Thompson, J.D., (1997) Sensori-motor Learning and its Relevance for Task Specialization in Bumble Bees, *Behaviour Ecology Sociobiology*, 41, 385-398.

19. Collevatti, R.G., Campos, L.A.O., Schoereder, J.H., (1997) Foraging Behaviour of Bee Pollinators on the Tropical Weed *Triumfetta semitriloba*: Departure Rules from Flower Patches, *Insectes Sociaux*, 44, 345-352.

20. Colorni, A., Dorigo, M., Maffioli F., Maniezzo, V., Righini, G., Trubian, M., (1996) Heuristics from nature for hard combinatorial optimization problems, *International Transactions in Operational Research*, 3, 1, 1-21.

21. Colorni, A., Dorigo, M., Maniezzo V., (1991) Distributed Optimization by Ant Colonies, In Proceedings First Europ. Conference on Artificial Life, Edited by F. Varela and P. Bourgine, Cambridge, MA: MIT Press, 134-142.

22. Colorni, A., Dorigo, M., Maniezzo V., (1992) An Investigation of Some Properties of An Ant Algorithm, In Proceedings 1992 Parallel Problem Solving from Nature Conference, Edited by R. Manner and B. Manderick, Amsterdam: Elsevier, 509-520.

23. Dammeyer, F., Voβ, S., (1993) Dynamic tabu list management using the reverse elimination method, *Annals of Operations Research*, 41, 31-46.

24. Dantzig, G. B., Fulkerson, D. R., Johnson, S. M., (1954) Solution of a large-scale traveling-salesman problem, Operations Research, 2, 393-410.

25. Deneubourg, J. L., Aron, S., Goss, S., Pasteels, J. M., (1990) The self-organizing exploratory pattern of the argentine ant, *Journal of Insect Behavior*, 3, 159-168.

26. deVries, H., Biesmeijer, J.C., (1998) Modelling collective foraging by means of individual behaviour rules in honey-bees, Behavioral Ecology and Sociobiology, 44, 2, 109-124.

27. Dorigo, M., Di Caro, G., Gambardella, L.M., (1999) Ant Algorithms for Discrete Optimization, Artificial Life, 5, 3, 137-172.

28. Dorigo, M., Gambardella L.M., (1997a) Ant colonies for the traveling salesman problem, *BioSystems*, 43, 73-81.

29. Dorigo, M., Gambardella L.M., (1997b) Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, *IEEE Transactions on Evolutionary Computation*, 1, 1, 53 – 66.

30. Dorigo, M., Gambardella, L.M., (1996a) A study of some properties of Ant-Q, In Proceedings of PPSN-IV, Fourth International Conference on Parallel Problem Solving from Nature, Berlin: Springer – Verlag, 656 – 665.

31. Dorigo, M., Maniezzo, V., Colorni, A., (1996) Ant System: Optimization by a Colony of Cooperating Agents, *IEEE Transactions on Systems, man, and Cybernetics - part B*, 26, 1, 29-41.

32. Dror, M., (1993) Modeling Vehicle Routing with Uncertain Demands as a Stochastic Program: Properties of the Corresponding Solution, European Journal of Operational Research, 64, 432-441.

33. Dror, M., Laporte, G., Louveaux, F., (1993) Vehicle Routing with Stochastic Demands and Restricted Failures, Operations Research, 37, 273-283.

34. Dror, M., Laporte, G., Trudeau, P., (1989) Vehicle Routing with Stochastic Demands: Properties and Solution Frameworks, Transportation Science, 23, 166-176.

35. Dror, M., Trudeau, P., (1986) Stochastic Vehicle Routing with Modified Savings Algorithm, European Journal of Operational Research, 23, 228-235.

36. Dukas, R., Real, L.A., (1991) Learning Foraging by Bees: a Comparison Between Social and Solitary Species, *Animal Behaviour*, 42, 269-276.

37. Dukas, R., Visscher, P.K., (1994) Lifetime Learning by Foraging Honey Bees, *Animal Behavior*, 48, 1007-1012.

38. Eglese, R. W., (1990) Simulated annealing: A tool for Operational Research, *European Journal of Operational Research*, 46, 271-281.

