Modelling of multicolour light source using distributed computing network. 2. Parallel implementation and experimental results

Aistė Ambrazevičiūtė, Rimantas Vaicekauskas [a]
Department of Informatics, Faculty of Mathematics and Informatics
Vilnius University, Naugarduko 24, Vilnius, Lithuania

Received 29 January 2011, accepted 20 February 2011

Abstract. Light-emitting diodes (LEDs) are widely applied in conventional lighting. We investigate white light sources made of three primary coloured LEDs with respect to colour rendition ability. The spectral power distribution of individual LED is simulated using a Gaussian function with selectable peak wavelength. The optimization problem of finding a trichromatic source having maximum colour rendering index was defined and solved using parallel branch and bound method. We have implemented and examined several centralized data exchange schemes on a parallel computing cluster. Speed and efficiency of the proposed algorithms were revealed.

Citations: Aistė Ambrazevičiūtė, Rimantas Vaicekauskas. Modelling of multicolour light source using distributed computing network. 2. Parallel implementation and experimental results – *Innovative Infotechnologies for Science, Business and Education,* ISSN 2029-1035 – 1(10)(2011) 21-29.

Keywords: LED lighting; Colour rendering; Global optimization; Distributed computing; MPI.
PACS: 07.05.Tp; 42.66.Ne; 02.60.Pn.
Short title: Modelling of light source - 2.

Introduction

Simulating physical phenomenon requires an accurate physical model and computational power. Our previous publication [1] was aimed to develop a mathematical model which describes the spectral distribution of the light source consisting of several coloured *Light-emitting diodes* (LEDs). This problem was solved relating physical parameters of light to the psychophysical colour quantities such as CRI. General approaches - how to quickly find optimal solution - were investigated in respect of applying distributed computing model - *Message Passing Interface* (MPI) technique, which requires effective management of distributed computing resources.

A problem of optimization investigated in this work concentrates on maximizing the CRI of a modelled light source of 3 LEDs with adjustable peak wavelenght. Though tabular functions are used for the analysis of the light source, analytical methods based on derivatives of the objective function may not be applied to solve this optimization problem. Trial and error method is very inefficient, so it cannot be used either. A similar problem presented in Ref. [2] was solved by using the stochastic *hill climbing* technique, although this method does not guarantee that local and global maximum values coincide.

[a]Corresponding author, email: *Rimantas.Vaicekauskas@mif.vu.lt*

1. Spectral distribution of light source

To have a model of compound light emmiter consisting of several LEDs the primary emitters are described at first.

1.1. Gaussian shaped LED emission

Radiation of light source can be described using *spectral power distribution* (SPD). The spectra of primary monochrome LEDs were approximated by Gauss-shaped distribution $f(x, \mu, \sigma)$ (see Fig. 1)

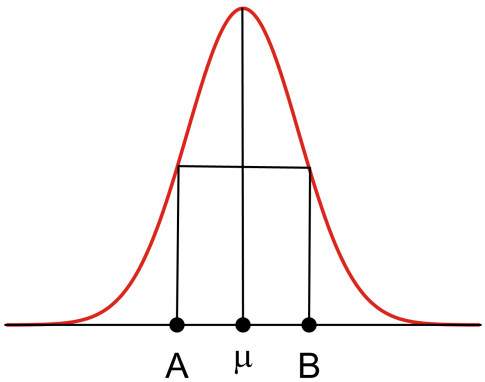

Fig. 1. Spectral power distribution of LED radiation as intensity function on wavelength. Half-width - σ=30 nm.

SPD $f(x,\mu,\sigma)$ expresses an behaviour where μ and σ represent the wevelength of peak and half-width, respectivelly, and x is functional parameter - wavelength. Due to physical circumstances - emmision stability of LEDs, we simplify the function $f(x,\mu,\sigma)$ by choosing a constant value of σ. This distribution $f(x,\mu)$ will depend only on the parameter of peak position μ.

$$f(x,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \quad (1)$$

$$f(x,\mu) = 15 \cdot \sqrt{\frac{\log(e)}{\log(4)}} \cdot \exp\left[\log 8.0 \cdot \left(\frac{x-\mu}{30}\right)^2\right] \quad (2)$$

We can then use this function $f(x,\mu)$ as a template to generate various SPDs. In the computation experiments following distribution function was employed.

If the height of peak in gauss distribution is h, we get the width of the distribution at half magnitude $\cdot h$ as a half-width - μ=30 nm. That is an average value for high-brightness LEDs made by using common (AlInGaP, InGaN) technologies and operating at typical modes. In our case, the peak position value λ_0 belongs to the visible wavelength range [360÷830] expressed in nm.

