Heat Load Modelling for District Heating Plants Using an OpenCL-based Algorithm

Erik Král, Petr Čápek, and Lubomír Vašek

***Abstract*—** This research paper explores an OpenCL-based algorithm to aid heat load modelling for district heating plants. Previous studies have proven that heat loads mostly depend on the external temperatures (temperature dependency component) and the time of the day (time dependency component). In this research we have used the sum of two truncated exponential functions to model the time dependency component and a generalised logistic function to model the temperature dependency component. The parameters of these functions are estimated using the traditional particle swarm optimisation (TPSO). The estimation of the parameters can be time consuming so to accelerate the process we have developed an OpenCL-based version of the algorithm. The critical part of the implementation of the algorithm in OpenCL is the use of different types of memories, especially the local memory and also the coalesced or broadcast read Access to Global Memory.

***Keywords*—** District Heating, Heat load, Modelling, Peak function, Particle Swarm Approximation, OpenCL, Graphic Processing Unit.

I. Introduction

THE algorithm presented is designed to be used in decision-support software for the combined heat and power (CH) production in a combined heat and power plant (CHP). The user of this software will have the possibility to test different scenarios according to the weather forecast and his experiences. The algorithm output is the heat load prediction for a given period, of usually days or weeks. The algorithm inputs are measured data from the previous time period and a weather prediction.

Many methods for heat load modelling have already been developed. A greater comparison of them can be found in [1]. Generally, there are three main approaches: black-box (ARIMA, neural networks, etc.)[2][3], when we have no physical knowledge about the problem, simulation models based on physical models, which need complex information about the system and a combination of both: grey box approach, when limited knowledge of a system is known [4][5][6].

Our model is similar to the simple model presented by Dotzauer [7] where the predicted heat load is modelled as a sum of two functions in which the temperature dependent component is approximated using piecewise linear function described by nine parameters. And the time dependent component is approximated using 168 parameters, one parameter for each hour of the week. This model does not require complex knowledge about real CHP system.

We have improved this model by using continuous functions for both components [8]. The temperature dependent component is approximated with a generalised logistic function. And, the time dependent component describes the daily pattern, instead of a weekly pattern, and is approximated using the sum of two hybrids of gaussian and truncated exponential functions (EGH). And there can be more sets of time dependent components parameters depending on the type of the day – weekends, working days etc. We have rapidly reduced the number of model parameters in comparison with Dotzauers model.

After several experiments, we have chosen the Particle Swarm Optimisation (PSO) as the most suitable algorithm for the estimation of the approximation functions parameters. The parameters estimation process using PSO generally takes some minutes on a modern CPU. As we want provide the best user experience for users of this software, we have developed a parallel version of our algorithm that runs on a Graphic Processing Unit (GPU) using OpenCL. By the GPU acceleration, we have greatly reduced the processing time of the algorithm.

All authors are with Faculty of Applied Informatics, Tomas Bata

II. Heat Load Calculation

The district heating networks have dynamic properties like water flow and propagation of heat from the DHP to the consumers and back again. Such networks can be mathematically modelled. Some of these methods are computationally intensive and require full physical modelling of the networks, but there are also simplified methods [9][10]. For the purpose of heat load modelling, a simpler model can be used. The district heating system can be approximated by the load centre of the mass of the system[11]:

$$P(t) = \dot{m}(t)c\left(\vartheta_1\left(t - \frac{T_{D1}}{2}\right) - \vartheta_0\left(t + \frac{T_{D0}}{2}\right)\right) \qquad (1)$$

where

$P(t)$	represents the heat load,
$\dot{m}(t)$	represents the measured mass flow,
c	represents the specific heat capacity,
ϑ_1	represents supply temperature,
ϑ_0	represents return temperature,
T_{D1}	represents the supply line transport time,

T_{D0}	represents the transport time of return line,
t	Represents time.

We can simplify the model, so the transport time is only dependent on the mass flow and the total mass volume of district heating network. Transport times can be calculated according to [11]:

$$R = \int_{t-T_{D1}}^{t} \dot{m}(\tau)\, d\tau \tag{2}$$

$$R = \int_{t}^{t+T_{D0}} \dot{m}(\tau)\, d\tau \tag{3}$$

where

R	represents the known mass volume,
T_{D1}	Represents the unknown transport time of the supply line,
T_{D0}	represents the unknown transport time of the return line.

III. Heat Load Approximation

The heat load is approximated by the sum of time dependent and temperature dependent components:

$$f_P(t, \vartheta_{ex}) = f_{time}(t) + f_{temp}(\vartheta_{ex}) \tag{4}$$

where

$f_{time}(t)$	represents the time dependent, component,
ϑ_{ex}	represents the outdoor temperature,
$f_{temp}(\vartheta_{ex})$	represents the outdoor temperature, dependent component.

A. Temperature Dependent Component

The temperature dependent component is at certain periods inversely proportional to the external temperature and we have chosen the generalised logistic function as the most appropriate for this approximation:

$$f_{temp}(\vartheta_{ex}) = A + \frac{K - A}{(1 + Qe^{-B(\vartheta_{ex}-M)})^{\frac{1}{v}}} \tag{5}$$

where

A	represents the lower asymptote,
K	represents the upper asymptote,
Q	represents the dependence on the value $f_{temp}(0)$,
B	represents affects near which asymptote maximum growth occurs,,
v	affects near which asymptote maximum growth occurs,
M	represents the time of maximum growth if Q = v.

