An evolutionary approach for the optimization of production-distribution network design

David leonardo Puerta jaramillo
Computer and Systems Engineer
Code: 2702340

Universidad Nacional de Colombia
Facultad de Ingeniería
Departamento de Ingeniería de Sistemas e Industrial
Bogotá, D.C.
Noviembre de 2016

An evolutionary approach for the optimization of production-distribution network design

David leonardo Puerta jaramillo
Computer and Systems Engineer
Code: 2702340

Thesis Work to Obtain the Degree of
Magister in Computer and Systems Engineering

Advisor
Jonatan Gomez Perdomo, Ph.D.
Ph.D. in Mathematics, Computer Science Concentration

Coadvisor
Gustavo Alfredo Bula, M.Sc.
Magister in Industrial Engineering

Research line
Evolutionary Algorithms

Research group
Grupo de Investigación en Vida Artificial - ALIFE

Universidad Nacional de Colombia
Facultad de Ingeniería
Departamento de Ingeniería de Sistemas e Industrial
Bogotá, D.C.
Noviembre de 2016

Title in English

An evolutionary approach for the optimization of production-distribution network design

Título en español

Un enfoque evolutivo para la optimización del diseño de redes de producción-distribución

Abstract: In this Thesis an evolutionary technique for finding (near) optimal solutions to the two-stage fixed charge transportation problem (finding minimum cost transportation configurations when considering per unit transportation cost, fixed charges associated to routes, limited capacity of production plants and unlimited capacity of distribution centers) is proposed. Basically, the Hybrid Adaptive Evolutionary Algorithm with three different domain specific genetic operators (one crossover: network; two mutations: distribution and production) is applied. Here a candidate solution is encoded using two matrices for each stage of the network. The crossover operator exchanges the transportation plan of the second stage between two networks. The distribution mutation operator closes a randomly selected distribution center, so products, that were distributed to customers by such center, return to their plants where those came from. The mutation operator changes the distribution plan in the first stage of the network from a randomly selected production plant. After applying an operator, a balance method is used. Finally, the fitness function is the sum of transportation costs, including the unit transportation costs and the fixed cost incurred when using a route. Computational experiments carried on twenty instances of the problem that are available in the literature, show that our approach is able to find equal or better solutions compared to those reported in the literature.

Resumen: En esta Tesis se presenta una técnica evolutiva para la búsqueda de soluciones optimas o cercanas al oprimo del problema de transporte de cargo fijo en red de dos etapas (encontrar configuraciones de distribución de menor costo teniendo en cuenta costo de transporte por unidad de producto, costo fijo por el uso de rutas, capacidad limitada de plantas de producción y capacidad ilimitada de centros de distribución). Para encontrar dichas soluciones, el Algoritmo Evolutivo Hibrido Adaptativo es usado con tres operadores genéticos específicos al problema (un operador de cruce y dos operadores de mutación). Aquí, una solución es codificada usando dos matrices (una por cada etapa de la red). El operador de cruce intercambia el plan de distribución de la segunda etapa entre dos redes. La mutación de distribución cierra un centro de distribución elegido de manera aleatoria, haciendo que el producto enviado hacia clientes regrese hacia las plantas de procesamiento. La mutación de producción cambia el plan de distribución de la primera etapa desde una planta de producción elegida de manera aleatoria. Después de aplicar un operador, un método de balance de red es utilizado. Finalmente, la función objetivo está definida como la sumatoria de la cantidad de producto transportado multiplicado por los costos fijos y el costo de transporte por unidad de producto. Los experimentos computacionales llevados a cabo sobre veinte instancias disponibles en la literatura, muestran que la técnica usada es capaz de encontrar buenas soluciones o mejores comparadas con las soluciones reportadas en la literatura.

Keywords: transportation problem, Fixed charge, Evolutionary algorithm, Supply chain, Optimization

Palabras clave: Problema de transporte, Cargo fijo, Algoritmo evlutivo, Cadena de suministro, Optimización

Acceptation Note

Thesis Work
Aprobado

"Meritoria o Laureada mention"

Jury
Por definir

Jury
Por definir

Jury
Por definir

Advisor
Jonatan Gomez

Coadvisor
Gustavo Bula

Bogotá, D.C., Noviembre 25 de 2016

Dedication

This Thesis is dedicated to my father and my mother for their unconditional love and support. It is also dedicated to my family who have waited so long. And especially to Ines Maria Diaz for helping me and giving me the encouragement and the strength needed in the most difficult moments.

Acknowledgements

I would like to thank to my Thesis advisors Jonatan Gomez Perdomo and Gustavo Bula for their dedication, direction and teachings. Additionally, I would like to thank the members of the ALIFE research group from Universidad Nacional de Colombia for their help and advises all these years.

Contents

List of Tables

List of Figures

Glossary

- **Crossover operators**
  Genetic operators that creates one or more individuals called ”offspring” from a combination of existing individuals identified as ”parents” selected from in the evolutionary algorithm population.
- **Evolutionary algorithm**
  Optimization technique based on the principles of Darwin’s biological evolution theory.
- **Fitness function** A fitness function is a particular type of objective function that is used to measure how close a given design solution is to achieving the set aims. The fitness function is determined by the objective function and allows modern heuristics to perform pairwise comparisons between solutions. It indicates the quality of the solution and it is based on, but not necessarily identical to the objective function. In general,, the objective function is based on the problem and model formulation, whereas the fitness function measures the quality of solutions from the perspective of modern heuristics [91, 78].
- **Fitness value** The fitness value is a measure of how good the solution represented by an individual is for the problem being considered. It is the main source for guiding the search process [36]
- **Genetic operators**
  Also called variation operators, these are techniques that changes or create new individuals from existing ones in the population, in order to find better solutions to an optimization problem.
- **Mutation operators**
  Genetic operators that makes changes on an existing individual of the population to form another.
- **Metaheuristic** Metaheuristics are solution methods that combine an interaction between local improvement procedures or heuristics, and a higher level strategies to create a process capable of escaping from local optima and perform robust search of a solution space [30].

Introduction

Supply chains encompass the companies and the business activities needed to design, construct, deliver, and consume a product or service [47]. Among the companies that are part of a supply chain are production plants, suppliers, storage centers, distribution centers, among others. In addition to these companies, customers are also part of the supply chain. Customers can be large supermarkets, neighborhood shops or people. These customers can also be called demand points. For an efficient operation of a supply chain, it is necessary to make decisions related to the underlying network configuration, the routes for product distribution, and the modes of transport through the supply chain. The supply chain management process is responsible for making these kind of decisions, decisions that not only contribute to meet customer demand in an efficient manner but also contribute to reduce costs, to increase profits and to improve the quality of service. Several different problems arise when taking such decisions, one of those is the transportation problem (proposed by Hitchcock in 1941[46]).

The classical transportation problem (TP) refers to a special class of linear programming problem where the objective is to find a way to transport homogeneous products from several sources to several destinations so that the total transportation cost can be minimized [50]. A basic assumption in the TP is that the cost of transportation is directly proportional to the number of units transported. However, this is not in accordance with reality where fixed charge may be incurred when units of product are sent from a given source to a given destination [16]. The fixed charge problem formulated by Hirsch and Dantzig [45] is an extension of TP dealing with shipping available quantities of goods to satisfy the demands at minimal total cost, on condition that any route has a fixed cost for using it, regardless the shipping amount of shipped goods, and a variable cost proportional to such quantity [88]. The fixed charge may represent costs like renting a vehicle; toll charges on a highway; landing fees at an airport; the set-up costs for machines in a manufacturing environment; and the time to locate a file in a distributed database system. In the presence of such one-time costs, the transportation problem is called the fixed charge transportation problem [16].

A variant of the fixed charge transportation problem (FCTP) is the two-stage FCTP (tsFCTP), where product is transported from plants to distribution centers, and from distribution centers to customers while minimizing the overall incurred costs [11]. Since the introduction of the TP by Hitchcock [46], many researchers have proposed solution procedures for solving this kind of problems. Exact methods, approximation algorithms, heuristics and meta-heuristics have been developed to solve distribution planning problems that are very important not only for real life industrial firms with supply chains but, are also important in the area of computer science.

Despite its similarity to a standard TP, the FCTP is significantly harder to solve due to the discontinuity in the objective function introduced by such fixed costs, meaning it is a NP-hard problem [45]. Geoffrion and Graves [31] were the first to tackle the tsFCTP. They used Benders Decomposition to find the optimal location of distribution centers between plants and customers considering fixed charges imposed for each active distribution center (DC) in a potential site. Heuristic and metaheuristic algorithms are computationally more efficient for finding near optimal solutions for this kind of problems than exact algorithms or approximation algorithms (such as Lagrangian relaxation, branch-and-bound, or adjacent extreme point search) [51]. Evolutionary approaches were first employed by Syarif, Yun and Gen [85]. The authors proposed a spanning tree-based genetic algorithm with Prüffer number representation to find solutions for the multi-stage transportation problem. A genetic algorithm (GA) was also used by Gen, Altiparmak and Lin [27] to solve an extension version of the two-stage TP which considers plant and DC capacity and a maximum number of DCs to be opened. Jawahar and Balaji [49] proposed a tsFCTP model considering unlimited capacity of DCs, limited capacity of plants, and fixed cost for each route used. They also presented twenty different instances of the model and also solutions which were obtained employing a GA and Simulated Annealing (SA) [6]. Chun and Yi [12] used a GA with matrix representation to solve a tsFCTP model that considers multiple transportation modes. Raj and Rajendran [5] proposed a GA coupled with an improvement scheme to solve the tsFCTP models presented by [27] and [49]. An Ant Colony Optimization algorithm (ACO) was proposed by Panicker, Venga and Sridharan [74] to solve the tsFCTP model proposed by Jawahar and Balaji [49]. This model was also solved by Kannan, Govidan and Soleimani [51] using an Artificial Immune System (AIS) and Sheep Flock Algorithm (SFA). Among the algorithms that tackled the tsFCTP presented by Balaji and Jawajar, the SFA presents the best solutions found. The tsFCTP with fixed charge for opening a DC presented by [27] was tackled by Calvete, Galé and Iranzo [11] using an hybrid evolutionary algorithm and chromosome representation based on DCs using a binary vector that indicates if a DC is open or closed. In this Thesis is presented a natural representation of the network as an individual using a matrix for each stage of the network. This representation, combined with genetic operators specific to the problem and the adaptation of these operators during the evolutionary process, seeks a greater exploration and exploitation of the search space to find the same or better solutions to those already present in the literature.

For this Thesis, the main goal is to propose an evolutionary approach for finding near optimal solutions for the two-stage fixed charge transportation problem. To achieve this goal, three specific objective are proposed.

- To design a two-stage network representation as an individual for the optimization by mean of evolutionary algorithms.
- To develop genetic operators and repair operators employed in the optimization process.
- To evaluate the performance of the proposed optimization technique by means of the comparison of results obtained against results found in the literature

In order to develop the appropriate network representation of a two-stage network is important to identify the existing representations (encoding and decoding schemes) used in evolutionary algorithms that have been proposed for solving this kind of problem. To achieve the second specific objective is very important not only to design and develop the genetic operators, but also to test them in order to evaluate their performance and to decide if the operators are suitable for the network representation and the optimization process. The first and second specific objectives have a great impact on the optimization carried out by the evolutionary algorithm because of the computational resources that the operators need in order to find the expected results. Comparison between the results found by the evolutionary approach proposed in this Thesis and solutions found in the literature by several algorithms are carry out in a fair manner by means of statistical calculations and graphical analysis of different aspects of the optimization and results.

The structure of this Thesis is as follows:

- Chapter 1 presents the concept and background about the transportation problem, the fixed charge transportation problem and the multi-stage fixed charge transportation problem
- Chapter 2 presents the evolutionary approach employed to find optimal and near optimal solutions for the two-stage fixed charge transportation problem. This chapter shows a description of the evolutionary algorithm, the network representation as individual in the evolutionary process and the genetic operators used.
- Chapter 3 Shows the computational experiments and comparisons of the results obtained by the proposed approach against results reported in the literature.
- Chapter 4 draws some conclusions and future work.

CHAPTER 1

Background

Since the introduction of the transportation problem (TP) by Hitchcock in 1941 [46], variants of this problem have been proposed with different characteristics and different levels of difficulty. Similarly, researchers have used and created different types of solution techniques to solve such problems. This chapter presents the literature review on the transportation problem, its variants and the different techniques used to find solutions for such problems. This kind of problems are very important not only for large firms that are based on SC but also an important problem for computer science given that it is known to be an NP-hard problem.

I	Total number of production plants ($i = 1$ to I)
J	Total number of DCs ($j = 1$ to J)
K	Total number of customers ($k = 1$ to K)
a_i	Capacity of plant i
p_i	Product shipped from plant i
e_j	Capacity of DC j
q_j	Product shipped from DC j
d_k	Demand of customer k
c_{ij}	Per unit transportation cost from plant i to DC j
c_{jk}	Per unit transportation cost from DC j to customer k
b_{ij}	Fixed cost associated with each shipment from plant i to DC j
b_{jk}	Fixed cost associated with each shipment from DC j to customer k
x_{ij}	Number of units distributed from plant i to DC j
x_{jk}	Number of units distributed from DC j to customer k
y_{ij}	Binary variable that specifies whether the product is distributed from plant i to DC j
y_{jk}	Binary variable that specifies whether the product is distributed from DC j to customer k

Table 1.1. Nomenclature used for the definition of the TP, the FCTP and the tsFCTP.

1.1 Transportation problem

As mentioned in the introduction, the classical transportation problem (TP) refers to a special class of linear programming problem where the objective is to find a way to transport homogeneous products from several sources to several destinations so that the total transportation cost can be minimized [50]. In the TP it is assumed that the total supply of sources is equal to the total demand of destination and the cost associated with each route is proportional to the number of units transported along the route [7]. The TP can be formulated as follows:

Minimize

$$Z = \sum_{j=1}^{I} \sum_{k=1}^{K} c_{jk} x_{jk} \tag{1.1}$$

Subject to

$$R_1 = \sum_{k=1}^{K} x_{jk} = q_i \qquad (\forall_j, \ j = 1 \ , \ldots , \ J) \tag{1.2}$$

$$R_2 = \sum_{j=1}^{J} x_{jk} = d_k \qquad (\forall_k, \ k = 1 \ , \ldots , \ K) \tag{1.3}$$

$$R_3 = x_{jk} \geq 0 \tag{1.4}$$

Where equation 1.1 minimizes the total transportation cost, constraint 1.2 states that the total transportation from a source node must be equal to its supply, constraint 1.3 states that the total transportation to a destination node must be equal to its demand, lastly, constraint 1.4 states that flow on each edge from source to destination must not be negative.

To solve the TP, Hitchcock used a constructive procedure that is similar to the simplex method [24]. Koopmans [58, 59] independently worked on the TP, reason why the TP is also known as the Hitchcock-Koopmans Transportation Problem [24]. Dantzig in 1951 [14] used a special form of the Simplex method for solving the TP. Ford and Fulkerson in 1956 [24] proposed a method based on a combinatorial procedure (called the Hungarian method) for solving the optimal assignment problem (a special type of transportation problem).

It was until 1991 that genetic algorithm (GA) was first proposed to solve this kind of problem with papers presented by Vignaux and Michalewicz [86] and Michalewicz et al. [68]. Vignaux and Michalewicz proposed the use of GA for solving the nonlinear TP, where the cost on a route is directly proportional to the amount transported. They used a matrix representation and investigated the balance between representation structures and genetic operators. Michalewicz et al also used matrix of real values to represent solutions in their GA for solving the nonlinear TP, where the cost on a route is not directly proportional to the amount transported and where the optimum solution doesn't need to have integer flows [68].

Gen, Ida and Li [26] proposed the use of a GA to solve the bicriteria solid transportation problem which is a special type of transportation problem that arises when there are heterogeneous modes of transportation available for shipment of product. Yang and Gen [90] employed an evolutionary algorithm for solving a bicriteria linear TP, which considers the minimization of the transportation cost and the minimization of the total deterioration of goods during transportation. Their evolutionary algorithm used an improved selection strategy and a technique, called extinction and immigration, used for the crossover op-

Transportation problem		
Year	Citation	Solution methodology
1941	[46]	Simplex method
1949	[58]	Theory
1951	[59]	Theory
1951	[14]	Simplex method
1956	[24]	Combinatorial algorithm
1994	[90]	Evolutionary algorithm
1991	[68]	Genetic Algorithm
1991	[86]	Genetic algorithm
1995	[26]	Genetic algorithm
1997	[63]	Genetic algorithm
1998	[28]	Genetic algorithm

TABLE 1.2. Solution methodologies used for the transportation problem

erator when the same chromosome is chosen to mate. Gen and Li [28] introduced the spanning tree-based GA which represents a candidate solutions as a spanning tree and Prüfer number as encoding mechanism. Gen ad Li used the proposed spanning tree-based GA for solving the bicriteria TP. Table 1.2 summarizes the literature review found for this Thesis about solution methodologies used for the TP.

1.2 Fixed charge transportation problem

Despite its similarity to a standard TP, the fixed charge transportation problem (FCTP) is significantly harder to solve due to the discontinuity in the objective function introduced by such fixed costs, meaning it is a NP-hard problem [45]. Guisewite and Pardalos [40] generalized the NP-hardness result to minimum concave cost network flows [81]. The FCTP can be formulated as follows:

Minimize

$$Z = \sum_{j=1}^{J} \sum_{k=1}^{K} (c_{jk} x_{jk} + b_{jk} y_{jk}) \tag{1.5}$$

Subject to

$$R_1 = \sum_{k=1}^{K} x_{jk} = q_j \qquad (\forall_j,\ j = 1\ , \ldots,\ J) \tag{1.6}$$

$$R_2 = \sum_{j=1}^{J} x_{jk} = d_k \qquad (\forall_k,\ k = 1\ , \ldots,\ K) \tag{1.7}$$

$$R_4 = 0 \leq x_{jk} \leq z_{jk} y_{jk} \tag{1.8}$$

$$z_{jk} = min(q_j, d_k) \tag{1.9}$$

Similar to the transportation problem the objective unction 1.5 minimizes the total transportation cost which includes the per unit transportation cost that is proportional to the amount od product transported, and the fixed charge which is not proportional to the amount of product transported. As in the Transportation Problem, constraint 1.6 states that the total transportation from a source node must be equal to its supply, constraint 1.7 states that the total transportation to a destination node must be equal to its demand. Finally, constraint 1.7 states that product transported from source j to destination k must not exceed z_{jk} which is de minimum between total supply from source node j and demand of destination k. For the FCTP it is also assumed that the total supply od sources is equal to the total demand of destination.

Since the introduction of the FCTP, different models have been proposed. These models can differ on characteristics like the fixed cost, the capacities of facilities, the number and the type of commodities being transported, transportation modes used, time consideration, among other characteristics. In the same way, many solution methodologies have been proposed to tackle this kind of problems. These methodologies can be classified in exact methods, heuristic methods and approximate methods.

