jet INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2010, VOL. 56, NO. 2, PP. 145-152
Manuscript received April 30, 2010; revised June, 2010. 10.2478/v10177-010-0019-y

Comparison of Evolutionary Algorithm and Heuristics for Flow Optimization in P2P Systems

Michal Kucharzak, Adam Siwek, and Krzysztof Walkowiak

***Abstract*—Nowadays, many Internet users make use of Peer-to-Peer (P2P) systems to download electronic content including music, movies, software, etc. Growing popularity in P2P based protocol implementations for file sharing purposes caused that the P2P traffic exceeds Web traffic and in accordance with to many statistics, P2P systems produce a more than 50% of the whole Internet traffic. Therefore, P2P systems provide remarkable income for Internet Service Providers (ISP). However, at the same time P2P systems generates many problems related to traffic engineering, optimization, network congestion. In this paper we focus on the problem of flow optimization in P2P file sharing systems. Corresponding to BitTorrent-based systems behaviour, the optimization of P2P flows is very complex and in this work we consider different heuristic strategies for content distribution and moreover we propose a new evolutionary algorithm (EA) for this problem. We compare results of the algorithms against optimal results yielded by CPLEX solver for networks including 10 peers and relation to random algorithm for 100-node systems. According to numerical experiments, the EA provides solutions close to optimal for small instances and all of the heuristics exhibit a superior performance over random search.**

***Keywords*—P2P, flows, network optimization.**

I. Introduction

FOR AT LEAST the past decade we have been experienced an explosion and extremely growing popularity in Peer-to-Peer systems (P2P) based applications for content distribution purposes. Now the P2P are widely used mechanisms to share resources via Internet. Very popular systems were designed to share CPU (*Seti@Home, XtremWeb, Entropia*), publish files (*Napster, Gnutella, Kazaa, BitTorrent*), realize Internet based telephony (*Skype*) or Internet television (*Joost*). Furthermore some systems were designed to share disk space (*Intermemory* [6], *PAST* [9], *OceanStore* [16]). In accordance to many statistics P2P is still producing more traffic in the Internet then all other applications combined. Its average proportion during the measurement period regionally varies between more than 40% in the Middle East and about 80% in Eastern Europe [1], [13], [14], [25]. Many Internet Service Providers (ISP) and other telecommunication operators cope with the problem how to decrease costs generated by the P2P systems [1]. On the whole it would be decisive to reduce this load simultaneously maintaining the performance of P2P systems at acceptable level, particularly for the reason that P2P traffic is expected to stay the largest share of Internet for next years [7].

This work is supported by The Polish Ministry of Science and Higher Education under the grant which is being realized in years 2008-2011.

M. Kucharzak, A. Siwek and K. Walkowiak are with the Department of Systems and Computer Networks, Wroclaw University of Technology, Poland (e-mail: michal.kucharzak@pwr.wroc.pl, adams47@o2.pl, krzysztof.walkowiak@pwr.wroc.pl).

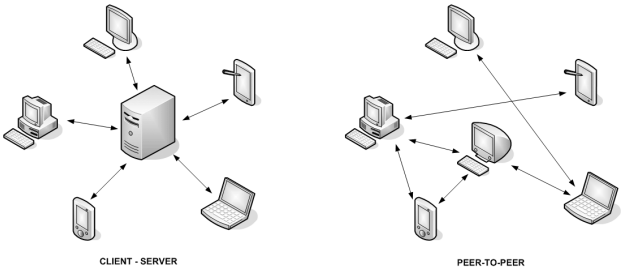

Fig. 1. Client-Server vs. Peer-to-Peer architectures.

A P2P system is defined as a system, in which each node acts both as a server (producer that provides data to other nodes) and as a client (consumer that retrieves data from other nodes, Fig. 1). P2P systems usually are built as overlay networks, i.e. they work on the top of underlying computer network. Peer-to-Peer systems can be divided into two categories: unstructured - the content stored on a given peer is unrelated and does not have any specific structure and structured - mainly based on Distributed Hash Tables (DHT) that provide a global view of data distributed among many peers independent of actual location. Unstructured P2P systems can be: centralized - a central server that stores only information (e.g. IP addresses) of peers where some content is available (e.g. Napster), pure - the system contains no central server and relies on flooding the information on desired content over the network (e.g. Gnutella 0.4 and Freenet) and hybrid - using a hierarchy of superpeers - servers that store content available to the connected peers together with their IP address (e.g. Gnutella 0.6 and JXTA) [25].

