On SOMA Parallelization with Android Devices

Michal BUKÁČEK

VSB - Technical University of Ostrava Department of Computer Science 17. listopadu 15, Ostrava-Poruba Czech Republic

*Corresponding Author: Michal Bukáček (Email: michal@bukacek.cz)
(Received: 28-May-2019; accepted: 11-Aug-2019; published: 30-Sep-2019)
DOI: http://dx.doi.org/10.25073/jaec.201933.247

Abstract. *Today's world is full of new, small, personal handhelds. They are called smartphones or tablets. The machines themselves always have less power than desktop computers or even mainframes which were left behind. Their computational power can be increased when they are joined together in a group and are addressing one common task. To check and demonstrate the possibility of the use of mobile devices being joined to a group, the SOMA algorithm was chosen. The well as known functions, for example; De Jong, Rosenbrock, Rastrigin or Schwefel will be used and their extremes (minimums) will be realised. The goal is to test the speed of these mobile devices to realise the extremes of more dimensional functions. The advantages and disadvantages of this swarm linking will be shown.*

Keywords

Soma, Android, Swarm, Parallel Algorithms.

1. Introduction

Android is Google's operating system which is designed primarily for mobile smartphones or tablets. Nowadays, there are more Android devices such as cameras, car radios, TV, toys, etc. The popularity of these devices is rising due to their utilities and technical parameters, light weight, mobility and price. A phone or tablet with GPS enabled has more options for navigation in the car, than the dedicated devices designed for that sole purpose. Mobile navigation on a phone can be both cheaper and more up to date. With all the benefits of a phone, having a camera or an electronic compass or perhaps almost anywhere online. No wonder the number of mobile devices grows every year, as shown in (The Radicati Group, Inc, 2016). Due to the electrotechnical advances, of this competitive environment, where the manufacturers of each device increase with better hardware because the OS is always the same. It is fair to say thatnot only the computing performance of these devices, but also their hardware, is constantly increasing. New technical capabilities (NFC, LTE, Bluetooth LE, Fingerprint readers, wireless charging, etc.) are being inovated and thus other applications with different combinations of functionality and then further use and then still further expansion.

In the future, Android will encompass the whole world and all the people around it.

The computing power of these mobile devices is too small compared to conventional computers and does not have a large memory capacity. So they can be at most compared with classical computers and not with supercomputers. For this reason, some companies, for example, regarding navigation applications, have calculated the shortest route on the server and internet navigation only sends the map data needed for that route and the coordinates of the junction points. There is a lot of unanswered question like :

- What if the apps should calculate the task individually?

- What if the app could work with the same app on other devices together to solve the same task?

To answer these questions, an example of looking for extremes of known features was used using the SOMA algorithm that was enhanced by network communication.

Extreme (minimum) for known functions First - De Jong and Schwefel's will be searched. The goal is to test the speed of these mobile machines on more dimensional functions.

First De Jong's function was initially introduced in his thesis entitled "An analysis of the behavior of a class of adaptive genetic systems". The formula of the function is shown as Eq. (1).

$$F_{First-DeJong}(X) = \sum_{i=1}^{D} X_i^2 \qquad (1)$$

Schwefel's function by author H.-P. Schwefel was introduced in his book "Numerical optimisation of computer models" in the year 1981. The formula of the function is shown as Eq. (2), where D is the dimension and $x = (x1, x2, \cdot \cdot \cdot, \mathrm{D})$ is a D-dimensional row vector.

$$\mathrm{F_{schwefel}(X)} = \sum_{i=1}^{D} -x_i Sin[\sqrt{Abs[x_i]}] \qquad (2)$$

2. EXPERIMENT DESIGN

2.1. Used algorithm

A used algorithm is SOMA. First, it is necessary to recall how the Soma algorithm works itself. Classical SOMA - Self-Organizing Migrating Algorithm describes [1] as a comparison of wolf populations seeking the greatest amount of food in their territory. The author of the algorithm has used this description for the general public and a detailed description of this algorithm can be found in [3,7,8,10].

