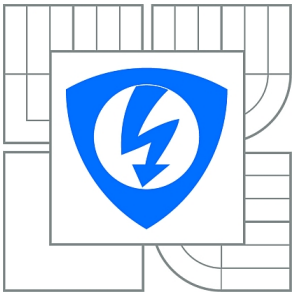
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF RADIO ELECTRONICS

EVOLUTIONARY SYNTHESIS OF ANALOG ELECTRONIC CIRCUITS USING EDA ALGORITHMS

EVOLUČNÍ SYNTÉZA ANALOGOVÝCH ELEKTRONICKÝCH OBVODŮ S VYUŽITÍM ALGORITMŮ EDA

DIZERTAČNÍ PRÁCE

DOCTORAL THESIS

AUTOR PRÁCE — Ing. JOSEF SLEZÁK

AUTHOR

VEDOUCÍ PRÁCE — Prof. Ing. Tomáš Dostál, DrSc.

SUPERVISOR

BRNO 2014

ABSTRAKT

Disertační práce je zameřena na návrh analogových elektronických obvodů pomocí algoritmů s pravděpodobnostními modely (algoritmy EDA). Prezentované metody jsou na základě požadovaných charakteristik cílových obvodů schopny navrhnout jak parametry použitých komponent tak také jejich topologii zapojení. Tři ruzné metody využití EDA algoritmů jsou navrženy a otestovány na příkladech skutečných problemů z oblasti návrhu analogových elektronických obvodů.

První metoda je určena pro návrh pasivních analogových obvodů a využívá algorithmus UMDA pro návrh jak topologie zapojení tak také hodnot parametrů použitých komponent. Metoda je použita pro návrh admitanční sítě s požadovanou vstupní impedancí pro účely chaotického oscilátoru.

Druhá metoda je také určena pro návrh pasivních analogových obvodů a využívá hybridní přístup - UMDA pro návrh topologie a metodu lokální optimizace pro návrh parametrů komponent.

Třetí metoda umožňuje návrh analogových obvodů obsahujících také tranzistory. Metoda využívá hybridní přístup - EDA algoritmus pro syntézu topologie a metoda lokální optimalizace pro určení parametrů použitých komponent. Informace o topologii je v jednotlivých jedincích populace vyjádřena pomocí grafů a hypergrafů.

KLÍČOVÁ SLOVA

Automatizovaná syntéza analogových elektronických obvodů, návrh analogových elektronických obvodů, evoluční algoritmus, evoluční elektronika, EDA, UMDA, chaotický oscilátor, fraktální kapacitor, pravděpodobnostní model.

ABSTRACT

Dissertation thesis is focused on design of analog electronic circuits using Estimation of Distribution Algorithms (EDA). Based on the desired characteristics of the target circuits the proposed methods are able to design the parameters of the used components and theirs topology of connection as well. Three different methods employing EDA algorithms are proposed and verified on examples of real problems from the area of analog circuits design.

The first method is capable to design passive analog circuits. The method employs UMDA algorithm which is used for determination of the parameters of the used components and synthesis of the topology of their connection as well. The method is verified on the problem of design of admittance network with desired input impedance function which is used as a part of chaotic oscillator circuit.

The second method is also capable to design passive analog circuits. The method employs hybrid approach - UMDA for synthesis of the topology and local optimization method for determination of the parameters of the components.

The third method is capable to design analog circuits which include also active components such as transistors. Hybrid approach is used. The topology is synthesized using EDA algorithm and the parameters are determined using a local optimization method. In the individuals of the population information about the topology is represented using graphs and hypergraphs.

KEYWORDS

Automated synthesis of analog electronic circuits, design of analog electronic circuits, evolutionary algorithm, evolutionary electronics, EDA, UMDA, chaotic oscillator, fractional capacitor, probabilistic model.

SLEZÁK J. *Evolutionary Synthesis of Analog Electronic Circuits Using EDA Algorithms*. Brno: Brno University of Technology, The Faculty of Electrical Engineering and Communication, 2014. 123 pages, Doctoral thesis.

PROHLÁŠENÍ

Prohlašuji, že svou disertační práci na téma Evoluční syntéza analogových elektronických obvodů pomocí algoritmů EDA jsem vypracoval samostatně pod vedením školitele a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené disertační práce dále prohlašuji, že v souvislosti s vytvořením této disertační práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení 152 trestního zákona č. 140/1961 Sb.

V Brně dne .....

.....
(podpis autora)

DECLARATION

I declare that I have elaborate my doctoral thesis on Evolutionary Synthesis of Analog Electronic Circuits Using EDA Algorithms independently under supervision and using literature and other information sources, which are all cited in the work and listed in the bibliography at the end of work. I furthermore declare that, concerning the creation of this doctoral thesis, I did not infringe the copyrights of third parties, in particular, I have not infringed any copyright. In particular, I have not unlawfully encroached on anyones personal copyright and I am fully aware of the consequences in the case of breaking Regulation 11 and the following of the Copyright Act No 121/2000 Vol., including the possible consequences of criminal law resulted from Regulation 152 of Criminal Act No 140/1961 Vol.

In Brno on .....

.....
(author's signature)

PODĚKOVÁNÍ

Děkuji svému školiteli prof. Ing. Tomáši Dostálovi, DrSc. za vedení, odbornou pomoc a podporu při vypracování této práce. Dále děkuji doc. Ing. Jiřímu Petrželovi, Ph.D. a Ing. Romanu Šotnerovi, Ph.D. za cenné připomínky a diskuze a prof. Dr. Ing. Zbyňku Raidovi za vstřícnost během celého doktorského studia. Speciálně děkuji mojí rodině, Alexandře Mateáskové, Pradhanu Balterovi, Radku Šebelovi a Michalu Kuncovi za porozumění a podporu.

ACKNOWLEDGMENT

I would like to thank my supervisor, prof. Ing. Tomáš Dostál, DrSc., for his methodological, pedagogical and professional support. Thanks to doc. Ing. Jiří Petržela, Ph.D. and Ing. Roman Šotner, Ph.D. for valuable discussions and their comments. Thanks to prof. Dr. Ing. Zbyněk Raida for his helpfulness and support. Special thanks to my whole family for support during my doctoral study and to Alexandra Mateásková, Pradhan Balter, Radek Šebela and Michal Kunc for their help and understanding.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Although there is a trend of using digital processing in most areas of the modern electronics, analog electronic circuits are still important part of today's electronic integrated chips. There is variety of applications where using of analog circuit processing is advantageous compared to digital solutions or such digital solutions even do not exist.

Probably the most important application of analog circuits is to provide connection between digital circuits and analog signals of "outer world". Analog signals from variety of different sensors (mechanical tension, air pressure, magnetic field) can be transformed to digital signals using analog to digital converters. Similarly digital signals of digital circuits can be transformed to analog signals using digital to analog converters. Important application of analog circuits are frequency filters which are necessary during sampling process to avoid aliasing effect (anti-aliasing filters). Another traditional application of analog circuits are radio frequency (RF) circuits which can operate on frequencies of several GHz. Analog circuits can be used as a part of digital circuits to provide constant reference voltage (voltage reference circuits) with stable conditions independent on the parameters of the environment.

Design of analog circuits is traditionally domain of experienced designers and usually is viewed as a kind of art where important part is designer's intuition involved in the design process. Therefore analog circuit design is expensive and time consuming. As was discussed in [1] although digital parts in mixed-type SoCs integrated chips generally occupy for more than 90 percent of the chip area, the most design cost and effort is spent on design of the analog part which accounts usually only 10 percent of the area of the whole chip. Therefore there is an effort to automatize the design process of analog circuits using automated computer analog circuit design tools.

With increasing performance of modern computing and advances in the areas of artificial intelligence, machine learning and optimization there is possibility to take over considerable amount of the time consuming procedures required to design of analog electronics to modern computer methods called automated analog circuit design or evolutionary electronics.

Another motivation for usage of the evolutionary electronics methods is demand of synthesis of analog circuits for which no methods of classical analog design exist. There is a variety of problems of analog electronics which can be hardly or even by no means solved using classical analog design methods. For example synthesis of computational circuits as discussed in [57].

Process of automated analog circuit synthesis consists of two main tasks which are formation of suitable topology and sizing of the components. Topology can

be selected from a library of possible topologies or it can be generated during the synthesis process. Since selection of the suitable topology from a library has not brought satisfactory results the research in topology formation methods is now focused mainly on the topology generation approach.

Given a variety of desired specifications of the evolved circuit evolutionary electronics systems are able to synthesize different types of electronic circuits. In some cases evolutionary electronics method can produce strange circuit structures with non-standard topologies. Although the resulting circuit meets the desired specifications the way how the evolutionary synthesis reached the target specification in not really what the designer wanted. Such behavior can be unacceptable for example in evolutionary design of operational amplifiers or OTA circuits where the circuit topologies are well-established. Optimization task of such circuits usually consists only in optimization of the parameters of the used transistors (width, height) while topology is fixed. The goal of the optimization is to achieve desired parameters of the circuit as slow rate, bandwidth or dynamic range. The problem of generation of non-standard topologies can be solved using evolutionary electronics synthesis methods which use a library of building blocks (voltage sources, current sources, current mirrors). Such method is able to reach target circuits with topologies similar to classical hand analog circuits design [22], [23].

On the other hand there are some problems of analog circuit synthesis where no well established topologies exist. For example passive analog networks with arbitrary input impedance function [3] [4] [5] [6] or computational circuits [57]. In these cases creative potential of the evolutionary electronics methods can be fully utilized with no restrictions of the target topologies.

Although evolutionary electronics systems and design automation tools for digital electronic circuits are at very good useful level, in the area of analog circuits the tools are still not mature enough for regular usage. One of the reasons of this fact is that complex digital circuits are usually partitionable to lower complexity functional blocks which can be designed and optimized separately. This way the synthesis of a complex digital system can be separated into partial low complexity designs. Since analog circuits usually work as systems where all blocks are connected through feedback connections and strongly interact between each other partitioning of an analog circuits is difficult task. By disconnecting of the feedback connections in the circuit lower complexity blocks usually lose theirs function. This behavior of analog circuits results in requirement of simultaneous optimization of the whole circuits which is computationally very expensive task.

There have been presented variety of different approaches to handle automated analog circuit synthesis problem. Systems based on optimization and machine learning methods such as genetic algorithms, genetic programming, ant colony optimiza-

tion, graph grammar and others have been presented.

Recently Estimation of Distribution Algorithms (EDA) have shown theirs superior performance compared to classical genetic algorithms. Drawbacks of genetic algorithms such as requirement of tight linkage in the encoding, deceptive functions and disruption of the building blocks after the crossover operation can be overcome using EDA [26].

The presented thesis is focused on evolutionary design of low complexity and medium complexity passive and active analog electronic circuits using Estimation of Distribution Algorithms.

2 EVOLUTIONARY ELECTRONICS

2.1 Basic Principle

> *Imagine a design space where each point in that space represents the design of an electronic circuit. All possible electronic circuits are there, given the component types available to electronics engineer, and the technological restrictions on how many components there can be and how they can interact. In this metaphor, we loosely visualize the circuits to be arranged in the design space so that similar circuits are close to each other. The design space is vast. There are oscillators and filters, finite-state machines, analogue computers, parallel distributed systems, Von Neumann computers, and so on. These nestle amongst the majority of circuits for which a use will never be found.*

The chapter is introduced with metaphor [2] which perfectly depicts the main idea of evolutionary electronics synthesis methods. Electronic circuits are synthesized by means of search algorithms which try to find suitable solution in a given search space (Fig. 2.1a). Principal flow chart of the evolutionary electronics synthesis methods is presented in Fig. 2.1b.

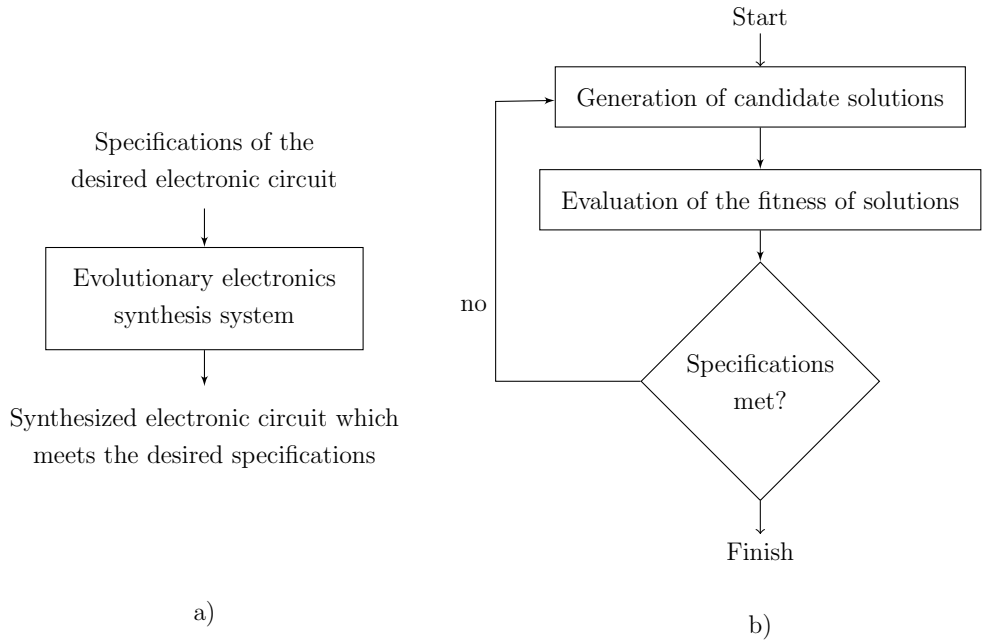


Fig. 2.1: a) Basic idea of evolutionary electronics system b) Principal flow chart of the evolutionary electronics methods.

As can be seen in Fig. 2.1b there are three basic phases of the program flow of the evolutionary electronics methods. In the first phase promising candidate solutions are generated. Various methods of local and global optimization, machine learning, artificial intelligence or theirs combinations can be used. In the next phase fitness values of the generated solutions are evaluated. In the last phase meeting of the desired specifications is tested. If any solution meeting the desired specifications has been found the whole evolutionary electronics system is terminated. Otherwise the whole process repeats.

2.2 Intrinsic and Extrinsic Systems

Based on the realization of the fitness evaluation phase evolutionary electronics systems can be intrinsic or extrinsic.

Intrinsic evolutionary electronics systems utilize real HW implementation in configurable device as for example FPTA (field transistor array). Generated solutions are directly implemented in the configurable device and its fitness value is calculated based on the measured results of the implemented solution.

Flow chart of evaluation of the fitness value in intrinsic systems is presented in Fig. 2.2a. Generated solution is transformed to a form suitable for configurable device and the corresponding circuit is implemented. Responses (voltage transfer, frequency response) of the configured device are measured and the fitness value is calculated.

The strong advantage of this solution is that circuit is implemented in real hardware including all parasitic influences. Also the speed of such fitness evaluation approach is very high therefore running time of intrinsic systems is much shorter than extrinsic systems. On the other hand intrinsic systems require specialized HW devices which are expensive compared to the extrinsic systems. Also the configurable devices are usually designed for specific small range of applications.

In the extrinsic systems evaluation of the fitness values is based on the responses obtained by simulation. Flow chart of evaluation of the fitness values in extrinsic systems is presented in Fig. 2.2b. Generated solution is transformed into netlist file suitable for circuit simulators and simulation is performed. Based on the results of the simulation fitness value of the solution is calculated.

The first most obvious advantage of the extrinsic systems compared to the intrinsic ones is that no additional HW is required. The extrinsic evolutionary synthesis systems can be used on any standard computer. Since the responses of the candidate solutions are obtained using circuit simulator the accuracy and reliability of evaluation of the fitness function fully depends on the quality of the used circuit

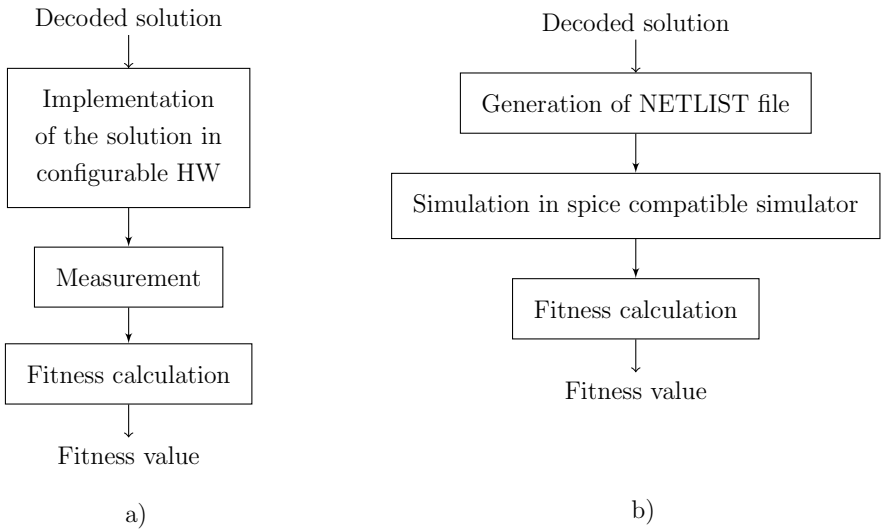


Fig. 2.2: Evaluation of the fitness value in the intrinsic system (a) and in the extrinsic system (b).

simulator. Usually spice compatible circuit simulators as hspice or ngspice are used. Although credibility of the modern circuit simulators is very high there can be still some differences between responses obtained using circuit simulator and responses obtained using real HW implementation in some cases.

Since the simulation run time of the circuit simulators is much longer than measurement in the intrinsic systems the extrinsic systems require much longer run time of the whole synthesis process.

2.3 Possible Strategies of Evolutionary Electronics Systems

Generally the automated analog circuits synthesis problem can be defined as two strongly related tasks. The first one is searching for the suitable topology and the second one is determining of the parameters of the components of the selected topology. Searching for the suitable topology can be defined as combinatorial optimization problem. Searching for the parameters of values can be defined as continuous or discrete continuous optimization problem.

Based on the used algorithm or combination of algorithms these two partial problems can be solved simultaneously or separately. In the case of simultaneous solving the whole problem of analog circuit synthesis is encoded into one encoding vector which includes information about the topology and the parameters of the components as well. Used optimization algorithm has to be able to deal with mixed

combinatorial/continuous types of problems. Mixed-type problems can be solved using simulated annealing as used in [3] [4] [5], messy genetic algorithm as used in [18] and others. Simultaneous solving of the topology and the components values results in lower running time of the whole synthesis process. On the other hand used encoding method has to be carefully designed to comply different demands of the both parts of the problem (topology and parameters). Another difficulty of the simultaneous solving is to selected, design or modify suitable optimization method which is able to handle such mixed type optimization problem.

The second approach is to separate the whole synthesis process into two partial tasks - searching for the topology and determining of the parameters. Although such strategy is usually more computationally expensive than simultaneous solving the advantage of this approach is simpler implementation and high variability of the whole method. There is no need of difficult design of suitable encoding and various methods of combinational optimization for searching of the topology can be used in combination with various methods of continuous optimization for searching of the parameters. Separate solving of the analog circuit synthesis problem was used in methods published for example in [22] or [23].

3 STATE OF THE ART

The first papers focused on the subject of the evolutionary electronics synthesis methods and analog circuit design automation were published in late eighties of 20th century. In the section review of the evolutionary electronics methods which are able to synthesize whole analog circuits (the topology and the parameters of the components) is presented.

One of the first attempts to handle the problem of the analog circuits design automation was published by Degrauwe et al. in [7] where system IDAC was described. In the first step the designer specifies the technology, desired building-block parameters, design options and one or more schematics from the program's library. After performing of the synthesis results as frequency responses, SPICE2 input file, input file for layout program ILAC [7] or input file for PPR-G [7] representing the datasheet can be obtained.

In [8] Harjani described tool OASYS which uses design plans to successively select the topologies and translate specifications in a knowledge based synthesis framework.

Another system called OPASYN which is suitable for synthesizing of CMOS transistor structures was presented by Koh et al. in [9]. Similar tool called DARWIN was presented by Kruiskamp et al. in [10]. Synthesis of VLSI circuits using parallel genetic algorithm system consisted of twenty distributed SPARC workstations was presented by Davis et al. in [11].

The pioneer of genetic programming (GP) J. Koza employed GP method in automated analog circuit synthesis system where analog electronic circuits are encoded using tree structures [31]. Despite extensive effort of J. Koza et al. evolutionary electronics systems based on genetic programming are computationally very expensive.

Another research line of the earliest works was focused on analog circuits design automation based on genetic algorithms (GA). Grimbleby have published several papers focused on employing of hybrid GA and direct encoding method for synthesis of passive analog circuits [13] [14]. The synthesis was performed in two steps. In the first one the topology was selected and its simulatability was verified using symbolic calculation routine. In the second step the parameters (values of the components) were determined using a numerical optimization method. As was stated in [13] computational cost of this approach was much lower than similar system based on GP.

Another attempt to handle analog circuit synthesis automation problem by means of GA was presented by Lohn and Colombano in [15]. The method uses simultaneous synthesis of the topology and the parameters approach. Compared

to the previous work presented by Grimbleby analog circuits are encoded using developmental encoding. The basic principle of developmental encoding is to encode sequence of circuit-building instructions (OP codes) which construct the topology of the circuit. The motivation for using developmental encoding method was demand to decrease number of dead (nonsimulatable) individuals created after recombination phase of the used GA. On the other hand developmental encoding can restrict possible encodable analog circuit topology in some cases.

More advanced approach of synthesis of passive and also active analog circuits was proposed by Zebulum who employed GA with variable chromosome representation [16]. Besides the main chromosome vector containing the analog circuit structure information the GA utilizes also mask vector which is used to define coding and noncoding segments of the main chromosome. There were proposed three approaches called ILG, OLG and UDIP to manipulate the bits of the mask vector. The method was also used for unconstrained synthesis of the active QR networks [17].

System employing GA with variable length chromosome was used in [18]. The paper also discusses modification of the parameters of the evolved analog circuit by addition of random noise. As was discussed in [18] this approach tries to model impact of the environment and guides the search to the topologies with higher resistivity to variation of the parameters (topologies with lover sensitivity).

Compared to the previous works where only synthesis in the frequency domain was handled. Paper [19] is focused on synthesis of sinusoidal oscillator circuit in time domain.

Mattiussi proposed method called analog genetic encoding (AGE) which is able to synthesize analog circuits and neural networks [20]. The system employs encoding method based on biological chromosomes principles. The proposed method was verified on a number of analog circuit synthesis problems [20].

Das and Vemuri have proposed several methods for automated analog circuit synthesis. The first method called GAPSYS was able to synthesis only passive analog circuits [21]. Another two methods separate the analog circuits synthesis task into two partial processes. Synthesis of the topology and sizing of the components. In the method presented in [22] selection of the topology is realized using adaptively generated building blocks. Evolutionary electronics synthesis method using graph grammar based approach was presented in [23].

Analog circuits encoding method based on adjacency matrix representation and special type of crossover were presented by Mesquite et al. in [24]. Compared to the incidence matrix representation the proposed method is able to preserve topologies of both parental circuits and to connect them in meaningful way through subset of nodes [24]. The adjacency matrix encoding method was used for synthesis of

low-sensitivity antenna matching network [25].

Recently Estimation of Distribution Algorithms (EDA) have shown theirs superior performance compared to classical genetic algorithms [26] . Univariate Marginal Distribution Algorithm (UMDA) [27] which is the simplest version of EDA was employed in evolutionary electronics system presented by Zinchenko [28]. The proposed system was verified on the problem of synthesis of low pass filter. Application of UMDA in mixed analog-digital circuits design system was described by Zinchenko in [29]. Another application of UMDA in automated analog circuits synthesis method was presented by Torres in [30].

The goal of the thesis is to design and verify three types of automated analog circuits synthesis methods based on Estimation of Distribution Algorithms. Synthesis capability of every method will be verified on an example of analog circuit synthesis problem.

4 REVIEW OF THE ENCODING METHODS

Besides the optimization method choosing or design of the suitable encoding approach is one of the key requirements of the success of the optimization. In the text bellow there will reviewed different types of encoding methods of analog circuits as were presented in literature focused on evolutionary analog circuit synthesis.

4.1 Direct Encoding Method

The most simple and straightforward method of the analog circuit encoding is direct encoding method. The principle of the approach will be demonstrated on the example of encoding of passive circuit including RLC components. Encoding vector e is formed of n integer numbers. Every component of the encoded circuit is described by sequence of 5 integer variables defining its type (variable v_{type}), connection nodes (n_1 and n_2), exponent (v_{exp}) and mantissa (v_{man}). Therefore maximal complexity of the encoded circuit which is encoded by vector e of length n is $n/5$. The sequence of the five mentioned definition variables is ordered as presented in Fig. 4.1a. An example of an analog circuit structure and its corresponding encoding vector e are presented in Fig. 4.1b and Fig. 4.1c respectively.

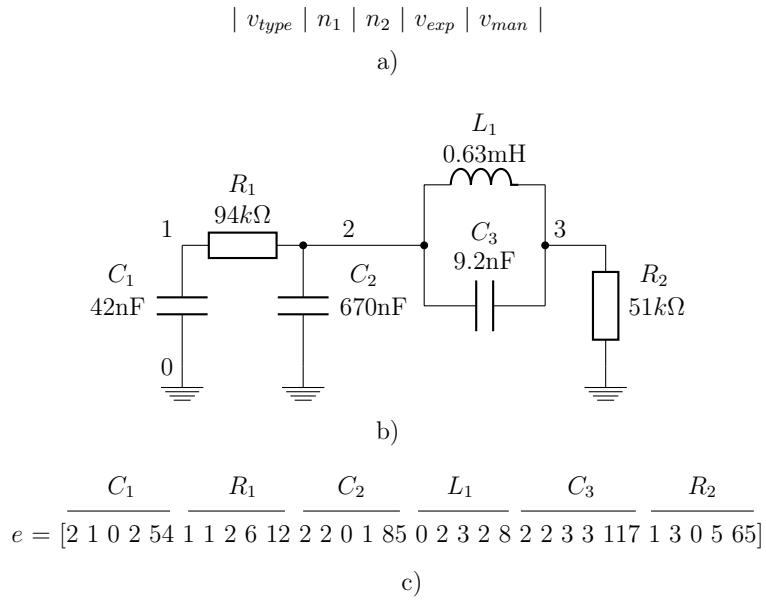


Fig. 4.1: a) Encoding variables sequence b) example of an analog circuit c) encoding vector of the circuit

Assignment of the values of variable v_{type} to the used types of the components is summarized in Tab. 4.1.

component type	inductor	resistor	capacitor
v_{type}	0	1	2

Tab. 4.1: Values of variable v_{type} for different types of the components.

Value of every component is calculated as $(v_{man}/128).v_{mult}$, where v_{man} is value of mantissa encoded in the encoding vector e and v_{mult} is multiplier which is assigned according to v_{exp} and v_{type}. Value of mantissa is selected from integer numbers in the range $\langle 1, 128 \rangle$. All possible values of variable v_{mult} are summarized in Tab. 4.2.

v_{exp}	0	1	2	3	4	5	6	7
$v_{type} = 0$ (inductor)	10^0	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
$v_{type} = 1$ (resistor)	10^0	10^1	10^2	10^3	10^4	10^5	10^6	10^7
$v_{type} = 2$ (capacitor)	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}	10^{-11}	10^{-12}

Tab. 4.2: Assignment of variable multiplier v_{mult}.

Although only passive components were used in the presented example the method is able to encode various active types of components such as transistors, current conveyors or operational amplifiers.

Direct encoding method is very simple, straightforward and easy to use. On the other hand there are some drawbacks which have to be taken into account in case of using simple genetic algorithm. As was discussed in [24] using direct encoding method and standard recombination operators such as single point crossover, two points crossover or bit-wise mutation usually results in producing high number of low-fitness individuals what is caused by generating of high number of unsimulatable topologies (unconnected components, unconnected branches). There have been presented a number of methods to overcome this problem. For example J. Grimbleby has presented a special type of mutation where the modification of the topology is performed in nondestructive way [13]. Another method is using of developmental encoding method or adjacency matrix representation what will be described in the next sections.

4.2 Parse Tree Representation

Genetic programming (GP) is evolutionary algorithm based approach where each individual of the evolved population is a computer program. In terms of evolutionary electronics synthesis every individual is defined as sequence of circuit constructing and circuit modifying instructions. The major research of GP was made by J. Koza et al. [31] [32] [33] [34].

In GP the sequence of the instructions is represented as a parse tree structure where every node defines an operator function and every terminal node defines an operand.

An example of the parse tree representation and corresponding circuit structure are presented in Fig. 4.2a and Fig. 4.2b respectively [12]. As can be seen in Fig. 4.2a two types of circuit-constructing instructions are used in the representation. The first function denoted as "+" represents adding of a component in series. The second function denoted as "//" represents adding of a component in parallel.

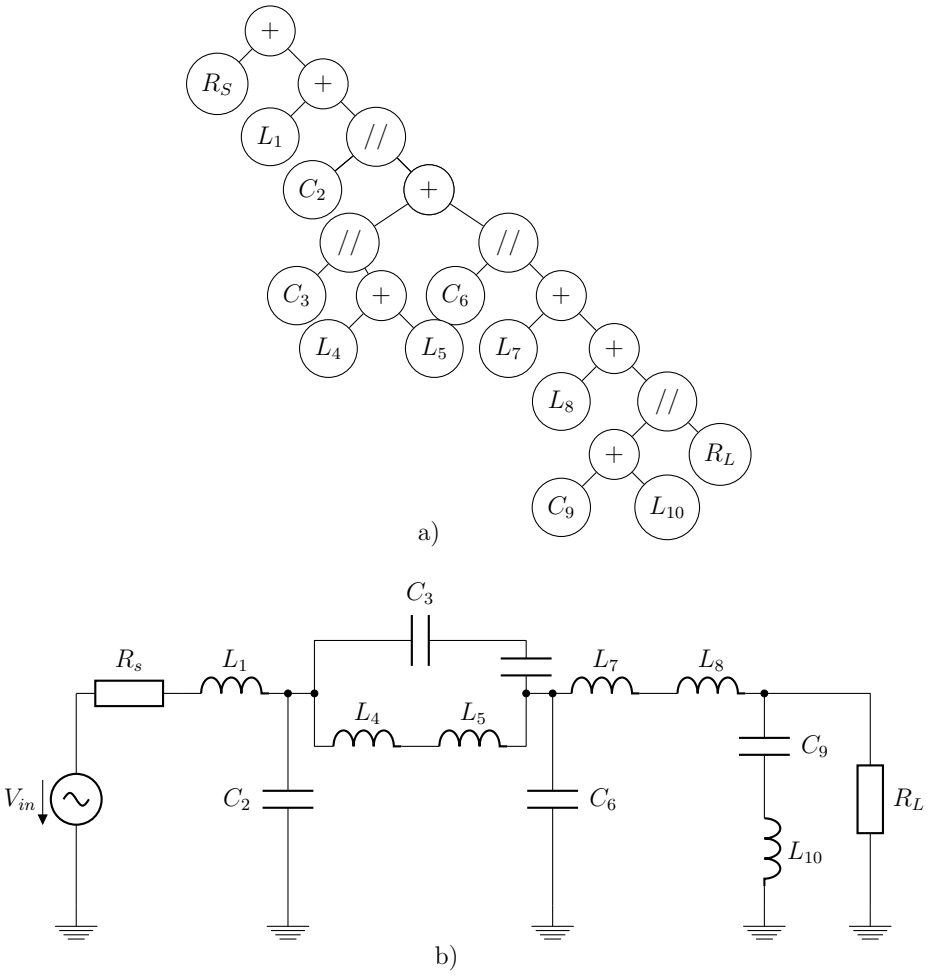


Fig. 4.2: a) Example of parse tree representation b) encoded circuit.

On the contrary to direct encoding method, parse tree representation is developmental method which means that the circuit is constructed progressively using sequence of circuit construction instructions. The process of the circuit synthesis begins at basic circuit structure called embryonic circuit. Its structure depends on the particular problem. An example of single-input and single-output embryonic circuit which can be used for example for synthesis of frequency filter is presented in Fig. 4.3.

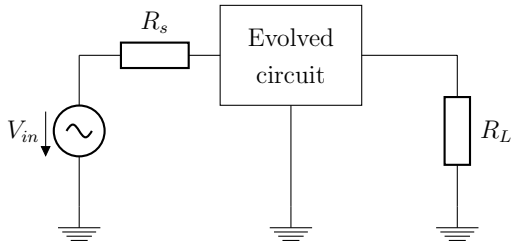


Fig. 4.3: Structure of the embryonic circuit.

Component-creating and connection-modifying functions were introduced in [31]. List of the component-creating functions and the connection-modifying functions which were described in the mentioned paper are presented in Tab. 4.3 and Tab. 4.4 respectively.

Name	number of arguments	Function
C	2	insert capacitor
L	3	insert inductor
QT0..GT11	3	insert transistor

Tab. 4.3: Component-creating functions.

