Noname manuscript No.
(will be inserted by the editor)

An integer linear formulation for the file transfer scheduling problem

Zorica Dražić · Aleksandar Savić · Vladimir Filipović

Received: date / Accepted: date

Abstract In this paper we consider a file transfer scheduling problem (FTSP). The problem is to minimize the overall time needed to transfer all files to their destinations for a given collection of various sized files in a computer network, i.e. to find the file transfer schedule with minimal length. Each computer in this network has a limited number of communication ports. The described problem is proved to be NP-hard in a general case. We propose a new integer linear programming (ILP) formulation of this problem. This formulation enables the problem to be solved by standard ILP solvers like CPLEX or Gurobi. In order to prove the validity of proposed ILP formulation two new reformulations of FTSP are presented. The results obtained by CPLEX and Gurobi solvers, based on this formulation, are presented and discussed. As it could be seen, in some cases the lower bound of the file transfer scheduling problem is not tight.

Keywords Integer linear programming · scheduling · file transfers · combinatorial optimization

Mathematics Subject Classification (2000) 90C11 · 90C35 · 68M10

This research was partially supported by Serbian Ministry of Education and Science under the grant no. 174010.

Z. Dražić · A. Savić · V. Filipović
Faculty of Mathematics, University of Belgrade, Studentski trg 16/IV, 11000 Belgrade, Serbia
E-mail: lolaz@sezampro.rs
A. Savić
E-mail: asavic@matf.bg.ac.rs
V. Filipović
E-mail: vladaf@matf.bg.ac.rs

1 Introduction

One of the basic problems in distributed computer systems is organization of transfers of large files between nodes. The problem which will be considered in this paper is minimization of overall transfer time for a given collection of files in a computer network. Optimizing file transfers is widely applicable to many areas such as Wide Area computer Networks (WAN), Local Area Networks (LAN), telecommunications, multiprocessor scheduling in a MIMD machines, task assignments in companies, etc.

File transfer scheduling problem (FTSP) is concerned with transferring a large collection of files between various nodes of a computer network. Each computer has a limited number of communication ports and a number of files that should be transferred between that computer and some other computer in a network. For each file that should be transferred between the pair of computers we know the amount of time needed to perform its transfer. The files are transferred without any forwarding, i.e. directly between their starting and ending computers, without involving any intermediary computers. It is assumed that once the transfer of a file begins it continues without any interruption until its completion. The objective of the problem is to schedule the file transfers, so the total time for the transfer process is minimal.

FTSP can be modeled using a weighted undirected multigraph $G = (V, E)$, which is called the file transfer graph. The vertices of the graph G correspond to the nodes of the network. Each vertex $v \in V$ is labeled with a positive integer $p(v)$ that represents the number of communication ports corresponding to v, i.e. the sum of uploads and downloads it can handle simultaneously. The edges of the graph G represent the files that should be transferred. Each edge $e \in E$ is labeled with a positive integer $L(e)$ that represents the amount of time needed for the transfer of the file e, expressed in some time units.

Let us formulate this problem using file transfer graph. This formulation will be denoted as P1.

Problem P1: The set of starting file transfer time moments forms the schedule for the problem. Given a file transfer graph G, a schedule can be formally represented as a function $s : E \to [0, \infty)$ that assigns a starting time $s(e)$ to each edge $e \in E$, such that for each vertex $v \in V$ and time $t \geq 0$

$$|\{e : v \text{ is an end point of } e \text{ and } s(e) \leq t \leq s(e) + L(e)\}| \leq p(v)\,. \tag{1}$$

The makespan length of a schedule s is defined as the largest finishing time, i.e., the maximum of $s(e) + L(e)$ over all edges $e \in E$. Formally, let us define the objective function of s, $Obj_{P1}(s) = \max_{e \in E}\{s(e) + L(e)\}$, i.e. the first time moment when all file transfers are complete.

The goal is to find a schedule s with the minimum possible makespan, denoted as $Opt_{P1}(G)$.