The algorithm has several methods to achieve a solution: All To One, All To Random, All To All and All To All Adaptive. For simplicity, we'll repeat how the All to One algorithm works:

1) The algorithm consists of several repetitive cycles called migratory wheels. We have a group of agents (a group of wolves) of a certain size of population whose task is to find the greatest extreme in the definition area that is represented in the search space.

2) Is the agent that has the greatest gain, so it is in the definition field among other agents at the most interesting value of the function, it is the extreme among other agents - we will call this agent Leader in the following migration round. In nature, it would be a wolf standing at the largest piece of food.

3) The migration round begins. All agents will move in leaps to the Leader. Each time they change their position in the state space, they calculate the current functional value and if it is more extreme than the default, it will replace it as a new default.

4) Stop size means where the individual stops in migration it is given by the PathLength parameter when it is equal to 1, stopping directly on the leader position if the parameter equals; for example, two stops at a double distance. from the starting distance from where it started.

5) The movement itself after the jumps are affected, it can be said to be disturbed by the PRT vector, which is generated randomly as 0 and 1 when zero cancels the change in the given direction. The PRT vector makes the algorithm non-deterministic.

6) If all individuals except the leader arrive at their PathLength target, the migration round ends. A new leader is chosen, one who remembers the most extreme above all (he does not have to do it anymore and our old leader may have another round as a new leader) he is marked as Leader in the new round and the others are returned to the

bases of their starting positions, the places where they have found their local extreme.

7) A new leader is chosen, one who remembers the most extreme function value of all, he is marked as the Leader in the new round and the others are returned to the bases of starting positions. Places where they found their local extreme. Next a new migration round begins until the number of migratory rounds is over. The number of migrations is given by the Migration parameter.

8) If everything is set correctly and the definition space contains one global extreme. So the agents (or wolves), after the last migration round, concentrate only around the leader and show the position of the extremity.

The "All to Random" algorithm works. Similarly, only the leader is selected at random. The "All To All " algorithm works as follows: Everyone in each migration round will gradually become a leader and everyone migrates to every position. Each agent is once in the migration round Leader. It's not so much about the Leader, but about the agent, he's just migrating to be. This solution is the most accurate but demanding of computing power. "All To All Adaptive" is "All To All" the difference is that the individual does not move to the new position after each migration end but after the new Leader has changed.

These known methods of the SOMA algorithm have been extended by the ALL TO ONE-NET. This is the ALL TO ONE method with an extended parameter of Network Leader. The method consists of gradually optimising the position of the Leader to be the best found position within the swarm of the same apps. But if a Leader has a better result on the machine after the end of the migration round than the others on the other machines, they will then replace this (local) Leader behind this lead of the network.

Description of communication via FCM

If we have a new incoming device to the swarm devices, as you can see in Fig. 1, the new device sends to the app server address information (Blue arrow). The App server stores this data and sends back to the new incoming device other addresses of the already connected devices (Full orange arrow). With FCM service, the app server sends the new device address to other already connected devices. (Dotted orange arrows) Communication is no longer performed via the app server during calculations. For example, if one device finds a new global leader (second from right), he will pass on information about the new global leader by using FCM service to other devices (Green arrows).

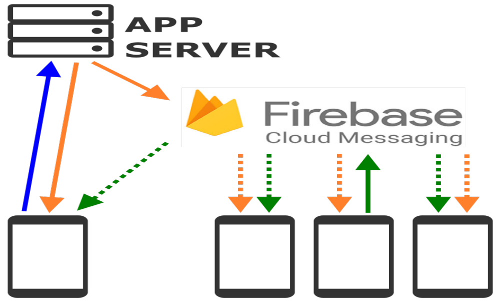

Fig. 1: FCM communication scheme Source: Custom Processing, Logo Firebase Cloud Messaging is from "https://firebase.google.com/docs/cloud-messaging/"

The SETUP button brings you to the setup of the given problem and then setting the solution parameters (Fig. 3).