1.2. Mixing of spectral components

By mixing two different light sources with different spectra, a third light source is made and it is called a resulting light source. The resulting spectrum depends on the ratio the initial sources were mixed. At first, the method describing the spectrum of the resulting light source has to be introduced. It has to be known in order to be able to evaluate the light source. The most convenient way to describe the spectrum of a resulting light source is when all initial light sources are normalized. Normalization of a light source defined by distribution of spectral power $S(\lambda,\lambda_0)$ is performed by finding tristimulus values X, Y, Z and division of the spectral power distribution by the sum of these values. Spectral power distribution of normalized light source is described as follows:

$$S_n(\lambda,\lambda_0) = \frac{S(\lambda,\lambda_0)}{X+Y+Z}. \quad (3)$$

It may be noted, that colour coordinates describing the colour of the light source remain the same after normalization of the light source, so these parameters used in colour mixing equations may be found at any time. Brief description of these parameters is in the next section. Spectrum of resulting light source, after normalization the initial sources, may be found using $S_{sum}(\lambda)$ expression. All further actions are performed using virtual source of such type.

$$S_{sum}(\lambda) = \sum_{j=1}^{3} c_j \cdot S_n(\lambda,\lambda_{0j}) \quad (4)$$

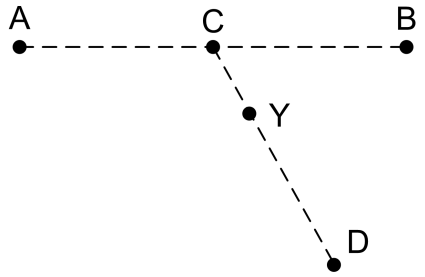

Fig. 2. Initial points A, B, D resulting in a point Y.

1.3. Obtaining colour mixing proportions

The resulting light source of LED mixture has to have particular colour coordinates. This is achieved by deciding, whether it is possible to have a resulting light source of required colour, and, if so, what light source intensities should be chosen. Three initial light sources and a required resulting light source have to be marked in a colour space. The initial sources are marked with points A, B, D and the resulting source is marked with Y - see Fig. 2.

If we need to get point Y(y_1, y_2), with points A(a_1, a_2), B(b_1, b_2) and D(d_1, d_2) given, we have to find at first the point C(c_1, c_2).

$$C(c_1,c_2) = \eta A(a_1,a_2) + (1-\eta)B(b_1,b_2) \quad (5)$$

Since $\eta \leq 1$, point C may be defined as follows:

$$c_1 = \eta a_1 + (1-\eta)b_1; \quad (6)$$

$$c_2 = \eta a_2 + (1-\eta)b_2. \quad (7)$$

Once we have the coordinates of point C(c_1, c_2), we may find the point Y(y_1, y_2) :

$$Y(y_1,y_2) = \mu D + (1-\mu)C; \quad (8)$$

$$Y(y_1,y_2) = \mu D + (1-\mu)\eta A + \Omega; \quad (9)$$

$$\Omega = (1-\mu)B - (1-\mu)\eta B; \quad (10)$$

where $\eta \leq 1$ and $\mu \leq 1$. In order to simplify the expression, we perform a substitution in order to receive resulting equation.

$$\gamma = (1-\mu)\cdot\eta; \quad (11)$$

$$Y = \mu D + \gamma A + (1-\mu-\gamma)B. \quad (12)$$

If we want to get a light source Y(y_1, y_2) from the three known sources A(a_1, a_2), B(b_1, b_2) and D(d_1, d_2) (with colour coordinates (a_1, a_2), (b_1, b_2) are (d_1, d_2), respectively) the parameters of intensity of a source, defined by γ and μ have to be found first. From an equation above, we get:

$$y_1 = \mu d_1 + \gamma a_1 + (1-\mu-\gamma)b_1; \quad (13)$$

$$y_2 = \mu d_2 + \gamma a_2 + (1-\mu-\gamma)b_2; \quad (14)$$

$$y_1 = \gamma(a_1-b_1) + \mu(d_1-b_1) + b_1; \quad (15)$$

$$y_2 = \gamma(a_2-b_2) + \mu(d_2-b_2) + b_2. \quad (16)$$

We apply Cramer's rule in order to solve this system with two unknown variables.

$$\gamma = \frac{\det \gamma}{\det}; \qquad \mu = \frac{\det \mu}{\det}; \tag{17}$$

$$\gamma \cdot (a_1 - b_1) + \mu \cdot (d_1 - b_1) = y_1 - b_1; \tag{18}$$

$$\gamma \cdot (a_2 - b_2) + \mu \cdot (d_2 - b_2) = y_2 - b_2. \tag{19}$$

Rearrange the system we retrieve the solutions from the values of $\det$, $\det \gamma$ and $\det \mu$:

$$\det = (a_1 - b_1)(d_2 - b_2) - (a_2 - b_2)(d_1 - b_1); \tag{20}$$

$$\det \gamma = (y_1 - b_1)(d_2 - b_2) - (y_2 - b_2)(d_1 - b_1); \tag{21}$$

$$\det \mu = (a_1 - b_1)(y_2 - b_2) - (a_2 - b_2)(y_1 - b_1). \tag{22}$$

This means that virtual source Y(y_1, y_2) could be constructed using sources A, B and D representing by colour components c_1, c_2, c_3, respectively.