The obvious lower bound for the makespan, which shall be called the elementary lower bound, can easily be found. Let E_u denote the set of all edges $e \in E$ that have one end point at the vertex $u \in V$ and $\sum_u$ the sum of all amounts of times needed to transfer all the files that are to be sent or received by vertex u, i.e. $\sum_u = \sum_{e \in E_u} L(e)$. Let p_u denote the port constraint $p(u)$ for vertex u, i.e. the number of uploads and downloads it can handle simultaneously. It is easy to see that the time to transmit all the files involving vertex u is at least $\lceil \sum_u / p_u \rceil$. Thus, the schedule with the minimum possible makespan for any graph G must satisfy $Opt_{P1}(G) \geq \max_u \lceil \sum_u / p_u \rceil$.

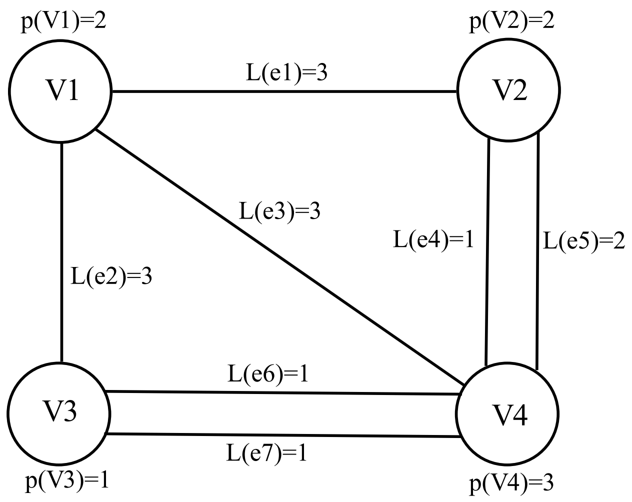

Fig. 1 A file transfer graph G1=(V,E)

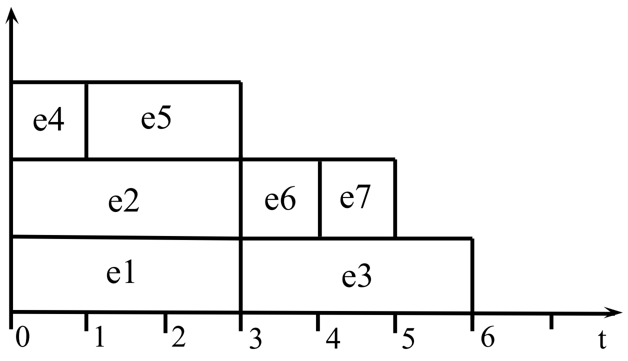

Fig. 2 An optimal schedule for graph G1=(V,E)

Example 1 Let us consider the graph $G1 = (V, E)$ shown in Figure 1. It's easy to see that:
$\sum_{V1} = L(e_1) + L(e_2) + L(e_3) = 3 + 3 + 3 = 9,$
$\sum_{V2} = L(e_1) + L(e_4) + L(e_5) = 3 + 1 + 2 = 6,$

$\sum_{V3} = L(e_2) + L(e_6) + L(e_7) = 3 + 1 + 1 = 5,$
$\sum_{V4} = L(e_3) + L(e_4) + L(e_5) + L(e_6) + L(e_7) = 3 + 1 + 2 + 1 + 1 = 8.$
From there, the value of elementary lower bound can be calculated as:
$\max\{\lceil\sum_{V1}/p_{V1}\rceil, \lceil\sum_{V2}/p_{V2}\rceil, \lceil\sum_{V3}/p_{V3}\rceil, \lceil\sum_{V4}/p_{V4}\rceil\} =$
$\max\{\lceil\frac{9}{2}\rceil, \lceil\frac{6}{2}\rceil, \lceil\frac{5}{1}\rceil, \lceil\frac{8}{3}\rceil\} = \max\{5, 3, 5, 3\} = 5.$
Following simple analysis shows that a makespan of 5 is not achievable.

- Consider the vertex $V1$ with three files that should be transferred;
- Since the port constraint of $V1$ is equal to two, it can transmit only two of its three files simultaneously;
- During this time the third transfer can not be in transmission. Since the transfer of each of files $e1$, $e2$, $e3$ requires 3 time units, the third transfer can start only when other two finish;
- Thus, it is easy to see that to finish all three transfers including vertex $V1$ at least 6 time units are needed;
- Therefore, the lower bound of 5 time units is not achievable.

One solution with schedule length equal to 6 is presented on Figure 2. Since it was previously proved that the optimal transfer of all files requires at least 6 time units, this solution is optimal.