Name	number of arguments	Function
FLIP	1	reversing of polarity
SERIES	3	insert component in series
PSS	4	insert component in parallel
VIA0..VIA7	2	connect distant parts of the circuit
GND	2	connect to the ground node
NOP	1	empty instruction
END	0	loose its writing head

Tab. 4.4: Circuit-modifying functions.

The structure of the circuit is developed using writing heads which are pointing to the nodes of current circuit development positions. An example describing two writing heads is presented in Fig. 4.4. The first writing head Z1 points to the wire between nodes 2 and 3 and realizes connection-modifying function FLIP. As can be seen in Tab. 4.4 the function realizes reversion of the polarity of the circuit between two nodes defined by the writing head. The second writing head Z0 realizes component-creating function C which places capacitor between nodes 3 and 4.

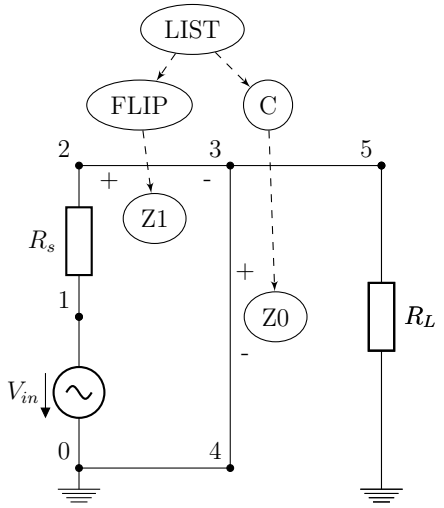


Fig. 4.4: Illustration of the functions FLIP and C.

BJT transistor placement function QT is presented in Fig. 4.5. Based on the context-sensitive subfunction selected from QT0 to QT11 transistor BJT is inserted to the nodes where the highlighted component is connected. After execution of the QT function there will be five new nodes and three new modifiable wires which are defined by writing heads (Z7, Z8, Z9).

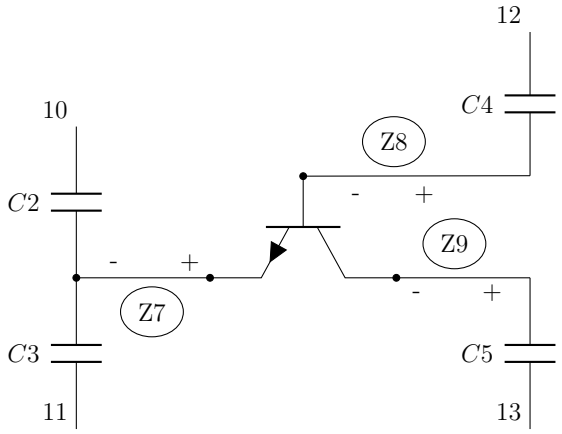


Fig. 4.5: Illustration of the function QT.

Another examples of connection-modifying functions presented in [31] are functions SERIES and PSS. Function SERIES places copy of the highlighted component in series to the original component (R1) (Fig. 4.6). After execution of the function SERIES there will be three writing heads which are pointing to the newly created component (R7), the new modifiable wire (Z6) and the original component (R1).

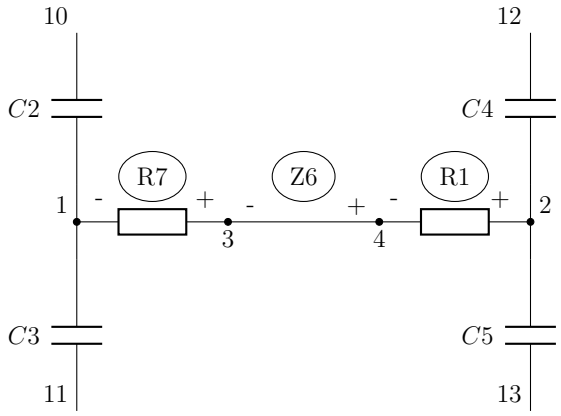


Fig. 4.6: Ilustration of the function SERIES.

The function PSS places newly created component (R7) in parallel to the original highlighted component (R1) (Fig. 4.7). After execution of PSS function there will be four writing heads which are pointing to the original component (R1), newly created component (R7) and two modifiable wires (Z8 and Z6).

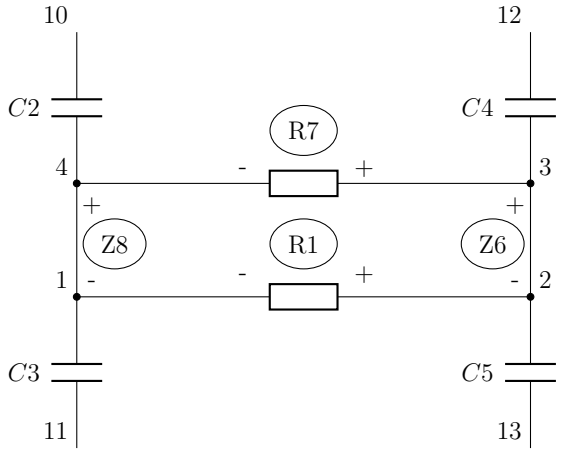


Fig. 4.7: Illustration of the function PSS.

For description of the other component-creating and connection-modifying functions or more detailed description of the parse tree approach please refer to [31] [32] [33] [34].

4.3 Linear Representation

Linear representation is another example of developmental encoding method. The approach was introduced by J. Lohn and S. Colombano in [15]. Sequences of instructions (called OPCODEs) are encoded into encoding vectors consisting of integer numbers. Compared to the direct encoding method where the order of the components encoding 5-tuples is not important in linear representation it strongly affects the whole structure of the resulting circuit.

Every component is encoded using four integer numbers. The first one is OPCODE and the next three numbers define the value of the component. The list of OPCODEs as was presented in [15] is summarized in Tab. 4.5.

Opcode	Destination Node	CN Register
x-move-to-new	newly-created node	newly-created node
x-cast-to-previous	previous node	unchanged
x-cast-to-ground	ground node	unchanged
x-cast-to-input	input node	unchanged
x-cast-to-output	output node	unchanged

Tab. 4.5: Summary of OPCODEs used in developmental encoding method.

Similarly to the parse tree representation method, the process of circuit synthesis begins at the basic embryonic circuit which is presented in Fig. 4.8.

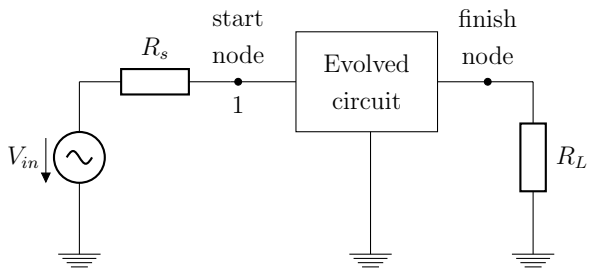


Fig. 4.8: Structure of the basic embryonic circuit.

The circuit synthesis starts from node 1 and continues according to the decoded sequence of OPCODEs until the output node (finish node) is reached or another termination criterion as maximal number of the decoded components is met. An example of sequence of OPCODEs and corresponding circuit are presented in Fig. 4.9 and Fig. 4.10 respectively.

```
C_MOVE_TO_NEW C 1 2 10837006
C_MOVE_TO_NEW C 2 3 3547581
C_CAST_TO_PREV C 3 2 9862769
L_CAST_TO_PREV L 3 2 6880711
L_CAST_TO_INPUT L 1 3 7668359
R_CAST_TO_PREV R 3 2 10844497
L_CAST_TO_INPUT L 1 3 6794434
L_CAST_TO_INPUT L 1 3 10915194
L_CAST_TO_INPUT L 1 3 10360067
L_CAST_TO_PREV L 3 2 53267
L_MOVE_TO_NEW L 3 4 7713100
C_MOVE_TO_NEW C 4 5 3568044
C_MOVE_TO_NEW C 5 6 8573596
C_CAST_TO_GND C 0 6 10481
L_CAST_TO_INPUT L 1 6 10897482
L_CAST_TO_INPUT L 1 6 2462113
R_MOVE_TO_NEW R 6 7 12757499
L_CAST_TO_PREV L 7 6 10259
C_CAST_TO_PREV C 7 6 10161281
L_CAST_TO_INPUT L 1 7 2452274
C_CAST_TO_PREV C 7 6 2457158
L_CAST_TO_INPUT L 1 7 15284140
L_CAST_TO_PREV L 7 6 1272684
R_MOVE_TO_NEW L 7 8 11316166
C_CAST_TO_GND C 0 8 24495
L_CAST_TO_PREV L 8 7 15793607
L_CAST_TO_INPUT L 1 8 3300694
L_MOVE_TO_NEW L 8 9 5376858
C_CAST_TO_GND C 0 9 25063
L_CAST_TO_PREV L 9 8 4863538
L_MOVE_TO_NEW L 9 10 4364222
C_CAST_TO_GND C 0 10 12564
L_CAST_TO_PREV L 10 9 4489818
```

Fig. 4.9: Example of sequence of OPCODEs.

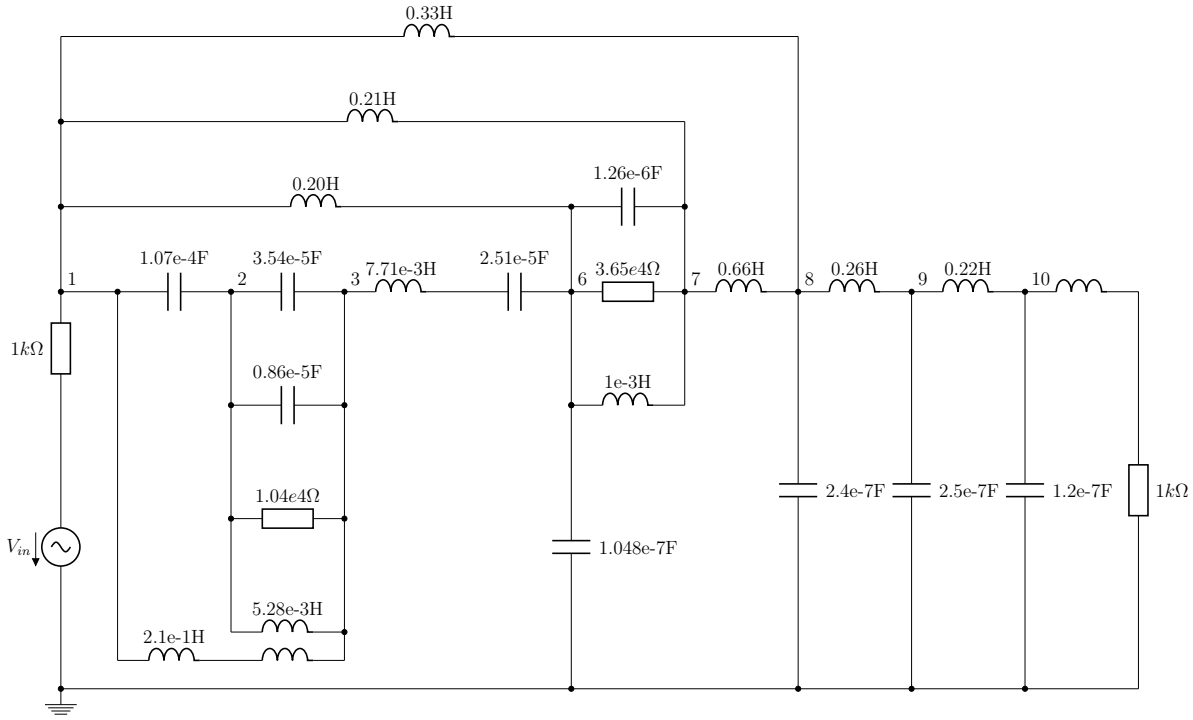


Fig. 4.10: Corresponding encoded circuit.

During the synthesis process register CN is used to store the number of the last created node. The principle of the circuit structure synthesis will be demonstrated for the first six instructions presented in Fig. 4.9.

1. `C_MOVE_TO_NEW C 1 2 10837006`

   Capacitor is placed between starting node 1 and newly created node 2. Number 2 is stored in CN register.

2. `C_MOVE_TO_NEW C 2 3 3547581`

   Capacitor is placed between node 2 and newly created node 3. Number 3 is stored in CN register.

3. `C_CAST_TO_PREV C 3 2 9862769`

   Capacitor is placed between node in CN (node 3) and previously created node (node 2). CN register unchanged.

4. `L_CAST_TO_PREV L 3 2 6880711`

   Inductor is placed between node in CN (node 3) and previously created node (node 2). CN register unchanged.

5. `L_CAST_TO_INPUT L 1 3 7668359`

   Inductor is placed between node in CN (node 3) and input (node 1). CN register unchanged.

6. `R_CAST_TO_PREV R 3 2 10844497`

   Resistor is placed between node in CN (node 3) and previously created node (node 2). CN register unchanged.

The motivation of development of linear representation encoding method is demand to overcome problems with high number of unsimulatable circuits produced by crossover of genetic algorithm in which direct encoding method is used. The proposed method is designed to construct closed analog circuit topologies what prevents generating a circuit topologies with dangling terminals or unconnected components. On the other hand using linear representation method can in some cases restrict circuit topologies which can be encoded [15].

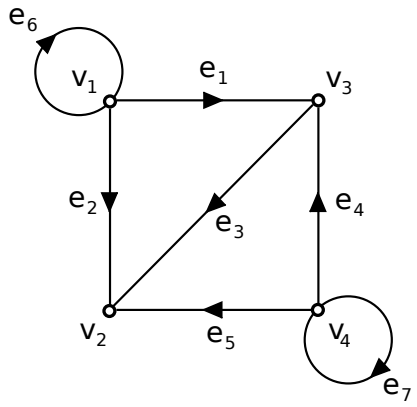
4.4 Adjacency Matrix Representation

Adjacency Matrix representation was proposed by A. Mesquita in [24]. The approach was developed to overcome some problems of other analog circuits encoding methods and to simplify the evolution process.

Let $G_a = (V_a, E_a)$ be a directed graph with no parallel edges, V_a is set of vertices defined as $V_a = \{v_1, v_2, \ldots, v_n\}$ and E_a is set of edges defined as $E_a = \{e_1, e_2, \ldots, e_n\}$. Then the adjacency matrix $M_a = [m_{ij}]$ of graph G_a is defined as n x n matrix, where m_{ij} is defined as:

$$m_{ij} = \begin{cases} 1, & \text{if } (v_i, v_j) \in E_a \\ 0, & \text{otherwise} \end{cases}$$

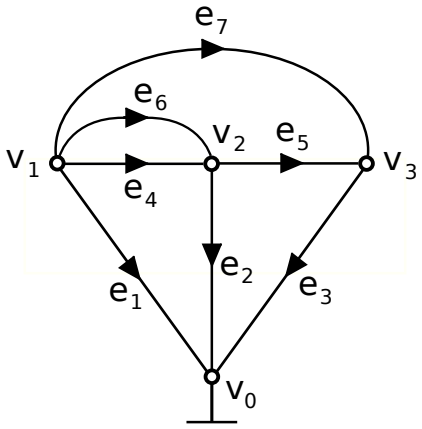
In Fig. 4.11 there is an example of generic floating graph with self-loops and its corresponding adjacency matrix [24].



$$M_a = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \end{array} \quad (4.1)$$

Fig. 4.11: Example of generic floating graph with self-loops G_a and its corresponding adjacency matrix M_a.

In the domain of analog circuits topologies the self-loops can be discarded. Also the ground nodes are redundant in the adjacency matrix. Then modified directed graph $G'_a = (V'_a, E'_a)$ and reduced adjacency matrix M'_a are defined. Example of graph representation of an analog circuit G'_a and corresponding reduced adjacency matrix M'_a are presented in Fig 4.12. Vertex v_0 of the modified graph G'_a represents the ground node of the circuit.



$$M'_a = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \end{array} \begin{array}{c} \begin{array}{ccc} v_1 & v_2 & v_3 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \end{array} \tag{4.2}$$

Fig. 4.12: Example of graph representation of analog circuit topology G'_a and its corresponding reduced adjacency matrix M'_a.

In the given example adjacencies to the ground node v_0 are represented by diagonal branches e_1, e_2, e_3. Parallel branches e_4 and e_6 represent two parallel components in the corresponding analog circuit.

For the purpose of evolutionary electronics system which employs genetic algorithm the adjacency matrix is transformed to sequence its rows. This way binary chromosome representation of the reduced adjacency matrix is formed.

In [24] special type of crossover (cascading of chromosomes) was proposed. It was demonstrated that using incidence matrix representation together with classical GA crossover results in construction of invalid graph topologies. On the contrary to the classical GA crossover the proposed method cascades the parent circuits through a subset of nodes in a meaningful way. This way topologies of both of the parental circuits are preserved.

As was discussed in [24] the proposed encoding method overperforms simple direct encoding method (incidence matrix representation) in terms of generation of low number of unsimulatable individuals after performing of GA crossover. Paper [24] also discusses another issues of the proposed methods such as employing of variable size chromosomes or generating of anomalous circuits.

5 AUTONOMOUS CIRCUITS, GRAPHS AND ENCODING METHODS

Autonomous circuits theory employs fully connected admittance networks which can be used for obtaining of new structures of analog circuits such as filters or oscillators [36]. There is direct relation between autonomous circuits and corresponding fully connected graphs. Autonomous circuits are described in section 5.1.

The main principle of the proposed automated analog circuits synthesis method is to utilize autonomous circuits idea in connection with Estimation of Distribution Algorithm. This way hand design using autonomous circuits theory as was presented in [36] will be replaced by probabilistic modeling of the promising circuit structures. Using powerful computation and probabilistic modeling approach allow to explore extensive solution space of large number of different combinations of suitable topologies of analog circuits.

Choosing of suitable encoding method is one of the key aspects of every evolutionary algorithm system. The most straightforward method of representation of the topology of analog circuit is graph. Detailed study of graph representations of the topologies of analog circuits can be found in [35]. Graph representations of the topologies of analog circuits with regard to using Estimation of Distribution Algorithms are presented in section 5.3.

Section 5.4 describes analog circuit design problem as problem of searching of subgraph. Design of passive, active and mixed passive/active circuits are discussed in the section.

Section 5.5 is focused on encoding methods of the topology of passive, active and mixed passive/active analog circuits. The proposed methods are designed with regard to usage of Estimation of Distribution Algorithms.

With respect to the encoding methods proposed in section 5.5 the problem of analog circuit synthesis has to be viewed as a problem with unitation constraints. This issue is discussed in section 5.6.

5.1 Autonomous Circuits

Autonomous circuits theory was described in [36]. The paper is also focused on employing of the theory for the purpose of obtaining new topologies of passive and active analog circuits.

Autonomous circuit (AC) can be formed of network of only passive components or combination of passive and active components (OTA, operational amplifier, cur-

rent conveyor, current mirror). In the thesis autonomous circuit is defined as presented bellow.

Given number of nodes n_n, autonomous circuit is defined as fully connected network of n_p ports generalized admittances, where n_p ports generalized admittances are connected between all possible combinations of the nodes. A n_p ports generalized admittance is component of n_p ports with no defined type. It can be replaced by any component of n_p ports or theirs parallel combination. Autonomous circuit with n_n nodes and with n_p ports generalized admittances is denoted as $n_p\,\mathrm{AC}\,n_n$.

Number of n_p ports generalized admittances of autonomous circuit with complexity n_n nodes can calculated according to 5.1.

$$n_{adm} = \binom{n_n}{n_p} = \frac{n_n!}{n_p!(n_n - n_p)!} \tag{5.1}$$

Autonomous circuit for $n_n = 5$ (nodes 0 to 4) consisting of 2 ports generalized admittances ($n_p = 2$) is presented in Fig. 5.1. According to the denotation presented above the autonomous circuit is denoted as 2AC5. The autonomous circuit contains ten 2 ports generalized admittances (see (5.1)).

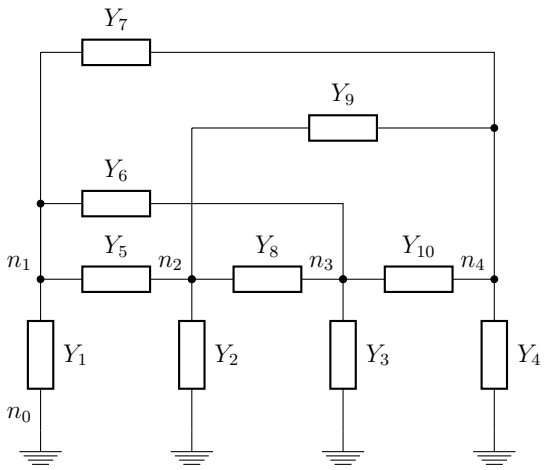


Fig. 5.1: Autonomous circuit 2AC5.

Assignment of the 2 ports generalized admittances to the nodes of AC is presented in Tab. 5.1. Later this assignment will be used in definition of binary encoding vector of the circuit topology.

generalized admittance	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	Y_{10}
node 1	n_0	n_0	n_0	n_0	n_1	n_1	n_1	n_2	n_2	n_3
node 2	n_1	n_2	n_3	n_4	n_2	n_3	n_4	n_3	n_4	n_4

Tab. 5.1: Assignment of the generalized admittances to nodes of 2AC5.

Autonomous circuit for $n_n = 4$ consisting of 3 ports generalized admittances ($n_p = 3$) is presented in Fig. 5.2. The autonomous circuit is denoted as 3AC4 and contains four 3 ports generalized admittances.

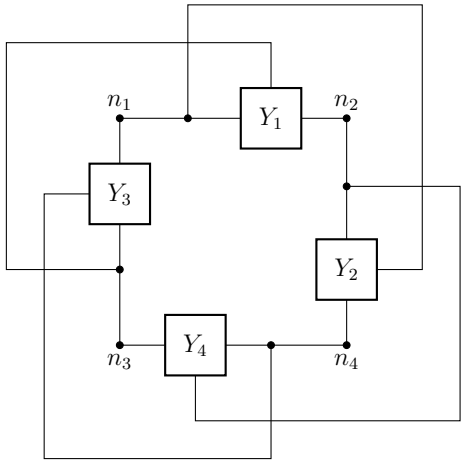


Fig. 5.2: Autonomous circuit 3AC4.

Assignment of the 3 ports generalized admittances to the nodes of AC is presented in Tab. 5.2.

generalized admittance	Y_1	Y_2	Y_3	Y_4
node 1	n_1	n_1	n_1	n_2
node 2	n_2	n_2	n_3	n_3
node 3	n_3	n_4	n_4	n_4

Tab. 5.2: Assignment of the generalized admittances to nodes of 3AC4.

Autonomous circuit with 3 ports generalized admittances enables using of 3 ports active components. Every single 3 ports generalized admittance of autonomous circuit in Fig. 5.2 can be replaced by defined type of 3 ports transistor (NPN, PNP, NMOS, PMOS, etc.).

5.2 Autonomous Circuits and Circuit Design

The autonomous circuits design method can be used for synthesis of different types of analog circuit applications as frequency filters or oscillators. In [37] autonomous circuits method was used for obtaining new biquad filters using two CFOA blocks. Study of general three ports current conveyors (GCC) utilizing autonomous circuits theory was presented in [38]. Filters and oscillators design based on autonomous circuits representation of generalized multi-port current conveyors was presented in [39].

In the text bellow design of passive analog circuits based on application of autonomous circuits as was presented in [36] will be discussed. Since the desired analog circuit is of passive type 2 ports generalized admittances ($n_p = 2$) are used.

Complexity of the used autonomous circuit (n_n) is user defined parameter and its selection is based on the target application of the designed circuit. For example if the target application of the designed circuit is frequency filter, the complexity of the autonomous circuit can be determined based on the order of the filter. Also previous experience of the designer is helpful during this phase. For the demonstration purpose autonomous circuit with complexity $n_n = 5$ is chosen. Its schematics is presented in Fig. 5.3.

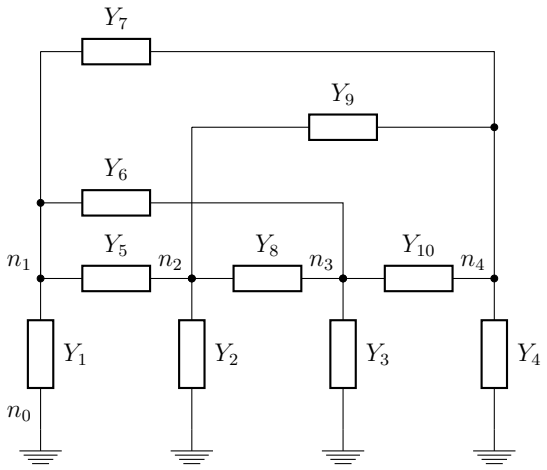


Fig. 5.3: Autonomous circuit with 2 ports generalized admittances and complexity $n_n = 5$.

Design of passive analog circuit using autonomous circuits consists of few steps as will be described bellow.

STEP1: 2 ports generalized admittances are manually replaced by real passive components, theirs parallel combination, or open circuit. The replacement is performed randomly or based on some initial requirements (all capacitors grounded). Also

previous experience of the designer or intuition is incorporated in the process. An example of such replacement is presented in Fig 5.4. The replacement of the components is summarized in Tab 5.3.

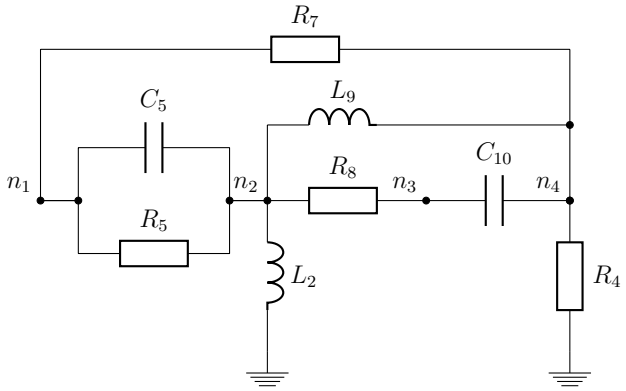


Fig. 5.4: Replacement of the generalized admittances by real components.

General admittance	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	Y_{10}
node 1	n_0	n_0	n_0	n_0	n_1	n_1	n_1	n_2	n_2	n_3
node 2	n_1	n_2	n_3	n_4	n_2	n_3	n_4	n_3	n_4	n_4
replacement	open	L_2	open	R_4	$R_5 \parallel C_5$	open	R_7	R_8	L_9	C_{10}

Tab. 5.3: Replacement of the generalized admittances by real components.

STEP2: After obtaining of a new structure of analog circuit (Fig. 5.4) characteristic equation in symbolic form is obtained. For different input and output connection nodes various impedance and transfer functions can be obtained. Selection of input and output nodes is determined by desired function of the designed analog circuit.

STEP3: For selected configuration of input and output nodes and resulting impedance or transfer function values of the components are calculated.

5.3 Graph Representations of Analog Circuits

5.3.1 Simple Graph Representation

Let's define simple parametrized undirected graph $G_s = (V_s, E_s, Q_s)$, where $V_s = \{v_0, v_1, \ldots, v_4\}$ is set of vertices, $E_s = \{e_1, e_2, \ldots, e_5\}$ is set of parametrized edges and $Q_s = \{q_1, q_2, \ldots, q_5\}$ is set of parameters of edges E_s. An example of graph G_s is presented in Fig. 5.5a. Numbers in the brackets behind the names of the edges denote parameters of the edges. Analog circuit C_s corresponding to graph G_s is presented in Fig. 5.5b. Branches of circuit C_s correspond to parametrized edges of set E_s. Set of nodes $N_s = \{n_0, n_1, \ldots, n_4\}$ of circuit C_s corresponds to set of vertices V_s.

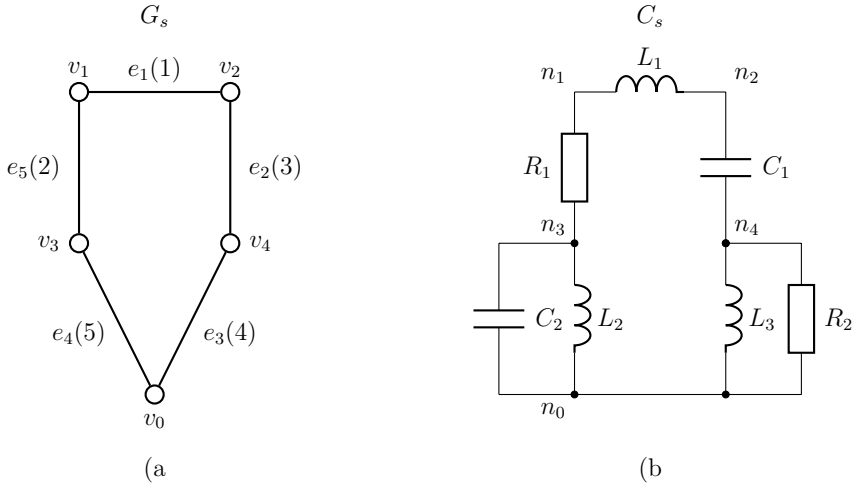


Fig. 5.5: a) Graph G_s b) Analog circuit C_s.

Since simple graph does not allow to use multiple edges (representation of parallel connection of the components), the configuration of the used components is represented as parameters of the edges of the simple graph. All possible parameters of the edges representing the combination of the passive components are summarized in Tab. 5.4.

edge parameter	1	2	3	4	5	6	7
configuration of components	L	R	C	$L\|\|R$	$L\|\|C$	$R \parallel C$	$L \parallel R \parallel C$

Tab. 5.4: Parameters of the edges and corresponding configurations of components.

Parameters in set Q_s define configuration of the components for the corresponding edges. For example edge e_3 is connected between vertices v_4 and v_0 and its parameter is $q_3 = 4$. Therefore according to Tab. 5.4 corresponding configuration of the components connected between nodes n_4 and n_0 is parallel combination of inductor and resistor ($L_3 \parallel R_2$).

5.3.2 Multigraph Representation

Let's define undirected labeled multigraph $G_m = (V_m, E_m, Q_m)$, where $V_m = \{v_0, v_1, \ldots, v_4\}$ is set of vertices, $E_m = \{e_1, e_2, \ldots, e_7\}$ is set of labeled edges and $Q_m = \{q_1, q_2, \ldots, q_7\}$ is set of labels of edges E_m. Self loops are not allowed. Three multiple edges between two vertices are allowed at the most. An example of graph G_m is presented in Fig. 5.6a. Numbers in the brackets behind the names of the edges denote the labels of the edges. Analog circuit C_m corresponding to graph G_m is presented in Fig. 5.6b. Branches of circuit C_m correspond to labeled edges E_m. Set of nodes $N_m = \{n_0, n_1, \ldots, n_4\}$ of circuit C_m corresponds to set of vertices V_m. Labels Q_m of edges E_m define selection of the passive components for the corresponding edges. For example edge e_6 is connected between vertices v_3 and v_0 and its label is $q_6 = 1$. Therefore according to Tab. 5.5 corresponding passive component connected between nodes n_3 and n_0 is inductor(L_2).

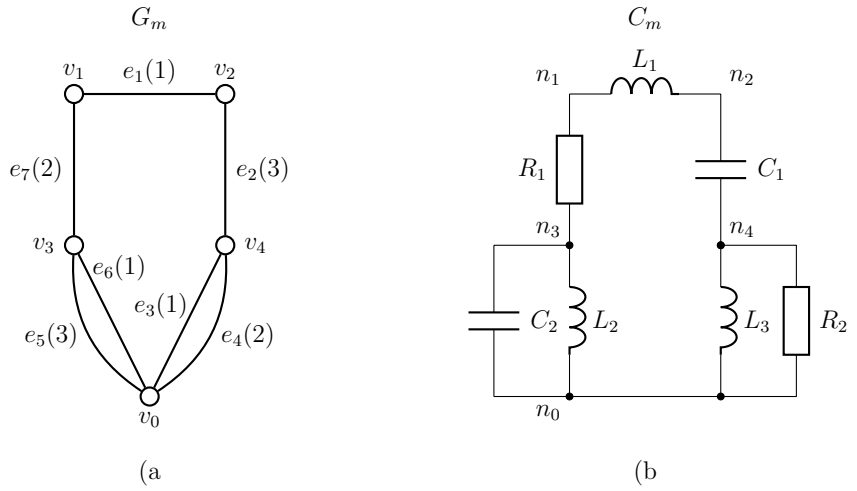


Fig. 5.6: a) Multigraph G_m b) analog circuit C_m.

Compared to simple graph representation every component of analog circuit C_m is represented by single edge of multigraph G_m. All possible values of the labels of the edges representing selection of the passive components are summarized in Tab. 5.5.

edge label	1	2	3
selected component	L	R	C

Tab. 5.5: Labels of the edges and corresponding selection of the component.

5.3.3 3-Uniform Labeled Hypergraph Representation

Compared to simple graph and multigraph representations described above, labeled hypergraphs allow to encode components with more than 2 ports. An example of 3-uniform labeled hypergraph which allows to encode 3 ports components (transistors) is presented in the section.