In this paper we focus on offline optimization of flows in P2P systems. We are aware of the fact that many P2P systems are very dynamic and stochastic, so offline modelling of such systems can be viewed as inaccurate. However, it is a common research approach that difficult and dynamic systems are reduced to simpler cases to enable creation of models that can be solved optimally in a reasonable time. Thus, some benchmarks that allow us to estimate performance of real systems can be obtained. Since the Integer Programming formulation of the flow optimization in P2P systems is very complex, only for very small networks we can use exact algorithms based on the branch-and-cut method [26], [27]. Therefore, in this work we propose an effective heuristic algorithm based on evolutionary approach for optimization of flows in P2P file sharing systems and present 4 hesuristics

which realize different strategies for content distribution which we previously described in [17].

The remainder of this paper is organized as follows. In the next section we discuss the related work on analysis and optimization of P2P systems. Section III includes the Integer Programming optimization model of flows in P2P systems. In Section IV we describe heuristic searches with their pseudocodes: Random Strategy, Cost Selection Strategy, Transfer Cost Strategy and Shortest Transfer First. Our Evolutionary Algorithm is presented in Section V. Section VI contains results of experiments. Finally, concluding remarks are provided in the last Section.

II. Related Work

Killian *et al.* consider the overlay network content distribution problem [15]. All content is organized in the form of unit-sized tokens - files can be represented as sets of tokens. The distributed schedule of tokens proceeds as a sequence of time-steps. There is a capacity constraint set on each overlay arc, i.e. only a limited number of tokens can be assigned to an arc for each time-step. Two optimization problems are formulated: Fast Overlay Content Distribution (FOCD) and Efficient Overlay Content Distribution (EOCD). The goal of the former problem is to provide a satisfying distribution schedule of minimum number of time-steps. The latter problem aims at minimizing the number of tokens' moves. Both problems are proved to be NP-complete. An Integer Program formulation of EOCD is presented. Various online approximation algorithms for distributed version of overlay content distribution problem are proposed and tested.

Authors of [10] consider the problem of transferring a large volume of data to a set of clients in the shortest possible time. A cooperative scenario under a simple bandwidth model is solved in an optimal solution involving communication on a hypercube-like overlay network. Also non-cooperative scenarios based on the principle of barter are discussed.

In paper [34] the performance characteristics of 2nd generation P2P applications, e.g., BitTorrent (BT) is examined including: construction of a deterministic model, analytical calculations of the average delay, proposition of a branching process model for a P2P system in the transient regime, and presentation of a steady state analysis for a P2P service capacity based on Markov chain model. Authors also evaluate some traces obtained from real BitTorrent network.

In [31] several protocols developed for P2P based file distribution are proposed. Moreover, authors introduce a centrally scheduled file distribution (CSFD) protocol, to minimize the total elapsed time of a one-sender-multiple-receiver file distribution task is proposed. A discrete-event simulator is applied to study the performance of CSFD and other approaches (e.g. BitTorrent).

Authors of [3] propose several routing algorithms to distribute data blocks on a network with limited diameter and maximum degree. The time scale of the system is divided into steps. A special attention is put on upload policy - a randomized approach is proposed and examined.

Authors of [20] propose a probabilistic model of coupon replication systems in order to study a P2P file swarming system based on BitTorrent. Markov processes are used to find necessary and sufficient stability conditions.

In [30] a new selection strategy for BitTorrent-like P2P systems is proposed. The major objective is to reduce the download time of BitTorrent. The proposed approach is based on the greedy strategy that a peer assigns each missing piece a weight according to total number of neighbour's downloaded pieces. Next, the peer selects the missing piece with the highest priority for next download. The simulation run on a discrete-event simulator shows that the new strategy can improve more than 15% average download time and reduce in average 60% total elapsed time comparing to the BitTorrent system.

There are some works that examine the problem of P2P-based streaming content distribution in overlay networks e.g. [2], [5], [29], [36]. The common assumption of these works that significantly simplifies the analysis is that the content is distributed via a multicast tree, i.e. all subsequent blocks (pieces) of the same content stream are transported on the same paths. Thus, there is no need to model the time scale of P2P system as subsequent time steps and to incorporate to the model the constraint on block possession. In contrast, the parts of non-streaming content can be distributed in P2P systems autonomously, i.e. different blocks can follow different paths in the overlay network, what yields additional constraints in the model.

In our previous works [26], [27] we proposed a generic approach to offline modelling of flows in P2P systems. We focus exclusively on P2P file sharing systems. We show how various constraints following from real P2P systems can be formulated. The formulations can be applied for designing and optimization of P2P systems. To illustrate our approach we present results obtained from CPLEX solver. These works are extended with more details and new aspects of P2P systems and/or algorithms in our previous papers [17], [18], [28]

To find more information on additional aspects of P2P systems refer to [25].