2 Previous work

File transfer scheduling problem is introduced by E.G. Coffman et al. in [2]. The authors proved that it is NP-complete and proposed several algorithms in which they have tried to compute the minimum makespan in polynomial time. Their solutions were limited by some conditions such as the file transfer graph G is bipartite and all the files have equal lengths; the graph G possesses no simple cycle except in the trivial case of 2-vertex cycle included by multiple edges and same value lengths; the graph G is a multiple-edged graph with even cycle and equal file lengths; the graph G is a multiple-edged graph with odd cycle, equal lengths of files, and all port capacities equal to one.

In [9] authors extended this problem to directed file transfer graphs and presented a polynomial-time algorithm for the problem when all file transfers require an equal amount of time. Also, they studied the case when the file transfer time is arbitrary and proved that this problem is NP-complete.

In [1] authors presented approach based on a deterministic modified Hopfield model of neural network for solving this problem. Their model was mapped onto a 2-dimensional neural network architecture and the experiments were carried out on instances with small dimension (up to 39 vertices and 100 files that should be transferred). For bigger instances algorithm did not converge. For all the instances where the solution was found, the theoretical lower bound of objective function was reached which indicated that all test instances were too easy.

A variable neighborhood search (VNS) algorithm for solving FTS problem was designed in [4] . The experimental results were carried out on instances up to 100 vertices and 500 files that should be transferred. Optimality of VNS solutions on smaller size instances has been verified by total enumeration, and for several larger instances optimality followed from reaching the elementary lower bound of a problem.

The performance of a greedy on-line algorithms was studied in [7] for similar problem in which file transfer requests become known to an algorithm one at a time and the routing decision must be made for each request before any future requests are known. Authors discussed the performance of several greedy on-line algorithms and showed that they are not always the best available.

In [8] authors reconsidered time-index formulation of the FTSP in multi-server and multi-user environments. They reduced the complexity of the optimization by transforming it into an approximation problem.

The simple version of file transfer scheduling problem was reduced to the edge-coloring problem in [10]. Authors considered the FTS problem in which each file has the same length, formulated as f-coloring problem. In the cases of FTS problem where the capacity of the channels is considered, an f-coloring problem is generalized to an fg-coloring problem.

3 Problem Reformulations

In some cases it is convenient to formulate optimization problems as integer programming models [5], and to apply well-known techniques for solving them. Moreover, some recent approaches can be utilized [11]. In this Section we present two new reformulations of the problem $P1$, described in previous Section. These reformulations are more suitable for obtaining the integer linear programming (ILP) formulation in Section 4, the first such to the authors knowledge.

Since all file transfer lengths are integers it would be easier to work with integer time intervals instead of continues time intervals. Therefore, a new formulation $P2$ of original problem will be introduced in order to consider only integer time intervals. Additionally, intervals represented in this way will enable the usage of indicator decision variables. These decision variables will indicate if the file transfers are executed in corresponded time intervals. Since there is a finite number of file transfers it is enough to have finite number of time intervals for their execution. Formulation $P3$ deals with this fact and introduces an upper bound on number of time intervals. The formal definition are given in the sequel.

Definition 1 Let $\tau \in Z^+$ represent the index of the time interval from $\tau - 1$ to τ.

Lemma 1 *Let s' be some feasible solution of FTS problem. There exists the function $s'' : E \to Z^+$ such that $Obj_{P1}(s'') \leq Obj_{P1}(s')$.*

Proof Let $m(s') = \min\{s'(e)|e \in E\}$. If $m(s') > 0$ then $s^*(e) = s'(e) - m(s')$ is also feasible FTSP solution with better objective value. From this, one can see that s^* is solution where at least one transfer starts at $t = 0$.

Let's define $s''(e) = \lfloor s'(e) \rfloor$. It is obvious that $OBJ(s'') \leq OBJ(s')$. Let us now prove that s'' is a feasible solution.

Let $e_1, e_2 \in E$ and $s'(e_1) + L(e_1) \leq s'(e_2)$, i.e. there is no overlapping in transferring files represented by e_1 and e_2. In this case $s'(e_2) - s'(e_1) \geq L(e_1)$. We can consider three different cases:
Case 1: $s''(e_2) - s''(e_1) = \lfloor s'(e_2) - s'(e_1) \rfloor$.
Case 2: $s''(e_2) - s''(e_1) = \lfloor s'(e_2) - s'(e_1) \rfloor + 1$.
Case 3: $s''(e_2) - s''(e_1) = \lfloor s'(e_2) - s'(e_1) \rfloor - 1$.