$$c_1 = \gamma; \tag{23}$$

$$c_2 = 1 - \mu - \gamma; \tag{24}$$

$$c_3 = \mu. \tag{25}$$

It is necessary to point out that the chromaticity point of light source Y may only be a result of mixing if it belongs to a triangle ABD bound by chromaticities A, B and D. This can be checked by solving three inequalities, or simply checking if c_1, c_2, $c_3 \geq 0$ meet. This equality is enough to know that point Y exists and can be found:

$$c_1 + c_2 + c_3 = 1. \tag{26}$$

1.4. Obtaining of total spectrum

Spectra mixing technique presented in section 1.2 works only with sources, whose sum of tristimulus values X, Y and Z equals 1, this does not apply in general. In our case, this condition is not satisfied for LEDs sources. Their max value of a spectrum equals one, but half-width is equal to 30 nm. To get the required properties on a resulting LED, intensity parameters of the initial LEDs need to be transformed. Procedure of transformation could be formulated as follows.

1. Tristimulus values X_i, Y_i, Z_i are calculated for our LEDs, where i = 1..3.

2. Colour components c_1^*, c_2^*, c_3^* are calculated by using technique presented earlier.

3. Normalization of each LED characteristics (i = 1..3) is performed using following equation.

$$LED_i^* = \frac{LED_i}{X_i + Y_i + Z_i} \tag{27}$$

4. Total spectrum of normalized light source LED_{Sum}^* is retrieved from LED_j^* and colour components c_j (j=1÷3).

5. Max value max_{Sum} is found in spectrum LED_{Sum}^*.

$$c_i^* = \frac{c_i}{X_i + Y_i + Z_i} \cdot \frac{1}{max_{Sum}} \tag{28}$$

6. c_i^* values are normalized, to get their total value equal 1.

$$c_1^* + c_2^* + c_3^* = 1 \tag{29}$$

Integration of functions defined in table. Integration is inevitable when performing various operations with LEDs. The integration is performed on functions with values known only in particular points. These values are described in tables, so, in order to integrate such function, it's value has to be approximated to a value known. The way of approximation depends on the occasion, which in our case is the most simple approximation algorithm - discontinuous step function, which may be integrated by a sum of all rectangular areas.

1.5. Obtaining of General Colour Rendering index.

The light source, resulting from a set of separate LEDs and satisfying all requirements, has to be evaluated. The evaluation of a light source is received by computing general CRI value. In case of our problem, the resulting sample emitter has a specific chromaticity (like reference light source). Lets assume the D_{65} emitter as a standard emitter. The correlated colour temperature C_{CT} depends only on a colour parameters x,y [3]:

$$C_{CT} = g_1 \cdot \beta^3 + g_2 \cdot \beta^2 + g_3 \cdot \beta + g_4; \tag{30}$$

$$g_1 = -437; g_2 = 3601; g_3 = -6861; g_4 = 5513.31; \tag{31}$$

$$\beta = \frac{x - 0.3320}{y - 0.1858} \tag{32}$$

Procedure of colour rendering evaluation may be based on spectrum of the emitter D_{65} itself. We normalize the parameters of both sources in a way, that their luminance Y would be equal to 100. Then, in order to evaluate the resulting light source, we calculate parameters X, Y, Z for every initial source and every function of reflection $\rho_i(\lambda)$:

$$X_{j,i} = \int_{360}^{830} a(\lambda) \cdot S(\lambda) \cdot \rho_i(\lambda) d\lambda \tag{33}$$

$$Y_{j,i} = \int_{360}^{830} b(\lambda) \cdot S(\lambda) \cdot \rho_i(\lambda) d\lambda \tag{34}$$

$$Z_{j,i} = \int_{360}^{830} c(\lambda) \cdot S(\lambda) \cdot \rho_i(\lambda) d\lambda \tag{35}$$

where subscript index i stands for reflection function and index j is for sample or a reference emitter. ΔE_i may be found using previously found values of $X_{j,i}$, $Y_{j,i}$, $Z_{j,i}$. The general CRI is found from an average of ΔE_i values. If our sample source gets a CRI value of 100, the computation can be stopped, because it is the perfect emitter.

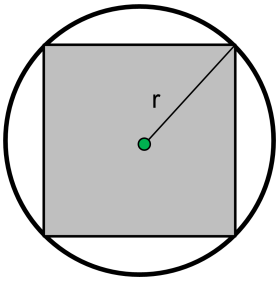

Fig. 3. Example of discardation of square range.