B. Time Dependent Component

The daily heat load pattern is typified by its morning and evening peaks. Thus, the time dependent component is approximated by the sum of the two peak functions. The Hybrid of Gaussian and truncated exponential function (EGH) was selected as most the convenient function due to its capability to incorporate asymmetric peaks and its fast convergence [12]. Hybrid of Gaussian and truncated exponential function is defined as

$$d = 2\sigma^2 + \tau(t - t_m)$$

$$f_{EGH}(t) = \begin{cases} H\, exp\left(\dfrac{-(t-t_m)^2}{d}\right), & d > 0 \\ 0, & d \leq 0 \end{cases} \tag{6}$$

where

H	represents the peak height,,
σ	represents the standard deviation of the parent Gaussian peak,
τ	represents the time constant of the precursor exponential decay,
k_L	represents the parameter of the speed of the fall of the leading trail,
t_m	represents the time of the peak.

And the $f_{time}(t)$ function is then the sum of two EGH functions, where $f_{EGH1}((t - t_0) \bmod 24) +$ describes the morning peak and $) + f_{EGH2}((t - t_0) \bmod 24)$ describes the evening peak of heat load demand. Because the function is periodical and describes the 24hour daily pattern, we have to shift time using time by using a time offset and function modulo to match the daily minimum around 1 am.

$$f_{time}(t) = f_{EGH1}((t - t_0) \bmod 24) + f_{EGH2}((t - t_0) \bmod 24) \tag{7}$$

where

t_0	represents the time offset.

IV. Parameter Estimation

A. Fitness Function

The fitness function using approximation functions is defined as

$$\min_{\beta} \sum_t (P(t) - f_P(t, \vartheta_{ex}, \beta))^2$$
$$0 < t_m^1 < t_m^1 < 24$$
$$0 < H^1 \tag{8}$$
$$0 < H^2$$

where

β	represents the vector of $f_P(t, \vartheta_{ex})$ parameters.

As depicted in Table 1, the function has 17 parameters. The time offset t_0 and mass volume R are known and set as constant, so there are a remaining 15 parameters to be estimated.

Table 1: Vector of $f_P(t, \vartheta_{ex})$ parameters

Function	Parameters	Number of parameters
$f_{temp}(\vartheta_{ex})$	A, K, Q, B, v, M	6
$f_{EGH1}(t)$	$H^1, \sigma^1, \tau^1, k_L^1, t_m^1$	5
$f_{EGH1}(t)$	$H^2, \sigma^2, \tau^2, k_L^2, t_m^2$	5
$f_{time}(t)$	t_0	1
$f_P(t, \vartheta_{ex})$	$A, K, Q, B, v, M,$ $H^1, \sigma^1, \tau^1, k_L^1, t_m^1,$ $H^2, \sigma^2, \tau^2, k_L^2, t_m^2,$ t_0	17

B. Particle Swarm Algorithm

The Particle swarm algorithm (PSO) [13] was chosen as the numeric optimisation algorithm suitable for problems without the explicit knowledge of the gradient of the function to be optimised. In this paper we describe a parallel version of the traditional PSO (TPSO), because its parallelization is more general than the standard PSO (SPSO) [14]. Therefore, the principles of this solution can be used for the implementation of different evolution algorithms, where the best fitness function value must be shared within the population. The parallel implementation of SPSO could be simpler because the best global fitness value does not have to be found. TPSO should be written in this form:

$$\mathrm{V}_{id}(k+1) = \omega \mathrm{V}_{id}(k) + c_1 r_1 \big(PB_{id}(k) - \mathrm{X}_{id}(k)\big) + c_2 r_2 \big(GB_d(k) - \mathrm{X}_{id}(k)\big) \quad (9)$$

$$\mathrm{X}_{id}(k+1) = \mathrm{X}_{id}(k) + \mathrm{V}_{id}(k) \quad (10)$$

where

i	represents the particle index $\mathrm{i} = 1,2, \ldots \mathrm{NP}$,
NP	represents number of particles in swarm,
d	represents the dimension index $\mathrm{d} = 1,2, \ldots \mathrm{D}$,
D	represents the dimension of the solution space,
k	represent index of iteration,
$\mathrm{X}_{id}(k)$	represents the particle position,
$\mathrm{V}_{id}(k)$	represents particle velocity
$PB_{id}(k)$	represents the particle best position,
$GP_d(k)$	represents the swarm best position,
ω	represents the inertia component,
c_1	represents the social component,
c_2	represents the cognitive component,
r_1, r_2	are uniformly distributed random numbers in interval $[0, 1]$.,

The particle velocity is limited to $\mathrm{V}_{id}(k) \in [-V_{max}, V_{max}]$, where V_{max} is the maximum particle velocity. The number of particles NP is usually set at two times more than the dimension D. The inertia component ω is set at about 0.8, the social component c_1 is set at about 1.4 and the cognitive component c_2 is set at about 0.6. We use MaxDistQuick as a stopping criterion as described in [14]. The optimization is stopped if the maximum distance of the majority of the particles is below a threshold *eps* or the maximum number of iteration is reached.

There are already many PSO implementations for GPU [16][17]. Our parallel implementation of PSO is similar to the CUDA-based solution in [16]

C. OpenCL

OpenCL is standard for general purpose programming across CPU, GPU and other processors [18]. It consists of an API for parallel computation and a cross platform programming language which is a subset of ISO C99 with an extension for parallelism.