In case 1, since $\lfloor s'(e_2) - s'(e_1) \rfloor \geq L(e_1)$, that implies $s''(e_2) - s''(e_1) = \lfloor s'(e_2) - s'(e_1) \rfloor \geq L(e_1)$. In case 2, $s''(e_2) - s''(e_1) = \lfloor s'(e_2) - s'(e_1) \rfloor + 1 \geq \lfloor L(e) \rfloor + 1 = L(e) + 1$. In case 3, we can write $1 = \lfloor s'(e_2) - s''(e_2) - (s'(e_1) - s''(e_1)) \rfloor$. Since $s'(e_2) - s''(e_2) \in [0, 1)$ and $s'(e_1) - s''(e_1) \in [0, 1)$ then $\lfloor s'(e_2) - s''(e_2) - (s'(e_1) - s''(e_1)) \rfloor < 1$ which is contradiction.

We proved that all pairs which are not overlapped in s' will stay such in s'', which means that s'' is also feasible. □

Now, we are in position to define active file transfer.

Definition 2 We shall say that in the schedule $s : E \to Z^+$ the $e \in E$ is active in time interval with index τ if $s(e) < \tau$ and $s(e) + L(e) > \tau - 1$.

Now let us define a problem P2.

Problem P2: For a schedule $s : E \to Z^+$, for all $\tau \in Z^+$ and $v \in V$
$|\{e : v \text{ is an end point of } e \text{ and } e \text{ is active in time interval with index } \tau\}| \leq p(v)$.

The makespan length of a schedule s, i.e. $Obj_{P2}(s)$, is defined as the largest τ from Definition 1. The goal is to find a schedule s with the minimum possible makespan, denoted as $Opt_{P2}(G)$.

Theorem 1 *For a given G, problems P1 and P2 have the same optimal objective function values.*

Proof (⇒) Let s^* be an optimal solution of problem P1. According to Lemma 1, there exists a feasible solution for problem P2 with integer starting moments where the solution is not greater than the initial solution, i.e. $Opt_{P2}(G) \leq Opt_{P1}(G)$.
(⇐) Other direction is trivial since $Z^+ \subset R$, so $Opt_{P2}(G) \geq Opt_{P1}(G)$.

□

Now let us define a problem P3.
Problem P3: For a schedule $s : E \to \{1, 2, ..., T_{\max}\}$, for all $\tau \in \{1, 2, ..., T_{\max}\}$ and $v \in V$
$|\{e : v \text{ is an end point of } e \text{ and } e \text{ is active in time interval with index } \tau\}| \leq p(v)$.

The makespan length of a schedule s, i.e. $Obj_{P3}(s)$, is defined as the largest τ from Definition 1. The goal is to find a schedule s with the minimum possible makespan, denoted as $Opt_{P3}(G)$.

Theorem 2 *For a given G and* $T_{\max} \geq \sum_{e_i \in E} L(e_i)$, *problems P2 and P3 have the same optimal objective function values.*

Proof ($\Rightarrow$) We can form a schedule where the first transfer e_1 starts at beginning, the second transfer e_2 at $L(e_1)$, the k-th transfer e_k starts at $\sum_{i=1}^{k-1} L(e_i)$. The last transfer will finish at $\sum_{e_i \in E} L(e_i)$. Since $T_{\max} \geq \sum_{e_i \in E} L(e_i)$ it follows $s(e) + L(e) \leq T_{\max}$ for all e. Due to the fact that $s(e)$ is an integer for problem P2 than $s(e) \in \{1, ..., T_{\max}\}$. Therefore, $Opt_{P2}(G) \geq Opt_{P3}(G)$.

($\Leftarrow$) Since $\{1, ..., T_{\max}\} \subset Z^+$ then $Opt_{P2}(G) \leq Opt_{P3}(G)$ □

4 New Integer Linear Programming Formulation

Since the theoretical considerations have been completed in the previous Section 3, all assumptions for a new ILP formulation are fulfilled.