III. Problem Formulation

The problem of flows in P2P systems was formulated in our previous works [26], [27]. Therefore, in this section we only briefly describe the most important elements of the model. Our formulation follows from previous works on this subject [2], [3], [4], [10], [15], [19], [20], [22], [23], [31], [32], [34].

The main issue in formulating an offline model of P2P system is stochastic nature of these systems. Therefore, in this work we assume that the time scale of the P2P system is divided into time slots that can be interpreted also as subsequent iterations of the systems. For the sake of simplicity each time slot has the same length. All actions of the system completed in the previous iteration are available in the beginning of the next time slot. For instance, if block b was transferred to vertex (peer) v in time slot t, then all other peers can try to get this block from v in time slot $t+1$. This assumption is based on the fact that there is an indexing system that provides detailed information on the current availability of blocks in the system. Our model is not limited to one exact implementation of the index, i.e. both a centralized (e.g. similar to BitTorrent [8]) or decentralized (e.g. DHT [25]) approaches can be used.

Data (content) to be transmitted in the P2P system is divided into blocks (pieces) of the same length, e.g. 256 KB. We assume that the transfer of one block is completed within one time slot. Furthermore, each peer participating the transfer wants to receive all data, so each block must be delivered to each node (peer). We consider an overlay network and each peer has a limited upload and download capacity. We do not take into account capacity constraints on overlay links. According to analysis presented in [36], node capacity constraint is sufficient in overlay networks. Moreover, the underlay core network of the overlay usually is considered as over-provisioned and the only bottlenecks are access links [2], [22], [31].

As the optimization objective we use the cost of transfer. Currently used P2P systems mostly ignore the underlying Internet topology and ISP link costs, since they are designed to randomly choose logical neighbors [33]. Thus, there are many cross-continental downloads that can potentially congest backbone networks. To estimate the transfer cost, it is necessary to provide an effective mechanism for localization of peers by using for instance [12], [35]: IP location databases, IP prefix, traceroute records, hop number and RTT (round-trip time). To denote the cost of one block transfer between peers w and v we use constant c_{wv}, which can be understood as number of hops between w and v, number of ISPs between w and v, RTT between w and v, distance in kilometres between w and v, cost of cross-ISP transfers, etc.

The optimization model of the P2P transfer is formulated as an Integer Program (IP) with binary variables. The objective is to transfer to each node (peer) all blocks in a given number of iterations (time slots) minimizing the transfer cost. We use the notation proposed in [24].

Basic flow problem in P2P file sharing systems:

indices

$b = 1, 2, ..., B$ blocks (chunks, pieces of content)
$w, v = 1, 2, ..., V$ vertices (peers, network nodes)
$t = 1, 2, ..., T$ time slots (iterations)

constants

g_{bv} = 1 if block b is located in node v before the P2P transfer starts (v is the seed); 0 otherwise (binary)
c_{wv} defines transfer cost of one block between from peer w to v
u_v upload capacity of peer v in each time slot (integer)
d_v download capacity of peer v in each time slot (integer)
M large number

variable

y_{bwvt} = 1 if transfer of block b to node v from node w starts in iteration t; 0 otherwise (binary variable)

objective

$$\text{minimize} \quad F = \sum_b \sum_w \sum_v \sum_t y_{bwvt} c_{wv} \tag{1}$$

constraints

$$g_{bv} + \sum_w \sum_t y_{bwvt} = 1 \quad b = 1, ..., B \quad v = 1, ..., V \tag{2}$$

$$\sum_v y_{bwvt} \leq M(g_{bw} + \sum_{i<t} \sum_s y_{bswi}) \quad b = 1, ..., B \quad w = 1, ..., V \quad t = 1, ..., T \tag{3}$$

$$\sum_b \sum_v y_{bwvi} \leq u_w \quad w = 1, ..., V \quad t = 1, ..., T \tag{4}$$

$$\sum_b \sum_w y_{bwvi} \leq d_v \quad v = 1, ..., V \quad t = 1, ..., T \tag{5}$$

In accordance to flow cost optimization the main goal of that, the problem objective (1) is addressed to assign the flows in the cheapest way. The result of optimization - y vector represents sequence of blocks transfers in the P2P network. To meet the requirement that each block must be transported to each network node we introduce the condition (2). It refers to completion constraints. Thus, either peer v posses block b before P2P process starts ($g_{bv} = 1$) or block b must be downloaded by peer v from any peer w exactly once during P2P transfer ($\sum_w \sum_t y_{bwvt} = 1$). It is possible to download block b from peer w in time slot t if and if only the block b is located in peer w before time slot t. To meet the possession and precedence requirement, formula (3) is introduced. Constraint (3) satisfies the key feature of P2P systems. Note that the right-hand side of possession and precedence inequality is a sum of constant $g_{bw} = 1$ (equals to 1 if block b is located in node w) and $\sum_{i<t} \sum_s y_{bswi}$ (=1 if block b is transferred to node w from any node s in any iteration i preceding the current time slot t). Consequently, the right hand side of (3) equals 1 only if it is possible for node w to upload block b in time slot t. Note that the constant M in (3) must be at least equal to $V - 1$ in view of the fact that peer w might upload each block at most $V - 1$ times in case w is the seed. For any node different than seed it can upload at most $V - 2$ transfers. Maximum upload rate in each time slot t defines constraint (4). Given formula assures that the number of blocks uploaded by node w cannot exceed a given threshold u_w. Additionally, the maximum download rate of node v constraints (5) is introduced by analogy to (4). Formulas (4) and (5) arise from the assumption that each node (peer) has a limited capacity of access link to the network.