Let's define 3-uniform labeled hypergraph $G_h = (V_h, E_h, Q_h)$, where $V_h = \{v_1, v_2, \ldots, v_6\}$ is set of vertices, $E_h = \{e_1, e_2\}$ is set of labeled hyperedges and $Q_h = \{q_1, q_2\}$ is set of labels of hyperedges of set E_h. An example of hypergraph G_h is presented in Fig. 5.7a. Numbers in the brackets behind the names of the hyperedges denote the labels of the hyperedges. Analog circuit C_h corresponding to graph G_h is presented in Fig. 5.7b. Labeled hyperedges of set E_h correspond to the transistors of analog circuit C_h where the labels of the hyperedges define "rotations" of the transistors (Tab. 5.6). Set of vertices V_h correspond to set of nodes $N_h = \{n_1, n_2, \ldots, n_6\}$ of analog circuit C_h. For example hyperedge e_1 is connected to vertices v_1, v_2 and v_3. Label of hyperedge e_1 is set to 3 what corresponds to transistor (Q1) connected as n_1 - C, n_2 - B and n_3 - E (Tab. 5.6). Similarly hyperedge e_2 represents transistor $Q2$.

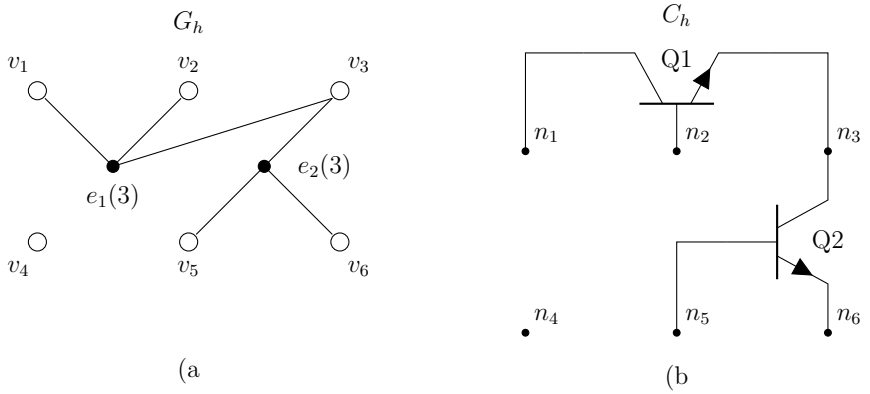


Fig. 5.7: a) 3-uniform labeled hypergraph G_h b) Analog circuit C_h.

Summary of the labels of the hyperedges and corresponding "rotations" of the transistors is presented in Tab. 5.6.

hyperedge label	1	2	3	4	5	6
node 1	B	B	C	C	E	E
node 2	C	E	B	E	B	C
node 3	E	C	E	B	C	B

Tab. 5.6: Labels of the hyperedges and corresponding "rotations" of the transistors.

5.4 Evolutionary Analog Circuits Design as a Problem of Searching of a Subgraph

5.4.1 Passive Circuits

Let's define complete graph $G_{cp} = (V_{cp}, E_{cp})$, where $V_{cp} = \{v_0, v_1, \ldots, v_3\}$ is set of vertices and $E_{cp} = \{e_1, e_2, \ldots, e_6\}$ is set of edges. Analog circuit corresponding to complete graph G_{cp} is AC with 2 ports generalized admittances and $n_n = 4$ denoted as 2AC4. Edges of set E_{cp} correspond to generalized admittances of autonomous circuit 2AC4. Vertices of set V_{cp} correspond to set of nodes $N_p = (n_1, n_2, \ldots, n_4)$ of circuit 2AC4. Complete graph G_{cp} and autonomous circuit 2AC4 are presented in Fig. 5.8.

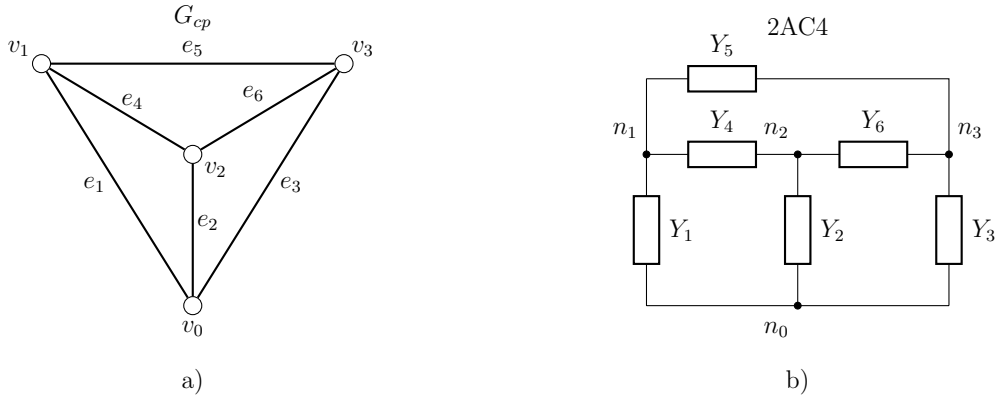


Fig. 5.8: a) Complete graph G_{cp} b) circuit 2AC4 corresponding to G_{cp}.

Now let's define passive analog circuit topology synthesis problem as problem of searching of a subgraph.

As was described in section 5.1 every 2 ports generalized admittance Y_x can be replaced by resistor, capacitor, inductor or theirs parallel combination. Therefore maximal combination of the components which can be used as a replacement of 2 ports generalized admittance Y_x is parallel combination of resistor, capacitor and inductor.

Every edge e_x of complete graph G_{cp} is replaced by triplet of edges $e_x(1), e_x(2), e_x(3)$ what corresponds to parallel connection of passive components inductor, resistor and capacitor (section 5.3.2). This way graph G_{cp} is expanded to graph G_{cpe} (Fig. 5.9).

Now analog circuit C_e corresponding to expanded graph G_{cpe} will be obtained. Every generalized admittance of autonomous circuit 2AC4 will be replaced by parallel combination of resistor, capacitor and inductor. This way autonomous circuit 2AC4 is expanded to circuit C_e. Analog circuit C_e corresponding to expanded graph G_{cpe} is presented in Fig. 5.10. Note that circuit C_e is maximal circuit which can be

obtained for 4 nodes and passive components RLC. Every possible passive analog circuit with complexity 4 nodes or less consisting of RLC components can be derived from circuit C_e by replacing its components with open circuits or short circuits.

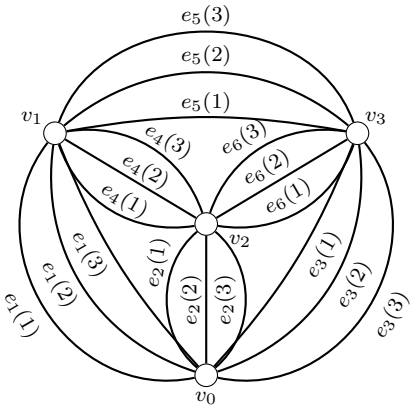


Fig. 5.9: Graph G_{cpe} (obtained by expansion of graph G_{cp}).

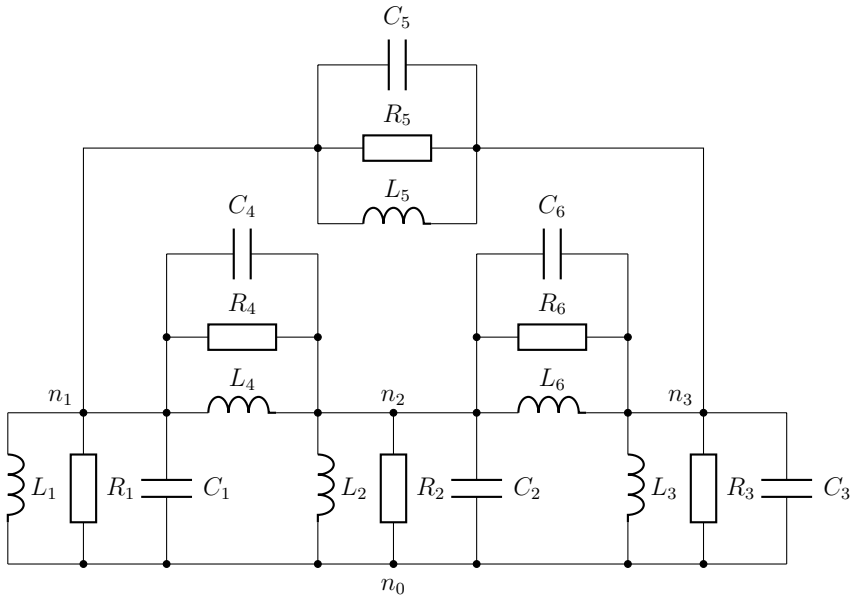


Fig. 5.10: Analog circuit C_e corresponding to expanded graph G_{cpe}.

Now the problem of the synthesis of a passive analog circuit topology can be defined as searching for a subgraph G_p on graph G_{cpe} what corresponds to selection of a subcircuit C_p of circuit C_e. Example of subgraph G_p and corresponding analog circuit C_p are presented in Fig. 5.11.

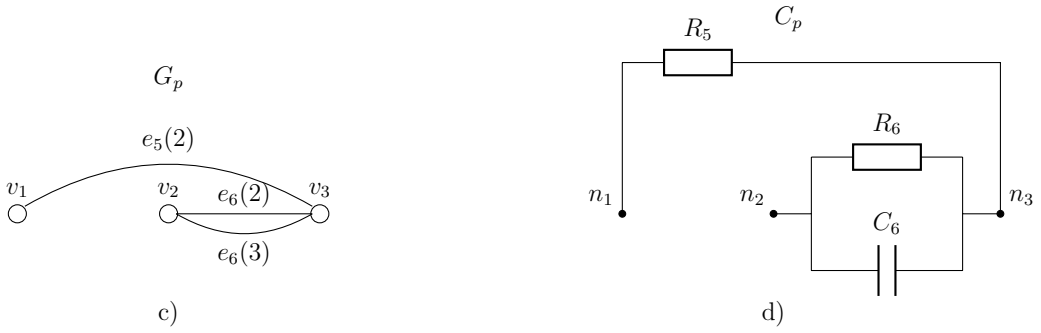


Fig. 5.11: a) Example of subgraph G_p b) Analog circuit C_p corresponding to subgraph G_p.

5.4.2 Transistor Circuits

Let's define complete 3-uniform hypergraph $G_{ct} = (V_{ct}, E_{ct})$, where $V_{ct} = \{v_1,\ v_2,\ v_3,\ v_4\}$ is set of vertices and $E_{ct} = \{e_1,\ e_2,\ e_3,\ e_4\}$ is set of hyperedges. Analog circuit corresponding to complete hypergraph G_{ct} is AC with 3 ports generalized admittances and $n_n = 4$ denoted as 3AC4. Every single hyperedge of hypergraph G_{ct} corresponds to a single generalized admittance of circuit 3AC4. Hypergraph G_{ct} and circuit 3AC4 are presented in Fig. 5.12.

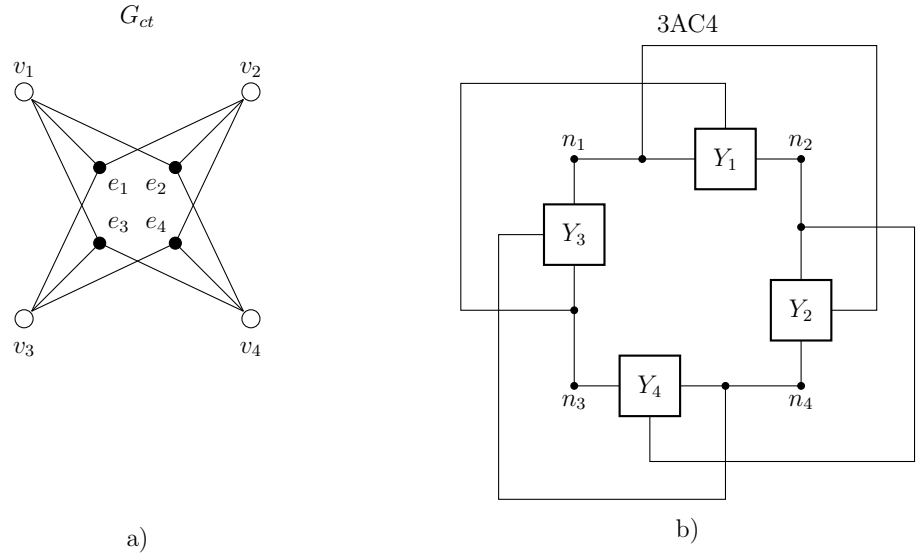


Fig. 5.12: a) Complete 3-uniform hypergraph G_{ct} b) circuit 3AC4 corresponding to G_{ct}.

There are six possible combinations of connection of 3 ports generalized admittance Y_x (admittances Y_1 to Y_4 in Fig. 5.12b) to three nodes. These connections are denoted as "rotations" (Tab. 5.6).

The problem of synthesis of the topology of connection of analog circuit consisting of transistors can be defined as searching for a subhypergraph G_t on complete hypergraph G_{ct} and definition of the labels of the hyperedges of subhypergraph G_t which define "rotations" of the transistors.

Example of labeled subhypergraph G_t is presented in Fig. 5.13a. Numbers in the brackets behind the names of the hyperedges define the labels of the hyperedges (Tab. 5.6). Circuit C_t corresponding to G_t is presented in Fig. 5.13b.

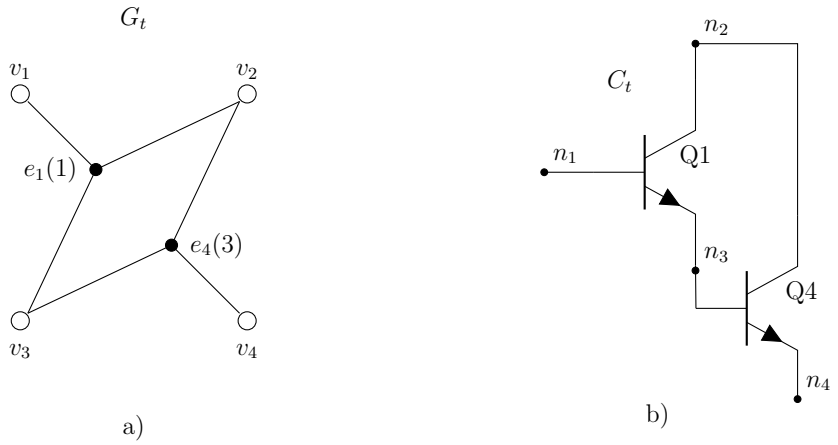


Fig. 5.13: a) Example of subgraph G_t on complete 3-uniform hypergraph G_{ct} b) analog circuit C_t corresponding to G_t.

5.4.3 Mixed Type Circuits

Representation of circuits of mixed type C_{mt} (circuit including transistors and passive components) is separated into two subcircuits C_{pas} and C_{act}. Subcircuit C_{pas} is formed of passive part of circuit C_{mt} (passive components) and subcircuit C_{act} is formed of active part of circuit C_{mt} (transistors). The passive part is defined by graph G_{pas} and it is represented as was described in section 5.4.1. The active part of the mixed circuit is defined by hypergraph G_{act} and it is represented as was described in section 5.4.2. An example of graph G_{pas} and corresponding analog circuit C_{pas} are presented in Fig. 5.14. An example of hypergraph G_{act} and corresponding analog circuit C_{act} are presented in Fig. 5.15.

After combination of both parts represented by graph G_{pas} and hypergraph G_{act} mixed type circuit in Fig. 5.16 is obtained.

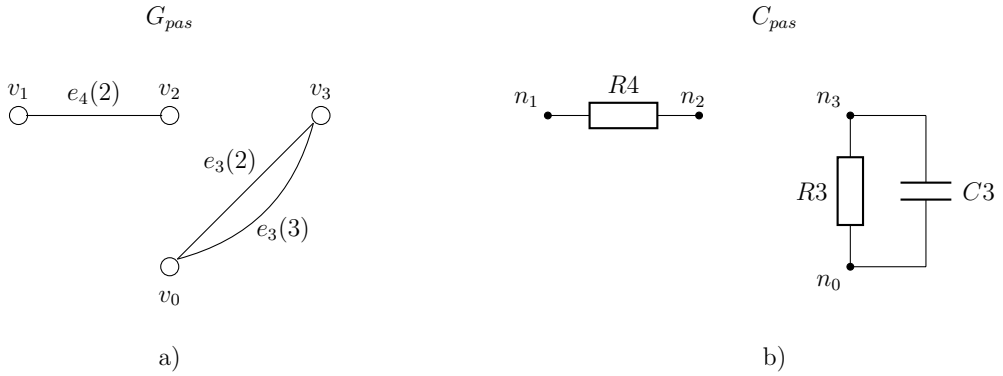


Fig. 5.14: a) Example of passive part graph G_{pas} b) analog circuit C_{pas} corresponding to graph G_{pas}.

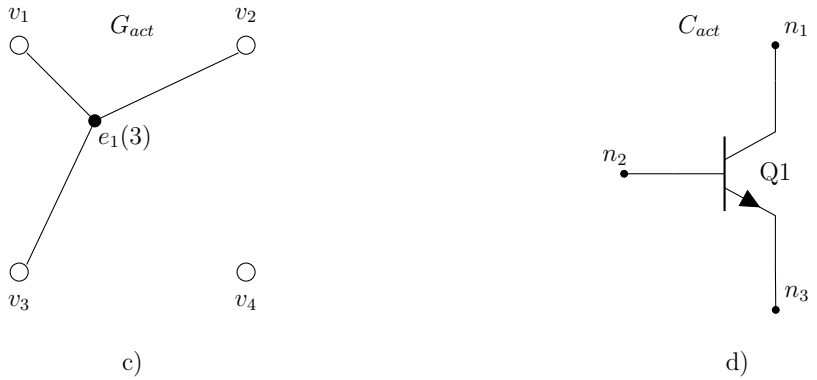


Fig. 5.15: a) Example of active part graph G_{act} b) analog circuit C_{act} corresponding to graph G_{act}.

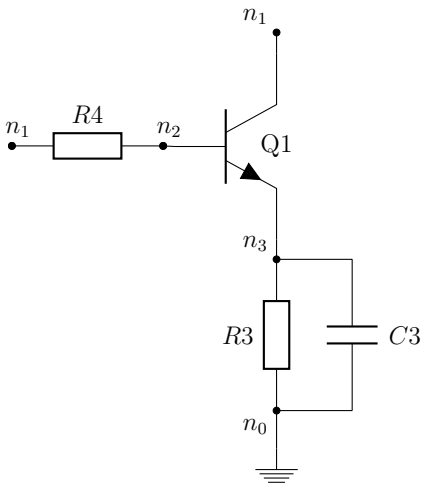


Fig. 5.16: Analog circuit of mixed type C_{mt} consisting of passive part circuit C_{pas} and active part circuit C_{act}.

5.5 Encoding of Graph Representations of Analog Circuits

5.5.1 Characteristic Vector of Simple Graph

Every simple graph G_s with n_n vertices can be encoded by binary vector e_s of length n_{sc}, where n_{sc} is number of edges of complete graph G_{sc} of order n_n vertices. Size n_{sc} of G_{sc} can be computed as $n_s = n_n(n_n - 1)/2$. Vector e_s is called characteristic vector of graph G_s. Every single bit of e_s represents including or not including of the corresponding edge in graph G_s. If characteristic vector $e_s(i) = 1$ for $i \in \{1, 2, \ldots n_s\}$ then vector e_s encodes complete graph G_{sc}. Examples of simple graphs and theirs characteristic vectors for $n_n = 3$ (G_{s1}) and $n_n = 4$ (G_{s2}) are presented in Fig. 5.17a and Fig. 5.17b respectively. Assignment of the edges to the combinations of the vertices is presented in Tab. 5.7.

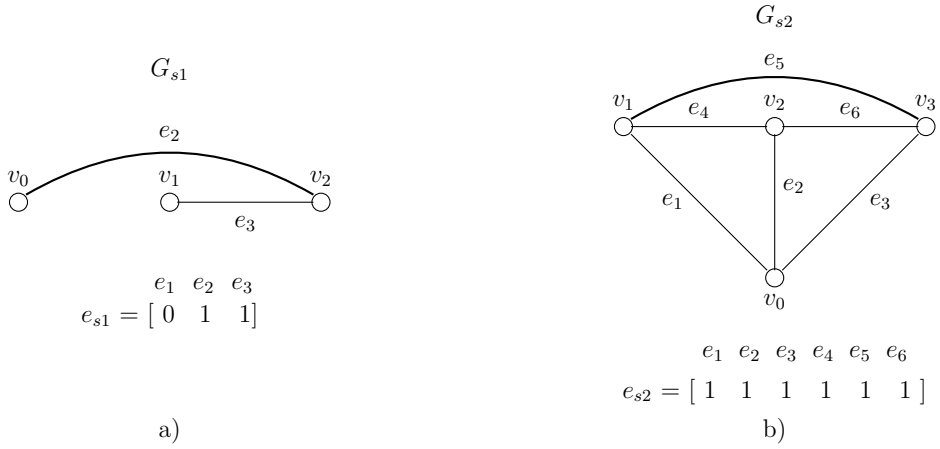


Fig. 5.17: a) Example of simple graph for $n_n = 3$ and its characteristic vector e_{s1} b) example of simple graph for $n_n = 4$ and its characteristic vector e_{s2}.

graph G_{s1}			
vertice 1	v_0	v_0	v_1
vertice 2	v_1	v_2	v_2
edge	e_1	e_2	e_3

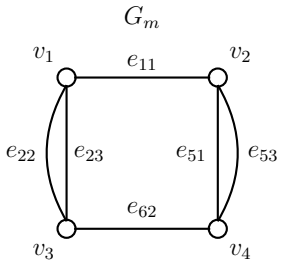
graph G_{s2}						
vertice 1	v_0	v_0	v_0	v_1	v_1	v_2
vertice 2	v_1	v_2	v_3	v_2	v_3	v_3
edge	e_1	e_2	e_3	e_4	e_5	e_6

Tab. 5.7: Assignment of the edges to the combinations of the vertices for simple graphs G_{s1} and G_{s2}.

5.5.2 Characteristic Vector of Multigraph

Since multigraphs allow using multiple edges between two vertices the definition of the characteristic vector which was presented in the previous section has to be modified. Let's define multigraph G_m with n_n vertices in which n_m multiple edges are allowed at the most. Selfloops are not allowed. Then multigraph G_m can be encoded using binary characteristic vector e_m of length $n_m\, n_{ec}$, where n_{ec} is number of edges of complete graph G_c of order n_n.

An example of multigraph G_m with 4 vertices ($n_n = 4$) and three multiple edges at the most ($n_m = 3$) is presented in Fig. 5.18a. Its characteristic vector e_m is presented in Fig. 5.18b. As can be seen in Fig. 5.18b characteristic vector e_m consists of six triplets of bits, where every single triplet corresponds to one combination of the vertices. Assignment of the edges to the combinations of the vertices is presented in Tab. 5.8.



a)

	e_{1x}			e_{2x}			e_{3x}			e_{4x}			e_{5x}			e_{6x}		
	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
$e_m =$ [	1	0	0	0	1	1	0	0	0	0	0	0	1	0	1	0	1	0]

b)

Fig. 5.18: a) Mutigraph G_m with 4 vertices b) characteristic vector e_m of multigraph G_m.

vertice 1	v_1	v_1	v_1	v_2	v_2	v_3
vertice 2	v_2	v_3	v_4	v_3	v_4	v_4
edge	e_{1x}	e_{2x}	e_{3x}	e_{4x}	e_{5x}	e_{6x}

Tab. 5.8: Assignment of the edges to the combinations of the vertices.

5.5.3 Encoding of Passive Analog Circuits

Topology of passive analog circuit is represented by graph G_p which is subgraph of complete expanded graph G_{cpe} (section 5.4.1). Examples of G_{cpe} and G_p are given in Fig. 5.9 and Fig. 5.11a respectively.

As was stated in section 5.4.1 graph G_p is always subgraph of complete expanded graph G_{cpe}. Circuit corresponding to graph G_{cpe} is the maximal circuit which can

be obtained (for given n_n nodes). Therefore G_{cpe} is the maximal graph which the encoding method has to be able to encode.

Encoding vector of graph G_p can be defined as binary vector e_p of length n_{edgcpe}, where n_{edgcpe} is number of edges of expanded complete graph G_{cpe} and can be calculated according to (5.2)

$$n_{edgcpe} = \frac{n_m n_n (n_n - 1)}{2} \tag{5.2}$$

, where n_m is maximal number of multiple edges of multigraph G_p what corresponds to maximal number of components of different type which can be connected in parallel. Typically for passive analog circuits there are three types of components (resistor, capacitor, inductor). Therefore $n_m = 3$. (section 5.4.1).

Every single bit of vector e_p denotes including or not including of the corresponding edge of expanded complete graph G_{cpe} in its subgraph G_p. Encoding vector e_p of graph G_p presented in Fig. 5.11a is presented in Fig. 5.19.

$$\begin{array}{c} \overline{e_1} \quad \overline{e_2} \quad \overline{e_3} \quad \overline{e_4} \quad \overline{e_5} \quad \overline{e_6} \\ \text{L R C L R C L R C L R C L R C L R C} \\ e_p = [\,0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\,] \end{array}$$

Fig. 5.19: Encoding vector e_p of graph G_p in Fig. 5.11a

5.5.4 Characteristic Vector of 3-Uniform Hypergraph

Every 3-uniform hypergraph G_h of order n_n can be encoded by binary vector e_h of length n_e, where n_e is size of complete 3-uniform hypergraph G_{hc} of order n_n. Size n_e of complete 3-uniform hypergraph G_{hc} can be calculated as $n_e = n_n(n_n-1)(n_n-2)/3$.

Every single bit of e_h represents including or not including of the corresponding hyperedge of hypergraph G_h. If characteristic vector $e_h(i) = 1$ for $i \in \{1, 2, \ldots n_e\}$ then vector e_h encodes complete 3-uniform hypergraph G_{hc}.

Two examples of 3-uniform hypergraphs for $n_n = 4$ and theirs characteristic vectors are presented in Fig. 5.20. Assignment of the hyperedges to the combinations of the vertices is presented in Tab. 5.9.

hyperedge	e_1	e_2	e_3	e_4
vertices	v_1, v_2, v_3	v_1, v_2, v_4	v_1, v_3, v_4	v_2, v_3, v_4

Tab. 5.9: Assignment of the hyperedges to the vertices.

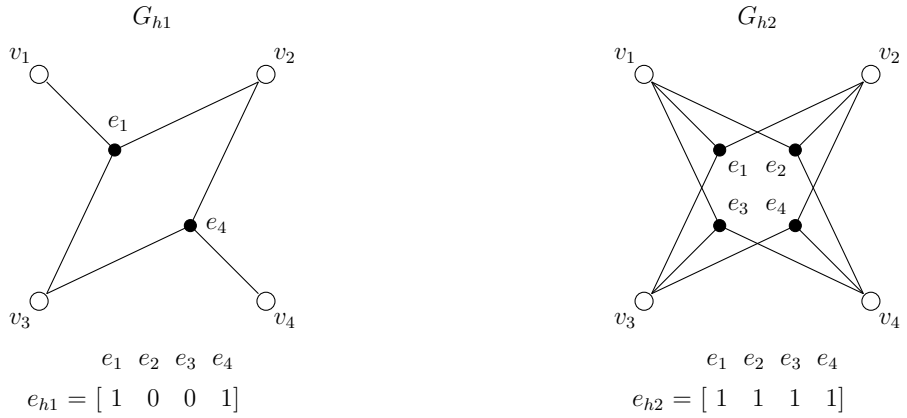


Fig. 5.20: Two examples of 3-uniform hypergraphs for $n_n = 4$ and theirs characteristic vectors.

5.5.5 Characteristic Vector of 3-Uniform Labeled Hypergraph

As was described in the previous section connection of three ports components can be defined using 3-uniform hypergraphs. Assuming three ports transistor there are six possible combinations of connection of the transistor to three nodes. These combinations are called "rotations" of the transistor in this thesis and are represented as labels of hyperedges. Label of every hyperedge can be set to numbers 1 to 6.

To include information about "rotation" characteristic vector of 3-uniform hypergraph which was defined in the previous section has to be modified. Labeled 3-uniform hypergraph G_{hl} of order n_n can be encoded by binary vector e_{hl} of length n_{hl}, where $n_{hl} = n_n(n_n - 1)(n_n - 2)$. Example of labeled 3-uniform hypergraph and its encoding vector e_{hl} are presented in Fig. 5.21a and Fig. 5.21b respectively.

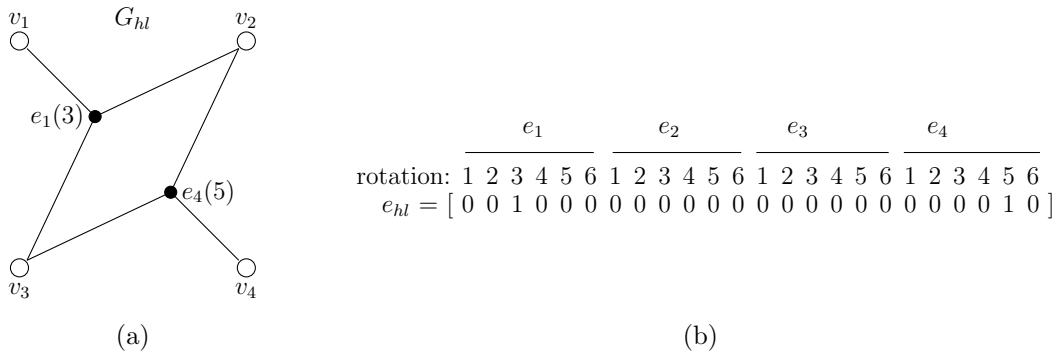


Fig. 5.21: Example of 3-uniform labeled hypergraph G_{hl} (a) and its encoding vector e_{hl} (b).

5.5.6 Encoding of Active Analog Circuits (Transistors)

Encoding method of active analog circuits containing transistors is described in section 11.3 on example of synthesis of analog circuit realization of cube root function.

5.6 Analog Circuit Synthesis and Unitation Constraints

Binary representation of the topology of passive analog circuits using graphs was described in section 5.5.3. Binary representation of the topology of transistor circuits using labeled hypergraphs was described in section 5.5.6. In both cases number of the components of the resulting encoded circuits is determined by number of "ones" of the corresponding encoding vectors (e_p for passive circuits and e_t for transistor circuits).

Generally desired specifications of analog circuits are easier to reach using analog circuit of higher complexity. Because of this fact evolutionary analog circuits synthesis methods tend to evolve analog circuits with complexity as large as possible. Without restriction of number of the components of the evolved circuit complexity of the evolved circuit becomes higher that necessary.

Therefore number of the components of the evolved circuit has to be restricted to user define value. Since in the proposed encoding methods number of the components of the encoded circuit is determined by number of "ones" of the binary encoding vectors the restriction of components leads to problem with unitation constraints [48].

Definition 1. Let's define vector $x = (x_1, x_2, \ldots, x_n) \in \Omega$. Then unitation value of x is defined as

$$u(x) := \sum_{i=1}^{n} x_i \tag{5.3}$$

Value of unitation function $u(x)$ depends only on the number of ones in an input vector x. Unitation values of two vectors with the same numbers of ones are equal.

Problem with unitation constraints is defined as solution e_s in which unitation value $u(e_s)$ (number of ones in solution e_s) is restricted to defined number [48].

6 THESIS OBJECTIVES

The goal of the thesis is to design and verify three types of automated analog circuits synthesis methods based on Estimation of Distribution Algorithms as listed bellow. Synthesis capability of every method will be verified on an example of analog circuit synthesis problem. The goals of the thesis are:

- Design and verify automated passive analog circuits synthesis method based on pure UMDA. The topology and the parameters (values of the components) are determined using UMDA algorithm. The method should be able to overcome two drawbacks of the method presented in [28] which are inability of encoding of parallel connection of the components and problems with circuits of larger complexity.

- Design and verify hybrid automated passive analog circuits synthesis method based on UMDA [27] and local search method. The topology of the desired circuit is synthesized using UMDA. The parameters of the desired circuit are determined using a local search method.

- Design hybrid automated active analog circuits synthesis method based on employing of univariate probabilistic model. As in the previous case, the topology of the desired analog circuit is synthesized using Estimation of Distribution Algorithm and the parameters of the desired circuit are determined using local search algorithm. Compared to the previous case the method should be able to synthesize active analog circuits which includes active components such as transistors.

7 USED APPROACHES

Approaches, methods and principles used in the proposed evolutionary analog electronics synthesis methods will be introduced in the section.

7.1 Estimation of Distribution Algorithms

Estimation of Distribution Algorithms (EDA) also referred to as Population Model-Building Genetic Algorithms (PMBGA) are evolutionary optimization techniques employing probabilistic models to generate new solutions. Pseudo-code of EDA is presented in Fig. 7.1.

step1: Set k = 1. Initialize population $P(k)$.
step2: Select population of promising individuals $P_s(k)$.
step3: Create probabilistic model $M(k)$ of population of selected individuals $P_s(k)$.
step4: Using probabilistic model $M(k)$ generate population of new solutions $P_g(k)$.
step5: Set $k = k + 1$. Create new population $P(k)$. If termination criteria not met go to **step2**.