Considering the problem P3, we can introduce the indicator variables

$$X_{e\tau} = \begin{cases} 1, \text{ if transfer } e \text{ is active in time interval with index } \tau \\ 0, \text{ otherwise} \end{cases} \tag{2}$$

Also, let $T_{\max}$ be one known upper bound for the optimal solution of P3. One such bounds is given by Theorem 2, but much tighter bound can be obtained by some methaheuristic, for example, variable neighborhood search - VNS [4]. Variable z represents the makespan length of a feasible schedule. Since we are considering discrete time intervals this variable will be integer. This enables us to introduce an ILP formulation of this problem.

Problem ILP:

$$\min z \tag{3}$$

subject to

$$\sum_{\tau=1}^{T_{\max}} X_{e\tau} = L(e), \qquad e \in E \tag{4}$$

$$\sum_{e \in E, e \ni v} X_{e\tau} \leq p(v), \qquad v \in V, 1 \leq \tau \leq T_{\max} \tag{5}$$

$$z \geq \tau \cdot X_{e\tau}, \qquad e \in E, 1 \leq \tau \leq T_{\max} \tag{6}$$

$$X_{ej} \geq X_{ei} + X_{ek} - 1, \qquad e \in E, 1 \leq i < j < k \leq T_{\max} \tag{7}$$

$$X_{e\tau} \in \{0, 1\}, \qquad e \in E, 1 \leq \tau \leq T_{\max} \tag{8}$$

$$z \in Z^+. \tag{9}$$

The constraint (4) ensure that the whole file (represented by the edge e) is sent and that its transfer is finished by the time $T_{\max}$. By (5) the maximum number of simultaneous transfers for each vertex is limited by its port constraint. The constraint (6) mean that there are no active transfers after time moment z. The constraint (7) provide that if the transfer is active in moments i and k it also has to be active for all moments j such that $i < j < k$, i.e. the transfers are carried out in continuous time intervals. With this constraint, when transfer stops it can not be continued for the same file in some future time moment. The constraint (8) represent the binary nature of variables $X_{e\tau}$. Constraint (9) ensures that a variable z is a positive integer. We shall prove that problems P3 and problem ILP have the same optimal values.

Theorem 3 *Let s^* be the optimal solution of problem P3. Then for the optimal solution $(z^*, X^*_{e\tau})$ of problem ILP, $z^* = Obj_{P3}(s^*)$ holds.*

Proof ($\Rightarrow$) Let s^* be the optimal solution of problem P3 and $Obj_{P3}(s^*) = A$. It is clear that $A \leq T_{\max}$ since $T_{\max}$ is the upper bound for some known feasible solution. For given optimal solution s^* time intervals when each edge e is active can be easily determined. This allow us to calculate each $X^*_{e\tau}$ and we set $z^* = A$. Now, we prove that $(z^*, X^*_{e\tau})$ is a feasible solution of problem ILP.

Since the $\sum_{\tau=1}^{T_{\max}} X_{e\tau}$ is equal to the number of active intervals of transfer e, the constraint (4) holds. For fixed $v \in V$ and $\tau \in \{1, 2, ..., T_{\max}\}$, the sum in (5) is equal to the number of active transfers (in time interval with index τ) beginning or ending in vertex v. Since this number is limited by $p(v)$, inequality (5) holds.

If $X_{e\tau} = 0$, (6) holds trivially. If $X_{e\tau} = 1$ then in time interval with index τ at least one transfer is active and consequently $Obj(s^*) = A = z^* \geq \tau$, so (6) holds also in this case.

The inequality (7) is the same as $X_{ei} + X_{ek} \leq 1 + X_{ej}$ for $i < j < k$. It is clear that this inequality holds if $X_{ei} = 0$ or $X_{ek} = 0$. In the opposite case, when $X_{ei} = X_{ek} = 1$, the inequality holds only for $X_{ej} = 1$. Since the transfer e is continuous, there are no time intervals $i < j < k$ such that the transfer is active at i and k but not at j. This proves the inequality (7) holds. From definition of $X_{e\tau}$ and z it is obvious that constraints (8) and (9) holds.

We proved that $(z^*, X^*_{e\tau})$ is a feasible solution of problem ILP, so the optimal value for problem ILP is not greater than $z^* = A = Obj(s^*)$.