39. Eiben, A.E., Aarts, E.H.L., Vanhee, K.M., (1991) Global Convergence of Genetic Algorithms – A Markov-Chain Analysis, *Lecture Notes in Computer Science*, 496, 4-12.

40. Faigle, U., Kern, W., (1992) Some convergence results for probabilistic tabu search, *ORSA Journal on Computing*, 4, 32-37.

41. Fiechter, C.N., (1994) A parallel tabu search algorithm for large traveling salesman problems, *Discrete Applied Mathematics*, 51, 3, 243-267.

42. Fogel, L.J., Owens, A.J., Walsh, M.J., (1966) Artificial Intelligence Through Simulated Evolution, Wiley, New York.

43. Gambardella L. M., Dorigo, M., (1995) Ant-Q: A reinforcement learning approach to the traveling salesman problem, In Proceedings of the Twelfth International Conference on Machine Learning, ML-95, Palo Alto, CA: Morgan Kaufmann, 252 – 260.

44. Gambardella L.M., Dorigo, M., (1996) Solving Symmetric and Asymmetric TSPs by Ant Colonies, In Proceedings of the IEEE Conference on Evolutionary Computation, ICEC 96, 622-627.

45. Gendreau, M., Laporte, G., Seguin, R., (1996) Stochastic Vehicle Routing, European Journal of Operational Research, 88, 3-12.

46. Glover, F., (1986) Future paths for integer programming and links to artificial intelligence, *Computers and Operation Research*, 13, 533-549.

47. Glover, F., (1993) A user's guide to tabu search, *Annals of Operations Research*, 41, 3-28.

48. Glover, F., Laguna, M., (1993) "Tabu Search," chapter in *Modern Heuristic Techniques for Combinatorial Problems*, C. Reeves, ed., Blackwell Scientific Publishing, 71-140.

49. Glover, F., (1994) Genetic Algorithms and Scatter Search: Unsuspected Potentials, *Statistics and Computing*, 4, 131-140.

50. Goldberg, D., (1989) Genetic Algorithms in Search*, Optimization and Machine Learning*. Reading, MA: Addison-Wesley.

51. Golden, B.L., Laporte, G., Taillard, E.D., (1997) An Adaptive Memory Heuristic for a Class of Vehicle Routing Problems With Minmax Objective, Computers and Operations Research, 24, 445-452.

52. Gordon, V., Whitley, D., (1993) Serial and Parallel Genetic Algorithms as Function Optimizers, *Proceedings of the Fifth International Conference on Genetic Algorithms: University of Illinois at Urbana-Champaing*, July 17-21, pp 177-183.

53. Goss, S., Aron, S, Deneubourg, J.L., Pasteel, J.M., (1989) Self-organized shortcuts in the Argentine ant, *Naturwissenschaften*, 76, 579-581.

54. Gould, J.L., (1987) Landmark Learning by Honey Bees, *Animal Behaviour*, 35, 26-34.

55. Hansen, P., (1986) The steepest ascent mildest descent heuristic for combinatorial programming, *Conf. On Numerical Methods in Combinatorial Optimization*, Capri, Italy.

56. Hansen, P., Mladenović, N., (2001) Variable neighborhood search: Principles and applications, *European Journal of Operational Research*, 130, 449-467.

57. Hasan, M., AlKhamis, T., Ali, J., (2000) A comparison between simulated annealing, genetic algorithm and tabu search methods for unconstrained quadratic Pseudo-Boolean function, *Computers & Industrial Engineering*, 38, 323-340.

58. Hertz, A., Kober, D., (2000) A framework for the description of evolutionary algorithms, *European Journal of Operational Research*, 126, 1-12.

59. Hill, P.S., Wells, P.H., Wells, H., (1997) Spontaneous Flower Constancy and Learning in Honey Bees as a Function of Colour, *Animal Behaviour*, 54, 615-627.

60. Holland, J.H., (1962) Outline for a logic theory of adaptive systems, *Journal of the ACM*, 3, 297.

61. Holland, J.H., (1975) Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, MI.

62. Kadmoon, R., Shmida, A., (1992) Departure Rules Used by Bees Foraging for Nectar: a Field Test, *Evolutionary Ecology*, 6, 142-151.