Due to reflection functions, reference light source and the colour are constant, our sample source depends only on these variables $\lambda_{01}, \lambda_{02}, \lambda_{03}$, the general CRI is a function of them: $R_a(\lambda_{01}, \lambda_{02}, \lambda_{03})$.

2. Branch and bound technique

As the basic questions regarding emitter modelling have been answered, the optimization method has to be decided. There are a lot of different techniques, but branch and bound method will be used in further solution of our problem.

2.1. Discarding of search range

It is useful to know when creating the branch and bound algorithm, that $R_a(\lambda_{01}, \lambda_{02}, \lambda_{03})$ satisfies Lipschitz condition (because the first partial derivatives are bounded in the search range).

$$| f(a) - f(b) | \leq L | a - b | \tag{36}$$

L is called a *Lipschitz constant*. It means that for every point c from interval [a÷b], value of a function satisfies these inequalities:

$$f(c) \leq f(a) + L | a - b |; \tag{37}$$

$$f(c) \geq f(a) - L | a - b | . \tag{38}$$

Therefore, satisfying of mentioned condition proposes the candidates to minimum and maximum, although minimum and maximum values could be distributed in next intervals. Accuracy may be increased by dividing them to smaller intervals and discarding these that may not possibly contain global optimal value.

Let's assume, that object function satisfies Lipschitz condition. That allows us to predict the distance, where optimal value can not be found, from our current point. This is a circle in 2D case and a sphere in 3D. Being inconvenient to divide the search range to spheres, it is divided to cubes inscribed in these spheres.

We can see that from one-dimensional case, this procedure differs in such way, that some points remain undiscarded, although it is known that they do not contain the optimal value.

2.2. Division of search range

In our case, the search range is continuous three-dimensional cube. This technique definining the search order and points where object function computation is performed, has to be chosen. Independently from technique, the search range should be divided to smaller ranges. Search range can be divided into a set of strictly identical elements or a set of elements depending on results of computations.

2.3. Division of search range to variable parts

Search range may be divided to a set of elements, where every element depends on the results of computation. Using this method, all search range can be divided into 8 separate parts of cuboid (see Fig. 4) and values may be calculated only in one point of a cuboid boundary. Once it is obvious that particular area may be discarded, the remaining range is divided into 7 parts and every one of them is computed as well.

Octree is sufficient for such range division, but known information for data division is not used efficiently. In addition to that, the same point which is partially computed would be examined again in next computation, so finding the next global maximum may take a while. In order to use our known info more effectively, it is better to examine middle range point instead of a boundary and divide the search range into 27 parts instead of 8 according to Fig. 5.

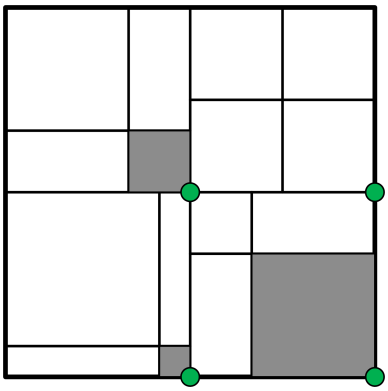

Fig. 4. Example of division of a 2D search range into 4 parts. Circles mark points where the value of function is calculated, grey rectangles mark discarded parts from a search range, white rectangles mark parts to be computed.

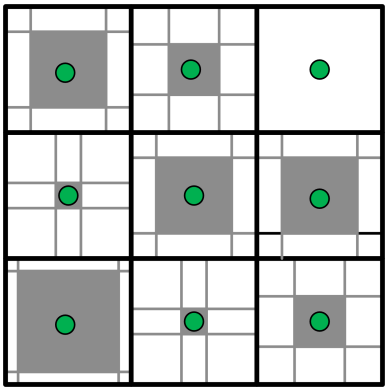

Fig. 5. Example of division of a 2D range into 9 parts, where the mid-range point is examined. Remarks are identical to Fig. 4.

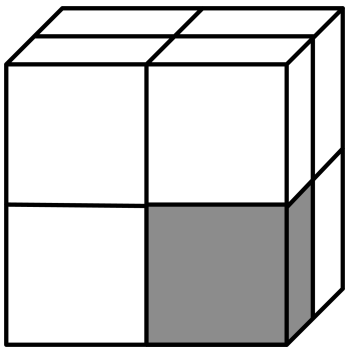

Fig. 6. Example of discrete division of search range into 8 parts.

To manage separate parts of a search range divided this way, a tree where every treetop would have no more than 27 children values. More difficult structure is applied in the second case, but ranges bigger up to 8 times may be discarded, it is also possible that new max values would be found more quickly using the second technique.