Described Android app AI SOMA can be installed from Google Play server here:

https://play.google.com/store/apps/details?id =cz.bukacek.ai_soma

Another option is to install with ShopPlay directly from an Android device and find the app using the AI SOMA name.

Figure 2 shows the application turned on. On the right, there is an application menu where you can turn on the launch of one migration round via the button "Make round" or all migration rounds via the button "Start algorithm".

The problem is given by solving the extremes of the functions as Fig. 4. We select the type of soma algorithm method and its parameters and close the Setup window.

Pseudo code of method “All To ONE-Net Adaptive” of “SOMA” algorithm.

```
All To One Net Adaptive
begin
        - Generate an initial local population.
        - Finding Local_Leader = Best from populacion ();
        - Setting Global_Leader = null;
        while num_Migration < Migration do
         begin
        - Algorithm for members migration to Local_Leader.
        - Finding Local_Leader = Best from populacion ();
                    if( Global_Leader == null or  Local_Leader > Global_Leader) then
                    begin
                        Global_Leader = Local_Leader;
                        Sending global leader via FCM ();
                    end;
        end;
end;
Parallel process
Receiver FCM (Incoming_Leader)
Begin
           if(Global_Leader != null and Incoming_Leader > Global_Leader ) then
           begin
                    Local_Leader = Incoming_Leader;
                    Global_Leader = Incoming_Leader;
           end;
end;
```

Turn on the algorithm via click "Start algorithm" and wait (Fig. 5) for its completion until the end of the last migration round.

Each time the migration round is completed, a new leader is chosen if it has a functional value that is more extreme than the network leader, thus replacing this network leader on all devices of the swarm.

On the device with the incoming network leaders location information is then evaluated as to whether its functional value is more extreme than its own lead value and if yes, this leader is replaced by that network leader.

The replacement occurs smoothly even while migration is running.

2.2. Used hardware

Selected Android devices, where the app is executable, were used to experiment, as you can see in Fig. 6. The devices were selected to cover different a wide spectrum of devices from different manufacturers as well as incorporating different versions of Android operating systems. These devices which were used are:

1) Google Nexus 7 (Android 5)
2) Lenovo Tab 4 - 10“ (Android 7.1)
3) Google Nexus 5X (Android 8.1)
4) Sony Sony Xperia X F5321 (Android 7.1)

Tab. 1: App settings on all devices.

Solution	**First De Jong, Schwefel**
Method	All To One
PathLength	3.0
Step	0.11
PRT	0.1
PopSize	20
Migration	200
Dimensions	50