Exact methods were the first kind of methods to be applied for solving the fixed charge transportation problem. Murty in 1968 [70] proposed an exact algorithm for a small instance of the FCTP based on searching among the adjacent extreme points. Gray [39] presented an alternate approach to Murty's algorithm using branch-and-bound algorithm. Steinberg in 1970 [82] presented an exact solution based upon a branch and bound approach computationally feasible for large problems. Kennington and Unger [53] used penalties for fathoming and guiding separation task of their proposed branch-and-bound procedure to solve the FCTP. Fisk and McKeown [23] presented a direct search procedure using LIFO (last in, first out) decision rule for branching to solve the pure FCTP. Also for solving the FCTP, McKeown [67] presented a branch-and-bound procedure that calculates bound separately on the sum of fixed costs and on the continuous objective values. Barr, Glover and Klingman [8] also used branch-and-bound for solving the FCTP, in which they used an augmented predecessor-thread index data structure to facilitate quick solutions to the problems that appear at each node of the tree. Schaffer [80] proposed a branch-and-bound with penalties combined with an adjacent extreme point heuristic procedure to provide a near optimal solution to the FCTPs. Pelekar [72] developed a conditional penalty for their branch-and-bound for the solving the FCTP, and claimed that it was stronger than the Lagrangian penalties employed by Cabot and Erenguc [10]. Hultberg and Cardoso [48] used a branch a bound algorithm for the teacher assignment problem, which is a special case of the FCTP. Bell, Lamar and Wallace [62] proposed a revised-modified penalties for solving the FCTP and showed that the conditional penalties proposed in [10, 72, 17], are not valid modified penalties. Bell, Lamar and Wallace [9] proposed conditional penalties, and three types of capacity improvements techniques and concave relaxations to reduce in nearly 90% the average number of branch-and-bound subproblems for solving the FCTP. Ortega and Wolsey [71] presented a branch-and-cut algorithm to solve the single-commodity uncapacitated FCTP. Cabot and Erenguc [10] used Lagrangean relaxation to improve the performance of branch-and-bound algorithm to solve the FCTP. Kim and Hooker [56] proposed a hybrid approach combining constraint programing and linear programming for the FCTP and other network flow problems. Sandrock [79] solved small instances of the FCTP using a low-tech algorithm.

Balinski [7] formulated the FCTP, explained its special properties, and proposed an approximate method that gives a lower and upper bounds for the optimal value of the instance being solved. Kuhn and Baumol [61] proposed an approximation method that uses the Ship Most at Least Cost method, called the degeneracy forcing algorithm, and compared it with the Vogels approximation method and the Ship Most at Least Cost method. Dwyer [18] proposed the use of completely reduced matrices for solving the FCTP. Cooper and Debres [13] proposed two approximation methods for solving the FCTP. Denzler [15] proposed an approximation method based on the simplex method. Robers and Cooper [77] proposed a search among the extreme points to improve the approximation method presented by Balinski. Wright and Lanzenauer [87] proposed a heuristic algorithm for solving large general fixed charge problem based on Lagrangian relaxation and cost allocation heuristics. Diaby [16] developed a heuristic procedure for the solution a generalized FCTP in which there are resource losses in addition to the fixed charges. Herer, Rosenblatt and Hefter [42] proposed a fast algorithm based on an implicit enumeration procedure to solve the single sink FCTP. Kim and Pardalos [55] proposed an approach for solving the general capacitated (or uncapacitated) fixed charge network flow problem using dynamic slope scaling procedure. Adlakha and Kowalski in 2003 [1] proposed a simple heuristic for solving the FCTP, and stated that the method is more time consuming than the algorithms for solving regular TP. Adlakha and Kowalski in 2004 [2] also proposed a simple algorithm based on the Vogel approximation method for a variation of the FCTP, called the source-induced FCTP, in which the fixed cost is incurred for every supply point that is used in the solution. Glover [34] presented a parametric approach, called Ghost Image Process, for solving fixed charge problems. Their implementation is specialized to handle the most prominently occurring types of fixed charge problems, which arise in the area of network applications. Adlakha et al [3] developed a simple, yet powerful, analytical heuristic to find a more-for-less solution for both classical TP and FCTP. Kannan et al. (2008) [52] proposed a local search heuristic, called Nedler-Mead method, for a proposed transportation model of a single-stage supply chain with fixed charge. Klose (2008) [57] proposed an algorithm based on dynamic programming and implicit enumeration for solving the single-sink FCTP. Kowalski and Lev (2008) [60] considered the step FCTP in which the fixed cost is in the form of a step function dependent on the load in a given route, and developed a simple heuristic algorithm for small instances of the problem. Raj and Rajendran [4] proposed heuristic algorithm based on the Vogel approximation method and an improvement scheme to solve the single stage FCTP.

Sun and McKeown in 1993 proposed a Tabu search algorithm for solving the general FCTP [84]. Sun et al [83] developed a Tabu search algorithm for the FCTP using the simplex method on a graph as a local search algorithm. Yang and Liu [89] proposed a hybrid intelligent algorithm based on the fuzzy simulation technique and tabu search algorithm for solving fixed charge solid transportation problems under a fuzzy environment, in which the direct costs, the fixed charges, the supplies, the demand, and the conveyance capacities were considered as fuzzy variables. Gottlieb and Paulmann [38] introduced the use of GA for solving FCTP. They presented two GAs, one GA based o the permutation representation, and the second GA based on a direct solution encoding using matrix representation with specialized operators to maintain feasibility. They concluded that the behavior of the permutation the GA using permutation representation is inferior to the matrix representation with specialized operators. Gen and Li [29] presented a spanning tree representation in a GA for solving the bicriteria FCTP. Gottlieb and Eckert [37] compared the permutation representation and the Prüfer number representation based on results obtained from two GA used for solving the FCTP. Eckert and Gottlieb [19] presented a direct representation for a GA to solve the FCTP. The direct representation allows the GA to use problem-specific operators and restrict search to basic solutions. Jo, Li and Gen [50] employed a spanning tree-based genetic algorithm with Prüfer number representation to solve the non-linear FCTP. Hajiaghaei-Keshteli et al [41] tackled the nonlinear FCTP using a spanning tree-based GA combined with a method to design chromosomes that does not need a repairing mechanism for feasibility. Xie and Jia [88] proposed a hybrid GA based on steady-state GA, and minimum cost flow algorithm as decoder for solving the nonlinear FCTP. Lotfi and Tavakkoli-Moghaddam [65] proposed a genetic algorithm using a modified priority-based encoding for the linear and non-linear FCTPs. Tari and Zahra [33] proposed priority based genetic algorithm to solve a vehicle allocation problem involving heterogeneous fleet of vehicles for delivering products from a manufacturing firm to a set of depots. El-Sherbiny [22] presented a mutation-based artificial immune algorithm to solve the step FCTP. They stated that their algorithm guarantees the feasibility of the candidate solutions and solves both balanced and unbalanced FCTP without the use of dummy supplier or customer. They also present a coding and decoding schema for the artificial immune algorithm. El-Sherbiny and Alhamali [21] introduced a hybrid particle swarm algorithm with artificial immune learning for solving the balanced and unbalanced FCTP. Table 1.3 summarizes the literature review on the solution methodologies used for the FCTP.

Year	Citation	Methodology	Solution methodology
1968	[70]	Exact approach	Search among adjacent extreme points
1968	[39]		Branch-and-bound
1970	[82]		Branch-and-bound
1976	[53]		Branch-and-bound
1979	[23]		Branch-and-bound
1981	[67]		Branch-and-bound
1981	[8]		Branch-and-bound
1989	[80]		Branch and bound
1990	[72]		Branch-and-bound
1997	[48]		Branch-and-bound
1997	[62]		Branch-and-bound
1999	[9]		Branch-and-bound
2003	[71]		Branch-and-cut
1984	[10]		Lagrangian relaxation for Branch and bound
2001	[56]		Hybrid approach
1988	[79]		Low-tech algorithm
1961	[7]	Heuristics	Approximate procedure
1962	[61]		Ship Most at Least Cost
1966	[18]		Completely reduced matrices
1967	[13]		Two Heuristics
1969	[15]		Simplex-based approximate method
1976	[77]		Approximate method
1989	[87]		Lagrangian relaxation
1991	[16]		Successive linear approximation
1996	[42]		Implicit enumeration procedures
1999	[55]		Dynamic slope scaling procedure
2003	[1]		Simple heuristic
2004	[2]		Vogels approximation-based simple algorithm
2005	[34]		Parametric approach
2006	[3]		More-for-less algorithm
2008	[52]		Local search heuristic
2008	[57]		Dynamic programming
2008	[60]		Heuristic
2009	[4]		Vogels approximation-based Heuristic algorithm
1993	[84]	Metaheuristics	Tabu search
1998	[83]		Tabu search
2007	[89]		Hybrid intelligent algorithm
1998	[38]		Genetic algorithm
1999	[29]		Genetic algorithm
2000	[37]		Genetic algorithm
2002	[19]		Genetic algorithm
2007	[50]		Genetic algorithm
2010	[41]		Genetic algorithm
2012	[88]		Hybrid genetic algorithm
2013	[65]		Genetic algorithm
2016	[33]		Genetic algorithm
2012	[22]		Artificial Immune Algorithm
2013	[21]		PSO and Artificial Immune Algorithm

TABLE 1.3. Previous solution methodologies for the fixed charge transportation problem

1.3 The Two-stage fixed charge transportation problem

The tsFCTP considered in this paper is the one proposed by Jawahar and Balaji [49]. As mentioned earlier, the objective function of this model considers the transportation cost of a distribution plan where a commodity is transported from plants to DCs, and from DCs to customers considering a per-unit transportation cost, a fixed cost incurred whenever a route is used, unlimited capacity of DCs and limited capacities of production plants. Table 1.1 shows the nomenclature used for the mathematical formulation of the problem.

1.3.1 Assumptions

As Balaji and Jawahar proposed, the assumptions for the tsFCTP are.

- The total supply of all plants is at least equal to the total demand of customers.

$$i.e. \quad \sum_{i=1}^{I} a_i \geq \sum_{k=1}^{K} d_k \tag{1.10}$$

- The capacity of each DC is at least equal to the total demand of all customers.

$$i.e. \quad e_j \geq \sum_{k=1}^{K} d_k \quad (\forall_j, j = 1 \quad to \quad J) \tag{1.11}$$

- A shipment from plant i to DC j or from DC j to customer k is shipped as one lot irrespective of the number of units shipped.

It is also important to mention that, in this model, a single commodity is considered; each customer can receive product from more than one distribution center; each DC can receive product quantities from any production plant; the number of production plants, DCs and customers are known; customer demand is met completely; damage or loss of product is not considered.

1.3.2 Mathematical model

Minimize

$$Z = \sum_{i=1}^{I}\sum_{j=1}^{J}(c_{ij}x_{ij} + b_{ij}w_{ij}) + \sum_{j=1}^{J}\sum_{k=1}^{K}(c_{jk}x_{jk} + b_{jk}w_{jk}) \tag{1.12}$$

Subject to

$$\sum_{j=1}^{J} x_{ij} \leq a_i \qquad (\forall_i, \ i = 1 \ to \ I) \tag{1.13}$$

$$\sum_{j=1}^{J} y_{jk} = d_k \qquad (\forall_k, \ k = 1 \ to \ K) \tag{1.14}$$

$$\sum_{i=1}^{I} x_{ij} = \sum_{k=1}^{K} y_{jk} \qquad (\forall_j,\ j = 1\ to\ J) \tag{1.15}$$

$$x_{ij} \geq 0 \text{ and integers}$$

$$x_{jk} \geq 0 \text{ and integers}$$

The objective function 1.12 considers the total transportation cost which includes the per-unit transportation cost and the fixed cost of each route used. Equation 1.13 refers to the plant capacity constraint which needs to be at most equal to the quantity shipped to DCs. Equation 1.14 is the demand constraint that refers to the quantity of product shipped from DC j which must be equal to de demand of customer k. It can also be considered the equation 1.15 which refers to flow conservation of the system and implies that the amount of product transported in the first stage of the network, must be equal to the amount of product transported in the second stage of the network.

1.3.3 Previous works

Geoffrion and Graves [31] proposed Benders Decomposition to solve the tsFCTP considering multiple commodities, capacitated plants, and fixed charges imposed for each DC opened in a potential site. Marin and Pelegrín [66] used Lagrangean decomposition to solve the tsFCTP considering fixed cost for the installation of transshipment points. Hindi, Basta and Pieńkosz[43] addressed a two-stage distribution-planning problem. The authors gave mathematical model for the problem and developed a branch and bound algorithm to solve it. They considered a fixed cost for opening a DC as well as an operating cost and a maximum capacity.

Syarif, Yun and Gen [85] used a spanning tree-based GA, which uses Prüfer number representation for solving the multi-stage transportation problem, where the objective is to find the minimum transportation cost given a maximum number of facilities (plants and DCs) to be opened. Gen, Altiparmak and Lin [27] developed priority-based Genetic Algorithm which uses an encoding method based on the priority-based encoding presented by Gen and Cheng [25], with new decoding and encoding procedures used to adapt to the characteristic of tsFCTP. Ekşioğlu, Romejin and Pardalos [20] proposed a primal-dual algorithm and a linear programming relaxation for an integrated production and transportation planning problem in a two-stage supply chain modeled as a network flow problem with fixed charges. Keskin and Üster[54] proposed meta-heuristic procedures, including a population-based scatter search with path relinking and trajectory-based local and tabu search, for solving multi-product two-stage production/distribution system design problem. Yun, Moon and Kim [92] proposed a hybrid genetic algorithm with adaptive local search for solving the multi-stage supply chain problem. Chun and Yi [12] proposed a genetic algorithm for a two-stage FCTP with multiple transportation modes selection.

Jawhar and Balaji [49] presented a genetic algorithm with matrix representation for solving the tsFCTP. They proposed a set of twenty instances of the tsFCTP considering per unit transportation cost, fixed cost associated with each route, uncapacitated DCs and limited capacity for plants. Balaji and Jawahar [6] developed a simulated annealing algorithm to solve the tsFCTP model considered in [49]. Panicker and Sridharan [73]

Year	Citation	Solution methodology
1974	[31]	Benders decomposition
1997	[66]	Lagrangian relaxation
1998	[43]	Branch and bound
2006	[20]	Primal dual algorithm and linear relaxation
2012	[76]	Nearest neighbor algorithm
2014	[75]	Nearest neighbor algorithm and local search
2007	[54]	Population-based scatter search and tabu search
2010	[6]	Simulated annealing
2002	[85]	Genetic algorithm
2006	[27]	Genetic algorithm
2009	[92]	Genetic algorithm
2009	[12]	Genetic algorithm
2009	[49]	Genetic algorithm
2012	[73]	Genetic algorithm and ant colony optimization
2012	[5]	Genetic algorithm
2011	[69]	Artificial immune system
2012	[74]	Ant colony optimization
2014	[51]	Artificial immune system and Sheep flock algorithm
2015	[11]	Hybrid evolutionary algorithm

Table 1.4. Literature review on solution methods used for the fixed charge transportation problem on networks of two or more stages.

compared a GA and an Ant Colony Optimization (ACO) algorithm for solving the tsFCTP, and conclude that ACO-based heuristic results were better than those obtained using GA in terms of total cost and computation time.

Molla-Alizadeh-Zavardehi et al [69] proposed an Artificial Immune Algorithm and a spanning tree-based GA using Prüfer number representation for solving the tsFCTP. Pintea et al [76] investigated hybrid variants of the Nearest Neighbor search algorithm based on different probabilities and tested on large scale instances of the capacitated tsFCTP. Raj and Rajendran [5] proposed genetic algorithm using matrix representation for solving the tsFCTP models proposed Jawahar and Balaji [49] and Gen, Altiparmak and Lin [27]. Panicker, Venga and Sridharan [74] proposed an ACO algorithm to solve the tsFCTP model considered by Jawahar and Balaji [49]. Kannan, Govidan and Soleimani [51] proposed an artificial immune system (AIS) and a sheep flock algorithm (SFA) to find good solutions for the tsFCTP model proposed by Jawahar and Balaji [49]. Pintea and Pop [75] proposed a improved hybrid algorithm combining the Nearest Neighbor heuristic with a powerful local search procedure for solving the capacitated tsFCTP. Calvete, Galé and Iranzo [11] developed a hybrid evolutionary algorithm that combines a chromosome representation based on DCs activity and minimum cost network flow problem to associate a feasible solution to each chromosome for solving the tsFCTP. Table 1.4 summarizes the literature review that was carried on this Thesis on the two-stage fixed charge transportation problem.

CHAPTER 2

Proposed evolutionary approach

Several metaheuristic techniques have been used to find good solutions to the problem of transport and its variants (Chapter 1 of this Thesis). Genetic Algorithms [28, 33, 85], Tabu search [84, 83] and Artificial Immune System [22, 21, 69, 51] among other, are some examples of metaheuristics used to tackle this kind of problems. The two-stage fixed charge transportation problem considered for this Thesis have also been considered by various researchers. It was proposed by Jawahar and Balaji in 2009 [49]. They used A genetic algorithm and Simulated annealing [6]. Raj and Rajendran also used a genetic algorithm to find solution to this model [5]. Ant Colony Optimization was employed by Panicker, Venga and Sridharan [74]. Kannan, Govidan and Soleimani found solutions using Artificial Immune System and a Sheep Flock Algorithm [51]. Given the different techniques and the quality of the solutions found, in this Thesis, the Hybrid Adaptive Evolutionary Algorithm (HaEa) proposed by Gomez [35] is used to find near optimal solutions for the two-stage fixed charge transportation problem proposed by Jawahar and Balaji [49].

One of the most important features of HaEa is to allow competition among genetic operators through operator rates, which are adjusted depending on the performance of the operators when applied. To represent a candidate solution in the evolutionary process, a direct representation is used, where the individual serve as a two stage network were each stage is represented with a matrix containing the flow of each edge of the network. This representation allows the evolutionary algorithm to apply the proposed problem-specific operators and carry out the fitness evaluation in a straightforward way. For the optimization process, three genetic operators are proposed (one crossover operator and two mutation operators). The crossover operator exchanges the distribution plan of the second stage of each network. As a result, the offspring inherit the distribution plan for the first stage of one network and the distribution plan for the second stage of the other network. One of the mutation operators, called Distribution mutation, randomly selects an active DC and closes it, returning distributed product from customers to plants. The mutation operator called Production mutation randomly changes the original distribution plan from a randomly selected plant to the existing DCs. Finally, the fitness function is the sum of transportation costs, including the unit transportation costs and the fixed cost incurred when using a route. This chapter describes the evolutionary algorithm, the network representation as individual and the proposed genetic operators.

