StuCoSReC

Proceedings of the 2014 1st Student Computer Science Research Conference

University of Primorska Press

StuCoSReC
Proceedings of the 2014 1st Student Computer Science Research Conference

Edited by
Iztok Fister jr. and Andrej Brodnik

Published by
University of Primorska Press
Titov trg 4, si-6000 Koper

Editor-in-Chief
Jonatan Vinkler

Managing Editor
Alen Ježovnik

Koper, 2014

ISBN 978-961-6963-03-9 (pdf)
www.hippocampus.si/ISBN/978-961-6963-03-9.pdf
ISBN 978-961-6963-04-6 (html)
www.hippocampus.si/ISBN/978-961-6963-04-6/index.html

CIP - Kataložni zapis o publikaciji
Narodna in univerzitetna knjižnica, Ljubljana

004(082)(0.034.2)

STUCOSREC [Elektronski vir] : proceedings of the 2014 1st Student Computer Science Research Conference / edited by Iztok Fister jr. and Andrej Brodnik. - El. knjiga. - Koper : University of Primorska Press, 2014

Način dostopa (URL): www.hippocampus.si/isbn/978-961-6963-03-9.pdf
Način dostopa (URL): www.hippocampus.si/isbn/978-961-6963-04-6/index.html

ISBN 978-961-6963-03-9 (pdf)
ISBN 978-961-6963-04-6 (html)
1. Fister, Iztok, ml.
275792896

Preface

Computer science is one of the quickest developing areas of Science. This area captures a lot of disciplines that have been developed independently. Thus, new disciplines have been arisen. On the other hand, the computer science starts connecting with the other natural sciences in order to draw an inspiration for solving their problems. Nowadays, incorporating principles from biology (e.g., Darwinian evolution or behavior of social living insects and animals) into the computer algorithms have been observed. In general, this means enough material for a computer science conference. This volume contains papers as presented in the first *Student Computer Science Research Conference 2014* (*StuCoSRec*) held in Maribor under the cover of the 17th MultiConference on Information Society in Ljubljana and ACM Slovenia.

The 1st Student Computer Science Research Conference is an answer to the fact that modern PhD and MSc programs foster early research activity among the students of computer science. The prime goal of the conference is to become a place for students to present their research work and hence further encourage students for an early research. Besides the conference, it also wants to establish an envi ronment where students from different institutions meet, let know each other, exchange the ideas, and nonetheless make friends and research colleagues. In line with this, we would like to exhibit research efforts of students in four Slovenian institutions, i.e., University of Maribor, University of Ljubljana, University of Primorska and Jožef Stefan international post-graduate school, neighboring countries, i.e., Austria, Croatia, Hungary and Italy, and the other countries in the world (e.g., one of the papers on out conference comes from China). At last but not least, the conference is also meant to serve as meeting place for students with senior researchers from different institutions.

Eleven papers addressed this conference,[1] covering several topics of the computer science. All the papers were reviewed by two international reviewers and accepted for the oral presentation. This fact confirms a good work with authors in their research institutions. The content of the papers will be presented in three sections that cover theory and applications, computer graphics, image processing and pattern recognition, and semantic web and data mining.

The conference is dedicated to graduate and under-graduate students of computer science and is therefore free of charge. In line with this, the organizing committee would like to thank to the University of Maribor, especially the Faculty of Electrical Engineering and Computer Science (FERI) for the support. Especially, we are grateful to the Institute of computer science at FERI Maribor for payment of conference costs arising during the organization. At the end, a special thank goes to the dean of UM FERI, prof. Borut Žalik for his unselfish support.

1 Conference site is at http://labraj.uni-mb.si/stucosrec2014/.

Contents

Spatially Embedded Complex Network Estimation Using Fractal Dimension

David Jesenko *
University of Maribor
Faculty of Electrical Engineering and Computer Science
Smetanova 17, SI-2000
Maribor, Slovenia
david.jesenko1@um.si

Domen Mongus
University of Maribor
Faculty of Electrical Engineering and Computer Science
Smetanova 17, SI-2000
Maribor, Slovenia
domen.mongus@um.si

Borut Žalik
University of Maribor
Faculty of Electrical Engineering and Computer Science
Smetanova 17, SI-2000
Maribor, Slovenia
borut.zalik@um.si

ABSTRACT
This paper describes complex networks in regards to their fractal characteristic. Firstly, a short background concerning fractal dimensionality based on box-counting is considered, while in the continuation its application on spatially embedded networks is proposed using rasterization at multiple scales. Finally, the obtained results are presented.

General Terms
Computer science, algorithms

Keywords
Complex networks, fractals, fractal dimension

1. INTRODUCTION
Today, complex networks are amongst the most advanced approaches for describing and analyzing structures composed of a plurality of blocks. When vertex has a fixed position in space, we are talking about spatially embedded networks [1]. Similar to graphs, networks are represented by a set of vertices and an unordered set of edges linking them. The main difference between them is of topological nature. Complex networks have nontrivial and unpredictable topology [2]. In addition, topology of complex networks hide important information about vertices. Evaluation of these features can reveal many otherwise hidden characteristics of geometric structures as for example separation, connectedness, compactness, or countability conditions [3].

With their characteristic self-similar property and huge amount of possibilities to analyse them, fractals are one of the most interesting areas of computer science. Among various dimensions that are extensively studied in the literature, Peitgen et al. introduced three main Mandelbrot fractal dimensions [4]:

- self-similarity dimension,
- compass dimension (also called divider dimension),
- box-counting dimension.

The box-counting dimension is the most frequently used in computer-supported applications. The box-counting algorithm uses a uniform grid with cell-size s_1, where the number of grid-cells N_1 containing the object is counted. Obviously, at an increased sampling scale s_2, an increased value of grid-cells containing structures N_2 is expected. A fractal dimension D, is then given by [5]:

$$D = \frac{\log N_1 - \log N_2}{\log s_1 - \log s_2}. \quad (1)$$

Measuring fractal dimension with the box-counting algorithm has been presented at various places. Conci and Proença in [6] used this method for visual inspection, which is important part of quality control in textile industry. In [7] Boshoff presented fast box-counting algorithm for determining the fractal dimension of sampled continuous functions. Zhang et al. in [8] presented coarse iris classification using box-counting algorithm to estimate fractal dimensions.

The actual fractals are difficult to be detected and evaluated. Frequently, a definition of fractal dimension is used for this purposes [4]. The dimension of fractals is usually smaller than the space in which fractals are embedded. However, the fractal dimension remains the same regardless of the resolution used for fractal dimensionality estimation. Although this definition allows for identification of fractal geometry, fractal dimension of topology is still not clearly defined [9].

In this paper we present an implementation of a box-counting method for estimating fractal dimension of topology of spatially embedded complex networks. Section 2 gives a short theoretical background on complex networks. In Section 3, a method for estimating fractal dimensions of topologies is presented. Implementation details are given in Section 4. Results are presented in Section 5, while Section 6 concludes the paper.

*Corresponding author.

2. COMPLEX NETWORKS

Nature consists of many complex networks established over various kinds of vertices with different relations. Obvious examples are social networks, where vertices represent people and edges defines different relations between them, e.g. business, friendship, or family [10]. In all these cases, a network N is defined by the set of vertices $V(N) = \{v_i\}$ and the set of edges $E(N) = \{e_{ij}\}$, where $e_{ij} = (v_i, v_j)$ represents a pair of vertices $v_i, v_j \in V(N)$.

2.1 Spatially embedded complex networks

Depending on the length of the edges between vertices, two kinds of spatially embedded networks are known [11]: edges can either exist only between the nearest neighbours, or they may spend over the entire network. In all the cases, power law distribution of edge lengths is given as [11]:

$$P(r) \sim r^{-\delta}, \tag{2}$$

where r is the distance between the vertices. Exponent δ affects the network dimension as:

- if $4 < \delta$, dimension is equal 2, because the value of $P(r)$ is low.
- if $2 \leq \delta \leq 4$, dimension monotonous grows from $2 \rightarrow \infty$ with decreasing δ.
- if $\delta < 2$, dimension converts towards infinity.

3. FRACTALS

Fractals are geometric objects, which cannot be described by ideal or primitive geometric objects, like cubes, spheres, triangles, or polygons. Usually they are related with recursive operations on geometric objects and number sets [4]. These operations are mathematically straightforward, however the results are complex, with interesting and remarkable properties. The most important property of fractals is self-similarity [4]. Examples of this can be found in the Cantor set (Figure 1), Sierpinski triangle (Figure 2) and Koch snowflake (Figure 3).

Figure 1: Cantor set.

Figure 2: Sierpinski triangle.

4. IMPLEMENTATION

First, an application for generating different kind of complex networks has been developed. As explained in Section 2, two types of spatially embedded networks exist. An example of a network where vertices are connected with their nearest neighbours is shown in Figure 4. For creating these types of networks, an efficient method for searching nearest neighbours, implemented in nano-flann library, available in [12] has been used. It is based on approximate nearest neighbours search algorithm introduced in [13]. In Figure 4, the brighter vertices are those with more edges than the darker one.

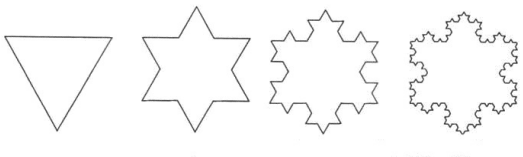


Figure 3: Koch snowflake.

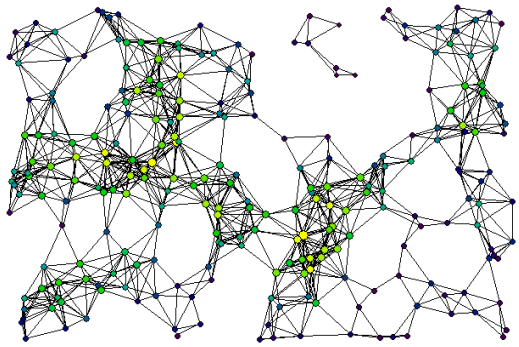


Figure 4: Complex network containing edges to their nearest neighbours.

When considering complex networks (as shown in Figure 5) with edges spanning over the entire space, the implementation was based on similarity measurement of vertices (according to a user-specified feature). Obviously, the construction of this type of network is significantly more expensive as it demands comparison of each pair of vertices. This leads to a quadratic time complexity, while approximate search for the nearest neighbours can be achieved in logarithmic time [14].

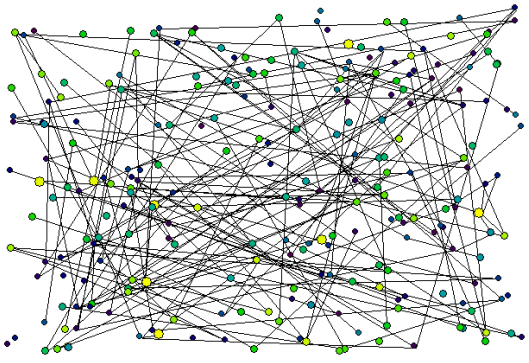


Figure 5: Complex network with edges over the whole network.

4.1 Box-counting algorithm

A double-step algorithm using only integer arithmetic was used for the implementation of line rasterization with minimized oversampling. As proposed in [15], the method uses the coding of changes in directions based on DDA (Digital Differential Analysis). In order to increase efficiency of box-counting, the scales of both uniform superimposed grids should be in the ration 1:2. In this approach each box from a grid is subdivided into four boxes each of half the size in the next grid. Equation for fractal dimension is now [4]:

$$\frac{\log N(2^{-(k+1)}) - \log N(2^{-k})}{\log 2^{k+1} - \log 2^{k}} = \log_2 \frac{N(2^{-(k+1)})}{N(2^{-k})}. \qquad (3)$$

The result uses the base 2 logarithm due to the factor by which the box-count increases from one grid-level to the next [4].

5. RESULTS

Implemented algorithm runs on a computer system with the following configuration:

- processor Intel Core2 Quad CPU Q9550 with beat 2.83GHz,
- memory DDR3 of size 8GB,
- graphical card ATI Radeon HD 4600 Series with 1GB video memory,
- operating system Windows 7 Pro - 64 bit.

Figures 6 and 7 shows spatially-embedded complex networks superimposed on grids of two different scales. For practical purposes it is often convenient to consider a sequence of grids where the mesh size is reduced by a factor of 1/2 from one grid to the next. In other words, if the number of boxes counted increases by a factor of 2^D when the box size is halved, then the fractal dimension is equal to D. More in [4].

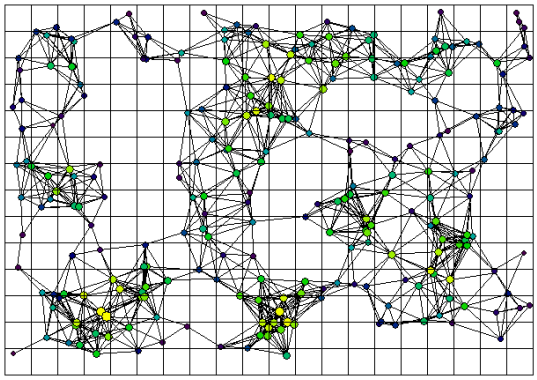


Figure 6: Fractal superimposed with grid, where $s = 1/20$ width of superimposed grid.

In Table 1, fractal dimension calculated by Eqn. (1) at a changing scale can be seen. Complex network was composed

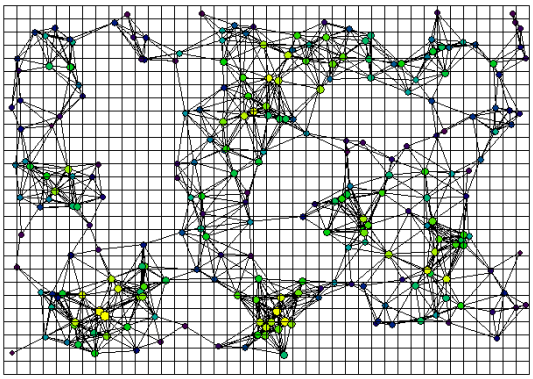


Figure 7: Fractal superimposed with grid, where $s = 1/40$ width of superimposed grid.

of $|V(M)| = 250$ vertices and $|E(M)| = 2554$ edges. With an increasing scale, CPU time is decreased. As seen, the box-counting fractal dimension is practically stable. The variations can be assigned to the point-to-cell algorithm, which approximate the complex network more accurately at larger resolutions.

Table 1: Box-counting fractal dimension.

Cell size	CPU time [ms]	Fractal dimension
1/35 and 1/40	59.4	1.31
1/30 and 1/40	56.1	1.33
1/25 and 1/40	50.4	1.33
1/20 and 1/40	45.3	1.34
1/20 and 1/35	39.7	1.35
1/20 and 1/30	33.5	1.35
1/20 and 1/25	28.9	1.36
1/15 and 1/20	26.2	1.37

The spent CPU efficiency of the implementation regarding different network size is given in Tables 2 and 3. The scales used were 1/20 and 1/40 width of superimposed grid. As expected, the larger networks demand longer time. Differences in CPU efficiency between both kinds of networks can be noticed when comparing Tables 2 and 3. It is a consequence of rasterization algorithm. Edges in networks, with edges over the whole network, are in average longer and their rasterization therefore take more time.

Table 2: Spent CPU time for evaluation of spatial dimension with nearest neighbour edges.

Number of vertices	Number of edges [millions]	Time [s]
100	0.001	0.01
500	0.008	0.1
1000	0.04	0.3
5000	0.9	0.6
10000	3.6	1.1
25000	23.7	4.7
50000	92.1	18.2
75000	207.5	37.4

Table 3: Spent CPU time for evaluation of spatial dimension with edges over the whole network.

Number of vertices	Number of edges [millions]	Time [s]
100	0.0008	0.015
500	0.005	0.2
1000	0.03	0.45
5000	0.8	0.7
10000	3.1	1.3
25000	18.3	5.4
50000	79.8	23.8
75000	176.1	47.1

6. CONCLUSION

In this paper, a brief theoretical background on complex networks and fractals (fractal dimension) has been given at first. The implementation of the box-counting algorithm for evaluation of fractal dimension of spatially embedded complex networks follows. The efficiency of the implementation is given for two types of complex networks.

7. REFERENCES

[1] L. Barnett, E. Di Paolo, and S. Bullock. Spatially embedded random networks. *Physical Review E*, 76(5):056115, 2007.

[2] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D. U. Hwang. Complex networks: Structure and dynamics. *Physics reports*, 424(4):175–308, 2006.

[3] R. Engelking. General topology. 1989.

[4] H.O. Peitgen, H. Jürgens, and D. Saupe. *Chaos and fractals: new frontiers of science*. Springer, 2004.

[5] B.S. Raghavendra and N.D. Dutt. Computing fractal dimension of signals using multiresolution box-counting method. *International Journal of Information and Mathematical Sciences*, 6(1):50–65, 2010.

[6] A. Conci and C. B. Proença. A fractal image analysis system for fabric inspection based on a box-counting method. *Computer Networks and ISDN Systems*, 30(20):1887–1895, 1998.

[7] H. F. V. Boshoff. A fast box counting algorithm for determining the fractal dimension of sampled continuous functions. In *Communications and Signal Processing, 1992. COMSIG'92., Proceedings of the 1992 South African Symposium on*, pages 43–48. IEEE, 1992.

[8] L. Yu, D. Zhang, K. Wang, and W. Yang. Coarse iris classification using box-counting to estimate fractal dimensions. *Pattern Recognition*, 38(11):1791–1798, 2005.

[9] O. Shanker. Defining dimension of a complex network. *Modern Physics Letters B*, 21(06):321–326, 2007.

[10] M. Aldana. Complex networks. *Available on: http://www.fis.unam.mx/ max/English/notasredes.pdf*, 2006.

[11] L. Daqing, K. Kosmidis, A. Bunde, and S. Havlin. Dimension of spatially embedded networks. *Nature Physics*, 7(6):481–484, 2011.

[12] Nano-flann. Available on: https://code.google.com/p/nanoflann.

[13] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923, 1998.

[14] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[15] Yong Kui Liu, Peng Jie Wang, Dan Dan Zhao, Denis Spelic, Domen Mongus, and Borut Zalik. Pixel-level algorithms for drawing curves. In *Theory and Practice of Computer Graphics*, pages 33–40. The Eurographics Association, 2011.

Graph coloring based heuristic for driver rostering

László Hajdu
University of Szeged
Dugonics square 13.
Szeged, Hungary
hajdul@jgypk.u-szeged.hu

Miklós Krész
University of Szeged
Dugonics square 13.
Szeged, Hungary
kresz@jgypk.u-szeged.hu

Attila Tóth
University of Szeged
Dugonics square 13.
Szeged, Hungary
attila@jgypk.u-szeged.hu

ABSTRACT
Nowadays, the companies and institutions have numerous employees therefore the crew rostering problem became increasingly important. The objective is to assign the crew members to previously generated daily shifts, while meeting the constraints, and to optimize the overall cost. In this paper we present the mathematical model of the problem and introduce a two-phase graph coloring method for the crew rostering problem. Our method has been tested on artificially generated and real life input data. The results of the new algorithm have been compared to the solutions of the appropriate integer programming model for problems with moderate size.

Categories and Subject Descriptors
H.4 [**Information Systems Applications**]: Miscellaneous; H.4.2 [**Types of Systems**]: Decision support(e.g., MIS)

General Terms
Algorithms, Management, Measurement, Performance

Keywords
Driver rostering, Graph coloring, Tabu Search

1. INTRODUCTION
The crew rostering appears in various areas. The main challenge is to assign an optimal number of crew members to the pre-defined daily shifts in a way where the resources are used in the most efficiently way, meanwhile, the solution meets the defined constraints of the workplace. It can be stated that it is possible to use the model in every sector which deals with crew members. Some of the typical application areas where the crew rostering can be applied: public transport companies, call centers, and nurse scheduling. The variations of the problem are NP-hard or NP-complete therefore extremely hard to solve. [8, 1, 10]

In most cases the length of the shifts are different, which means that the employees don't spend the same amount of time working. These differences may produce an overtime cost which is added to the basic salary. However, the companies and institutions must guarantee the basic salary for everyone, even if the employee does not spend the normal amount of time at work. This causes an additional cost for the companies.

In this paper we define our crew rostering problem, introduce the mathematical model and give our new heuristic solution for the problem. The results of the algorithm have been compared to the results of the appropriate integer programming model for problems with moderate size. These results show that our algorithm is very efficient for solving this problem. Our solution is provided for the driver rostering problem, although it can be utilized in crew rostering within a wider range of propositions.

2. CREW ROSTERING
The problem is based on generalized set covering model. Dantzig was the first to deal with its mathematical application[4]. The literature gives numerous examples of the issue. The most significant ones are the airline [5], the call center[12], the nurse scheduling [3] and the driver scheduling, what is the topic of our research. Since giving an exact solution to these problems is only possible by moderate sample size, thus these are solved usually with heuristics. Several solutions can be found in the literature, such as: ant colony optimization, dynamic programming, genetic algorithm, simulated annealing or tabu search method. A good overview of the above methods with respect to driver rostering can be found in [11].

Our crew rostering problem is formally defined as the following. Let c be the set of employees. A set of shifts denoted by S needs to be carried out. A shift is composed of a series of tasks. A shift is defined by its date, starting and ending time, duty time, and its working time. The working time can be different from the duty time because of breaks or idle activities. Our objective is to assign the crew members to the shifts. Consequently let $f = s \rightarrow c$ an assignment where the shifts are covered by the employees in a way where there is a person assigned to every shift. These solutions need to correspond the regulations of the company as well as the norms of the european union. For example, that the crew members can get at most one shift in one day. Besides this, the following attributes were regulated during this problem:

- Minimum rest time between two shifts.
- Maximum worktime during one week.
- Minimum free days in one month.
- Maximum consecutive workdays.

The aim is to give a solution which meets the regulations above, while minimizing the cost. Let $ct(i)$ be the type of contract belonging to driver i. For any contract type c, let $aw(c)$ return the expected daily working hours that belong to c. Each driver i has such a contract that defines his required daily working time in average. Based on their contract it is possible to calculate the expected worktime in the planning period. In our case every crew member had a contract with eight hours of worktime. The expected worktime could be defined the following way:

$$expected\ worktime = number\ of\ workdays * aw(ct(i))$$

The employee needs to be paid based on their expected worktime written in their contract. Also, in the case of working over the expected worktime, the employers have to pay extra for this overtime. The optimal case is when every employee works in their expected worktime. Therefore, the cost is the following:

$$cost = \alpha * overtime + \beta * employment\ cost$$

Where α, β are weights. Hence the objective was that the overtime is as low as possible while the number of the employees is minimal.