Fig. 7.1: Pseudo-code of Estimation of Distribution Algorithm.

The main idea behind EDA is to replace recombination phase of genetic algorithms (GA) by probabilistic modeling of the promising areas of the solution space. The motivation for developing EDA was demand to overcome some drawback of genetic algorithms as requirement of tight linkage, problems with deceptive functions or disruption of the building blocks after performing crossover operation. As can be seen in Fig. 7.1 the pseudo-code of EDA is similar to pseudo-code of classic GA. The only difference is probabilistic modeling used in **step3** and **step4**.

In **step1** population $P(1)$ is initialized. Initialization is performed randomly or with defined seed.

In **step2** good individuals (based on fitness values) are selected and selected population $P_s(k)$ is formed.

In **step3** probabilistic model $M(k)$ of individuals of selected population $P_s(k)$ is created. There are many types of probabilistic models which can be employed. Some of them will be briefly introduced in the next sections.

In **step4** created probabilistic model $M(k)$ is used to generate population of new solutions $P_g(k)$. This phase of EDA algorithms is usually called sampling.

In **step5** new population P_{k+1} which is created of population of new solutions $P_g(k)$ or combination of $P_g(k)$ and $P(k)$ is created. Described process continues until one of the termination criteria such as desired fitness function, maximal number of objective function evaluations or maximal running time is met.

There have been proposed many different types of EDA algorithms which vary in the used probabilistic models or theirs principles. Some of them will be briefly introduced bellow.

7.1.1 Univariate Marginal Distribution Algorithm (UMDA)

Univariate Marginal Distribution Algorithm [27] is the simplest variant of EDA algorithm. The basic principle of the building of the probabilistic model is to count the marginal frequencies of occurrence of the components (the univariate marginal probability) in a population of the candidate solutions. The probabilistic model is constructed under assumption that the components of the solution are independent of each other. The computed marginal frequencies of the components are used for generation of new solutions during sampling phase of the algorithm.

7.1.2 Bivariate Marginal Distribution Algorithm (BMDA)

Bivariate Marginal Distribution Algorithm is a extension of UMDA algorithm [40]. While UMDA uses simple univariate marginal distributions, algorithm BMDA uses pair gene dependencies. Algorithm BMDA is special case of Factorial Distribution Algorithm however there is not used any problem specific knowledge in the initial stage. Dependencies between the variables of the solution vector are discovered during the run of the algorithm.

7.1.3 Population Based Incremental Learning (PBIL)

The main objective of the algorithm is to reduce memory required by the classical genetic algorithm [41]. Characteristics of the population of the candidate solutions are represented by real-valued prototype probabilistic vector $p_l(x)$. During the sampling phase probabilistic vector $p_l(x)$ is used for generation of new candidate solutions.

New probability vector is created based on the probability vector of the previous generation in addition to the recently sampled solutions.

7.1.4 Compact Genetic Algorithm (cGA)

Similarly to PBIL algorithm cGA uses real-valued probability vector which represents probability of occurrence of each component in candidate solution [42].

7.1.5 Factorization Distribution Algorithm (FDA)

The algorithm uses the same probabilistic model during the whole optimization process of the algorithm [43]. The algorithm requires factorization and decomposition of the problem to be given by an expert what is usually not available. This differs from the classic EDA algorithm where the probabilistic model of the candidate solutions is usually created in every generation.

7.1.6 Bayesian Optimization Algorithm (BOA)

In the algorithm the model of the promising solutions is realized using Bayesian network [44]. The process of learning of the Bayesian network consist of searching for the suitable structure using a search algorithm and scoring every candidate using scoring metric. As the scoring metric Bayesian Dirichlet equivalence (BDe) method is used. The complexity of the search space is reduced allowing only k incoming edges to every node of the Bayesian network.

There have been published several extensions of BOA algorithm.

Mixed Bayesian Optimization Algorithm (MBOA) extends algorithm BOA to using mixed type variables (binary and continuous) [45]. MBOA is based on binary decision trees and idea of CART model (Classification and Regression Tree).

MBOA is further modified by combination with variance adaptation as implemented in Evolutionary Strategies. Experiments have shown that the modified algorithm called Adaptive Mixed Bayesian Optimization Algorithm (AMBOA) [46] is more efficient than the original MBOA algorithm.

Another extension of BOA is Hierarchical Bayesian Optimization Algorithm (hBOA^{TM}) [47] which combines Bayesian optimization algorithm, local structures in Bayesian networks and a powerful niching technique. Experiments have shown that ability of hBOA of solving hierarchical traps and other difficult problems is very strong.

7.2 UMDA for Problems with Unitation Constraints

As was described in section 5.6 the problem of analog circuit synthesis has to be viewed as problem with unitation constraints [48] [49]. Therefore modified version of UMDA algorithm which is able to handle problems with unitation constraints has to be used. Modification of Factorized Distribution Algorithm (FDA) [43] algorithm which enables to solve problems with unitation constraints was described in [48]. Application of UMDA algorithm for problems with unitation constraints was presented in [49]. The proposed modification of UMDA algorithm which enables

to solve problems with unitation constraints was inspired by [48]. Pseudo-code of original UMDA algorithm [27] is presented in Fig. 7.2.

step0: Set $k = 1$. Generate $n_i \gg 0$ points randomly.
step1: Select $n_s \leq n_i$ points according to a selection schedule. Compute the marginal frequencies $r_{k;i}(x_i)$ of the selected set.
step2: Generate n_i new points according to the distribution $q_{k+1}(x) = \prod_{i=1}^{n} r_{k;i}(x_i)$. Set $k = k + 1$.
step3: If not terminated, go to **step1**.

Fig. 7.2: Pseudo-code of original UMDA.

As was described in paper [48] only generation phase (sampling) of FDA algorithm was modified to handle unitation constraints problems. Presented approach was adopted also in the proposed modification of UMDA algorithm. UMDA algorithm was implemented using toolbox MATEDA 2.0 [59]. Pseudo-code of modified UMDA algorithm is presented in Fig. 7.3.

step0: Set $k = 1$. Generate $n_i \gg 0$ points randomly.
step1: Select $n_s \leq n_i$ points according to a selection schedule. Compute the marginal frequencies $r_{k;i}(x_i)$ of the selected set.
step2: Generate n_i new points according to the distribution $q_{k+1}(x) = \prod_{i=1}^{n} r_{k;i}(x_i)$. Set $k = k + 1$.
step3: With regard to unitation constraints repair generated points.
step4: If not terminated, go to **step1**.

Fig. 7.3: Pseudo-code of modified UMDA.

One additional phase was added to the original UMDA. In **step3** generated samples are repaired to satisfy desired unitation constraints.

The repairing function is applied to every individual of the population of generated samples in **step2**. Only n_c "ones" with the highest marginal frequencies $r_{k;i}$ of every generated sample are accepted. The rest of "ones" of the samples are set to zero. If the number of "ones" of the sample is equal or lower than n_c then the sample is accepted without any modification and no repairing is performed. This way number of "ones" (what corresponds to the number of the components of the encoded analog circuit) of every generated sample is always n_c or less.

Note that the repairing function is applied only for the part of the encoding vector which encodes the topology of the solution.

7.3 Hybrid Genetic Algorithm

Hybrid genetic algorithm [50] is formed of classical genetic algorithm and one or more another search methods. The motivation for using of this approach is demand for enhancing search capabilities, improving accuracy of the solutions, improving efficiency (faster convergence speed, lower population size) or to guarantee feasible solutions [50].

Two methods of incorporation of a local search method into genetic algorithm - Lamarckian and Baldwinian hybrid genetic algorithm will be briefly presented in the section. Local search methods use local knowledge to help genetic algorithm to locate the promising individuals in the population and prefer them during the selection process.

7.3.1 Lamarckian Hybrid Genetic Algorithm

In every generation of Lamarckian genetic algorithm local search algorithm tries to improve accuracy of defined percentage of individuals of the population. The individuals and their fitness values are modified to match the result found by local search method. Since Local search method tries to improve selected individuals of the population it can be called refinement genetic operator.

7.3.2 Baldwinian Hybrid Genetic Algorithm

As in the case of Lamarckian Hybrid Genetic Algorithm, local search algorithm tries to improve accuracy of defined portion of individual of the population. However compared to Lamarckian approach genotypes of the selected individuals are not modified. Only fitness values of the individuals are modified according to the results obtained by the local search method. Improvement of the fitness value after execution of the local search method reflects potential of the individual to improve its quality. This way chances of selection of individuals with potential to improve theirs fitness values during selection phase of the genetic algorithm are increased.

8 EVOLUTIONARY SYNTHESIS OF ANALOG CIRCUITS USING EDA

Analog circuit design method using autonomous circuits theory was briefly described in section 5.2. Design flow of the method is presented in Fig. 8.1. Let's assume that complexity of the initial autonomous circuit was selected based on the target application of the desired circuit.

step1: Generalized admittances of the autonomous circuit are replaced by real components, their parallel combination or open circuit.
step2: Characteristic equations of derived topologies are analyzed.
step3: Using characteristic equations of the promising topologies values of the components are calculated.

Fig. 8.1: Procedure of manual design of analog circuits using autonomous circuits approach.

All of the presented steps of the design were performed manually taking into account some designer's intuition or previous experience. The principle of the automated analog circuit synthesis method proposed in the thesis is to automatize described process of the replacement of the generalized admittances of the autonomous circuit using Estimation of Distribution Algorithms. Thus the intuition and the previous experience of the designer will be replaced by probabilistic modeling of the promising areas of the design search space and generating of high probable solutions. Pseudo-code of the proposed automated synthesis method which is inspired by manual design presented in Fig. 8.1 is presented in Fig. 8.2. Let's assume that complexity of the autonomous circuit is already chosen.

step0: Set $k = 1$. Randomly initialize population of solutions $P(1)$.
step1: Select set of promising individuals $P_s(k)$ of population $P(k)$.
step2: Build probabilistic model $M(k)$ of selected individuals $P_s(k)$.
step3: Using probabilistic model $M(k)$ generate population of new solutions $P_g(k)$.
step4: Set $k = k + 1$. Based on $P(k-1)$ and $P_g(k-1)$ create new population $P(k)$. If termination criteria are not met go to **step1**.

Fig. 8.2: Pseudo-code of the proposed automated analog circuit synthesis method based on EDA.

In the next sections three different types of automated analog circuit synthesis methods based on Estimation of Distribution Algorithms will be presented.

The first proposed method is based on UMDA [27] algorithm which is used to synthesize the topology and the parameters of the desired admittance network. The method can be viewed as improved version of method presented in [28]. As was described in section 5.6 the problem of analog circuit synthesis has to be solved as problem with unitation constrains [48]. Therefore modified version of UMDA as described in section 7.2 was used. The method is verified on problem of synthesis of admittance network with desired input impedance. Afterwards the synthesized network is employed in a chaotic oscillator circuit which is simulated.

In the second proposed method hybrid approach based on UMDA and a local search algorithm is used. The topology of the desired analog circuit is synthesized using UMDA and the parameters of the desired circuit are determined using the local search algorithm. As in the previous method modified version of UMDA (section 7.2) was used. For verification of the proposed method problem of synthesis of fractional capacitor circuit is adopted.

Compared to the first two presented methods which were focused on synthesis of passive analog circuits the third proposed method is able to synthesize active analog circuits as well. The method is based on utilization of univariate marginal probability model where individuals of the population are represented as graphs and hypergraphs. Hybridization with local search algorithm is employed and used for solving of the parameters of the components. For verification of the synthesis capability of the proposed method problem of analog circuit realization of cube root function was adopted.

9 EVOLUTIONARY SYNTHESIS OF CHAOTIC DYNAMICS USING PURE UMDA

The paper is focused on the synthesis of the topology and the parameters of analog impedance network with arbitrary input impedance function using the univariate marginal distribution algorithm. As a test problem design of chaotic oscillator circuit is adopted. Impedance network with desired input impedance function is synthesized and employed in the chaotic oscillator circuit. Afterwards functionality of the oscillator is verified using PSpice simulation.

9.1 Definition of the Problem

9.1.1 Mathematical Background of Chaotic Dynamics

Suppose the general class of three-segment piecewise-linear vector fields which can be described by the following set of the differential equations [51]:

$$\dot{x} = y \qquad \dot{y} = \dot{z} \qquad \dot{z} = q_1 z - q_2 y + q_3 x + h(.) \tag{9.1}$$

where argument of the nonlinear feedback function is:

$$h(.) = [(p_1 - q_1)z - (p_2 - q_2)y + (p_3 - q_3)x] \tag{9.2}$$

and function itself:

$$h(x) = 0.5(|x + 1| - |x - 1|) \tag{9.3}$$

Generally argument of the h(.) is a linear combination of all state variables. Having this mathematical model of some real physical system we have also some clues about geometry of the vector field. First of all the state space is separated by two parallel boundary planes into three linear segments. The equation for each plane can be considered as an argument of h(.). The dynamical motion in each segment is uniquely determined by eigenvalues, i.e. roots of the characteristic polynomials. For inner segment we get:

$$\lambda^3 - p_1\lambda^2 + p_2\lambda - p_3 = 0 \tag{9.4}$$

and analogically for both outer segments:

$$\lambda^3 - q_1\lambda^2 + q_2\lambda - q_3 = 0 \tag{9.5}$$

It is evident that the system behavior can be also uniquely specified by the so-called equivalent eigenvalues, p_i and q_i which are always real numbers. Typically chaotic solution is characterized by both stable and unstable manifolds for the individual fixed points. Recently it turns out that this is not a strict requirement for the existence of chaotic solution. For example the configuration:

$$\begin{aligned} p_1 = 0.363 \quad p_2 = 1.063 \quad p_3 = 0.277 \\ q_1 = -10.286 \quad q_2 = -1.2 \quad q_3 = -2.719 \end{aligned} \tag{9.6}$$

as well as values:

$$\begin{aligned} p_1 = 0.8 \quad p_2 = 100.21 \quad p_3 = 20.018 \\ q_1 = -2.4 \quad q_2 = -0.71 \quad q_3 = -3.27 \end{aligned} \tag{9.7}$$

lead to the following geometry of the vector field:

$$D_o: \ \Re^3 \in E_u^2 \oplus E_u^1 \qquad D_{\pm 1}: \ \Re^3 \in E_u^2 \oplus E_s^1 \tag{9.8}$$

where D_0 denotes inner segment and symbol $D_{\pm 1}$ marks outer segments. Speaking in numerical terms the set (9.6) leads to the eigenvalues:

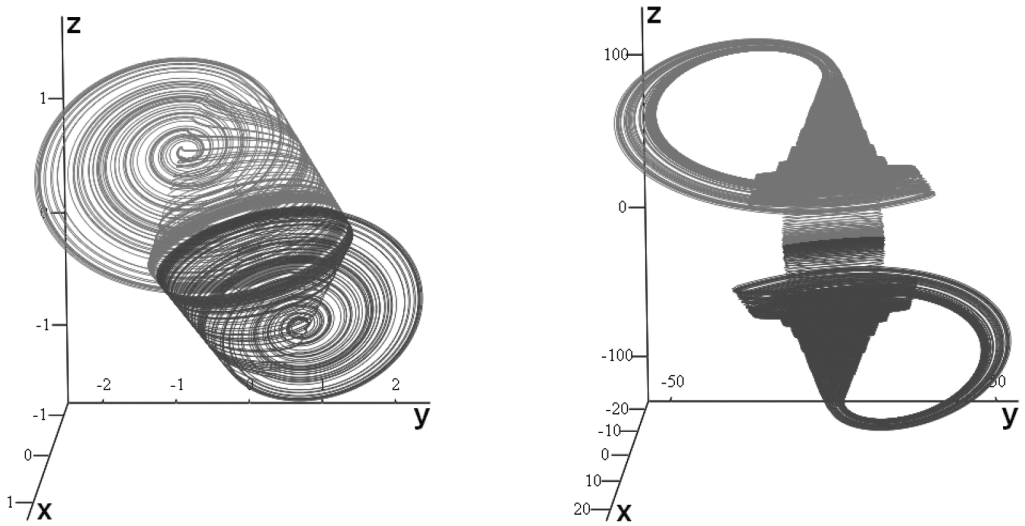
$$\begin{aligned} D_o: \ \lambda_{1,2} = 0.048 \pm 1.017j \quad \lambda_3 = 0.267 \\ D_{\pm 1}: \ \lambda_{1,2} = 0.07 \pm 0.506j \quad \lambda_3 = -10.426 \end{aligned} \tag{9.9}$$

Analogically for the set (9.7) results into the eigenvalues:

$$\begin{aligned} D_o: \ \lambda_{1,2} = 0.3 \pm 10j \quad \lambda_3 = 0.2 \\ D_{\pm 1}: \ \lambda_{1,2} = 0.3 \pm 1j \quad \lambda_3 = -3 \end{aligned} \tag{9.10}$$

It seems that dynamical system described by (9.1) and (9.2) with an fully unstable equilibria at origin is something special because the state space trajectory entering inner segment is repealed towards outer segments along every direction. Since by definition the trajectory can not cross an eigenspace system has two symmetrical chaotic attractors separated by an unstable eigenplane. From the viewpoint of circuit theory this property also suggest not only one but rather several grounded and/or floating one-ports. Negative resistors, capacitors and inductors are unwanted since it makes the final circuits much more complicated. The number of the negative elements can be minimalized as will be clarified later.

As the reference state space trajectory the numerical integration of the mathematical model together with a given set of the parameters can be considered. For this purpose Mathcad and build-in fourth-order Runge-Kutta method has been used. The typical chaotic attractor for (9.6) is shown in Fig. 9.1a and similarly for (9.7) it is demonstrated in Fig. 9.1b.



(a) (b)

Fig. 9.1: a) Numerical integration of the first set of parameters (9.1) b) Numerical integration of the second set of parameters (9.2).

9.1.2 Circuit Realization of the Proposed Chaotic Oscillator

The proposed realization of the chaotic oscillator consists of parallel connection of two parts, PWL resistor and admittance network Y(s). The block diagram is presented in Fig. 9.2.

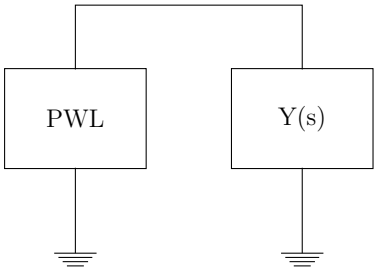


Fig. 9.2: Block diagram of the proposed chaotic oscillator.

The three-segment PWL resistor is realized by diode limiter and imitance converters. Further details can be found in [52]. Since in our case the experimental verification is restricted to PSpice circuit simulations the idealized approach can be adopted. One possible implementation of the function (9.3) is provided in Fig. 9.3a. The breakpoints are defined by the voltage controlled switches, where on and off states are represented by $1\mu\Omega$ and $1\mathrm{T}\Omega$ resistor on the switch output port. Input DC characteristic of the PWL resistor is presented in Fig. 9.3b.

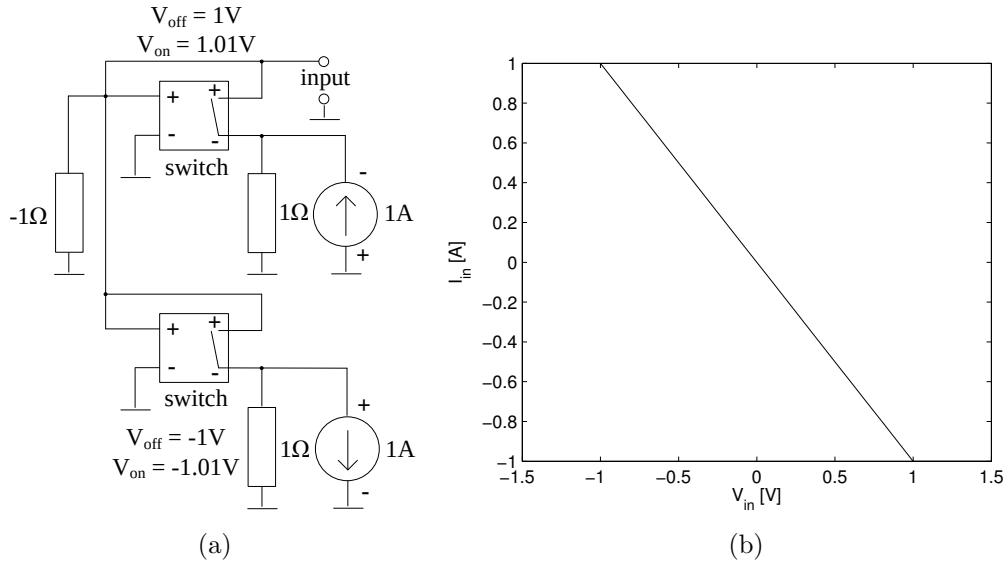


Fig. 9.3: a) Schematic of PWL resistor b) Input DC characteristic of PWL resistor.

The second part, admittance network Y(s), will be synthesized using pure UMDA algorithm as will be described in the next section. Desired impedance function of the admittance network is (9.11) for the first set of the parameters and (9.12) for the second set of the parameters.

$$Z_{in1} = \frac{10.649s^2 - 2.263s + 2.996}{s^3 + 10.286s^2 - 1.2s + 2.719} \tag{9.11}$$

$$Z_{in2} = \frac{3.2s^2 - 100.92s + 23.288}{s^3 + 2.4s^2 - 0.71s + 3.27} \tag{9.12}$$

Magnitude and phase characteristics for the first impedance function (9.11) are presented in Fig. 9.4a and Fig. 9.4b respectively. Magnitude and phase characteristics for the second impedance function (9.12) are presented in Fig. 9.4c and Fig. 9.4d respectively.

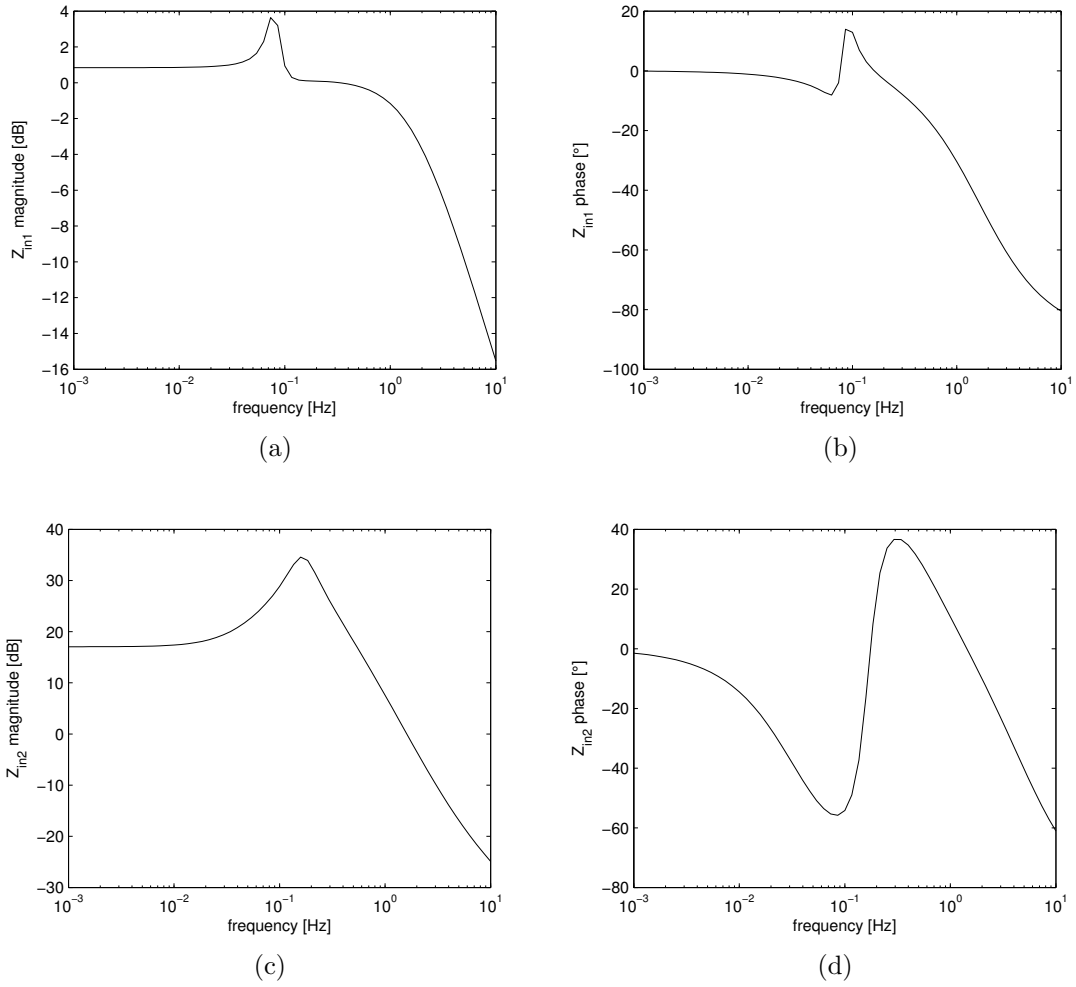


Fig. 9.4: a) Magnitude of input impedance Z_{in1} b) Phase of input impedanceZ_{in1} c) Magnitude of input impedance Z_{in2} d) Phase of input impedance Z_{in2}

9.2 Description of the Method

9.2.1 Used Encoding

As was described in section 5.6 the problem of synthesis of the topology of analog circuit employing encoding method presented in section 5.5.3 has to be solved as problem with unitation constraints [48]. Topology and values of the components of the desired admittance network are synthesized using modified UMDA algorithm as was described in section 7.2. Note that the repairing function (section 7.2) is applied only for the part of encoding vector e which includes information about the topology of the solution (bits $b1$ to $b135$).

As an encoding method characteristic vector approach as described in section 5.5.3 is used. Number of the nodes of the used AC (5.1) was chosen to $n_n = 10$. Therefore according to (5.2) the topology is encoded using binary vector of length 135 bits.

Values of the components are represented in mantissa-exponent form where every single mantissa v_{man} is encoded using 7 bits and every single exponent v_{exp} is encoded using 3 bits. Since maximal number of the components n_c is set to 15, encoding vector e encodes parameters for 15 components. Bits $b136$ to $b180$ encode exponents. Bits $b181$ to $b285$ encode mantissas. Gray coding for the exponents and mantissas was used. See Fig. 9.5 for schematic diagram of the whole used encoding vector e.

$$e = [\ \overbrace{b1\ b2\ b3\ \dots\ b135}^{\text{topology}}\ ]\ [\ \overbrace{b136\ b137\ b138\ \dots\ b180}^{\text{exponents}}\ ]\ [\ \overbrace{b181\ b182\ b183\ \dots\ b285}^{\text{mantissas}}\ ]$$

Fig. 9.5: Schematic diagram of the used encoding vector e.

Note that only topological part (bits $b1$ to $b135$) of the encoding vector e is viewed as problem with unitation constraints (modified UMDA approached used). The rest of the encoding vector e (bits $b136$ to $b285$) is solved as in classical UMDA algorithm.

9.2.2 Objective Function

Cost value *cost* is according to (9.15) computed as weighted summation of magnitude differences (9.13) and phase differences (9.14). Difference of magnitude Δ_m is calculated as weighted absolute value of differences between desired magnitude function f_{md} and magnitude of current solution f_{mc} over 61 frequency points in range 1e-3 rad/s to 10 rad/s. Similarly difference of phase Δ_p is calculated as weighted absolute value of differences between desired phase function f_{pd} and phase of current solution f_{pc}.

$$\Delta_m = \frac{1}{m}\sum_{i=1}^{m} w_{dm}|f_{md}(i) - f_{mc}(i)| \tag{9.13}$$

$$\Delta_p = \frac{1}{m}\sum_{i=1}^{m} w_{dp}|f_{pd}(i) - f_{pc}(i)| \tag{9.14}$$

$$cost = \Delta_m w_{cm} + \Delta_p w_{cp} \tag{9.15}$$

For the first set of parameters (9.1) weights w_{cm} and w_{cp} are set to 1 and 0.35 respectively. For the second set of parameters (9.2) weights w_{cm} and w_{cp} are set to 1 and 0.1 respectively. Setting of weights w_{dm}, w_{dp} for both sets of parameters is summarized in Tab. 9.1.

Parameters (9.1)			Parameters (9.2)		
angular frequency (rad/s):	w_{dm}	w_{dp}	angular frequency (rad/s):	w_{dm}	w_{dp}
1e-3 to 0.0293	1	1	1e-3 to 0.0293	1	1
0.0341 to 0.1359	2	2	0.0341 to 0.1359	2	2
0.1585 to 10	1	1	0.1585 to 10	1	1

Tab. 9.1: Setting of weights w_{dm} and w_{dp}.

Frequency responses of the current solution f_{mc} and f_{pc} are obtained using nodal analysis method implemented in Matlab.

9.2.3 Settings of the Algorithm

The algorithm is configured to use only passive RLC components. Values of the resistors can be also negative. Maximal circuit complexity of the desired solution is restricted to use AC of 10 nodes and 15 components ($n_c = 15$) at the most. Input is fixed to node 1. Initial population is selected randomly with uniform probability distribution. Population size is set to 200 individuals. There were performed 500 generations in every single run of the algorithm what corresponds to 1e5 objective function evaluations per a run. Summary of all parameters of the presented synthesized problem is presented in Tab. 9.2. Algorithm UMDA is realized using Matlab toolbox Mateda 2.0 (section 13.1).

maximal number of nodes	10
maximal number of components	15
used types of components	R,L,C
negative resistors	allowed
problem complexity	285 bits (6.2e85)
angular frequency range	1e-3 rad/s to 10 rad/s
number of points	61 (15 points/decade)
population size	200
generations per run	500
objective function evaluations per run	1e5

Tab. 9.2: Summary of the parameters of the used UMDA algorithm.

Configuration of the initialization file of toolbox Mateda 2.0 for realization of UMDA algorithm is presented in Fig. 9.6.

```
PopSize = 200; n = 285; cache  = [0,0,0,0,0];
Card = 2*ones(1,n); MaxGen = 500;
Cliques = CreateMarkovModel(n, 0);
edaparams{1} = {'learning_method','LearnFDA',{Cliques}};
edaparams{2} = {'sampling_method','SampleFDAmodif',{PopSize}};
edaparams{3} = {'stop_cond_method','maxgen_maxval',stop_cond_params};
[AllStat,Cache]=RunEDA(PopSize, n, F, Card, cache, edaparams);
```

Fig. 9.6: Configuration of toolbox Mateda 2.0 for realization of UMDA algorithm.

9.2.4 Experiments and Solutions

There were performed 20 runs of the presented algorithm for each set of the parameters (9.1)(9.2). Average run time of the algorithm was 4 minutes and 27 seconds. The results for the first set of the parameters (9.1) are presented in Tab. 9.3. The results for the second set of the parameters (9.2) are presented in Tab. 9.4.

id of run	1	2	3	4	5	6	7	8	9	10
cost value	1.23	1.58	1.78	0.17	1.55	0.60	0.93	1.06	1.36	1.21
id of run	11	12	13	14	15	16	17	18	19	20
cost value	1.36	1.20	1.23	1.72	0.2	0.31	1.15	1.35	0.82	1.37

Tab. 9.3: Results for the first set of the parameters (9.1).

id of run	1	2	3	4	5	6	7	8	9	10
cost value	2.85	4.69	10.3	6.95	0.16	0.62	4.52	6.49	1.93	4.61
id of run	11	12	13	14	15	16	17	18	19	20
cost value	5.37	2.69	7.57	2.34	9.97	4.34	5.31	3.23	3.02	6.25

Tab. 9.4: Results for the second set of the parameters (9.2).

As you can see in Tab. 9.3 the best results for the first set of the parameters were achieved in runs 4 and 15. Schematic of solution 4 and its magnitude and phase characteristics are presented in Fig. 9.7. Schematic of solution 15 and its magnitude and phase characteristics are presented in Fig. 9.8.

For the second set of the parameters the best results were achieved in runs 5 and 6. Schematic of solution 5 and its magnitude and phase characteristics are presented in Fig. 9.9. Schematic of solution 6 and its magnitude and phase characteristics are presented in Fig. 9.10.