An OpenCL model consists of a host connected to one or more OpenCL devices like CPU or GPU. An OpenCL device is divided into one or more compute units, which are divided into one or more processing elements (PE). The function executed on an OpenCL device is called kernel. When a kernel is submitted for execution, an index space is defined and is called NDRange. The NDRange is an N-dimensional index space, where N is one, two or three. An instance of the kernel is executed for each point in this index space and is called a work-item. Each work-item has a unique global ID. The work items are organized into work-groups which have their own unique work-group ID. Work-items are also assigned a unique local ID within a work-group. We use a one dimensional index, so in this case, the global ID can be defined as:

$$\mathrm{g} = \mathrm{w\,S} + \mathrm{sx} \quad (11)$$

where

g	represents the work-item global ID,
w	represents the work-group ID,
S	represents the work-group size
s	represents the work-item local ID

The number of work-groups that can be computed is defined as:

$$\mathrm{W} = \mathrm{G}/S \quad (12)$$

where

W	represents number of work-groups,
G	NDRange size (number of work-items)

A memory model in OpenCL is critical for efficient algorithm implementation. There are four types of memory regions:

- global memory – permits read/write access in all work-groups..
- constant memory – a region of global memory that remains constant during the execution of a kernel,
- local memory – a memory local to work-group,
- Private memory – a memory private to a work item.

On GPU, the private memory is considered to be the fastest memory together with the local memory; the slowest memory is the global memory. The memory scheme of our algorithm is depicted in Figure 2.

V. Algorithm

The algorithm runs k-iterations and then updates the User Interface (UI) as depicted in Figure 1, so the UI does not freeze and the operator can observe and control the intermediate results.

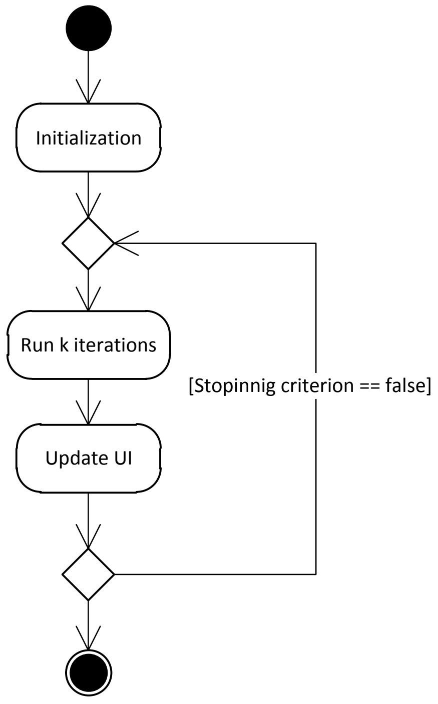


Figure 1: Activity diagram – Algorithm and user interface updates

In presented, one work item represents one particle and each work item is processed in one Processing Element, as depicted in Figure 2. Each work-item initializes the particle position X_i and the velocity V_i.using an OpenCL implementation of a pseudo random number generator (PRNG) [19]. The private memory is the fastest memory, but it has limited capacity. As there are only 17 parameters in vector β, so the dimension of solution space D is only 17, the particle position X_i, and velocity V_i can be stored in the private memory as the one-dimensional arrays. Using the private memory for X_i, V_i we can greatly improve the speed of the algorithm. The indexes of the particles best positions and the values of the fitness functions of the particles in each work group are stored in the one dimensional array VL and FL respectively. Arrays VL and FL are stored in the fast local memory and are shared within one work-group. The work group size is set equal to the number of Compute Units, so the maximum use of the local memory is achieved. Parallel minimum reduction is used for finding minimum PB_{min} from all particles in the local group using arrays VL and FL (Figure 5). The indexes and value of the fitness functions of the best particles from each work groups are stored in arrays VG and FG respectively, which size is equal to the number of work groups. Then, the parallel minimum reduction is again used for finding the index of the best particle in population i_g and the related value of fitness function F_g (Figure 4). Finally, the particle positions and velocities are updated according to equations (9) and (10) and the algorithm continues until the stopping criterion is valid or the maximum number of iteration is reached.