2.4. Division of search range into sized parts

Search range may be divided into a predefined amount of even-sized parts, as it is represented by a net in Fig. 6. A point is chosen in every part systematically and, if it is seen that the part can be discarded - it is done so. Otherwise, the part is further divided into even smaller parts and every of them is computed as well. This way, discrete range is received instead of continuous. The advantage of this technique is notice of size of parts that the range is divided into, although it may take by far much more divisions to discard certain areas.

Decision was made that in order to effectively use known information, the mid-point of an area should be investigated by using the dynamic partition technique. To manage it effectively, a tree where every node has no more than k children where k is the amount of parts the range is divided at every step.

Partition of a search area to variable parts enables the reduction of search range every time, so the range would constitute of a set of different sized search areas.

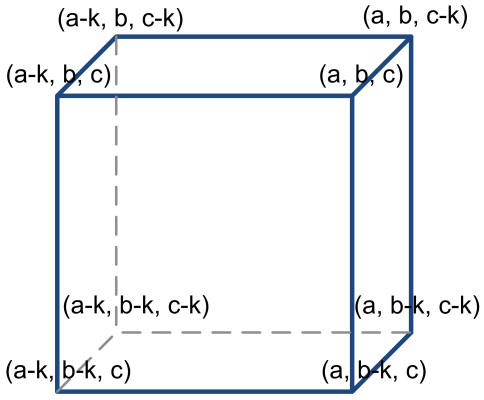

Fig. 7. Boundaries of search area in 3D case.

In order to assign the computation to processors more efficiently after parallelization, it is needed to even out these areas. In order to achieve it, this technique may be transformed to even out the areas or more sophisticated techique might be used. This problem is avoided by using the second division technique where every part is even, though it may be less effective in serial computation because the search range is divided into bigger parts. The second technique was chosen after analysis of every case to make it more effective in parallel computation case.

2.5. Evaluation of search range without solution

In a process of problem solution, quite often occurs a case, when the solution is not at the examined point. It is needed to find out whether the solution exists in the whole search area, and how to evaluate it if it does exists. A closer look has to be taken into the examined area. In 3D case, the area is a cube, represented in Fig. 7.

In order to evaluate whether a point representing the combination of emitters exists in this cube, a set bound by the cube has to be moved to the colour diagram. Separate task was visualised using tridimensional cube model. Fig. 8 represents the search area of point D_{65} by Planck curve.

The most convenient way to be sure that a point belongs to a set accessible with our current light sources is fixing two points, named a^*, b^* and checking whether a solution exists in a set of points c^* through c^*-k. Every possible combination should be checked in this way. An array holding colour characteristics for every wavelength is the most convenient way to analyse variants presented in Table 3.

Importance of Lipschitz constant. Once a proper point is found, its distance from the center of a cube should be evaluated and its minimal value may be found by using the Lipschitz constant. Algorithm calculating the maximal difference between objective function value of light source (when wavelength of one of the initial LEDs changes by 0.5 nm) was used to define the value of Lipschitz constant L, which is essential for the search results.

The resulting value is bigger than it was expected - L=40.9127. This means, that in the worst case, when one LED changes by 1 nm, the CRI may increase or decrease by a value equal to $(2 * L)$=81.8254 nm.

3. Description of algorithms

3.1. Serial algorithm

In serial execution case, the program does not need a structure of data storage. The program execution is performed as follows: data for initial computations is retrieved, the initial search area is formed and, using the recursion, depth search is performed. Operations presented in Table 4 are performed on every program execution step.