The model is sufficient to mirror only the major characteristics of the BitTorrent-like P2P system. However, for additional constraints that can be incorporated to the model please refer to our previous works [17], [18], [26], [27], [28].

IV. Heuristic Algorithms

This section is devoted to brief description of heuristic algorithms that we developed in order to simulate BitTorrent-like P2P system. Our approaches follow mainly from the BitTorrent file sharing protocol. However, some simplifications had to be made in order to adjust the heuristics to the optimization problem presented in the previous section and previous works. Since the goal of our research is to examine different strategies for P2P flows assignment and compare

them to evolutionary algorithm and optimal results, we do not simulate more details of the systems than modelled in Section III.

A. Random Strategy (RS)

Mostly randomized algorithm is presented in the pseudocode below:

```
1  procedure RANDOM STRATEGY
2    INITIALIZE t=1;
3    while TransferIsNotCompleted do
4       while IsTransferPossible(t) do
5          v=selectRandomPeerToDownload(t)
6          b=selectRandomBlockToDownload(v,t)
7          w=selectRandomPeerToUpload(b,v,t)
8          TransferBlock(b,w,v,t)
9        end while
10       t=t+1;
11   end while
12 end procedure
```

First of all, simulations refer to synchronous model, thus the system works in iterations. All of the presented algorithms consist of main loop as it is shown in lines 3-11. Flow is assigned until $TransferIsNotCompleted$ and completion constraint is not satisfied. For exact time slot t algorithm performs transfers until the transfer is possible (inner loop 4-9) then system state proceeds to the next time slot. To model the stochastic nature of BitTorrent-like P2P system the RS heuristic randomly select the download peer v among all feasible peers (line 5). A download peer v is feasible if it can download at least one block from other peer satisfying all constraints of the system. Next, a block b to be transferred is also chosen randomly among all feasible blocks (line 6). It is possible to download block b if at least one node can upload this block to v under all constraints of the system. Finally, the uploading peer w is randomly selected among all feasible upload peers (line 7). Function $TransferBlock(b, w, v, t)$ (line 8) transfers block b from w to v in time slot t and updates state of the P2P system (upload and download limits and possession of the block). Note that, the main loop of the algorithm (lines 3-11) provides meeting requirement of completion but the last time slot (maximum t stated as T in the model demonstrated in previous section) is not defined. The total P2P processed time may varies in dependence on current blocks' transfer performance.

B. Cost Selection Strategy (CSS)

The third approach - Cost Selection Strategy (CSS) - takes into account transfer costs.

```
1  procedure COST SELECTION STRATEGY
2    INITIALIZE t=1;
3    while TransferIsNotCompleted do
4       while IsTransferPossible(t) do
5          v=selectRandomPeerToDownload(t)
6          b=selectRandomBlockToDownload(v,t)
7          w=selectCheapestPeerToUpload(b,v,t)
8          TransferBlock(b,w,v,t)
9        end while
10       t=t+1;
11   end while
12 end procedure
```

The block b to be transferred is selected at random (line 6), but the closest (in terms of the cost), feasible peer w is chosen for upload the block b (line 7).

C. Transfer Cost Strategy (TCS)

The modification of CSS can be modelled as Transfer Cost Strategy (TCS). The pseudocode of TCS is presented below:

```
1  procedure TRANSFER COST STRATEGY
2    INITIALIZE t=1;
3    while TransferIsNotCompleted do
4       while IsTransferPossible(t) do
5          v=selectRandomPeerToDownload(t)
6          w=selectCheapestPeerToUpload(v,t)
7          b=selectRandomBlockToDownload(w,v,t)
8          TransferBlock(b,w,v,t)
9        end while
10       t=t+1;
11   end while
12 end procedure
```

In contrary to CSS, the TCS algorithm models case in which randomly selected peer from the (list of feasible peers, in line 5) gets the cheapest possible connection to any uploading node w (function in line 6). Eventually block b to be transferred is selected randomly (line 7). Note that, the sequence of functions $selectCheapestPeerToUpload$ and $selectRandomBlockToDownload$ changed and due to that fact functionalities of both of them are modified also.