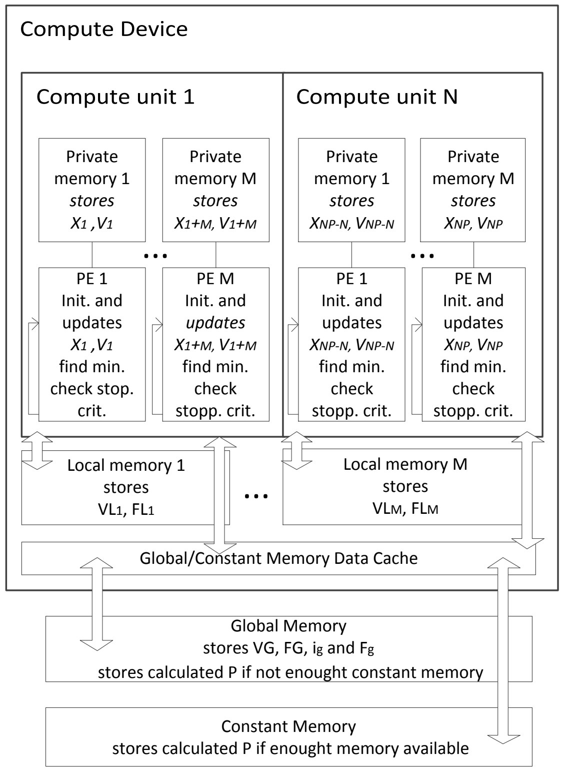

Compute Device
Compute unit 1
Compute unit N
Private memory 1 stores X_1, V_1
Private memory M stores X_{1+M}, V_{1+M}
Private memory 1 stores X_{NP-N}, V_{NP-N}
Private memory M stores X_{NP}, V_{NP}
PE 1 Init. and updates X_1, V_1 find min. check stop. crit.
PE M Init. and updates X_{1+M}, V_{1+M} find min. check stopp. crit.
PE 1 Init. and updates X_{NP-N}, V_{NP-N} find min. check stopp. crit.
PE M Init. and updates X_{NP}, V_{NP} find min. check stopp. crit.
Local memory 1 stores VL_1, FL_1
Local memory M stores VL_M, FL_M
Global/Constant Memory Data Cache
Global Memory stores VG, FG, i_g and F_g stores calculated P if not enought constant memory
Constant Memory stores calculated P if enought memory available

Figure 2: Parallel TPSO memory scheme

The population of the particles is split into N groups (Figure 3 and Figure 4) according to the number of working groups $W$and the number of particles in swarm NP. NDRange size (number of work-items) G is set equally to the NP.

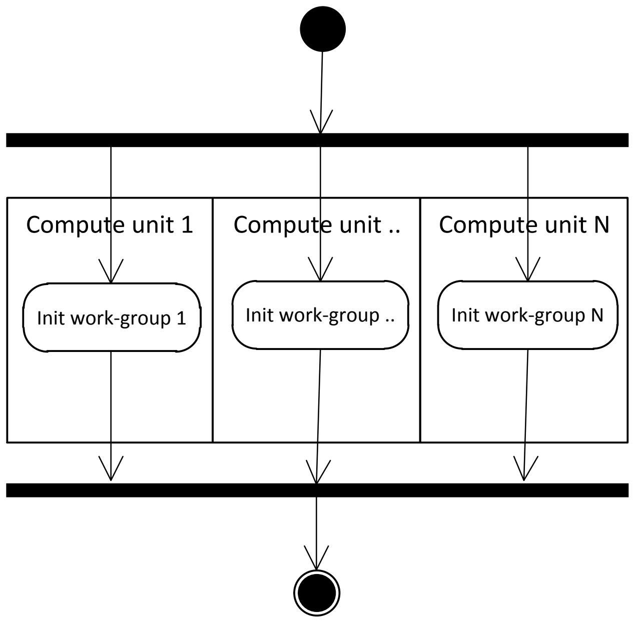


Figure 3: Activity diagram - Initialization

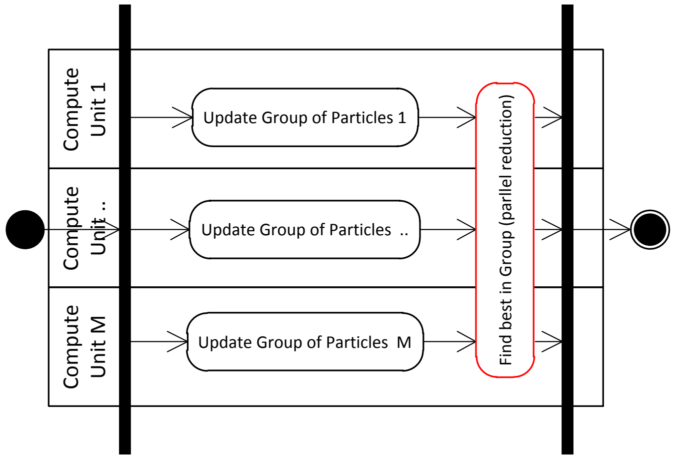

Compute Unit 1
Compute Unit ..
Compute Unit M
Update Group of Particles 1
Update Group of Particles ..
Update Group of Particles M
Find best in Group (parllel reduction)

Figure 4: Activity diagram - Update group of particles

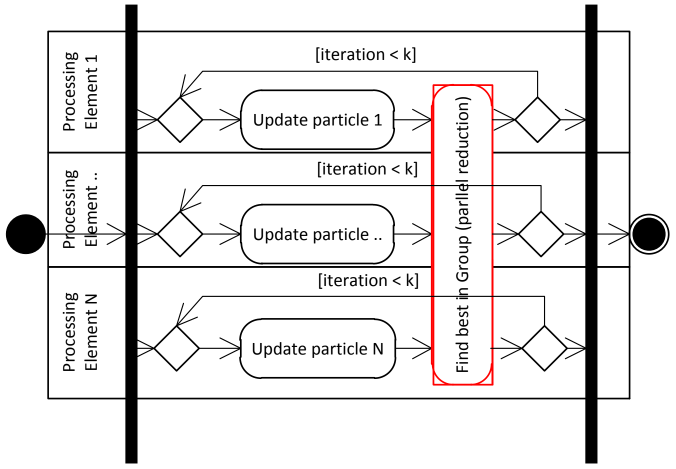


Figure 5: Activity diagram - Particle update

VI. Experiments

We have evaluated the presented algorithm on the data measured during the two heating seasons I and II at the two CHP plants A and B. Firstly, we ran the approximation for every day in the heating seasons, then only for the working days and lastly for the nonworking days (weekends and public holidays) only. In Table 2 and Table 3 respectively the Root Mean Square Errors is presented (RMSE) for the given season, day class and the CPH plant.