63. Kaufman, A., Gupta, M., (1985) *Introduction to Fuzzy Arithmetic*, New York, Van Nostrand Reinhold.

64. Kaufmann, A., and Gupta, M. (1988) *Fuzzy Mathematical Models in Engineering and Management Science*. New York: Elsevier Science.

65. Keasar, T., Shmida, A., Motro, U., (1996) Innate Movement Rules in Foraging Bees: Flight Distances are Affected by Recent Rewards and are Correlated with Choice of Flower Type, *Behaviour Ecology Sociobiology*, 39, 381-388.

66. King, R. L., Russ, S. H., Lambert, A. B., Reese, D.S., (2001) An artificial immune system model for intelligent agents, *Future Generation Computer Systems*, 17, 335-343.

67. Kirikpatrick, S., Gellat, L., Vecchi, M., (1983) Optimization by simulated annealing, *Science*, 220, 671-680.

68. Kosko, B., (1992) *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, Englefood Cliffs, NJ: Prentice-Hall.

69. Kosko, B., (1993) *Fuzzy Thinking*. New York: Hyperion.

70. Lambert, V., Laporte, G., Louveaux, F.V., (1993) Designing Collection Routes through Bank Branches, Computers & Operations Research, 20, 783-791.

71. Laporte, G., (1992) The Travelling Salesman Problem: An overview of exact and approximate algorithms, European Journal of Operational Research, 59, 231-247.

72. Lin, S., (1965) Computer Solutions of the Traveling Salesman Problem, Bell Systems Technology Journal, 44, 2245 – 2269.

73. Locatelli, M., (2000) Convergence of a Simulated Annealing Algorithm for Continuous Global Optimization, *Journal of Global Optimization*, 18, 219-234.

74. Mamdani E.H., Assilian S. (1975) An experiment in linguistic synthesis with a fuzzy logic controller, *International Journal of Man-Machine Studies*, 7, 1-13.

75. McCulloch, W.S., Pitts, W. (1943) A Logical Calculus of the Ideas Immanent in Nervous Activity, *Bulletin of Mathematical Biophysics*, 5, 115-133.

76. Mendel, J.M., (1995) Fuzzy Logic Systems for Engineering: A Tutorial, *Proceedings of the IEEE*, 83, 345-377.

77. Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E., (1953) Equation of state calculations by fast computing machines, *Journal of Chemical Physics*, 21, 1087-1092.

78. Onbasoglu, E., Ozdamar, L., (2001) Parallel Simulated Annealing Algorithms in Global Optimization, *Journal of Global Optimization*, 19, 27-50.

79. Pappis, C., Mamdani, E., (1977) A Fuzzy Controller for a Traffic Junction, *IEEE Transactions on Systems, Man and Cybernetics*, ASMC 7, 707-717.

80. Parker, R.G., Rardin, R. L., (1988) Discrete Optimization, Academic Press, Inc.

81. Pasteels, J. M., Deneubourg, J. L., Goss, S., (1987) Self-organization mechanism in ant societies (i): Trial recruitment to newly discovered food sources, *Experientia Supplementum*, 54, 155-175.

82. Peleg, B., Shmida, A., Ellner, S., (1992) Foraging Graphs: Constraint Rules on Matching Between Bees and Flowers in a Two-sided Pollination Market, *Journal of Theoretical Biology*, 157, 191-201.

83. Pham, D.T, and Karaboga, D., (2000) Intelligent Optimization Techniques Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks, Springer.

84. Rechenberg, I., (1965) Cybernetic solution path of an experimental problem, Royal Aircraft Establishment Transl. 1122, B.F. Toms Transl. Ministry of Aviation, Royal Aircraft Establishment, Farnborough, Hants, UK.

85. Rochat, Y., Taillard, E.D., (1995) Probabilistic Diversification and Intensification in Local Search for Vehicle Routing, *Journal of Heuristics*, 1, 147-167.

86. Seeley, T.D., (1992) The Tremble Dance of the Honey Bee: Message and Meanings, *Behavior Ecology Sociobiology*, 31, 375-383.

87. Seeley, T.D., Visscher, P.K., (1988) Assesing the Benefits of Cooperation in Honeybee Foraging: Search Costs, Forage Quality, and Competitive Ability, *Behavior Ecology Sociobiology*, 22, 229-237.