D. Shortest Transfer First (STF)

The last of the described algorithm - Shortest Transfer First (STF) - realizes the cheapest possible transfer by cheapest possible transfer synchronously in global meaning. Practically, the STF is unable to be implemented in real BitTorrent-like system because P2P architecture is characterized by decentralized form. However we propose the STF strategy to obtain benchmark results.

```
1  procedure SHORTEST TRANSFER FIRST
2    INITIALIZE t=1;
3    while TransferIsNotCompleted do
4       while IsTransferPossible(t) do
5          (w,v)=selectPeersToCheapestTransfer(t)
6          b=selectRandomBlockToDownload(w,v,t)
7          TransferBlock(b,w,v,t)
8        end while
9        t=t+1;
10   end while
11 end procedure
```

Function $selectPeersToCheapestTransfer$ in line 5 returns pair of nodes: uploading w and downloading v that transfer of any block is feasible in time slot t. To finalize transfer, block b is selected randomly from the list of all block which are possible to be proceeded between peers w and v (line 6).

V. EVOLUTIONARY ALGORITHM

Major issue when designing evolutionary algorithms is to cast a solution into a single specimen [21]. Representation of the solution must be simple and created in a way that allows creating further solutions in a evolutionary process called

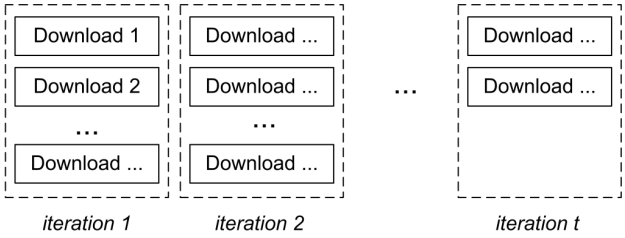

Fig. 2. Chromosome coding (single solution).

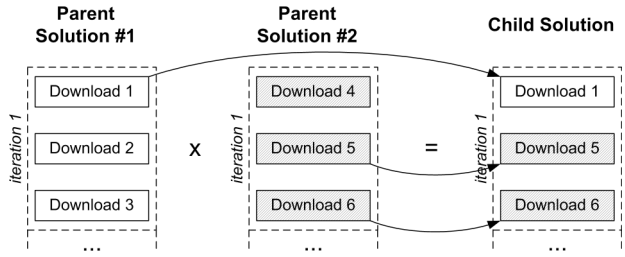

Fig. 3. Crossover operation.

crossover. In this section authors suggest a two dimensional data structure modelling the entire process of exchanging blocks. Root of the model is a list of data structures called "Iteration". "Iteration" consists of list containing specific transfers that were executed in this iteration. Fig. 2 explains this concept.

This approach is very simple and allows to create child solution from several parent solutions. Crossover is performed on 2 or more parent solutions, and creates one child solution. When creating new solution, next downloads from specific iteration from the parents are taken and inserted to the same iteration in child solution after checking if they fit all the constraints. Fig. 3 generally explains the crossover in the single iteration.

Order of inserting transfers from parent donors doesn't have to be specified as presented above. In the beginning of the crossover a random decision is made, whether to take one download from one parent and move to the next parent, or each time choose donor randomly.

With these settings results of improvement in next generations of solutions were best. It's obvious that not all transfers are suitable and doesn't fit the constraints. In the process of adding new transfer there is a third party function that checks all the constraints, if transfer is correct, it is added to child solution, otherwise discarded. There is no guarantee that created solution will cover all transfers needed. There was a need to create a function that will complete the new solution. Completing function takes turns and revises iterations one by one and adds one transfer to each. Once it reaches the last iteration, it comes back to the beginning. Function works till solution is complete or it is impossible to add next transfer. Adding a transfer can be executed in two ways. The transfer with lowest possible cost in current moment can be inserted, or a random transfer. Decision which one to chose is made randomly. The odds of picking the best - lowest cost transfer is 80% and random 20%. The same completing function is used to create base solutions to start population, because the task is the same - create a correct, complete solution. In contrary to completing solution which is a result of crossover, when creating new solution the odds of inserting the best - lowest cost transfer is 20% and random 80%. All other procedures are performed in the same way.

It is obvious that crossover is performed with certain amount of randomness. It eliminates a need to enrich and differentiate pool of solutions using additional mutation. In this case completing function has two tasks: fill the missing transfers in order to create correct solution, and bring diversity to population of solutions.