3. MATHEMATICAL MODEL

Let D be the days of the planning period, D^{Week} the days of the week and D^{Mon} is the days of a months. Meanwhile the lenght of the planning period needs to be defined denoted by L, and let L_m the number of days in the months denoted by m. Furthermore let j and p ($j \in D, p \in D$) the days and w the weeks. The set of the drivers are denoted by C, and let S be the shifts where S_j is a shift on day j. Let i ($i \in C$) the driver i and k and q ($k \in S_j, q \in S_j$) one shift each. SS_{pq}^{jk} is a compatibility relation, which takes up the value of 1 if the shift k on day j and shift q on day p can be assigned to the same person, and 0 otherwise. Let WT be the maximal worktime on a week, WD the maximum number of consecutive workdays and D_p^{WD} the consecutive days beginning from day p (WD+1). The maximum number of free days is denoted by RD. For every driver a worktime being defined in the contract is needed. Let this worktime be $ct(i, m)$. In order to be able to minimize the number of employees let define $cc(i)$ as the operational cost. Let $wt(j, k)$ be and $st(j, k)$ be the worktime and the duty time of every shift k on every day j. Finally let ET be the value used by multiplying the overtime.

We needed the followings in order to be able to define the model:

Driver i gets shift k on day j:

$$x_{ijk} \in \{0, 1\}$$

Driver i works on day j:

$$z_{ij} \in \{0, 1\}$$

Driver i works in the planning period:

$$y_i \in \{0, 1\}$$

Overtime of driver i:

$$\pi_i \geq 0$$

The constraints of the appropriate integer programming model were the followings:

Every driver needs to be assigned to exactly one shift.

$$\forall j \forall k \sum_i x_{ijk} = 1 \tag{1}$$

A driver only works on a certain day if there is a shift assigned to them and only one person can be assigned to a shift.

$$\forall i \forall j \sum_i x_{ijk} = z_{ij} \tag{2}$$

A driver is employed if they are working in the planning period.

$$\forall i \sum_j z_{ij} \leq L y_i \tag{3}$$

The following constraint excludes the possibility of assigning two incompatible shifts to a driver.

$$\forall i \forall j \forall p \forall k \forall q x_{ijk} + x_{ipq} \leq SS_{pq}^{jk} + 1 \tag{4}$$

The worktime should not exceed the maximum working time in a week.

$$\forall i \forall w \sum_{j \in D_W^{Week}} \sum_k x_{ijk} wt(j, k) \leq WT \tag{5}$$

A driver can only have the maximum consecutive shifts.

$$\forall i \forall p \sum_{j \in D_p^{WD}} z_{ij} \leq WD \tag{6}$$

At least the required free days need to be given in a month.

$$\forall i \forall m \sum_{j \in D_m^{Mon}} z_{ij} \leq L_m - RD \tag{7}$$

Let define the overtime.

$$\forall i \sum_m (\sum_{j \in D_m^{Mon}} \sum_k x_{ijk} wt(j, k) - ct(i, m)) \leq \pi_i \tag{8}$$

The objective function minimizes the sum of the overtime and the operational cost.

$$\min \sum_i (ET \pi_i + y_i cc(i)) \tag{9}$$

4. HEURISTIC METHOD

The algorithm is a two phase heuristic method based on graph coloring. M. Gamache et. al [7] developed a method, where graph coloring algorithm based tabu search was used for scheduling pilots. This gave the idea for our problem where the rules being discussed above had to be met. Besides the solution had to match the important aspects of the driver rostering. The heuristic has the following steps:

4.1 Initial steps

4.1.1 Set of employees

We can give an estimation for the number of employees for the problem using the worktime of the shifts and the contracts of the workers. Using this, a lower bound for the total overtime can be calculated. The lower bound of the algorithm depends on the number of the employees and the working time. The number of the crew members is the following:

$$C = round(\frac{global\ worktime}{contract\ type})$$

It is important to note that a trivial lower bound for the number of employees can also be given by the number of the shifts on the busiest days, since one crew member can have at most one shift a day.

4.1.2 Days-off patterns

A days off patterns defines the fixed free days for an employee. For scheduling the free days of the employees, in many cases days off patterns are used for the heuristic solutions of crew rostering [6]. We propose a 6-1 days-off pattern, which means that the workers get one day off after every six workdays. This pattern meets both of the minimal free days and the maximal consecutive workdays, and causes only a few exclusion during the graph coloring, and the tabu search will have a relatively big state space.

4.1.3 Graph Building

Let every shift in the planning period be a vertex of a graph. There is an edge between two vertices if the shifts represented by the two linked vertices cannot be performed by the same person. This way every day is a clicque and there are additional edges between the clicques because of the rest time constraint.

4.2 Graph Coloring

During graph coloring each color mean one of the drivers. The initial estimation was usually correct but some input with too short of a worktime might have result that the graph can not be colored with the given number of colors. Therefore the algorithm had to be able to bring in new colors. The DSATUR algorithm gave a proper solution for the coloring, whereabout the article of Brelaz [2] written in 1979 writes in details. In this case the algorithm was modified by using the set of employees estimated during the graph coloring. Consequently a k-coloring was made in a way where adding a new color is possible. The graph coloring gives an initial solution which meet the given constraints and assigne every shift to a driver. As we know the duration of the shifts every person's worktime could be calculated for the initial solution. This way, since the real working time of each driver and their expected worktime were known, an initial cost could be calculated for the solution.

4.3 Tabu Search

The tabu search was introduced by Glover in 1986 and it is one of the most famous local search techniques nowadays [9]. The state space of the tabu search consists every feasible coloring, and every state is a coloring. Let SL be all of the possible solutions of the graph coloring. The objective function had to be minimized in SL. Every $sl \in SL$ possible solution has a neighbourhood denoted by $N(sl)$. The algorithm visits $sl_0, sl_1, ...sl_n$ solutions during the runtime where sl_0 is the initial solution produced by the graph coloring, and $sl_{i+1} \in N(sl_i)$. The next neighbour is chosen by first fit method. This means that the first neighbour solution being better than the actual solution is chosen. Neighbourhoods chosen in one step are stored in the tabu list denoted by TL. The steps in the list are not being performed for a time. The $N(sl)$ set of neighbours of the sl solution can be defined by using the following neighbourhoods:

1. *Recoloring of a vertex:* During this step a new color is given to a vertex of the graph. Meaning that a shift is taken a way from a driver and given to another one who is able to perform the task while keeping the constraints. Shifts are taken away from the drivers having the most overtime and given to the ones having undertime.
2. *Swapping the colors of two vertices:* During the swap, the colors of two vertices are switched, therefore shifts between two drivers are swapped in a way where both of them receives a shift which they can carry out keeping the constraints. Usually this happens between the drivers having the most overtime and the ones having undertime.

After carrying out the vertex recoloring or the vertex swapping, the step is added to the tabu list. Multiple methods can be applied as a stopping criteria. In most cases the practice uses criteria based on iteration number or time bound. In our case a stopping criteria was defined where the algorithm stops if it can not find a better solution within a certain number of iteration.

5. TEST RESULTS

The solutions of the heuristic method were compared to the appropriate integer programming's solutions using a relatively small input (a maximum daily shifts of 50). The algorithm was implemented in Java language and IBM Cplex software was used during solving the appropriate integer programming. Throughout the testing the following, typical basic regulations were used:

- The minimum rest time between two shifts is 12 hours.
- The worktime should be less than 48 hours during one week.
- Employees should have at least 4 free days in one month.

- The number of the maximum consecutive workdays is 6.

In our case every employee had a contract with 8 hours of worktime. Artificially generated and real life inputs were used in the test with one month planning period. The inputs being used during the test had a worktime between 7 and 9 hours, and the duty time between 6 and 10 hours. While solving the problem, the objective function is divided into two parts. The first is the set of people which is estimated in the initial phase, while the minimization of the overtime is considered during the tabu search. The tables below show the under- and overtime by optimal employee number and the maximum amount of daily shifts being 25. The results by the input size being 50 and by real input are listed below as well.

Table 1: Test results on generated inputs

Input	Lower bound	Time IP	Cost IP	Time heur	Cost heur
gen25_1	4807	18.77	4900	3.2	4807
gen25_2	4658	495.2	4691	3.6	4658
gen25_3	1825	154.8	1855	4.1	1825
gen25_4	4371	22.96	4457	3.1	4371
gen25_5	4815	18.11	4911	5.4	4815

The stopping criteria of the algorithm was given to stop when it could not find a better solution after 200 iteration, or there were only drivers with under- or overtime left.

Table 2: Test results on generated inputs

Input	Lower bound	Time IP	Cost IP	Time heur	Cost heur
gen50_1	3359	875.1	3465	5.4	3359
gen50_2	2581	844.8	2581	15.2	2581
gen50_3	3000	1102.34	3047	10.4	3000
gen50_4	1671	1252.78	1696	4.8	1671
gen50_5	1925	1529.56	1960	10.3	1925

The real examples belonged to the database of the Tisza Volan and consisted the local trips, shifts and the drivers in Szeged. The real input consisted approximately 3000 shifts and the employees had a maximum number of 200 during the solution. The results of the real input were tested with appropriate integer programming model, but there was no solution by the 2 % gap. The best results can be seen in the following table.

Table 3: Test results on real life input

Input	Lower bound	Time IP	Cost IP	Time heur	Cost heur
volan	4897	21418.56	5012.87	9.87	4897

The test results show that the task could be solved with the set of estimated number of employees in every case. We obtained that our algorithm is able to handle relatively large inputs, and in the majority of the test cases, it has reached the theoretical lower bound with producing a satisfactory running time, and it produced a feasible solution in all cases.

6. CONCLUSION AND FUTURE WORKS

The important aspects of the crew rostering in the scheduling of bus drivers were introduced above. We proposed a two-phase graph coloring method for crew rostering. In the first step, a graph was built and colored, while in the second step, the graph was recolored with the tabu search method by our algorithm. In the majority of test cases, the algorithm has reached the theoretical lower bound. Our method has been tested with artificially generated and real inputs. For moderate size problems, the results of the new algorithm have been compared to the solutions of the appropriate integer programming model. The heuristic produced a satisfactory running time, reached the lower bount in most cases and returned a feasible solution in all cases. In the future the running time could be improved with paralell programming. In addition, the ability of handling drivers with different kind of contracts with the heuristic would be desired.

7. ACKNOWLEDGEMENT

This work was partially supported by the European Union and the European Social Fund through project Supercomputer, the national virtual lab (grant no.: TAMOP-4.2.2.C-11/1/KONV-2012-0010).
I would also like to thank the Tisza Volan for the test data and the technical consultation.

8. REFERENCES

[1] J. Bartholdi. A guaranteed-accuracy round-off algorithm for cyclic scheduling and set covering. *Operation Research*, 29:501–510, 1981.

[2] D. Brélaz. New methods to color vertices of a graph. 22(4):251–256, Apr. 1979.

[3] S. R. D. Sitompul. Nurse scheduling models: a state-of-the-art review. *J. Soc. Health Syst.*, 7:441–499, 2004.

[4] G. Dantzig. A comment on edies traffic delays at toll booths. *Operations Research*, 2:339–341, 1954.

[5] K. M. M. H. Dowling, D. and D. Sier. Staff rostering at a large international airport. *Annals of Operations Research*, 72:125–147, 1997.

[6] M. Elshafei and H. K. Alfares. A dynamic programming algorithm for days-off scheduling with sequence dependent labor costs. *J. of Scheduling*, 11(2):85–93, Apr. 2008.

[7] M. Gamache, A. Hertz, and J. O. Ouellet. A graph coloring model for a feasibility problem in monthly crew scheduling with preferential bidding. *Comput. Oper. Res.*, 34(8):2384–2395, Aug. 2007.

[8] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[9] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[10] D. Marx. Graph colouring problems and their applications in scheduling, 2003.

[11] K. Nurmi, J. Kyngäs, and G. Post. Driver rostering for bus transit companies. *Engineering Letters*, 19(2):125–132, December 2011.

[12] S. O. T. R. T. Grossman, D. Samuelson. Call centers. *Technical Report, Haskayne Sch. Business,Univ. Calgary*, 1999.

A New Heuristic Approach for the Vehicle and Driver Scheduling Problems

Viktor Árgilán
University of Szeged
Árpád tér 2.
Szeged, Hungary
gilan@jgypk.u-szeged.hu

Balázs Dávid
University of Szeged
Boldogasszony sgt. 6.
Szeged, Hungary
davidb@jgypk.u-szeged.hu

ABSTRACT

Both vehicle and driver scheduling, the two main problems in the optimization of public transport, are NP-hard. Because of this, finding an exact solution for a real-life instance is usually not possible. As a result, research of this field is concentrated on developing efficient heuristic solution methods for these problems. Balogh and Békési give a method to create driver schedules based on an existing vehicle schedule. However, the running time of their algorithm is large. In this paper, we examine the possibility of a heuristic acceleration technique, which is tested both on randomly generated and real data.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Theory

Keywords

Vehicle schedule, driver schedule, sequential approach, combined approach

1. INTRODUCTION

Cost reduction is important for public transport companies, and this can mostly be achieved by reducing operational costs. These costs are derived from the long-term schedules created by the company. These are then executed on a daily basis. The optimization of these schedules is a difficult problem, which is based on the pre-determined timetable of the company. The timetable consists of trips, which are carried out on a fixed route. There are two main problems connected to the trips: vehicle and driver scheduling. Vehicle scheduling gives the duties that different vehicles have to execute, while driver scheduling defines the shifts that driver carry out. Both problems are NP-hard, and literature usually offers two different approaches for their solution: sequential or combined. The operational costs of the company come from the combined costs of the above two problems, which leads to a complex optimization task.

The aim in both cases is to create schedules where are trips are executed exactly once. The sequential approach creates a feasible vehicle schedule as a first step, and then uses this to as a basis for driver schedules, if it is possible. The combined method aims to solve both problems in one step. This would mean that the combined method can provide an optimal solution, but the size of the problem is usually too big to handle. On the other hand, while the solution given by the sequential method most likely will not be optimal, the problem can be handled more easily in two steps. Both the vehicle and driver scheduling problems are NP-hard [2], which makes it difficult to solve real problems with a large number of trips. This lead to a research into different heuristics for these problems, for example in [9, 11]. Using the idea found in [7], Balogh and Békési propose a sequential solution algorithm for the problem in [1]. We introduce a heuristic acceleration technique that can speed up the running time of their algorithm.

In this paper, we first give a overview of the vehicle and driver scheduling problems, then briefly overview their sequential application. After this, we introduce our heuristic method, which we apply to the solution methodology proposed in [1]. We present the efficiency of our heuristic both on random and real instances.

2. THE VEHICLE AND DRIVER SCHEDULING PROBLEM

Related works for the vehicle scheduling problem concern two different groups, depending on the types of available vehicles and the geographical location of the depots. The single depot vehicle scheduling problem (SDVSP) has a single vehicle type, and every vehicle starts at the same geographical location. This problem can be solved in polynomial time. If there are more vehicle types or starting geographical locations, then the problem has at least 2 depots. This case is called a multiple depot vehicle scheduling problem, which is NP-hard [2]. Vehicle scheduling problems arising in the real world usually belong to this group.

The vehicle scheduling problem assigns vehicles to the timetabled trips, so that the following criteria are met:

- Every trips has to be executed exactly once.
- Every trip is compatible with the depot and vehicle type of its assigned vehicle.
- Each trip t_i and t_j assigned to the same vehicle have to be compatible. Two trips t_i and t_j are compatible, if they can be served by the same vehicle type, and there is enough time between the arrival time of t_i and the departure time of t_j for the vehicle to travel from the arrival location of t_i to the departure location of t_j. Such a vehicle journey (without executing a timetabled trip) is called a deadhead trip.
- The problem wants to minimize the arising cost, which is usually the combination of two main components:
  - a daily cost for each vehicle, if it leaves its depot to execute a trip,
  - and a travel cost for each vehicle, which is proportional to the travelled distance.

Many different mathematical models can be given for the vehicle scheduling problem. For an overview of the possible representations, refer to [3].

Driver scheduling is also a main problem in public transit, because personell give a high percent of the operational costs of a company.

The problem considers a set of driver tasks (which are mainly the timetabled trips), and the aim is to return driver shifts, which represent a sequence of the tasks. Similarly to vehicle scheduling, every task has be assigned to a shift exactly once, and the tasks belonging to the same shift have to be compatible. The most important constraints for driver schedules are defined by the European Union, but companies usually define their own set of rules also. The most important of these constraints are:

- maximum working hours,
- total length of the daily break(s),
- and the number and length of the breaks.

Mathematical formulation of the problem is usually given by set partitioning or a set covering relaxation. These problems are known to be NP-hard [6]. For an overview of the existing models and methods, refer to [5].

3. SOLUTION WITH COLUMN GENERATION

Balogh and Békési give a sequential solution method for vehicle and driver scheduling in [1]. Their methodology is strongly based on the ideas of Gintner et al. [7] and Steinzen et al. [10]. This method considers an arbitrarily given vehicle schedule, and uses this as an initial step to construct the driver schedule.

As we mentioned earlier, set covering is one of the usual approaches to model this problem. We give a 0-1 integer programming model for this problem, which is originally presented in [1]:

$$\sum_{d \in D} \sum_{k \in K^d} c_k^d x_k^d \tag{1}$$

$$\sum_{d \in D} \sum_{k \in K^d(t)} x_k^d \geq 1 for \forall t \in T \tag{2}$$

$$x_k^d \in 0, 1, d \in D, k \in K^d \tag{3}$$

where

- T is the set of timetabled trips,
- D is the set of depots,
- K^d is the set of possible driver schedules from depot d,
- $K^d(t)$ is the set of driver schedules covering trip t from depot d,
- c_k^d is the cost of schedule k from depot d.

The main problem of is model is that the number of possible driver schedules is high, and generating all of them is usually not possible. To address this problem, the column generation method in [4] is applied, generating new columns using the time space network model in [8]. However, the running time of this method is still large.

4. THE PROPOSED HEURISTIC

As we mentioned at the end of the last section, running time of the above method is still an issue: it can be as big as several days. This running time might not be acceptable for a solution method applied in practice. Our aim is to develop a method that decreases the size of the problem itself, and results in a model with a smaller number of variables, which is easier to solve.

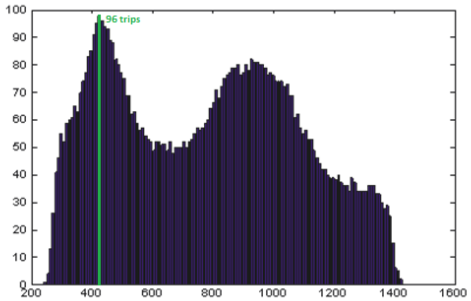


Figure 1: Traffic load of a sample problem

The decrease in size is achieved by the following method: consider the timeline of the daily traffic load as our planning period, which is defined by the timetabled trips. This planning period starts with the earliest trip departure time,

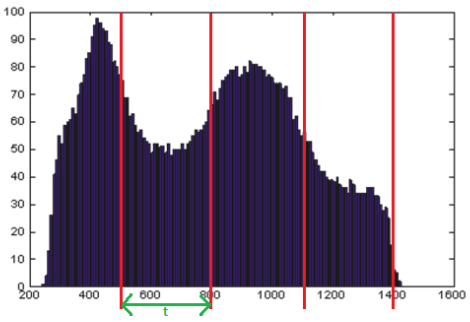


Figure 2: An interval division of the problem

and ends with the latest trip arrival time. An example for this can be seen in Figure 1.

We divide the planning period into smaller, fixed-length intervals. The heuristic will consider these sequentially, starting with the earliest interval. An example for such a division can be seen in Figure 2.

The aim of the heuristic is to decrease the number of trips in the input, which is achieved by merging more trips together, which are then considered as a single trip. First, as we mentioned earlier, we divide our planning period into smaller intervals of equal size t. The number k of dividing points needed for this can be calculated by

$$k = \frac{(max\{arrival\_time_{trip}\} - min\{departure\_time_{trip}\})}{t}$$

Using these dividing points, we partition the trips of the input into sets S_i $(1 \leq i \leq k)$. A set S_i contains the trips that depart in the time interval defined by its dividing points. The set S_k will contain trips starting after the last dividing point k. It is possible, that there are only arriving trips, and no trips are departing after time k, which will result in $S_k = \emptyset$. Naturally,

$$S_i \bigcap S_j = \emptyset \quad \text{if} \quad i \neq j$$

and

$$\bigcup_{i=1}^{k} S_i = S$$

Our heuristic aims to solve the vehicle scheduling problem sequentially for every interval S_i. An arbitrary solution method can be used for this. As the newly merged trips act more like the pieces of a shift, each trip t is assigned a number db_t, which represents the number of driver breaks that can be carried out between its sub-trips. Naturally, this value will be 0 for normal trips, but it can be positive for merged trips.

The vehicle scheduling problem is solved for the trips belonging to set S_1. This will result in a series of vehicle duties, each such duty consisting of at least one trip. The trips of a duty will be merged together to form a new trip. The departure time and location of this new trip will correspond to that of the first trip of the duty, and the arrival time and location will be defined by the last trip of the duty.

As one of the most important driver rules is the assignment of breaks, we also have to check if there is enough idle time between the tasks of a vehicle duty to assign a break. If there is such a possibility, then $db_t = 1$ for a merged trip t. These newly constructed trips will be added to a set M.

The heuristic continues sequentially with the sets S_i $(1 < i \leq k)$. For each step i, the vehicle scheduling is carried out on the set $S_i \cup M$. After the scheduling is completed, a newly merged trip t defined by a driver duty will have $db_t = \sum_{t'} db'_t$, where t' are the trips of the duty. If there is enough time for and additional driver break in the new duty, db_t is further increased by 1. Set M will be then updated to only contain the trips defined by this new duty.

We explain the algorithm on the example given in Figure 3, where the set of trips is given for the problem over a timeline.

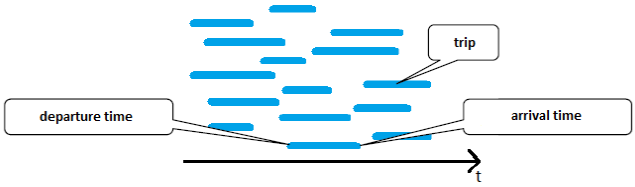


Figure 3: Input of the problem

Suppose that the first dividing point for the heuristic will be at k_1. We will only consider trips starting before k_1, as can be seen in Figure 4.

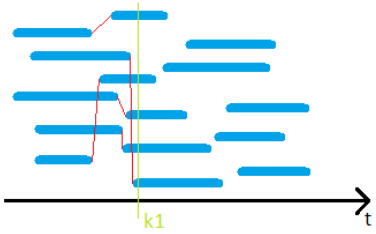


Figure 4: First iteration of the algorithm

The vehicle scheduling problem is solved for these trips, and some of them are merged together, forming the new trips of set M. The merged trips that come as a result of the first phase can be seen in Figure 5

Figure 5: Merged trips after the first iteration

Using these newly created trips, we now advance to the second iteration, which has the dividing point k_2. This iteration

considers the original trips departing between k_1 and k_2, together with the newly merged trips of set M. This can be seen in Figure 6.

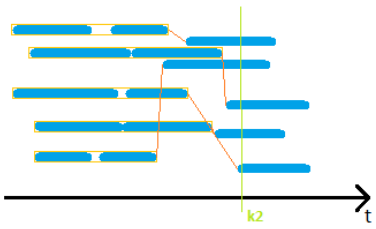


Figure 6: Second iteration of the algorithm

The vehicle scheduling problem is once again solved for the trips, and the resulting vehicle duties are merged into single trips. The results of this iteration can be seen in Figure 7.