Fig. 2: App AI SOMA Source: Custom processing – app screenshot

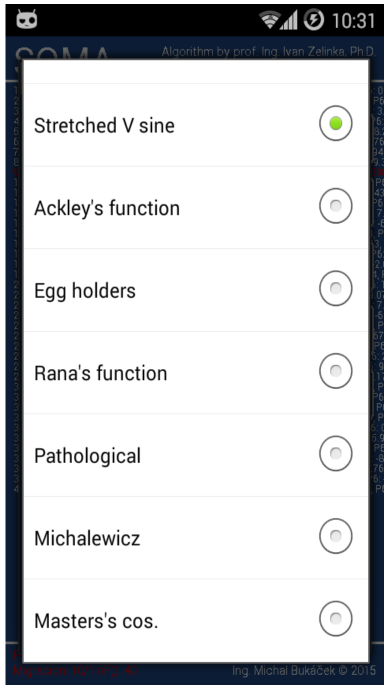

Fig. 4: Select problem function Source: Custom processing – app screenshot

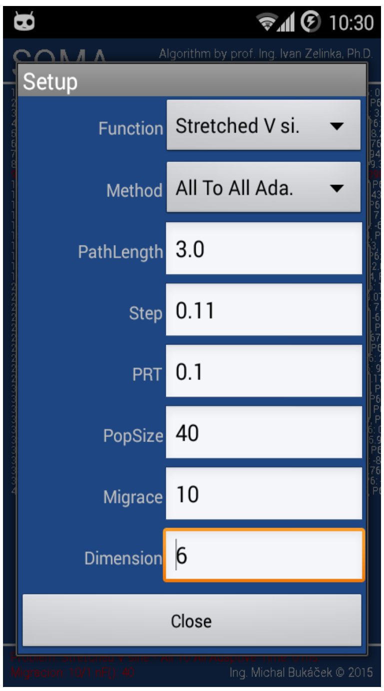

Fig. 3: App setup Source: Custom processing – app screenshot

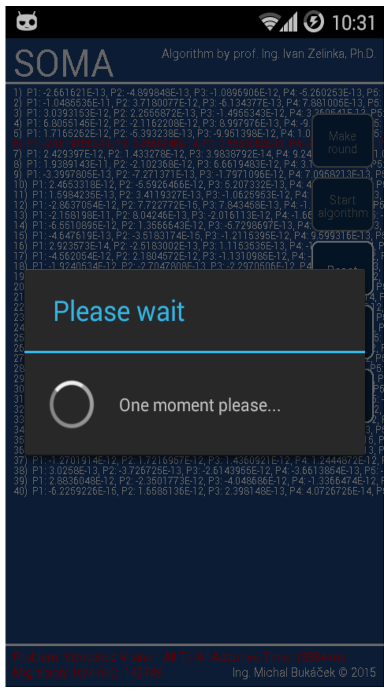

Fig. 5: Running migration round Source: Custom processing – app screenshot

5) Vodafone Smart Prime 6 (Android 5.0)

6) Vodafone Smart Prime 6 (Android 5.0)

7) Xiaomi Redmi Note LTE (Android 4.4)

Devices were first compared between themselves and then for group solving the same task. As a common comparison function, First De Jong and Schwefel were selected in 50 dimensions. In

Fig. 6: Used devices, from the bottom, left - Google Nexus 7, Lenovo Tab 4, Google Nexus 5x, Sony Sony Xperia X, 2x Vodafone Smart Prime 6, Xiaomi Redmi Note Source: Custom Processing - Photograph of used devices

the application, time was measured to determine how long the algorithm is running and auto-stop when the algorithm finds the result, so that it finds at least a certain number of the same agents that have the same function as the leader, that is to say, the same with a certain precision for the given decimal places. So, in this case, the following settings on all devices are set, as shown in Table 1.

Auto Shut Off was turned on. With some 5 same agents as a leader, leaders who are the same leader in 6 decimal places, then at this point, it is considered that the result is found. On all devices, was run the Soma algorithm with the All To One method and then on all the improved All To One Net Adaptive, all of them were set to the same values, as depicted in Fig. 7.

Source: Custom Processing

Fig. 7: a common solution with method All To One Net Adaptive Source: Custom Processing - Photograph of used devices

3. CONCLUSIONS

1) Each device is different, its computing power is also used by other background services, therefore its speed with which to find the result is also different.

2) From this experiment and the resultant measurement obtained it can be stated, that some devices are quicker computing the result if it is connected to the charger as opposed to being without it, the speed is also greatly affected by whether the device is hot, either due to charging or due to any previous calculations, such a warm the device which performs the calculation was therefore much longer than when it is at normal ambient temperature.

3) As for the the equipment being set in the same conditions, the measurement itself and the result will always be found with a similar number of migratory wheels, but with different times.

4) From the results of the group obtained by the measurements, we can see that if there is a small group of 2-5 devices, the results are at best the same or worse than the results of the fastest individual device tested. The measured values of the extended group of seven devices already shows better results than the fastest individual device, but what is interesting is that the results were found

Tab. 2: Measurement results for 4 of the used devices to search for the extremes of the functions of First DeJong and Schwefel.