We have run the experiments on the Intel Core i5 i5-520M / 2.4 GHz with NVIDIA GeForce 310M GPU. GForce 310M provides 16 Cuda Cores. The OpenCL implementation has only float precision while the standard CPU implementation has double precision. The OpenCL is more than 10 times faster than the CPU solution.

The Previous researches [16][17] prove that the acceleration rate increases with the number of processing elements (for example Cuda Cores or Steam Processors). Therefore, it is expected that the algorithm running time can be reduced by using the GPU with high number of the Processing Elements.

Table 2: CHP plant A results from 1st October – 1st May

Season	Day class	RMSE [kW]
I	All	5919,9
I	Working	5547,0
I	Nonworking	5885,4
II	All	6215,3
II	Working	5880,7
II	Nonworking	6047,5

Table 3: CHP plant B results from 1st October – 1st May

Season	Day class	RMSE [kW]
I	All	3424,1
I	Working	3451,1
I	Nonworking	3304,7
II	All	2842,4
II	Working	2747,7
II	Nonworking	2909,9

A. *Temperature dependent component*

As you can see in the Figures 6 – 9, the approximated temperature dependent component parameters are similar for both working and nonworking days. That means that it is not dependent on the class of the day. If we compare the heating seasons (Figure 6 andFigure 7 and Figure 8 andFigure 9, respectively), the temperature dependent component changes over a long time period. It will be more preferable to update the temperature dependent component parameters continuously with the new measurements while dropping the oldest measurements.