Figure 7: Merged trips after the second iteration

The results in Figure 7 also present the consideration of driver breaks. One of the merged trips has enough time for a break (colored green in the figure). As a result, the possibility of the break is noted by the assigned variable.

The pseudo-code of the algorithm can be seen in Algorithm 1. The algorithm first calculates the dividing points and assigns trips to every set S_i, then sequentially solves the vehicle scheduling problem for these sets. The possible number of driver breaks is calculated using the method described above.

5. COMPUTATIONAL RESULTS

We tested the above size reduction heuristic on both randomly generated and real-life data provided by Tisza Volán Zrt, the bus company of the city Szeged, Hungary. The data we used was the same that Balogh and Békési used for testing in [1]. The tests run on the following computer:

- Processor: Intel Core I7 X980 3.33 Ghz,
- Memory: 12Gb,
- Operation System: Microsoft Windows 7 Ultimate 64bit

Table 1: Test results of heuristic method

Problem	Trips	Vehicles	Drivers	Running time
#1	100	5	9	69min
#2	900	39	65	264min
#3	2711	105	231	4989min

Algorithm 1 Heuristic Size Reduction.

```
1: procedure SizeReduce(S, t, B, p)
2:     M=∅
3:     k = ((max{arrival_time_trip} − min{departure_time_trip}) / t)
4:     for i = 0 to k do
5:         if i ∗ t < arrival_time_trip ≤ (i + 1 ∗ t) then
6:             db_t rip = 0
7:             trip ⇒ S_i
8:         end if
9:     end for
10:    for i = 0 to k do
11:        X = M ⋃ S_i
12:        M′ = solve vehicle scheduling for X
13:        for j = 1 to m' do
14:            Merge tasks on duty j to a single trip
15:            Calculate db_j
16:        end for
17:        M = M′
18:    end for
19: end procedure
```

Table 1 presents our results, while we present their solutions in Table 2 as a comparison. The driver rules we considered were the ones in practice at the transportation company.

Table 2: Test results from [1]

Problem	Trips	Vehicles	Drivers	Running time
#1	100	5	9	81min
#2	900	38	62	362min
#3	2711	103	224	6616min

It can be seen in the last column that the running time of the new method is smaller, even for large examples. Our new heuristic is faster for every instance than the old one. However, it can also be seen in the third and fourth column that the number of vehicles and drivers given by our method are higher in every instance, which results in a larger cost. As we expected, the heuristic managed to speed up the solution process significantly, but it produces solutions with a higher cost.

6. CONCLUSION AND FUTURE WORKS

This is a heuristic acceleration of an earlier method. (of Balogh and Bekesi). It produces less running time and a bit higher cost, but the gap between the value of our solution and the optimal value is small. As a future work, we plan to apply a new matching based method instead of greedy heuristic. And we plan to examine other heuristics as well.

7. ACKNOWLEDGEMENT

This work was partially supported by the European Union and the European Social Fund through project Supercomputer, the national virtual lab (grant no.: TAMOP-4.2.2.C-11/1/KONV-2012-0010).

8. REFERENCES

[1] J. Balogh and J. Békési. Driver scheduling for vehicle schedules using a set covering approach: a case study. *Abstracts of the ICAI 2014 - 9th International Conference on Applied Informatics*, to appear.

[2] A. A. Bertossi, P. Carraresi, and G. Gallo. On some matching problems arising in vehicle scheduling models. *Networks*, 17(3):271–281, 1987.

[3] S. Bunte and N. Kliewer. An overview on vehicle scheduling models. *Journal of Public Transport*, 1(4):299–317, 2009.

[4] M. Desrochers and F. Soumis. A column generation approach to the urban transit crew scheduling problem. *Transportation Science*, 23(1):1–13, 1989.

[5] A. T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *European journal of operational research*, 153(1):3–27, 2004.

[6] M. Fischetti, A. Lodi, S. Martello, and P. Toth. A polyhedral approach to simplified crew scheduling and vehicle scheduling problems. *Management Science*, 47(6):833–850, 2001.

[7] V. Gintner, N. Kliewer, and L. Suhl. A crew scheduling approach for public transit enhanced with aspects from vehicle scheduling. In *Computer-aided Systems in Public Transport*, pages 25–42. Springer, 2008.

[8] N. Kliewer, T. Mellouli, and L. Suhl. A time–space network based exact optimization model for multi-depot bus scheduling. *European journal of operational research*, 175(3):1616–1627, 2006.

[9] A.-S. Pepin, G. Desaulniers, A. Hertz, and D. Huisman. A comparison of five heuristics for the multiple depot vehicle scheduling problem. *Journal of Scheduling*, 12(1):17–30, 2009.

[10] I. Steinzen, V. Gintner, L. Suhl, and N. Kliewer. A time-space network approach for the integrated vehicle-and crew-scheduling problem with multiple depots. *Transportation Science*, 44(3):367–382, 2010.

[11] A. Tóth and M. Krész. An efficient solution approach for real-world driver scheduling problems in urban bus transportation. *Central European Journal of Operations Research*, 21(1):75–94, 2013.

Multi-population Firefly Algorithm

Jani Dugonik
Faculty of Electrical Engineering and Computer Science
University of Maribor
2000 Maribor, Slovenia
jani.dugonik@um.si

Iztok Fister
Faculty of Electrical Engineering and Computer Science
University of Maribor
2000 Maribor, Slovenia
iztok.fister@um.si

ABSTRACT

This paper proposes a meta-heuristic Multi-Population Firefly Algorithm (MPFA) for single-modal optimization using two multi-population models, i.e., one is based on the island model while the other on the mainland-island model. The unique characteristics of each sub-population is evolved independently and the diversity of the entire population is effectively increased. Sub-populations communicate with each other to exchange information in order to expand the search range of the entire population. In line with this, each sub-population explores a specific part of the search space and contributes its part for exploring the global search space. The main goal of this paper was to analyze the performance between MPFA and the original Firefly Algorithm (FA). Experiments were performed on a CEC 2014 benchmark suite consisting of 16 single-objective functions and the obtained results show improvements in most of them.

Keywords

swarm intelligence, island model, multi-population, firefly algorithm

1. INTRODUCTION

An optimization problem is defined as a quadruple $OP = \langle I, S, f, goal \rangle$, where I denotes a set of all input instances $\mathbf{x} \in I$ in the form of $\mathbf{x} = \{x_1, x_2, ..., x_D\}$ where D is the dimensionality of the problem, $S(x)$ a set of all feasible solutions $\mathbf{y} = S(\mathbf{x})$, f is the objective function estimating the feasible solution, and *goal* determines an optimal criteria that can be either the minimum or maximum value of the objective function. A task of the optimization algorithm is to find the value of $\mathbf{y}^*$ that minimizes or maximizes (depending on the *goal*) the value of the objective function $f(\mathbf{y})$. The domain values of input variables $x_i \in [lb_i, ub_i]$ are limited by their lower lb_i and upper ub_i bounds.

Nature has evolved over millions of years and has found perfect solutions to almost all encountered problems. We can thus learn the success of problem-solving from nature and develop nature-inspired heuristic and/or meta-heuristic algorithms in order to solve optimization problems with which developers are confronted today. Two sources from the nature have particularly inspired developers of new optimization algorithms, i.e., Darwinian evolution and the behavior of social living insects (e.g., ants, bees, termites, etc.) and other creatures (e.g., birds, dolphins, fireflies, etc.). As a result, two main classes of nature-inspired algorithms exist nowadays, i.e., evolutionary algorithms (EA) [3] and swarm intelligence (SI)-based algorithms [1]. While the former already came in mature years, the latter has experienced rapid development. Almost every day, we are witnessing the birth of a new SI-based algorithm.

One of the younger members of the SI-based algorithms is the Firefly Algorithm (FA) as proposed by Yang in [11]. FA is inspired by a chemical phenomenon bioluminiscence needed by natural fireflies to find their prey, on the one hand, and to attract their mating partners, on the other hand. This algorithm belongs to a class of population-based algorithms [5, 4, 11, 12]. Population-based approaches maintain and improve multiple candidate solutions, often using population characteristics to guide the search. Two major components of any population-based search algorithms are exploitation and exploration. Exploitation refers to searching within neighborhoods for the best solutions and ensures that the solutions can converge into optimality, while the exploration uses randomization in order to avoid the solutions being trapped within a local optima and while at the same time increasing the diversity of the solutions. A good combination of these two components may usually ensure that the global optimality is achieved [11].

One of the possible ways of how to improve the exploration and exploitation in the original FA algorithm can be splitting the FA population into more sub-populations (so-called multi-populations). For instance, the authors in [10] presented a multi-population FA for correlated data routing within underwater wireless sensor networks. They designed three kinds of fireflies and their coordination rules in order to improve the adaptabilities of building, selecting, and optimizing a routing path. Groups are represented as sub-populations, where each sub-population conducts its own optimization in order to improve the convergence speed and solution precision of the algorithm. The author in [13] analyzed the ability of a multi-population differential evolution to locate all optima of a multi-modal function. The explo-

ration was ensured by the controled initialization of sub-populations while a particular differential evolution algorithm ensured the exploitation. Sub-populations were communicating via archive where all located optima were stored. The authors in [8] used an evolutionary algorithm on punctuated equilibria. The theory of punctuated equilibria calls for the population to be split into several sub-populations. These sub-populations have isolated evolution (computation) and scattered with migration (communication).

This paper aimed to evaluate whether it is possible to outperform the performance of the original FA algorithm by splitting its population into more sub-populations. The proposed multi-population FA (MPFA) supports two multi-population models, i.e., Island [13] and Mainland-Island [9]. In these multi-population models, sub-populations evolve independently, thus the unique characteristics of each sub-population can be effectively maintained, and the diversity of the entire population is effectively increased. Sub-populations communicate with each other by exchanging information in order to expand the search range of the entire population. The search technique based on a population has proved to have good ability regarding global searching and can find a set of solutions in one-shot operation. The proposed multi-population FAs were compared with the original FA on single-objective CEC 2014 benchmark functions [7].

The remainder of this paper is organized as follows. In Section 2 the original FA will be presented. In Section 3 a multi-population FA with two multi-population models are presented in detail. Section 4 presents experiments, where the number of tests were performed in order to compare the proposed approach with the original FA. The paper is concluded with Section 5, where our opinion on the obtained results is given.

2. THE ORIGINAL FIREFLY ALGORITHM

The Firefly Algorithm (FA) [11] has two fundamental factors: light intensity and attractiveness. Light intensity I reflects the firefly location and determines its direction of movement, while the degree of attractiveness determines the distance that a firefly has moved. Both factors are constantly updated in order to achieve the objective of the optimization.

For simplicity, the author in [11] used the following three idealized rules:

- All fireflies are unisex so that one firefly will be attracted to other fireflies regardless of their sex.
- Attractiveness is proportional to their brightness and for any two flashing fireflies, the dimmer one will move towards the brighter one. They both decrease as their distance increases. If there is no brighter one, it will move randomly.
- The brightness of a firefly is affected or determined by the landscape of the objective function.

Based on these three rules, the basic steps of the firefly algorithm (FA) can be summarized as the pseudo-code shown in Algorithm 1. Light intensity of a firefly is defined as:

Algorithm 1 Firefly Algorithm

```
Objective function f(x), x = (x_1, ..., x_D)^T
Generate initial population of fireflies x_i (i=1,2,...,Np)
Light intensity I_i at x_i is determined by f(x_i)
Define light absorption coefficient γ
while (t < G_max) do
    for i=1 to n fireflies do
        for j=1 to n fireflies do
            if (I_j > I_i) then
                Move firefly i towards firefly j Eq. (3)
            end if
            Evaluate new solution and update light intensity Eq. (1)
        end for
        Rank fireflies and find the current best
    end for
    Post-process results and visualization
end while
```

$$I(r) = I_0 \cdot e^{-\gamma \cdot r^2} \tag{1}$$

where I_0 is the original light intensity at the location of $r = 0$, γ is the light absorption coefficient and r is the distance between two fireflies. The distance between any two fireflies i and j at x_i and x_j can be expressed as Cartesian distance $r_{ij} = ||x_i - x_j||$. As firefly attractiveness is proportional to the light intensity, we can define the attractiveness of a firefly using the following equation:

$$\beta(r) = \beta_0 e^{-\gamma \cdot r^2} \tag{2}$$

where β_0 is their attractiveness at $r = 0$. Firefly i that is attracted to another more attractive firefly j is determined by:

$$x_i = \beta_0 e^{-\gamma \cdot r^2} \cdot (x_j - x_i) + \alpha \cdot \varepsilon_i \tag{3}$$

which is randomized with the vector of random variable ε_i, being drawn from a Gaussian distribution, and step factor $\alpha \in [0, 1]$.

3. THE MULTI-POPULATION FIREFLY ALGORITHM

The multi-population firefly algorithm (MPFA) can be summarized in the pseudo-code as shown in Algorithm 2. MPFA will consider there to be an overall population P of Np fireflies (individuals) that is split into N sub-populations $P_1, P_2, ...P_N$. Each sub-population has Nsp individuals and the number of N sub-populations is calculated with the following equation:

$$N = \frac{Np}{Nsp} \tag{4}$$

For sub-populations to communicate with each other, the magnitude and frequency of that communication are necessary. These two parameters determine the amount of isolation and interaction between sub-populations. Periods of isolated evolution are referred to as *epoch*, with migration

Algorithm 2 Multi-Population Firefly Algorithm

Objective function $f(\mathbf{x})$, $\mathbf{x} = (x_1, ..., x_D)^T$
Calculate number of sub-populations (N)
for all $n \in N$ **do**
    Generate initial sub-population P_n of fireflies $\mathbf{x}_i$ (i=1,2,...,Np)
end for
Light intensity I_i at $\mathbf{x}_i$ is determined by $f(\mathbf{x}_i)$
Define light absorption coefficient γ
for e=1 to $\frac{G_{max}}{epoch}$ **do**
    for all $n \in N$ **do**
        for g=1 to $epoch$ **do**
            for i=1 to n $fireflies$ **do**
                for j=1 to n $fireflies$ **do**
                    if $(I_j > I_i)$ **then**
                        Move firefly i towards firefly j
                    end if
                    Evaluate new solutions
                    Update light intensity
                end for
                Rank fireflies and find the current best
            end for
        end for
    end for
    Migrate fireflies
end for
Find the best firefly from all sub-populations
Post-process results and visualization

occurring at the end of each epoch except the last. The length of the epoch determines the frequency of interaction and is usually specified by a number of generations ($epoch$) that P_n evolves in isolation. During the epoch, each sub-population executes a sequential FA for $epoch$ independently. At the end of each epoch, individuals are migrated between sub-populations. There are many various migration strategies in multi-population models. The following two models are described and used in this paper: island and mainland-island model.

3.1 Island Model

The island Model Firefly Algorithm (MPFA-In, where n determines the number of sub-populations) consist of islands, where islands are referred to as sub-populations. When each sub-population is executed a sequential FA for $epoch$ generations, individuals are migrated between sub-populations, as shown in Algorithm 3. The magnitude of the communication is defined, for instance, as $N_m = 25\%$. Then, N_m percent of migrants are chosen for each sub-population which were exchanged with other sub-populations, as shown in Figure 1. Let us assume two sub-populations P_1 and P_2 with $Nsp = 10$ and $N_m = 20\%$ are defined. Then, two individuals from sub-population P_1 are exchanged with two individuals in sub-population P_2. Thus, the sizes of the sub-populations remain the same. After the algorithm reaches the termination criteria, the best individual is taken from all sub-populations.

3.2 Mainland-Island Model

The Mainland-Island Model Firefly Algorithm (MPFA-Mn, where n determines the number of sub-populations) consist of mainland and islands, where mainland and islands are referred to as sub-populations. When each sub-population has executed a sequential FA for G_i generations, individuals are migrated from sub-populations $P_2, ..., P_N$ to sub-population P_1, as shown in Algorithm 4. Let us assume two sub-populations P_1 and P_2 with $Nsp = 10$ and $N_m = 20\%$ are defined. Then, two individuals per sub-population $P2, ..., P_N$ are moved to sub-population P_1, as shown in Figure 2. At the end of migration, sub-population P_1 is sorted according to the fitness values of migrated individuals. In order to keep the size of sub-population P_1 the same, the top Nsp individuals are retained, while the others are discarded. After the algorithm has reached the terminating criteria, the best individual was taken from the sub-population P_1 (mainland).

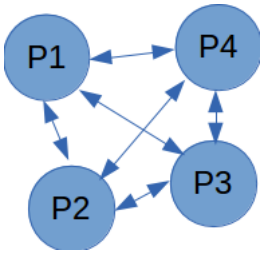


Figure 1: Island Model

Algorithm 3 Multi-Population Firefly Algorithm with Island Model - Migration

Get the number of migrants N_m to migrate
for $i = 1$ to N **do**
    Choose N_m individuals from P_i that are mutually different and save them to the matrix M
end for
Mutually exchange individuals that are defined in matrix M

Algorithm 4 Multi-Population Firefly Algorithm with Mainland-Island Model - Migration

Get the number of migrants N_m to migrate
for $i = 2$ to N **do**
    Choose N_m individuals from P_i that are mutually different and save them to the matrix M
end for
Copy chosen individuals from sub-populations $P_2, ..., P_N$ to sub-population P_1
Sort individuals in P_1 by light intensity
Keep top Nsp individuals, remove others, so the size of the sub-population P_1 remains the same

4. EXPERIMENTS AND RESULTS

The goal of this experimental work was to show that MPFA can outperform the results of the original FA algorithm. In our experiments, the results of the original FA were compared with the results of the following MPFA: MPFA-I2 and MPFA-I4 (i.e., MPFA with island model using two or four sub-populations), and MPFA-M2 and MPFA-M4 (i.e., MPFA with mainland-island model using two or four sub-populations). Additionally, the following three population models were used during tests, i.e., small with an original

Table 1: Comparison between FA algorithms for population model Np=100 and D=10

Func.	FA	MPFA-I2	MPFA-I4	MPFA-M2	MPFA-M4
1	1.0243e+06 ± 2.0147e+06	7.6582e+05 ± 3.4941e+06	6.4129+e05 ± 2.3202e+06	7.6582e+05 ± 3.4929e+06	**5.0199e+05** ± 9.3012e+05
2	1.3862e+04 ± 8.2668e+03	6.1151e+03 ± 4.8489e+03	**5.4070+e03** ± 4.0991e+03	6.1151e+03 ± 4.8489e+03	6.5121e+03 ± 8.0172e+03
3	2.0877e+04 ± 1.9394e+04	2.2167e+04 ± 2.0586e+04	**1.6543e+04** ± 1.3981e+04	2.2167e+04 ± 2.0586e+04	2.3253e+04 ± 2.6483e+04
4	7.7409e+00 ± 1.6024e+01	6.9890e+00 ± 1.1157e+01	7.0024e+00 ± 2.9768e+01	7.1320e+00 ± 1.0983e+01	**6.8717e+00** ± 8.8252e+00
5	2.0107e+01 ± 0.0704e+00	2.0059e+01 ± 0.0405e+00	**2.0035e+01** ± 0.0260e+00	2.0059e+01 ± 0.0405e+00	2.0044e+01 ± 0.0380e+00
6	8.4299e+00 ± 4.0688e+00	8.9432e+00 ± 2.5879e+00	**8.3829e+00** ± 2.6099e+00	8.9432e+00 ± 2.5879e+00	9.6953e+00 ± 3.2605e+00
7	5.4650e+00 ± 6.2455e+00	9.9181e+00 ± 1.0126e+01	5.8841e+00 ± 4.6803e+00	9.9181e+00 ± 1.0126e+01	**5.3097e+00** ± 5.9105e+00
8	**1.6925e+01** ± 1.5711e+01	2.3883e+01 ± 2.3057e+01	2.1892e+01 ± 1.4542e+01	2.3883e+01 ± 2.3057e+01	3.0847e+01 ± 2.5583e+01
9	**1.6924e+01** ± 1.2980e+01	1.8907e+01 ± 2.0379e+01	1.9903e+01 ± 1.6285e+01	1.8907e+01 ± 2.0379e+01	3.0847e+01 ± 2.7831e+01
10	1.1733e+03 ± 9.0738e+02	1.3110e+03 ± 1.1059e+03	**1.0351e+03** ± 8.9937e+02	1.3110e+03 ± 1.1157e+03	1.2901e+03 ± 1.4476e+03
11	1.1434e+03 ± 9.9968e+02	1.2193e+03 ± 1.0618e+03	**9.1012e+02** ± 8.3705e+02	1.2193e+03 ± 1.0618e+03	1.3165e+03 ± 1.1339e+03
12	0.4707e+00 ± 0.9716e+00	0.3272e+00 ± 1.0509e+00	**0.1942e+00** ± 0.4033e+00	0.3272e+00 ± 1.0509e+00	0.4343e+00 ± 1.7646e+00
13	0.3973e+00 ± 0.3133e+00	0.3949e+00 ± 0.3414e+00	**0.3137e+00** ± 0.2133e+00	0.3949e+00 ± 0.3414e+00	0.3578e+00 ± 0.3217e+00
14	0.3618e+00 ± 0.2481e+00	0.3578e+00 ± 0.2386e+00	**0.3260e+00** ± 0.1413e+00	0.3578e+00 ± 0.2386e+00	0.3378e+00 ± 0.2627e+00
15	**1.3559e+01** ± 1.5484e+01	1.7627e+01 ± 1.5247e+01	2.1337e+01 ± 1.8450e+01	1.7627e+01 ± 1.5247e+01	2.8024e+01 ± 2.6350e+01
16	3.7235e+00 ± 0.5893e+00	3.8952e+00 ± 0.7525e+00	**3.7126e+00** ± 0.7731e+00	3.8952e+00 ± 0.7525e+00	4.0390e+00 ± 0.7963e+00

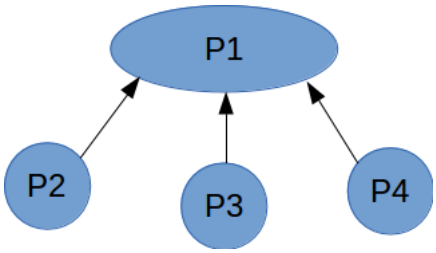


Figure 2: Mainland-Island Model

population size of $Np = 100$, medium with $Np = 200$ and large with $Np = 400$. The original population size was divided between sub-population according to Eq. (4). The same number of generations $G_{max} = 1,000$ was used for each sub-population for each algorithm.

The FA algorithms used the following parameter settings. The maximum number of evaluations was set as $MAX\_FEs = 10,000 \cdot D$. The randomized factor was fixed at $\alpha = 0.5$, the lights absorption at $\gamma = 1$, and the attractiveness at the beginning $\beta_0 = 1$. In each generation the randomized factor α was updated by the following equation: $\alpha = \alpha \cdot (1 - \delta)$, where $\delta = 1.0 - (\frac{10^4}{0.9})^{\frac{1}{G}}$ [5]. For the MPFA algorithms, some additional parameters were used, like the number of epochs as $epoch = 100$, and migration probability $N_m = 25\%$. Tests were conducted on all three population models using five FA algorithms, i.e., FA, MPFA-I2, MPFA-I4, MPFA-M2, and MPFA-M4. In summary, 15 tests were performed, in which 51 independent runs were performed.