2.1 HAEA

One important feature of HaEa is that it is capable of using as many genetic operators as required during the evolutionary process. HaEa encodes each individual together with an operator rate for each genetic operator. The operator rates adapt during the evolutionary strategy using a randomized learning rule that update each operator rate depending on its performance. the operator rate is rewarded if a child, produced by the genetic operator, has better fitness than the parent or it is punished otherwise. Algorithm 1 presents the pseudo-code of the evolutionary algorithm. In each generation, every individual selects only one operator according to the operator rates encoded into the individual (line 8). When a non-unary operator is applied, additional parents (the individual being evolved is considered a parent) are chosen according to any selection strategy (Line 9). Among the offspring produced by the genetic operator, only one individual is chosen as child (line 11). The individual (parent or offspring) that has the highest fitness is chosen to be part of the population in the next generation.

Algorithm 1 Hybrid Adaptive Evolutionary Algorithm (HaEa)

HAEA(λ, terminationCondition)

1: $t_0 = 0$
2: $P_0 =$ initPopulation(λ),
3: **while** (terminationCondition(t, P_t) is false) **do**
4:     $P_{t+1} =$
5:     **for** each ind $\in P_t$ **do**
6:         rates = extract_rates(ind)
7:         δ = random(0,1)   // *learning rate*
8:         oper = Op_Select(operators,rates)
9:         parents = ParentSelection(P_t,ind)
10:         offspring = apply(oper,parents)
11:         child = Best(offspring,ind)
12:         **if** (fitness(child) $>$ fitness(ind)) **then**
13:             rates[oper] = (1.0 + δ)*rates[oper]   *//reward*
14:         **else**
15:             rates[oper] = (1.0 - δ)*rates[oper]   *//punish*
16:         **end if**
17:         normalize_rates(rates)
18:         set_rates(child, rates)
19:         $P_{t+1} = P_{t+1} \cup$ {child}
20:     **end for**
21:     $t = t + 1$
22: **end while**

2.2 Chromosome representation

Different representations of individuals and encoding/decoding mechanisms have been employed in evolutionary algorithms to find good solutions to the transportation problem and the FCTP. Michalewicz, Vignauz and Hobbs [68, 86] were the first to use GA for solving the linear and non-linear TP. They stated that bit strings are not well suited to represent solutions to solve the TP, they instead introduced the matrix representation were each chromosome is represented using a $m \times n$ matrix with $m + n - 1$ positive elements, and permutation representation where an individual's genotype is a permutation of 1,...,m*n where m represents the source nodes and n represents the destination nodes [86]. Gottlieb and Eckert [37] compared the Prüfer number representation and the permutation representation. They stated that permutation-based GA, exhibits better performance than the GA that uses Prüfer number. They also stated that matrix representation is the most natural one for the TP, and allows to calculate fitness in a straightforward way. A direct transportation tree representation was presented by Eckert and Gottlieb, where a basic solution has at most (m+n-1) positive entries in (x_{ij}), and the positive entries form a tree in the transportation graph, or a forest if there are less than (m+n-1) non-zero entries [19]. They stated that the main advantage of the representation is that restricts search to basic solutions and allows using problem-specific variation operators. Authors compared the direct representation against Prüfer number, permutation, and matrix representation. Results showed that their approach clearly outperformed previous techniques.

Gen and Li [28] proposed the spanning tree-based GA using the Prüfer number encoding, which, over the years, was one of the most popular encoding procedure. In TP, with m origins and n destinations, Prüfer number encoding needs a string length of (m+n-2) to represent a chromosome [28]. The drawback of the encoding is that, it may produce infeasible individuals after genetic operators are applied, reason why, repair mechanism are needed. Permutation representation ensures that there are at most (m+n-1) basic cells in the offspring after crossover operation is applied, but it needs a string length of (m+n-2) to completely represent a solution [5]. Another encoding procedure is the Priority-based encoding developed by Gen, Altiparmak and Lin [27] to scape from repair mechanisms required in spanning tree-based representation using Prüfer number. For the TP, a chromosome consists of priorities of sources and depots to obtain the transportation tree, and the length is equal to the total number of sources and depots, i.e (m+n) [27]. Lofti and Tavakkoli-Moghaddam [65] proposed a priority-based GA with special encoding/decoding procedures for transportation trees that do not create unfeasible chromosomes. stated that direct transportation tree representation needs an adaptive mechanism to drive the population away from local optima that dominates the search for many subsequent generations.

Other representations have been used in evolutionary algorithms for the solution of transportation problems. For instance, Liu et al [64] proposed a representation where each chromosome was corresponded a basic feasible solution, which includes m+n-1 basic variables, where each variable represents the transported amount of product from one source to one depot [65]. Calvete, Galé and Iranzo [11] proposed a binary vector representation used in a hybrid EA for the tsFCTP. In this representation, each chromosome is a J-dimensional vector where J is the number of depots in the network. The same tsFCTP was also considered previously in [27, 5]. Gen, ALtiparmak and Lin priority-based encoding, whereas Raj and Rajendran employed permutation-based encoding. As Calvete, Galé and Irazno pointed out, [27, 5] both developed a GA based on sequentially obtaining the transportation trees of each stage of the network. They first obtain the transportation tree for the second stage, then they obtain the transportation tree for the distribution plan of the first stage of the network. Calvete, Galé and Irazno also stated that they only explore feasible solutions of the problem, which are formed by the union of two transportation trees, which may result in both algorithms being unable to obtain a optimal solution, even for a very simple instance [11].

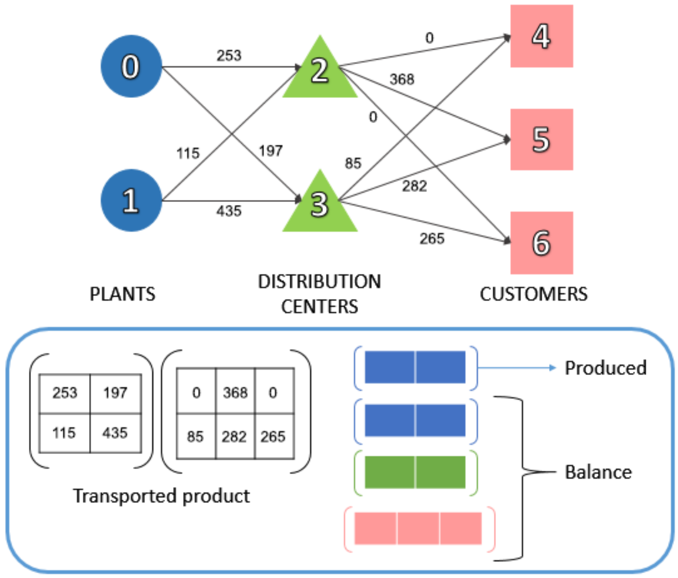


FIGURE 2.1. Example of a two stage network representation by HAEA.

In this work, an candidate solution is represented as a two stage network using two matrices, one for each stage of the network. The first stage is a matrix of $I \times J$ which stores the transportation of product from production plants to DCs. The second stage of the network is represented using a matrix of $J \times K$ which stores the product transported from each DC to each customer. In addition to the matrices that stores the transported product, four arrays are used. One array stores the quantity produced by each plant and three arrays (one for each node type) are used to control the balance of each plant, DC or customer of the network. The use of this representation (shown in Figure 2.1) allows HAEA to apply three (domain specific) genetic operators and carry out the fitness evaluation in a straightforward way. As noted above, with this representation, HAEA does not need encoding or decoding mechanisms, and the operators do not generate infeasible solutions. This is possible thanks to the mechanism of each operator and the balance procedure that is performed when an operator is applied.

2.3 Random distribution algorithm

A random distribution algorithm is presented (Algorithm 2), which makes a random selection of edges to send certain amount of product in quantities that are also randomly generated. The procedure generates product quantities ranging from 1 to Q, where Q is the total amount of product available. In addition to sending small random amounts of product, the algorithm makes a random selection of edges that are used to send such amounts. To achieve this, the algorithm creates an array of length equal to the number of edges available for shipping product and where each location contains the random selection of one of the edges. While the total amount of product is not sent, iteratively, an edge is selected and a random amount of product is sent to the destination node. The random distribution algorithm is used in the random assignment of flow in the population initialization and to balance a network after a genetic operator is applied.

Algorithm 2 Random distribution algorithm

RandomDistribution(vertex, quantity, balance)

```
Q = quantity
randomQuantity = 0
edges = getAvailableEdges()
while quantity != 0 do
    if edges.length == 1 then
        sendProduct(vertex, edges.get(0), quantity)
        quantity = 0
    else
        randomEdges = getRandomEdges()
        for each edge ∈ randomEdges do
            randomQuantity = random(1,Q)
            if quantity < randomQuantity then
                randomQuantity = quantity
            end if
            sendProduct(vertex, edge, randomQuantity)
            quantity -= randomQuantity
            if balance and edge.to.balance == 0 then
                edges = getAvailableEdges()
                break
            end if
        end for
    end if
end while
```

2.4 Initial population

As described In this work, a candidate solution is represented using a matrix for each stage of the network. The creation of a candidate solution for the initial population is carried out in a similar manner to the work presented in [49, 5] in that the Least Cost Method (LCM), Vogel's Approximation Method (VAM) and a random allocation are used. First of all, HaEa uses the random distribution algorithm to allocate product to each plant which serves as the quantity produced (equal to the total demand of customers) taking into account the capacity of each production plant. Next, HaEa randomly selects (from LCM, VAM and a random allocation) the method used to complete the creation of the individual. If HaEa selects VAM or LCM, it also selects the costs (between per-unit transportation cost and fixed cost) to be used. Lastly, if HaEa selects the random allocation, then the candidate solution is generated using the random transportation algorithm.

2.5 Genetic operators

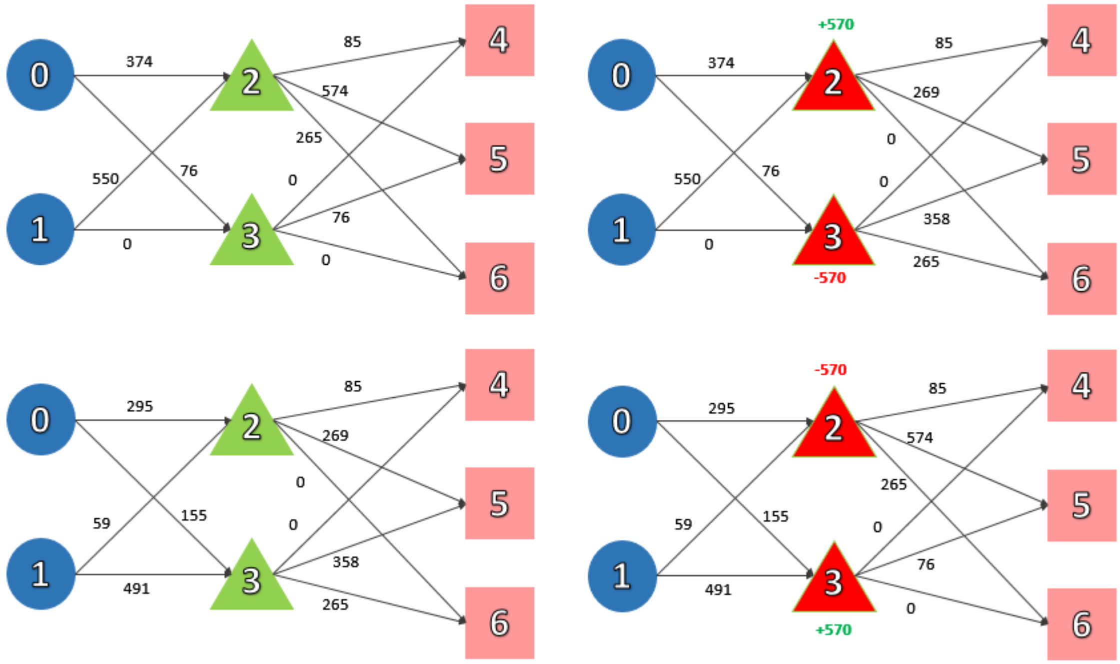
A crossover operator and two mutation operators were created to be applied and compete in the evolutionary process of HaEa. It is noteworthy that, after applying a genetic operator, the network must go through a process of balance, in which the random distribution algorithm is used. In the crossover operator, networks that result from applying the operator inherit the transportation plan of the first stage of a parent, and the distribution plan of the second stage of the other parent. Proposed mutation operators are applied depending on the mutation rate that randomly chooses a node of the network, which is modified according to the type of mutation applied.

2.5.1 Crossover operator

The crossover operator swaps the flow of the second stage of each network as it can be noticed in Figure 2.2b. The offspring's inherit the flow of product from DCs to customers from one of the parents and the flow of product from plants to DCs of the other parent. Changes of the flow in the second stage causes an imbalance in DCs, where one or more may end up with negative balance (outbound greater than inbound flow), or positive balance (inbound greater than outbound flow). To balance the network, product is transported from positive balanced DCs to negative balanced DCs using one or more production plants. This is accomplished by using the random transportation algorithm, and starts by iterating over the existing DCs. If the actual DC has a positive balance, the random distribution algorithm is employed to send product to plants using edges with positive flow (Figure 2.2c). The balance of the network is completed after the product is transported from plants with positive balance, to negative balanced DCs (Figure 2.2c-d).

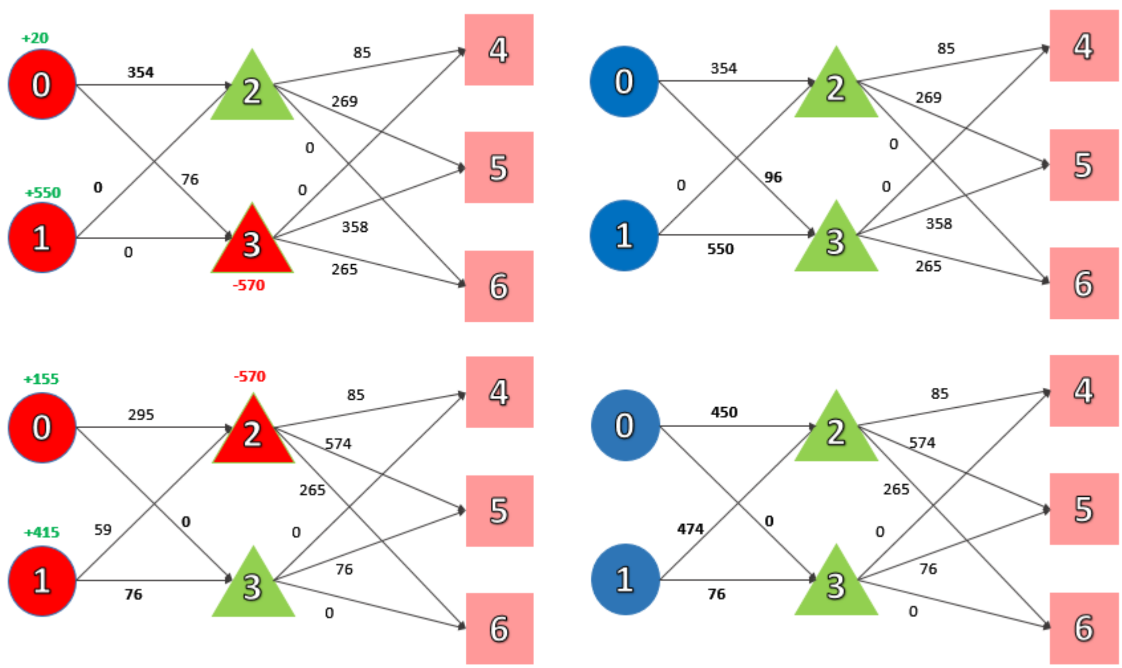
2.5.2 Production mutation

For both mutation (production, and distribution) operators, a mutation probability of $1/n$ is employed, where n depends on the number of vertices (plant or DCs) of the network. Similar to the well-known bitrate mutation, one plant is randomly selected and modified accordingly. In the case of production mutation, using the random transportation algorithm, the operator changes the outbound of product from the selected plant to all the



(a) Parents.

(b) Second stage swap.

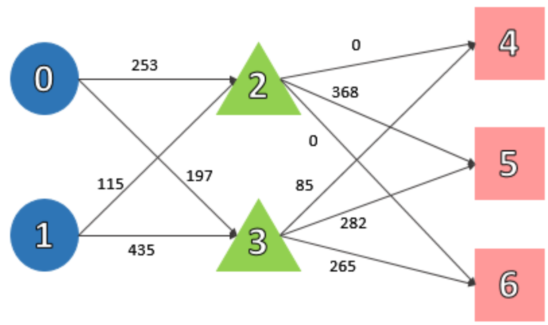


(c) Product is returned from positive DC to plants.

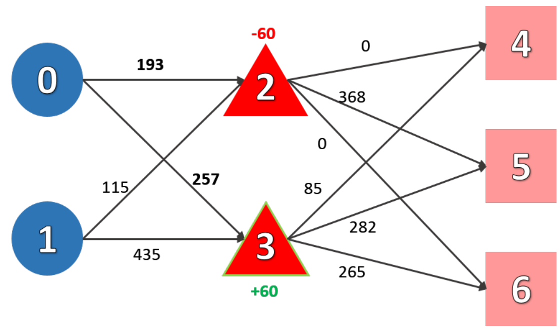
(d) Balance of childs.

Figure 2.2. Example of crossover operator.

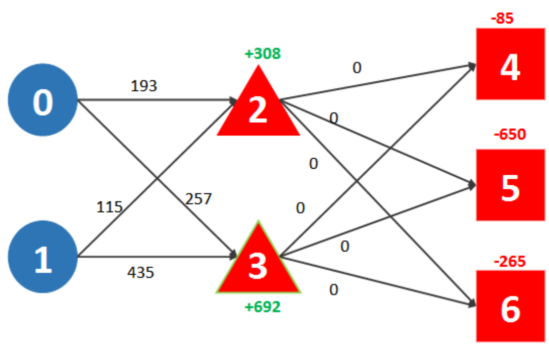
DCs allocating quantities to each route without exceeding plant capacity. As the crossover operator, the mutation of the selected plant causes an imbalance in all the DCs. To balance the DCs and the whole network, it is necessary to compute a new distribution plan for the second stage of the network. To achieve this, as shown in Figure 2.3, first the distribution plan of the second stage is erased (Figure 2.3c), next, the random transportation algorithm is applied from every DC to transport the new inbound to every customer completing their demand and balancing the network.



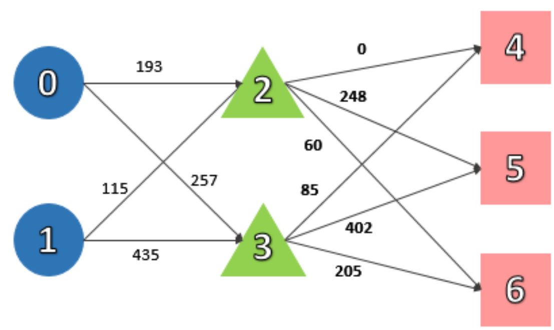
(a) Original network before production mutation is applied.



(b) A new distribution from plant 0 is calculated.



(c) The total amount of product from customers is returned to the DCs.