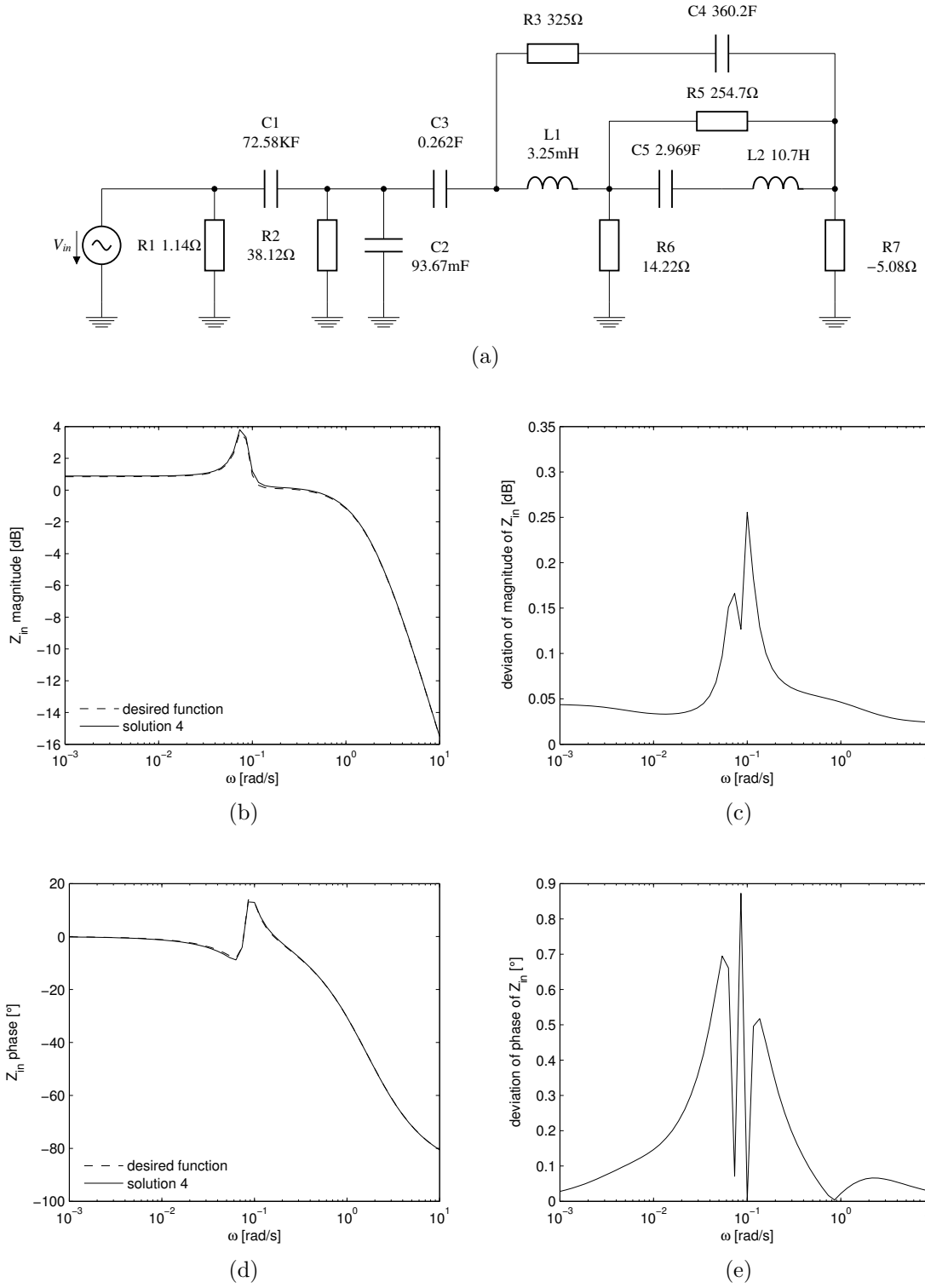
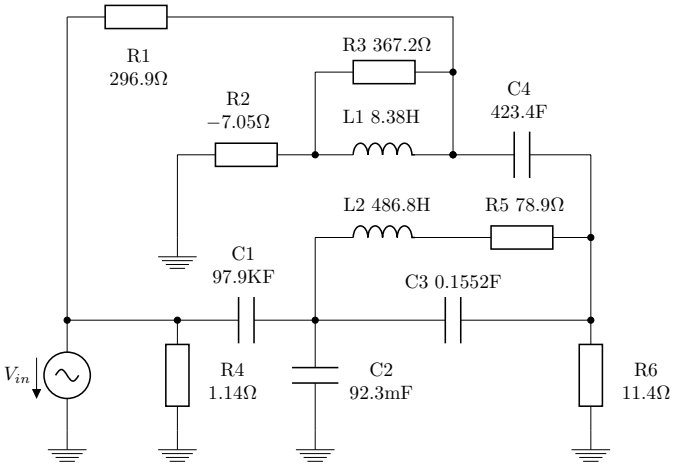
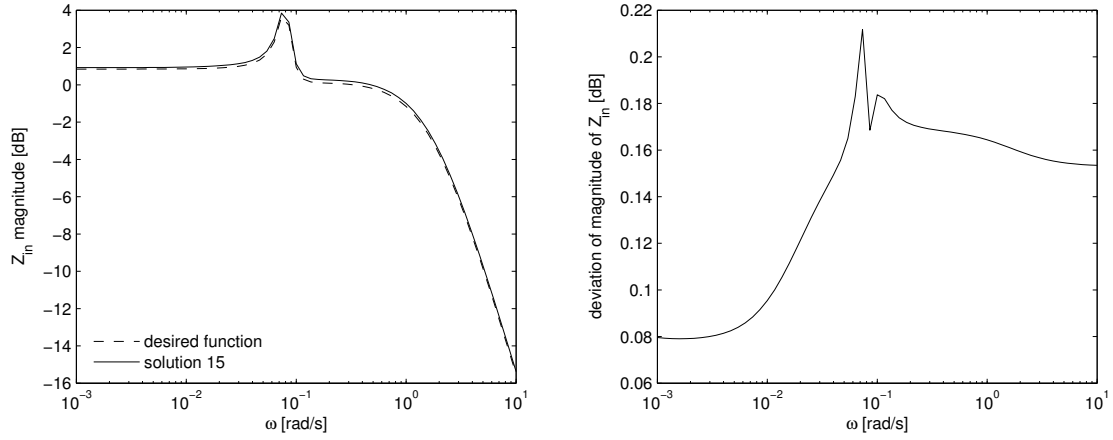


Fig. 9.7: Parameters (9.1), solution 4: schematic of the synthesized circuit (a), magnitude of Z_{in} (b), deviation of magnitude Z_{in} and desired function (9.11)(c), phase of Z_{in} (d), deviation of phase Z_{in} and desired function (9.11)(e)

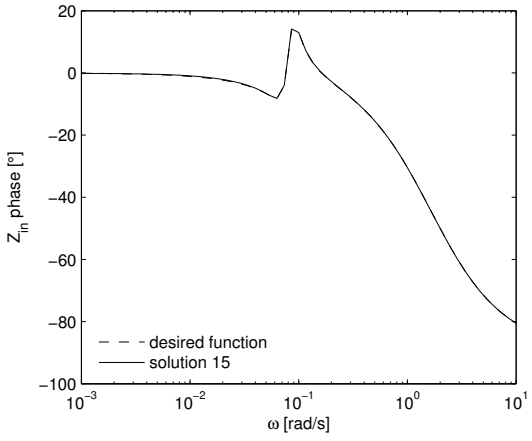


(a)

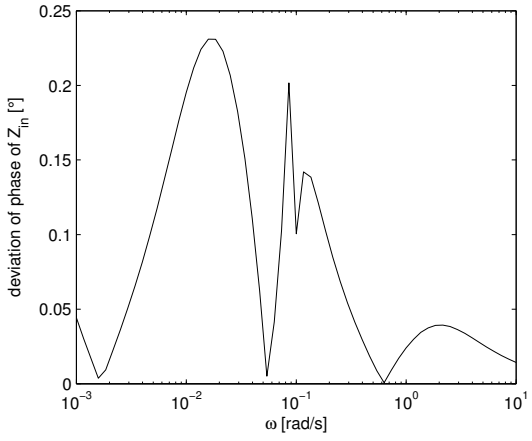


(b)

(c)

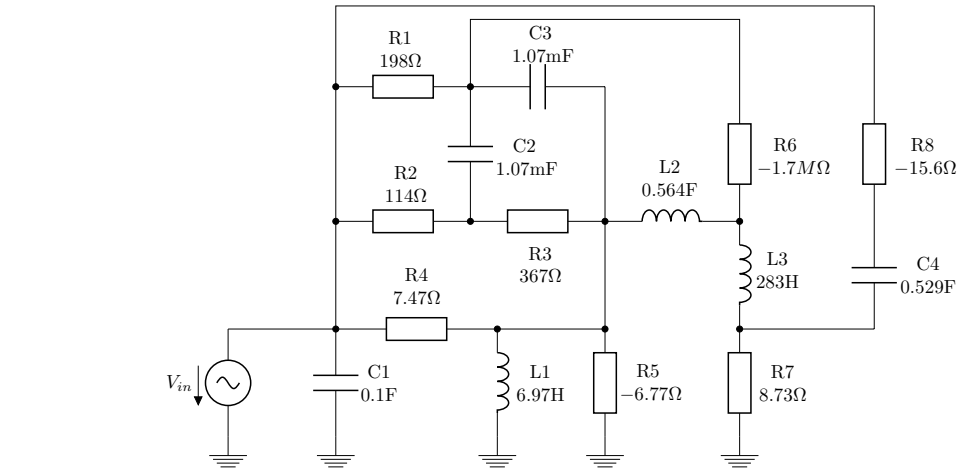


(d)

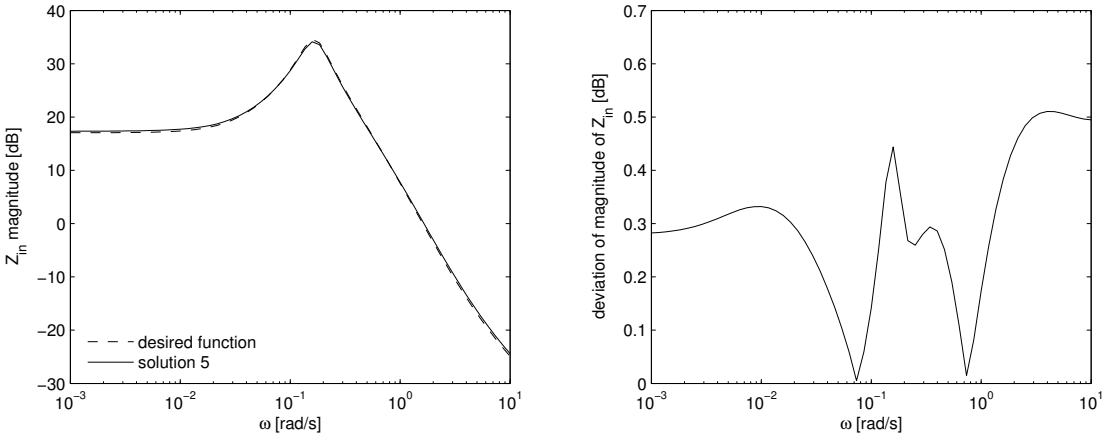


(e)

Fig. 9.8: Parameters (9.1), solution 15: schematic of the synthesized circuit (a), magnitude of Z_{in} (b), deviation of magnitude Z_{in} and desired function (9.11)(c), phase of Z_{in} (d), deviation of phase Z_{in} and desired function (9.11)(e)

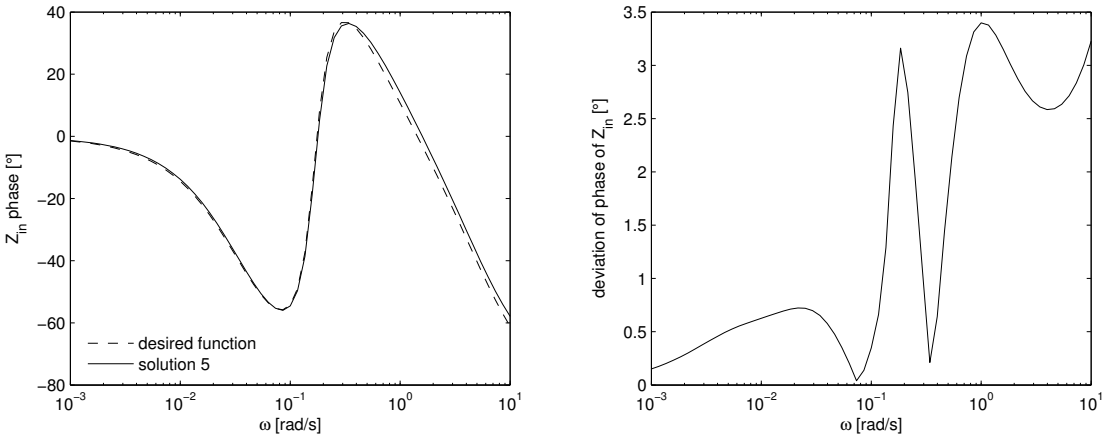


(a)



(b)

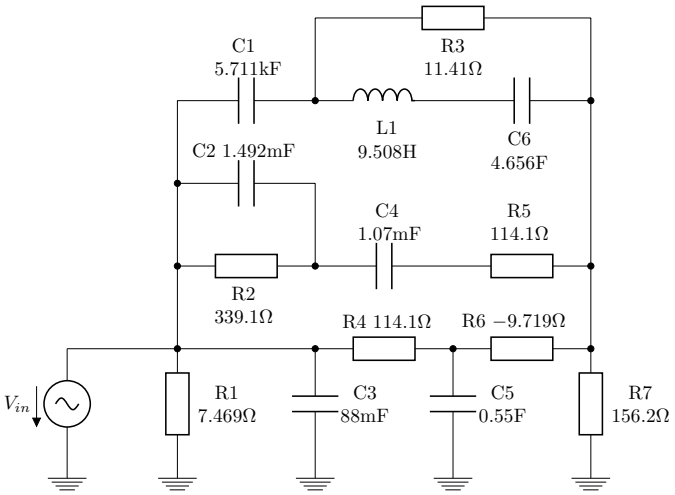
(c)



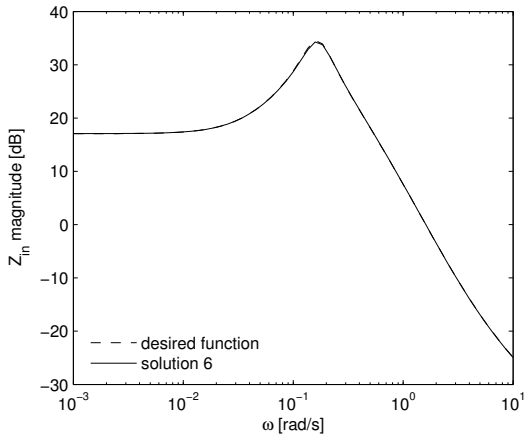
(d)

(e)

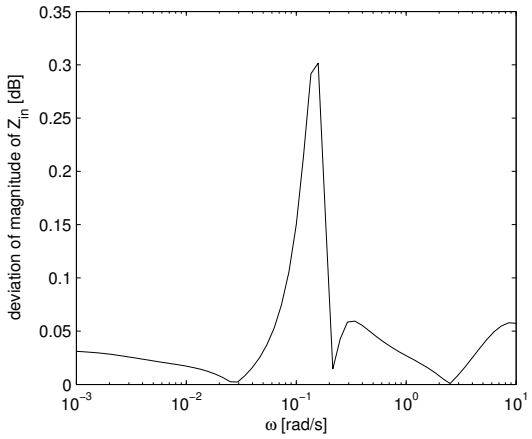
Fig. 9.9: Parameters (9.2), solution 5: schematic of the synthesized circuit (a), magnitude of Z_{in} (b), deviation of magnitude Z_{in} and desired function (9.12)(c), phase of Z_{in} (d), deviation of phase Z_{in} and desired function (9.12)(e)



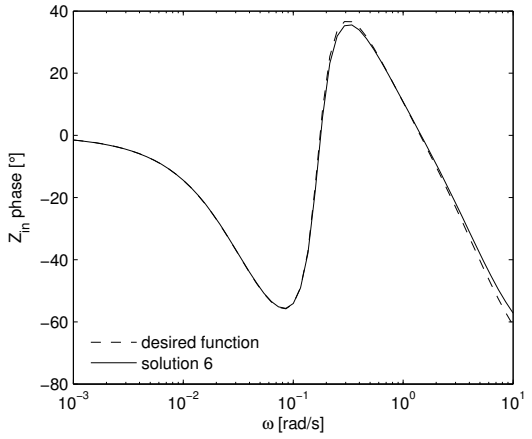
(a)



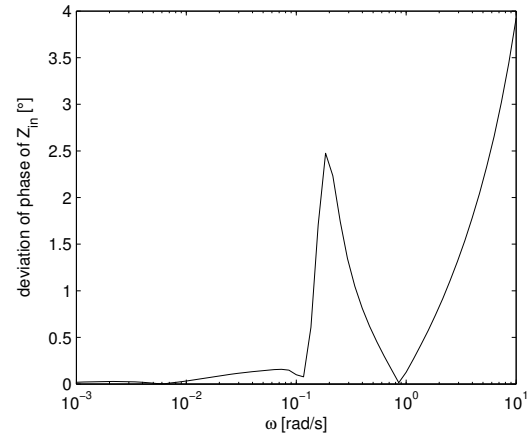
(b)



(c)



(d)

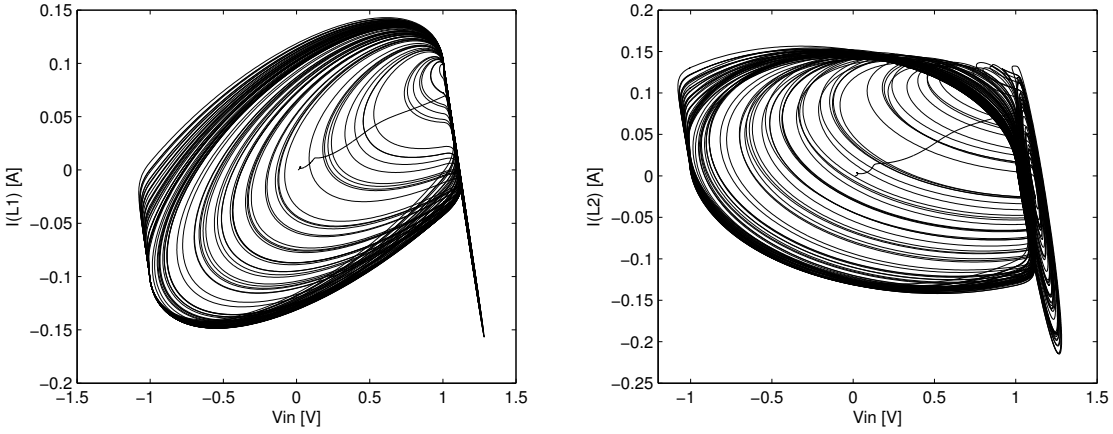


(e)

Fig. 9.10: Parameters (9.2), solution 6: schematic of the synthesized circuit (a), magnitude of Z_{in} (b), deviation of magnitude Z_{in} and desired function (9.12)(c), phase of Z_{in} (d), deviation of phase Z_{in} and desired function (9.12)(e)

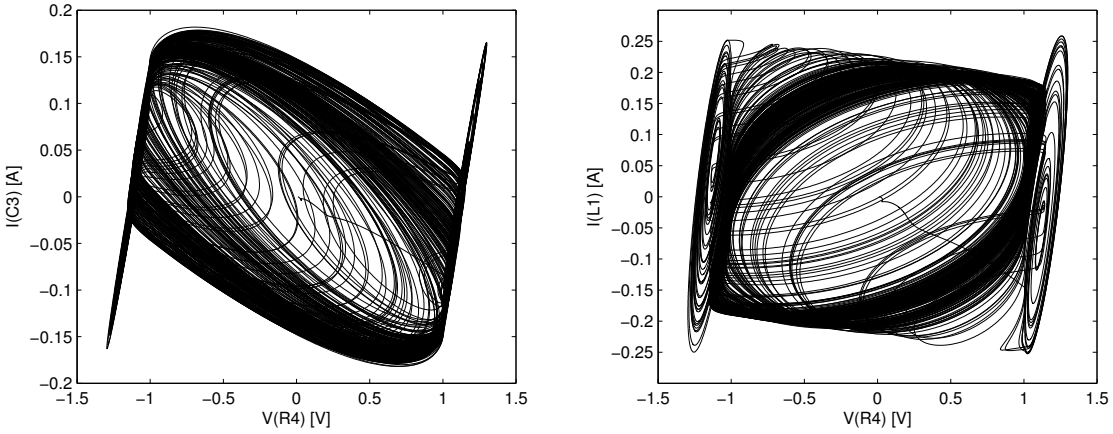
9.2.5 Simulation of the Chaotic Oscillator

For verification of the synthesized admittance networks the resulting structures are employed in the proposed chaotic oscillator circuit. Examples of the state projections of the output signals of the oscillator for parameters (9.1) and solutions 4 and 15 are presented in Fig. 9.11 and Fig. 9.12 respectively. Examples of the state projections of the output signals of the oscillator for parameters (9.2) and solutions 5 and 6 are presented in Fig. 9.13 and Fig. 9.14 respectively.



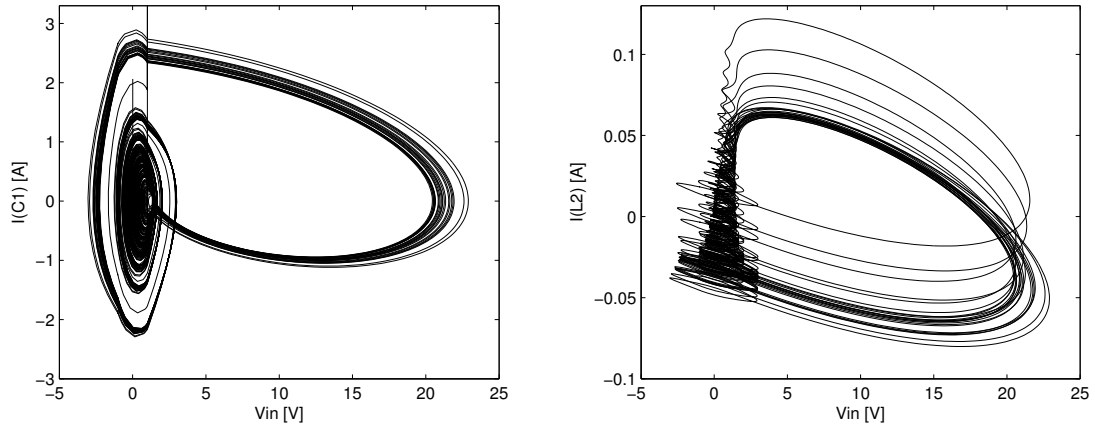
(a) (b)

Fig. 9.11: State projection for parameters (9.1) and solution 4: I(L1) vs. Vin (a), I(L2) vs. Vin (b)



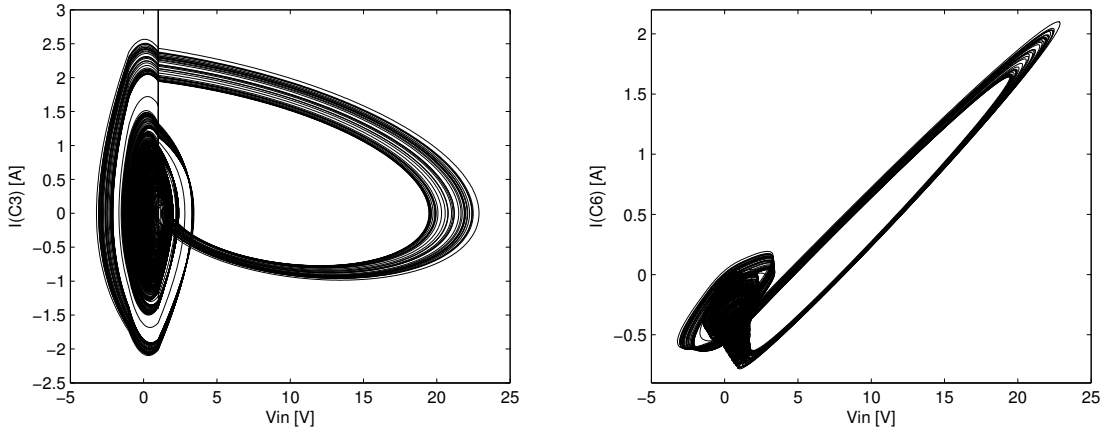
(a) (b)

Fig. 9.12: State projection for parameters (9.1) and solution 15: I(C3) vs. V(R4) (a), I(L1) vs. V(R4) (b).



(a) (b)

Fig. 9.13: State projection for parameters (9.2) and solution 5: I(C1) vs. Vin (a), I(L2) vs Vin (b).



(a) (b)

Fig. 9.14: State projection for parameters (9.2) and solution 6: I(C3) vs. Vin (a), I(C6) vs. Vin (b).

9.3 Conclusion

Method of evolutionary synthesis of passive analog network with desired input impedance function using UMDA algorithm was presented. Both parts of the analog circuit synthesis problem (synthesis of the topology and determination of the parameters) were solved using UMDA. To enable UMDA to deal with the problem of determination of the parameters which is continuous problem in nature discretization of the parameters was used. Algorithm UMDA was modified to enable solving problems with unitation constraints.

Compared to method presented in [28] the proposed method enables encoding of parallel connection of the components and synthesis of circuits with higher complexity.

Four analog networks for two sets of the parameters were synthesized. Comparison of magnitude and phase characteristics of the synthesized circuits and desired function show very good accuracy of the synthesized solutions. The synthesized circuits were verified in the chaotic oscillator and resulting state projections were presented. Correct function of the chaotic oscillator confirms good accuracy of the synthesized analog networks.

The problem of synthesis of analog network for purpose of the chaotic oscillator was solved also in [3] where simulated annealing method (SA) was used. SA was capable to produce solutions with similar accuracy using comparable number of objective function evaluations. However there are two drawbacks of using SA. Since SA is not population based evolutionary algorithm there is no possibility of parallel evaluation of the objective function. The second problem is that SA is single objective optimization method. Although this issue can be partially solved using weighted summation of the fitness values of the particular objectives fully multiobjective approaches as pareto approach can not by applied here.

Compared to SA the proposed method is population based evolutionary algorithm. Therefore parallel evaluation of objective function is possible. Also fully multiobjective approach such as pareto ranking can be applied.

10 SYNTHESIS OF FRACTIONAL CAPACITOR USING HYBRID UMDA ALGORITHM

10.1 Definition of the Problem

Authors of [53] describe approximation of fractional capacitors $(1/s)^{1/\mathrm{n}}$ by a regular newton process and they present its RLC network representations which are obtained using classical method of analog circuit synthesis. Another informations concerning the fractional order systems can be found in [54] [55] [56]. The problem of the synthesis of the circuit approximation of (10.1) was adopted from the mentioned paper and will be used for demonstration of the synthesis capability of the proposed method. In [53] function (10.1) was approximated using 5^{th} order function (10.2).

$$Z_{in} = s^{-0.6} \tag{10.1}$$

$$Z_{in} = \frac{8.58s^4 + 255s^3 + 405s^2 + 35.9s + 0.169}{1s^5 + 94.2s^4 + 472s^3 + 134.8s^2 + 2.627s + 9.8 \times 10^{-3}} \tag{10.2}$$

In Fig. 10.1 there is circuit realization of function (10.1) as presented in [53]. For the rest of the chapter, the circuit will be called original approximation circuit.

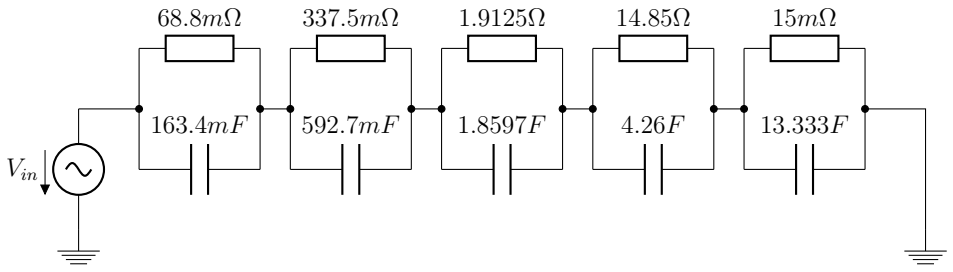


Fig. 10.1: Schematic of the original approximation circuit.

Comparison of magnitude of Z_{in} for original approximation circuit and for (10.1) is presented in Fig. 10.2a. The broken line represents function (10.1) and the solid line represents original approximation circuit. Since the deviation is not clear from the picture absolute value of difference of both curves presented in Fig. 10.2a is presented in Fig. 10.2b. Comparison of phase of Z_{in} for original approximation circuit and for (10.1) is presented in Fig. 10.2c. The absolute value of difference of both curves plotted in Fig. 10.2c is presented in Fig. 10.2d.

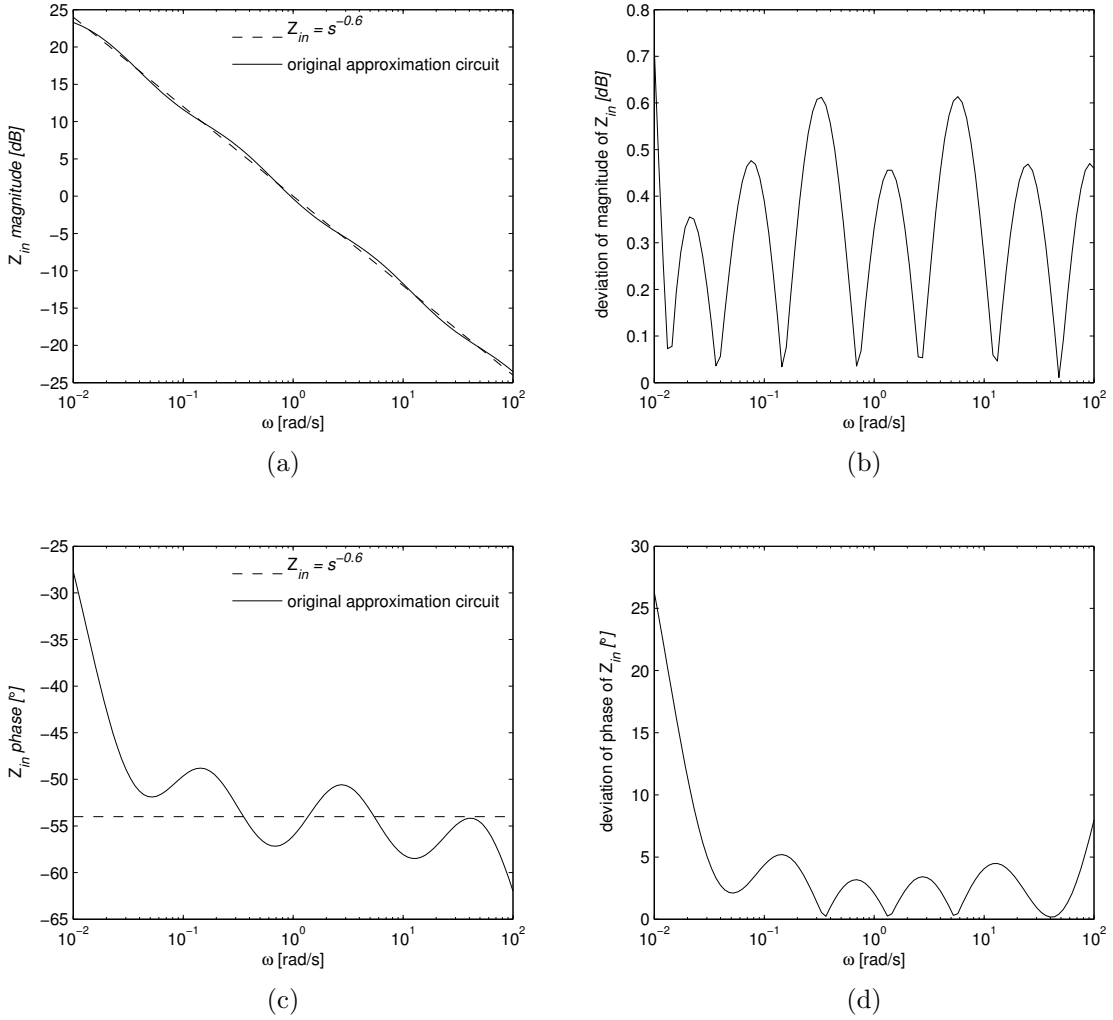


Fig. 10.2: a) Comparison of the magnitude characteristics of ideal function (10.1) and original approximation circuit b) deviation of the magnitude c) comparison of the phase characteristics of ideal function (10.1) and original approximation circuit d) deviation of the phase.

The highest deviations of magnitude and phase of the original approximation circuit are presented in Tab. 10.1.

Magnitude			
$\omega[rad/s]$	0.01	100	0.3311
$\Delta_m[dB]$	0.71	0.46	0.61

Phase			
$\omega[rad/s]$	0.01	100	0.14
$\Delta_p[\circ]$	26.3	7.98	5.2

Tab. 10.1: The highest deviations of magnitude and phase of the original approximation circuit.

The following sections will be focused on synthesis of the presented fractional capacitor circuit problem using hybrid UMDA algorithm.

10.2 Hybrid UMDA Algorithm

10.2.1 Main Flowchart of the Method

The proposed method is based on combination of UMDA and a local search algorithm. Its principal flow chart is presented in Fig. 10.3.