All algorithms were tested on the 16 single-objective unimodal and simple multi-modal CEC 2014 benchmark functions [7]. For uni-modal functions the convexity guarantees that the final optimal solution is also the global optimum. The global maximum was measured according to an error value ER. The error value for each function is calculated by subtracting the value of global optima from the obtained value according to the following equation:

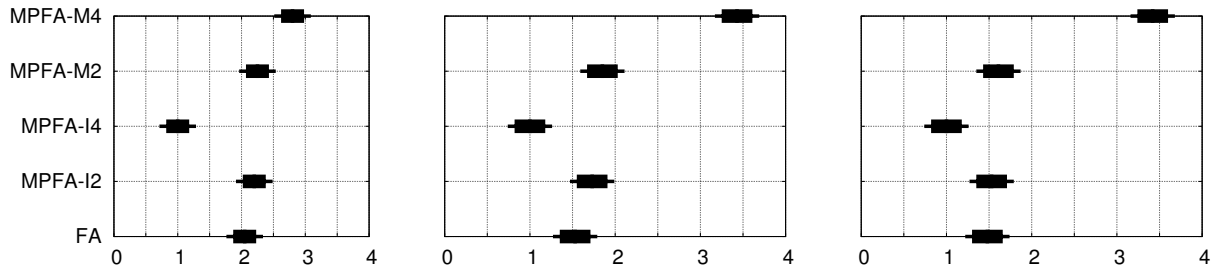
$$ER_i = f_i(\mathbf{x}) - f_i(x^*), \text{ [7]} \quad (5)$$

where i is the function number, $f_i(x)$ is the obtained value, and $f_i(x*) = 100 \cdot i$ is the value of global optima for i-th function. Note that error values smaller than 10^{-8} were taken as zero. In order to limit a search space, each problem variable can capture the value from the range $x_i \in [-100, 100]^D$, where values -100 and 100 represent its upper and lower bounds. The dimensionality of all problems was limited to $D = 10$.

The results of the mentioned FA algorithms are illustrated in Table 1. The results for all population models (i.e., small, medium, large) were obtained. Small population model ($Np = 100$) gave the best results, and due to the paper's length limitation only these results are presented in this table. In line with this, the original FA algorithm is compared with the MPFA-I2 and MPFA-M2 using two sub-populations of size $Nsp = 50$, and MPFA-I4 and MPFA-M4 using four sub-populations of size $Nsp = 25$. The table presents mean and standard deviation over 51 independent runs for each algorithm. The results in Table 1 show that MPFA-I2 as well as MPFA-M2 outperformed the original FA on 7 out of 16 test functions, MPFA-M4 on 8 out of 16 test functions and MPFA-I4 on 12 out of 16 test functions. The best results were obtained with MPFA-I4 which outperformed the other MPFAs and the original FA on 10 out of 16 functions.

In order to evaluate the quality of the results statistically, Friedman tests [6] were conducted that compare the average ranks of the compared algorithms. Thus, a null-hypothesis is placed that states: two algorithms are equivalent and therefore, their ranks should be equal. When the null-hypothesis is rejected, the Bonferroni-Dunn test [2] is performed. In this test, the critical difference is calculated between the average ranks of those two algorithms. If the statistical difference is higher than the critical difference, the algorithms are significantly different.

Three Friedman tests were performed regarding data obtained by optimizing 16 functions of three different population sizes according to five measures. As a result, each algorithm during the tests (also the classifier) was compared with regard to the 48 functions x 5 measurements this means, 240 different variables. The tests were conducted at a significance level of 0.05. The results of the Friedman non-parametric test can be seen in Figure 3 that is divided into three diagrams. Each diagram shows the ranks and confidence intervals (critical differences) for the algorithms under



(a) *Small* ($Np = 100$) (b) *Medium* ($Np = 200$) (c) *Large* ($Np = 400$)

Figure 3: Results of the Friedman non-parametric test

consideration with regard to the dimensions of the functions. Note that the significant difference between the two algorithms is observed if their confidence intervals denoted as thickened lines in Fig. 3 do not overlap.

As can be seen from Fig. 3, the MPFA-I4 outperformed the results of the original FA as well as the other algorithms using all three observed population size model significantly. The MPFA-M4 achieved the results that are significantly worse than the results of the other FA algorithms. The performances of the other three algorithms were comparable with each other.

5. CONCLUSION

In this paper, we proposed MPFA using two multi-population models, i.e., Island and Mainland-Island Models. The proposed MPFAs were compared with the original FA algorithm using three different population size models (i.e., small, medium, large) by solving the CEC-14 benchmark function suite. Based on the obtained results, we can see that the most promising results were obtained by the MPFA-I4. This fact encourage us to continue with the experiments of multi-population FA in the future.

The future work could be especially focused on the migration probability and dimension of the problem. Current migration probability was fixed for all multi-population models, but the migration probability can be modified or even adapted during the algorithm run. On the other hand, all the performed tests were done on small dimensions of the problem. Thus, the algorithm with few number of evaluations and larger population sizes did not reach the migration phase at all. With larger dimensions, the number of evaluations would be increased and the multi-population strategies could perform even better.

6. REFERENCES

[1] C. Blum and D. Merkle. *Swarm Intelligence*. Springer-Verlag, Berlin, 2008.

[2] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[3] A. Eiben and J. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, Berlin, 2003.

[4] I. Fister, I. Fister Jr, X.-S. Yang, and J. Brest. A comprehensive review of firefly algorithms. *Swarm and Evolutionary Computation*, pages 34–46, 2013.

[5] I. Fister, X.-S. Yang, I. Fister, and J. Brest. Memetic firefly algorithm for combinatorial optimization. In *Proceedings of the Fifth International Conference on Bioinspired Optimization Methods and their Applications - BIOMA 2012*, pages 75–86, 2012.

[6] Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11:86–92, March 1940.

[7] J. J. Liang, B.-Y. Qu, and P. N. Suganthan. Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization. Technical report, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, 2013.

[8] W. N. Martin, J. Lienig, and J. P. Cohoon. *Island (migration) models: evolutionary algorithms based on punctuated equilibria*. Handbook of Evolutionary Computation, Oxford University Press, 1997.

[9] M. Slatkin and L. Voelm. FST in a Hierarchical Island Model. *Genetics*, 1991.

[10] M. Xu and G. Liu. A Multipopulation Firefly Algorithm for Correlated Data Routing in Underwater Wireless Sensor Networks. *International Journal of Distributed Sensor Networks*, 2013.

[11] X.-S. Yang. *Nature-Inspired Metaheuristic Algorithms*. Luniver Press, 2008.

[12] X.-S. Yang. Multiobjective firefly algorithm for continuous optimization. *Engineering with Computers*, pages 175–184, 2013.

[13] D. Zaharie. A Multipopulation Differential Evolution Algorithm for Multimodal Optimization. In *Proceedings of Mendel 2004 - 10th International Conference on Soft Computing*, pages 17–22, 2004.

Monte Carlo Path Tracing with OpenCL

Andrej Bukošek
FRI, University of Ljubljana
Večna pot 113,
1000 Ljubljana, Slovenia
andrej.bukosek@gmail.com

ABSTRACT

We introduce an interactive Monte Carlo path tracer that uses the OpenCL framework. A path tracer draws a photorealistic image of a 3D scene by simulating physical effects of light. Interactivity enables the user to move around the scene in real time, while OpenCL makes it possible to run the same code on either CPU or GPU architectures. The results presented here are a starting point for our further work on path tracing on heterogeneous architectures.

Categories and Subject Descriptors

I.3.7 [**Computer Graphics**]: Raytracing, Animation; D.1.3 [**Software**]: Parallel programming

General Terms

Algorithms, Performance

Keywords

path tracing, OpenCL, rendering, computer graphics

1. INTRODUCTION

The path tracing algorithm draws a photorealistic image of a 3D scene by simulating the bouncing of light around the scene in a physically-correct way [2]. Rendering consists of shooting rays of light from the camera into the scene, checking whether they intersect any object in their path, and recursively bouncing them off the intersected object based on physical laws. How the light is reflected off objects or passed through them depends on their material. We can describe this process using functions such as: BRDF (*bidirectional reflectance distribution function*), BTDF (*bidirectional transmittance distribution function*), and BSSRDF (*bidirectional surface scattering reflectance distribution function*).

BRDF describes how light reflects off opaque objects [2, 5] and defines the look of the material (e.g. whether it looks more metallic or more plastic). BTDF describes how light travels through a translucent object (e.g. glass), while BSSRDF describes the scattering of light under the surface of an object before it leaves the object (an example of such a material is human skin).

1.1 BRDF

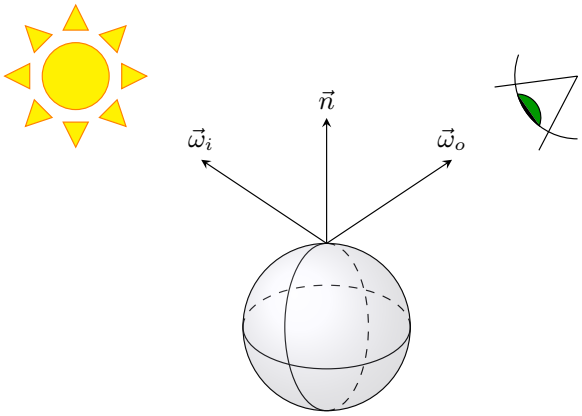


Figure 1: BRDF vectors.

The BRDF has two parameters: direction towards the light ($\vec{\omega}_i$) and direction towards the camera ($\vec{\omega}_o$), as shown in Figure 1. Based on these two directions and the internal parameters of the BRDF model, we can calculate the reflectance of the object's surface in the point hit by the ray of light.

To obtain physically-correct results, the BRDF must satisfy two properties: reciprocity and conservation of energy. A BRDF is reciprocal if it returns the same result if the light and camera direction vectors are swapped. A BRDF conserves energy if the amount of light reflected off an object is less than or equal to the amount of incident light. This means that a BRDF cannot generate light.

1.2 Lambertian BRDF

The simplest BRDF is the Lambertian model, which doesn't exist in nature (the closest materials available are unprocessed wood and printer paper, but only if we look at it straight on and not from the sides).

$$\mathrm{BRDF}\,(\vec{\omega}_i, \vec{\omega}_o) = \frac{\rho}{\pi} \tag{1}$$

Equation 1 shows the BRDF for this model. ρ represents the reflectance of the point hit by light.

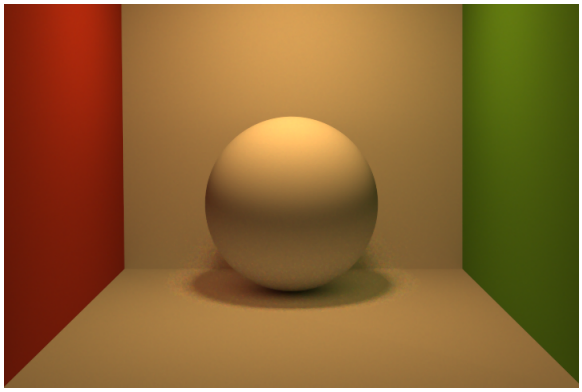


Figure 2: Example of the Lambertian BRDF.

Figure 2 shows a sphere and walls made out of a Lambertian material.

1.3 Path tracing

We can calculate the luminance of each point in a 3D space using the global illumination equation (2).

$$\underbrace{L_o(\vec{x},\vec{\omega}_o,\lambda,t)}_{\text{outgoing luminance}} = \underbrace{L_e(\vec{x},\vec{\omega}_o,\lambda,t)}_{\text{emitted luminance}} + \underbrace{\int_\Omega \text{BRDF}(\vec{x},\vec{\omega}_i,\vec{\omega}_o,\lambda,t)\cdot L_i(\vec{x},\vec{\omega}_i,\lambda,t)\cdot\langle-\vec{\omega}_i,\vec{n}\rangle\,\mathrm{d}\vec{\omega}_i}_{\text{contributed luminance from all possible directions}} \quad (2)$$

Outgoing luminance L_o, directed outward from point $\vec{x}$ in direction $\vec{\omega}_o$ in time t at wavelength λ, is the sum of luminance emitted by the object itself (L_e) and the contribution of luminance from all possible incident directions $\vec{\omega}_i$ (as shown in Figure 3).

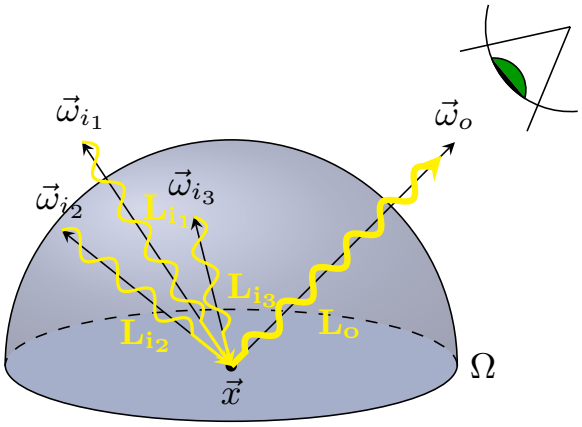


Figure 3: Graphical representation of Equation 2.

The luminance contribution is a sum of all luminances (L_i) contributed by incident rays of light coming from the area of the hemisphere Ω (which extends from point $\vec{x}$ on the object in the direction of the normal $\vec{n}$). Each contribution is attenuated based on the incident angle and the BRDF (see subsection 1.1) of the object's material.

The global illumination equation (2) has no general analytical solution and has to be solved numerically using a Monte Carlo approach.

1.4 Numerical solution

The path tracing algorithm shoots rays from the camera into the scene, bounces them around the objects within, and calculates the luminance of points hit along the way. A random sample of the hemisphere is needed when calculating the contributions, since it's impossible to check every conceivable direction. However, such an approach provably converges towards the right solution over time [3]. To improve the quality of generated images, supersampling is usually used. The more samples we use, the less noise is present in the output image, however, the render time increases.

1.5 OpenCL

OpenCL is a framework for executing programs written in a special C99-based language across heterogeneous platforms [4].

A platform can have many compute devices attached to a host processor, each of them can execute a kernel (a program written in OpenCL's C-like language). Compute devices can be standard processors (CPUs), graphics processing units (GPUs), digital signal processors (DSPs), etc.

The main advantage is that the programmer writes the kernel only once and OpenCL recompiles it for the compute devices available in the system at runtime.

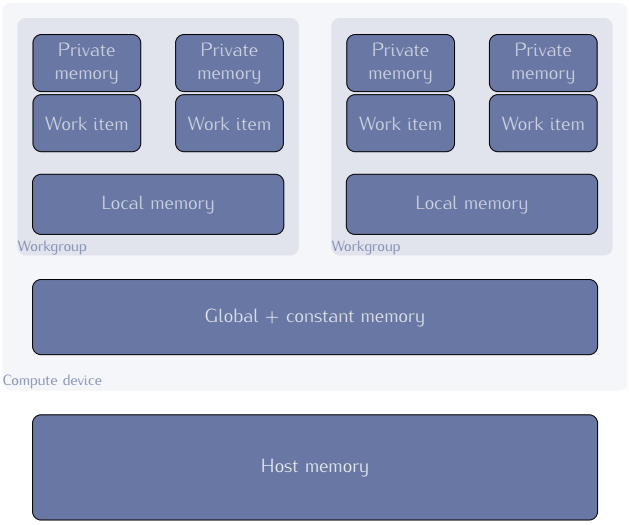
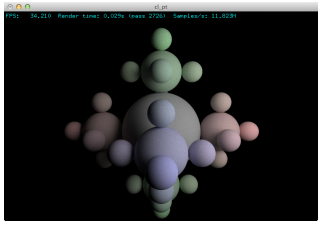


Figure 4: OpenCL memory model.

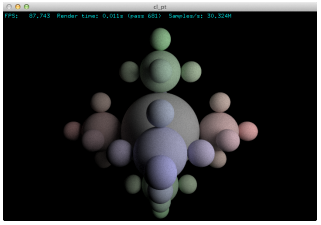
Figure 4 shows OpenCL's memory model, which is loosely based on actual hardware implementations from AMD and NVIDIA. Access to global memory is slow and should be minimized for optimal performance.

OpenCL's dialect of C includes memory region qualifiers for pointers, which determine where in the memory hierarchy the data they point to resides (`__global`, `__local`, `__constant`, `__private`). It also offers a variety of specialised fixed-length vector types that map directly to SIMD registers on the underlying hardware, thus making optimisation easier. While the language is similar to C99, it omits function pointers, bit fields, variable-length arrays, and most notably—recursion. Instead of the standard C library, a set of math-centered functions is provided.

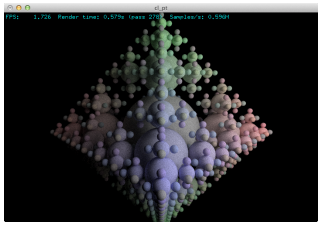
2. IMPLEMENTATION



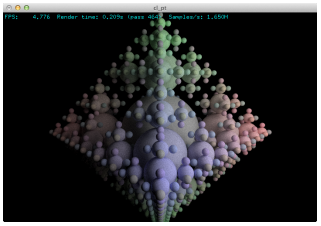
(a) 38 objects, CPU, ∼34 FPS.



(b) 38 objects, GPU, ∼88 FPS.



(c) 938 objects, CPU, ∼1.7 FPS.



(d) 938 objects, GPU, ∼4.8 FPS.

Figure 5: Comparison of rendering on CPU vs. GPU.

The test implementation renders a sphereflake (a fractal structure made out of spheres) using the path tracing algorithm. Figure 5 shows a comparison of rendering speeds on a CPU (Intel Core i7-4771: 4 cores, 8 threads) and GPU (integrated Intel HD4600: 280 compute units). The GPU is about 2–3-times faster in all tested cases, as was expected. Adjusting the OpenCL workgroup size might give even better performance.

The program is implemented in C and OpenCL 1.0, it consists of 783 lines of code in total (545 lines for the C driver and 238 lines for the OpenCL kernel). The user interface is implemented with GLUT and OpenGL (OpenGL is used only for drawing text and pixels into the window, all computation is done with OpenCL).

Rendering is parallelised by pixels (each pixel represents an independent work unit). The objects, emitters (lights), and camera parameters are saved into the constant memory of the compute device, while the raw framebuffer, random number generator state (seed) for each pixel, and the pixels themselves are stored in the device's global memory.

While the natural implementation of path tracing is recursive, it is also possible to implement it iteratively, which is what we did. The main reason for this was because OpenCL doesn't support recursion, as it is expensive and slows down execution.

Normally, the bouncing of rays is terminated using Russian roulette [2], however, that is not the most appropriate solution to use on GPUs. Russian roulette terminates the path based on a randomly generated number, which gives an uneven workload distribution. GPUs perform best when all threads within a work unit execute the same code. If a single thread has to perform a branch, it slows the other threads down. To solve this problem, a constant number of bounces is used instead.

Random number generation is an important part of every Monte Carlo simulation. Marsaglia's xorshift pseudorandom number generator (PRNG) was chosen for this program, as it is very fast and simple to implement on both CPU and GPU architectures [1]. Figure 6 shows the implementation of the aforementioned pseudorandom number generator.

```
static float gen_random(unsigned int *seed)
{
    unsigned int y = *seed;

    y ^= y << 13;
    y ^= y >>  7;
    y ^= y << 17;

    *seed = y;

    /* Convert random bits to float between 0 and 1 */
    union {
        float        f;
        unsigned int ui;
    } res;
    res.ui = (y & 0x007fffff) | 0x40000000;

    return (res.f - 2.0f) * 0.5f;
}
```

Figure 6: PRNG implementation in OpenCL.

3. CONCLUSION

A simple interactive path tracer capable of rendering scenes consisting of spheres was implemented using OpenCL. It runs on both CPU and GPU architectures, with GPUs outperforming CPUs, as expected.

Further work might include sharing the load between CPU and GPU (currently, rendering is performed on either the CPU or the GPU, but not on both at the same time), which should increase performance. A system for automatically tuning the amount of samples rendered on the CPU vs. GPU based on their speed would also be a useful addition. Another crucial improvement would be to use a spatial indexing structure to accelerate ray–scene intersections (currently, a naive linear traversal is used).

4. REFERENCES

[1] G. Marsaglia. Xorshift RNGs. *Journal of Statistical Software*, 8(14), 2003. http://www.jstatsoft.org/v08/i14/paper.

[2] M. Pharr and G. Humphreys. *Physically Based Rendering, Second Edition: From Theory To Implementation*. Morgan Kaufmann Publishers Inc., 2010.

[3] P. Shirley and R. K. Morley. *Realistic Ray Tracing, 2nd edition*. A. K. Peters, Ltd., 2003.

[4] J. E. Stone, D. Gohara, and G. Shi. OpenCL: A parallel programming standard for heterogeneous computing systems. *Computing in science & engineering*, 12(3):66, 2010.

[5] C. Wynn. An introduction to BRDF-based lighting. http://www.cs.ucla.edu/~zhu/tutorial/An_Introduction_to_BRDF-Based_Lighting.pdf.

A Garlic Clove Direction Detection based on Pixel Counting

Pavel Fičur
University of Primorska, UP FAMNIT
Glagoljaška 8, SI6000 Koper
Slovenia
pavel.ficur@upr.si

ABSTRACT
In agricultural communities it is well known that planting garlic is very labor intensive work. The main problem consists of proper orientation of garlic clove at the time of inserting it in the soil. Because of the garlic's irregular shape, this is usually done efficiently by human hand. As development of technology makes use of small cameras very accessible for a wide range of applications, garlic clove orientation detection could be efficiently implemented in garlic planting with machinery. A detection algorithm using simple pixel counting method and an experimental result are presented. Experimental rate of proper direction detection was 100%.

Keywords
garlic planting, computer vision, direction detecting

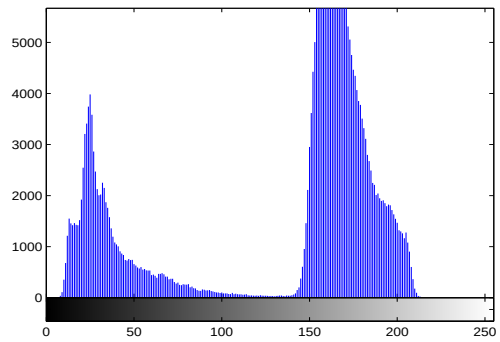


Figure 1: Histogram of the initial capture.