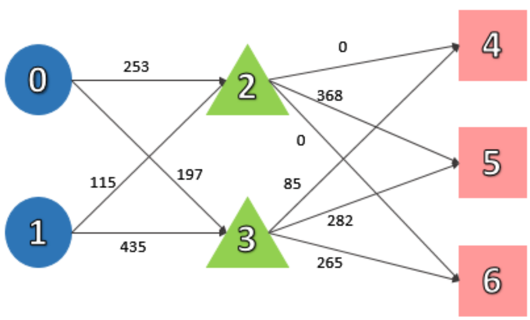


(d) New distribution plan for the second stage using the random distribution algorithm.

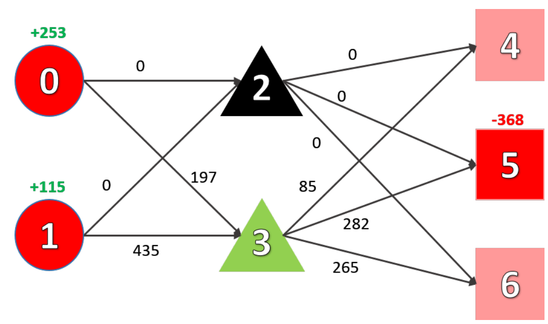
FIGURE 2.3. Example of Production Mutation.

2.5.3 Distribution mutation

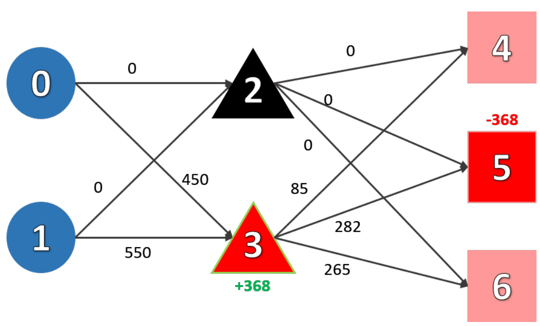
In the case of the distribution mutation, one DC is randomly selected with a mutation probability of $1/J$. If there exist two or more active DCs, the selected DC is closed, and product is returned from customers to plants. Consequently, customers are left with incomplete demand, and plants with positive balance (Figure 2.4b). To balance the network, the operator iterates over the network plants. In this iteration, if the current plant has product in stock, product is redistributed to active DCs using the random distribution algorithm. After the product is distributed to active DCs, the random distribution procedure is used again to complete customers demand from DCs (Figure 2.4d).



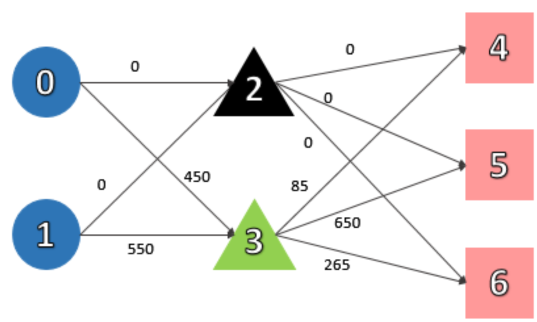
(a) Original network before a randmoly selected DC is closed.



(b) DC number 2 is closed and product is returned from clients to original plants.



(c) Product from plants is distributed to DC that are active.



(d) Product prom DC 3 is distributed to customer 5 to complete demand and balance the network.

FIGURE 2.4. Example of a Distribution mutation.

CHAPTER 3

Results and analysis

To evaluate the performance of our technique, HaEa is tested in twenty instances proposed by Jawahar and Balaji [49]. Each instance is of different size, has high or low costs and may differ in production capacity and customer demand. Since several techniques have been used to search for good solutions to these instances, a fair comparison between the results obtained by our technique and those shown in [49, 6, 5, 74, 51] is carried out. Given the number of comparisons, a computational experiment involves finding a near optimal solution of one of the considered tsFCTP instance. In this way, HAEA was run thirty different times for each instance. For each one of this thirty runs, we take the best solution and its fitness value, and use them to calculate the mean, the median and the median standard deviation [1] (the standard deviation using the median of the mean of the sample) over the thirty runs. This thirty runs gives us the statistical information in terms of the convergence of the algorithm for each instance of the problem.

Computational experiments are carried out on a laptop equipped with 8 GB of RAM and a 2.4 GHz Intel(R) Core(TM) i7-5500U. The parameters of HaEa are set to: For the distribution (production) mutation operator, a node of the distribution plan is randomly chosen using probability of $1/n$, where n is the number of nodes (for the production mutation is the number of production plants I, for the distribution mutation is the number of distribution centers J); for the selection operator, tournament selection by four individuals is applied. For a fair comparison of solutions found by HaEa against the solutions found in the literature, computational experiments are carried out using six combinations of population size and number of iterations (Table 3.4).

[1] We decided to use the median because it is a more robust statistical estimator of the mean

TABLE 3.1. Instance of tsFCTP model presented by [49]

Instance	Size		
	I	J	K
1	2	2	3
2	2	2	4
3	2	2	5
4	2	2	6
5	2	2	7
6	2	3	3
7	2	3	4
8	2	3	6
9	2	3	8
10	2	4	8
11	2	5	6
12	3	2	4
13	3	2	5
14	3	3	4
15	3	3	5
16	3	3	6
17	3	3	7a
18	3	3	7b
19	3	4	6
20	4	3	5

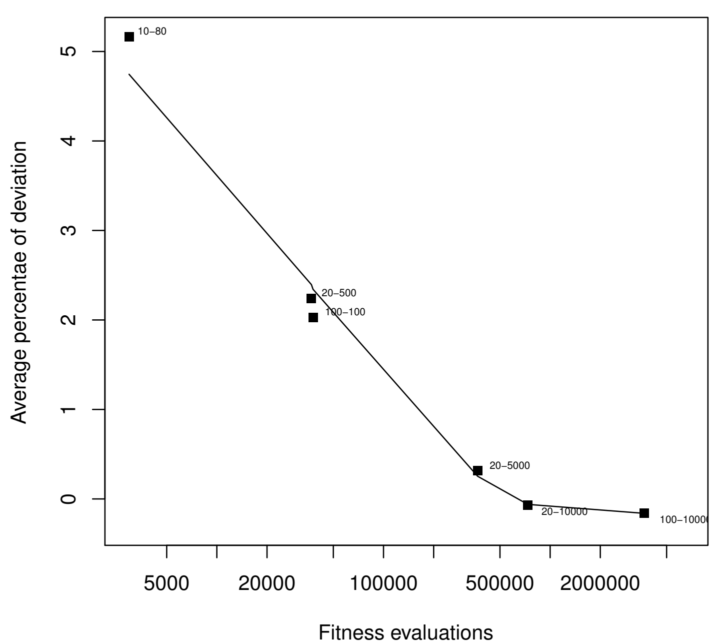


FIGURE 3.1. Plot (fitness evaluations against average percentage of deviation)

TABLE 3.2. Computational experiments using different population size and number of iterations

No.	10 × 80		20 × 500		20 × 5000		100 × 100		20 × 10000		100 × 10000	
	Median	St Dev	Median	St Dev	Median	St Dev	Median	St Dev	Median	St Dev	Median	St Dev
1	112600	0,00	112600	0,00	112600	0,00	112600	0,00	112600	0,00	112600	0,00
2	237750	0,00	237750	0,00	237750	0,00	237750	0,00	237750	0,00	237750	0,00
3	187700	2887,20	182250	1287,63	180450	375,70	183030	1333,72	180450	0,00	180450	0,00
4	169450	262,14	168378	1150,63	165650	384,14	169158	1443,07	165650	412,62	165650	0,00
5	170640	4582,93	167674	1765,49	164762	1169,02	167354	2140,69	162850	1385,22	162490	385,29
6	61600	1241,72	61200	1107,70	59500	670,82	61200	1060,03	59500	670,82	59500	0,00
7	33600	235,84	32830	444,58	32150	0,00	32970	496,20	32150	0,00	32150	0,00
8	72266	3447,69	67170	2435,49	67170	342,16	67170	1133,20	67170	90,74	67170	0,00
9	281050	4062,86	280420	4676,22	266236	3425,89	278046	4737,70	262940	2260,34	262970	1767,89
10	87600	1760,49	85200	2346,80	79452	1942,54	81324	3221,93	79238	1771,69	78800	1558,47
11	81340	5508,86	81340	2538,45	75065	2806,27	81340	1380,39	75065	1594,92	75065	0,00
12	47140	0,00	47140	0,00	47140	0,00	47140	0,00	47140	0,00	47140	0,00
13	182846	7427,02	178350	2194,70	178350	1992,43	175350	2095,71	175350	2438,50	175350	1549,19
14	65400	3652,98	57100	3239,27	57100	2029,54	57100	2105,91	57100	2020,47	57100	657,27
15	160664	5155,31	153865	1965,42	152800	1239,01	153694	1826,41	152800	1854,97	152800	807,69
16	132890	0,00	132890	0,00	132890	0,00	132890	0,00	132890	0,00	132890	0,00
17	107395	742,05	107285	2267,30	102425	1510,09	107395	1932,93	101965	1424,83	101210	1430,51
18	287360	4324,55	287360	2151,60	287360	3964,11	287360	702,81	282665	4475,88	281730	28,67
19	92150	3539,00	82763	2003,40	80364	1578,03	84450	1911,31	79500	1888,82	79500	1561,83
20	124450	511,18	118450	2541,81	118450	0,00	118450	3351,95	118450	497,51	118450	0,00
Exec Time	3.11 seconds		24.14 seconds		3.83 minutes		24.98 seconds		7.38 minutes		36.76 minutes	
Fit Eval	2972		36927		367176		37766		734038		3669868	

St Dev: Median standard deviation.
Exec Time: Total Execution time.
Fit Eval: Average fitness evaluations per computational experiment.

Table 3.3. Computational experiments using different population size and number of iterations

		10 × 80		20 × 500		100 × 100		20 × 5000		20 × 10000		100 × 10000	
Instance	Best	Median	Deviation (%)	Median	Deviation (%)	Median	Deviation (%)	Median	Deviation (%)	Median	Deviation (%)	Median	Deviation (%)
1	112600	112600	0	112600	0	112600	0	112600	0	112600	0	112600	0
2	237750	237750	0	237750	0	237750	0	237750	0	237750	0	237750	0
3	180450	187700	4,017	182250	0,998	183030	1,430	180450	0	180450	0	180450	0
4	165650	169450	2,293	168378	1,647	169158	2,118	165650	0	165650	0	165650	0
5	162490	170640	5,015	167674	3,190	167354	2,993	164762	1,398	162850	0,222	162490	0
6	59500	61600	3,529	61200	2,857	61200	2,857	59500	0	59500	0	59500	0
7	32150	33600	4,510	32830	2,115	32970	2,551	32150	0	32150	0	32150	0
8	67380	72266	7,251	67170	-0,312	67170	-0,312	67170	-0,312	67170	-0,312	67170	-0,312
9	258730	281050	8,626	280420	8,383	278046	7,466	266236	2,901	262940	1,627	262970	1,639
10	77400	87600	13,178	85200	10,078	81324	5,070	79452	2,651	79238	2,375	78800	1,809
11	80865	81340	0,587	81340	0,587	81340	0,587	75065	-7,172	75065	-7,172	75065	-7,172
12	47140	47140	0	47140	0	47140	0	47140	0	47140	0	47140	0
13	175350	182846	4,274	178350	1,711	175350	0	178350	1,711	175350	0	175350	0
14	57100	65400	14,535	57100	0	57100	0	57100	0	57100	0	57100	0
15	152800	160664	5,146	153865	0,697	153694	0,585	152800	0	152800	0	152800	0
16	132890	132890	0	132890	0	132890	0	132890	0	132890	0	132890	0
17	104115	107395	3,150	107285	3,045	107395	3,150	102425	-1,623	101965	-2,065	101210	-2,790
18	281100	287360	2,226	287360	2,227	287360	2,227	287360	2,227	282665	0,557	281730	0,224
19	76900	92150	19,830	82763	7,624	84450	9,818	80364	4,505	79500	3,381	79500	3,381
20	118450	124450	5,065	118450	0	118450	0	118450	0	118450	0	118450	0

Best: Best solution found among algorithms found in the literature
Deviation (%): Percentage of Deviation from Best $\left(\frac{HaEa-Best}{Best}\right) * 100$

Table 3.2 shows the median values calculated for the thirty computational experiments, the median standard deviation, the execution time and the fitness evaluations using each combination of population size and total number of iterations. As it can be noticed, for all combinations of population size and number of iterations and for the twenty instances of the problem, the median standard deviation is very small. Using a combination of ten individuals and eighty iterations, the highest median standard deviation is 7427,02. Using twenty individuals and five hundred iterations, the highest median standard deviation is 4676,22. Using one hundred individuals and ten thousand iterations the highest median standard deviation is 1767,89. These values indicate that the majority of the individuals converge to the median value for a given instance of the problem.

Table 3.3 shows (for each combination of population size and number of iterations) the median values obtained from the computational experiments and the percentage of deviation from the best solution found among the ones presented by the techniques found in the literature. The percentage of deviation is calculated using the median value found by HaEa from the thirty computational experiments. If the percentage of deviation is negative, it means that HaEa improved the best solution found for the given instance. Similarly to the standard deviation values presented in Table 3.2, the percentages of deviation from the best results found in the literature are also very small. As it can be noticed, the highest percentage of improvement is 7.17% (instance 11) and the highest percentage of deviation is 19,830% for the instance 14 using ten individuals and eighty iterations. The average percentage of deviation using 10 individuals and eighty iterations is 5,162%. Using twenty individuals and five hundred iterations is 2,242%. Using one hundred individuals end one hundred iterations, the average percentage of deviation is 2,027%. The average percentage of deviation using twenty individuals and five thousand iterations is 0,314%. Using twenty individuals and ten thousand iterations the average is -0,069%. As for a combination of one hundred individuals and ten thousand iterations, the average percentage of deviation is -0,161%. The decreasing percentage of deviation shows the improvement on the solutions found by HaEa as its given more resources to carry out the optimization. This behavior is also shown in Figure 3.1 where the curve that approximate the points shows that the proposed technique will always find better solutions if it continues with optimization. Appendix A.1 shows the tables for the results of the computational experiments for all combinations of population size and number of iterations.

Table 3.4. Average performance obtained by HaEa

Comparison of average performance				
Previous works		HaEa	Pob x Iter	Avg fit eval
GA - [49]	132240	134794,55	10 x 80	2972
SA - [6]	132057			
TSGA - [5]	131050	**129883,2**	20 x 5000	367176
ACO - [74]	129607,75	132102,25	20 x 500	36927
AIS - [51]	129349,5			
SFA - [51]	129040,5			
Avg fit eval: Average fitness evaluations		131788,55	100 x 100	37766
		129161,15	20 x 10000	734038
		129038,25	100 x 10000	3669868

3.1 Comparison against genetic algorithm and simulated annealing proposed by Jawahar and Balaji

Results from Table 3.2 are compared against six solution methodologies found in the literature. The median values from column 1 are compared against the results obtained by the GA proposed by Jawahar and Balaji [49] and against results obtained by the SA algorithm proposed by Balaji and Jawahar [6]. Jawahar and Balaji employed ten chromosomes and between fifty and eighty iterations. As for the SA, Balajiand Jawahar used a parameter FR_CNT that checks whether the algorithm could be frozen or not. The temperature was set to 475 and the cooling rate was set to 0,9. When FT_CNT reaches a predefined value or when the temperature reaches a value of twenty, the algorithm stops. Given the parameters used by the GA and the SA, the population size and the number of iterations for HaEa are set to ten individuals and eighty iterations. As it can be noticed in Table 3.5, using this combination of population size and number of iterations, HaEa, compared to the GA, could find better solutions for two instances and could find equal solutions for four instances. The highest percentage of deviation was $10,35\%$, and the lowest percentage of deviation was $-13,985$. Column 7 shows the percentage of deviation from SA. In this case, HaEa found better solution for one instance (instance 11) and found equal solution for 5 instances. The highest percentage of deviation was $19,288$ and the lowest was $-14,572$. difference in the results obtained by HaEa compared to SA are not very large, since the average percentage of deviation from this algorithm does not exceed 6.5% (Table 3.5). As expected, HaEa shows a similar behavior to the GA proposed by Jawahar and Balaji and the SA proposed by Balaji and Jawahar given the same amount of resources. Local search techniques such as simulated annealing, tend to converge to local optimum. Population-based algorithms as HaEa are able to find better solutions when given the necessary time and resources.

3.2 Comparison against genetic algorithm (TSGA) proposed by Raj and Rajendran

Median values from column 6 (Table 3.2) are compared against the results obtained by the GA proposed by [5] (Table 3.6). In this case, the GA, called two-stage genetic algorithm (TSGA), uses a two phase improvement scheme to find better solutions after a crossover. Raj and Rajendran completed computational experiments using two versions of the TSGA (TSGA-RN and TSGA-ARN). They employed twenty chromosomes as initial population and ten thousand iterations or 3600 CPU-time seconds as a termination criterion, using the one that ends earlier the optimization. Although they employed the termination condition mentioned above, we set the termination criterion to five thousand iterations for HaEa. Results compared to the ones obtained by the TSGA show that HaEa could perform better, obtaining equal or lower costs for eighteen of twenty instances. In consequence, the average performance of HaEa (129883,2) using five thousand iterations and twenty candidate solutions is lower than the TSGA (131050). It is also important to note that, these results are also better than the ones obtained by the GA and SA presented in Table 3.5, showing not only that EAs can perform better than other methods like SA and similar algorithms, but also shows the unfairness of the comparisons among the publications found in the literature.