		Sony Model F 5321		**Vodafone Smart prime 6**		**Vodafone Smart prime 6**		**Redmi Note 4**	
Method		**ms**	**migr ation**	**ms**	**migr ation**	**ms**	**migr ation**	**ms**	**migr ation**
First De Jong	**1**	35268	132	17297	127	16687	126	53147	129
	2	32797	123	16929	126	16761	127	52241	130
	3	29677	117	18638	136	17006	129	51573	129
	4	23569	125	18850	137	17598	131	57388	138
	5	25017	85	18297	136	17598	131	52843	132
	6	34247	129	18135	131	17519	130	47108	117
	7	33517	126	17789	131	18455	134	52600	131
	8	30761	126	17297	125	17702	134	56616	139
	9	32272	128	18117	134	17377	131	51210	129
	10	28902	132	16860	125	18198	132	52612	127
diameter		30602,7	122,3	17820,9	130,8	17490,1	130,5	52733,8	130,1

		Sony Model F 5321		**Vodafone Smart prime 6**		**Vodafone Smart prime 6**		**Redmi Note 4**	
Method		**ms**	**migr ation**	**ms**	**migr ation**	**ms**	**migr ation**	**ms**	**migr ation**
Schwefel	**1**	87820	201	29047	214	27000	205	77398	233
	2	93112	213	30186	229	26913	202	69133	211
	3	95356	221	29531	220	27456	210	65135	214
	4	100544	230	28567	217	29628	220	67131	210
	5	90354	205	30337	226	28904	217	72619	223
	6	94281	216	26992	201	29753	224	69405	208
	7	94487	215	29129	221	27925	213	69621	216
	8	95239	217	27322	212	27933	209	70304	220
	9	93549	214	28075	215	28324	213	64170	204
	10	96536	221	27548	209	27320	204	71857	219
diameter		94127,8	215,3	28673,4	216,4	28115,6	211,7	69677,3	215,8

to be on average faster with a larger number of migratory rounds. In my view, the network leader is completely rebalancing the already existing cluster of agents which had migrated gradually to their local leader and the new leader position then completely alters the existing agents' routes and their concentration around the local maximum. This leads to a greater number of migratory rounds.

It follows then that if there are a few agents in one local flock (on one device) and that then the search for extremes is a function that has a lot of local extremes, it may happen that the agents often become stiff in local extremes. That's another reason why the number of devices was expanded to seven devices. When working through network sharing and a group leader, it can also happen, but with a greater number of devices in the group, it is then less likely to

Source: Custom Processing

Tab. 3: Measurement results for 3 used devices to search for extremes of functions of First DeJong and Schwefel.

		Lenovo Tab 4		Nexus 7 tab		Nexus 5x	
Method		**ms**	**migration**	**ms**	**migration**	**ms**	**migration**
First De Jong	**1**	22621	134	19080	136	13787	138
	2	22564	133	19007	134	13017	129
	3	22088	131	17107	122	13136	132
	4	22088	131	18205	130	13968	125
	5	21051	125	19118	136	14888	135
	6	22100	131	18642	133	12828	127
	7	22369	132	18734	134	13245	130
	8	21650	130	18523	133	13234	131
	9	21791	129	18769	133	12718	127
	10	22805	135	17769	127	20025	134
diameter		22112,7	131,1	18495,4	131,8	14084,6	130,8

		Lenovo Tab 4		Nexus 7 tab		Nexus 5x	
Method		**ms**	**migration**	**ms**	**migration**	**ms**	**migration**
Schwefel	**1**	57659	215	30980	217	22581	219
	2	56644	211	31245	219	20243	208
	3	59457	221	31047	216	18605	210
	4	57628	216	31047	216	22245	218
	5	57300	212	30481	205	21073	216
	6	54872	205	30160	212	22774	219
	7	56418	209	32245	226	18504	212
	8	57169	213	30212	212	24883	229
	9	55189	205	31171	219	24014	218
	10	59706	223	30383	212	18182	207
diameter		57204,2	213	30897,1	215,4	21310,4	215,6

do so, because other devices would negate this local extreme. This situation is shown in Fig. 8, where the red dot represents the local leader by which other blue agents migrate to on the same device. The green dot represents the global leader from another device of which it will migrate from in the next round.