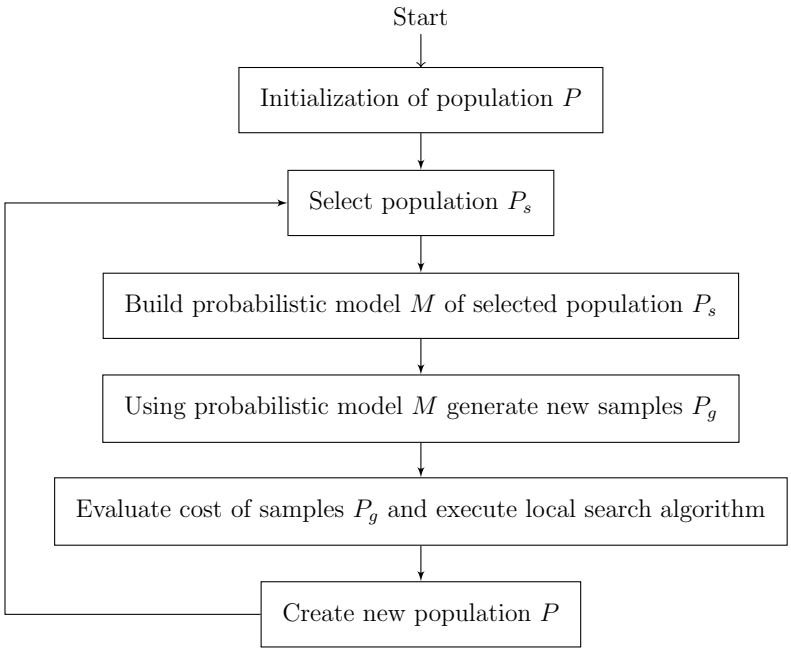


Fig. 10.3: Principal flowchart of the proposed method.

Population P is formed of binary vectors of length 135 bits which are initialized randomly with seeding of 10 bits ($n_c = 10$). Parameters storage e_{ps} is formed of vector of length 135 consisting of real numbers in the range <0,1>. Parameters storage e_{ps} is dynamically optimized during the whole synthesis process and it is adapted to the selected topologies in the selection phase of the algorithm. The vector includes a component value for every single admittance of the used AC. During initialization phase e_{ps} is set randomly with uniform distribution.

In the next step good individuals are selected using truncation selection method. Selected population P_s is formed of good individuals of the previous population P.

Probabilistic model M of selected population P_s is built. For more informations on building probabilistic model in UMDA algorithm please refer to [27].

Probabilistic model M built in the previous step is used for generation of new samples of solutions P_g. New samples are generated using Stochastic Universal Sampling method (SUS) and are repaired using the method described in section 7.2.

Generated samples are evaluated using the topological information stored in P_g and the parameters stored in e_{ps}. If the condition of execution of the local search

algorithm is fulfilled, the local search algorithm tries to optimize the parameters of the current solution. If accuracy of the current solution is improved then parameters storage e_{ps} is updated according to the results of the local search algorithm. Detailed description of the cost evaluation phase and the local search algorithm is presented in section 10.2.2.

In the next step new population P is formed of the best individuals of P_g and selected population P_s. Described process is repeated until one of the termination criteria of the algorithm is met.

10.2.2 Evaluation of Cost Value and Local Search Method

Evaluation of the cost values and using of the local search algorithm will be discussed in the section. Principal flow chart of this phase is presented in Fig. 10.4.

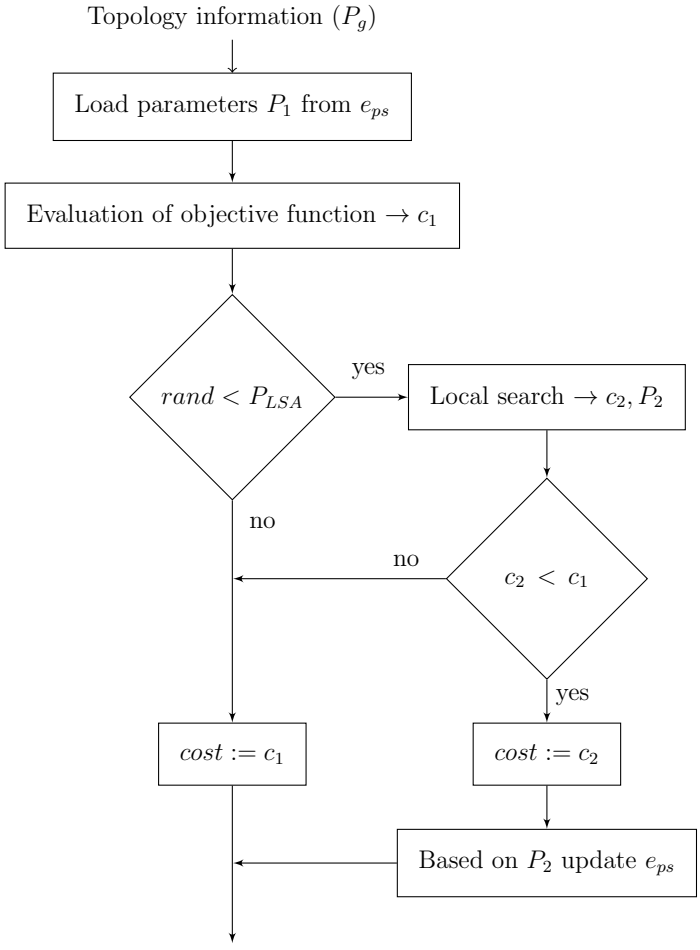


Fig. 10.4: Flow chart of the evaluation cost and local search phase.

The following procedure which is described bellow is performed for every single individual $P_g(i)$ of population of generated samples P_g.

Based on the topology information stored in the binary vector of individual $P_g(i)$ appropriate set of parameters P_1 is loaded from parameters storage e_{ps} and cost value c_1 of individual $P_g(i)$ is evaluated.

If the condition of execution of the local search algorithm (LSA) is fulfilled ($rand < P_{LSA}$) LSA tries to improve accuracy of individual $P_g(i)$. The probability of execution of LSA is set to $P_{LSA} = 0.02$. Initial point of the search of LSA is set to parameters set P_1. Number of maximal function evaluations of LSA is set to $MaxFunEvals = 100$. The results of LSA are cost value of optimized solution c_2 and set of optimized parameters P_2.

If LSA was successful in improving of the accuracy of individual $P_g(i)$ and its cost value was improved ($c_2 < c_1$) then value *cost* of individual $P_g(i)$ is set to c_2 ($cost := c_2$) and appropriate parameters of parameters storage e_{ps} are updated according to parameters set P_2.

If condition for execution of LSA was not fulfilled ($rand > P_{LSA}$) or LSA was not able to improve cost value of individual $G(i)$ ($c_2 \geq c_1$) resulting value *cost* of individual $P_g(i)$ is set to c_1 ($cost := c_1$).

After performing of the described process cost value *cost* of individual $P_g(i)$ and updated parameters storage e_{ps} are obtained. During the run of the algorithm the parameters stored in e_{ps} are adapted to the topological information of good individuals selected in the selection phase of the algorithm (Fig. 10.3). This way information about the topology stored in P_g and information about the parameters stored in e_{ps} are mutually optimized and the whole synthesis process is directed towards the promising areas of the solution space.

10.2.3 Encoding Method

As a encoding method characteristic vector approach as described in section 5.5.3 is used. Number of the nodes of used AC (section 5.1) was chosen $n_n = 10$. Therefore according to (5.2) the topology is encoded using binary vector of length 135 bits.

Schematic of the used encoding vector e is presented in Fig. 10.5. The topological information is encoded using set of binary variables S_t = {b1, b2, b3, ..., b135} and the parameters (values of the components) are encoded using set of real numbers S_p = {dbl1, dbl2, db3, ..., dbl10}.

$$e = [\ \overbrace{\text{b1 b2 b3} \ldots \text{b135}}^{\text{topology}}\ ]\ [\ \overbrace{\text{dbl1 dbl2 dbl3} \ldots \text{dbl10}}^{\text{parameters}}\ ]$$

Fig. 10.5: Schematic diagram of encoding vector e.

Based on the admittances selected in the topological part of the information (set S_t) corresponding parameters values are loaded from parameters storage e_{ps} and copied to the parameters part of encoding vector e (set S_p). For example let's assume that set S_t is chosen S_t = {b1, b3, b15, b18, b28, b52, b78, b92, b107, b115} then the corresponding set of the parameters loaded from parameters storage e_{ps} is S_l = {p1, p3, p15, p18, p28, p52, p78, p92, p107, p115}. Afterwards the assignment of set S_l to set S_p is p1→ dbl1, p3→dbl2, p15→dbl3, p18→dbl4, p28→dbl5, p52→dbl6, p78→dbl7, p92→dbl8, p107→dbl9, p115→dbl10.

Based on the parameters in set S_p the values of the components are calculated using formula (10.3)

$$v = \frac{2 \times 10^6}{1 + e^{(-1.4(10r-14))}} \tag{10.3}$$

, where r are values of the parameters loaded from parameters set S_p. Formula (10.3) was formed to map the values of the parameters stored in set S_p to suitable range of the component's values. Since the parameters in set S_p are set in the range <0,1> corresponding component values for the lowest $r = 0$ and for the highest $r = 1$ are $v_{min} = 0.0061$ and $v_{max} = 7.3685 \times 10^3$ respectively. Note that formula (10.3) is used for all three types of components RLC. Since the used angular frequency range is from 0.01 rad/s to 100 rad/s, the values of the components are set to non-realistic values.

10.2.4 Objective Function

Cost value *cost* is according to (10.6) computed as weighted summation of the magnitude and phase differences. Difference of magnitude Δ_m is according to (10.4) calculated as weighted absolute value of differences of desired magnitude function f_{md} and magnitude of current solution f_{mc} over $m = 101$ frequency points in range 0.01 rad/s to 100 rad/s. Similarly difference of phase Δ_p is according to (10.5) calculated as weighted absolute value of differences of desired phase function f_{pd} and phase of current solution f_{pc} .

$$\Delta_m = \frac{1}{m}\sum_{i=1}^{m} w_{dm}(i)|f_{md}(i) - f_{mc}(i)| \tag{10.4}$$

$$\Delta_p = \frac{1}{m}\sum_{i=1}^{m} w_{dp}(i)|f_{pd}(i) - f_{pc}(i)| \tag{10.5}$$

$$cost = \Delta_m w_{cm} + \Delta_p w_{cp} \tag{10.6}$$

Weights w_{cm} and w_{cp} were set to 1 and 2 respectively. Setting of weights w_{dm}, w_{dp} is presented in Tab. 10.2.

angular frequency range:	w_{dm}	w_{dp}
0.01 rad/s to 0.0398 rad/s	1.3	1.3
0.0437 rad/s to 20.8930 rad/s	1	1
22.9087 rad/s to 100 rad/s	1.3	1.3

Tab. 10.2: Setting of weights w_{dm} and w_{dp}.

Frequency responses of current solution f_{mc} and f_{pc} are obtained using nodal analysis method implemented in Matlab.

10.2.5 Settings of the Proposed Algorithm

The goal of the synthesis is to design a circuit which approximates function (10.1). The only informations supplied to the system are desired magnitude and phase characteristics (10.1) and maximal number of the used components n_c.

To allow direct comparison of accuracy of the original approximation circuit presented in [53] and approximation circuits obtained using the proposed method maximal number of the components in the synthesized circuit was set $n_c = 10$.

The synthesis method is configured to use only passive components resistors, capacitors and inductors. Negative values of the resistors are not allowed. Number of objective function evaluations (evals) required by the proposed algorithm consists of number of evaluations required for calculation of the cost values of all individuals of the population ($PopEvals$) and number of evaluations required by the used local search algorithm ($LSevals$). Population size was set $PopSize = 200$ and number of generation was set $MaxGen = 200$. Therefore $PopEvals = 40e3$. The local search method requires $MaxFunEvals = 100$ evaluations in each its run and it is executed with probability $P_{LSA} = 0.02$ (2% Lamarckian approach - see section 7.3). The condition of its execution is tested during each objective function evaluation. Therefore $LSevals = 80e3$ and total number of objective function evaluations required by the proposed algorithm is 120e3 ($PopEvals + LSevals$). Local search algorithm was realized using Matlab function $fmincon$.

Parameters of the synthesized problem and settings of the proposed algorithm are summarized in Tab. 10.3.

maximal number of nodes (n_n)	10 (0 to 9)
maximal number of components (n_c)	10
used types of components	R,L,C
negative resistors	not allowed
angular frequency range	0.01 rad/s to 100 rad/s
number of points (m)	101 (25 points/decade)
population size ($PopSize$)	200
number of generations ($MaxGen$)	200
probability of local search (P_{LSA})	0.02
w_{cm}	1
w_{cp}	2

Tab. 10.3: Setting of the parameters of the problem.

Algorithm UMDA is realized using Matlab toolbox MATEDA 2.0 (section 13.1). Settings of the algorithm are presented in Fig. 10.6

```
PopSize = 200; n = 135; cache  = [0,0,0,0,0];
Card = 2*ones(1,n); MaxGen = 200; MaxVal = -1e-2;
stop_cond_params = {MaxGen,MaxVal};
Cliques = CreateMarkovModel(n, 0);
edaparams{1} = {'seeding_pop_method','seedingFract',25};
edaparams{2} = {'learning_method','LearnFDA',{Cliques}};
edaparams{3} = {'sampling_method','SampleFDAmodif',{PopSize}};
edaparams{4} = {'stop_cond_method','maxgen_maxval',stop_cond_params};
[AllStat,Cache]=RunEDAfractal(PopSize, n, F, Card, cache, edaparams);
```

Fig. 10.6: Configuring of MATEDA 2.0 toolbox for realization of UMDA algorithm.

The whole program flow of UMDA algorithm is realized using functions of MATEDA 2.0 toolbox. The only exception is sampling phase of the algorithm which is modified to enable dealing with problems with unitation constraints (section 5.6). For more information on the modified sampling method please see section 7.2.

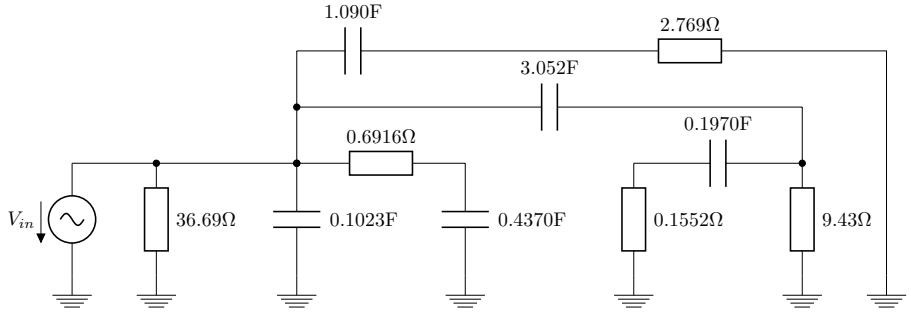
10.2.6 Experiments and Solutions

There were executed 20 instances of the proposed algorithm. Average running time of a single execution was 11 min. Cost values of the solutions are presented in Tab. 10.4.

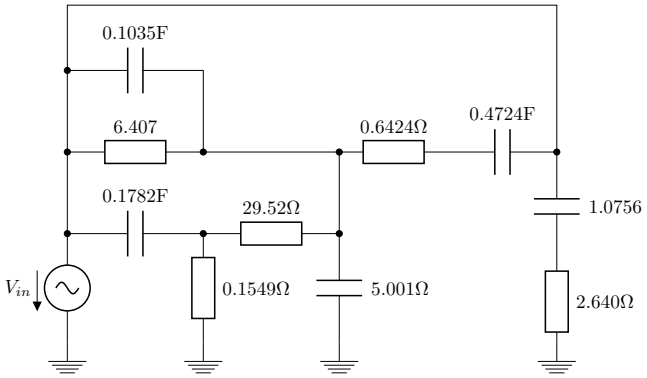
id of run	1	2	3	4	5	6	7	8	9	10
cost value	3.805	2.442	3.805	2.431	2.428	2.437	2.443	3.653	2.433	2.436
id of run	11	12	13	14	15	16	17	18	19	20
cost value	2.432	3.662	5.197	5.663	5.353	6.071	3.805	5.691	6.072	2.429

Tab. 10.4: Results of 20 runs of the proposed algorithm.

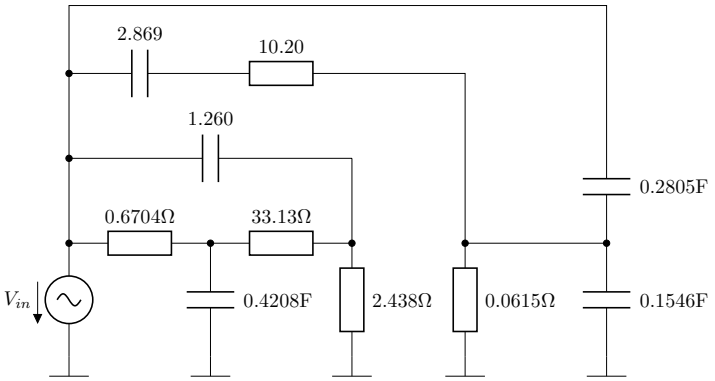
As can be seen in Tab. 10.4 the best solution was achieved in run 5 with cost value 2.428. Its schematic diagram is presented in Fig. 10.7a. Schematic diagrams of another three good solutions (run 4, run 11, run 20) are presented in Fig. 10.7b, Fig. 10.7c and Fig. 10.7d respectively.



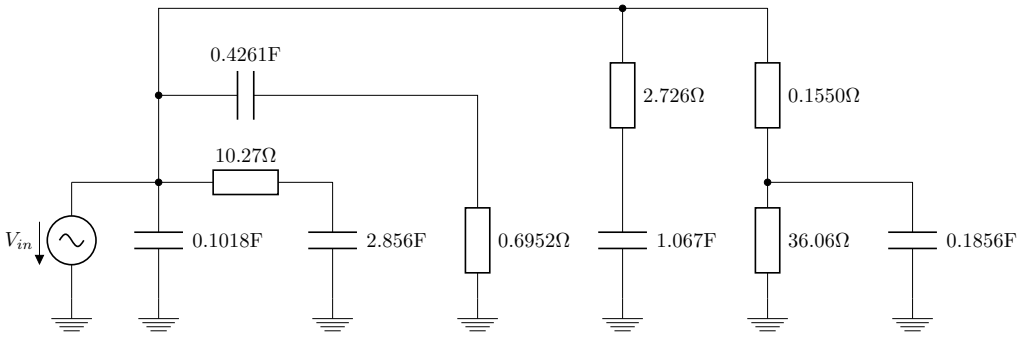
(a)



(b)



(c)



(d)

Fig. 10.7: Schematics of the approximation circuits synthesized in run 5 (a), run 4 (b), run 11 (c), run 20 (d).

Comparison of magnitude and phase characteristics of Z_{in} of the best found approximation circuit and desired function (10.1) are presented in Fig. 10.8a and Fig. 10.8c respectively. Absolute value of deviation of magnitude and phase of the best synthesized circuit are presented in Fig. 10.8b. and Fig. 10.8d respectively.

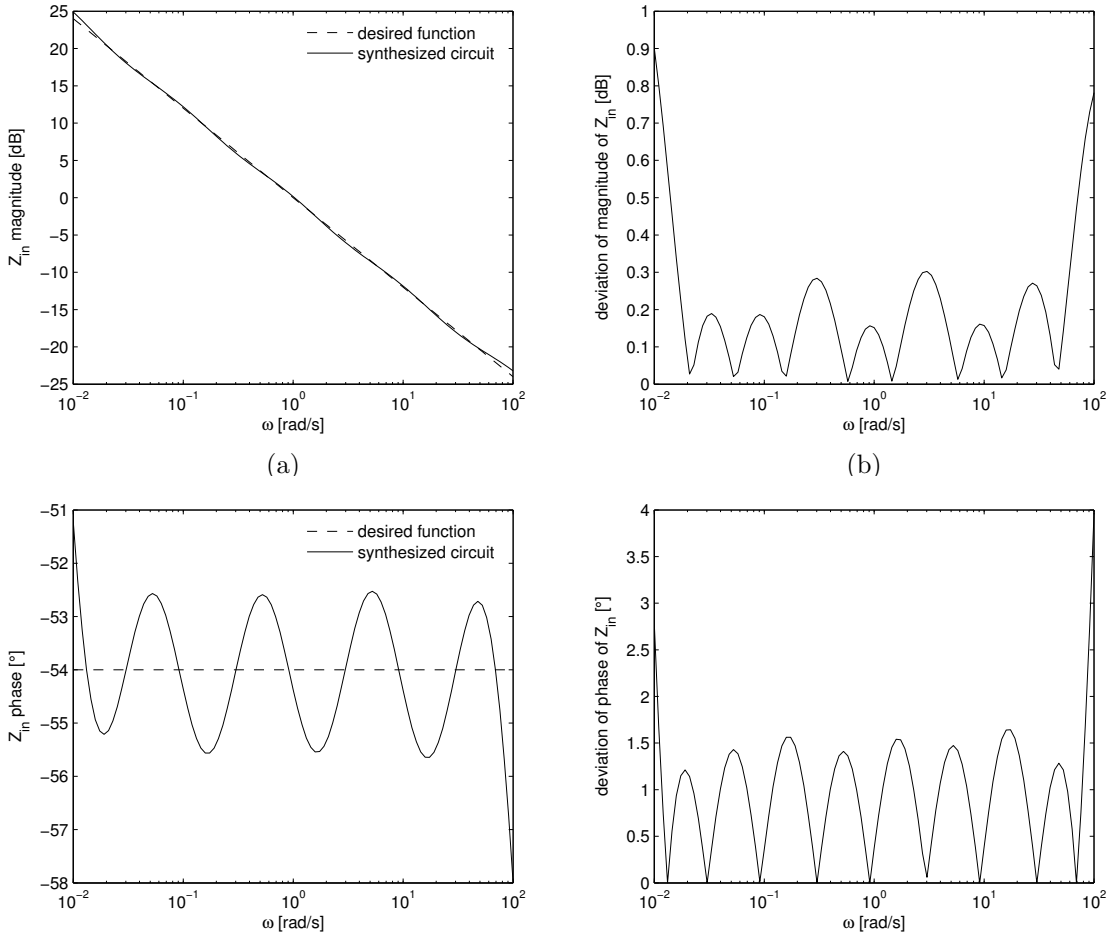


Fig. 10.8: a) Comparison of magnitude of Z_{in} of the best circuit and (10.1) b) Deviation of magnitude of Z_{in} of the best circuit c) Comparison of phase of Z_{in} of the best circuit and (10.1) d) Deviation of phase of Z_{in} of the best circuit.

Three highest deviations of magnitude and phase characteristics of Z_{in} of the best synthesized circuit are summarized in Tab. 10.5.

Magnitude			
$\omega[rad/s]$	0.01	100	3.02
$\Delta_m[dB]$	0.9	0.78	0.30

Phase			
$\omega[rad/s]$	0.01	100	17.38
$\Delta_p[\circ]$	2.78	3.97	1.64

Tab. 10.5: The highest deviations of magnitude and phase of the best synthesized approximation circuit.

As can be seen in Fig. 10.8a and Fig. 10.8c the maximal deviations of magnitude and phase responses are located at the boundaries of the used frequency range. At these frequencies unoptimized areas of the frequency response (0 to 10^{-2} rad/s and 10^2 to ∞ rad/s) affect the circuit's behavior in the area where the optimization was performed. Zeros and poles diagram of the synthesized circuit is presented in Fig. 10.9.

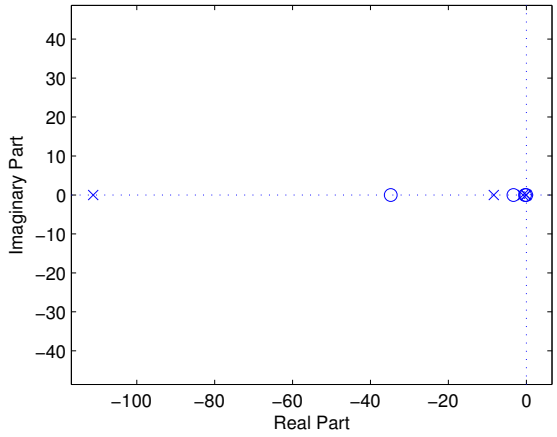


Fig. 10.9: Zeros and poles diagram of the best synthesized circuit.

Although the probability of using of all three component types (resistors, capacitors, inductors) was equal during the synthesis process, all synthesized circuits presented in Fig. 10.7a to Fig. 10.7d do not include any inductors. As the proposed algorithm was constrained to use only 10 components at the most, it seems that using only capacitors and resistors allows the method to reach lower cost values than in solutions where inductors are included.

10.2.7 Comparison of the Best Synthesized Solution and Original Approximation Circuit

In the section the best synthesized approximation circuit obtained using the proposed algorithm will be compared to original approximation circuit designed in [53] by classical method of analog circuits design.

Since the proposed evolutionary synthesis method was configured to use the same circuit complexity (10 components at the most) as original approximation circuit, accuracy of both circuits can be directly compared.

Except deviations at boundaries of the used frequency range (as was commented above) for the original approximation circuit the highest deviation of magnitude is $\Delta_m = 0.61$db at angular frequency 0.33 rad/s. For the solution of the proposed

method the highest deviation of the magnitude is $\Delta_m = 0.27$ dB at angular frequency 0.30 rad/s. Thus in terms of deviation of magnitude the accuracy of the synthesized circuit is more than twice better than the original approximation circuit. Comparison of the deviations of magnitude of Z_{in} for both circuits is presented in Tab. 10.6.

original	$\omega[rad/s]$	0.01	100	0.33
circuit	$\Delta_m[dB]$	0.71	0.46	0.61
synthesized	$\omega[rad/s]$	0.01	100	0.30
circuit	$\Delta_m[dB]$	0.93	0.92	0.27

Tab. 10.6: Comparison of deviations of magnitude of Z_{in} for original approximation circuit and for the best synthesized circuit.

The highest phase deviation inside the used frequency range is for original circuit $\Delta_p = 5.2^\circ$ at angular frequency 0.14 rad/s and for the synthesized circuit it is $\Delta_p = 1.5^\circ$ at angular frequency 0.58 rad/s. Thus phase accuracy of the synthesized circuit is more than three times better than the original approximation circuit. Comparison of maximal deviations of phase of Z_{in} is presented in Tab 10.7.

original	$\omega[rad/s]$	0.01	100	0.14
circuit	$\Delta_p[\circ]$	26.3	7.98	5.2
synthesized	$\omega[rad/s]$	0.01	100	0.58
circuit	$\Delta_p[\circ]$	3.0	4.2	1.5

Tab. 10.7: Comparison of deviations of phase of Z_{in} for original approximation circuit and for the best synthesized circuit.

10.3 Conclusion

Automated analog circuits synthesis method based on hybrid approach employing UMDA and a local search algorithm was presented. Used hybrid approach enables to employ specialized methods for both sub problems of different nature (synthesis of the topology and determination of the parameters). Synthesis of the topology which is combinational optimization problem was solved using UMDA. Synthesis of the parameters which is continuous optimization problem was solved using a local search algorithm. The principle of the method is based on mutual interaction of synthesis of the topology (UMDA) and determination of the parameters (local search algorithm) of the desired solution. Note that modified version of UMDA which allows to solve problems with unitation constrains was used.

The proposed method was verified on the problem of synthesis of fractional capacitor circuit introduced in [53]. Presented experiments have shown that the proposed algorithm is capable to synthesize solutions whose accuracy overperform solutions obtained using classical method of analog circuit design given in [53]. Accuracy of magnitude of the best obtained solution was more than twice better then the original approximation circuit. Accuracy of phase of the best obtained solution was more than three times better then the original approximation circuit.

In [4] the problem of fractional capacitor analog circuit realization was solved using simulated annealing method (SA). Accuracy of the circuits synthesized using SA was comparable to the solutions produced using the proposed EDA method. However SA required much higher number of objective function evaluations. While the proposed EDA method required 120e3 objective function evaluations for synthesis of the same problem SA required 440e3 objective function evaluations.

11 SYNTHESIS OF CUBE ROOT FUNCTION USING GRAPH EDA METHOD

11.1 Definition of the Problem

The problem of synthesis of circuit realization of cube root function was proposed by John Koza as a benchmark problem for evolutionary analog electronics synthesis systems in his paper discussing synthesis of analog circuits using genetic programming [57]. The problem was also adopted in [17] where unconstrained optimization genetic algorithm with oscillating length representation was used. The target voltage response of the desired circuit is (11.1).

$$U_2 = \sqrt[3]{U_1} \tag{11.1}$$

In other words the goal of the synthesis is to design analog circuit in which output voltage U_2 is cube root of its input voltage U_1. Graphical representation of (11.1) is presented in Fig. 11.1.

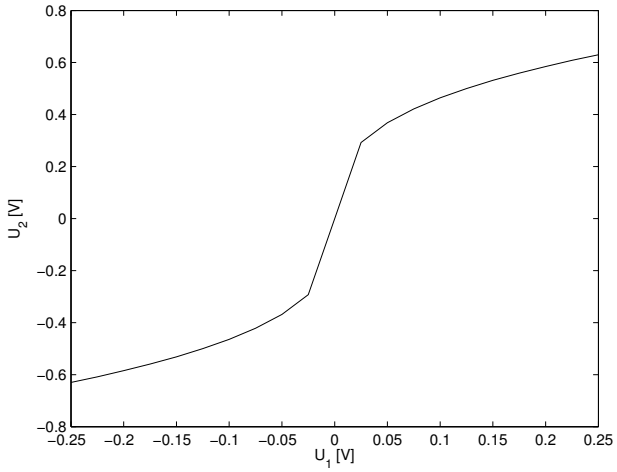


Fig. 11.1: Target response of the desired circuit realization of cube root function.

To enable direct comparison of the results obtained using the proposed method and the results of other authors the objective function in the proposed method is defined exactly the same way as was presented in original paper [57].

11.2 Introduction of Graph Estimation of Distribution Algorithm

Synthesis capability of the proposed GhEDA method will be demonstrated on problem of circuit realization of cube root function. The cube root function circuit realization consists of bipolar transistors NPN and PNP, resistors and positive and negative voltage sources. The goal of the synthesis is to design topology of connection of the transistors NPN and PNP, topology of connection of the resistors, the parameters of the resistors (the values of the resistors) and define nodes of connection of the positive and negative voltage sources. Pseudo-code of the proposed method is presented in Fig. 11.2.

step0: Initialize population P of n_i individuals.
step1: According to selection method select population P_s.
step2: Build probabilistic model M of selected population P_s.
step3: Using probabilistic model M generate set of new samples P_g consisting of n_g individuals.
step4: Using objective function evaluate cost values of set of samples P_g.
step5: Based on P and P_g create new population P_n and replace old population ($P := P_n$).
step6: According to the topologies of n_{opt} randomly selected individuals of P optimize values of parameters storage e_{ps}. Go to **step1**.

Fig. 11.2: Pseudo-code of the proposed method.

Initial population P consisting of n_i individuals is set randomly respecting maximal number of components of every type n_{npn}, n_{pnp}, n_{res}, n_{vccp}, n_{vccn}. Parameters storage e_{ps} is initialized randomly with uniform distribution. Detailed description of the encoding method and parameters storage e_{ps} is presented in section 11.3.

After evaluation of the cost values of population P, selected population P_s is formed. Tournament selection method with tournament size 2 is used.

In the learning phase probabilistic model M of selected population P_s is created. Marginal frequencies of the components included in selected population P_s are calculated. Every single component connected to specific set of connection nodes is represented by corresponding edge of the graph (resistors and positive and negative voltage sources) or hyperedge of the hypergraph (transistors NPN and PNP). Therefore the marginal frequencies of the components correspond to the marginal frequencies of the edges of the graphs and the hyperedges of the hypergraphs encoded in

the individuals of selected population P_s. Detailed description of the learning phase is presented in section 11.4.

In the next phase probabilistic model M is used to generate population of new samples of solutions P_g which consists of n_g individuals. Detailed description of the sampling phase is described in section 11.5.

New individuals are simulated and theirs cost values are calculated using objective function described in section 11.6.

In the replacement phase new population P_n is formed of best $n_i - n_g$ individuals of current population P and whole population of new samples P_g. Afterwards current population P is replaced by new population P_n ($P := P_n$).

In the optimization phase the local search algorithm tries to improve (decrease) cost values of n_{opt} randomly selected individuals of population P. Detailed description of the optimization phase is presented in section 11.7.

11.3 Encoding Method

Graphs are the most straightforward method of representation of the topology of analog circuits. The desired circuit realization of cube root function consists of resistors, bipolar transistors NPN and PNP and positive and negative voltage sources. As will be described bellow topology of connection of the resistors and connection of the positive and the negative voltage sources are represented by corresponding graphs. Topology of connection of the transistors NPN and PNP are represented by 3-uniform hypergraphs.