Other evolutionary operators are a matter of user's choice. In this work we tested the following combinations of operators: 4 selection methods (roulette, strongest, youngest, random), 3 breeding methods (strongest, youngest, random) and 3 removing methods (oldest, weakest, hybrid).

Variety of problems solvable by evolutionary algorithms is very wide. Thus it's obvious that algorithm must be adapted to the problem. Besides the problem coding and crossover described above, authors suggested following modifications that improved quality of solutions of the considered problem in a great way.

First, we introduce a kind of global mutation operator called *shot*. Evolutionary algorithm base on executing a certain amount of iterations, during which population breeds and some specimens die and generation after generation solutions improve. During many experiments with the evolutionary algorithm authors spotted that solution's quality no longer improves after performing about 1000 iterations on one population. The improvement is lesser even after 500 iterations. Therefore, authors suggested performing less iterations on many start populations. Creating one start population and performing few hundreds iterations on it was called a shot. This solution combined with inserting best specimen (solution) into next start populations gave extreme improvement in solution's quality and prevented optimization from stopping in local minimum of objective function.

Another modification is related to *sections* - nodes are divided into sections based on their distance between each other or having same ISP. While choosing transfers during creating, crossing over and completing solutions, sections mechanism prevents algorithm from picking transfers that aren't developmental. I.e. transferring block from remote section while it's present in the same section or closer, or transferring block from other ISP when it's present in a node that belongs to the

TABLE I
CUSTOMIZABLE PARAMETERS OF THE EA

Parameter	Min	Max
Start population count	30	300
Parents count	2	10
Population to reproduce [%]	10	100
Population to remove [%]	0	80
Iterations per shot	10	100000
Shots	1	1000

same ISP. Sections mechanism prevents high cost transfers and as an effect, reduces cost of entire solution.

Customizable parameters of the algorithm are listed in Table I, together with range they can be set in.

VI. Experimentation Results

To solve the model (1)-(5) in optimal way we apply CPLEX 11.0 solver [11]. The evolutionary algorithm and all heuristics were implemented in C#, Visual Studio 2008.

The methodology of tests was similar to [26], [27]. To compare results of the evolutionary algorithm and heuristics against optimal results, we had to limit sizes of the problem instances in order to obtain optimal results approximately in one hour. After several experiments we decide to test networks consisting of 10 vertices (peers), 3 blocks to be transferred and 4 time slots. According to [12] we assume that peers are located in large cities worldwide. Unnecessary P2P transfers crossing different ISPs, countries and continents increase the operating cost of an ISP significantly [1], [12]. To examine this fact, we set the cost of one block transfer between two peers located in two cities just as the distance in kilometres between these two cites. We consider three networks: E5A5 - 5 peers in Europe and 5 peers in North America, E7A3 - 7 peers in Europe and 3 peers in North America, and E10 - 10 peers in Europe.

In the next part of our experiments we run for larger networks consisting of 100 peers, with 50 blocks to be transferred and the number of time slots set to 40. Two networks were considered: E50A50 (50 peers in Europe and 50 peers in North America) and E50A30A20 (50 peers in Europe, 30 peers in North America and 20 peers in Asia). First, authors carried out tests to tune the algorithm and determine best configuration of parameters. Test were made for 20 10-node networks. The best configuration is reported in Table II. Fig. 4 shows the cost as a function of iteration and shots number.

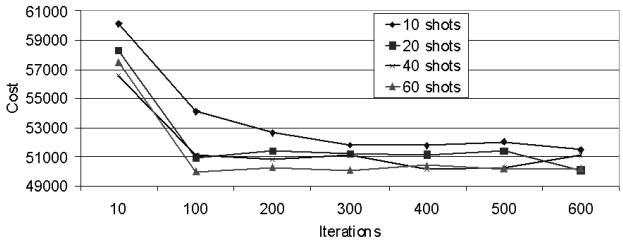

Fig. 4. Results dependency on iterations and shots.

TABLE II
Best Configuration of the EA

Parameter	Value
Start population count	100
Parents count	2
Population to reproduce [%]	50
Population to remove [%]	20
Iterations per shot	300
Shots	40
Selection type	Strongest
Breeding type	Random
Removing type	Weakest

TABLE III
Average Percentage Distance in Relation to Optimal Results

Scenarios	EA	RS	CSS	TCS	SFT
E5A5	**2.91%**	80.8%	27.4%	16.4%	116%
E7A3	**2.43%**	66.9%	16.9%	14.4%	70.2%
E10	**1.53%**	48.3%	22.6%	21.1%	24.5%

TABLE IV
Average Percentage Gap to Random Strategy Results

Scenarios	EA	CSS	TCS	SFT
E50A50 (a)	**72.7%**	52.5%	64.5%	56.2%
E50A50 (r)	**74.1%**	61.7%	68.1%	67.9%
E50A50 (s)	74.6%	59.2%	65.0%	**79.8%**