5) Solution with FCM works, even though we limit the size of the message being sent and the number of network lead parameters. The communication itself is relatively fast, even if Google does not guarantee it and when sending a large number of messages, posting may be delayed, or messages from Google may be removed. The communication itself also cuts off some of the computing power from the device itself. Therefore, during group communication, it is better to communicate directly with a larger group, it is the same load for the device as the small group. Due to the observation of devices as a group, I realised one thing. The calculations accelerated, perhaps I would have liked them to have been even faster, but that was not important. What was really new to me, was something I did not realise, the huge role played by these functions with one global extreme of the first random deployment of the agent's population, because it may happen that a

Source: Custom Processing

Tab. 4: Measurement results for 3 groups of devices to search for extremes of functions of First DeJong and Schwefel.

		Group 2		Group 5		Group 7	
Method		**ms**	**migration**	**ms**	**migration**	**ms**	**migration**
First De Jong	**1**	14653	125	13187	125	13489	134
	2	13440	131	12931	125	13102	131
	3	13524	131	13047	134	13633	133
	4	12390	125	12819	130	12907	127
	5	13471	132	13344	130	13072	130
	6	12734	128	14631	138	12874	128
	7	14669	131	13122	128	13874	138
	8	13983	127	13027	133	13104	131
	9	12983	130	12678	125	13768	108
	10	13079	131	13572	133	13274	115
diameter		13493	129,1	13235,8	130,1	13309,7	127,5

		Group 2		Group 5		Group 7	
Method		**ms**	**migration**	**ms**	**migration**	**ms**	**migration**
Schwefel	**1**	19699	234	22338	255	20417	233
	2	21563	235	20117	227	22622	258
	3	23230	219	19193	225	18694	222
	4	28828	216	19910	232	20105	238
	5	27921	212	19499	231	20659	243
	6	27651	209	17296	212	18146	218
	7	25316	206	23966	252	19725	239
	8	19799	201	19518	220	18480	228
	9	21861	231	20206	232	22161	245
	10	21254	220	21788	238	21696	245
diameter		23712	218,3	20383,1	232,4	20270,5	236,9

Group 2 = Nexus5x + Vodafone smart prime 6
Group 5 = Nexus5x + Vodafone smart prime 6 + Redmi Note 4 + Nexus 7 tab + Lenovo tab 4
Group 7 = Nexus5x + 2x Vodafone smart prime 6 + Redmi Note 4 + Nexus 7 tab + Lenovo tab 4 + Sony model F 5321

randomly deployed population on a slower device in the first migratory wheels finds a better Leader than the previously established network Leader which was built on a faster device. This network Leader is then overwritten, and others in the faster group are even quicker due to this coincidence. It is then possible to say the following: The result of the observation was a clear conclusion, that a slower device can find a faster result, thanks to the help of faster devices to help them with the new network leader sent to it. The slower devices, thanks to the random position of agents, can help their faster colleagues in the first migration rounds, this is because, due to the coincidence, it can happen that for faster devices, the position of agents in the first migration wheels is not so advantageous. A major influence of the algorithm regarding this application is by constantly playing an element of chance; in any case, the number of agents pertain-