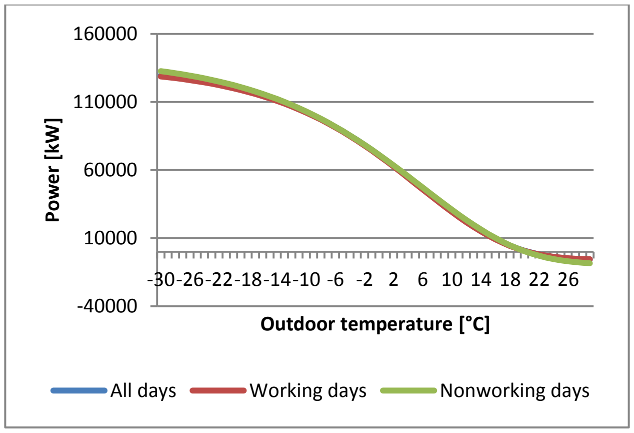


Figure 6: Temperature dependent component, CHP Plant A, season I

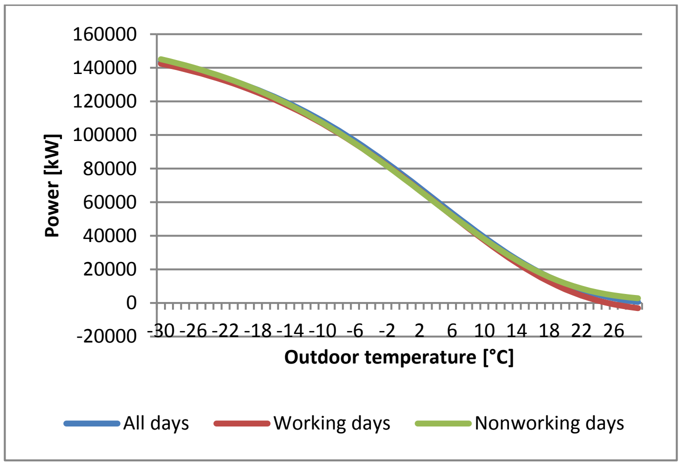


Figure 7: Temperature dependent component, CHP Plant A, season II

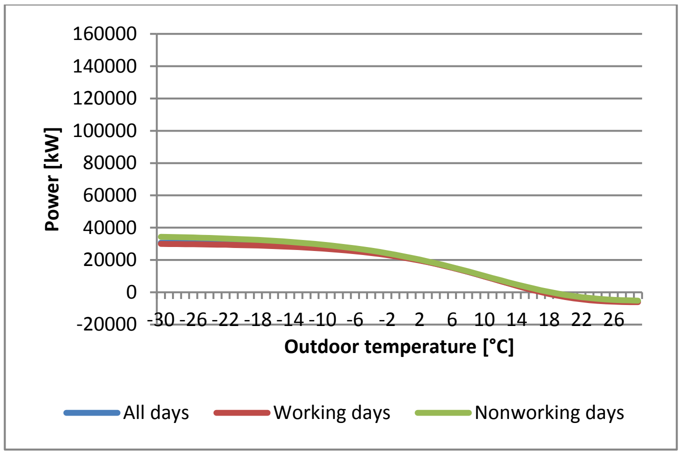


Figure 8: Temperature dependent component, CHP Plant B, season I

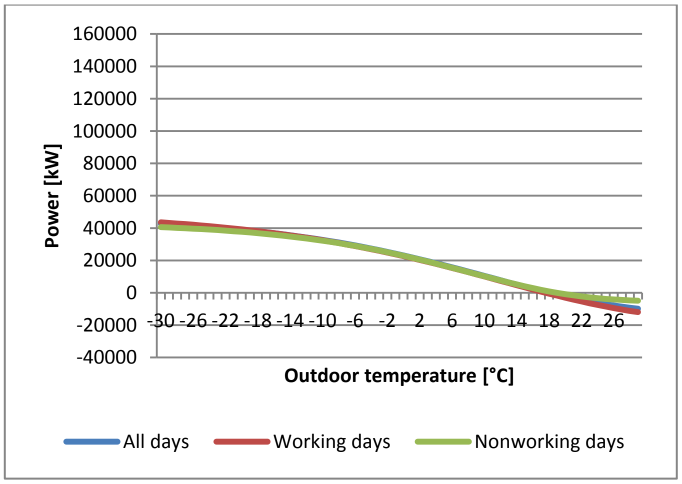


Figure 9: Temperature dependent component, CHP Plant B, season II

B. *Time dependent component*

As you can see in the Figures 10 – 13, the approximated time dependent component is different for each day class. There is noticeably higher peak in the mornings on working days while the evening peak is similar for all day classes. If we compare the seasons in the same CHP plant, the time dependent component parameters also change over a long time period.