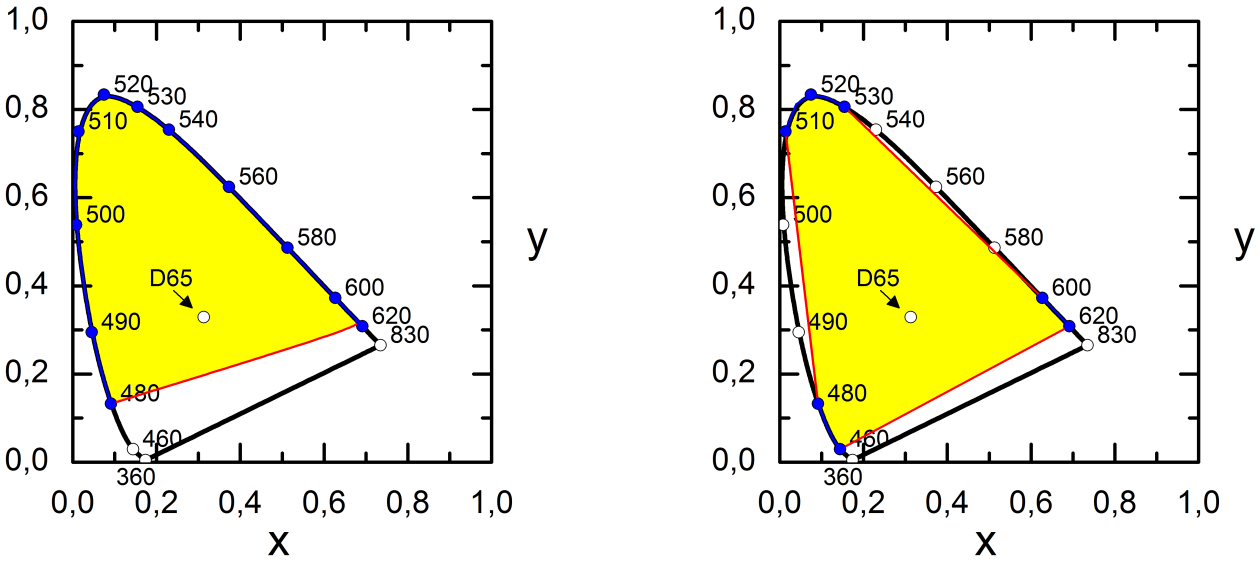

Fig. 8. Colour area bound by a different search cube:
left) a=620, b=620, c=620, k=140;
right) a=620, b=530, c=480, k=20.

3.2. Parallel algorithm

The *master-slave* communication structure is adopted for these computations. This structure defines the way of communication when one process assigns the jobs for the other processes, known as slaves, and retrieval of their results. This communication structure is centralized and the effectiveness in a case of a specific problem depends on the level of communication between the processes.

Four programs implementing parallel computations were investigated. They differ in the way of communication between processes. In the case of the first program, slaves communicate with the master process only to request a job or return a result. In the second program, this is defined by a system of exchanging the information about new or current global max value. In the third program, slaves inform the master process about a new max value found, and, if it is not found for a specific amount of cycles, a slave requests the master for the newest search results. The fourth program is similar to the third one, except the last job which is divided furthermore.

Output data depends on two parameters - Lipschitz constant L and constant of minimal cube border length m. While increasing of Lipschitz constant L, the search velosity decreases, but the accuracy of search increases.

Algorithms were tested in accordance with the following parameters: Lipschitz constant L=81.8254, minimal cube border length m=0.5 nm, reference light source - D65, the initial search range is divided into 125 parts. Additional data is required for the program to operate: colour matching functions, standard 8 - reflection functions, spectrum of a reference light source.

4. Testing and evaluating

In order to evaluate a parallel algorithm, it is essential to know the amount of time it takes for serial algorithm to find a solution - the time amount of time found experimentally was 19231 seconds. The solution was the best combination of emitters defined by wavelengths: 462.996; 540.473; 610.973. The CRI value of the trichromatic system with these wavelengths is 89.1242. The result agrees with condition and outlooks presented in Ref. [2]. Fig. 9 represents the original result: spectral power distributions of standard D65 source and CRI-optimal trichromatic light source.

Parallel algorithm - case of minimal communication. Test results are presented in Table 5a. The initial search range is divided into 125 parts.

Table 3. Search variants for separate task.
Selecter parameters:a=620, b=530, c=480, k=20.

a	b	c
620	530	480÷460
620	510	480÷460
600	530	480÷460
600	510	480÷460
480	530	620÷600
480	510	620÷600
460	510	620÷600
460	530	620÷600
620	480	530÷510
620	460	530÷510
600	480	530÷510
600	460	530÷510

Table 4. Steps of the search algorithm.

1.	Emitter spectra are generated.
2.	Decision whether it is possible to obtain a reference chromaticity from a combination of LEDs is made. If it is true:
2.1.	total spectrum is generated and its CRI is calculated;
2.2.	if the CRI value is higher than any previous, it is memorized along with the point.
3.	If the specific light source can not be composed from current LED combination, a search defining whether there is a point in search area defining the initial emitters suitable for the needed light source, is performed:
3.1.	if a point exists, CRI is evaluated at that point;
3.2.	if the value is higher than any previous, it is memorized as well as the point. Distance from the center of a search range is performed and by using the Lipschitz condition, the lower boundary of a mid-point in cube is evaluated.
4.	If a solution exists and the border of cube is equal to or higher than the determined lower boundary, we make a decision whether another solution may exist in the cube by using the Lipschitz condition. If it may exist:
4.1.	we divide the search cube into 8 parts;
4.2.	perform the search for the first part;
4.3.	if the conditions are still met, the search is performed for the second part and so on.

Master process assigns jobs with initial maximum value to slaves and idles until slave returns his results and, if there are any jobs left, it assigns a new job to a slave. Slaves perform a job similar to the serial algorithm. The highest efficiency was achieved with 4 processors, slight increase in performance is also seen when using 12 nodes. Speedup characteristics are constantly increasing, although the rate drops rapidly, until 20 working nodes were reached.