($\Leftarrow$) Now, assume that $(z^*, X^*_{e\tau})$ is the optimal solution of problem ILP. Let us construct the solution s^* of problem P3. For each $e \in E$, take $s^*(e) = -1 + \min\{\tau | X_{e\tau} = 1\}$. For fixed e, the constraint (7) provides that the transfer e is active only in consecutive time intervals, so the transfer in uninterrupted. The length of transfer e is equal to $L(e)$ by (4). Since $\tau \leq T_{\max}$, each transfer is finished at last by time $T_{\max}$. From (5), at each time interval with index τ the number of transfers starting or ending in vertex v is limited by $p(v)$. From these consideration it can be deduced that s^* is a feasible solution of problem P3.

Let τ^* be the index of last time interval for which at least one $X_{e\tau^*} = 1$. Since $\tau^* \leq T_{\max}$ this value exists. From (6), $z^* \geq \tau^* \cdot X_{e\tau^*} = \tau^* \cdot 1 = \tau^*$. In the solution s^*, in the time interval with index τ^*, at least one transfer is active and in all consequent time intervals there are no active transfers, so $Obj(s^*) = \tau^* = z^*$. Since this is a feasible solution, the value of optimal solution of problem P3 is not greater than z^*. □

5 Experimental study

This section presents the experimental results which show the effectiveness of the the proposed problem ILP formulation. All the tests were run on a a single core of the Intel Core 2 Duo 2.67 GHz PC with 4 GB RAM under Windows XP operating system. We used the CPLEX [3] version 12.4 and Gurobi [6] 5.02 solvers for testing the problem ILP formulation on extended set of smaller size instances from [4]. Note that in [4] a metaheuristic approach was implemented which could handle larger instances but may not reach optimal solutions in all cases. Even more, in cases when metaheuristic approach reaches optimal solution it could not prove optimality. The computation effort of optimality verification process is usually a few orders of magnitude greater than process of reaching optimal solution. Therefore, in this paper only the instances from [4] are considered where CPLEX and Gurobi finished its work and verified the optimality of the solutions.

The instances used in this paper include different number of vertices of the graph ($|V| = 5, 10, 20, 30$) and number of edges up to 50 ($|E| = 10, 20, 30, 40, 50$). Construction of these instances starts with randomly generating integer port constraint for each vertex. These numbers are chosen from interval $[1, v_{\max}]$, where $v_{\max}$ is 8 or 15, depending on the number of vertices in graph. The length of each edge is random integer number from the interval $[1, e_{\max}]$, where $e_{\max}$ is 8 or 15. The graph adjacency matrix is randomly generated avoiding self-loops. Disconnected graphs are ignored in process. For each pair of $|V|$ and $|E|$ ten instances were generated by using different random seeds.

Tables 1 and 2 present the results of CPLEX and Gurobi solvers using proposed problem ILP formulation on FTSP instances with number of edges $|E| \leq 30$ and $|E| \geq 30$ respectively. In the columns labeled as "Inst." of Tables 1 and 2 the test instance name is given. It contains the information about the number of vertices, number of edges and the ordinal number of the instance of that type, respectively. For example, the instance ftsp_5_10_7 has 5 vertices, 10 edges and number 7 is ordinal number (between 0 and 9). In the columns labeled as "LB" is the elementary lower bound for that instance, calculated as described in Section 1. Columns marked as "UB" contain the upper bound, obtained by VNS from [4]. Columns "opt" contain the optimal solution value found by CPLEX and/or Gurobi on proposed problem ILP formulation. "CPLEX" and "Gurobi" columns contain running time needed to obtain and verify optimal solution by CPLEX and Gurobi solvers, respectively. There was a time limitation of 7.200 seconds for both solvers. If CPLEX or Gurobi did not produced an optimal result, the symbol "-" is written in the corresponding column.

As it can be seen from the Tables 1 and 2, at least one or both solvers were capable to obtain optimal solutions for most instances, 143 out of 160 instances. CPLEX reached 133 and Gurobi 131 out of 143 obtained optimal solutions. For the instances where both solvers reached and verified the optimal solution, CPLEX was faster than Gurobi in most cases: for 96 out of 121 instances it reached optimal solution faster than Gurobi.

Table 1 Experimental results for CPLEX and Gurobi on small size instances, $|E| \leq 30$.