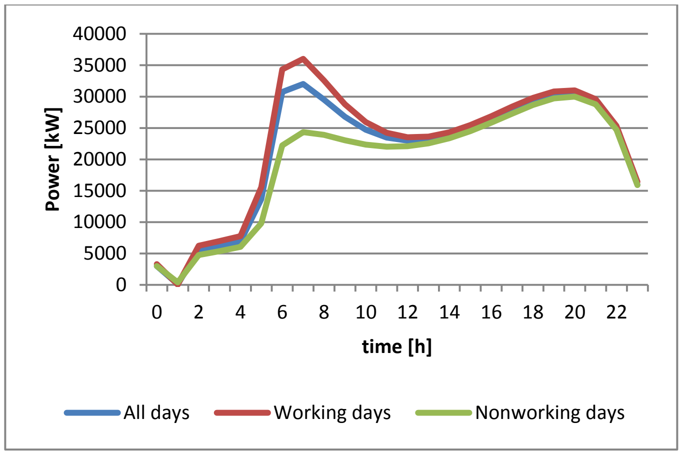


Figure 10: Temperature dependent comp., CHP Plant A, season I

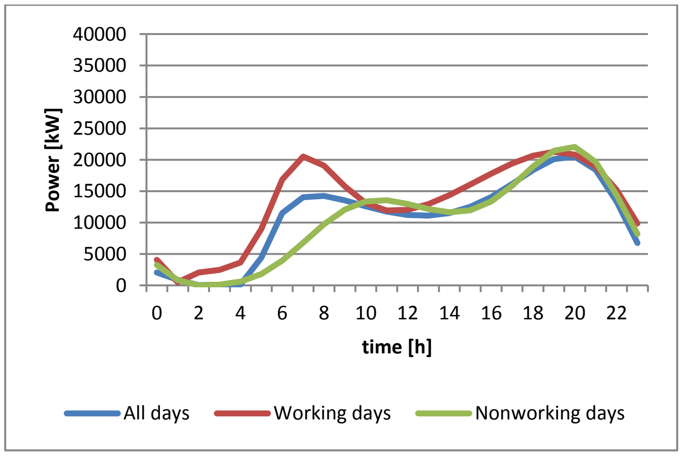


Figure 11: Temperature dependent comp., CHP Plant A, season II

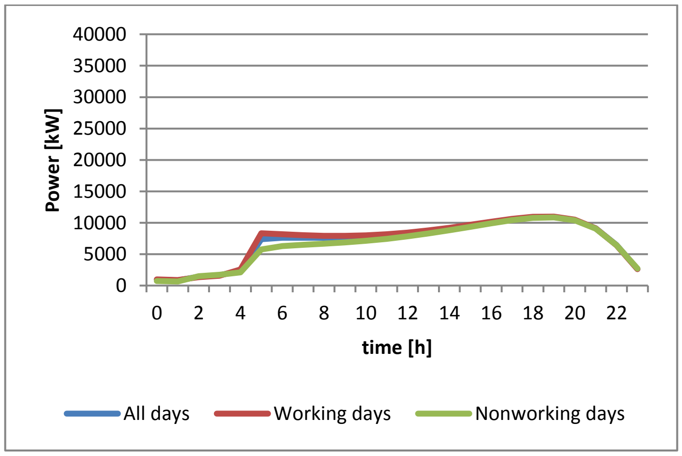


Figure 12: Temperature dependent comp., CHP Plant B, season I

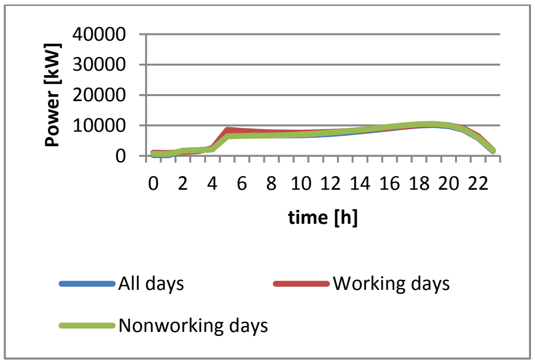
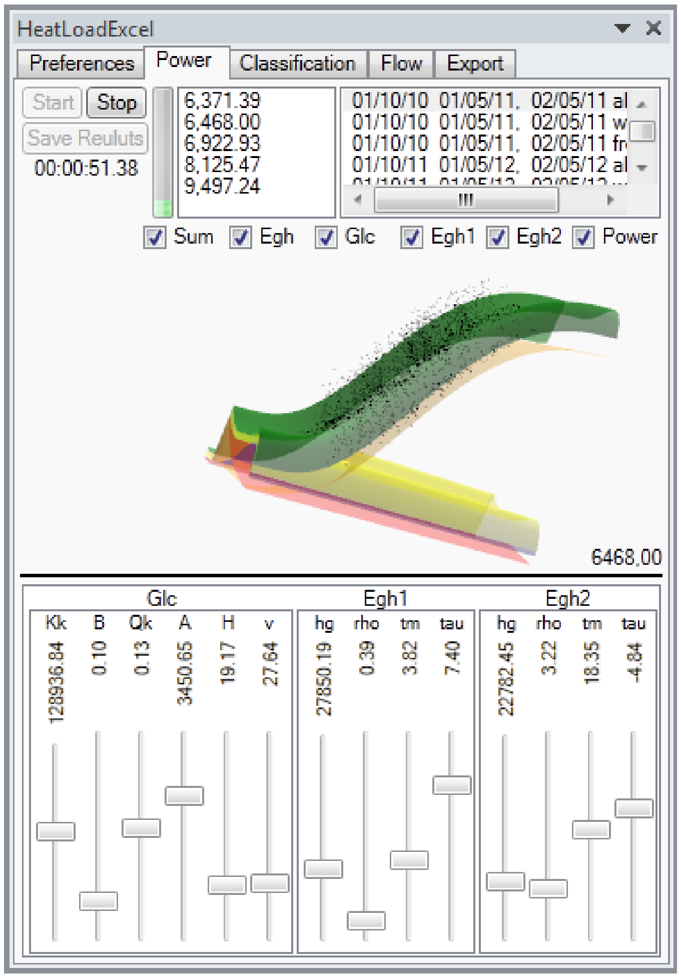


Figure 13: Temperature dependent compoponent, CHP Plant B, season II

VII. Implementation

The algorithm was implemented as a Microsoft Excel 2007 and 2010 Addin application. Figure 14 depicts the Panel that is used for estimating and evaluating the approximation parameters.



00:00:51.38

6,371.39
6,468.00
6,922.93
8,125.47
9,497.24

01/10/10 01/05/11, 02/05/11 al
01/10/10 01/05/11, 02/05/11 w
01/10/10 01/05/11, 02/05/11 fr
01/10/11 01/05/12, 02/05/12 al

✓ Sum ✓ Egh ✓ Glc ✓ Egh1 ✓ Egh2 ✓ Power

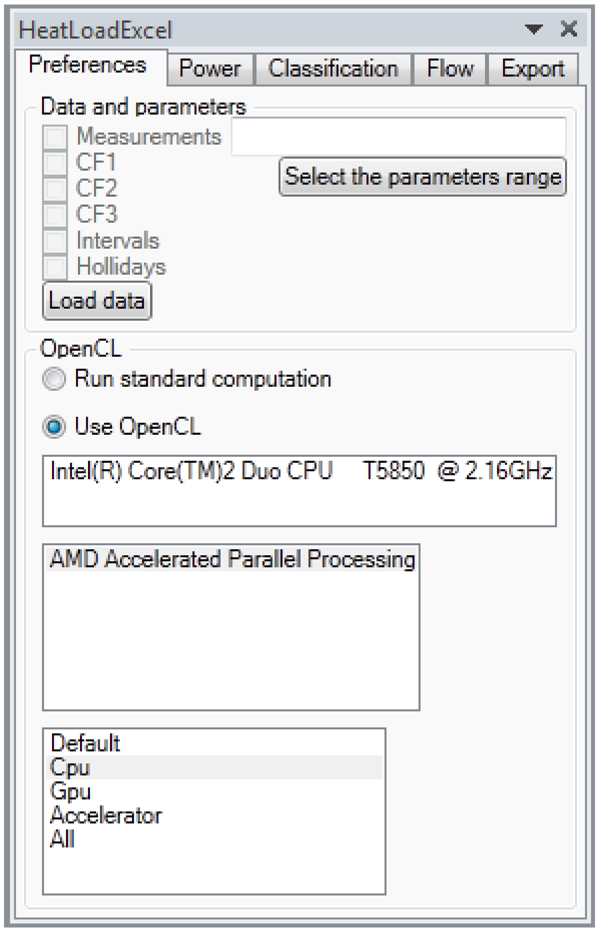
6468,00

Glc						Egh1				Egh2			
Kk	B	Qk	A	H	v	hg	rho	tm	tau	hg	rho	tm	tau
128936.84	0.10	0.13	3450.65	19.17	27.64	27850.19	0.39	3.82	7.40	22782.45	3.22	18.35	-4.84

Figure 14: Excel Addin Panel

The application was designed to be compatibility with the Model View ViewModel (MVVM) [20] design pattern to provide further extensibility and maintainability.

Figure 15 depicts the preferences panel. The user can choose the CPU or GPU (OpenCL) version of the algorithm and can also load measurements and others necessary data.