Instances		Results obtained by			Percentage of deviation from	
No	Size	GA[a]	SA[b]	HaEa	$\left(\frac{HaEa-GA}{GA}\right)*100$	$\left(\frac{HaEa-SA}{SA}\right)*100$
1	2-2-3	112600	112600	112600	0	0
2	2-2-4	237750	237750	237750	0	0
3	2-2-5	180450	180450	187700	4,018	4,018
4	2-2-6	165650	165650	169450	2,294	2,294
5	2-2-7	162490	162490	170640	5,016	5,016
6	2-3-3	59500	59500	61600	3,529	3,529
7	2-3-4	32150	32150	33600	4,510	4,510
8	2-3-6	69970	70480	72266	3,218	2,534
9	2-3-8	264680	263000	281050	6,185	6,863
10	2-4-8	85200	80400	87600	2,817	8,955
11	2-5-6	94565	95215	81340	-13,985	-14,572
12	3-2-4	47140	47140	47140	0	0
13	3-2-5	178950	178950	182846	2,177	2,177
14	3-3-4	57100	51150	65400	14,536	14,536
15	3-3-5	152800	152800	160664	5,147	5,147
16	3-3-6	132890	132890	132890	0	0
17	3-3-7a	106615	104115	107395	0,732	3,150
18	3-3-7b	302350	287360	287360	-4,958	0
19	3-4-6	83500	77250	92150	10,359	19,288
20	4-3-5	118450	118450	124450	5,065	5,065

[a] Genetic Algorithm - [49]
[b] Simulated Annealing -[6]

Table 3.5. Performance comparisons against Genetic Algorithm and Simulated Annealing

Instances		Results obtained by		Percentage of deviation from
No	Size	TSGA	HaEa	$\left(\frac{HaEa-TSGA}{TSGA}\right)*100$
1	2-2-3	112600	112600	0
2	2-2-4	237750	237750	0
3	2-2-5	180450	180450	0
4	2-2-6	165650	165650	0
5	2-2-7	162490	164762	1,398
6	2-3-3	59500	59500	0
7	2-3-4	32150	32150	0
8	2-3-6	67380	67170	-0,312
9	2-3-8	258730	266236	2,901
10	2-4-8	84600	79452	-6,085
11	2-5-6	80865	75065	-7,172
12	3-2-4	47140	47140	0
13	3-2-5	178950	178350	-0,335
14	3-3-4	61000	57100	-6,393
15	3-3-5	156900	152800	-2,613
16	3-3-6	132890	132890	0
17	3-3-7a	106745	102425	-4,047
18	3-3-7b	295060	287360	-2,610
19	3-4-6	81700	80364	-1,635
20	4-3-5	118450	118450	0
TSGA - [5]				

Table 3.6. Performance comparison against GA proposed by [5]

3.3 Comparison against ACO, AIS and SFA

Median values from column 4 are compared against the Ant Colony Optimization (ACO) proposed by Panicker, Venga and Sridharan [74], the Artificial Immune System (AIS) and the Sheep Flock Algorithm (SFA) proposed by Kannan, Govidan and Soleimani[51]. Kannan, Govidan and Soleimani compared the results obtained by the AIS, the SFA, and the results obtained by the ACO algorithm which employed one hundred ants and ten iterations. Kannan, Govidan and Soleimani employed ten individuals for the AIS and twenty individuals for the SFA. They also used a termination criterion of five hundred iterations for both algorithms. This is the reason why the termination condition and the populations size for HaEa are set to five hundred iterations and twenty individuals to compare the results against the ACO algorithm, the AIS and the SFA. Despite using similar population sizes and number of iterations, HAEA may be at a slight disadvantage compared to algorithms such as ACO and AIS, which tend to consume a large amount of computational resources. AIS for example, generates (via cloning) many individuals during optimization, therefore, the size of the final population may be very large compared to algorithms using a fixed population size. In the case of SFA, because it is a fairly new heuristic, it is also difficult to make a fair comparison of the performance and the results obtained by HaEa.

As it can be noticed in Tables (3.5,3.6,3.7), it is clear that the SFA proposed by Kannan, Govidan and Soleimani [51] found the best results among the techniques found in the literature that considers this tsFCTP. The AIS and the SFA were able to improve or match the solutions found by previously used algorithms. In Table 3.7, results obtained by HaEa using twenty individuals and five hundred iterations, are compared against ACO algorithm , AIS, and SFA. The Results obtained by HaEa, compared against the ACO show that it could find equal solutions for six instances of the problem and better solutions for two instances. The average percentage of deviation against ACO did not exceed 3% for solutions that are not improved by HaEa. Results compared against those obtained by the AIS, HaEa also found equal or better solutions for eight instances of the problem and the average percentage of deviation was 3.43% for solutions that are not improved by HaEa. As for the solutions that are compared against SFA, HaEa could improve the solution for one instance, and found equal solutions for six instances. Lastly, the average percentage of deviation was 3.47% for solutions that are not improved by HaEa.

Instance		Results obtained by				Percentage of deviation from		
No.	Size	ACO[a]	AIS[b]	SFA[c]	HaEa	$\left(\frac{HaEa-ACO}{ACO}\right)*100$	$\left(\frac{HaEa-AIS}{AIS}\right)*100$	$\left(\frac{HaEa-SFA}{SFA}\right)*100$
1	2-2-3	112600	112600	112600	112600	0	0	0
2	2-2-4	237750	237750	237750	237750	0	0	0
3	2-2-5	180450	180450	180450	182250	0,998	0,998	0,998
4	2-2-6	165650	165650	165650	168378	1,647	1,647	1,647
5	2-2-7	162490	162490	162490	167674	3,190	3,190	3,190
6	2-3-3	59500	59500	59500	61200	2,857	2,857	2,857
7	2-3-4	32150	32150	32150	32830	2,115	2,115	2,115
8	2-3-6	69045	67380	67380	67170	-2,716	-0,312	-0,312
9	2-3-8	258730	258730	258730	280420	8,383	8,383	8,383
10	2-4-8	80900	77400	77400	85200	5,315	10,078	10,078
11	2-5-6	80865	80865	80865	81340	0,587	0,587	0,587
12	3-2-4	47140	47140	47140	47140	0	0	0
13	3-2-5	178950	178950	175350	178350	-0,335	-0,335	1,711
14	3-3-4	57100	57100	57100	57100	0	0	0
15	3-3-5	152800	152800	152800	153865	0,697	0,697	0,697
16	3-3-6	132890	132890	132890	132890	0	0	0
17	3-3-7a	105715	105715	104115	107285	1,485	1,485	3,045
18	3-3-7b	281730	281730	281100	287360	1,998	1,998	2,227
19	3-4-6	77250	77250	76900	82763	7,137	7,137	7,1624
20	4-3-5	118450	118450	118450	118450	0	0	0

[a] Ant Colony Optimization - [74]
[b] Artificial Immune System - [51]
[c] Sheep Flock Algorithm - [51]

Table 3.7. Performance comparison against Ant Colony Optimization, Artificial Immune System, and Sheep Flock Algorithm

3.4 Comparison against all techniques using large population size and number of iterations

The results obtained by HaEa using one hundred individuals and ten thousand iterations were compared against the results obtained by the methods mentioned previously. For this comparison, in addition to calculating the percentage of deviation, the Wilcoxon signed-rank test was carried out. Table 3.8 shows the percentage of deviation from the six methods found in the literature. As it can be noticed, HaEa could match or improve solutions for the majority of the instances. For example, percentage of deviation from TSGA shows that it could match solutions for ten instances and improve 9 with an a average of deviation (improvement) of 4.19%.

For the Wilcoxon signed-rank test, the null and alternative hypothesis were established as:

- H_0 : There is no significant improvement in the results obtained by HaEa compared to previous results obtained by (GA, SA, TSGA, ACO, AIS, SFA) methodology (H_0 : Difference > 0).
- H_1 : HaEa provides better results compared to results obtained by (GA, SA, TSGA, ACO, AIS, SFA) solution methodology (H_1 : Difference ≤ 0).

As shown in Table 3.9, with a level of significance $\alpha = 0.05$, the Wilcoxon test shows that there is a significant difference between the results obtained by HaEa and the results obtained by three solution methodologies, these are, genetic algorithm proposed by [49], simulated annealing proposed by [6], and the two stage genetic algorithm proposed by

TABLE 3.9. Wilcoxon test for solutions found by HAEA against other techniques.

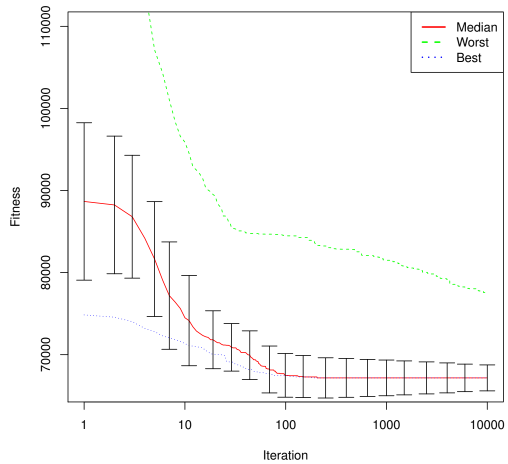
Algorithm	N	Rank+	Rank-	Critical value	p-value	Reject H0
GA	8	36	0	3	0.01427	YES
SA	8	33	3	3	0.04232	YES
TSGA	10	49	6	8	0.0322	YES
ACO	7	20	8	2	0.3525	NO
AIS	7	10	10	2	0.5541	NO
SFA	7	13	13	2	0.9326	NO
Level of significance $\alpha = 0{,}05$						

[5]. As for the ant colony optimization algorithm proposed by [74], the artificial immune system and the sheep flock algorithm proposed by [51], the Wilcoxon test there is no significant difference between these results and the results obtained by our HAEA.

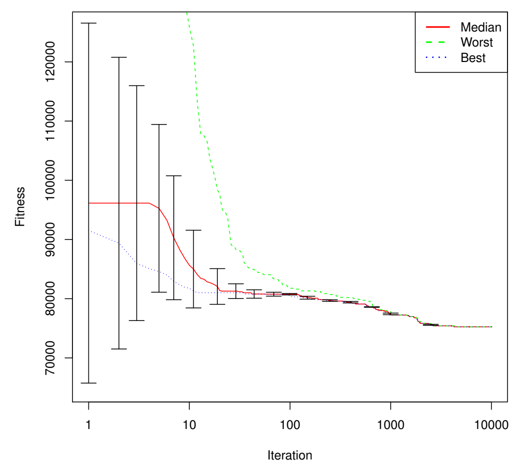
TABLE 3.8. Percentage of deviation from previous solution methodologies found in the literatures with respect to HAEA using 100 individuals and 10.000 iterations.

Instance	Percentage of deviation from					
	GA	SA	TSGA	ACO	AIS	SFA
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	-4,002	-4,696	-0,312	-2,716	-0,312	-0,312
9	-0,646	-0,011	1,639	1,639	1,639	1,639
10	-7,512	-1,990	-6,856	-2,596	1,809	1,809
11	-20,621	-21,163	-7,172	-7,172	-7,172	-7,172
12	0	0	0	0	0	0
13	-2,012	-2,012	-2,012	-2,012	-2,012	0
14	0	0	-6,393	0	0	0
15	0	0	-2,613	0	0	0
16	0	0	0	0	0	0
17	-5,070	-2,790	-5,185	-4,261	-4,261	-2,790
18	-6,820	-1,959	-4,518	0	0	0,224
19	-4,790	2,913	-2,693	2,913	2,913	3,381
20	0	0	0	0	0	0

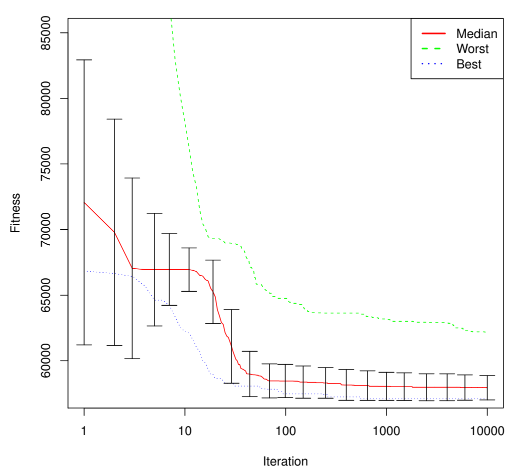
For a further analysis of the performance of HAEA, the evolution of the population and the evolution of the operator rates are shown in Figure 3.2 and Figure 3.3 using one hundred individuals and ten thousand iterations for six instances of the problem. Figure 3.2 clearly shows how the population converges to a good solution needing approximately 100 iterations. After one hundred iterations, the population is able to scape local optima to find better solutions. Figure 3.3 shows how the crossover operator is applied much more times than the mutation operators until the population converges. The high probability of the crossover operator in early stages of the optimization is an expected behavior given that this operator makes more changes to the individuals than the mutation operators, carrying out a great exploration of the search space. Appendix A.2 shows the figures of the evolution of the population and the evolution of the operator rates for the remaining instances of the problem.



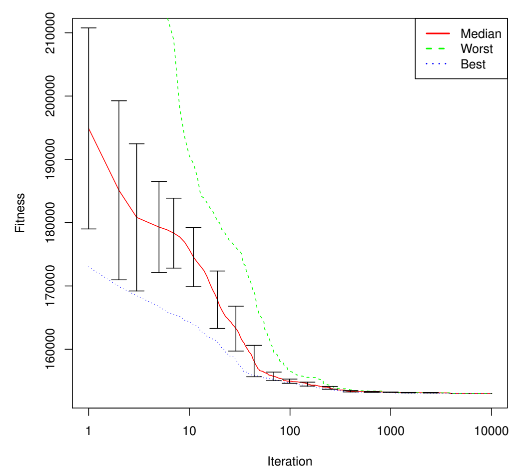
(a) Instance 2-3-6



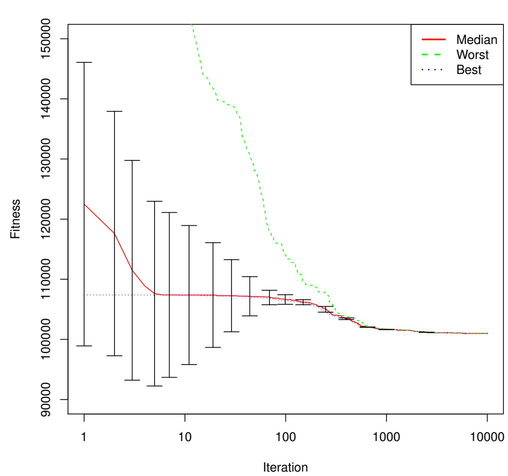
(b) Instance 2-5-6



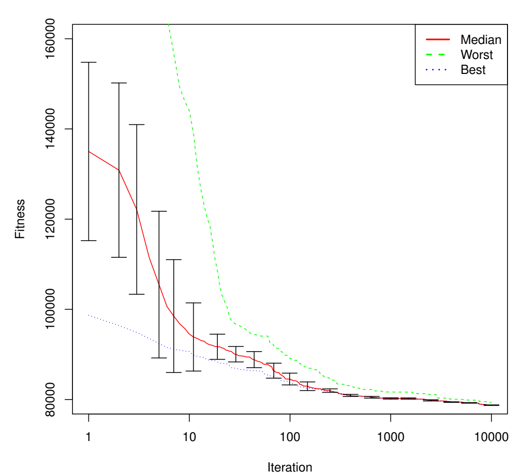
(c) Instance 3-3-4



(d) Instance 3-3-5

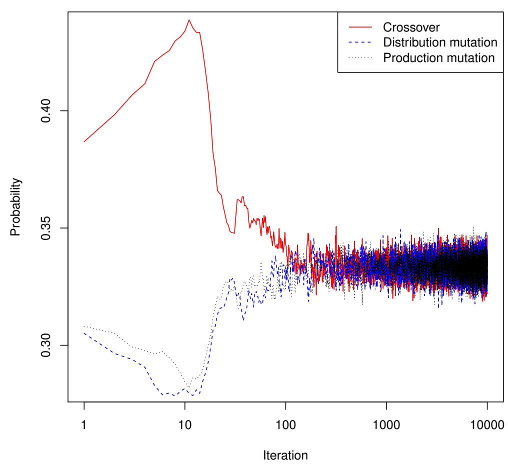


(e) Instance 3-3-7a

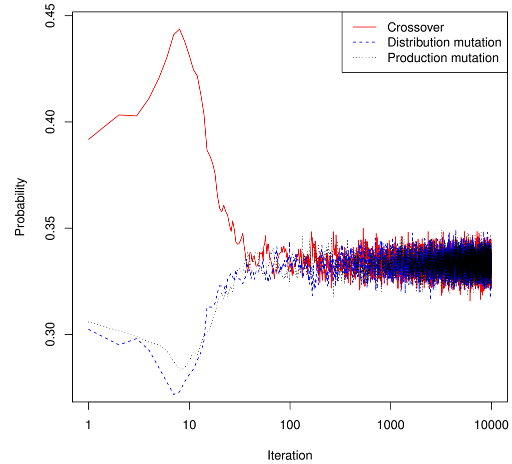


(f) Instance 3-4-6

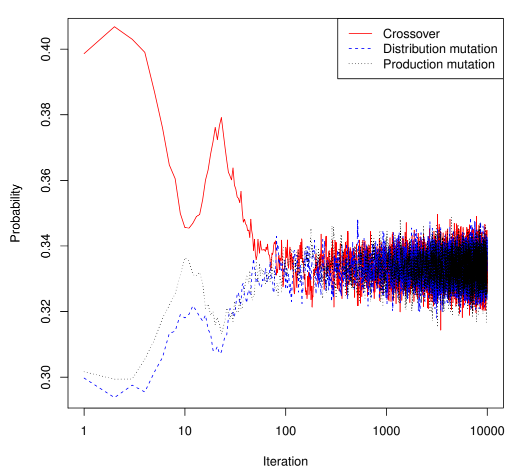
FIGURE 3.2. Average evolution of fitness value of instances 2-3-6 (a), 2-5-6 (b), 3-3-4 (c), 3-3-5 (d), 3-3-7a (e) and 3-4-6 (f) for 30 computational experiments using 100 individuals and 10.000 iterations.



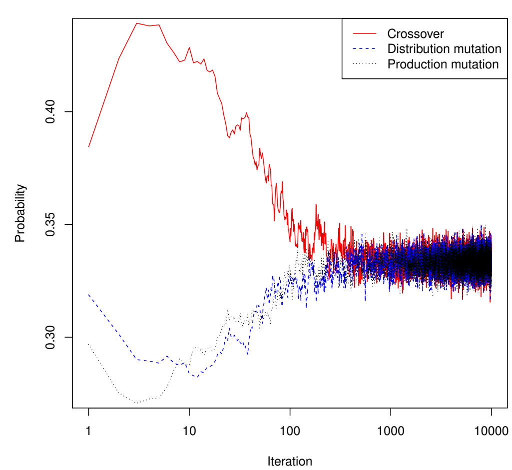
(a) Instance 2-3-6



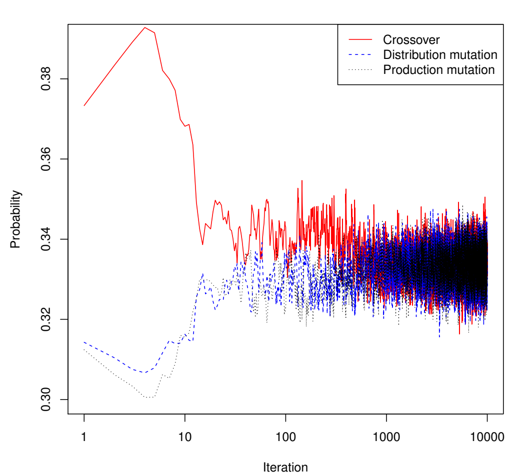
(b) Instance 2-5-6



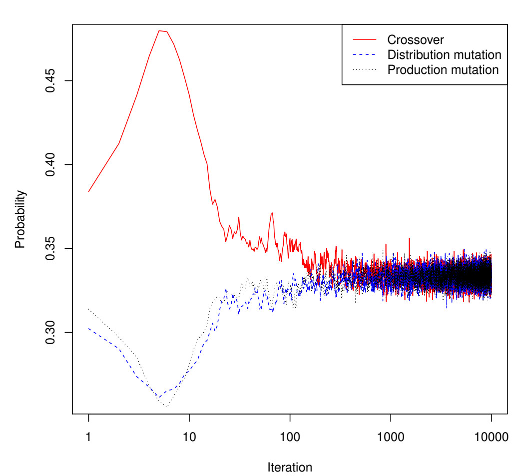
(c) Instance 3-3-4



(d) Instance 3-3-5



(e) Instance 3-3-7a



(f) Instance 3-4-6

FIGURE 3.3. Average evolution of operator probabilities of instances 2-3-6 (a), 2-5-6 (b), 3-3-4 (c), 3-3-5 (d), 3-3-7a (e) and 3-4-6 (f) for 30 computational experiments using 100 individuals and 10.000 iterations.