Inst.	LB	UB	opt	CPLEX t (sec)	Gurobi t (sec)	Inst.	LB	UB	opt	CPLEX t (sec)	Gurobi t (sec)
ftsp_5_10_0	14	14	14	1.461	5.441	ftsp_10_10_0	12	15	15	0.054	0.044
ftsp_5_10_1	10	15	15	0.247	1.257	ftsp_10_10_1	12	12	12	0.065	0.030
ftsp_5_10_2	12	12	12	0.243	0.936	ftsp_10_10_2	5	14	14	0.082	0.076
ftsp_5_10_3	14	14	14	0.238	3.713	ftsp_10_10_3	19	19	19	0.645	10.764
ftsp_5_10_4	10	10	10	0.102	0.645	ftsp_10_10_4	17	17	17	0.194	0.264
ftsp_5_10_5	6	13	13	0.533	1.864	ftsp_10_10_5	10	15	15	0.066	0.045
ftsp_5_10_6	12	13	13	0.921	2.116	ftsp_10_10_6	3	15	15	0.063	0.090
ftsp_5_10_7	8	8	8	0.022	0.014	ftsp_10_10_7	11	14	14	0.046	0.045
ftsp_5_10_8	11	11	11	0.083	1.416	ftsp_10_10_8	11	14	14	0.049	0.036
ftsp_5_10_9	26	26	26	0.984	40.966	ftsp_10_10_9	2	14	14	0.077	0.033
ftsp_10_15_0	20	20	20	0.609	5.354	ftsp_15_15_0	7	15	15	0.112	0.140
ftsp_10_15_1	6	15	15	0.091	0.067	ftsp_15_15_1	7	22	22	0.690	1.416
ftsp_10_15_2	4	15	15	0.118	0.151	ftsp_15_15_2	10	15	15	0.115	0.134
ftsp_10_15_3	16	16	16	0.297	3.180	ftsp_15_15_3	14	17	17	0.307	2.001
ftsp_10_15_4	20	21	21	31.786	56.177	ftsp_15_15_4	16	44	44	7.969	314.362
ftsp_10_15_5	24	25	25	1.286	12.481	ftsp_15_15_5	6	15	15	0.087	0.068
ftsp_10_15_6	35	35	35	3.478	41.873	ftsp_15_15_6	20	20	20	0.480	0.620
ftsp_10_15_7	16	16	16	0.925	0.740	ftsp_15_15_7	20	20	20	0.629	6.404
ftsp_10_15_8	6	20	20	4.392	7.617	ftsp_15_15_8	31	31	31	2.245	26.019
ftsp_10_15_9	5	15	15	0.098	0.067	ftsp_15_15_9	12	16	16	0.145	0.322
ftsp_10_20_0	10	31	31	2.952	27.349	ftsp_15_20_0	15	15	15	0.305	0.692
ftsp_10_20_1	7	18	18	2.074	13.098	ftsp_15_20_1	27	33	33	4.019	52.382
ftsp_10_20_2	20	21	21	13.887	90.275	ftsp_15_20_2	11	15	15	0.137	0.188
ftsp_10_20_3	24	26	26	193.308	418.957	ftsp_15_20_3	17	17	17	2.617	4.654
ftsp_10_20_4	16	28	28	2.187	18.627	ftsp_15_20_4	4	16	16	0.933	3.410
ftsp_10_20_5	27	27	27	1.685	3.195	ftsp_15_20_5	21	21	21	8.340	11.456
ftsp_10_20_6	20	20	20	5.627	7.391	ftsp_15_20_6	8	15	15	0.146	0.098
ftsp_10_20_7	10	20	20	23.271	49.249	ftsp_15_20_7	14	15	15	1.527	6.621
ftsp_10_20_8	22	28	28	264.629	220.997	ftsp_15_20_8	14	15	15	0.104	0.157
ftsp_10_20_9	21	23	23	13.804	29.771	ftsp_15_20_9	30	30	30	76.704	118.245
ftsp_20_20_0	36	36	36	5.674	102.325	ftsp_10_30_0	17	43	43	15.185	440.760
ftsp_20_20_1	16	18	18	3.205	10.215	ftsp_10_30_1	18	21	21	92.831	256.741
ftsp_20_20_2	9	15	15	0.135	0.099	ftsp_10_30_2	28	28	28	3.839	145.970
ftsp_20_20_3	34	39	39	434.654	334.495	ftsp_10_30_3	13	81	-	-	-
ftsp_20_20_4	20	20	20	8.160	111.262	ftsp_10_30_4	48	48	48	22.930	1459.141
ftsp_20_20_5	7	15	15	0.097	0.079	ftsp_10_30_5	57	57	57	35.126	-
ftsp_20_20_6	24	24	24	1.387	7.664	ftsp_10_30_6	32	32	32	-	543.770
ftsp_20_20_7	8	15	15	0.137	0.084	ftsp_10_30_7	18	19	19	24.677	18.444
ftsp_20_20_8	15	15	15	0.093	0.082	ftsp_10_30_8	10	28	28	95.991	167.492
ftsp_20_20_9	12	15	15	0.130	0.097	ftsp_10_30_9	23	28	28	-	319.673