1. INTRODUCTION
In the process of planting garlic, proper orientation of garlic cloves is crucial. The garlic cloves have an irregular shape and are of different length and width. These properties make automated planting more difficult. An efficient orientation detection is needed, in order to make sure that the garlic clove is inserted in correct direction into a planting mechanism. Some research was recently done in the field of computer vision in agriculture [2, 3]. Most research was done on recognition of fruit, not on seeds. In a recent paper [1], authors reported a very high rate, 97%, of correctly oriented garlic cloves using computer vision. The method proposed in the article is to collect information of root or bud [1] and to calculate edge flexion for determining if it is the root or the bud. The aim of this work is to use the entire shape and compare the information from root and bud using a simple method of pixel counting. Authors [1] also noticed the problem of outdoor light which is solved in our case. Experimental work was conducted in laboratory and showed that the 100% rate of correctly detected orientation is possible. With a precise design, also on the field such result can be obtained because of the artificial light, which is crucial for the correct image.

2. METHODS
In this section data collection, image manipulation and detection algorithm are presented in detail.

2.1 Collecting data
For further data analysis of each clove, a black and white (binary) picture is needed. The picture is first taken in RGB mode and consequently set to binary (0/1) mode. In order to avoid the influence of daylight, a small dark box is used. Under the box there is an artificial light source emitting the light through an opaque glass. On this glass there is the garlic clove. The box is designed in such a way that the garlic clove can enter only in right (root down) or wrong (root up) direction [1]. In this way the RGB picture is of high contrast composed of near white pixels for the background and near black pixels for the clove shape. The histogram of ligt level on x-axis in **Figure 1** shows the gap between two main groups of pixels. The picture is further transformed into a matrix with 0,1 entries, 0 for background (white), 1 for foreground. The 1 are determined by the pixels covered by the clove. In **Figure 2** we can se the original and binarized picture of the same garlic clove.

2.2 Extracting relevant elements
As the garlic cloves vary in size, there may be some zero columns in the matrix, belonging to the left and the right parts of the matrix, and also there may be some zero rows, belonging to the upper and the bottom parts of the matrix.

These pixels belong to background and are not relevant for our direction detection, moreover, this pixels could disturb the direction detection algorithm. All these zero columns and rows must be deleted. At this point there is a matrix representing the shape of the garlic clove without any empty (only 0) row or column.

2.3 The orientation detection algorithm

As the picture is a binary matrix, it is easy to count ones in each row and adding the count to the sum of previous rows. This procedure is done twice, first from top to bottom and second from bottom to top. If the root of the clove is below, the bottom sum increases faster than the top sum and opposite for the situation where the root is up. At this moment we can select the sum results of two central rows of the clove from top and bottom, subtract the bottom sum from the top sum and check the sign. If the matix contain odd number of rows, we omit the central one. If the sign is negative, then the clove is correctly oriented (root below), if the sign is positive, then the clove is in wrong direction (root above) and must be rotated before going further in the machinery.

3. EXPERIMENTAL WORK

All steps conducted in laboratory are explained here. At the end of section the results are presented.

3.1 Getting and preprocessing pictures

Three hundred garlic seed cloves were selected for testing and inserted one by one into the box of dimensions 3 by 6 centimeters. From downside the clove was artificially illuminated through the opaque glass. The seeds were of different height and width and freely oriented (up, down, different angle). Then the RGB picture was acquired, cropped, converted in black and white mode and saved in the size of 400 by 800 pixels. See **Figure 2**. For this purpose the everyday usual photographic camera Canon G10 mounted on a tripod was used. GIMP, GNU immage manipulation program was used for image manipulation.

3.2 Direction detecting

For matrix manipulation Matlab was used. Each saved image was imported, counting of pixels in each row was performed from top to bottom and from bottom to top. In **Figure 3** the line of the difference (bottom line) is non-positive sign, that is, the clove is correctly oriented. In **Figure 4** the line of the difference (bottom line) is non-negative sign, that is, the clove is incorrectly oriented. Both figures also show the cumulative sums generated by algorithm. The sum that increases faster is the sum of the root which is wider than the bud. For the purpose of answer, the two middle rows were selected and the sign of difference of the sums was checked. For this part of work the maximum time used was 0,3s in Matlab on an i5 core personal computer.

3.3 Results

Orientation of all the three hundred cloves was correctly detected, so 100% of correctly orientation detected cloves was achieved. The reason for success of the method is quite obvious because of the natural shape of garlic cloves.

Figure 2: Original and binarized picture of the same clove.

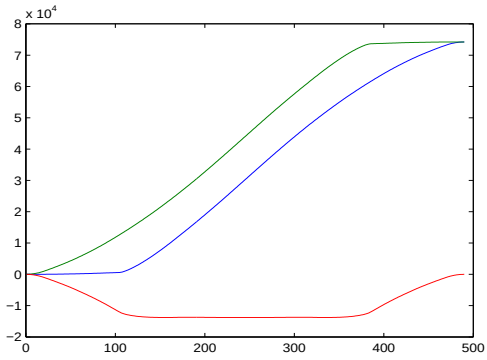


Figure 3: Diagram of a correctly oriented clove. ***The x-axis represents the row distance from bottom or top, the y-axis represents the cumulative sum of both countings. The line starting and ending at zero level is the difference.***

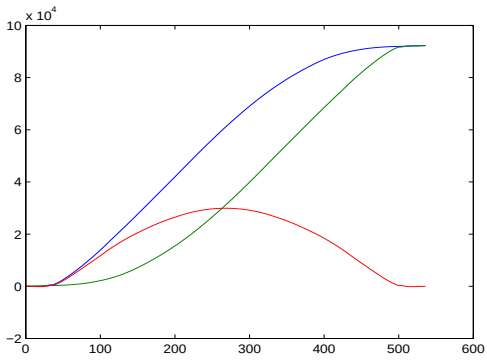


Figure 4: Diagram of an incorrectly oriented clove. ***The x-axis represents the row distance from bottom or top, the y-axis represents the cumulative sum of both countings. The line starting and ending at zero level is the difference.***

4. CONCLUSIONS

In an experimental way we can conclude that the problem of detecting garlic clove direction can be solved. Now we have to pass on the field. A small camera has to be connected to some small computer, e.g. Raspberry PI and mounted on a real planting machinery. The most important data we need is the speed of planting. Is the couple camera-computer enough fast? Another question is the minimum amount of pixels for efficient run on the real machine in the sense of correct detection.

5. REFERENCES

[1] G. Chi and G. Hui. Direction identification system of garlic clove based on machine vision. *Telkomnika*, 11(5):2323–2329, 2013.

[2] Y. Wang, L. Xu, X. Zhao, and X. Hou. Maize seed embryo and position inspection based on image processing. In D. Li and Y. Chen, editors, *Computer and Computing Technologies in Agriculture VII*, volume 420 of *IFIP Advances in Information and Communication Technology*. Springer Berlin Heidelberg, 2014.

[3] S. Yamamoto, K. Kobayashi, and Y. Kohno. Evaluation of a stawberry-harvesting robot in a field test. *Biosystems Engineering*, 15(2):160–171, 2010.

Simple approach to finding repeated patterns in opponents Texas Hold'em no-limit game

Gregor Vohl
University of Maribor
Faculty of Electrical Engineering and Computer Science
Smetanova ulica 17
Maribor, Slovenia
gregor.vohl@gmail.com

Janez Brest
University of Maribor
Faculty of Electrical Engineering and Computer Science
Smetanova ulica 17
Maribor, Slovenia
janez.brest@um.si

ABSTRACT

Collecting information about opponents in poker games is a hard task. You need to pay close attention to everything they are doing. Watch them closely and try to figure out if they are doing anything that is abnormal - playing with chips, breathing, shaky hands, swallowing saliva, etc. Anyone who has played poker before knows that this is the difference between good and great players. It is a hard task to do at a table with living opponents but it is even harder if you do not see your opponents and can not get any information about the players behavior or their emotions. Poker-bots see only actions from opponents and based on that they must build an opponent profile of play actions.

This article shows how it is possible to find repeated patterns in opponents games. We will show an easy approach to search for important patterns in history games. With the help of additional information a poker-bot can become a better and smarter player and win more chips from opponents.

1. INTRODUCTION

Texas Hold'em belongs to games with imperfect information. That means that when a player has to make a decision some information is not available. A single player does not see the cards of other players, does not know the next card from the deck and can only hope for a better end combination than an opponent. This is a game of experience and intuition. The important part for the better play is analysis of old games and learning from good and lost games. The key concept is to find if there are repeated patterns in opponents games. When playing online poker players can make notes about the opponents which are available later when the same opponent is back at the table. Because there are a lot of different players, notes can be useful for identifying and remembering the weaknesses in opponents games. At a live poker table it is a bit complicated because it is not allowed to write notes on paper. Players can make notes only in their heads. Finding the weaknesses of opponents is always a hard task. Opponents are always trying to improve their games and limit their signs of weaknesses. A lot of poker players wear sun glasses, are totally frozen, are not talking, are trying to show no emotions, etc. The famous poker quote about finding the weakest player at the poker table says: "If after ten minutes at the poker table you do not know who the patsy is − you are the patsy" [8]. Try to imagine now how hard it would be to find the weakness of an opponent if you can not see him acting behind the table and he will have no emotional downfall.

Analyzing old poker games was the key concept at implementation of poker-bot Rembrant. We used the advantage of the communication protocol from the University of Alberta. They developed an easy communication protocol which is used at ACPC competitions [1]. With the help of communication protocol it is very easy to have a great overview of historical games. We have written a simple wrapper around the history games and implemented a searching algorithm for patterns of similar play actions of opponents. Our approach was to search in the history games for repeated actions. Because the opponents are poker-bots it can happen that they will repeat the same actions even if they are losing or winning more often than living players. A poker-bot can extend the number of chips from an opponent when a play pattern is detected and repeated more frequently.

The University of Alberta developed a poker-bot called Poki [4] which uses opponent modeling with the help of a weight table. A weight table is a group of variables collected in a single table. Values of the table are called weights. The weights in the table are changed according to the method of modeling. The modeling technique focuses on all possible starting hands because each opponent plays a different strategy.

The University in New Zealand developed poker modeling with the help of machine learning techniques. They used decision trees and artificial neural networks [6].

This article is organized as follows. Section 2 shows the basics about the Texas Hold'em poker game. What are the poker actions and what are the end combinations which decide the winner. Section 3 presents our algorithm for find-

ing patterns with the help of communication protocol. A version of an algorithm with and without finding patterns approaches is presented in Section 4. Section 5 concludes this paper.

2. TEXAS HOLD'EM BASICS

Poker is part of card games where the end combinations are ranked. Poker games are unique because of betting rounds. The game has two main parts: preflop and postflop [2].

Preflop is part of the game with pocket cards. Each player gets two closed cards hidden to other players. There is only one round of betting. From a deck of 52 cards it is possible to form 1326 different starting hands [5]. The best starting hand is pocket aces and the worst is 7 and 2 offsuit (different colors). There are two manditory bets called small blind and big blind. The player next to dealer must pay the small blind bet and the player next to him must pay the big blind bet. Blinds are set before the start of current game. Big blind is normally two times bigger then the small blind value.

Postflop is part of a game where the community cards are visible to all the players. There are 3 levels (flop, turn and river) and 3 rounds of betting. 3 community cards are added on the flop and one each for turn and river. The player with the best combination of 5 cards (pocket cards and community cards) is the winner.

End combinations:

- royal flush: A K Q J T suited,
- straight flush: five suited consecutive cards (e.g. 4 5 6 7 8),
- quads: four cards with the same value (e.g. 7 7 7 7 *),
- full house: trips plus pair (e.g. A A A Q Q),
- flush: five cards of the same color,
- straight: five offsuited consecutive cards (e.g. 5 6 7 8 9),
- trips: three card of the same value (e.g. 8 8 8 * *),
- two pairs: (e.g. A A K K *),
- pair: two cards of the same value (e.g. 9 9 * * *) and
- high card: (e.g. A * * * *).

Poker actions:

- CHECK − player is not betting and the right to play moves to the next player.
- BET or RAISE − player pushes desirable amount of chips into play.

Table 1: Preflop action patterns

Patterns	Poker actions
fold to bet	rf
re-raise	rr
call bet from 1st position	rc
limp raise	cr
call to re-raise bet from 1st	rrc
fold to re-raise bet from 1st	rrf

- RE-RAISE − player pushes higher amount of chips into play than the player before him.
- CALL − player equals the amount of chips from the bet.
- FOLD −player is out of the current game.
- ALL-IN − player pushes all of the chips into the game.

3. COLLECTING INFORMATION ABOUT OPPONENTS

The most important fact when collecting information about opponents is how often they fold or how often they call a raise of our poker-bot. The algorithm searches through the history of games for patterns where the opponents were forced to fold because of bad cards or because of too high bets and for situations where opponents called raises when a poker-bot had weak starting hands. When a pattern was discovered the poker-bot was trying to play the same combination more often. We defined variables for some possible actions and then incremented the values when they appeared. We calculated the percentage values based on the number of played games. With the help of percentage values we defined poker-bot actions. When an opponent folds more frequently it is better to check/call the actions to gain more chips. If the opponent is an aggressive player then a good strategy is to play passive and tight and wait for good starting hands because the amount of gaining chips will be raised.

Collecting information was separated for preflop and postflop. For preflop games the key fact was how often the opponent called or re-raised bets from the 1st position and for the postflop games how often the opponent folded to bets or re-raised them.

3.1 Collecting information on the preflop game

During the preflop phase of poker game the algorithm searches for action patterns as shown in Table 1. First two lines of Table 1 are common patterns for preflop and are possible in every single game. The next two lines are possible only when poker-bot starts the preflop game. The last two lines cover three actions and the opponent must start the preflop game.

The poker-bot Rembrant tries to play almost every preflop hand. The exception is when the starting cards are not very good. In this case the poker-bot will try to minimize the amount of chips so that the flop community cards will be as cheap as possible. The important fact is the percentage of games where opponent is betting after limping from the

Table 2: Postflop action patterns

Patterns	Poker actions
fold to bet	rf
fold to re-raise	rrf
call bet	rc

1st position. If the percentage is high then the poker-bot Rembrant will not limp from preflop and will fold instead. A high percentage of opponent preflop raise is optimal when starting cards are good. In this case the poker-bot wants as many chips as possible in the pot.

3.2 Collecting information on the postflop game

During the postflop phase of a poker game our algorithm searches for action patterns shown in Table 2. The patterns are equally balanced for all three postflop levels. We decided to collect only the more common postflop actions here. In Table 2 are only three actions. The first two lines of the table present the patterns where the opponent folds to bet or re-raise. The last line of the table shows the pattern where the opponent is calling the bets. This pattern can occur more than once during a single game.

Every time before making an action in the postflop game the variables for the patterns are checked. The most important one is the call bet variable. When the percentage of calls is high then the poker-bot will try not to raise in case of semi-bluff situations. Instead he will try to get the next community card for free or for low price. Aggressive play is suitable only when the fold variables have high percentages. If the fold and call variables have low percentage value then poker-bot will try to play tight aggressive - raise only with good cards and fold in the case of weak end combination, to any bet of the opponent.

3.3 Collecting other information

Another very important thing to focus on is the amount of a raise by the opponent. Our algorithm calculates the average value of opponents raises through the preflop and postflop games.

3.3.1 *Preflop opponents average raise*

In preflop we decided to track the raise from the 1st position and the re-raise amount of the opponent. The average raise was valid only after a certain number of games had been played. This is important because the average raise value must become stable and the new values can not change the average too much. Preflop raise from the 1st position was an average raise if the following condition was met.

$$oppRaise - 2*BigBlind < oppRaise < oppRaise + 2*BigBlind \quad (1)$$

If the raise amount was not within that range the poker-bot starts to play careful − can mean that the opponent has strong starting hands. When the average raise was over the limit the weak starting hands are automatically folded. All preflop raises from an opponent are called when the lower average raise limit is not reached.

Normal preflop re-raise from an opponent is defined as:

$$oppReRaise > botRaise * 3 \quad (2)$$

If the re-raise amount of opponent was bigger than 3 times the raise of poker-bot then the poker-bot continues to play only with strong starting hands. Calling a big re-raise with weak hand means loss of chips. If the poker-bot has good starting cards (pocket pairs, AK or AQs) it will re-raise to all-in. In the case of a normal re-raise, the weak starting cards are folded anyway.

3.3.2 *Postflop pot size*

The raise amount in postflop is the same for all three levels of postflop. The raise amount is no longer measured on the raise of the poker-bot but on the pot size. Pot size is the current amount of all the chips in the play.

We defined three different raise types:

- under pot raise.
- pot raise.
- over pot raise.

Under pot raise is a raise from opponent where the amount of opponents' raise is under the current pot size. Pot size raise equals the amount of opponents' raise. Over pot raise is a raise from opponent where the amount of opponents' raise is over the current pot size.

As normal raise we defined only the pot raise. That means that the opponent is raising for as many chips as in the current pot. The problems is when the opponent is betting under or over the current pot size. When betting over the pot size it can mean that he is bluffing and wants us to fold the current hand. This is very common action on the river when acting first - it can mean monster hand or air. Variables for action patterns play an important role in this scenario. If it is possible to detect the purpose of opponent's raise with variables poker-bot has an easy choice. Other way around it is very hard to predict and in most cases the better options is to fold if poker-bot has no good end combination.

In the case of under pot raise an opponent can have a very strong end combination and wants some more chips. To make under pot raise while bluffing is not a common action because small pot bets are normally called every time. Poker-bot will call all raises of opponents when the under pot size conditions are fulfilled. Poker-bot is calling or re-raising raises only with good end combinations. Hands with multiple outs to hit a better end combination are called only when a normal raise from opponent is played.

3.3.3 *Opponent moves all-in*

A very important fact to know is also how often an opponent pushes all-in. All-in is a special bet or re-raise when the player moves the entire stack into the game. Losing all-in means game over. Players usually play all-in with the best cards or good end combinations. All-in move is also a good bluff method and can be performed in the preflop and postflop phases of the game. When putting all chips in play in the preflop game this is normally a coin flip situation and a little luck is needed to win it. We defined a variable which is counting the number of opponents all-in moves. If

Table 3: Results of Rembrant 3.0 and Rembrant 1.0 vs Lucky7

Experiment	Rembrant 3.0 [chips]	Rembrant 1.0 [chips]
1	38114	-2489
2	25198	8055
3	28961	7115

the percentage value is high that means that the opponent is bluffing most of the time. In this case poker-bot will try to call all-in moves from opponents more often with lower ranked end combinations.

4. TESTING AND RESULTS

For testing we used the poker-bot Rembrant version 1.0 [7] which does not use the algorithm for finding patterns. Our latest version of the poker-bot Rembrant is 3.0 and it uses an approach for finding repeated patterns. We wanted to test and to see whether the introduced approach improved our poker-bot. Comparison between two different versions was not enough so we also tested our algorithm against a poker-bot called Lucky7 [3] which was the result of another team from our University. Lucky7 was one of the top 3 poker-bots at ACPC 2011 competition. Lucky7 is a poker-bot which uses multi poker-bots to define the game action. Every poker-bot has a different style of play and a voting system defines the current game action.

The testing model was divided into two parts with 5000 hands. After the end of the first part the poker-bots switched positions and the starting cards. All the collected information from the bots were deleted. Switch players was important to erase the factor of luck. Small blind is 50 and big blind is 100. Both players have a starting stack of 20.000 chips to leave enough space for combinations. Starting stack was reset after every single hand was played. Decks of cards are generated with a random number generator. The generator used the same starting seed every experiment to ensure the same deck of cards in every game.

4.1 Results

Table 3 shows the results of the experiment against the Lucky7 poker-bot. Rembrant 1.0 was better only in the first experiment of the test and lost the other two experiments. All in all the Lucky7 poker-bot was better. The results show that the Rembrant 3.0 poker-bot improved a lot with the help of the approach for finding patterns in opponents games. Rembrant 3.0 was better in all three experiments and won a lot more chips from the opponent than the previous version. The pattern approach helped the new version of poker-bot to save chips in the cases of weak hands and to gain more chips from opponents when its hands were good. This was possible because Lucky7 is still a case based poker-bot and its actions started to repeat after a certain amount of time. The algorithm detected the repeated patterns and tried to take advantage of them. With the help of the algorithm it reduced the loss and improved the gained chips by almost 4 times, to the version Rembrant 1.0.

In tests against better opponents poker-bot will not win all the games but will reduce the amount of chips when playing a losing hand. Reducing the chips when losing a hand is crucial. Because opponents got better and they improved their games and it got harder to get chips from them. To protect the gained chips is therefore the key element of the poker-bots and that is why reducing the losing chips is so important.

5. CONCLUSION

We have introduced a simple approach for modeling opponents actions by searching for patterns in history game. Variables for each pattern were converted to nominal values of played games. The pattern approach is used before making the final decision about the current action the pattern approach is used. When a pattern is discovered current action can change in order to improve the amount of chips in play. In the case of losing hands poker-bot can decrease the number of losing chips and in the opposite case it can improve the number of winning chips. The results from Table 3 shows that the algorithm for patterns is effective and improved the previous version of poker-bot Rembrant. The latest version won more chips from opponents than the previous one.

For future work we will try to add more pattern variables to the algorithm. An increased number of variables would give a much better overview over the actions from the opponents. We will also try to improve the current game actions when a play pattern is discovered to gain even more chips from opponent or to lose less chips. One of our goals is also to create a virtual simulation of the current game with all possible future actions to see how a single action of poker-bot can effect opponents games.

6. REFERENCES

[1] Annual computer poker competition. `http://www.computerpokercompetition.org/`. Accessed: 2014-08-10.

[2] S. Braids. *The intelligent guide to Texas Hold'em poker*. Intelligent Games Publishing, 2003.

[3] B. Butolen, S. Zemlic, M. Cof, and M. Zorman. Lucky7 poker playing multi agent system. `http://www.computerpokercompetition.org/downloads/competitions/2011/presentations/Team_Lucky7_Poster.pdf`. Accessed: 2014-08-29.

[4] A. Davidson, D. Billing, J. Shaeffer, and D. Szafron. Improved opponent modeling in poker. *Proceedings of The 2000 International Conference on Artificial Intelligence (ICAI'2000), Las Vegas, Nevada*, pages 1467–1473, 2000.

[5] R. D. Harroch and L. Krieger. *Poker for dummies*. Wiley Publishing, 1997.

[6] McNally, Patrick, Raffi, and Zafar. Opponent modeling in poker using machine learning techniques, northwestern university. `http://www.cs.northwestern.edu/~ddowney/courses/349_Fall2008/projects/poker/Poker.pdf`. Accessed: 2014-08-20.

[7] G. Vohl, B. Boskovic, and J. Brest. A rembrant poker bot program. *Elektrotehniski vestnik*, 79(1-2):13–18, January 2012.

[8] M. J. Whitman and M. Shubik. *The Aggressive Conservative Investor*. Random House, New York, 1979.

Histogram of Oriented Gradients Parameter Optimization for Facial Expression Recognition

Uroš Mlakar
University of Maribor
Smetanova 17
Maribor, Slovenia
uros.mlakar@um.si

ABSTRACT

This paper proposes a method for optimal parameters regarding the Histogram of Oriented Gradients (HOG) filter. Differential evolution (DE) was used for the HOG parameter tuning using images from the MMI Facial expression database. The database was split into three parts: training, development and testing set. HOG parameters were tuned on the development set. The obtained result show an improvement of the evolved parameters during the evolution, while the experiments on the testing set failed to provide a better performance in the end.