Maximal complexity of the desired analog circuit is defined by maximal number of nodes n_n and maximal number of transistors NPN, transistors PNP and resistors denoted as n_{npn}, n_{pnp}, n_{res}. Maximal number of nodes connected to the positive and negative voltage sources are denoted as n_{vccp} and n_{vccn} respectively. Every individual of population P consist of informations about topology of connection of the transistors NPN and PNP, topology of connection of the resistors and connection of the positive and negative voltage sources. Parameters of the resistors are stored in parameters storage e_{ps} which is described in section 11.7.

The topology of connection of the resistors is represented by simple undirected graph G_{res}. Since maximal complexity of the synthesized circuit is restricted to n_n nodes graph G_{res} is always subgraph of complete graph G_{resc} which includes n_n vertices and $n_{edgres} = n_n\,(n_n - 1)/2$ edges. Complete graph G_{resc} for $n_n = 4$ and corresponding topology of connection of the resistors are presented in Fig. 11.3a and Fig. 11.3b respectively. Example of graph G_{res} and corresponding topology of connection of the resistors are presented in Fig. 11.3c and Fig. 11.3d.

Graph G_{res} is defined by its characteristic vector. Maximal number of edges of graph G_{res} is given by number of edges n_{edgres} of corresponding complete graph G_{resc}. Characteristic vector of graph G_{res} can be defined as binary vector e_{res} of length n_{edgres} bits. Every single bit of e_{res} corresponds to including or not including corresponding edge of complete graph G_{resc} in its subgraph G_{res}. Characteristic vector e_{res} of graph G_{res} in Fig. 11.3c is presented in Fig. 11.3d.

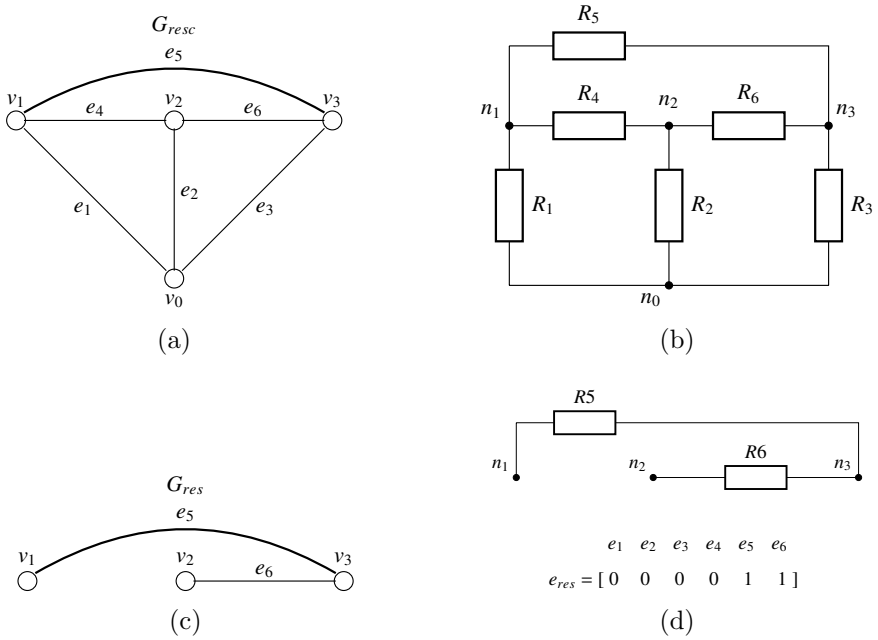


Fig. 11.3: a) Complete graph G_{resc} for $n_n = 4$ b) Analog circuit corresponding to G_{resc} c) Subgraph G_{res} d) Analog circuit corresponding to G_{res} and its encoding vector e_{res}.

Assignment of the edges to the vertices for graphs G_{res} and G_{resc} and assignment of the resistors to the nodes for the corresponding circuits (Fig. 11.3b and Fig. 11.3d) are presented in Tab. 11.1.

edge (resistor)	e_1 (R_1)	e_2 (R_2)	e_3 (R_3)	e_4 (R_4)	e_5 (R_5)	e_6 (R_6)
vertex 1 (node 1)	v_0 (n_0)	v_0 (n_0)	v_0 (n_0)	v_1 (n_1)	v_1 (n_1)	v_2 (n_2)
vertex 2 (node 2)	v_1 (n_1)	v_2 (n_2)	v_3 (n_3)	v_2 (n_2)	v_3 (n_3)	v_3 (n_3)

Tab. 11.1: Assignment of the edges to the vertices for graphs G_{resc} and G_{res} and assignment of the resistors to the nodes for circuits in Fig. 11.3b and Fig. 11.3d.

Topology of connection of the transistors NPN is represented by labeled 3-uniform hypergraph G_{npn} and is restricted to n_n nodes. Similarly to the repre-

sentation of the topology of the resistors there can be defined complete 3-uniform hypergraph G_{npnc} which includes n_n vertices and $n_{edgnpn} = n_n(n_n - 1)(n_n - 2)/6$ hyperedges. Hypergraph G_{npn} is always subhypergraph of complete hypergraph G_{npnc}.

Compared to the representation of the topology of connection of the resistors, representation of the topology of connection of the transistors requires another additional parameter "rotation" of the transistors. While connection nodes of every single encoded transistor are defined by the connection vertices of the corresponding hyperedge of hypergraph G_{npn}, "rotation" of the transistor is defined by the label of the corresponding hyperedge. Given 3 ports transistor there are six possible combinations ("rotations") of the assignment of the nodes to the ports of the transistor. Complete 3-uniform hypergraph G_{npnc} for $n_n = 4$ is presented in Fig. 11.4a. Larger white circles represent vertices of the hypergraph. Smaller black circles represent hyperedges. Corresponding analog circuit is presented in Fig. 11.4b.

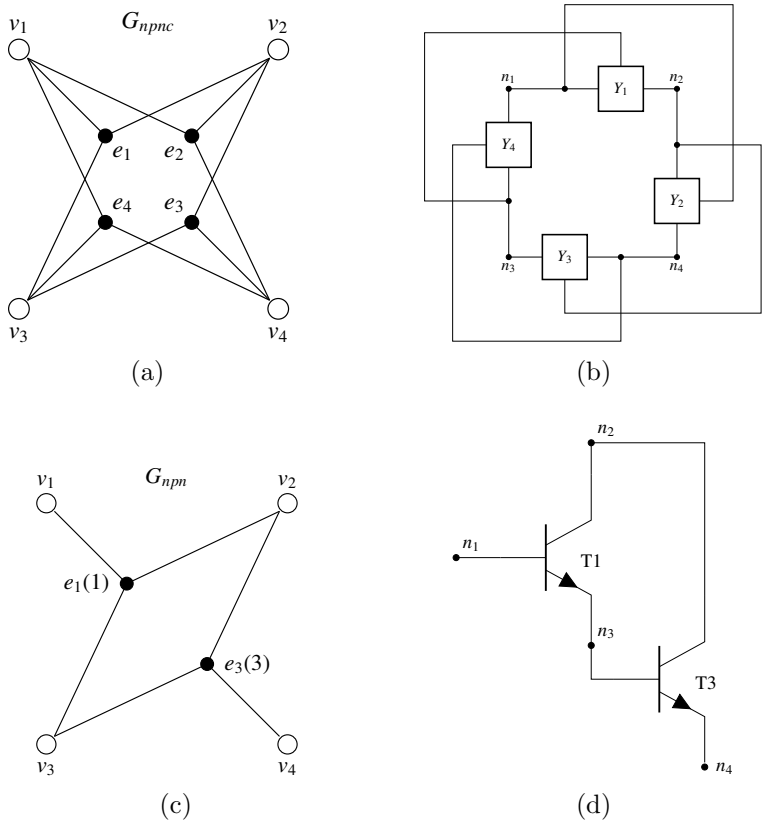


Fig. 11.4: a) Complete 3-uniform hypergraph G_{npnc} for $n_n = 4$ b) Analog circuit representation of G_{npnc} c) Example of 3-uniform hypergraph G_{npn} d) Analog circuit representation of G_{npn}.

Assignment of the hyperedges to the vertices for hypergraphs G_{npnc} and G_{npn} and assignment of the 3 ports generalized admittances (Y_1 to Y_4) to the nodes for corresponding circuits (Fig. 11.4b and Fig. 11.4d) are presented in Tab. 11.2.

hyperedge (generalized 3-ports admittance)	e_1 (Y_1)	e_2 (Y_2)	e_3 (Y_3)	e_4 (Y_4)
vertex 1 (node 1)	v_1 (n_1)	v_1 (n_1)	v_2 (n_2)	v_1 (n_1)
vertex 2 (node 2)	v_2 (n_2)	v_2 (n_2)	v_3 (n_3)	v_3 (n_3)
vertex 3 (node 3)	v_3 (n_3)	v_4 (n_4)	v_4 (n_4)	v_4 (n_4)

Tab. 11.2: Assignment of the hyperedges to the vertices for hypergraphs G_{npnc} and G_{npn} and assignment of the 3 ports generalized admittances (Y_1 to Y_4) to the nodes for circuits in Fig. 11.4b and Fig. 11.4d.

Since "rotation" labels are not specified in complete 3-uniform hypergraph G_{npnc}, generalized three-ports admittances Y_1 to Y_4 are used in Fig. 11.4b. An example of labeled 3-uniform hypergraph G_{npn} is presented in Fig. 11.4c. The numbers in the brackets behind the names of the hyperedges define the labels of the hyperedges. For hyperedges e_1 and e_3 of hypergraph G_{npn} the labels "rotation" are set to 1 and 3 respectively. Assignment of the labels of the hyperedges to "rotations" of the transistors is defined in Tab. 11.3.

label of hyperedge ("rotation")	1	2	3	4	5	6
node 1	B	B	C	C	E	E
node 2	C	E	B	E	B	C
node 3	E	C	E	B	C	B

Tab. 11.3: Assignment of the labels of the hyperedges to the connection nodes of the transistors.

Since every possible configuration of hypergraph G_{npn} is subhypergraph of complete 3-uniform hypergraph G_{npnc}, characteristic vector of hypergraph G_{npn} can be defined as binary vector e_{npn} of length n_{edgnpn} bits. Every single bit of e_{npn} corresponds to including or not including corresponding hyperedge of complete hypergraph G_{npnc} in its subhypergraph G_{npn}. Encoding vector e_{npn} is further extended to include information about "rotation" of the encoded transistors. There are six possible combinations of connection of the ports of a transistor to three nodes. Therefore the final encoding vector e_{npn} is defined as binary vector of length $6\,n_{edgnpn}$. Encoding vector e_{npn} of hypergraph G_{npn} presented in Fig. 11.4c is presented in Fig. 11.5.

	Y_1						Y_2						Y_3						Y_4					
rotation:	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
e_{npn} = [	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0]

Fig. 11.5: Encoding vector e_{npn} of hypergraph G_{npn}.

Topology of connection of the PNP transistors is represented by labeled 3-uniform hypergraph G_{pnp} and encoded by vector e_{pnp} exactly the same way as was described above for the topology of connection of the NPN transistors.

The last type of information which has to be encoded is connection of the positive and the negative voltage sources what is represented by graphs G_{vccp} and G_{vccn} respectively.

As can be seen in example in Fig. 11.6a graph G_{vccp} includes vertex V_{vccp} which represents positive voltage source. Edges between vertices V_{vccp} and v_1 and v_3 represent connection of positive voltage source V_{ccp} to nodes n_1 and n_3.

Similarly in graph G_{vccn} vertex V_{vccn} is connected to vertices v_2 and v_4 what corresponds to connection negative voltage source V_{ccn} to nodes n_2 and n_4 (Fig. 11.6a). Schematic representation of connection of the positive and the negative voltage sources corresponding to graphs in Fig. 11.6a is presented in Fig. 11.6b.

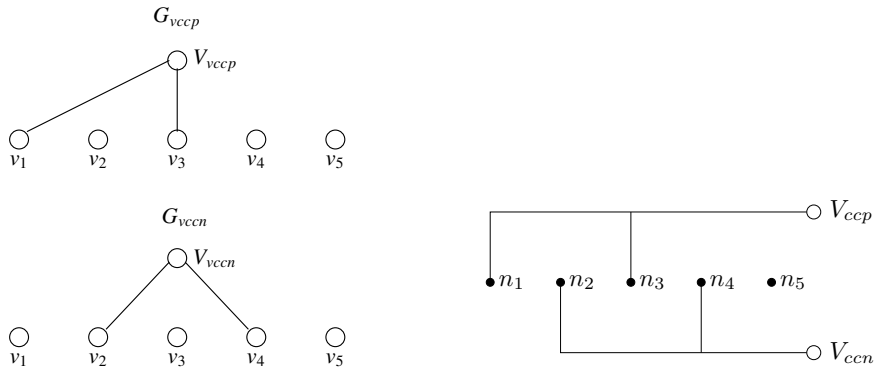


Fig. 11.6: a) Graphs G_{vccp} and G_{vccn} b) Connection of the voltage sources defined by graphs G_{vccp} and G_{vccn}.

Encoding vectors e_{vccp} and e_{vccn} of graphs G_{vccp} and G_{vccn} are represented by binary vectors of length n_n, where every single bit represents including or not including of an edge between voltage source (V_{vccp} or V_{vccn}) and corresponding vertex (v_1 to v_5 in the example). Encoding vectors e_{vccp} and e_{vccn} are presented in Fig. 11.7.

Parameters of the resistors (values of the resistors) are stored in parameters storage e_{ps} which is vector of real numbers of length n_{edgres}. Vector e_{ps} includes a value

$$e_{vccp} = \begin{array}{c} \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\ [\,1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0\,] \end{array} \qquad e_{vccn} = \begin{array}{c} \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\ [\,0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0\,] \end{array}$$

Fig. 11.7: Encoding vector e_{vccp} of graph G_{vccp} and encoding vector e_{vccn} of graph G_{vccn}.

for every possible resistor connected to nodes n_1 and n_2, where $n_1 \in \{0, 1, \ldots, n_n - 1\}$ and $n_2 \in \{0, 1, \ldots, n_n - 1\}$.

During every single evaluation of the objective function the cost value is obtained based on two types of information. The first one informs about the topology and is stored in encoding vector of individual (e_{res}, e_{npn}, e_{pnp}, e_{vccp}, e_{vccn}) in population P. The second one informs about the parameters of the encoded resistors and is stored in parameters storage e_{ps}.

The only way how to modify the values of parameters storage e_{ps} is execution of the local search algorithm (LSA) in optimization phase (**step6** in Fig. 11.2). The synthesis process consists of mutual interaction between selection of the promising topologies (**step1** in Fig. 11.2) and optimization of the values of parameters storage e_{ps}. In the optimization phase LSA tries to optimize the values of e_{ps} to adapt them to the promising topologies selected in the selection phase. This way the values stored in parameters storage e_{ps} are evolved during the whole synthesis process.

11.4 Learning of the Probabilistic Model

For every single component type (transistors NPN, transistors PNP, resistors, positive voltage sources, negative voltage sources) marginal frequencies of the edges contained in current selected population P_s are calculated and saved in vectors v_{npn}, v_{pnp}, v_{res}, v_{vccp}, v_{vccn} which are encoded the same way as encoding vectors e_{npn}, e_{pnp}, e_{res}, e_{vccp}, e_{vccn}. Values of vectors v_{npn}, v_{pnp}, v_{res}, v_{vccp}, v_{vccn} represent numbers of appearing of the corresponding edges in current selected population P_s. Examples of vectors v_{npn}, v_{pnp}, v_{res}, v_{vccp}, v_{vccn} are presented in Fig. 11.8a to Fig. 11.8d. For example $v_{npn}(3) = 4$ (number 4 in the third position of vector v_{npn}) denotes that current selected population P_s includes four individuals with $e_{npn}(3) = 1$. This corresponds to the fact that transistor NPN with C connected to n_1, B connected to n_2 and E connected to n_3 (see Tab. 11.3) was used four times in the current selected population P_s. Similarly $v_{res}(2) = 9$ denotes that the resistor connected to nodes 0 and 2 (see Tab. 11.1) was used nine times in P_s and it becomes the most frequently used resistor in the individuals of current selected population P_s. In other words there is high probability that this resistor will appear in topology of good individual.

	Y_1						Y_2						Y_3						Y_4					
rotation:	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
v_{npn} = [	0	0	4	0	0	0	3	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0]
v_{pnp} = [	0	2	0	0	0	0	0	0	0	0	0	0	1	0	4	0	0	0	0	0	0	0	0	0]

(a)

	R_1	R_2	R_3	R_4	R_5	R_6
v_{res} = [	2	9	5	4	7	3]

(b)

	n_1	n_2	n_3	n_4	n_5	n_6
v_{vccp} = [	4	1	2	7	2	1]

(c)

	n_1	n_2	n_3	n_4	n_5	n_6
v_{vccn} = [	3	6	2	5	2	1]

(d)

Fig. 11.8: Examples of vectors v_{npn} and v_{pnp}(a), v_{res} (b), v_{vccp} (c) and v_{vccn} (d).

After calculation of the marginal frequencies of the edges for all types of components, values of vectors v_{npn}, v_{pnp}, v_{res}, v_{vccp}, v_{vccn} are sorted from the highest to the lowest and this way vectors s_{npn}, s_{pnp}, s_{res}, s_{vccp}, s_{vccn} are obtained. Vectors of sorted marginal frequencies s_{npn}, s_{pnp}, s_{res}, s_{vccp}, s_{vccn} are used for determination of the most probable components during the phase of generation of new individuals (sampling phase). Sorted information about the marginal frequencies of the used components in current population P_s stored in five vectors s_{npn}, s_{pnp}, s_{res}, s_{vccp}, s_{vccn} is denoted as probabilistic model M.

11.5 Sampling of the Probabilistic Model

Created probabilistic model M is used to generate new solutions of the promising topologies of the given solution space. To increase diversity of the created samples some portion of the edges of the generated samples is created randomly. Presented sampling method was inspired by sampling principle of Estimation of Distribution Algorithm based on graphs kernels presented in [58]. Pseudo-code of the used sampling method is presented in Fig. 11.9.

step1: Randomly select individual I of population P_s.
step2: Randomly with probability P_{rem} remove edges of graphs G_{res}, G_{vccp}, G_{vccn} and hyperedges of hypergraphs G_{npn}, G_{pnp} of individual I.
step3: Add edges to graphs G_{res}, G_{vccp}, G_{vccn} and hyperedges to hypergraphs G_{npn}, G_{pnp} of selected individual I.

Fig. 11.9: Flow chart of the sampling phase.

In **step1** individual I of current selected population P_s is chosen randomly and is used as a basis for new generated sample.

In **step2** edges of graphs G_{res}, G_{vccp}, G_{vccn} and hyperedges of hypergraphs G_{npn}, G_{pnp} of individual I are removed randomly with probability P_{rem} which is typically set to 0.2. In other words approximately $100.P_{rem}$ percent of the edges of graphs G_{res}, G_{vccp}, G_{vccn} and the hyperedges of hypergraphs G_{npn}, G_{pnp} of individual I are removed.

In **step3** new edges are added to graphs G_{res}, G_{vccp}, G_{vccn} and new hyperedges are added to hypergraphs G_{npn}, G_{pnp}. There are two ways how to perform this step. In the first one the process of the addition of the edges and the hyperedges is guided using information about the promising areas of the solution space stored in probabilistic model M. The edges and the hyperedges with high values of the marginal frequencies in vectors s_{npn}, s_{pnp}, s_{res}, s_{vccp}, s_{vccn} are more favorable than those with lower values. This way modification of the topologies of graphs G_{res}, G_{vccp}, G_{vccn} and hypergraphs G_{npn}, G_{pnp} is guided to include edges which are frequently used in good individuals of the population. The second way is random addition of the edges and the hyperedges what helps to maintain diversity of the generated samples. Probability of using of probabilistic model M to guide the process of addition of the edges and the hyperedges is defined as P_{add} and is typically set to 0.8.

11.6 Objective Function

Information about the topology stored in the individuals of population P_g and information about the parameters stored in parameters storage e_{ps} are transformed into netlist representation suitable for external spice compatible circuit simulator.

After obtaining of voltage frequency characteristic, cost value is calculated using objective function (11.2). To enable direct comparison of the results obtained using the proposed method to the results of other authors the objective function is defined exactly the same way as was presented in original paper [57].

$$cost = \sum_{i=1}^{m} w_v(i)|f_{vd}(i) - f_{vc}(i)| \tag{11.2}$$

According to (11.2) cost value *cost* is defined as weighted summation of absolute values of differences between voltage response of desired solution f_{vd} and voltage response of current solution f_{vc} over $m = 21$ equidistant voltage values in range -250 mV to 250mV. There is penalization of the cost value by 10 if the output voltage response is not within 1% deviation of the target voltage characteristic. In such case weight w_v is set to 10, otherwise $w_v = 1$. Voltage response of the current solution is obtained using circuit simulator ngspice (section 13.5).

11.7 Parameters Optimization

In the last phase of the proposed method the parameters of the resistors stored in parameters storage e_{ps} are optimized according to the topologies of n_{opt} randomly selected individuals of newly created population P. In every generation of the proposed method the parameters optimization routine is executed with probability P_{LSA}. Pseudo-code of the parameters optimization phase is presented in Fig. 11.10.

step1: Randomly choose individual I of population P.
step2: Based on topology of individual I load parameters p_1 from parameters storage e_{ps}.
step3: Using topology information stored in I and parameters p_1 evaluate cost value of individual I.
step4: Execute local search algorithm. Optimized parameters p_2 and cost value of optimized solution c_2 are obtained.
step5: If $c_2 < c_1$ then replace parameters p_1 in e_{ps} with parameters p_2.

Fig. 11.10: Pseudo code of the optimization phase.

Individual I of current population P is selected randomly (**step1**). Based on the topology encoded in individual I corresponding parameters p_1 of parameters storage e_{ps} are loaded and cost value c_1 of individual I is evaluated (**step2**, **step3**).

In **step4** the local search algorithm (LSA) tries to improve accuracy of individual I. Parameters p_1 loaded in **step2** are used as a starting point for LSA. After finishing LSA new optimized parameters p_2 and cost value of the optimized individual c_2 are obtained.

If LSA was successful in improving of cost value of individual I ($c_2 < c_1$) then parameters p_1 in parameters storage e_{ps} are replaced by optimized parameters p_2. If LSA was not successful in improving accuracy of individual I then parameters optimization process is terminated with no modification of parameters storage e_{ps} (**step5**).

As LSA Matlab function fmincon was used. The function was configured to use Interior-Point algorithm and maximal number of function evaluations $MaxFunEvals$ was set to 800.

According to [17] values of the resistors are chosen from E12 series in five decades. Thus resistance of every resistor can be set to one of 60 possible values. The lowest and the highest possible values of resistors were 10Ω and 820kΩ respectively.

11.8 Experiments and Solutions

The proposed algorithm was implemented in 64-bit version of Matlab 8.0 (R2012b). Experiments were performed on 64-bit dual core PC with processor AMD Athlon II X2 245, 8GB RAM and operational system Centos 6.5. Parameters of the algorithm used in the experiments are summarized in Tab. 11.4.

Number of total function evaluations n_{evals} consists of number of function evaluations required by evaluation of cost values of P_g (**step4** in Fig. 11.2) and number of function evaluations required by parameters optimization phase (**step6** in Fig. 11.2) and can be computed as $n_{evals} = n_{gen}\, d + n_{gen}\, P_{LSA}\, n_{opt}\, MaxFunEvals$.

maximal number of nodes (n_n)	17
used types of components	R, NPN, PNP, Vccp, Vccn
maximal number of resistors (n_{res})	12
maximal number of transistors NPN (n_{npn})	14
maximal number of transistors PNP (n_{pnp})	14
maximal number of nodes connected to Vccp (n_{vccp})	6
maximal number of nodes connected to Vccn (n_{vccn})	6
negative resistors	not allowed
size of P (n_i)	400
size of P_g (n_g)	200
generations per run (n_{gen})	3000
number of objective function evaluations (n_{evals})	1.5e6
probability of parameters optimization (P_{LSA})	0.15
number of optimized individuals (n_{opt})	4
number of function evaluations of LSA	500

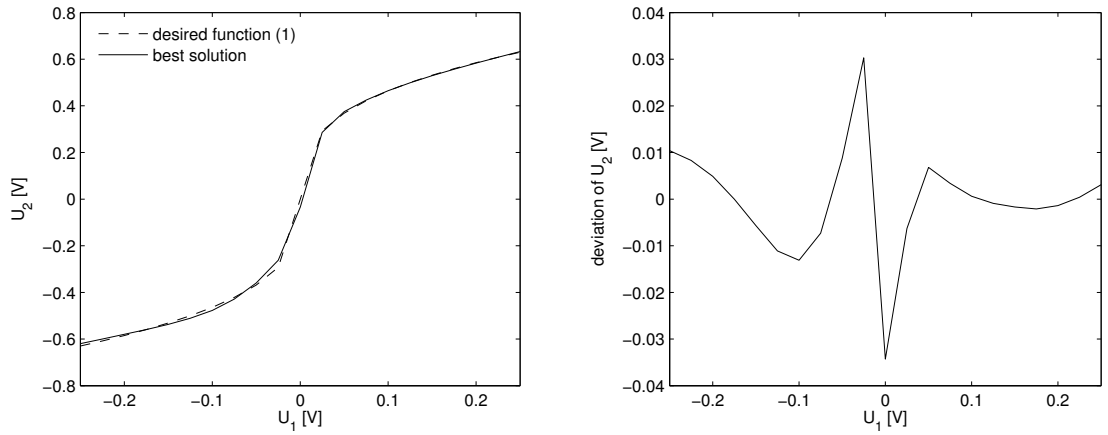
Tab. 11.4: Summary of the parameters of the proposed algorithm.

The proposed algorithm was executed in four parallel threads. Five runs per single thread. Therefore 20 runs of the proposed algorithm in total. Average run time of a single run was 14 hours. Average time of a single evaluation of the objective function was 0.0336 second. Results of the runs are presented in Tab. 11.5.

Tread 1					
id of run	1	2	3	4	5
cost value	6.88	5.42	20.6	44.2	4.05
Tread 2					
id of run	1	2	3	4	5
cost value	47.1	21.7	35.4	6.53	4.99
Tread 3					
id of run	1	2	3	4	5
cost value	6.15	7.65	1.44	5.67	1.91
Tread 4					
id of run	1	2	3	4	5
cost value	3.92	8.90	85.5	26.3	8.78

Tab. 11.5: Results of 20 runs of the proposed algorithm.

The best solution was found in run 3 of thread 3. Comparison of the output characteristics of the best solution and desired function (11.1) is presented in Fig. 11.11a. Since both curves in Fig. 11.11 are almost merged together, deviation of U_2 is presented in Fig. 11.11b.



(a) (b)

Fig. 11.11: a) Comparison of the output voltage characteristic $U_2 = f(U_1)$ of the best solution and desired function (11.1) b) deviation of the output voltage characteristic $U_2 = f(U_1)$ of the best solution and function (11.1).

Netlist of the best solution obtained in the proposed experiments is presented in Fig. 11.12. Bipolar transistors NPN and PNP are denoted as bjtnpn and bjtpnp respectively. Default models were used for both types of the transistors. To reduce convergence problems caused by unconnected components and dangling terminals all nodes of the encoded analog circuit are connected to GND (node 0) through resistance 1GΩ (resistors Rg1 to Rg16). Resistors R_{in} and R_L are input and output resistances respectively and are set to $1k\Omega$. Schematic corresponding to the netlist of the best evolved solution in Fig. 11.12 is presented in Fig. 11.13. Since transistors q1 and q11 have no function in the synthesized circuit (netlist in Fig. 11.12) these transistors were not used in the resulting schematic (Fig. 11.13). Voltage V_{IN} and voltage on resistor R_L are input and output voltages respectively (Fig. 11.13).

```
R1 1 15 4.7e+05
R2 2 6 5.6e+02
R3 4 6 1.5e+04
R4 6 11 6.8e+02
R5 8 11 2.7e+04
R6 10 16 4.7e+03
q1 0 5 4 bjtnpn
q2 3 11 1 bjtnpn
q3 2 14 12 bjtnpn
q4 15 3 7 bjtnpn
q5 4 14 9 bjtnpn
q6 6 14 12 bjtnpn
q7 8 0 2 bjtpnp
q8 7 4 0 bjtpnp
q9 11 0 8 bjtpnp
q10 11 8 0 bjtpnp
q11 13 9 0 bjtpnp
q12 9 16 0 bjtpnp
q13 6 1 14 bjtpnp
q14 4 10 9 bjtpnp
q15 16 6 8 bjtpnp
q16 14 15 8 bjtpnp
Rn1  12 nvccn 1e-3
Rp1  10 nvccp 1e-3
Rg1 1 0 1e9
Rg2 2 0 1e9
Rg3 3 0 1e9
Rg4 4 0 1e9
Rg5 5 0 1e9
Rg6 6 0 1e9
Rg7 7 0 1e9
Rg8 8 0 1e9
Rg9 9 0 1e9
Rg10 10 0 1e9
Rg11 11 0 1e9
Rg12 12 0 1e9
Rg13 13 0 1e9
Rg14 14 0 1e9
Rg15 15 0 1e9
Rg16 16 0 1e9
Rin x1 1 1e3
RL 2 0 1e3
.options TRTOL=7
.model bjtnpn npn
.model bjtpnp pnp
vdcp nvccp 0 dc 10
vdcn 0 nvccn dc 10
vin x1 0 dc 0 ac 1
.dc vin -0.25 0.25 0.025
.save v(2)
.end
```

Fig. 11.12: Netlist of the best solution.

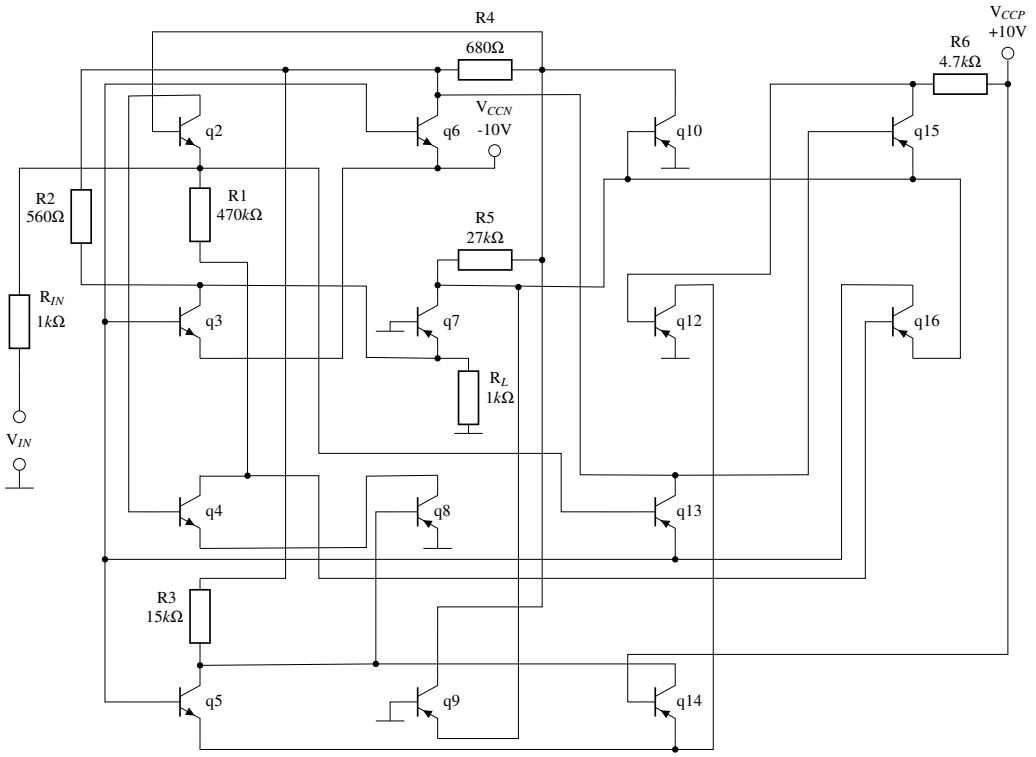


Fig. 11.13: Schematic of the best solution.

11.9 Comparison to Other Methods

As was stated above the problem of circuit realization of cube root function which was introduced by Koza et al. in [57] was adopted also by Sapargaliyev et al. in [17]. In [57] Koza et al. employed genetic programming (GP) approach. In [17] unconstrained genetic algorithm with oscillating length representation (GA OLG) was used. Comparison of the best solutions of both mentioned authors and the best solution of the proposed method GhEDA is presented in Tab. 11.6.

method	best cost	objective function evaluations
GP (Koza et al. [57])	1.68	37e6
GA OLG (Sapargaliyev et al. in [17])	2.27	4e6
GhEDA	1.44	1.5e6

Tab. 11.6: Comparison of the results of GhEDA to GP and GA OLG.