Source: Custom Processing

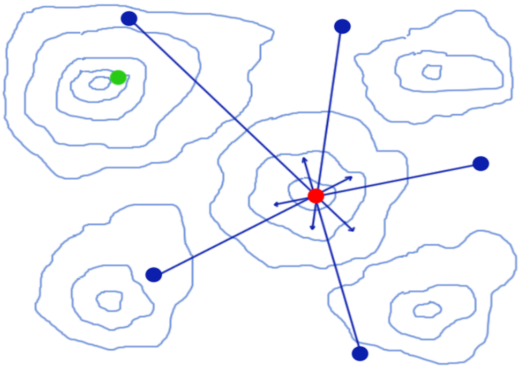

Fig. 8: Global Leader prevents stiffing in local extreme
Source: Custom graphics processing

ing to each device; increases the chance of finding the result faster. On the whole, it is clear that more devices resolve the same problem together more quickly than as one device and even more precisely because the remaining time of the remaining migration rounds can be devoted to the refinement of the result. All applications may not be running at the same time. Even the last-on device may, due to coincidence and due to network leader, help to improve the result or to find a better result.

6) In the real world, I can imagine an application, such as weather monitoring or calendar planning, an application that makes life easier and does not require performance. Such an application would then be able to solve a remote task in the background to search for one extreme in some very large definition space to test to see if a combination of these or other variables is appropriate, etc. These applications mustn't even be turned on at the same time or do not even have to perform many migration rounds.

7) One of the advantages of this swarm algorithm is the fact that, if one or more agents leave the swarm and then only one remains, then the calculation does not end but in fact continues and will reach a result. The application could also be modified so that agents - users could come and go at any time during the calculation.

ACKNOWLEDGEMENT

The following grants are acknowledged for the financial support provided for this research by Grant of SGS No. 2019/137, VSB-Technical University of Ostrava and under the support of NAVY and MERLIN research lab.

References

[1] Zelinka, I., 2002. Artificial Intelligence in problems of global optimization. BEN, Prague.

[2] Zelinka, I., Vasek, V., Lampinen J., 2002. SOMA and Differential Evolution-New Algorithms of Global Optimisation. Fine Mechanics and Optics, 4, pp. 112–117.

[3] Zelinka, I., 2001. Prediction and analysis of behavior of dynamical systems by means of artificial intelligence and synergetic (Doctoral dissertation, Ph. D. thesis).

[4] Schwefel, H.P., 1981. Numerical optimization of computer models. John Wiley & Sons, Inc..

[5] Meier, R. and Lake, I., 2018. Professional Android. John Wiley & Sons.

[6] AL-Madi, N.A., 2014. De Jong's sphere model test for a human community based genetic algorithm model (HCBGA). International Journal of Advanced Computer Science and Applications, 5(1).

[7] Ivan, Z. and Jouni, L., 2000. SOMA-Self-organizing migrating algorithm mendel. In 6th International Conference on Soft Computing, Brno, Czech Republic.

[8] ZELINKA, I., SOMA-Self-Organizing Migrating Algorithm Nostradamus 2000. In 3rd International Conference on Prediction and Nonlinear Dynamic, Zlin, Czech Republic.

[9] Ivan, Z., Vasek, V. and Lampinen, J., New Algorithms of Global Optimization. Automatizace (Journal of Automatization, Czech Ed.), 10(01), pp.628-634.

[10] Zelinka, I., Lampinen, J. and Nolle, L., 2001. On the theoretical proof of convergence for a class of SOMA search algorithms.

[11] FCM – Firebase Cloud Messaging - Technology used to forward data between devices. https://firebase.google.com/docs/cloud-messaging/.

[12] THE RADICATI GROUP, INC. (2016). Mobile Growth Forecast, 2016-2020. https://www.radicati.com/wp/wp-content/uploads/2016/01/Mobile-Growth-Forecast-2016-2020-Executive-Summary.pdf.

About Authors

Michal BUKÁCEK He works as a software developer. He likes to program apps for Android OS, likes to use swarm algorithms. In free time solves a TSP problem.