Keywords

Facial expression recognition, histogram of oriented gradients, evolutionary algorithm, feature selection, differential evolution

1. INTRODUCTION

Facial expression recognition has been a very interesting topic since the early 90s. There have been many advances over the past decade in the areas of face detection, face tracking, feature selection and facial expression recognition. Facial expressions are a very powerful tool for humans to communicate, show emotions and their intentions, and thus makes facial expression recognition an interesting research topic in the area of human-computer interaction. Over recent years facial expression recognition has gained attention because of its wide of applications, which reach many different areas like computer graphics, robotics, security, driver safety, etc. The main goal of researchers in the field is to build rich and responsive graphical user interfaces. The most promising application is the application of driver attention control. When the driver looses attention (sleepiness) the car automatically slows down [11] [6]. Regardless of the advancement of technology, facial expression recognition remains a difficult task to this day. Researches in the area of facial expression recognition usually focus on recognizing six prototypic emotional states: 'Disgust', 'Anger', 'Surprise', 'Fear', 'Happiness', and 'Sadness'. Each of the six emotional states can be seen in Figure 1.

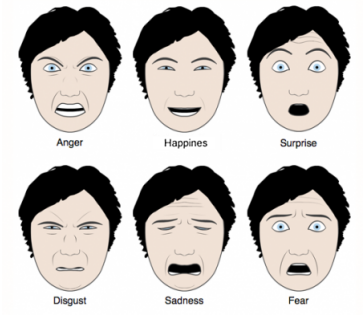


Figure 1: Six prototypic emotions [1]

Facial expression recognition applications can be divided into two areas, depending on how the facial features are extracted: geometric feature-based and appearance-based methods. The main difference between the two, without going into detail, is the way information is extracted from the face image. With appearance based methods, a filter is usually applied to the face image, while geometric feature-based methods rely heavily on the shapes of important facial features i. e. eyes, mouth, nose, and wrinkles. In this paper appearance-based methods were used for facial expression recognition. A facial expression recognition system usually consists of three parts: namely face detection/tracking, facial feature selection and facial feature classification. In the first part a face is detected using the Viola-Jones detector or the active appearance models. The next step after localizing the human face is to extract facial features. Such features are used to describe the facial expression and, consequently, the emotion of monitored person in the digital image or video clip. The focus of this paper was the facial feature selection part, where an evolutionary algorithm (EA) was used to select the optimal parameters for the appearance descriptor Histogram of Oriented Gradients (HOG), to achieve maximum facial expression recognition efficiency. The last part of a facial expression recognition system is a classification module, which classifies extracted facial features into one prototypic emotion (usually into six or seven emotions).

The rest of the paper is organized as follows. In Section 2 some of related work is presented in the area of feature selection. In Section 3, an EA will be presented, which was used for this study, then in section 4 HOG appearance descrip-

tor will be described. In Section 5 we present the proposed method and in Section 6 the results will be presented. With Section 7 we will conclude this paper with some findings.

2. RELATED WORK

A number of works in parameter optimization/tuning using evolutionary algorithms have been made. In [10] a genetic algorithm (GA) was used for optimizing the parameters of a set of Gabor filters for the purpose of on-road vehicle detection. Liu et. al [7] used GA for face recogition. The original data was projected to a lower dimension using principal component analysis (PCA), where the rotations of the basis vectors were optimized using a fitness function defined by performance accuracy and class separation. Yen et. al [12] used GA for optimal ellipse parameters estimation for facial feature extraction.

3. EVOLUTIONARY ALGORITHM

An evolutionary algorithm is a generic population-based optimization algorithm. It uses mechanisms which are inspired by biological evolution, such as reproduction, mutation, recombination, and selection. The solutions for the optimization problems are the individuals in the population, and the fitness function provides a metric for measuring the quality of the solutions. In this paper an EA called differential evolution (DE) has been used for optimal parameter selection of the HOG filter. Hereinafter the DE algorithm will be described in detail.

3.1 Differential Evolution

Differential evolution (DE) [9] is a population based optimization algorithm used for global optimization. It is simple, yet very effective in solving various real life problems. The idea of DE is a simple mathematical model, which is based on vector differences.

The population of the DE algorithm consists of Np individuals $\mathbf{x}_i$, where $i = 1, 2, ..., Np$, and each vector consists of D-dimensional floating-point encoded values $\mathbf{x}_i = \{x_{i,1}, x_{i,2}, x_{i,1}, ..., x_{i,D}\}$. In the evolutionary process, individuals are evolved using crossover, mutation and selection operators, which are controled by a scale factor F and crossover rate C_r. The following mutation strategy was used:

$$\mathbf{m}_i = \mathbf{x}_{r1} + F * (\mathbf{x}_{r2} - \mathbf{x}_{r3}). \quad (1)$$

For the creation of a mutant vector $\mathbf{m}_i$, three random vectors from the population are selected, defined by indexes $r1$, $r2$ and $r3$. indexes are mutually different and different from i. They are selected uniformly within the range $\{1, 2, ..., Np\}$. The scale factor F is defined within the range $[0, 2]$. The mutant vector $\mathbf{m}_i$ is obtained by adding a scaled vector difference to a third vector.

The trial vector $\mathbf{t}_i$ is then generated using the crossover rate C_r and a corresponding vector $\mathbf{x}_i$ from the population as:

$$t_{i,j} = \begin{cases} m_{i,j} & \text{if } rand(0,1) \leq C_r \text{ or } j = j_{rand}, \\ x_{i,j} & otherwise. \end{cases} \quad (2)$$

As can be seen from Eq. 2, the crossover rate C_r is defined at the interval $[0, 1]$ and it defines the probability of creating the trial vector parameters $t_{i,j}$. The j_{rand} index is responsible for the trial vector to contain at least one value from the mutant vector. After crossover, some values of the vector may be out of bounds, meaning that these values must be mapped to the defined search space.

The next step in the evolutionary process is the selection of the fittest individuals. During the selection process the trial vector $\mathbf{u}_i$, competes with vector $\mathbf{x}_i$ from the population. The one with the better fitness value survives and is transfered to the next generation.

4. HOG DESCRIPTOR

Histograms of oriented gradients (HOG) were developed by Dalal and Triggs [4] in 2005. They are based on orientations of image gradients and have been widely used for pedestrian detection. Some researchers have tried it for facial expression recognition problems and obtained very good results. Orrite et al. [8] proposed a facial expression recognition system based on class specific edge distribution and log-likelihood maps, the latter were used for building a list of HOG features supplemented by SVM classifier. Experiments on the CK database pointed out 83.3% recognition rate. Gritti et al. [5] studied HOGs, local binary patterns (LBP) and local ternary patterns (LTP) for emotion recognition. Recognition rate of 92.9% on the CK database was obtained by using the LBP appearance descriptors. The idea of the descriptor is very simple, yet it provides a powerful description of the object within the image. The feature vector obtained with the HOG descriptor is calculated on a grid of uniformly spaced cells, which are grouped into overlapping blocks, which account for contrast normalization. Let us take a closer look at the parameters with the following example:

Bins	Cell size	Block size	Orientation	Clip value
9	16	2	0	0.2

In the provided example an image is divided into cells which are 16×16 pixels in size. Within each cell gradients are computed using the filter [-1 0 1] in both directions. The orientations are then placed into 9 evenly distributed bins. Whether oriented gradients are used, the bins are distributed from 0° to 360° or 0° to 180° otherwise. These cells are then grouped together into 2×2 blocks. In the final stage all values in the feature vector are clipped using the clipping value. In Figure 2 HOG feaure vector calculation is presented. For more information refer to [4].

5. PROPOSED METHOD

In this section we will briefly describe our algorithm for optimal parameter selection of the HOG appearance descriptor. The values of the HOG filter were described in Section 4, so we will not go into detail about them here. All parameters were bound by lower and upper bounds as:

- Bin size [8, 16],
- Cell size [8, 16],
- Block size [2, 5],

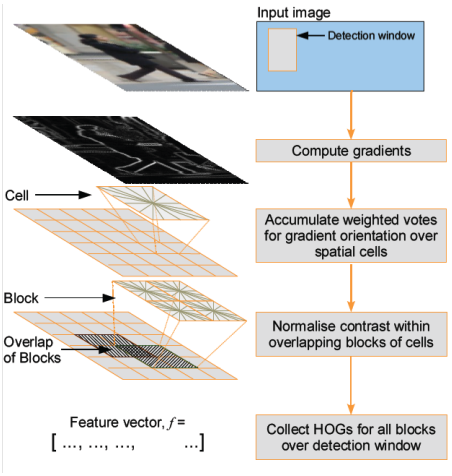


Figure 2: HOG feature vector calculation [4]

- Orientation [0, 1],
- Cliping value [0, 1].

All values, except the cliping value, are rounded to integer values at the time of evaluation. Parameter selection was done in the following matter. After each generation of DE all individuals were evaluated in the following way: Each image in the training set was transformed using the current individual $\mathbf{x}_i$ of HOG parameters, giving a feature vector $\mathbf{f}$. All vectors $\mathbf{f}$ were sorted according to the emotion label of the images. With that in mind, after sorting, we ended up with six types of feature vectors, each belonging to a prototypic emotion. The next step was to evaluate the efficiencies of the current set of HOG parameters using a classifier. The classifier used in this paper was support vector machines (SVM) using a radial basis function kernel. An implementation of SVM from the LibSVM toolbox [3] was used. A 5-fold-cross validation was carried out in order to find optimal parameters for SVM and our classification problem. Six SVMs were built, each to distinguish one emotion from the others. All images in the development set were transformed to feature vectors, and then classified using the trained SVMs. The final output of the evaluation procedure of a single HOG parameters' set is a metric called efficiency of EA (EEA) (Eq. 3), which is defined as:

$$EEA = \frac{N_d}{A_d} * 100, \tag{3}$$

where N_d is the number of correctly classified images and A_d the total number of all images in the development set.

Lastly we must also mention the jDE strategy which was used in this paper. As mentioned in Section 3, the DE algorithm has three control parameters: Np, F and Cr. The values of these control parameters are usually problem dependent, and their values adapt during the evolutionary process. A self-adaptive control strategy was used as in [2] for the control of values F and Cr. With the inclusion of control parameters during the evolution, each individual in the population had the form of $\{x_{i,1}, x_{i,2}, x_{i,1}, ..., x_{i,D}, F_i, Cr_i\}$. In each generation right before the mutation operation, the new values of F and Cr are computed using the following equations:

$$F_i = \begin{cases} F_l + rand(0,1) * F_u & \text{if } rand(0,1) \leq \tau_1, \\ F_i & otherwise \end{cases} \tag{4}$$

$$Cr_i = \begin{cases} rand(0,1) & \text{if } rand(0,1) \leq \tau_2, \\ Cr_i & otherwise. \end{cases} \tag{5}$$

$\tau_1 = 0.1$ and $\tau_2 = 0.1$ in Eq. 4 and 5 are probabilities for adapting the values of F and Cr. F_l and F_u define the interval on which the factor value F is defined.

6. RESULTS

In this section, the obtained results are presented, and also the experimental environment is described.

6.1 Experimental environment

Our method was validated on a publicly available dataset called the MMI Facial Expression database. The MMI database is a web searchable collection of video clips and images in which the subjects display a wide variety of facial expressions. Subjects differ in age, ethnicity and sex. There are a total of 69 subjects in the database. Some videos have been annotated with emotions labels. For testing purposes only those subjects from the MMI database having emotions labels were selected. Thus, human face images from just 27 volunteers were used in our evaluation procedure. Because data in the MMI database is in the form of video clips, those frames had to be extracted, where the displayed emotion was at the peak (apex) throughout its duration. In other words, just those frames where emotions were the most expressive were selected in our analysis (3 frames).

All the acquired frames were then split into three subsets: training, development and testing set. By splitting the database attention was paid that no subject appeared in two sets at the same time (all three sets were unique). That gave us a total of 10 subjects for the testing set, 10 for the training set and 6 for development. In Table 1 we can see the total number of images per emotion in the whole dataset, while Table 2 presents a total of images per splited dataset.

Table 1: Number of images per emotion in the whole dataset

Dis.	Ang.	Sur.	Fea.	Hap.	Sad.	Σ
102	108	132	105	132	108	687

The split was done randomly, with 40 % of all subjects selected for training(10), another 40 % for testing(10) and 20 % for the development set(6).

6.2 Experimental results

The following settings of the DE were used for the experiments in this paper : Np was set to 20, G to 100, while the

Table 2: Number of images in each dataset

Training	Development	Test
249	147	291

F and Cr were set to 0.9 and 0.1 respectively at the start of the evolutionary process. The Np and G were set to smaller values so the results could be obtained in reasonable time. In each generation $Np * 6$ classifier learning was done.

In Figure 3 the results of the best, worst, median and mean are reported for each generation on the development set . It can be seen from the graph that the DE algorithm was successful in tuning the optimal parameters on the development set. For the final experiment we also tried the tuned HOG parameters on the test set and compared them to the randomly generated parameters in the first generation. The results of the comparison are collated in Table 3.

Table 3: Comparison between the first and last generation parameters on the test set

	1st generation	100th generation
best	394.37	389.32
worst	100.00	340.70

After testing the tuned parameter sets on the testing set of the MMI database the experiment showed intersting results. The best parameter set in the last generation gave worse result then the best in the first generation, in spite of giving better results on the development set. We investigated the cause of this problem and found an answer in the database itself. The MMI Facial expression database mostly contains videos of subjects which are performing spontaneous emotional responses, that we try to classify as six emotional states. The problem at hand is that each subject expresses emotions in their own particular way. Because the training, development and test sets were completely independent of each other, and because of the way people show spontaneous emotions, the obtained results are slightly worse than those on the development set.

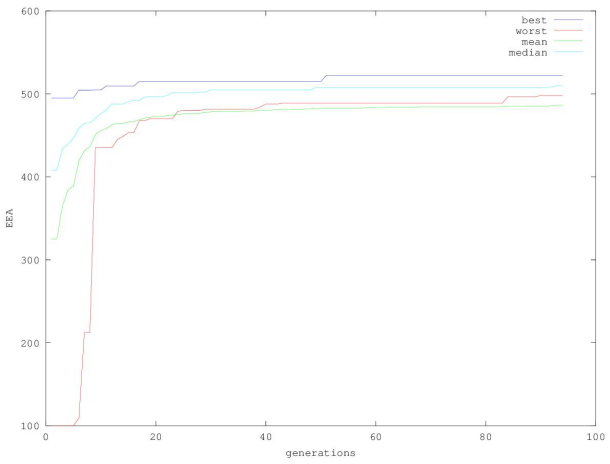


Figure 3: Graph of best, worst, mean and median for each generation during tuning on development set

7. CONCLUSION

This paper proposed a method for optimal HOG descriptor parameters selection using the self-adaptive DE. It was tested on a publicly available Cohn Kanade dataset, which was separated into three sets: testing, training and development. The HOG parameter tuning was done with the help of the training and development sets. The results show an improvement in selecting the appropriate HOG parameters on the development set. Due to fairly large feature vectors, the learning part of the proposed method was very time consuming, therefore smaller values for Np and G were selected. The obtained results from tuning were also tested on the testing set. A comparative study was made between the best and worst results of the tuning parameters in the first and last generations on the test set. Because of the spontaneous emotions displayed by subjects in the MMI Facial expression database the results in the case of the best tuned parameter in the last generation gave worse results than those in the first generation.

In the future we would like to address the problem of spontaneous emotions during HOG filter parameter optimization, and also test the proposed algorithm on more available datasets.

8. REFERENCES

[1] Six prototypic emotions. https://managementmania.com/en/six-basic-emotions. Accessed: 2014-07-18.

[2] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *Evolutionary Computation, IEEE Transactions on*, 10(6):646–657, 2006.

[3] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at urlhttp://www.csie.ntu.edu.tw/ cjlin/libsvm.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In Cordelia Schmid, Stefano Soatto, and Carlo Tomasi, editors, *International Conference on Computer Vision & Pattern Recognition*, volume 2, pages 886–893, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005.

[5] T. Gritti, C. Shan, V. Jeanne, and R. Braspenning. Local features based facial expression recognition with face registration errors. In *Automatic Face & Gesture Recognition, 2008. FG'08. 8th IEEE International Conference on*, pages 1–8. IEEE, 2008.

[6] Qiang Ji and Xiaojie Yang. Real-time eye, gaze, and face pose tracking for monitoring driver vigilance. *Real-Time Imaging*, 8(5):357–377, 2002.

[7] C. Liu and H. Wechsler. Evolutionary pursuit and its application to face recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(6):570–582, 2000.

[8] C. Orrite, A. Gañán, and G. Rogez. Hog-based decision tree for facial expression classification. In

Pattern Recognition and Image Analysis, pages 176–183. Springer, 2009.

[9] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[10] Z. Sun, G. Bebis, and R. Miller. On-road vehicle detection using evolutionary gabor filter optimization. *Intelligent Transportation Systems, IEEE Transactions on*, 6(2):125–137, 2005.

[11] Mohan M Trivedi, Tarak Gandhi, and Joel McCall. Looking-in and looking-out of a vehicle: Computer-vision-based enhanced vehicle safety. *Intelligent Transportation Systems, IEEE Transactions on*, 8(1):108–120, 2007.

[12] G. G. Yen and N. Nithianandan. Facial feature extraction using genetic algorithm. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 2, pages 1895–1900. IEEE, 2002.

Constructing domain-specific semantic dictionaries to supplement domain-specific knowledge bases

Goran Hrovat*
University of Maribor
Faculty of Electrical Engineering and Computer Science
Smetanova 17, 2000 Maribor
Slovenia
goran.hrovat@um.si

Milan Ojsteršek
University of Maribor
Faculty of Electrical Engineering and Computer Science
Smetanova 17, 2000 Maribor
Slovenia
milan.ojstersek@um.si

ABSTRACT
Semantic dictionaries as well as knowledge bases are important source of information for natural language processing. Using corpus and algorithms for constructing semantic space, we can quickly construct semantic dictionary, which is exploit to supplement the knowledge base. Algorithm for constructing semantic space, COALS was chosen. Semantic distance between terms in semantic space reveals their similarity, which is used to add semantic relationships in the dictionary. Semantic dictionary also serves to assist in identifying entities and in relation extraction, to supplement the knowledge base.

Keywords
semantic space, correlation, semantic dictionary, knowledge base

1. INTRODUCTION
Semantic dictionaries play important role in tasks of natural language processing (e.g., named entity recognition [1], relation extraction [2], information extraction [3] [4]) as well as knowledge bases. In semantic dictionary relations between words are defined, such as hypernym, hyponym, antonym, synonym, etc. Knowledge base is even more complex, where any possible relation can be defined between terms to construct graph. It is usually represented in RDF format. An example of semantic dictionary is WordNet [5] and an example of knowledge base is DBpedia [6]. Some approaches for dictionary construction have already been proposed in the past. Xu et al. [7] constructed a medical dictionary of disease terms from randomized clinical trials (RCT) using an automated, unsupervised, iterative pattern learning approach. Navigli et al. [8] developed an automatic method for constructing BabelNet, a wide-coverage multilingual knowledge resource. The method is consisted of two steps, where the first step produces a mapping between a multilingual encyclopedic knowledge repository (Wikipedia) and a computational lexicon of English (WordNet). The second step further use a machine translation system to collect a very large amount of multilingual concept lexicalization. Morsey et al. [9] constructed DBpedia which continually extracts structured information from Wikipedia and store them in files as RDF triples. This article proposes the method for constructing domain-specific semantic dictionary using domain-specific corpus or collection of documents in order to supplement domain-specific knowledge base.

*Corresponding author

2. ISSUES AT CONSTRUCTING DOMAIN-SPECIFIC SEMANTIC DICTIONARIES
The main requirements for constructing domain-specific semantic dictionary are domain-specific knowledge and domain-specific corpus. For domain-specific corpus bachelor thesis, journals etc. can be used. For domain-specific dictionary a bag of words from corpus are considered and further analyzed. All words are morphologically tagged and stop words such as pronouns and conjunctions are excluded. The most important feature of semantic dictionaries is relations between words. The examination of n^2 word pairs in case of n words need to be done to define relations (e.g. hypernym) between all word pairs. Because of such amount of pairs manual examination is unfeasible and some automatic or semi-automatic approach is necessary.

3. CONSTRUCTING DOMAIN-SPECIFIC DICTIONARY USING SEMANTIC SPACE
To reduce the n^2 problem space, algorithms for constructing semantic space is incorporated in our method. Semantic space is high dimensional space where each word is placed based on its meaning. Words in this space close to each other by specific measure (e.g. Euclidean distance, cosine distance, correlation distance) are therefore semantically close, which can be exploit for efficiently constructing semantic dictionary. To place words in semantic space one of the semantic algorithms is performed on corpus. The most known algorithms for constructing semantic space are HAL [10], LSA [11], Random Indexing [12], COALS [13] and some algorithms based on dictionary WordNet [14], which do not use corpus. In our case COALS is used on bachelor theses

of University of Maribor. All documents had been processed where unimportant part of texts (e.g. table of contents, references) were excluded.

3.1 Corpus preprocessing

Domain-specific corpus is required for using COALS (Correlated Occurrence Analogue to Lexical Semantic). In our case the bachelor theses from University of Maribor from the field of computer science are used. Bachelor theses are in PDF format, loaded in digital library of University of Maribor by students. Preprocessing of the corpus was performed in next phases:

1. Converting PDF format to plain text using Apache Tika [15].
2. Segmentation of each the bachelor thesis to first page, table of content, table of figures, content and references.
3. Tokenization of the documents' contents.
4. Morphologically tagging and lemmatization of the documents' contents using TreeTagger [16] learned on morphologically tagged corpus FidaPlus [17].
5. Only semantically meaningful words, meaningful verbs, nouns, adjectives, adverbs and abbreviations are preserved. Other morphological types do not reflect semantics of the text and are therefore excluded.
6. Converting to lower case letters.

3.2 Calculating semantic vectors using algorithm COALS

Semantic vector of the word represents the word in the semantic space. Semantic vector is also context vector, because it contains encoded context of the word. Algorithm COALS calculates semantic vectors in the next phases.