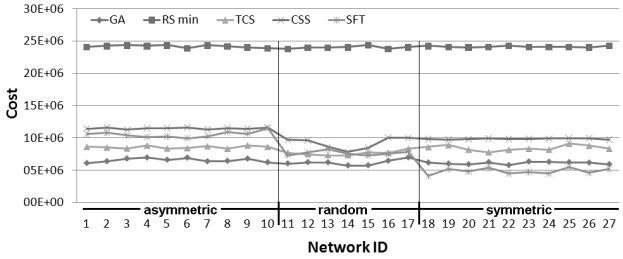

Fig. 5. Comparision of algorithms for 100-node scenarios with different types of access links.

The next goal of experiments was to compare the EA and heuristics against optimal results generated by CPLEX solver. In Table III we report an average percentage gap between algorithms in relation to the optimal results 3 different scenarios (about 30 networks was tested for each scenario). The EA provides results close to optimal - the average gap is about 2%, moreover it exhibits a superior performance over other heuristics. The EA outperforms RS by about 78%, CSS by 25%, TCS by 14% and STF by over than 113%. Each heuristic was repeated with 100000 iterations.

Experimentations on 100-node networks were additionally divided into three basic categories concerning features of access links in the systems: network with asymmetric links (derived from ADSL concept), symmetric (the same upload and download limits) and random (proportional number of symmetric and asymmetric links). Table IV presents an average percentage gap of heuristics solutions to results obtained by Random Strategy (bold values are best). The EA provided with the best results for scenarios with asymmetric (a) and random (r) links but for systems with symmetric access links (s) the best heuristic is STF. The general trend for E50A50 systems is shown in Fig. 5. Furthermore, it should be noted that the execution time of EA for 10-node networks was about 1 minute, RS, CSS, TCS and STF worked about 20-25 seconds, while the CPLEX solver in many cases needed more than 1 hour. In contrast, the execution time of EA for 100-node networks took about 1 hour while for other heuristics was approximately equal to 20 minutes (100 repetitions of algorithms).

VII. Concluding Remarks

Distribution of blocks in P2P network is a sophisticated problem, that is really difficult to model so it has resemblance to actual networks. However, model used in this work is realistic and is very useful in optimization process. We proposed a new evolutionary algorithm solving the P2P flow optimization problem and compared it to random based heuristics with cost selection strategies. Results of the EA are very close to optimal results for small instances. Either the algorithm or heuristics - comparing to branch-and-cut algorithm - need less computational time and can be applied for much larger networks. Experimentations show that, it is worthy implementing some mechanisms from evolutionary algorithm or cost strategies in P2P file sharing applications therefore transport costs might be reduced to about 50-80 percent than randomized flow assignment. Eventually, presented algorithms may be useful in planning distribution of i.e. updates to software using BitTorrent, which is most efficient way of exchanging data especially when number of participating nodes is enormous.

References

[1] V. Aggarwal, A. Feldmann, and C. Scheideler, "Can ISPS and P2P users cooperate for improved performance?" *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 3, pp. 29–40, July 2007.

[2] B. Akbari, H. R. Rabiee, and M. Ghanbari, "An optimal discrete rate allocation for overlay video multicasting," *Comput. Commun.*, vol. 31, no. 3, pp. 551–562, 2008.

[3] D. Arthur and R. Paningrahy, "Analyzing BitTorrent and Related Peer-to-Peer Networks," *In: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pp. 961–969, 2006.

[4] A. Bharambe, C. Herley, and V. Padmanabhan, "Analyzing and Improving BitTorrent Performance," *In: Proceedings of IEEE INFOCOM*, 2006.

[5] S.-S. Byun and C. Yoo, "Minimum dvs gateway deployment in dvs-based overlay streaming," *Comput. Commun.*, vol. 31, no. 3, pp. 537–550, 2008.

[6] Y. Chen, J. Edler, A. Goldberg, A. Gottlieb, S. Sobti, and P. Yianilos, "A Prototype Implementation of Archival Intermemory," *In: Proceedings of the Fourth ACM International Conference on Digital Libraries*, 1999.

[7] Cisco Systems, "Cisco visual networking index forecast and methodology, 20072012," White Paper, 2008.

[8] B. Cohen, "Incentives build robustness in bittorrent," Online, available at http://www.bittorrent.org/bittorrentecon.pdf, 2003.

[9] P. Druschel and A. Rowstron, "PAST: A Large-scale, Persistent Peer-to-Peer Storage utility," *In: Procedings of HOTOS*, pp. 75–80, 2001.