APPENDIX A

Tables and Figures

A.1 Tables

Table A.1. Computational results used for Table 2. using 10 individuals and 80 iterations

Instance	Best	Worst	Mean	Median	Standard deviation
223	112600.0	112600.0	112600.00	112600.00	0.00
224	237750.0	237750.0	237750.00	237750.00	0.00
225	181320.0	189100.0	186593.93	187700.00	2887.20
226	168222.0	186320.0	169433.87	169450.00	262.14
227	162850.0	188885.0	172385.50	170640.00	4582.93
233	59500.0	67600.0	62008.00	61600.00	1241.72
234	32790.0	33600.0	33473.00	33600.00	235.84
236	67170.0	83902.0	72905.10	72266.00	3447.69
238	269503.0	358741.0	281345.47	281050.00	4062.86
248	81400.0	91100.0	87571.23	87600.00	1760.49
256	75065.0	96140.0	83964.00	81340.00	5508.86
324	47140.0	47140.0	47140.00	47140.00	0.00
325	175350.0	204310.0	184668.17	182846.00	7427.02
334	57100.0	73531.0	63924.70	65400.00	3652.98
335	153670.0	181408.0	161682.93	160664.00	5155.31
336	132890.0	132890.0	132890.00	132890.00	0.00
337a	104283.0	135875.0	107217.33	107395.00	742.05
337b	287360.0	301188.0	289621.30	287360.00	4324.55
346	84450.0	105842.0	91820.67	92150.00	3539.00
435	122355.0	124450.0	124294.27	124450.00	511.18

TABLE A.2. Computational results used for Table 3. using 20 individuals and 5000 iterations

Instance	Best	Worst	Mean	Median	Standard deviation
223	112600.0	112600.0	112600.00	112600.00	0.00
224	237750.0	237750.0	237750.00	237750.00	0.00
225	180450.0	182250.0	180557.00	180450.00	375.70
226	165650.0	167754.0	165720.13	165650.00	384.14
227	162490.0	166298.0	164500.20	164762.00	1169.02
233	59500.0	61000.0	59800.00	59500.00	670.82
234	32150.0	32150.0	32150.00	32150.00	0.00
236	66092.0	80007.0	67185.17	67170.00	342.16
238	260730.0	291744.0	267109.57	266236.00	3425.89
248	78800.0	85073.0	80458.10	79452.00	1942.54
256	75065.0	81340.0	76320.00	75065.00	2806.27
324	47140.0	47140.0	47140.00	47140.00	0.00
325	175350.0	180770.0	177398.33	178350.00	1992.43
334	57100.0	68864.0	58136.20	57100.00	2029.54
335	152800.0	156414.0	153400.07	152800.00	1239.01
336	132890.0	132890.0	132890.00	132890.00	0.00
337a	99095.0	107130.0	102457.20	102425.00	1510.09
337b	281730.0	300270.0	287706.00	287360.00	3964.11
346	76900.0	82716.0	80268.37	80364.00	1578.03
435	118450.0	118450.0	118450.00	118450.00	0.00

TABLE A.3. Computational results used for Table 4. using 20 individuals and 500 iterations

Instance	Best	Worst	Mean	Median	Standard deviation
223	112600.0	112600.0	112600.00	112600.00	0.00
224	237750.0	237750.0	237750.00	237750.00	0.00
225	180450.0	184210.0	182104.50	182250.00	1287.63
226	165650.0	201634.0	168393.33	168378.00	1150.63
227	162850.0	169890.0	167147.63	167674.00	1765.49
233	59500.0	66610.0	60790.00	61200.00	1107.70
234	32150.0	33525.0	32826.67	32830.00	444.58
236	67170.0	87081.0	68178.40	67170.00	2435.49
238	267836.0	303530.0	278099.07	280420.00	4676.22
248	81150.0	87600.0	84712.20	85200.00	2346.80
256	75065.0	86065.0	80421.67	81340.00	2538.45
324	47140.0	47140.0	47140.00	47140.00	0.00
325	175350.0	192900.0	178033.67	178350.00	2194.70
334	57100.0	67200.0	58950.33	57100.00	3239.27
335	152800.0	158600.0	154785.93	153865.00	1965.42
336	132890.0	132890.0	132890.00	132890.00	0.00
337a	101965.0	107395.0	105975.67	107285.00	2267.30
337b	283200.0	300270.0	287811.50	287360.00	2151.60
346	77854.0	90004.0	82687.63	82763.00	2003.40
435	118450.0	124450.0	119868.23	118450.00	2541.81

Table A.4. Computational results using 100 individuals and 100 iterations

Instance	Best	Worst	Mean	Median	Standard deviation
223	112600.0	112600.0	112600,00	112600,00	0,00
224	237750.0	237750.0	237750,00	237750,00	0,00
225	180450.0	189100.0	182623,67	183030,00	1333,72
226	165650.0	208130.0	168418,67	169158,00	1443,07
227	162490.0	174808.0	167040,33	167354,00	2140,69
233	59500.0	70581.0	60736,67	61200,00	1060,03
234	32150.0	33600.0	32803,67	32970,00	496,20
236	67170.0	89241.0	67685,20	67170,00	1133,20
238	266822.0	332176.0	276820,80	278046,00	4737,70
248	78800.0	91100.0	82945,00	81324,00	3221,93
256	75065.0	86772.0	81237,17	81340,00	1380,39
324	47140.0	47140.0	47140,00	47140,00	0,00
325	175350.0	202243.0	176661,33	175350,00	2095,71
334	57100.0	72625.0	57954,33	57100,00	2105,91
335	152800.0	170156.0	154584,93	153694,00	1826,41
336	132890.0	132890.0	132890,00	132890,00	0,00
337a	102660.0	131145.0	106313,93	107395,00	1932,93
337b	283692.0	317487.0	287198,80	287360,00	702,81
346	81233.0	107488.0	84572,47	84450,00	1911,31
435	118450.0	151570.0	120758,97	118450,00	3351,95

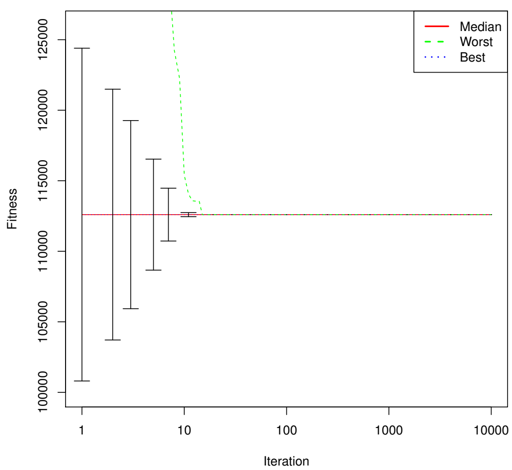
Table A.5. Computational results using 20 individuals and 10000 iterations

Instance	Best	Worst	Mean	Median	Standard deviation
223	112600.0	112600.0	112600,00	112600,00	0,00
224	237750.0	237750.0	237750,00	237750,00	0,00
225	180450.0	180450.0	180450,00	180450,00	0,00
226	165650.0	167910.0	165725,33	165650,00	412,62
227	162490.0	165562.0	163676,67	162850,00	1385,22
233	59500.0	61000.0	59800,00	59500,00	670,82
234	32150.0	32150.0	32150,00	32150,00	0,00
236	67170.0	79605.0	67186,57	67170,00	90,74
238	260230.0	285750.0	263474,77	262940,00	2260,34
248	78550.0	83700.0	80010,73	79238,00	1771,69
256	75065.0	81340.0	75506,43	75065,00	1594,92
324	47140.0	47140.0	47140,00	47140,00	0,00
325	175350.0	194850.0	177055,33	175350,00	2438,50
334	57100.0	64694.0	57953,13	57100,00	2020,47
335	152800.0	159815.0	153684,57	152800,00	1854,97
336	132890.0	132890.0	132890,00	132890,00	0,00
337a	99095.0	103310.0	101742,37	101965,00	1424,83
337b	281730.0	300270.0	284815,60	282665,00	4475,88
346	76900.0	83726.0	79347,33	79500,00	1888,82
435	118450.0	121175.0	118540,83	118450,00	497,51

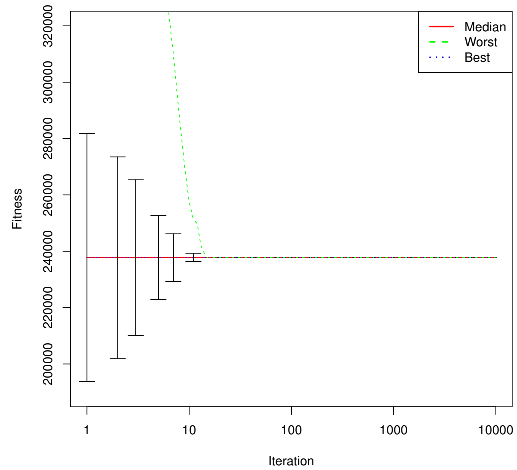
Instance	Best	Worst	Mean	Median	Standard deviation
223	112600.0	112600.0	112600,00	112600,00	0,00
224	237750.0	237750.0	237750,00	237750,00	0,00
225	180450.0	180450.0	180450,00	180450,00	0,00
226	165650.0	165650.0	165650,00	165650,00	0,00
227	162490.0	164538.0	162582,27	162490,00	385,29
233	59500.0	59500.0	59500,00	59500,00	0,00
234	32150.0	32150.0	32150,00	32150,00	0,00
236	67170.0	81457.0	67170,00	67170,00	0,00
238	260230.0	287496.0	262755,33	262970,00	1767,89
248	77836.0	83500.0	79421,57	78800,00	1558,47
256	75065.0	75065.0	75065,00	75065,00	0,00
324	47140.0	47140.0	47140,00	47140,00	0,00
325	175350.0	190950.0	176150,00	175350,00	1549,19
334	57100.0	66250.0	57220,00	57100,00	657,27
335	152800.0	156100.0	153093,07	152800,00	807,69
336	132890.0	132890.0	132890,00	132890,00	0,00
337a	99095.0	103310.0	100988,43	101210,00	1430,51
337b	281616.0	299388.0	281722,60	281730,00	28,67
346	76900.0	88422.0	78800,43	79500,00	1561,83
435	118450.0	118450.0	118450,00	118450,00	0,00

Table A.6. Computational results using 100 individuals and 10000 iterations

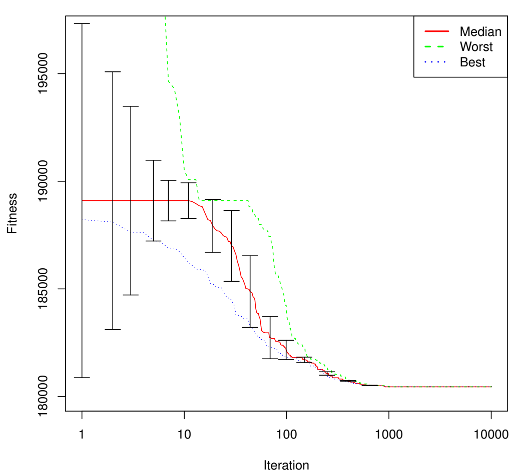
A.2 Figures



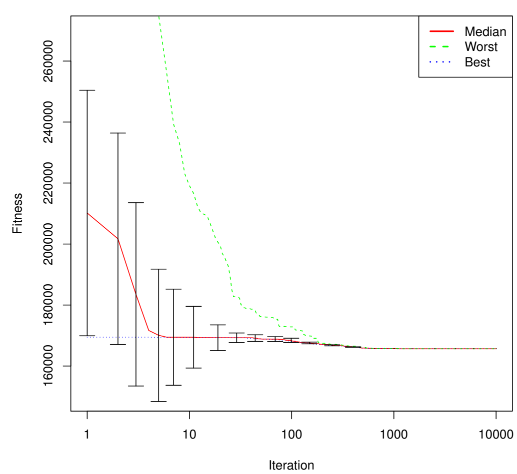
(a) Instance 2-2-3



(b) Instance 2-2-4

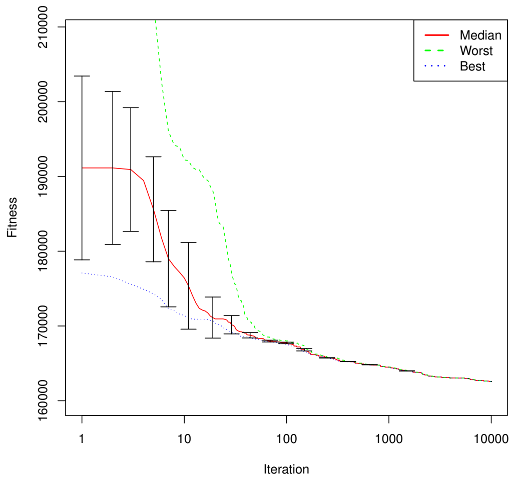


(c) Instance 2-2-5

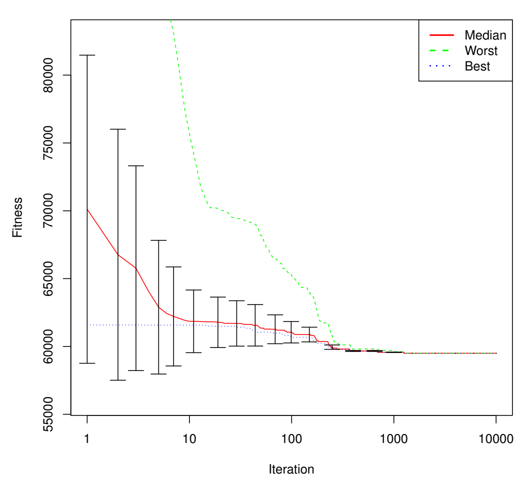


(d) Instance 2-2-6

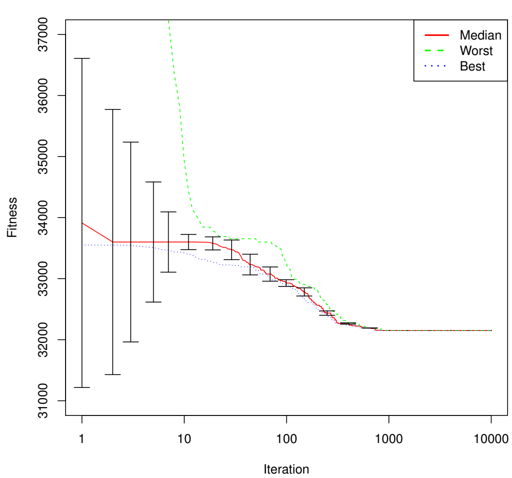
FIGURE A.1. Average evolution of fitness value of instances 2-2-3 (a), 2-2-4 (b), 2-2-5 (c), 2-2-6 for 30 computational experiments using 100 individuals and 10.000 iterations.



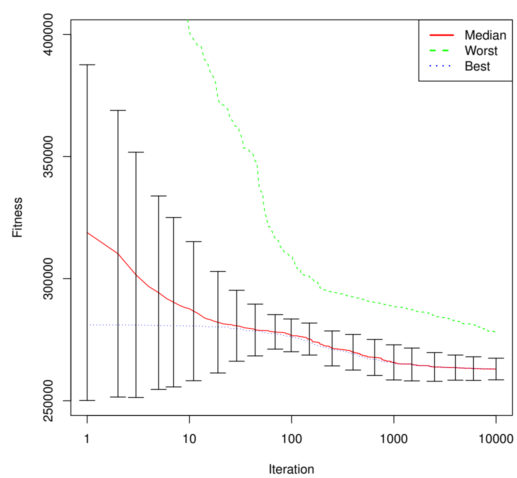
(a) Instance 2-2-7



(b) Instance 2-3-3

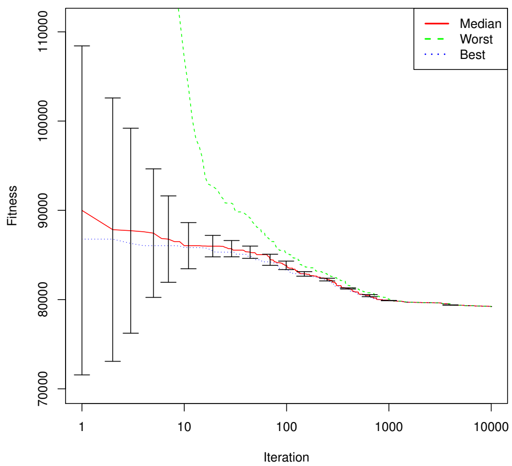


(c) Instance 2-3-4

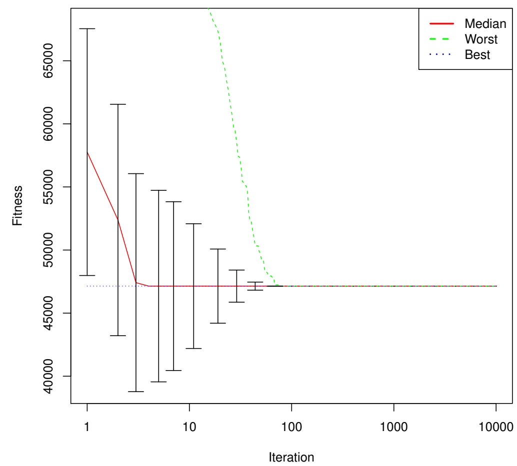


(d) Instance 2-3-8

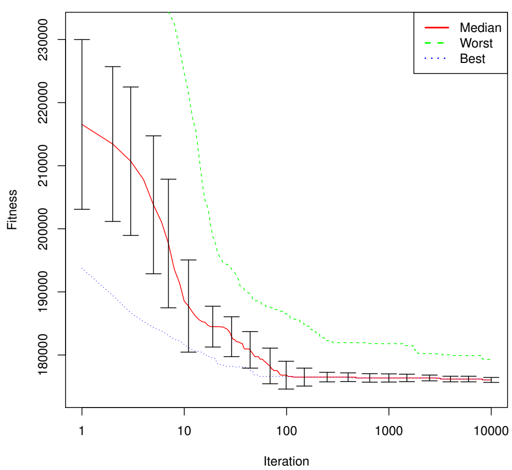
FIGURE A.2. Average evolution of fitness value of instances 2-2-7 (a), 2-3-3 (b), 2-3-4 (c), 2-3-8 for 30 computational experiments using 100 individuals and 10.000 iterations.