1. Construction of the co-occurrence matrix, where each row represents observed word and each column represents context word frequency. This matrix is called word by word matrix in comparison with more known LSA where word by document matrix is constructed. For observed word weights of context words are incremented according to its occurrence in text. In our case window of four words is used, which means that next the nearest context word get weight of 4, one context word away get weight of 3, etc. In contrast, LSA uses whole document window where all words are weighted equally. Number of columns or dimensionality of semantic vectors depends on the number of unique words in corpus. Because the number of unique words can be large we can limit it to 100,000 most frequent words and thus limit the dimension to 100,000.
2. The matrix is then transformed using correlation normalization to reduce the impact of high frequencies. Correlation normalization is calculated with Equation 2.

$$T = \sum_i \sum_j w_{i,j} \tag{1}$$

$$w'_{a,b} = \frac{T - \sum_j w_{a,j} \sum_i w_{i,b}}{\sqrt{\sum_j w_{a,j}(T - \sum_j w_{a,j}) \sum_i w_{i,b}(T - \sum_i w_{i,b})}} \tag{2}$$

3. Each cell in the matrix is further transformed with Equation 3.

$$a_{i,j} = \begin{cases} 0, & x < 0 \\ \sqrt{x} & x \geq 0 \end{cases} \tag{3}$$

4. Last step is dimensionality reduction of vectors which is performed using SVD (singular value decomposition) [18] shown in Equation 4. In our case dimension was reduced to 800.

$$M = U\Sigma V^* \tag{4}$$

3.3 Semantic similarity

Semantic similarity between words is calculated using their semantic vectors. For calculating similarity between two semantic vectors well known similarity measures are used. In case of HAL Euclidean distance is used, for LSA cosines distance is used and in our case for COALS correlation distance is used on the two vectors presented in Equation 5.

$$S_{a,b} = \frac{\sum(a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum(a_i - \bar{a})^2 \sum(b_i - \bar{b})^2}} \tag{5}$$

Equation 5 returns similarities between -1 and 1 where 0 means no correlation between words and values 1 or -1 mean strong correlation between words. On Fig. 1 words are presented in two dimensional semantic space. For representational purposes dimension is reduced from 800 to 2 using MDS (multidimensional scaling) [19] where correlation similarity is used as a distance measure. Three different groups are marked for clear presentation of the method. Words mac and widows are close together, which means that they appear in similar context, but not necessary together.

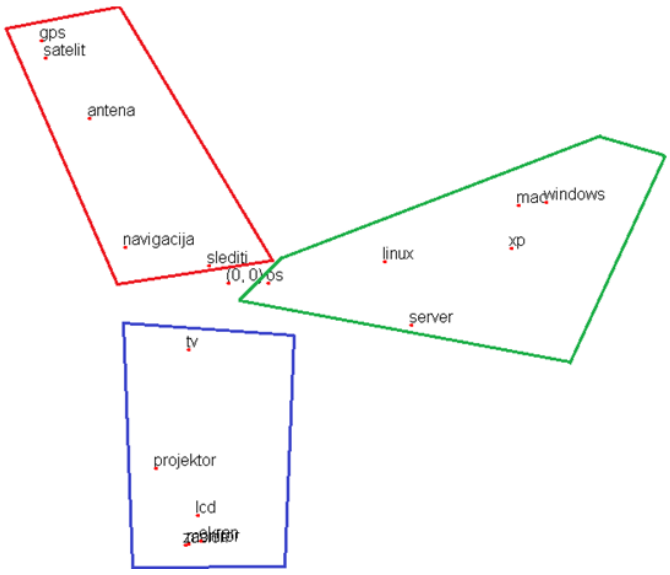


Figure 1: Visualization of the words in two dimensional space

3.4 Preparation of the results of COALS

Algorithm COALS is used to compute semantic vectors where correlation similarity metric gives us similarity between the words. Values closer to 1 or -1 mean stronger similarity whereas 0 means no similarity. Using this information the data can be prepared for constructing semantic dictionary where experts can define type of relations for their domain. Word pairs is then ordered by correlation absolute value in descending order so an expert can consider the most similar word pairs first. Construction of the semantic dictionary is much faster using this method.

3.5 Construction of the semantic dictionary and relations

An expert defines relations between the most interesting word pairs in the semantic dictionary. The interestingness is calculated with the algorithm COALS, what significantly reduces the number of interesting word pairs. An expert has some definitions predefined however his own can also be added. Predefined relations are:

- synonym (two words are synonyms, when they have same meaning)
- antonym (two words are antonyms when they have opposite meaning)
- hypernym (word A is a hypernym of word B, if A has more general meaning)
- hyponym (word A is a hyponym of word B, if A has more specific meaning)
- meronym (word A is a meronym of word B when A is a part of B, e.g., CPU is a meronym of computer)
- holonym (inverse relation of a merony, e.g., computer is a holonym of CPU)
- troponym (word A is a troponym of word B when A more specifically expresses an action of B, e.g., typing is a troponym of writing)
- coordinate word (word A and B are coordinate words to each other, when they have same hypernym, e.g. CD and DVD are coordinate words, because they have same hyperny medium)
- related word (when expert can define any of the above mentioned relations, then he can define his own relation or set only that word A and B are related words)

4. SUPPLEMENTING DOMAIN SPECIFIC KNOWLEDGE BASE

Domain specific knowledge base contains only knowledge from the single domain. Knowledge base is represented with an ontology where also rules are defined used for reasoning. Knowledge base can also contain relations from semantic dictionary, which can be supplemented from the corpus using the methods of natural language processing (e.g., named entity recognition, relation extraction). For each relation its frequency can be also acquired using the corpus.

4.1 Named entity recognition

Named entity recognition can further improve the semantic dictionary. It is useful for defining hypernyms and is also prerequisite for relation extraction. Named entity recognition is used for recognition of people, animals, location, time etc. If dog is recognized as an animal, can be that inserted in the domain specific knowledge base.

4.2 Relation extraction

Relation extraction enables us to find relations between the words from the corpus, which are then inserted into the knowledge base. For relation extraction patterns are needed which define relations. Examples of the patterns for relation "is husband of" is shown in Table 1.

Table 1: Examples of the patterns for relation "is husband of" used for relation extraction

Pattern	Relation "is husband of"
A is husband of B	A "is husband of" B
A has wife B	A "is husband of" B

A is married with B	if A is male then
	A "is husband of" B
	else
	B "is husband of" A

A and B went on the honeymoon	if A is male then
	A "is husband of" B
	else
	B "is husband of" A

Semantic dictionary can be used for manually defining the patterns for relation extraction, which can serve us for improving knowledge base.

5. CONCLUSION

The paper presents the method for constructing domain specific dictionaries using domain specific corpuses. The use of the algorithm COALS for constructing semantic dictionary is presented. The COALS is crucial for reducing the number of word pairs, which an expert need to consider to define relations between them. On Fig. 1 results of the COALS is shown for some interesting words. In subsection 3.5 some relations are defined however an expert is encouraged to create their own. In section 4 the procedure for supplementing knowledge base is presented using named entity recognition and relation extraction. For further research on supplementing knowledge base, methods of machine translation can be incorporated.

6. ACKNOWLEDGEMENT

We thank the authors of the FidaPLUS corpus, who grant us its use for morphological tagging and lemmatization.

7. REFERENCES

[1] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3–26, 2007.

[2] A. Schutz and P. Buitelaar, "Relext: A tool for relation extraction from text in ontology extension," in *The Semantic Web–ISWC 2005*, pp. 593–606, Springer, 2005.

[3] E. Tutubalina and V. Ivanov, "Unsupervised approach to extracting problem phrases from user reviews of products,"
[4] A.-M. Popescu and O. Etzioni, "Extracting product features and opinions from reviews," in *Natural language processing and text mining*, pp. 9–28, Springer, 2007.
[5] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller, "Introduction to wordnet: An on-line lexical database*," *International journal of lexicography*, vol. 3, no. 4, pp. 235–244, 1990.
[6] "Dbpedia." http://dbpedia.org. Accessed: 2014-09-25.
[7] R. Xu, K. Supekar, A. Morgan, A. Das, and A. Garber, "Unsupervised method for automatic construction of a disease dictionary from a large free text collection," in *AMIA Annual Symposium Proceedings*, vol. 2008, p. 820, American Medical Informatics Association, 2008.
[8] R. Navigli and S. P. Ponzetto, "Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network," *Artificial Intelligence*, vol. 193, pp. 217–250, 2012.
[9] M. Morsey, J. Lehmann, S. Auer, C. Stadler, and S. Hellmann, "Dbpedia and the live extraction of structured data from wikipedia," *Program: electronic library and information systems*, vol. 46, no. 2, pp. 157–181, 2012.
[10] K. Lund and C. Burgess, "Producing high-dimensional semantic spaces from lexical co-occurrence," *Behavior Research Methods, Instruments, & Computers*, vol. 28, no. 2, pp. 203–208, 1996.
[11] T. K. Landauer, P. W. Foltz, and D. Laham, "An introduction to latent semantic analysis," *Discourse processes*, vol. 25, no. 2-3, pp. 259–284, 1998.
[12] M. Sahlgren, "An introduction to random indexing," in *Methods and applications of semantic indexing workshop at the 7th international conference on terminology and knowledge engineering, TKE*, vol. 5, 2005.
[13] D. L. Rohde, L. M. Gonnerman, and D. C. Plaut, "An improved method for deriving word meaning from lexical co-occurrence," *Cognitive Psychology*, vol. 7, pp. 573–605, 2004.
[14] W. D. Lewis, "Measuring conceptual distance using wordnet: the design of a metric for measuring semantic similarity," 2001.
[15] "Apache. tika.." http://tika.apache.org/. Accessed: 2014-09-25.
[16] H. Schmid, "Probabilistic part-of-speech tagging using decision trees," in *Proceedings of the international conference on new methods in language processing*, vol. 12, pp. 44–49, Manchester, UK, 1994.
[17] "Fidaplus." http://www.fidaplus.net. Accessed: 2012.
[18] G. Golub and W. Kahan, "Calculating the singular values and pseudo-inverse of a matrix," *Journal of the Society for Industrial & Applied Mathematics, Series B: Numerical Analysis*, vol. 2, no. 2, pp. 205–224, 1965.
[19] J. B. Kruskal, "Nonmetric multidimensional scaling: a numerical method," *Psychometrika*, vol. 29, no. 2, pp. 115–129, 1964.

Am I overtraining? A novel data mining approach for avoiding overtraining

Iztok Fister Jr.
University of Maribor
Faculty of Electrical Engineering and Computer Science
Smetanova 17, 2000 Maribor
Slovenia
iztok.fister1@um.si

Goran Hrovat
University of Maribor
Faculty of Electrical Engineering and Computer Science
Smetanova 17, 2000 Maribor
Slovenia
goran.hrovat@um.si

Samo Rauter
University of Ljubljana
Faculty of Sport
Gortanova 22, 1000 Ljubljana
Slovenia
samo.rauter@fsp.uni-lj.si

Iztok Fister
University of Maribor
Faculty of Electrical Engineering and Computer Science
Smetanova 17, 2000 Maribor
Slovenia
iztok.fister@um.si

ABSTRACT

Overtraining is one of the biggest problems in process of sport training. Especially, freshmen's and amateur athletes who do not have enough knowledge about behavior of their body, training and who do not have personal trainers encounter overtraining. So far some theoretical and practical solutions for avoiding overtraining have been arisen. In this paper, we propose a novel automatic solution which helps athletes to avoid overtraining. It is based on data mining which is applied to athlete's workout history. First experiments and simulations showed the promising results.

Keywords

data mining, apriori, sport training, overtraining

1. INTRODUCTION

Recently, an expansion of massive sport events and competitions have been emerged. These events capture usually sport disciplines, like running, cycling and triathlon [1]. Consequently, more people joined to a sport and healthy lifestyle. In fact, a lot of health and medicine organizations in the world also encourage people to participate in such events and live a healthy lifestyle. When people start to deal with sport, they do not have any special goals. There, only one goal is presented, i.e., I have to finish a specific sport race/event in which I invested so much effort! However, when someone finish more races, he/she want to go forward and compare his/her achievement with other people. This stage naturally leads to a competition. To compete in sport and fight for a medals is not easy anymore. Firstly, athletes have a devotion to the sport and in line with this assume themselves tremendous amount of work. Secondly, they have to build/create a good and effective training sessions. To finish race or to win a race is a huge difference and this difference is usually reflected in training, eating and resting. When amateur athletes start to deal with a sport, they do not have enough knowledge about these three components of each preparation for sport competitions:

- Training: Usually, the following set questions is asked by athletes. What is a proper training? When I have to train? How much to train? What kind of training should I train? Are intervals better than long distance training? Where should I train? With whom should I train?
- Eating: The following question are to be answered in line with this. What I have to eat? Why I need to eat so much carbohydrates? Do I need to increase vegetables and fruits intake? Is it good to drink any powders or isotonic drinks? Why do I need to avoid eating sugar? Is it good to take additional proteins into my food? Why eating a lot of fish is useful?
- Resting: Here, the following questions could be emerged, e.g., Do I need to take a rest every week? When I need rest and free days? What is a good resting? Can I still have a simple and short run/cycle? Swimming helps for relaxation of muscles and body?

These three components define well trained athlete. Training and resting are equally important. If you train a lot you get overtraining. On the other hand, you cannot be trained enough and suffer in the races. Finally, eating has a great

impact to athlete's performance. It is important for recovery and regeneration. Nowadays, overtraining is still one of the biggest problems in sport. Almost every freshman encounter this problem. When freshman encounter this problem, they usually suffer for psychological problems in mind. Moreover, it is very hard to overtake this problem. So far many theoretical, practical and abstract solutions have been developed to deal with this problem. In this paper we propose a new solution to help athletes to avoid overtraining. Data mining methods are applied on athlete's workouts which were created by sport trackers. Our solution send an alarm to athletes before overtraining. Therefore, they can change a training or take a more rest and prevent themselves from overtraining. The paper is structured as follows: in the next section we describe some basics about sport training. The third section outlines overtraining, while section 4 reviews current overtraining prevention methods. Section 5 describes data mining methods. Section 6 is devoted to the sport trackers. Section 7 proposes a novel solution and the last section conclude the paper with some remarks for future work.

2. SPORT TRAINING

Definition of Sports Training is based on scientific and pedagogical principles of planning the systematic activities or processes in order to achieve the highest results in the chosen sports discipline. The final effect of the systematic training process can be manifested as

1. athlete's well sport form,
2. the increased capacity of the athlete's organism, or in the worst case
3. overtraining syndrome.

Sport form can be described as a phenomenon of short-term increased capacity of the athletes organism in relation to the expected competitive capacity that is perceived on the subjective level. In such conditions athlete feels that he/she overcomes a certain load of sport activity with a little effort or that he/she is able to overcome a higher load at the maximum effort [2]. In simple terms, it means "to be in the best shape at the right time." Efficiency of the process of sports training also means that the athlete rationalize the time devoted to the training [3]. All this is happening in a real process of sports training, where in order to achieve sports form, athletes include different methods in the process of sport training and intensity of workload. Also the rest period is very important. For instance, Table 1 presents an example of proper training two weeks before half marathon race.

2.1 Overtraining

Overtraining is a special state of an athlete occuring when someone undergoes a bigger amount of intensity training and consequently body is not able to recover. In such state, athletes are not able to compete on high level because their body is exhausted. Overtraining leads to overtraining syndrome. Overtraining syndrome manifests itself in physiological and psychological symptons and has an effect on athlete's performance [4]. These physiological and psychological indicators as proposed in [4] and presented in Table 2.

Physiological	Psychological
Higher resting heart rate	Sleep disturbances
Changes in normal blood pressure	Loss of self-confidence
Delayed return to normal heart rate	Drowsiness and apathy
Elevated basal metabolic rate	Irritability
Elevated body temperature	Emotional and motivational imbalance
Weight loss/excessive thirst	Excessive, prolonged weariness
Impedded respiration	Lack of appetite
Subcostal aching	Fatigue
Bowel disorders	Depression
	Anxiety
	Anger/hostility
	Confusion

Table 2: Physiological and Psychological indicators of overtraining syndrome

	Causes
1	Length of the competitive season
2	Monotony of training
3	Feelings of claustrophobia
4	Lack of positive reinforcement
5	Feelings of helplessness
6	Abusiveness from authorities
7	Stringent rules
8	High levels of competitive stress

Table 3: Causes of overtraining syndrome as proposed in [4]

As it can be seen from Table 2, there are many indicators telling the athletes are overtrained. However, better than curing overtraining is avoiding overtraining. In [4], authors presented some causes of overtraining syndrome.

A lot of research were done on overtraining in the past, e.g., in [5, 6, 7, 8, 9].

2.2 Current methods of avoiding overtraining

The main herald of overtraining syndrome is a hard training and an inadequacy of the rest. People transferred training or competition stress differently. Athlete tolerance on training stress varies throughout the season periods. In line with this, a training process should be adapted and varied through the season period. Especially, the right strategy in the competition period is very important. Too much high intensity workout with too short rest period may results in bad results at the competition or in the worst case may leads to overtraining. To avoid overtraining syndrome is not so easy. Athletes are most of the time very tired. It is hard to distinguish when the first signs of fatigue happen. Prevention requires good nutrition, complete hydration, and rest periods between exercises themselves [7]. The effective and good organize periodization process is necessary to ensure adequate adaption of the athlete organism to the requirements of the

DAY	METHODS	DURATION	INTENSITY
1	Race		
2	Easy jogging with some acceleration	30 min	Low
3	Rest day		
4	Running (interval training 2 x 10 min fast; between sets 5 min - 8 min of easy jogging)	45 min	Low & High
5	Rest day		
6	Running (interval training 4 x 4 min fast; between sets 3 min - 4 min of easy jogging)	45 min	Low & High
7	Running (interval training 2 sets of 3 - 5 x 90 sec accelaration)	45 min	Low & High
8	Rest day		
9	Rest day		
10	Endurance running	90 min	Low
11	Endurance running	75 min	Low
12	Rest day		
13	Endurance running	105 min	Low
14	Running (interval training 2 sets of 3 x 2 min very fast with 1 min recovery; between sets 5 min - 8 min of easy jogging)	75 min	Low & High

Table 1: Example of training for 21 km race (last 14 days before the competition)

competition in chosen sport discipline.

To avoid overtraining syndrome it can help an individual monitoring of:

- achievements,
- mood states and
- heart rate variability.

To overcome the overtraining syndrome it can help:

- low intensity workout, rest and relaxation strategy;
- exercise with very short high intensity sprints with long rest/low intensity period, the confidence to progress well.

2.3 Monitoring the sport activities

Recently, training sessions of athletes are monitored with sport trackers. Sport trackers are a combination of software and hardware which are used for tracking activities during sport workouts. Hardware is a smart phone, while software are applications which run on smart phones. For a more detailed description of sport trackers, readers are invited to read papers [10, 11].

3. DATA MINING OF MONITORED DATA

Data obtained during the sport session of athletes can be transferred to personal computers in a form of TCX data sets. Normally, these data sets are analyzed using different data mining methods. In computer science, the data mining (DM) is a method, where the main goal is extraction of information from a data sets and conversion of these data to a form understandable by humans. The more frequently used methods in data mining are: clustering, classification, regression, association rule mining. Data mining methods involve also some machine learning methods like decision trees, logistic regression, support vector machines, etc. But nowadays also some nature-inspired algorithms [12, 13] are employed in the data mining field. In data mining, a broad spectrum of many applications has been developed, e.g., these methods have been used for weather prediction, fraud detection, sensor data mining, surveillance and many more.

3.1 Association rule mining

One of the data mining tasks is association rule mining. It was discovered by Agrawal and Srikant [14, 15] who developed Apriori. Other algorithms for association rule mining also emerged e.g. FP-growth [16], which perform faster. Association rule mining is a task of discovering rules from set of transactions. The well known example is basket market analysis however association rule mining can be applied for other datasets such as sports data. Each transaction is consisted of items and is also called item-set. Discovered rules are the form $X \Rightarrow Y$, where X is subset of items and Y is usually only one item. As a result many rules are discovered, for this reason a lot of measures are proposed to evaluate and rank these rules. The most known interestingness measures are support and confidence [14], others are lift [17], all-confidence [18], conviction, leverage, χ^2 [19] etc. Support is proportion of transactions containing items of a rule and confidence is defined as $conf(X \Rightarrow Y) = supp(X \Rightarrow Y)/supp(X)$. In our case transactions is consisted of items e.g. training distance, average heart rate, motivation, eat-

ing, sleeping, etc., and an example of rule which can be discovered is $('MOTIVATION','EATING') \Rightarrow ('SLEEPING', 1.000)$, where confidence and support are 1 and means that everyone who well eats and is motivated also well sleeps.

4. PROPOSED METHOD FOR AVOIDING THE OVERTRAINING

Our proposed method consists of the following steps:

- definition and design of framework,
- discretization,
- creating transactions,
- applying Apriori algorithm and
- interpretation of results.

In the remainder of the paper, these steps are illustrated in detail.

4.1 Definition and design of framework

At first, a framework needs to be defined that enable an athlete to store, maintain and use data. Let us suppose that after every training session athlete uploads a file in TCX format. This file is automatically parsed and data (total duration, total distance, average heart rate, max heart rate) are stored into the database. During uploading the file, athlete needs to answer to the five predefined questions that characterize current training session. These questions are:

1. Did athlete train intervals? (Possible answer values: YES, NO)
2. Did athlete feel any special pains during the training session? (Values: YES, NO)
3. Do athlete have a great motivation for future trainings? (Values: YES, NO)
4. Did athlete sleep well today? (Values: YES, NO)
5. Can atlete eat normally? (Values: YES, NO)

When athlete answers to these questions, he/she is linked with today's workout.

4.2 Discretization

When all data are collected a discretization process is performed. In this process, numerical values have to be discretized, since Apriori algorithm works only with categorical values. An example of discretized attributes is presented in Table 4.

4.3 Creating transactions

After attributes are discretized, a transactions are created. In our case, we consider one transaction for one training. Therefore, after 2 months of training, there will be around 45 transactions. More transactions we have, more accurate rules can be inferred and in line with this also the overtraining can be predicted more precisely.

4.4 Applying Apriori algorithm

In this step, Apriori algorithm is launched using the prepared data (transactions). Apriori algorithm discovers rules as a result. For example, a rule $('MOTIVATION', 'EATING') \Rightarrow ('SLEEPING', 1.000)$ means that athlete who has a motivation and can eat well, can also sleep well.

4.5 Interpretation of results

The last step in the proposed approach is an interpretation of results. From the rules which we get with Apriori algorithm, we want to get a real situation.

5. EXPERIMENTS AND SPORT INTERPRETATION OF RULES

The first experiments of the proposed method for predicting the overtraining were conducted on a real dataset. This dataset was produced during the training sessions of a professional mountain biker. The total number of workouts/transactions were limited to 50. In Figure 1, some transactions are presented that were used in our experiments.

```
SHORT_RIDE,SHORT_DURATION,HIGH_RATE,INTERVAL,NO_PAINS,MOTIVATION,NO_SLEEPING,EATING
MEDIUM_RIDE,MEDIUM_DURATION,MEDIUM_RATE,NO_INTERVAL,NO_PAINS, MOTIVATION,NO_SLEEPING,NO_EATING
LONG_RIDE,LONG_DURATION,MEDIUM_RATE,NO_INTERVAL,NO_PAINS,MOTIVATION, SLEEPING,EATING
MEDIUM_RIDE,SHORT_DURATION,HIGH_RATE,INTERVAL, NO_PAINS,MOTIVATION,SLEEPING,NO_EATING
LONG_RIDE,LONG_DURATION,LOW_RATE,NO_INTERVAL,NO_PAINS,MOTIVATION,SLEEPING,NO_EATING
```

Figure 1: Example of transactions

From experiments, interesting results were obtained. For instance, the rules are inferred from the transactions, as follows.