As can be seen in Tab. 11.6 proposed method GhEDA overperforms other two methods in terms of accuracy of the best solution and number of required objective function evaluations as well. Comparison of the number of the components of the best synthesized circuits of methods GP, GA OLG and GhEDA is presented in Tab. 11.7.

method	GP	GA OLG	GhEDA
number of transistors	36	24	14
number of resistors	12	12	7
number of diodes	2	2	0

Tab. 11.7: Comparison of the number of the components of the best solutions of methods GP, GA OLG and GhEDA.

As can be seen from Tab. 11.7 the complexity of the synthesized circuit was highest for GP. Method GA OLG was able to reach circuit of lower complexity compared to GP. The best result was achieved using GhEDA method which was able to synthesize circuit twice smaller than circuit produced using GP.

11.10 Conclusion

There was presented graph based hybrid estimation of distribution algorithm (GhEDA). Its synthesis capability was demonstrated on problem of circuit realization of cube root function. Results of the proposed method were compared to results of Koza et al. (GP) [57] and Sapargaliyev et al. (GA OLG) [17] who adopted the same problem of cube root function circuit realization. Experiments have shown that in terms of accuracy of the solution and number of required objective function evaluations the proposed method overperforms both other methods.

The proposed method employs simple univariate probabilistic model based on the assumption that there are no dependencies between the variables of the solution vector. Although the presented experiments have shown that the used probabilistic model was suitable for the proposed method univariate probabilistic model can be replaced by more advanced multivariate probabilistic model which is capable to capture higher order dependencies between the variables of the solution vector. This might be interesting and promising area of another research. Since some multivariate models can incorporate some portion of previous knowledge (prior) another interesting area of the future research might be usage of different priors based on the target application of the synthesized circuit.

Since the proposed method is population based evolutionary algorithm, multiobjective approach as pareto ranking can be incorporated into the method. Also parallel computation of the cost values of the individuals of the population can be applied.

12 SUMMARY AND FUTURE DIRECTIONS

In many areas of evolutionary computation Estimation of Distribution Algorithms (EDA) deliver great performance and overperform classical genetic algorithms. Also in the area of automated analog circuit synthesis methods EDA algorithms produce very good results.

Three different methods of automated analog circuit synthesis based on EDA algorithms were proposed, described and verified. As was stated in section 2.3 the two tasks of analog circuit synthesis problem - synthesis of the topology and determination of the parameters can be solved separately or simultaneously. While synthesis of the topology is combinational optimization problem determination of the parameters is continuous optimization problem in nature.

Simultaneous solving of the problem brings a demand of careful design of the encoding method which allows to combine different requirements of both parts of the problem (synthesis of the topology and determination of the parameters).

The first proposed method employs simultaneous solving of the problem. To enable UMDA to solve this problem continuous part of the problem (parameters of the components) is discretized.

Another possibility of solving automated analog circuit synthesis problem is employing of probabilistic models of mixed type as Bayesian networks for variables of mixed discrete/continuous type. Probabilistic model for variables of mixed type was used in algorithm MBOA [45] however experimentation with using MBOA for problems of analog circuit synthesis has not brought satisfactory results. However there are some another possibilities of solving problems of mixed discrete/continuous type (mixtures of distributions, clustering, heterogeneous factorized models (some factors are continuous and some are discrete)). This is one of the areas of possible future research.

The second approach to the automated analog circuit synthesis - separate solving of the topology and the parameters was used in the last two proposed methods. In the first step the candidate topology is chosen using EDA method. In the second step the parameters (the values of the components) of the selected topology were determined using local optimization algorithm. Strong advantage of the separate solving is that this approach allows to use completely different and independent methods for the topology selection and for the parameters determination. From the viewpoint of easy implementation and variability the separate solving approach is advantageous compared to the simultaneous solving.

Another interesting area of the research is utilization of multivariate probabilistic models which are capable to capture higher order dependencies between variables of the solution vectors.

Some of the probabilistic models enable to incorporate some portion of previous knowledge about the problem (priors). Experiments in the area of utilization of different prior informations based on the type of the target application of the analog circuit can be another interesting area of another future research.

13 IMPLEMENTATION NOTES

There have been used several tools and toolboxes which enable proper analysis of the proposed evolutionary electronic methods or estimation of distribution algorithms generally.

All experiments presented in the thesis were performed on 64-bit dual core PC with processor AMD Athlon II X2 245, 8GB RAM. The first two problems (Evolutionary Synthesis of Chaotic Dynamics Using Pure UMDA and Synthesis of Fractional Capacitor Using Hybrid UMDA Algorithm) were implemented in 64-bit version of Matlab 7.8.0 (R2009a) and operational system Ubuntu 12.04. The third problem (Synthesis of Cube Root Function Using Graph EDA Method) was implemented in 64-bit version of Matlab 8.0 (R2012b) and operational system Centos 6.5.

13.1 MATEDA 2.0

There are three main parts of MATEDA 2.0 toolbox [59]. The first one is focused on realization of various types of Estimation of Distribution Algorithms as for example UMDA, FDA, EBNA and MOA. Single-objective and also multi-objective problems can be solved. The second part allows analysis and visualization of particular features of the learned probabilistic models. Last part allows to use the probabilistic models as fitness modeling tools. MATEDA 2.0 toolbox is based on Bayes Net Toolbox for Matlab (BNT package) and Structure Learning Package for Bayes Net Toolbox (BNT SLP package).

13.2 BNLEARN

BNLEARN [60] is R package which allows to learn graphical structure of Bayesian networks, estimate theirs parameters and compute inference. The package implements several constraint-based structure learning algorithms, score-based structure learning algorithms, hybrid structure learning algorithms, local discovery algorithms and Bayesian network classifiers.

13.3 DEAL

DEAL [61] is R package which allows to learn Bayesian networks with variables of discrete, continuous or mixed discrete/continuous types. Compared to package

BNLEARN, the package DEAL is able to learn networks with variables of mixed types.

13.4 COPULAEDAS

This R package allows to implement various types of estimation of distribution algorithms based on copulas and vines. Using the package behavior of the algorithms can be studied and new algorithms can be implemented [62].

13.5 NGSPICE

NGSPICE is open source mixed-level/mixed-signal circuit simulator based on Spice3f5, Cider1b1 and Xspice packages. The package is very useful especially in connection with operational system Linux. The features of the simulator are comparable to commercial circuit simulator hspice. For purposes of research or experiments Ngspice can be used as very good open source alternative to commercial hspice. Simulated circuit is represented in the form of NETLIST.

13.6 HSPICE

HSPICE is commercial spice based circuit simulator which can be considered as industry's standard for accurate circuit simulation. As for any other spice based circuit simulator, simulated circuit is represented in the form of NETLIST.

REFERENCES

[1] DOBOLI, A., VEMURI, R. Exploration based High Level Synthesis of Linear Analog Systems Operating at Low/Medium Frequencies. *IEEE Transactions on CAD*, 2003, Vol. 22, No. 11, pp. 1556-1568.

[2] THOMPSON, A., LAYZELL, P. Analysis of unconventional evolved electronics. *Communications of the ACM*, 1999, Vol. 42, No. 4, pp. 71-79.

[3] SLEZÁK, J., PETRŽELA, J., ŠOTNER, R. On the derivation of Piecewise-Linear Chaotic Oscillators using Simulated Annealing Method and Hspice. *Przeglad Elektrotechniczny*, 2011, vol. 87, no. 1, pp. 262-265, ISSN 0033-2097.

[4] SLEZÁK, J., PETRŽELA, J., ŠOTNER, R. Evolutionary Synthesis of Fractional Capacitor Using Simulated Annealing Method. *Radioengineering*, 2012, vol. 21, no. 4, pp. 1252-1259, ISSN 0033-2097.

[5] SLEZÁK, J. Evolutionary Synthesis of Analog Electronics Using Simulated Annealing. *In Proceedings of Student Conference EEICT 2012*, Brno, 2012, pp. 25-30.

[6] SLEZÁK, J. Evolutionary Synthesis of Chaotic Dynamics Using Univariate Marginal Distribution Algorithm. *In Proceedings of Student Conference EEICT 2013*, Brno, 2013, pp. 19-23.

[7] DEGRAUWE, M. G. R., NYS, O., DIJKSTRA, E., RIJMENANTS, J., BITZ, S., GOFFART, B. L. A. G., VITTOZ, E. A., CSERVENY, S., MEIXENBERGER, C., VAN DER STAPPEN, G., OGUEY, H. J. IDAC: an interactive design tool for analog CMOS circuits.*IEEE Journal of Solid-State Circuits*, 1987, vol. 22, no. 6, pp. 1106-1116.

[8] HARJANI, R. OASYS: a framework for analog circuit synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1989, vol. 8, no. 12, pp. 1247-1266. ISSN 0278-0070.

[9] KOH, H. Y., SEQUIN, C. H., GRAY, P. R. OPASYN: a compiler for CMOS operational amplifiers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1990, vol. 9, no. 2, pp. 113-125.

[10] KRUISKAMP, W., LEENAERTS, D. DARWIN: CMOS opamp Synthesis by Means of a Genetic Algorithm. *32nd Conference on Design Automation DAC'95*, CA: San Francisco, 1995, pp. 433-438, ISBN 0-89791-725-1.

[11] DAVIS, M., LIU, L., ELIAS, J. G. VLSI circuit synthesis using a parallel genetic algorithm. *Proceedings of the First IEEE Conference on Evolutionary Computation*,FL: Orlando, 1994, pp. 104-109, ISBN 0-7803-1899-4.

[12] CHANG, S. J., HOU, H. S., SU, Y. K. Automated passive filter synthesis using a novel tree representation and genetic programming. *IEEE Transactions on Evolutionary Computation*, 2006, vol. 10, no. 1, pp. 93-100.

[13] GRIMBLEBY, J. B. Automatic analogue network synthesis using genetic algorithms. *In Proceeding of First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA)*, Sheffield, 1995, pp. 5358.

[14] GRIMBLEBY, J.B. Hybrid Genetic Algorithms for Analogue Network Synthesis, *Proceedings of the 1999 Congress on Evolutionary Computation CEC 99*, DC: Washington, 1999, ISBN 0-7803-5536-9.

[15] LOHN, J. D., COLOMBANO, S. P. A circuit representation technique for automated circuit design. *IEEE Transactions on Evolutionary Computation*, 1999, vol. 3, no. 3, pp. 205-219.

[16] ZEBULUM, R. S., PACHECO, M. A., VELLASCO, M. M. *Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms*, Florida: CRC Press, 2001, ISBN 0-8493-0865-8.

[17] SAPARGALIYEV, Y., KALGANOVA, T. Unconstrained Evolution of Analogue Computational QR Circuit with Oscillating Length Representation. *In proceeding of: Evolvable Systems: From Biology to Hardware ICES 2008*, Czech Republic: Prague, 2008, pp. 1-10.

[18] ANDO, S., IBA, H. Analog circuit design with a variable length chromosome. *Proceedings of Congress on the Evolutionary Computation*, CA: La Jolla, 2000, vol. 2, pp. 994-1001.

[19] AGGARWAL, V. Evolving sinusoidal oscillators using genetic algorithms. *Proceedings of NASA/DoD Conference on Evolvable Hardware*, 2003, pp. 67-76.

[20] MATTIUSSI, C., FLOREANO, D. Analog Genetic Encoding for the Evolution of Circuits and Networks, *IEEE Transactions on Evolutionary Computation*, 2007, Vol. 11, No. 5, pp. 596-607.

[21] DAS, A., VEMURI, R. GAPSYS: A GA-Based tool for automated passive analog circuit synthesis. *in Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, 2007, pp. 2702-2705.

[22] DAS, A., VEMURI, R. Topology synthesis of analog circuits based on adaptively generated building blocks. *in Proc. of IEEE/ACM Design Automation Conference (DAC)*,CA: Anaheim, 2008, pp. 44-49, ISBN 978-1-60558-115-6.

[23] DAS, A., VEMURI, R. A Graph Grammar Based Approach to Automated Multi-Objective Analog Circuit Design. *In Proc. of Design, Automation, and Test in Europe (DATE)*, France: Nice, 2009, pp. 700-705, ISBN 978-1-4244-3781-8 .

[24] MESQUITE, A., SALAZAR, F. A., CANAZIO, P. P. Chromosome representation through adjacency matrix in evolutionary circuits synthesis. *Proceedings of NASA/DoD Conference on Evolvable Hardware*, 2002, pp. 102-109.

[25] DE SÁ, L. B., Pedro, F. V., MESQUITA, A. Evolutionary Synthesis of Low-Sensitivity Antenna Matching Networks using Adjacency Matrix Representation. *Proceedings of the Eleventh conference on Evolutionary Computation*,Trondheim, 2009, pp. 1201-1208, ISBN 978-1-4244-2958-5.

[26] LARRAÑAGA, P., LOZANO, J. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Kluwer Academic Publishers, 2002.

[27] MÜHLENBEIN, H., PAAß, G. From Recombination of Genes to the Estimation of Distributions I. Binary Parameters, *In Lecture Notes in Computer Science 1411: Parallel Problem Solving from Nature - PPSN IV*, pp. 178-187.

[28] ZINCHENKO, L., MÜHLENBEIN, H., KUREICHIK, V., MAHNING, T. Application of the univariate marginal distribution algorithm to analog circuit design. *Proceedings of NASA/DoD Conference on Evolvable Hardware*, 2002, pp. 93-101.

[29] ZINCHENKO, L., RADECKER, M., BISOGNO, F. Multi-Objective Univariate Marginal Distribution Optimisation of Mixed Analogue-Digital Signal Circuits. *in Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2007)*, England: London, 2007, pp. 2242-2249, ISBN 978-1-59593-697-4.

[30] TORRES, A., PONCE, E. E., TORRES, M. D., DIAZ, E., PADILLA, F. Comparison of Two Evolvable Systems in the Automated Analog Circuit Synthesis. *Eighth Mexican International Conference on Artificial Intelligence (MICAI 2009)*, Guanajuato, 2009, pp. 3-8, ISBN 978-0-7695-3933-1.

[31] KOZA, J. R., BENETT III, F. H., ANDRE, D., KEANE, M. A. Automated WYSIWYG Design of both the topology and component values of electrical circuits using genetic programming. *In Proceedings of the First Annual Conference on Genetic Programming*, 1996, pp. 123-131.

[32] KOZA, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Cambridge: MIT Press, 1994.

[33] KOZA, J. R., ANDRE, D., BENNETT III, F. H., KEANE, M. *Genetic Programming 3: Darwinian Invention and Problem Solving*, San Francisco: Morgan Kaufman, 1999.

[34] KOZA, J. R., KEANE, M. A., STREETER, M. J., MYDLOWEC, W., YU, J., LANZA, G. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*, Kluwer Academic Publishers, 2003.

[35] STARZYK, J. A. Topological Analysis and Diagnosis of Analog Circuits, Gliwice, Wydawnictwo Politechniki Ślaskiej, 2007.

[36] KOTON, J., VRBA, K. Method for Designing Frequency Filters using Universal Current Conveyors. *International Transactions on Computer Science and Engineering*, 2005, Vol. 13, No. 1, pp. 144-154.

[37] VOCHYÁN, J. Synthesis of New Biquad Filters Using Two CFOAs, *Radioengineering*, 2006, vol. 15, no. 4, pp. 76-79, ISSN 1210-2512.

[38] BIOLEK, D., VRBA, K., ČAJKA, J., DOSTÁL, T. General three-port current conveyor: a useful tool for network design, *Journal of ELECTRICAL ENGINEERING*, 2000, vol. 51, no 1-2, pp. 36-39.

[39] DOSTÁL, T., VRBA, K., ČAJKA, J. On Multi-port Current Conveyors, *In Proceedings of the 4th WSEAS international conference on Applications of electrical engineering (AEE'05)*, Wisconsin: Stevens Point, 2005, pp. 261-264, ISBN:960-8457-13-0.

[40] PELIKAN, M., MÜHLENBEIN, H. The bivariate marginal distribution algorithm. *In Advances in Soft Computing Engineering Design and Manufacturing*, London:Springer-Verlag, 1999, pp. 521535.

[41] BALUJA, S. Population-Based Incremental Learning: A method for integrating genetic search based function optimization and competitive learning. *Technical report*, 1994, Carnegie Mellon Report.

[42] HARIK, G. R., LOBO, F. G., GOLDBERG, D. E. The compact Genetic Algorithm. *IEEE Transactions on Evolutionary Computation*, 1999, pp. 287297.

[43] MÜHLENBEIN, H., MAHNIG, T., OCHOA, A. Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, 1999, pp. 215247.

[44] PELIKAN, M., GOLDBERG, D. E., CANTÚ-PAZ, E. BOA: The Bayesian optimization algorithm, Morgan Kaufmann, 1999, pp. 525532.

[45] OČENÁŠEK, J., SCHWARZ, J. Estimation Distribution Algorithm for mixed continuous-discrete optimization problems. *In Proceedings of the 2nd Euro-International Symposium on Computational Intelligence*, 2002, pp. 227-232, ISBN 1-58603-256-9.

[46] OČENÁŠEK, J., KERN, S., HANSEN, N., MÜLLER, S., KOUMOUTSAKOS, P. A Mixed Bayesian Optimization Algorithm with Variance Adaptation. *Lecture Notes in Computer Science*, 2004, pp. 352-361, ISBN 3-540-23092-0.

[47] CHICKERING, D. M., HECKERMA, D., MEEK, C. A Bayesian approach to learning Bayesian networks with local structure. *Technical Report MSR-TR-97-07*, 1997, Redmond: Microsoft Research.

[48] SANTANA, R., OCHOA, A., SOTO, M. R. Factorized Distribution Algorithms for functions with unitation constraints. *In Proceedings of the Third Symposium on Adaptive Systems (ISAS-2001)*, 2001, pp. 158-165.

[49] LOZADA-CHANG, L., SANTANA, R. Univariate marginal distribution algorithm dynamics for a class of parametric functions with unitation constraints, *Information Sciences*, 2011, vol. 181, no. 11, pp. 2340-2355.

[50] EL-MIHOUB, T. A., HOPGOOD, A. A., NOLLE, L., BATTERSBY, A. Hybrid Genetic Algorithms: A Review. *Engineering Letters*, 2006, vol. 13, no. 2, pp. 124-137, ISSN: 1816-093X.

[51] POSPÍŠIL, J., KOLKA, Z., HORSKÁ, J., BRZOBOHATÝ, J. Simplest ODE Equivalents of Chua s Equations. *International Journal of Bifurcation and Chaos*, 2000, vol. 10, no. 1, ISSN 0218-1274.

[52] TOH, M. Synthesis of electronic circuits for simulating nonlinear dynamics. *International Journal of Bifurcation and Chaos*, 2001, vol. 11, no. 3, pp. 605653.

[53] CARLSON, G. E., HALIJAK, C. A. Approximation of Fractional Capacitors (1/s)1/n by a Regular Newton Process. *IEEE Trans. On Circuit Theory*, 1964, Vol. 11, No. 2, pp. 210-213, ISSN 0018-9324.

[54] HARTLEY, T., LORENZO, C., QAMMER, H. K. Chaos in a Fractional Order Chua's System. *IEEE Trans. on CAS-I: Fund. Theory and Applications*, 1995, Vol. 42, No. 8, pp. 485-490, ISSN 1057-7122.

[55] STEIGLITZ, K. An RC Impedance Approximantion to $s^{1/2}$. *IEEE Trans. On Circuit Theory*, 1964, Vol. 11, No. 1, ISSN 0018-9324.

[56] ORTIGUEIRA, M. D. An Introduction to the Fractional Continuous Time Linear Systems, *IEEE Circuits and Systems Magazine*, 2008, Vol. 8, No. 3, pp. 19-26, ISSN 1531-636X.

[57] KOZA, J. R., BENNETT, F. H., FORREST, H., LOHN, J., DUNLAP, F., ANDRE, D., KEANE, M. A. Automated synthesis of computational circuits using genetic programming. *In: IEEE conference on evolutionary computation*,IN: Indianapolis, 1997, pp. 447452, ISBN 0-7803-3949-5.

[58] HANDA, H. Use of graph kernels in Estimation of Distribution Algorithms. *In: IEEE congress on evolutionary computation (CEC)*, QLD: Brisbane, 2012, pp. 1-6, ISBN 978-1-4673-1510-4.

[59] SANTANA, R., ECHEGOYEN, C., MENDIBURU, A., BIELZA, C., LOZANO, J. A., LARRAÑAGA, P., ARMAÑANZAS, R., SHAKYA, S. MATEDA: A suite of EDA programs in Matlab. *Technical Report EHU-KZAA-IK-2/09*,2009, University of the Basque Country.

[60] SCUTARI, M. Learning Bayesian Networks with the bnlearn R Package. *Journal of Statistical Software*, 2010, vol. 35, no. 3, pp. 1-22, 2010.

[61] BTTCHER, S. G., DETHLEFSEN, C. DEAL: A Package for Learning Bayesian Networks, *Journal of Statistical Software*, 2003, vol. 8, no. 20, pp. 200-208.

[62] GONZÁLES-FERNÁNDEZ, Y., SOTO, M. Copulaedas: Estimation of Distribution Algorithms Based on Copula Theory, R package version 1.0.1 (2011), http://CRAN.R-project.org/package=copulaedas.

ABBREVIATIONS

AC	Autonomous Circuit
AGE	Analog Genetic Coding
AMBOA	Adaptive Mixed Bayesian Optimization Algorithm
BDe	Bayesian Dirichlet Equivalence
BJT	Bipolar Junction Transistor
BMDA	Bivariate Marginal Distribution Algorithm
BOA	Bayesian Optimization Algorithm
CART	Classification and Regression Tree
CFDA	Constraint Factorization Distribution Algorithm
CFOA	Current Feedback Operational Amplifier
cGA	Compact Genetic Algorithm
EBNA	Estimation of Bayesian Network Algorithm
EDA	Estimation of Distribution Algorithm
FDA	Factorization Distribution Algorithm
FPTA	Field Programmable Transistor Array
GA	Genetic Algorithm
GCC	General Three Port Current Conveyor
GhEDA	Graph Hybrid Estimation of Distribution Algorithm
GP	Genetic Programming
hBOA	Hierarchical Bayesian Optimization Algorithm
HW	Hardware
ILG	Increasing Length Genotype
LSA	Local Search Algorithm
MBOA	Mixed Bayesian Optimization Algorithm
MOA	Markovian Optimization Algorithm
OLG	Oscillation Length Genotype
OTA	Operational Transconductance Amplifier
PBIL	Population Based Incremental Learning
PLS	Probabilistic Logic Sampling
PMBGA	Population Model-Building Genetic Algorithms
PWL	Piece-Wise Linear
RF	Radio Frequency
RLC	Resistors, Inductors and Capacitors
SA	Simulated Annealing
SoC	A System On Chip
SUS	Stochastic Universal Sampling
UDIP	Uniformly Distributed Initial Population
UMDA	Univariate Marginal Distribution Algorithm
VLSI	Very Large Scale Integration

SYMBOLS

Δ_m	difference of magnitude
Δ_p	difference of phase
$b1, b2, \ldots bx$	bits of encoding vector
C_{act}	analog circuit consisting of transistors
C_e	analog circuit corresponding to graph G_{cpe}
C_h	analog circuit corresponding to hypergraph G_h
C_m	analog circuit corresponding to multigraph G_m
C_{mt}	analog circuit of mixed type (passive components and transistors)
C_p	analog circuit corresponding to graph G_p
C_{pas}	analog circuit consisting of passive components
C_s	analog circuit corresponding to graph G_s
C_t	analog circuit corresponding to hypergraph G_t
$cost$	cost value
c_1	cost value of the solution before execution of LSA
c_2	cost value of the solution after execution of LSA
$dbl1, \ldots, dblx$	parameters of encoding vector
E_a	set of edges of graph G_a
E'_a	set of edges of graph G'_a
E_{cp}	set of edges of graph G_{cp}
E_{ct}	set of hyperedges of hypergraph G_{ct}
E_h	set of labeled edges of hypergraph G_h
E_m	set of parametrized edges of multigraph G_m
E_s	set of parametrized edges of graph G_s
e	encoding vector
$e_0, e_2, \ldots e_x$	edge of graph or hyperedge of hypergraph
e_h, e_{h1}, e_{h2}	characteristic vector of 3-uniform hypergraph
e_{hl}	characteristic vector of labeled 3-uniform hypergraph
e_m	characteristic vector of multigraph
e_{npn}	encoding vector representing topology of NPN transistors
e_p	encoding vector of topology of passive circuit
e_{pnp}	encoding vector representing topology of PNP transistors
e_{ps}	parameters storage vector
e_{res}	encoding vector representing topology of resistors
e_s, e_{s1}, e_{s2}	characteristic vector of simple graph
e_t	encoding vector of topology of circuit consisting of transistors
e_{vccp}	encoding vector of connection of positive voltage sources
e_{vccn}	encoding vector of connection of negative voltage sources
f_{mc}	magnitude function of the current solution
f_{md}	desired magnitude function
f_{pc}	phase function of the current solution
f_{pd}	desired phase function

f_{vc}	voltage function of the current solution
f_{vd}	desired voltage function
G_a	generic floating graph with self-loops
G'_a	graph corresponding to adjacency matrix representation
G_{act}	graph representation of active part of a mixed type circuit
G_c	complete graph corresponding to multigraph G_m
G_{cp}	complete graph of representation of passive circuit
G_{cpe}	complete expanded graph (topology of passive circuit)
G_{ct}	complete 3-uniform hypergraph (topology of transistors)
G_h	3-uniform hypergraph
G_{hc}	complete 3-uniform hypergraph
G_{hl}	3-uniform labeled hypergraph
G_m	multigraph
G_{npn}	3-uniform hypergraph representing topology of NPN transistors
G_{npnc}	complete hypergraph of representation of topology of NPN transistors
G_p	graph representing topology of passive circuit
G_{pas}	graph representation of passive part of a mixed type circuit
G_{pnp}	3-uniform hypergraph representing topology of PNP transistors
G_{res}	graph representing topology of resistors
G_{resc}	complete graph of representation of topology of resistors
G_s, G_{s1}, G_{s2}	simple graph
G_{sc}	complete graph corresponding to simple graph G_s
G_t	3-uniform hypergraph (topology of connection of transistors)
G_{vccn}	graph representing connection of V_{ccn}
G_{vccp}	graph representing connection of V_{ccp}
$I, I(i)$	individual of population
k	counter of generations
$LSevals$	number of objective function evaluations for LSA (one generation)
$M, M(k)$	probabilistic model
M_a	adjacency matrix corresponding to graph G_a
M'_a	adjacency matrix corresponding to graph G'_a
$MaxFunEvals$	number of objective function evaluations of LSA (one run)
$MaxGen$	maximal number of generations per run
m	number of points of frequency response
m_{ij}	component of adjacency matrix M_a
N_h	set of the nodes of analog circuit C_h
N_m	set of the nodes of analog circuit C_m
N_p	set of nodes of passive analog circuit
N_s	set of nodes of analog circuit C_s
n	length of encoding vector

$n_0, n_2, \ldots n_x$	node of analog electronic circuit
n_{adm}	number of generalized admittances of AC
n_c	maximal number of the components of the desired circuit
n_e	number of edges or hyperedges
n_{edgcpe}	number of edges of expanded complete graph G_{cpe}
n_{edgnpn}	number of hyperedges of hypergraph G_{npn}
n_{edgres}	number of edges of complete graph G_{resc}
n_{evals}	number of objective function evaluations
n_{gen}	number of generations
n_{hl}	length of vector e_{hl}
n_i	size of population
n_m	number of multiple edges of multigraph
n_n	number of nodes of an analog circuit
n_{npn}	maximal number of NPN transistors of the desired circuit
n_{opt}	number of optimized individuals
n_p	number of ports of generalized admittance
n_{pnp}	maximal number of PNP transistors of the desired circuit
n_{res}	maximal number of resistors of the desired circuit
n_{sc}	size (number of edges) of graph G_{sc}
n_{vccn}	maximal number of nodes connected to V_{ccn}
n_{vccp}	maximal number of nodes connected to V_{ccp}
$P, P(k)$	population of individuals
P_1	set of parameters before execution of LSA
P_2	set of parameters after execution of LSA
P_{add}	probability of adding edges using probabilistic model M
$P_g, P_g(k)$	population of generated individuals (samples)
P_{LSA}	probability of execution of LSA
P_n, P_nk	new population
P_{rem}	probability of removing edges and hyperedges
$P_s, P_s(k)$	selected population of individuals
$PopEvals$	number of objective function evaluations (for evaluation of P)
$PopSize$	size of population
$p1, p2, \ldots p135$	parameters loaded from e_{ps}
$p_l(x)$	probabilistic vector
Q_h	set of labels of hyperedges E_h
Q_m	set of labels of edges E_m
Q_s	set of parameters of edges E_s
$q_1, \ldots, q_x$	parameter of parametrized edge or hyperedge
r	parameter loaded from e_{ps}
$r_{l;i}(x_i)$	marginal frequency
S_l	set of parameters loaded from e_{ps}
S_p	set of variables of e consisting information about parameters
S_t	set of variables of e consisting information about topology

s_{npn}	sorted information of vector v_{npn}
s_{pnp}	sorted information of vector v_{pnp}
s_{res}	sorted information of vector v_{res}
s_{vccn}	sorted information of vector v_{vccn}
s_{vccp}	sorted information of vector v_{vccp}
U_2	output voltage
$u(x)$	unitation value
V_a	set of vertices of graph G_a
V'_a	set of vertices of graph G'_a
V_{ccn}	negative voltage source
V_{ccp}	positive voltage source
V_{cp}	set of vertices of graph G_{cp}
V_{ct}	set of vertices of hypergraph G_{ct}
V_h	set of vertices of hypergraph G_h
V_m	set of vertices of multigraph G_m
V_s	set of vertices of graph G_s
$v_0, v_2, \ldots v_x$	vertex of graph or hypergraph
v_{exp}	value of exponent
v_{in}, U_1	input voltage
v_{man}	value of mantissa
v_{mult}	value of multiplier
v_{npn}	vector of marginal frequencies of NPN transistors
v_{pnp}	vector of marginal frequencies of PNP transistors
v_{res}	vector of marginal frequencies of resistors
v_{type}	type of the component
v_{vccn}	vector of marginal frequencies of nodes connected to V_{ccn}
v_{vccp}	vector of marginal frequencies of nodes connected to V_{ccp}
val	value of a component
val_{max}	maximal value of a component
val_{min}	minimal value of a component
w_{cm}	weight of difference of magnitude
w_{cp}	weight of difference of phase
$w_{dm}(i)$	weight values for differences of magnitude
$w_{dp}(i)$	weight values for differences of phase
w_v	weight of cube root objective function
$Y_1, Y_2, \ldots Y_x$	generalized admittance
$Y(s)$	admittance network
Z_{in}, Z_{in1}, Z_{in2}	input impedance function

CURRICULUM VITAE

Personal Data:

Name:	Josef Slezák
Date of Birth:	June 17, 1982
Address:	Havlíčkova 1280, 76502 Otrokovice
E-mail:	xsleza08@stud.feec.vutbr.cz

Education:

- **2007 - 2014:** Brno University of Technology, Brno, Czech Republic.

  Ph.D. Candidate, Electrical Engineering.

  Thesis: "Evolutionary Synthesis of Analog Electronic Circuits Using EDA Algorithms".

  Supervisor: prof. Ing. Tomáš DOSTÁL, DrSc. .

- **2001 - 2007:** Brno University of Technology, Brno, Czech Republic.

  Master degree, Electrical Engineering.

  Thesis: "Demonstration of Programmable Logic Devices".

  Supervisor: Doc. Ing. Jaromír KOLOUCH, CSc.

Selected Publications

SLEZÁK, J., PETRŽELA, J. Evolutionary Synthesis of Cube Root Computational Circuit Using Graph Hybrid Estimation of Distribution Algorithm. Accepted in *Radioengineering*.

SLEZÁK, J., PETRŽELA, J., ŠOTNER, R. Evolutionary Synthesis of Fractional Capacitor Using Simulated Annealing Method. *Radioengineering*, 2012, vol. 21, no. 4, pp. 1252-1259, ISSN 0033-2097.

SLEZÁK, J., PETRŽELA, J., ŠOTNER, R. On the derivation of Piecewise-Linear Chaotic Oscillators using Simulated Annealing Method and Hspice. *Przeglad Elektrotechniczny*, 2011, vol. 87, no. 1, pp. 262-265, ISSN 0033-2097.

SLEZÁK, J., ŠOTNER, R. Circuit synthesis using admittance network modification in MATLAB. *In Proceedings of Conference on Mixed Design of Integrated Circuits and Systems (MIXDES'09)*, Poland: Lodz, 2009, pp. 613-617, ISBN 978-1-4244-4798-5.

SLEZÁK, J., PETRŽELA, J., ŠOTNER, R. Designing of arc oscillator using CFA amplifier. *In Proceedings of 17th International Electrotechnical Conference*, Slovenija: Portorož, 2008, pp. 47-50, ISBN 1-58145-720-0.