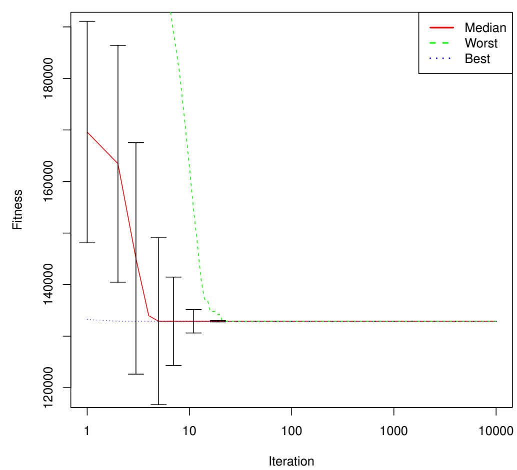
(a) Instance 2-4-8



(b) Instance 3-2-4

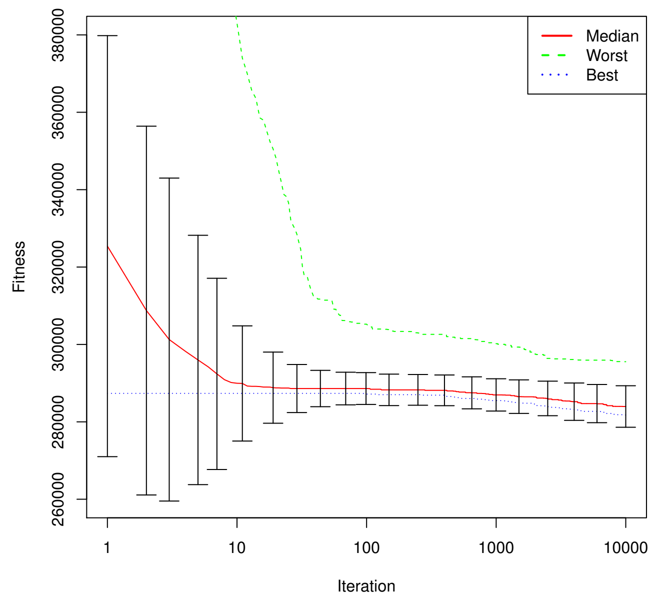


(c) Instance 3-2-5

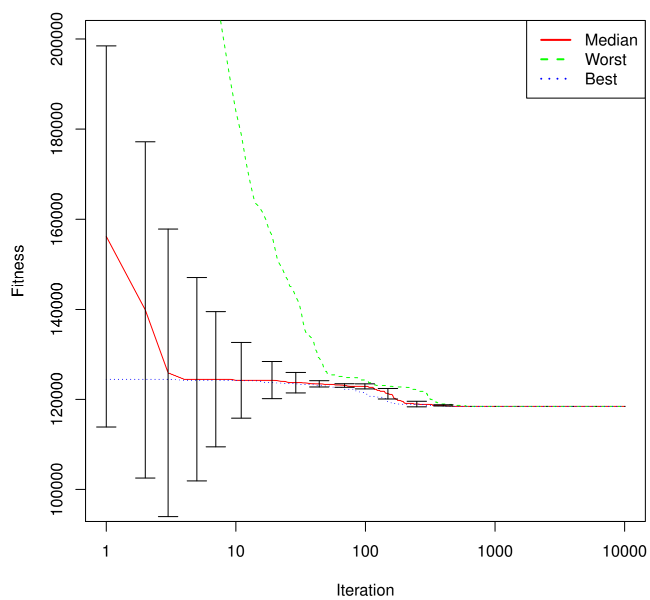


(d) Instance 3-3-6

FIGURE A.3. Average evolution of fitness value of instances 2-4-8 (c), 3-2-4 (d), 3-2-5 (e) and 3-3-6 (f) for 30 computational experiments using 100 individuals and 10.000 iterations.

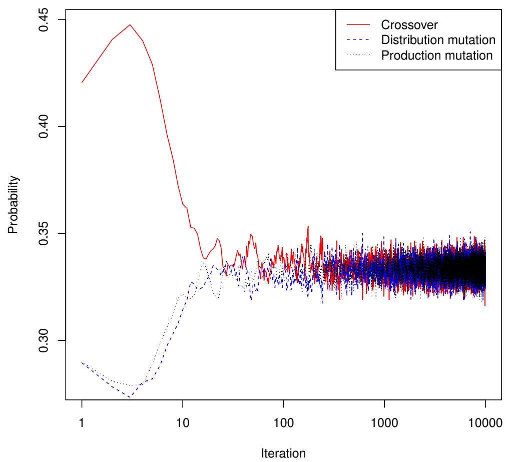


(a) Instance 3-3-7b

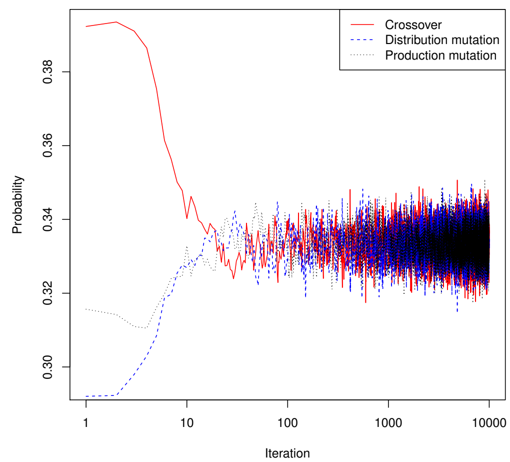


(b) Instance 4-3-5

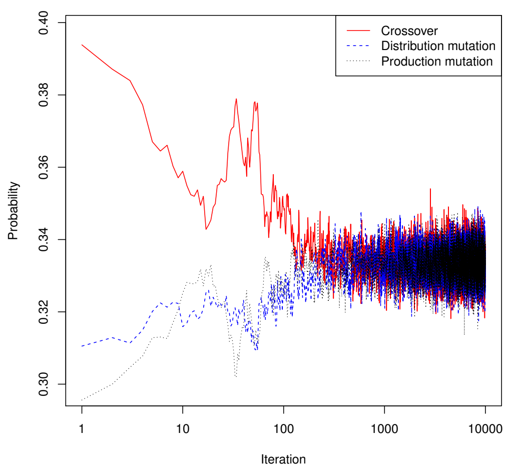
FIGURE A.4. Average evolution of fitness value of instances 3-3-7b (a) and 4-3-5 (b) for 30 computational experiments using 100 individuals and 10.000 iterations.



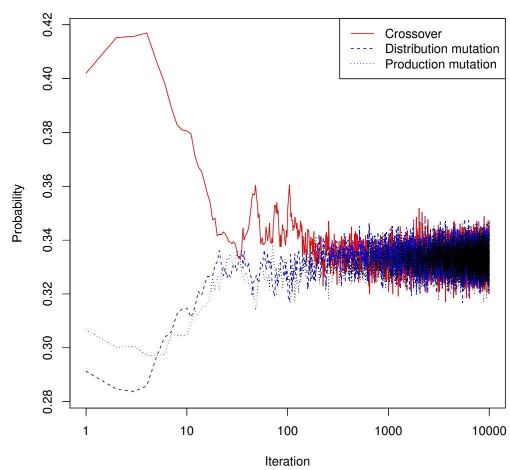
(a) Instance 2-2-3



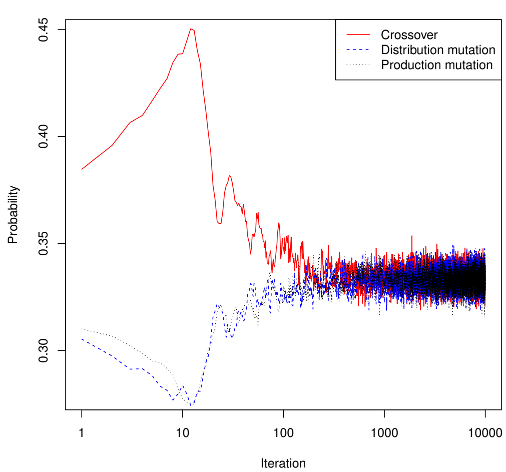
(b) Instance 2-2-4



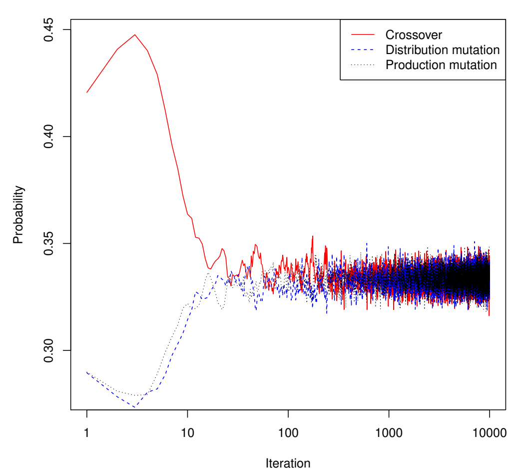
(c) Instance 2-2-5



(d) Instance 2-2-6

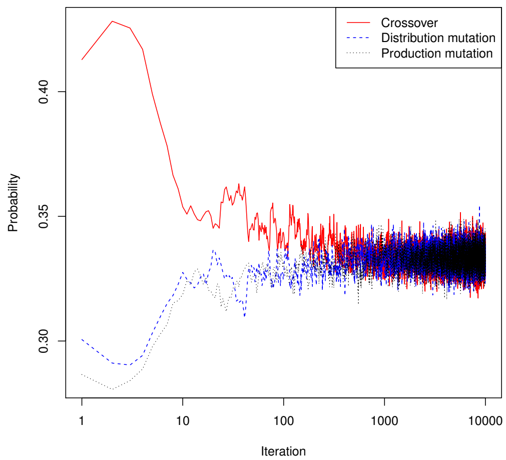


(e) Instance 2-2-7

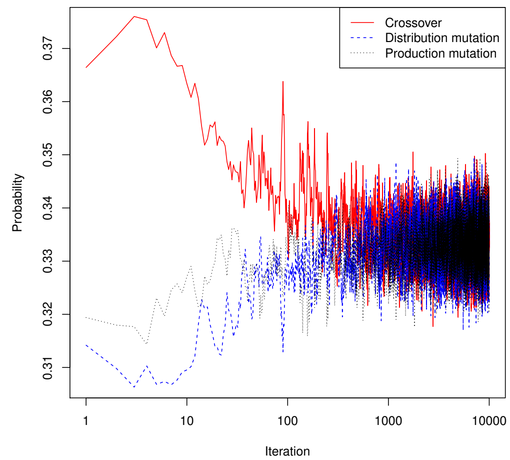


(f) Instance 2-3-3

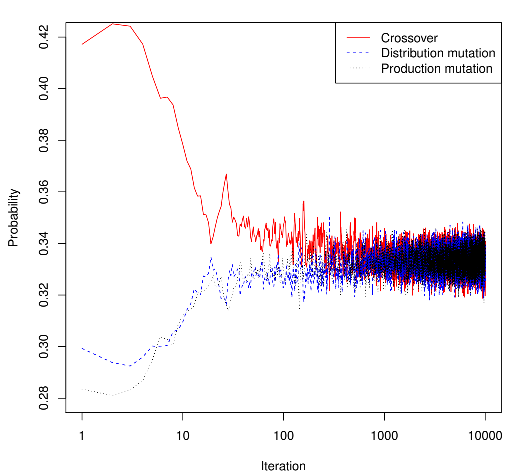
FIGURE A.5. Average evolution of fitness value of instances 2-2-3 (a), 2-2-4 (b), 2-2-5 (c), 2-2-6 (d), 2-2-7 (e) and 2-3-3 (f) for 30 computational experiments using 100 individuals and 10.000 iterations.



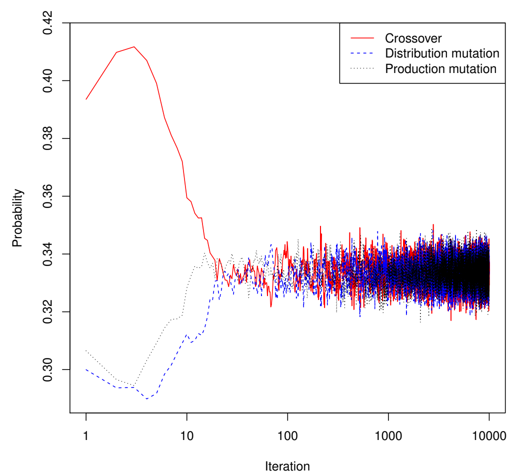
(a) Instance 2-3-4



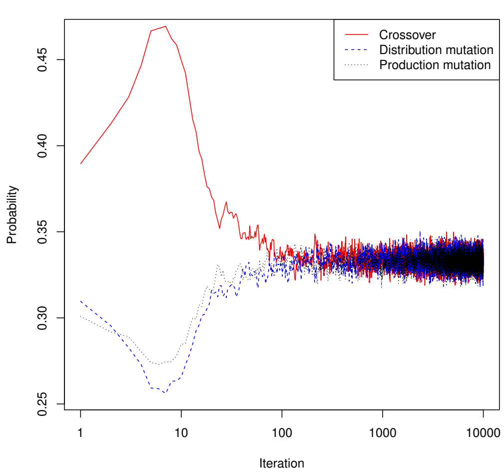
(b) Instance 2-3-8



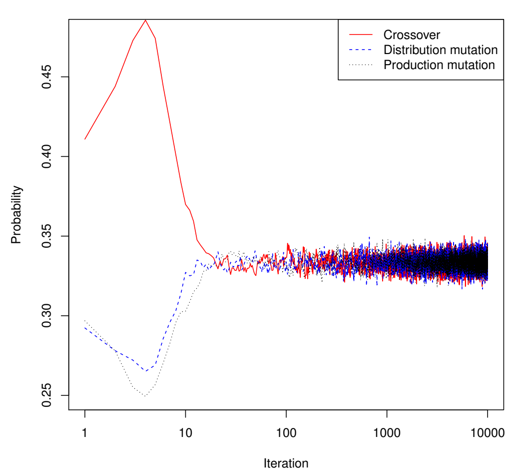
(c) Instance 2-4-8



(d) Instance 3-2-4

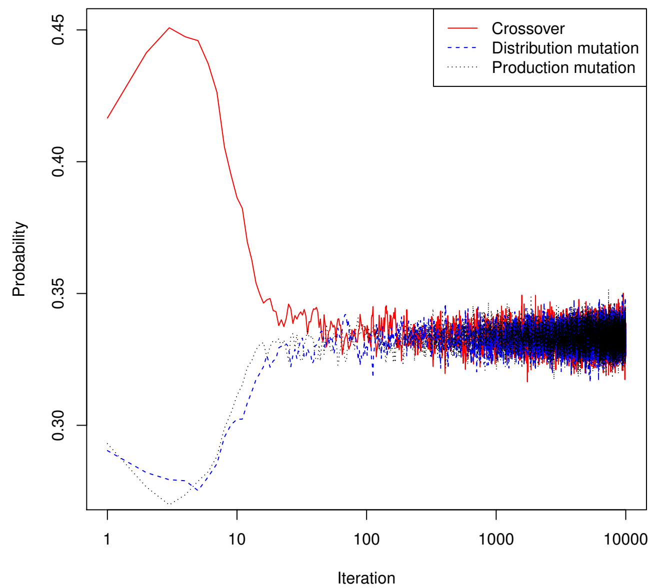


(e) Instance 3-2-5

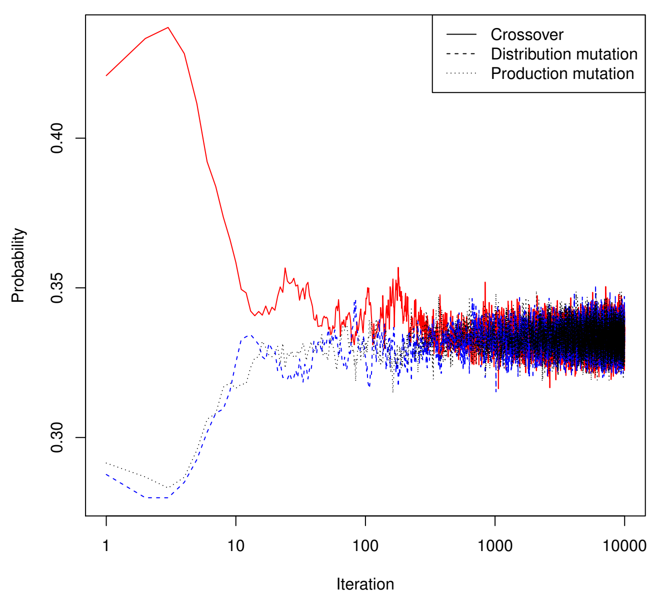


(f) Instance 3-3-6

FIGURE A.6. Average evolution of fitness value of instances 2-3-4 (a), 2-3-8 (b), 2-4-8 (c), 3-2-4 (d), 3-2-5 (e) and 3-3-6 (f) for 30 computational experiments using 100 individuals and 10.000 iterations.



(a) Instance 3-3-7b



(b) Instance 4-3-5

FIGURE A.7. Average evolution of fitness value of instances 3-3-7b (a) and 4-3-5 (b) for 30 computational experiments using 100 individuals and 10.000 iterations.

Conclusions

- The results obtained by the Hybrid Adaptive Evolutionary Algorithm (HaEa), combined with the proposed operators, as well as the representation of the network as individual using matrices, have shown that the the proposed technique is capable of finding optimal and near optimal solutions for the two-stage fixed charge transportation problem.

- Comparison analysis of the results obtained by HaEa showed that the evolutionary algorithm is very competitive with respect to metaheuristics as Simulated annealing, Genetic algorithms, Ant colony optimization, Artificial immune system and Sheep flock algorithm.

- The proposed network representation as individual using a matrix for each stage of the problem and arrays to manage de balance of the network have proven to be a good choice given the results obtained by HaEa and the execution time required to find such results using different population sizes and different number of iterations.

- Execution times of HaEa using other representation were higher compared to the proposed network representation, reducing execution time of a computational experiment from an hour and a half to half a hour using one hundred individuals and ten thousand iterations.

- Analysis of the evolution of the operator rates and the evolution of the population during the optimization process showed the capabilities of HaEa and the proposed genetic operators. During the optimization process, using ten thousand iterations, it took HaEa approximately 100 iterations to converge to good solutions. Operator rates show that HaEa used the crossover operator in a greater extent to converge to good solutions and combined this operator together with the production mutation and the distribution mutation to find better solutions in subsequent iterations.

- The statistical analysis of the results not only demonstrated that the algorithm could improve the solutions found by other methods of solution, but also demonstrated that, given more resources (fitness evaluations), the algorithm is able to continue improving the solutions found.

- The Literature review shows the unfair way in which different researchers have been comparing their results. For this reason, in this Thesis was decided to obtain results using different population sizes and different numbers of iterations.