[10] P. Ganesan and M. Seshadri, "On cooperative content distribution and the price of barter," in *In proc. of the 25th IEEE International Conference on Distributed Computing Systems*, 2005.

[11] ILOG, Inc, "ILOG CPLEX 11.0: User's manual," France, 2007.

[12] A. Iosup, R. Iosup, P. Garbacki, J. Pouwelse, and D. Epema, "Correlating topology and path characteristics of overlay networks and the internet," in *In 6th Int. Workshop on Global and Peer-to-Peer Computing (GP2PC), held in conjunction with the IEEE/ACM CCGrid*, 2006.

[13] Ipoque, "Internet study 2007," Online, available at http://www.ipoque.com/resources/internet-studies/internet-study-2007, 2007.

[14] Ipoque, "Internet study 2008/2009," Online, available at http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009, 2008.

[15] C. Killian, M. Vrable, A. C. Snoeren, A. Vahdat, and J. Pasquale, "The overlay network content distribution problem," UCSD/CSE, Tech. Rep. CS2005-0824, 2005.

[16] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "Oceanstore: An Architecture for Global-scale Persistent Storage," *In: Proceedings of ACM ASPLOS, New York*, 2000.

[17] M. Kucharzak and K. Walkowiak, "File Sharing-based Heuristics for Flow Assignment in P2P Systems," in *2nd International Symposium on Logistics and Industrial Informatics LINDI 2009*, Linz, Austria, September 2009.

[18] ——, "Optimal flows in peer-to-peer based architectures for file sharing services," in *Proc. of the 16th Polish Teletraffic Symposium PTS 2009*, Lodz, Poland, September 2009.

[19] B. Leuf, *Peer to Peer: Collaboration and Sharing over the Internet*. Addison Wesley, 2002.

[20] L. Massoulie and M. Vojnovic, "Coupon replication systems," *SIGMETRICS Perform. Eval. Rev.*, vol. 33, no. 1, pp. 2–13, 2005.

[21] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs (3rd ed.)*. London, UK: Springer-Verlag, 1996.

[22] J. Mundinger and R. R. Weber, "Efficient file dissemination using peer-to-peer technology," Statistical Laboratory Research Reports, Tech. Rep. 2004–01, 2004.

[23] A. Oram, Ed., *Peer-to-Peer : Harnessing the Power of Disruptive Technologies*. O'Reilly, 2001.

[24] M. Pióro and D. Medhi, *Routing, Flow, and Capacity Design in Communication and Computer Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

[25] R. Steinmetz and K. Wehrle, *Peer-to-Peer Systems and Applications (Lecture Notes in Computer Science)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[26] K. Walkowiak, "Offline Approach to Modeling and Optimization of Flows in Peer-to-Peer Systems," in *Proc. NTMS 2008 - New Technologies, Mobility and Security*. Tangier, Morocco: IEEE Press, November 2008, pp. 1–5.

[27] ——, "On Transfer Costs in Peer-to-Peer Networks Systems: Modeling and Optimization," in *Proc. PGTS 2008 - 5th Polish-German Teletraffic Symposium*, Berlin, Germany, October 2008, pp. 217–226.

[28] K. Walkowiak and M. Kucharzak, "New Approaches to Modeling of Flows in Peer-to-Peer Systems," in *Proc. Modelling and Simulation of Systems MOSIS 2009*. Roznov pod Radhostem, Czech Republic: Ostrava: MARQ, April 2009.

[29] C. Wu and B. Li, "On Meeting P2P Streaming Bandwidth Demand with Limited Supplies," in *Proc. of the Fifteenth Annual SPIE/ACM International Conference on Multimedia Computing and Networking (MMCN 2008)*, 2008.

[30] C. Wu, C. Li, and J. Ho, "Improving the Download Time of BitTorrent-like Systems," *In: Proceedings of IEEE International Conference on Communications*, pp. 1125–1129, 2007.

[31] G. Wu and T. Chiueh, "Peer to peer file download and streaming. rpe report, tr-185," 2005.

[32] J. Wu, *Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless and Peer-to-Peer Networks*. Auerbach Publications, 2006.

[33] S. Yamazaki, H. Tode, and K. Murakami, "Cat: A cost-aware bittorrent," in *In Proc. of 32nd IEEE Conference on Local Computer Networks*, 2007, pp. 609–614.

[34] X. Yang and G. de Veciana, "Service capacity of peer to peer networks," in *In proc. of INFOCOM 2004*, 2004, pp. 2242–2252.

[35] L. Ying and A. Basu, "Traceroute-based fast peer selection without offline database," in *ISM '06: Proceedings of the Eighth IEEE International Symposium on Multimedia*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 609–614.

[36] Y. Zhu and B. Li, "Overlay networks with linear capacity constraints," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 2, pp. 159–173, 2008.