88. Smith, D. K., (1982) Network Optimisation Practice - A Computational Guide, John Wiley & Sons.

89. Steinhofel, K., Albrecht, A., Wong, C.K., (2000) Convergence analysis of simulated annealing-based algorithms solving flow shop scheduling problems, *Algorithms and Complexity*, *Lecture Notes in Computer Science*, 1767, 277-290.

90. Stützle, T., (1998) Parallelization Strategies for Ant Colony Optimization, Parallel Problem Solving from Nature – PPSN V, Lecture Notes in Computer Science, 1498, 722 – 731.

91. Stützle, T., Hoos, H., (1997) The Max-Min ant system and local search for the traveling salesman problem, In Proceedings of IEEE-ICEC-EPS'97, IEEE international Conference on Evolutionary Computation and Evolutionary Programming Conference, T. Baeck, Z. Michalewicz and X. Yao, editors, IEEE Press, 309-314.

92. Sullivan, K. A., (1999) A Convergence Analysis of Generalized Hill Climbing Algorithms, Dissertation (PhD), Industrial and Systems Engineering, Virginia Tech.

93. Teodorović, D., (1994) Invited Review: Fuzzy Sets Theory Applications in Traffic and Transportation, *European Journal of Operational Research*, 74, 379-390.

94. Teodorović, D., (1999) Fuzzy Logic Systems for Transportation Engineering: The State of the Art, *Transportation Research*, 33A, 337-364.

95. Teodorović, D., Lučić, P., (2000) Intelligent Vehicle Routing System, 2000 Intelligent Transportation Systems, Conference Proceedings, Dearborn (MI), USA, October 1-3, 482 - 487.

96. Teodorović, D., Pavković, G., (1992) A Simulated Annealing Technique Approach to the Vehicle Routing Problem in the Case of Stochastic Demand, Transportation Planning and Technology, 16, 261-273.

97. Teodorović, D., Pavković, G., (1996) The fuzzy set theory approach to the vehicle routing problem when demand at nodes is uncertain, Fuzzy Sets and Systems, 82 (3), 307-317.

98. Teodorović, D., Vukadinović, K., (1998) *Traffic Control and Transport Planning: A Fuzzy Sets and Neural Networks Approach*, Kluwer Academic Publishers.

99. Waddington, K.D., Nelson, C.M., Page, R.E.Jr., (1998) Effects of Pollen Quality and Genotype on the Dance of Foraging Honey Bees, *Animal Behaviour*, 56, 35-39.

100. Williams, N.M., Thompson, J.D., (1998) Trapline Foraging by Bumble Bees: III. Temporal Patterns of Visitation and Foraging Success at Single Plants, *Behavioral Ecology*, 9, 612-621.

101. Wolpert, D.H., Macready, W.G., (1997) No free lunch theorems for optimization, *IEEE Transactions on Evolutionary Computation*, 1, 1, 67-82.

102. Yang, W-H., Mathur, K., Ballou, R.H., (2000) Stochastic Vehicle Routing Problem with Restocking, Transportation Science, 34, 99-112.

103. Youssef, H., Sait, S.M., Adiche, H., (2001) Evolutionary algorithms, simulated annealing and tabu search: a comparative study, *Engineering Application of Artificial Intelligence*, 14, 167-181.

104. Zadeh, L (1965) Fuzzy Sets, *Information and Control*, 8, 338-353.

105. Zimmermann, H. J., (1991) *Fuzzy Set Theory and Its Applications*, Boston: Kluwer.

VITA

Panta Lučić was born in Belgrade, Yugoslavia on June 7th of 1968. In December 1992 he received his Bachelor of Science degree in Transportation Engineering from the Faculty of Transport and Traffic Engineering, University of Belgrade. He joined the same institution in 1994 and completed his MS degree in 1996.

Panta Lučić has working experience in academia (University of Belgrade) and in industry (Yugoslav Airlines).

In August 1999, he enrolled in the Ph.D. program at the Department of Civil and Environmental Engineering at Virginia Polytechnic Institute and State University. He worked as a teaching assistant and research assistant mostly on NSF granted project titled "Swarm Intelligence Applications in Transportation."