1. Rule: $('EATING',) \Rightarrow ('MOTIVATION', 1.000)$ Interpretation: From this rule we can see that proper eating is connected with motivation. If we eat a lot we also have a motivation.
2. Rule: $('NO\_MOTIVATION',) \Rightarrow ('NO\_SLEEPING', 1.000)$ If athlete does not have a motivation, he can not sleep. It might means that he is overtrained.
3. Rule: $('LONG\_RIDE',) \Rightarrow ('NO\_INTERVAL', 1.000)$ From this rule we can see what is usually in practice. If we ride a long rides, we do not do intervals, but going the distance only.
4. Rule: $('SLEEPING',) \Rightarrow ('MOTIVATION', 1.000)$ Here we see can again that sleeping and motivation are connected. If athlete can sleep well, then he is very motivated and for sure not overtrained.
5. Rule: $('NO\_MOTIVATION',) \Rightarrow ('NO\_EATING','NO\_SLEEPING', 1.000)$ In this rule we see that athlete who does not have a motivation, can not eat and sleep. In this case, he is very overtrained.
6. Rule: $('LONG\_RIDE','NO\_EATING') \Rightarrow ('NO\_INTERVAL', 1.000)$ This rule defines long ride and that athlete can not eat. We can also see that in this case athlete were not doing an intervals.

From previous rules, it can be seen some rules which were discovered from a real dataset. In some cases we can see how to determine overtraining in rules. However, after running

Attribute	Current values	Discretized values	How to discretize?
Distance	Numerical (km)	LONG_RIDE MEDIUM_RIDE SHORT_RIDE	> 120 km ≤ 120 km and > 50 km ≤ 50 km
Duration	Numerical (min)	LONG_DURATION MEDIUM_DURATION SHORT_DURATION	> 300 min ≤ 300 min and > 150 min ≤ 150 min
Average heart rate	Numerical (BPM)	HIGH_RATE MEDIUM_RATE LOW_RATE	> 170 BPM ≤ 170 BPM and > 130 BPM ≤ 130 BPM
Intervals training?	Categorical (Yes, No)	INTERVAL NO_INTERVAL	
Pains?	Categorical (Yes, No)	PAINS NO_PAINS	
Motivation?	Categorical (Yes, No)	MOTIVATION NO_MOTIVATION	
Sleeping?	Categorical (Yes, No)	GOOD_SLEEPING BAD_SLEEPING	
Eating?	Categorical (Yes, No)	GOOD_EATING BAD_EATING	

Table 4: Discretization of our attributes

LONG_RIDE	MEDIUM_DURATION	HIGH_RATE	PAINS	MOTIVATION	GOOD_SLEEPING	NO_INTERVAL	BAD_EATING

Table 5: An example of one transaction

more experiments we encounter that more transactions we have, more rules we can obtain. In line with this, we do not obtain only rules connected with overtraining, but also some others tackling the basic habits of athletes, like rule 1. On the other hand, rule 5 shows a basic example of overtraining.

6. CONCLUSION

In this paper we developed a prototype solution for predicting overtraining of an athlete. We used an Apriori data mining method and applied it to the real dataset that was created by mountain biker. Experiments showed that this method is interesting for such manners. However there are also some limitations like size of the dataset. In our case we took 50 trainings into the account to get the first picture of an athlete. In the future, we will also perform more tests with different athletes.

7. REFERENCES

[1] Samo Rauter. Mass sports events as a way of life (differences between the participants in a cycling and a running event). *Kinesiologica Slovenica*, 2014.

[2] Kuno Hottenrott, Sebastian Ludyga, and Stephan Schulze. Effects of high intensity training and continuous endurance training on aerobic capacity and body composition in recreationally active runners. *Journal of sports science & medicine*, 11(3):483, 2012.

[3] Riggs J Klika, Mark S Alderdice, John J Kvale, and Jay T Kearney. Efficacy of cycling training based on a power field test. *The Journal of Strength & Conditioning Research*, 21(1):265–269, 2007.

[4] Mary Black Johnson and Steven M Thiese. A review of overtraining syndromeâĂŤrecognizing the signs and symptoms. *Journal of athletic training*, 27(4):352, 1992.

[5] Laurel T MacKinnon. Overtraining effects on immunity and performance in athletes. *Immunology and cell biology*, 78(5):502–509, 2000.

[6] Shona L Halson and Asker E Jeukendrup. Does overtraining exist? *Sports medicine*, 34(14):967–981, 2004.

[7] Lucille Lakier Smith. Overtraining, excessive exercise, and altered immunity. *Sports Medicine*, 33(5):347–364, 2003.

[8] Dianna Purvis, Stephen Gonsalves, and Patricia A Deuster. Physiological and psychological fatigue in extreme conditions: overtraining and elite athletes. *PM&R*, 2(5):442–450, 2010.

[9] Karen Birch and Keith George. Overtraining the female athlete. *Journal of Bodywork and Movement Therapies*, 3(1):24–29, 1999.

[10] Iztok Fister, Duan Fister, and Simon Fong. Data mining in sporting activities created by sports trackers. In *Computational and Business Intelligence (ISCBI), 2013 International Symposium on*, pages 88–91. IEEE, 2013.

[11] Iztok Fister, Dušan Fister, Simon Fong, and Iztok Fister Jr. Widespread mobile devices in applications for real-time drafting detection in triathlons. *Journal of Emerging Technologies in Web Intelligence*, 5(3):310–321, 2013.

[12] Iztok Fister Jr, Xin-She Yang, Iztok Fister, Janez Brest, and Dušan Fister. A brief review of nature-inspired algorithms for optimization. *Elektrotehniški Vestnik*, 80(3):116–122, 2013.

[13] Simon Fong. Opportunities and Challenges of Integrating Bio-Inspired Optimization and Data Mining Algorithms. In Xin-She Yang, Zhihua Cui, Renbin Xiao, Amir Hossein Gandomi, and Mehmet Karamanoglu, editors, *Swarm Intelligence and*

Bio-Inspired Computation, pages 385–402. Elsevier, 2013.

[14] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM, 1993.

[15] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[16] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *ACM SIGMOD Record*, volume 29, pages 1–12. ACM, 2000.

[17] Sergey Brin, Rajeev Motwani, Jeffrey D Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market basket data. In *ACM SIGMOD Record*, volume 26, pages 255–264. ACM, 1997.

[18] Edward R Omiecinski. Alternative interest measures for mining associations in databases. *Knowledge and Data Engineering, IEEE Transactions on*, 15(1):57–69, 2003.

[19] Craig Silverstein, Sergey Brin, and Rajeev Motwani. Beyond market baskets: Generalizing association rules to dependence rules. *Data mining and knowledge discovery*, 2(1):39–68, 1998.

An Improved Algorithm of DPM for Two-dimensional Barcode

Tao Gao
North China Electric Power University
Department of Automation
Baoding, Hebei Province, 071003
China
gaotao81@foxmail.com

Xiao-cheng Du
North China Electric Power University
Department of Automation
Baoding, Hebei Province, 071003
China

Iztok Fister Jr.
University of Maribor
Faculty of Electrical Engineering and Computer Science
Smetanova 17, 2000 Maribor
Slovenia
iztok.fister1@um.si

ABSTRACT
Data Matrix is a two-dimensional matrix code which has many advantages like as large information density, high capacity, and small size and so on. It is widely used in industrial automation, logistics, transportation and other fields. A two-dimensional barcode includes printing and DPM barcodes, which depending on the differences of the application at backgrounds of the two-dimensional barcode. DPM is the abbreviation for Direct Part Mark and has some weaknesses, like low contrast, more noise jamming, complicated background, uneven illumination during the data appreciation process. The main mission of this paper was to put forward a series of image preprocessing methods which links binaryzation and morphological transformation based on handheld device, and achieve adaptive smoothing fuzzy and adaptive morphological transform through the DPM detection. The experimental results show that the method can overcome problems such as too large middle gap, the uneven illumination and noise.

Keywords
Data Matrix Barcode, Binary Image Processing, Direct Part Mark

1. INTRODUCTION
Two-dimensional barcode is a neotype barcode technology based on a one-dimensional barcode, and according to criteria stores data and symbolic information on special geometries using black and white, which are distributed along the horizontal and vertical areas of two-dimensional planar space. Due to some specialties regarding significant information storage capacity, good robustness and low cost [1] two-dimensional barcoding has been applied gradually within the commercial, transportation, finance, health care and other fields.

2. DATA MATRIX BARCODE
Presently, the more commonly used international two-dimensional barcode includes the Data Matrix, PDF417, QR code, etc. The Data Matrix has the minimum size of all the barcodes, and is especially suitable for small parts logos and can be printing on to the entity directly, so that the sign is widely used for small objects like drugs, integrated circuits, and manufacturing processes from production lines [2].

2.1 The structure of the data matrix 2D barcode
The Data Matrix of a two-dimensional barcode is shown in Figure 1. Its symbol's structure is composed of position area and data areas. The position area includes an 'L' solid boundary and an 'L' dotted boundary as shown in Figure 2, and has a dead zone with one data unit. Position area is the Data Matrix of border the two-dimensional barcode, and is mainly used for limiting the physical size, orientation and symbol distortion of DM. The 'L' dotted boundary is mainly used for finite unit structure of a symbol as well, which can also help to determine physical size and distortion. As shown in Figure 3, the data area contains the symbols which to be encoded, and contain coded information, like Numbers, letters and Chinese characters according to certain encoding rules. The DM code is the lattice and consists of two kinds of color, a black and white combination, and every black or white square with the same size is a unit of data, representing a binary 1 and a binary 0 [5].

Figure 1: Data matrix of two-dimensional barcode

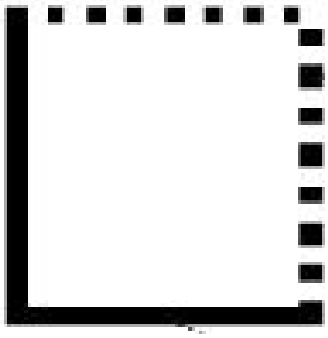


Figure 2: Position area of DM

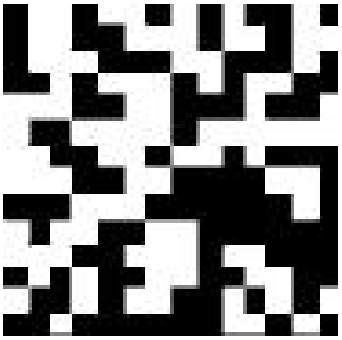


Figure 3: Data area of DM

2.2 DPM barcode

DPM is the abbreviation of Direct Part Mark. It was originally used for machinery and electronics industrial parts, and can record lots of information about parts like production, quality inspection. The factory embosses the DPM into an image by means of an etching method like the laser etching is usage can be extended to automobile manufacturing, pharmaceutical, medical, military firearms management, and so on. So the DPM barcode is a kind of important information within the Internet of technology. The DPM mark is the main carrier of the two-dimensional barcode because the two-dimensional barcode has certain characteristics like large coding capacity, high density, high information security. Compared with the two-dimensional barcode printed on paper, the generating method of the DPM bar code varies, like ink-jet printing. In addition, it can also be generated by certain other methods like laser etching, striking a hitting machine, and electrochemical corrosion. The material of the parts with DPM barcode sculpture varies from cast iron, aluminum, glass, hard plastic, to wood.

3. THE SOURCE CODE

Barcode recognition technology has been widely used, and the main open source code includes 'zxing' open source code and 'zbar' open source code. In this section, the improvement is based on the 'zxing' open source code. In the open zxing source code, there are certain advantages as follows: it can be installed within intelligent phones, and it not only has high speed for identification but also has a short recognition run-time. In addition, it can identify the one-dimensional and two-dimensional barcodes at the same time, and can also search online for products according to the recognized barcode. So it is highly convenient for developers to use this barcode.

3.1 The Decoding Process of The 'Zxing' Open Source Code

In the open 'zxing' source code, it needs to open its build camera first, and then photograph the barcode to be identified. The color model of the image obtained is a RGB color model. However, the RGB color space has some disadvantages like: No intuitive, asymmetric, and dependency on the hardware device [3]. So, the RGB color model should be converted to YUV color model, for the reason that it is important that the luminance Y is separate from the chroma U and V in the YUV color space. The formula for the RGB color model converting into YUV color model is as follows.

- Y=0.299R+0.587G+0.114B
- U=-0.147R-0.289G+0.436B
- V=0.615R-0.515G-0.1B

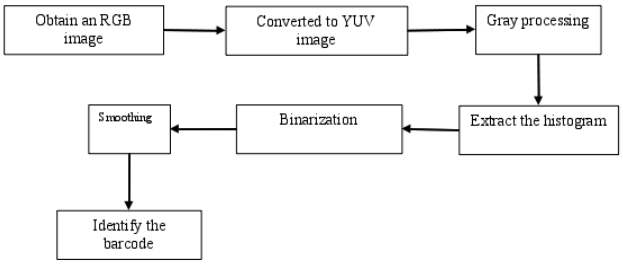


Figure 4: The basic process of zxing source code

After obtaining the YUV image, it is necessary to convert the image into a grayscale image, and then convert it into a binary image. In addition, the histogram needs to be obtained at the same time, for then to identifing the barcode. The description above is the whole barcode image processing of the open source code identification of 'zxing'. The basic process of 'zxing' source code is shown in Figure 4.

3.2 Program flowchart

The method put forward in this paper is based on "Z-Xing" project which is an open source two-dimensional code recognition project of Google, its main purpose is to do some pretreatment to the initial two-dimensional code image. It is achieved the image binarization by referring to the binarization of Yang Shu, and attaching some appropriate process to dotted data matrix bar code processing. The size of the point module will be get by the process of blob detection. Combined with morphological processing the image can be identified, the program flow chart shown in Figure 6. Among them, it is necessary to focus on these folders mainly with the names "android", "camera", "encode", and "result". The process of the initialization program is as follows: firstly, to load the main activity and create the object of the Capture Activity Handler in this class, and then is object starts the camera to realize the automatic focusing. Secondly, to create Decode handler threads, and then create the object of a

Decode handler, and this object then obtains original byte data from the camera, all the above during the first stage. After obtaining the data, it parses out the two-dimensional barcode, and then parses out the character of the barcode, at the same time removing the characters without analysis to be handle by 'Capture Activity Handlers'. This class is called the 'decode function' of the main activity used for completing the analyse of the characters, and the image is finally displayed on the screen. In this way, it completes the parsing of the barcode.

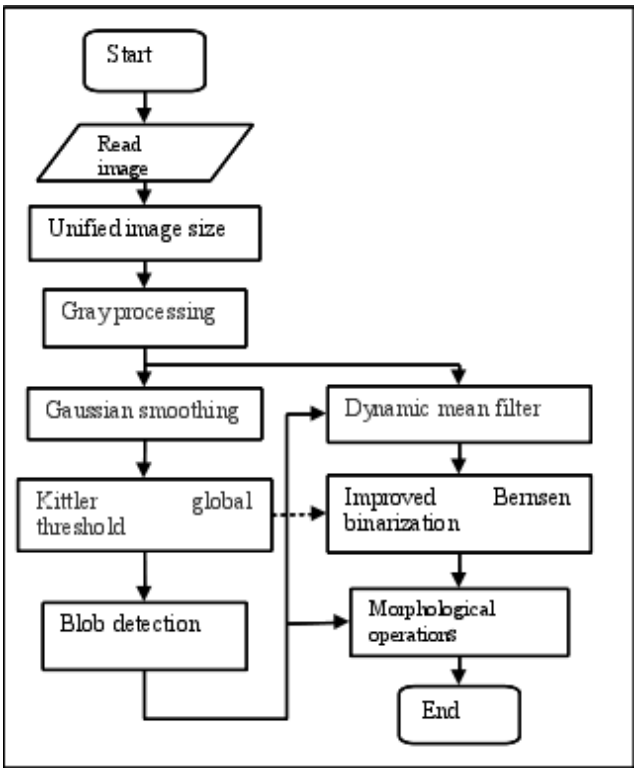


Figure 5: The basic process of zxing source code

4. BINARY PROCESSING OF DATA MATRIX CODES

4.1 The binary image processing

The key to binarization is to define the boundary between black and white. The image has been transformed into gray image, with every point represented by a gray value, which is needed for defining a gray value (namely threshold). If the value of the point is greater than that value, it can be defined as white (1), otherwise, it is defined as black (0).

4.2 The improvement of a binarization algorithm

As shown in Figure 6, in 'zxing' barcode, the binarization processing is implemented in Binarizer class, where the one-dimensional code uses the **get-BlackRow** method, and the two-dimensional code uses **getBlackMatrix** method. There are two classes generated from Binarizer class: **Global-Histogram-Binarizer** and **Hybrid-Binarizer**. The implementations of the **getBlackMatrix** method for these two classes are different, so we modify the binarization process briefly for the **Global-Histogram-Binarizer** class. In the improved **Global-Histogram-Binarizer** class, the threshold is calculated by the kittler algorithm. Using the kittler binarization algorithm, we deal with the gray images of the DataMatrix barcodes during binarization processing, and then the global threshold (T) can be obtained. The kittler binarization processing formula is: For pages other than the first page, start at the top of the page, and continue in double-column format. The two columns on the last page should be as close to equal length as possible.

$$T = \sum_x \sum_y e(x,y) f(x,y) / \sum_x \sum_y e(x,y) \qquad (1)$$

In Equation (1), $f(x,y)$ is the original gray image, $e(x,y) = max\{|e_x|, |e_y|\}$ represents the maximum gradient. In $e(x,y)$, $e_x = f(x-1,y) - f(x+1,y)$ represents the horizontal gradient, $e_y = (x,y-1) - f(x,y+1)$ represents the vertical gradient [4]. The relevant procedure is shown in Figure 7.

```
public BitMatrix getBlackMatrix() throws NotFoundException {
  LuminanceSource source = getLuminanceSource();
  int width = source.getWidth();
  int height = source.getHeight();
  BitMatrix matrix = new BitMatrix(width, height);
  byte[] localLuminances = source.getMatrix();
  int threshold = estimateThreshold(localLuminances,width,height);
  for (int y = 0; y < height; y++) {
    int offset = y * width;
    for (int x = 0; x< width; x++) {
      int pixel = localLuminances[offset + x] & 0xff;
      if (pixel < threshold) {
        matrix.set(x, y);
      }
    }
  }

  return matrix;
}
```

Figure 6: The improved binarization process

```
private static int estimateThreshold(byte[] localLuminances,int width,
        int height) throws NotFoundException
{
    int E=0;
    int EF=0;
    for(int y=1;y<height-1;y++)
    {
        int offset = y * width;
        for(int x=1;x<width-1;x++)
        {
            int grey = localLuminances[offset + x] & 0xff;
            int grey1 = localLuminances[offset + x-1] & 0xff;
            int grey2 = localLuminances[offset + x+1] & 0xff;
            int grey3 = localLuminances[offset + x-width] & 0xff;
            int grey4 = localLuminances[offset + x+width] & 0xff;
            int Ex=grey1-grey2;
            int Ey=grey3-grey4;
            int ex=Math.abs(Ex);
            int ey=Math.abs(Ey);
            int exy=Math.max(ex,ey);
            E+=exy;
            EF+=exy*grey;
        }
    }
    int threshold;
    threshold=((EF/E)-1)/255;
    return threshold;
}
```

Figure 7: Kittler algorithm

5. EXPERIMENTS

The process of test uses the Huawei smartphone as a carrier. The main parameters of the phone are as follows: the CPU is mediatek MT6592M, the frequency is 1433 MHZ, memory is 2 GB. We deal with Data Matrix barcode on different specifications, color, clarity, and 34 DPM pictures are tested. In this section, we identified the Data Matrix barcode by soft wares created from the 'Z-xing' package which have been improved and haven't been improved respectively, and compare the recognized effects with each other. The results are shown

Software	Original	Improved
Number of images	34	34
Number of successful images	19	25
Success rate	55.9%	73.5%
Number of images with the brightness difference	17	17
Number of successful images	7	11
Success rates	41.2%	64.7%
Number of DM barcode images (2*2)	5	5
Number of successful images	2	4
Success rates	40%	80%
Number of images sampled from artifacts directly	12	12
Number of successful images	7	9
Success rate	58.3%	75%

Table 1: Comparison

in Table 1. Because the method will be used in handheld device finally, so the method need have more instantaneity. So in the experiment, shoot the image from different angles for 100 times, then count average correct recognition rate and speed. Through analyzing the experimental result, it can be known that the elapsed time is relevant to the contrast, roughness and illumination of image. The method sacrifices efficiency for accuracy because of using improved Bernsen method. The elapsed time is acceptable through comparing the time with the elapsed time of standard DataMatrix. Counting the time that each part spends through simulation, it can be reached that the improved Bernsen method takes more time when the spot detection takes second place. The size of image does not much affect the result because the program adds size unitizing method.

6. DISCUSSION AND CONCLUSION

It can be seen from Table 1, that after improving 'zxing' binarization, the success rates of various types of Data Matrix barcode recognition were improved, especially in the test with the brightness difference, and the effect was very obvious, which embodies the characteristics of the kittler algorithm. It can also more effectively find the area of uneven illumination. The improved algorithm is more stable and adaptive, so as to improve the success rate of recognition. In addition, the improved 'zxing' still has some problems:

- It can be seen from Table 1, that the differences between the success rates are higher, and the difference is around 20 percents, some even higher, and this is due to the pictures being a slightly less. Table 1 can reflect the effect of improvement, but it is not as great as the numbers reflected in the Table.
- In the identification process of 2 X 2 DM code or 1 X n DM code, the recognition rate had improved after the improvement, but certain process of image recognition took a long time, so this kind of barcode recognition effect of the improved 'zxing' is still unsatisfactory.
- When we recognized the pictures that sampled from the artifacts directly, we needed to extract the tested sections manually, otherwise, the recognition effect was bad, in other word, and it was easily affected by the background.

7. ACKNOWLEDGEMENTS

This work is supported by National Natural Science Foundation of China (No. 71102174 and No. 51306058), the Fundamental Research Funds for the Central Universities (No. 2014QN46).

8. REFERENCES

[1] C.-H. Cheung and L.-M. Po. Novel cross-diamond-hexagonal search algorithms for fast block motion estimation. *Multimedia, IEEE Transactions on*, 7(1):16–22, 2005.

[2] T. Jinzhao and Q. Jie. Fast and precise detection of straight line with hough transform. *Journal of Image and Graphics*, 13(2):234–237, 2008.

[3] Q. Q. Ruan and Y. Z. Ruan. *Digital image processing (second edition)*. Electronic Industry Press, Beijing, 2005.

[4] S. Yang and Z.-h. Shang. A new binarization algorithm for 2d bar code image. *Journal of Kunming University of Science and Technology (Science and Technology)*, 1:011, 2008.

[5] Y.-x. Zou and G.-b. Yang. Research on image pre-processing for data matrix 2d barcode decoder. *Computer Engineering and Applications*, 34, 2009.

StuCoSReC

University of Primorska Press
www.hippocampus.si
ISBN 978-961-6963-03-9
Not for resale