Parallel algorithm - case of internal (extra) communication. Test results are presented in Table 5b. The initial search range is divided to 125 parts. Master process assigns tasks with initial max value to slaves. Master idly waits for reports on process: it may be either new max value found, or the assigned job complete. Then, if needed, master process updates it's data concerning the max value and transmits it to slave. In a second case, master assigns a new job to slave or, if there are none left, sends him an end message. Slave processes execute a slightly modified parallel algorithm. They arrange the initial data and create a structure storing the search ranges. After that, they evaluate the mid-point of cube. If a new max value is found during that, it is sent to master process. After that, slave process waits for an update of currently highest max value. Once slave finishes its work it sends a message to master process and either gets a new job or an end message.

In this case, the highest efficiency value is received using 4 nodes aswell, the second way of increase in efficiency is seen when node count reaches 12. The highest point of speedup value is achieved when using 20 nodes and after that, even a drop is noticeable. It is possible, that it is a consequence of higher level of communication between nodes.

Parallel algorithm - case of moderate communication. Test results are presented in Table 6a. This algorithm is analogical to one used in the second case, except the slave process has an extra counter, which is used to synchronize the max value with the master process.

Table 5. Test results of parallel algorithms: a) case of minimal communication; b) case of internal (extra) communication.

a)				b)			
Node count	Completion time	Speedup	Efficiency	Node count	Completion time	Speedup	Efficiency
2	15204.80	1.265	0.632	2	15776.45	1.219	0.609
3	8192.25	2.347	0.782	3	8384.12	2.294	0.765
4	5842.66	3.291	0.823	4	5795.79	3.318	0.830
5	4753.81	4.045	0.809	5	4766.80	4.034	0.807
6	4254.51	4.520	0.753	6	4115.99	4.672	0.779
7	3810.73	5.047	0.721	7	3921.33	4.904	0.701
8	3577.95	5.375	0.672	8	3542.58	5.429	0.679
9	3242.96	5.930	0.659	9	3211.48	5.988	0.665
10	2958.71	6.500	0.650	10	2892.39	6.649	0.665
12	2378.63	8.085	0.674	12	2278.54	8.440	0.703
15	2151.00	8.941	0.596	15	2082.21	9.236	0.616
20	1974.89	9.738	0.487	20	1827.99	10.520	0.526
25	1975.22	9.736	0.389	25	1944.87	9.888	0.396
30	1986.74	9.680	0.323	30	1901.59	10.113	0.337
35	1884.70	10.204	0.292	35	1923.72	9.997	0.286

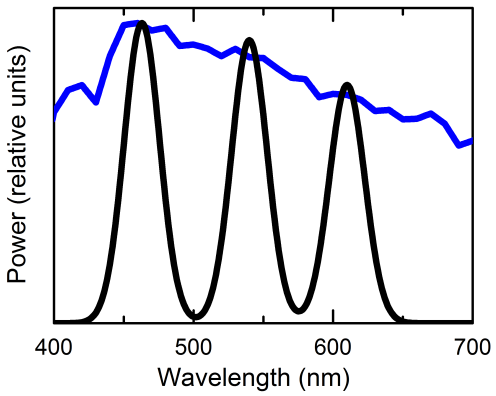

Fig. 9. Spectral power distributions of standard D65 (blue) and CRI-optimal trichromatic (black) light sources.

If the slave does not find a new max value in a specific amount of cycles (20 in this case), it sends a message to master process requesting an update on current highest max value. In this case, similarly to the others, the highest efficiency is achieved using 4 nodes, the next increase, although slighter one is seen at 12 nodes. The max speedup value is reached with 30 nodes.

Parallel Algorithm - 2nd case of moderate communication. Test results are presented in Table 6b. This algorithm is similar to the 3rd one, except the last job is divided to smaller jobs. This is used in order to reduce the idle time on nodes.

In this case, an increase in efficiency is visible until node count does not exceed 6. There is a possibility, that in order to increase the efficiency with more than 6 nodes, several last jobs should be divided instead of just the last one. As observed previously, the highest efficiency is achieved with 4 nodes and slight increase is seen at 12 nodes. The growth in speedup drops drastically, when node count reaches 25.