Run standard computation [✓ Use OpenCL]

Intel(R) Core(TM)2 Duo CPU T5850 @ 2.16GHz

AMD Accelerated Parallel Processing

Default
[✓ Cpu]
Gpu
Accelerator
All

Figure 15: Preferences panel

VIII. Conclusion

An OpenCL based parallel solution for heat load approximation and prediction in a district heating system that's runs on GPU was presented in this paper. This model was successfully implemented using PSO as a parallel algorithm.

The running speed of the algorithm is more than 10 times faster than a CPU solution. This model does not require complex knowledge about real CHP system and we have also reduced the number of model parameters in comparison with similar model. The preliminary experiments proved that the running speed of the algorithm was greatly reduced. We will now concentrate on the implementation of different evolution algorithms.

Acknowledgment

The work was undertaken with the financial support of the the European Regional Development Fund under the Project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089.

REFERENCES

[1] Heller J., *Heat-load modelling for large systems*, Applied Energy, Vol. 72, No. 1, May 2002, pp. 371-387, ISSN 0306-261.

[2] Vařacha P., Jašek R., *ANN Synthesis for an Agglomeration Heating Recent Researches in Automatic Control. Montreux*, WSEAS Press, 2011, pp. 239-244, ISBN 978-1-61804-004-6.

[3] Chramcov B., *Heat Demand Forecasting for Concrete District Heating System*. International Journal of Mathematical Models and Methods in Applied Sciences, 2010, Vol. 4, No. 4, pp. 231-239.

[4] Nielsen H., Madsen, H., *Modelling the heat consumption in district heating systems using a grey-box approach*, 2006, Vol. 38, pp. 63-71.

[5] Dolinay V., Vašek L., *Simulation of Municipal Heating Network Based on Days with Similar Temperature*, International Journal of Mathematics and Computers in Simulations, 2011, Vol. 5, No. 5, pp. 470-477, ISSN 1998-0159.

[6] Vasek, L., Dolinay, V. *Simulation model of heat distribution and consumption in municipal heating network*, International Journal of Mathematical Models and Methods in Applied Sciences, 2010, Vol. 4, No. 4, pp. 240-248.

[7] Dotzauer, E., *Simple model for prediction of loads in district-heating systems*, Applied Energy, 2002, No. 73, pp. 277-284.

[8] Král E.. Vašek L., Dolinay V.; Čápek P., *The Use of Peak Functions in Heat Load Modeling of Distric Heating System*, International Journal of Mathematical Models and Methods in Applied Science, 2011, Vol. 5, pp. 1241-1248. ISSN 1998-0140.

[9] Larsen H., Pálsson H., Bøhm B., Ravn H., *Aggregated dynamic simulation model of district heating networks*, Energy Conversion and Management, Vol. 43, No. 8, May 2002, pp. 995-1019, ISSN 0196-8904.

[10] Helge L., Benny B., Michael W., A comparison of aggregated models for simulation and operational optimisation of district heating networks, *Energy Conversion and Management*, Vol. 45, No 7-8, May 2004, pp. 1119-1139, ISSN 0196-8904.

[11] Saarinen L.. *Modelling and control of a district heating system*, Uppsala University, 2008, pp. 67 s., ISSN 1650-8300.

[12] Jianwei Li, Comparison of the capability of peak functions in describing real chromatographic peaks, *Journal of Chromatography A*, Volume 952, Issues 1-2, 5 April 2002, Pages 63-70, ISSN 0021-9673, DOI: 10.1016/S0021-9673(02)00090-0.

[13] JKennedy J., Eberhart R., Particle Swarm Optimization IEEE International *Conference on Neural Networks*, Pert, WA, Australia, Nov. 1995, pp.1942-1948.

[14] Bratton D., Kennedy J., Defining a Standard for Particle Swarm Optimization, *IEEE Swarm Intelligence Symposium*, April 2007, pp.120-127.

[15] Zielinski K., Laur R., Stopping criteria for a constrained single-objective particle swarm optimization algorithm, *Informatica*, Vol. 31, No. 1, pp. 51-59, 2007.

[16] You Zhou, Ying Tan, GPU-based Parallel Particle Swarm Optimization, 2009, *IEEE Congress on Evolutionary Computation (CEC 2009)*, pp. 1493-1500.

[17] Cadenas-Montes, M., Vega-Rodriguez, M.A., Rodriguez-Vazquez, J.J., Gomez-Iglesias, A.: Accelerating particle swarm algorithm with GPGPU, 2011, *In: 19th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pp. 560–564. IEEE (2011)

[18] Khronos Group. *The OpenCL Specification 1.2*, 2011

[19] John K. Salmon, Mark A. Moraes, Ron O. Dror, and David E. Shaw, Parallel random numbers: as easy as 1, 2, 3, *In Proceedings of 2011International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, New York, NY, USA, 2011.

[20] Garofalo R., *Building Enterprise Applications with Windows Presentation Foundation and the Model View ViewModel Pattern.* Microsoft Press; 1 edition , March 31, 2011, ISBN 978-0735650923

The first author, Ing. et Ing. Erik Král is a PhD. candidate and assistant leading lectures for the Department of Digital communication and Hardware of communication systems at Tomas Bata University in Zlin. Between 2003 and 2006 he worked as a Software developer (MS Navision DB, CRM system, .NET, c#). Between 2006 and 2011 he worked as a researcher on National Research Program II, The intelligent system controlling an energetic framework of an urban agglomeration (successfully finished in 2011). He teaches Object-oriented Programming, FPGA design and Programming.