Future work

Literature found on the solution of the fixed charge transportation problem suggest future research that aims for the solution of larger and more complex models that are closer to reality. Nowdays,fixed charge transportation problems of more than two stages, that considers several types of capacities, that transport multiple products are not enough. For this reason this Thesis suggest some areas for future work:

- Future research can be aimed at solving multiobjective models that describe the transportation of product in a more realistic way.
- Consider the Hybrid Adaptive Evolutionary Algorithm (HAEA) to tackle other problems related to supply chains and the transportation of goods.
- improvement of the proposed genetic operators as well as the creation of new operators capable of improving existing solutions for the two stage fixed charge transportation problem and other problems alike.

Bibliography

[1] Veena Adlakha and Krzysztof Kowalski, *A simple heuristic for solving small fixed-charge transportation problems*, Omega **31** (2003), no. 3, 205–211.

[2] ———, *A Simple Algorithm for the Source-Induced Fixed-Charge Transportation Problem*, The Journal of the Operational Research Society **55** (2004), no. 12, 1275–1280.

[3] Veena Adlakha, Krzysztof Kowalski, R Vemuganti, and Benjamin Lev, *More-for-less algorithm for fixed-charge transportation problems*, Omega **35** (2006), no. 1, 116–127.

[4] K. Antony Arokia Durai Raj and Chandrasekharan Rajendran, *Fast heuristic algorithms to solve a single-stage Fixed-Charge Transportation Problem*, International Journal of Operational Research **6** (2009), no. 3, 304.

[5] ———, *A genetic algorithm for solving the fixed-charge transportation model: Two-stage problem*, Computers and Operations Research **39** (2012), no. 9, 2016–2032.

[6] A N Balaji and N Jawahar, *A Simulated Annealing Algorithm for a two-stage fixed charge distribution problem of a Supply Chain*, International Journal of Opera **7** (2010), no. 2, 192–215.

[7] Michel L. Balinski, *Fixed-cost transportation problems*, Naval Research Logistics Quarterly **8** (1961), no. 1, 41–54.

[8] Richard S. Barr, Fred Glover, and Darwin Klingman, *A New Optimization Method for Large Scale Fixed Charge Transportation Problems*, Operations Research **29** (1981), no. 3, 448–463 (en).

[9] Gavin J. Bell, Bruce W. Lamar, and Chris A. Wallace, *Capacity improvement, penalties, and the fixed charge transportation problem*, Naval Research Logistics **46** (1999), no. 4, 341–355.

[10] A. Victor Cabot and S. Selcuk Erenguc, *Some branch-and-bound procedures for fixed-cost transportation problems*, Naval Research Logistics Quarterly **31** (1984), no. 1, 145–154.

[11] Herminia I. Calvete, Carmen Galé, and José A. Iranzo, *An improved evolutionary algorithm for the two-stage transportation problem with fixed charge at depots*, OR Spectrum **38** (2016), no. 1, 189–206.

[12] F. Chun and Z. Yi, *A genetic algorithm of two-stage supply chain distribution problem associated with fixed charge and multiple transportation modes*, 2009 Fifth International Conference on Natural Computation, vol. 4, Aug 2009, pp. 76–80.

[13] Leon Cooper and C Drebes, *An approximate solution method for the fixed charge problem*, Naval Research Logistics Quarterly **14** (1967), no. 1, 101–113.

[14] George B Dantzig, *Application of the simplex method to a transportation problem*, Activity Analysis of Production and Allocation–Proceedings of a Conference (Tj. C. Koopmans, ed.), Wiley, New York, 1951, pp. 359–373.

[15] David R Denzler, *An approximative algorithm for the fixed charge problem*, Naval Research Logistics Quarterly **16** (1969), no. 3, 411–416.

[16] Moustapha Diaby, *Successive Linear Approximation Procedure for Generalized Fixed-Charge Transportation Problems*, The Journal of the Operational Research Society **42** (1991), no. 11, 991–1001.

[17] Norman J Driebeek, *An algorithm for the solution of mixed integer programming problems*, Management Science **12** (1966), no. 7, 576–587.

[18] Paul S Dwyer, *Use of completely reduced matrices in solving transportation problems with fixed charges*, Naval Research Logistics Quarterly **13** (1966), no. 3, 289–313.

[19] Christoph Eckert and Jens Gottlieb, *Direct representation and variation operators for the fixed charge transportation problem*, pp. 77–87, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[20] Sandra Duni Ekşioğlu, H. Edwin Romeijn, and Panos M. Pardalos, *Cross-facility management of production and transportation planning problem*, Computers & Operations Research **33** (2006), no. 11, 3231 – 3251, Part Special Issue: Operations Research and Data Mining.

[21] Mahmoud M. El-Sherbiny and Rashid M. Alhamali, *A hybrid particle swarm algorithm with artificial immune learning for solving the fixed charge transportation problem*, Computers & Industrial Engineering **64** (2013), no. 2, 610–620.

[22] Mahmoud Moustafa El-Sherbiny, *Alternate mutation based artificial immune algorithm for step fixed charge transportation problem*, Egyptian Informatics Journal **13** (2012), no. 2, 123–134.

[23] John Fisk and Patrick McKeown, *The pure fixed charge transportation problem*, Naval Research Logistics Quarterly **26** (1979), no. 4, 631–641.

[24] LR Ford Jr and Delbert Ray Fulkerson, *Solving the transportation problem*, Management Science **3** (1956), no. 1, 24–32.

[25] M. Gen and R. Cheng, *Genetic algorithms and engineering design*, A Wiley Interscience publication, Wiley, 1997.

[26] M. Gen, K. Ida, and Yinzhen Li, *Solving bicriteria solid transportation problem by genetic algorithm*, Systems, Man, and Cybernetics, 1994. Humans, Information and Technology., 1994 IEEE International Conference on, vol. 2, Oct 1994, pp. 1200–1207 vol.2.

[27] Mitsuo Gen, Fulya Altiparmak, and Lin Lin, *A genetic algorithm for two-stage transportation problem using priority-based encoding*, OR Spectrum **28** (2006), no. 3, 337–354.

[28] Mitsuo Gen and Yin-Zhen Li, *Spanning tree-based genetic algorithm for bicriteria transportation problem*, Computers & Industrial Engineering **35** (1998), no. 3, 531 – 534.

[29] Mitsuo Gen and Yinzhen Li, *Spanning tree-based genetic algorithm for the bicriteria fixed charge transportation problem*, Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, vol. 3, IEEE, 1999.

[30] Michel Gendreau and Jean-Yves Potvin, *Handbook of metaheuristics*, vol. 2, Springer, 2010.

[31] A. M. Geoffrion and G. W. Graves, *Multicommodity Distribution System Design by Benders Decomposition*, Management Science **20** (1974), no. 5, 822–844 (en).

[32] Joseph Geunes and Panos M Pardalos, *Supply chain optimization*, vol. 98, Springer Science & Business Media, 2006.

[33] Farhad Ghassemi Tari and Zahra Hashemi, *A priority based genetic algorithm for nonlinear transportation costs problems*, Computers & Industrial Engineering **96** (2016), 86–95.

[34] Fred Glover, *Parametric Ghost Image Processes for Fixed-Charge Problems: A Study of Transportation Networks*, Journal of Heuristics **11** (2005), no. 4, 307–336.

[35] Jonatan Gomez, *Self adaptation of operator rates in evolutionary algorithms*, Genetic and Evolutionary Computation Conference, Springer, 2004, pp. 1162–1173.

[36] Teofilo F Gonzalez, *Handbook of approximation algorithms and metaheuristics*, CRC Press, 2007.

[37] Jens Gottlieb and Christoph Eckert, *A Comparison of Two Representations for the Fixed Charge Transportation Problem*, Parallel Problem Solving from Nature – {PPSN VI} (2000), 345–359.

[38] Jens Gottlieb and Lutz Paulmann, *Genetic algorithms for the fixed charge transportation problem*, Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, IEEE, 1998, pp. 330–335.

[39] Paul Gray, *Exact Solution of the Fixed-Charge Transportation Problem*, Operations Research **19** (1968), no. 6, 1529–1538.

[40] G. M. Guisewite and P. M. Pardalos, *Minimum concave-cost network flow problems: Applications, complexity, and algorithms*, Annals of Operations Research **25** (1990), no. 1, 75–99.

[41] M. Hajiaghaei-Keshteli, S. Molla-Alizadeh-Zavardehi, and R. Tavakkoli-Moghaddam, *Addressing a nonlinear fixed-charge transportation problem using a spanning tree-based genetic algorithm*, Computers and Industrial Engineering **59** (2010), no. 2, 259–271.

[42] Y. T. Herer, M. J. Rosenblatt, and I. Hefter, *Fast Algorithms for Single-Sink Fixed Charge Transportation Problems with Applications to Manufacturing and Transportation*, Transportation Science **30** (1996), no. 4, 276–290 (en).

[43] K.S. Hindi, T. Basta, and K. Pieńkosz, *Efficient solution of a multi-commodity, two-stage distribution problem with constraints on assignment of customers to distribution centres*, International Transactions in Operational Research **5** (1998), no. 6, 519 – 527.

[44] Warren M. Hirsch and George B. Dantzig, *Notes on linear programming: the fixed charge problem*, Rand Corp., Santa Monica Calif., 1955.

[45] ______, *The fixed charge problem*, Naval Research Logistics Quarterly **15** (1968), no. 3, 413–424.

[46] Frank L. Hitchcock, *The Distribution of a Product from Several Sources to Numerous Localities*, Journal of Mathematics and Physics **20** (1941), no. 1-4, 224–230.

[47] Michael H Hugos, *Essentials of supply chain management*, vol. 62, John Wiley & Sons, 2011.

[48] Tim H. Hultberg and Domingos M. Cardoso, *The teacher assignment problem: A special case of the fixed charge transportation problem*, European Journal of Operational Research **101** (1997), no. 3, 463–473.

[49] N. Jawahar and A. N. Balaji, *A genetic algorithm for the two-stage supply chain distribution problem associated with a fixed charge*, European Journal of Operational Research **194** (2009), no. 2, 496–537.

[50] Jung-Bok Jo, Yinzhen Li, and Mitsuo Gen, *Nonlinear fixed charge transportation problem by spanning tree-based genetic algorithm*, Computers & Industrial Engineering **53** (2007), no. 2, 290 – 298.

[51] Devika Kannan, Kannan Govindan, and Hamed Soleimani, *Artificial immune system and sheep flock algorithms for two-stage fixed-charge transportation problem*, Optimization **63** (2014), no. 10, 1465–1479.

[52] G. Kannan, P. Senthil, P. Sasikumar, and V. P. Vinay, *A Nelder and Mead methodology for solving small fixed-charge transportation problems*, Innovations in Supply Chain Management for Information Systems: Novel Approaches (John Wang, ed.), IGI Global, jan 2008 (English).

[53] Jeff Kennington and Ed Unger, *A New Branch-and-Bound Algorithm for the Fixed-Charge Transportation Problem*, Management Science **22** (1976), no. 10, 1116–1126 (en).

[54] Burcu B. Keskin and Halit Üster, *Meta-heuristic approaches with memory and evolution for a multi-product production/distribution system design problem*, European Journal of Operational Research **182** (2007), no. 2, 663 – 682.

[55] Dukwon Kim and Panos M. Pardalos, *A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure*, Operations Research Letters **24** (1999), no. 4, 195–203.

[56] Hak-Jin Kim and John N Hooker, *Solving fixed-charge network flow problems with a hybrid optimization and constraint programming approach*, Annals of Operations Research **115** (2002), no. 1, 95–124.

[57] Andreas Klose, *Algorithms for solving the single-sink fixed-charge transportation problem*, Computers & Operations Research **35** (2008), no. 6, 2079–2092.

[58] Tjalling C. Koopmans, *Optimum utilization of the transportation system*, Econometrica **17** (1949), 136–146.

[59] Tjalling C Koopmans and Stanley Reiter, *A model of transportation*, Activity Analysis of Production and Allocation–Proceedings of a Conference (Tj. C. Koopmans, ed.), Wiley, New York, 1951, pp. 222–259.

[60] Krzysztof Kowalski and Benjamin Lev, *On step fixed-charge transportation problem*, Omega **36** (2008), no. 5, 913–917.

[61] Harold W Kuhn and William J Baumol, *An approximative algorithm for the fixed-charges transportation problem*, Naval Research Logistics Quarterly **9** (1962), no. 1, 1–15.

[62] Bruce W Lamar and Chris A Wallace, *Revised-Modified Penalties for Fixed Charge Transportation Problems*, Management Science **43** (1997), no. 10, 1431–1436.

[63] Yinzhen Li, Kenichi Ida, and Mitsuo Gen, *Improved genetic algorithm for solving multiobjective solid transportation problem with fuzzy numbers*, Computers & Industrial Engineering **33** (1997), no. 3-4, 589–592.

[64] Linzhong Liu, Xinfeng Yang, Haibo Mu, and Yonglan Jiao, *The fuzzy fixed charge transportation problem and genetic algorithm*, Fuzzy Systems and Knowledge Discovery, 2008. FSKD'08. Fifth International Conference on, vol. 5, IEEE, 2008, pp. 208–212.

[65] M. M. Lotfi and R. Tavakkoli-Moghaddam, *A genetic algorithm using priority-based encoding with new operators for fixed charge transportation problems*, Applied Soft Computing Journal **13** (2013), no. 5, 2711–2726.

[66] Alfredo Marín and Blas Pelegrín, *A branch-and-bound algorithm for the transportation problem with location of p transshipment points*, Computers & operations research **24** (1997), no. 7, 659–678.

[67] Patrick G. McKeown, *A branch-and-bound algorithm for solving fixed charge problems*, Naval Research Logistics Quarterly **28** (1981), no. 4, 607–617.

[68] Zbigniew Michalewicz, George A. Vignaux, and Matthew Hobbs, *A Nonstandard Genetic Algorithm for the Nonlinear Transportation Problem*, ORSA Journal on Computing **3** (1991), no. 4, 307–316 (en).

[69] S. Molla-Alizadeh-Zavardehi, M. Hajiaghaei-Keshteli, and R. Tavakkoli-Moghaddam, *Solving a capacitated fixed-charge transportation problem by artificial immune and genetic algorithms with a prüfer number representation*, Expert Systems with Applications **38** (2011), no. 8, 10462 – 10474.

[70] Katta G. Murty, *Solving the Fixed Charge Problem by Ranking the Extreme Points*, Operations Research **16** (1968), no. 2, 268–279 (en).

[71] Francisco Ortega and Laurence A. Wolsey, *A branch-and-cut algorithm for the single-commodity, uncapacitated, fixed-charge network flow problem*, Networks **41** (2003), no. 3, 143–158.

[72] Udatta S. Palekar, Mark H. Karwan, and Stanley Zionts, *A Branch-and-Bound Method for the Fixed Charge Transportation Problem*, Management Science **36** (1990), no. 9, 1092–1105 (en).

[73] V. Panicker and R. Sridharan, *Development and analysis of heuristic algorithms for a two-stage supply chain allocation problem with a fixed transportation cost*, International Journal of Services and Operations Management **12** (2012), no. 2, 244.

[74] Vinay V. Panicker, Ratnaji Vanga, and R. Sridharan, *Ant colony optimisation algorithm for distribution-allocation problem in a two-stage supply chain with a fixed transportation charge*, International Journal of Production Research **51** (2013), no. 3, 698–717.

[75] C.-M. Pintea and P.C. Pop, *An improved hybrid algorithm for capacitated fixed-charge transportation problem*, Logic Journal of the IGPL **23** (2014), no. 3, 369–378.

[76] Camelia-M. Pintea, Corina Pop Sitar, Mara Hajdu-Macelaru, and Pop Petrica, *A Hybrid Classical Approach to a Fixed-Charged Transportation Problem*, Hybrid Artificial Intelligent Systems (Emilio Corchado, Václav Snášel, Ajith Abraham, Michał Woźniak, Manuel Graña, and Sung-Bae Cho, eds.), Lecture Notes in Computer Science, vol. 7208, Springer Berlin Heidelberg, Berlin, Heidelberg, mar 2012, pp. 557–566.

[77] Philip Robers and Leon Cooper, *A study of the fixed charge transportation problem*, Computers & Mathematics with Applications **2** (1976), no. 2, 125–135.

[78] Franz Rothlauf, *Design of modern heuristics: principles and application*, Springer Science & Business Media, 2011.

[79] Keith Sandrock, *A Simple Algorithm for Solving Small , Fixed-Charge Transportation Problems*, The Journal of the Operational Research Society **39** (1988), no. 5, 467–475.

[80] Joanne R. Schaffer and Daniel E. O'Leary, *Use of penalties in a branch and bound procedure for the fixed charge transportation problem*, European Journal of Operational Research **43** (1989), no. 3, 305–312.

[81] Susann Schrenk, Gerd Finke, and Van-Dat Cung, *Two classical transportation problems revisited pure constant fixed charges and the paradox*, Mathematical and Computer Modeling **54** (2011), no. 91-10, 2306 – 2315.

[82] David I. Steinberg, *The fixed charge problem*, Naval Research Logistics Quarterly **17** (1970), no. 2, 217–235.

[83] Minghe Sun, Jay E. Aronson, Patrick G. McKeown, and Dennis Drinka, *A tabu search heuristic procedure for the fixed charge transportation problem*, European Journal of Operational Research **106** (1998), no. 97, 441–456.

[84] Minghe Sun and Patrick G. McKeown, *Tabu search applied to the general fixed charge problem*, Annals of Operations Research **41** (1993), no. 4, 405–420.

[85] Admi Syarif, YoungSu Yun, and Mitsuo Gen, *Study on multi-stage logistic chain network: A spanning tree-based genetic algorithm approach*, Computers and Industrial Engineering **43** (2002), no. 1-2, 299–314.

[86] G. A. Vignaux and Z. Michalewicz, *A genetic algorithm for the linear transportation problem*, IEEE Transactions on Systems, Man, and Cybernetics **21** (1991), no. 2, 445–452.

[87] Don D. Wright and Christoph Haehling von Lanzenauer, *Solving the fixed charge problem with Lagrangian relaxation and cost allocation heuristics*, European Journal of Operational Research **42** (1989), no. 3, 305–312.

[88] Fanrong Xie and Renan Jia, *Nonlinear fixed charge transportation problem by minimum cost flow-based genetic algorithm*, Computers and Industrial Engineering **63** (2012), no. 4, 763–778.

[89] Lixing Yang and Linzhong Liu, *Fuzzy fixed charge solid transportation problem and algorithm*, Applied Soft Computing **7** (2007), no. 3, 879–889.

[90] Xiaofeng Yang and Mitsuo Gen, *Evolution program for bicriteria transportation problem*, Computers & Industrial Engineering **27** (1994), no. 1, 481–484.

[91] Xin-She Yang, *Artificial intelligence, evolutionary computing and metaheuristics: in the footsteps of alan turing*, vol. 427, Springer, 2012.

[92] YoungSu Yun, Chiung Moon, and Daeho Kim, *Hybrid genetic algorithm with adaptive local search scheme for solving multistage-based supply chain problems*, Computers & Industrial Engineering **56** (2009), no. 3, 821–838.