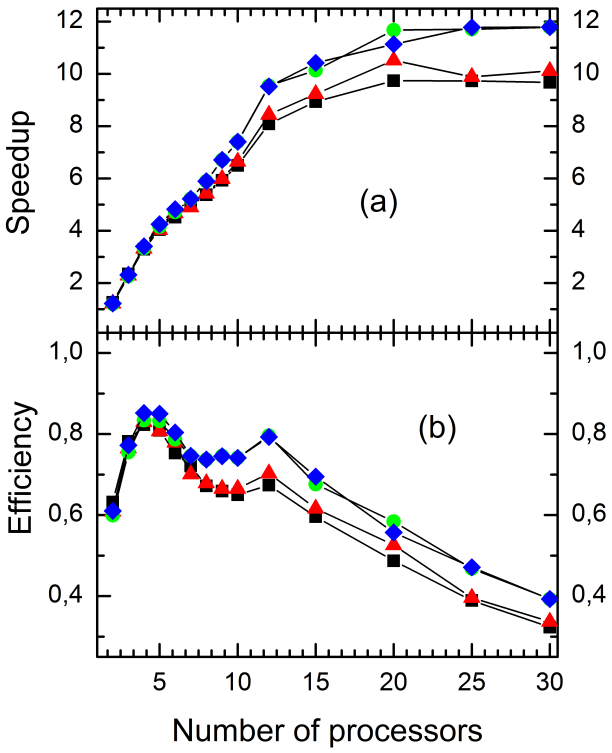

Fig. 10. Algorithm speedup dependence (a) and algorithm efficiency dependence (b) on number of CPU.
Red triangle - internal communication; Black square - minimal communication; Green circle - regular communication; Blue diamond - regular communication, case 2.

Comparison of algorithms. Fig. 10 visualizes speedup dependence and efficiency dependence on number of CPU.

Table 6. Test results of parallel algorithms, regular (constant) communication: a) 1st case; b) 2nd case.

a)				b)			
Node count	Completion time	Speedup	Efficiency	Node count	Completion time	Speedup	Efficiency
2	16063.44	1.197	0.599	2	15751.05	1.221	0.610
3	8490.68	2.265	0.755	3	8307.36	2.315	0.772
4	5768.01	3.334	0.834	4	5644.77	3.407	0.852
5	4624.90	4.158	0.832	5	4523.91	4.251	0.850
6	4073.22	4.721	0.787	6	3988.68	4.821	0.804
7	3683.34	5.221	0.746	7	3684.91	5.219	0.746
8	3259.90	5.899	0.737	8	3261.36	5.897	0.737
9	2866.05	6.710	0.746	9	2867.01	6.708	0.745
10	2595.43	7.410	0.741	10	2595.88	7.408	0.741
12	2013.87	9.549	0.796	12	2020.50	9.518	0.793
15	1896.29	10.141	0.676	15	1845.22	10.422	0.695
20	1646.92	11.677	0.584	20	1726.12	11.141	0.557
25	1642.23	11.710	0.468	25	1632.17	11.783	0.471
30	1632.18	11.782	0.393	30	1631.44	11.788	0.393
35	1632.45	11.780	0.337				

It is obvious that speed of computations increases the most until the node count does not exceed 12. Also, as observed more than 20 nodes for these computations should not be used either, because the speedup increase is slight or even negative. During parallel computations efficiency maximum value is achieved with 4 nodes, slight increase is seen at 12 nodes as well.

We can see in the chart, that the curves of efficiency for every algorithm are almost identical until node count reaches 4. The highest efficiency is achieved by a program of regular communication with the division of the last job. In the case, when we want to get the solution faster by using more nodes, it is best to use 12 nodes and one of the regular communication programs. In addition to that, it might be useful to try to divide several of the last jobs to reduce the idle time even more.

Conclusions

The optimization problem was successfully solved and the speedup of parallel algorithm was investigated experimentally. It was found that the magnitude of the Lipschitz constant has huge affect on the overall computing time. The most "interesting" ranges with positive CRI value have much lower Lipschitz value, therefore in order to reduce the computation time, several Lipschitz values could be used: for positive and negative CRI values. Also, a technique could be developed to discard negative search areas more efficiently, but ensuring that none of the possible optimal points were discarded.

In this work, technique enabling modelling of multicolour light sources was examined. A branch and bound algorithm was adapted for solving this problem and software, enabling the search of optimal solution on a distributed computer network, was created. Several parallel algorithms using centralized data exchange scheme were created and tested. There was no need for a decentralized data exchange scheme as communication between processes is not frequent (new max value is found rarely).

VU MIF distributed computer network was used to test these algorithms and recommendations for such algorithm performance on such computing network were given. During the research, the node count providing the best efficiency was determined: the highest efficiency is achieved with 4 nodes, but there is always an increase in efficiency at around 12 nodes. This may be useful, when the problem has to be solved quickly by increasing the amount of nodes, to maintain efficiency. In all cases, using more than 20 nodes should be avoided, because the speedup increase is minor or even negative, therefore the efficiency drops significantly.

References

1. Aistė Ambrazevičiūtė, Rimantas Vaicekauskas. Modelling of multicolour light source using distributed computing network. 1. Statement of the physical and computational problems – *Innovative Infotechnologies for Science, Business and Education,* ISSN 2029-1035 – 1(10)(2011) 13-20.
2. Žukauskas A., Vaicekauskas R., Ivanauskas F., Gaska R. and Shur M. S. Optimization of white polychromatic semiconductor lamps – *Appl. Phys. Lett.* 80 (2002) 234–6.
3. Wyszecki, G. and Stiles, W. S. Color Science: Concepts and Methods, Quantitative Data and Formulae – New York: Wiley, 2000.