As it can be seen from Tables 1 and 2, in 94 out of 160 cases the optimal solution differs from the elementary lower bound. Let us denote by *gap* the relative error of elementary lower bound with respect to optimal solution. This *gap* can be large, for example on *ftsp*_10_10_9 it is 85.7%.

It is obvious that dimension of a graph directly influence the solvability and overall running times. Since FTSP is NP-hard the number of operations to obtain and verify optimal solution increase exponentially with problem dimension.

Table 2 Experimental results for CPLEX and Gurobi on small size instances, $|E| \geq 30$.

Inst.	LB	UB	opt	CPLEX	Gurobi	Inst.	LB	UB	opt	CPLEX	Gurobi
				t (sec)	t (sec)					t (sec)	t (sec)
ftsp_20_30_0	11	14	14	0.158	0.131	ftsp_10_40_0	36	36	36	-	3276.791
ftsp_20_30_1	18	38	38	154.538	324.670	ftsp_10_40_1	45	45	-	-	-
ftsp_20_30_2	31	31	31	5.189	94.557	ftsp_10_40_2	29	29	29	202.266	636.223
ftsp_20_30_3	49	49	49	21.705	1373.608	ftsp_10_40_3	16	83	83	31.655	-
ftsp_20_30_4	12	23	23	20.226	66.238	ftsp_10_40_4	13	21	21	-	334.108
ftsp_20_30_5	17	28	28	3.270	29.501	ftsp_10_40_5	30	30	30	-	874.824
ftsp_20_30_6	14	19	19	6.202	17.756	ftsp_10_40_6	12	16	16	7.577	31.812
ftsp_20_30_7	11	15	15	0.158	0.139	ftsp_10_40_7	17	17	17	9.710	31.639
ftsp_20_30_8	14	50	50	23.353	1321.919	ftsp_10_40_8	35	48	-	-	-
ftsp_20_30_9	22	22	22	1.552	11.659	ftsp_10_40_9	34	34	34	-	2031.908
ftsp_20_40_0	25	44	44	20.052	1056.1960	ftsp_30_40_0	33	33	33	8.625	241.868
ftsp_20_40_1	28	40	40	-	1793.316	ftsp_30_40_1	15	26	26	3.415	33.603
ftsp_20_40_2	36	36	36	-	2354.555	ftsp_30_40_2	19	30	30	5.712	43.360
ftsp_20_40_3	56	56	56	31.384	-	ftsp_30_40_3	61	61	-	-	-
ftsp_20_40_4	22	22	22	2.228	26.643	ftsp_30_40_4	19	49	-	-	-
ftsp_20_40_5	115	115	-	-	-	ftsp_30_40_5	17	20	20	18.791	30.364
ftsp_20_40_6	19	31	31	614.618	973.255	ftsp_30_40_6	41	41	41	15.391	590.495
ftsp_20_40_7	46	46	46	21.266	447.105	ftsp_30_40_7	11	15	15	0.192	0.183
ftsp_20_40_8	50	50	-	-	-	ftsp_30_40_8	14	15	15	2.656	11.453
ftsp_20_40_9	26	26	26	3.726	107.873	ftsp_30_40_9	9	25	25	3.095	29.720
ftsp_10_50_0	26	92	-	-	-	ftsp_20_50_0	26	44	44	25.082	-
ftsp_10_50_1	17	81	81	37.750	-	ftsp_20_50_1	25	51	51	26.102	-
ftsp_10_50_2	40	42	-	-	-	ftsp_20_50_2	50	50	-	-	-
ftsp_10_50_3	31	57	-	-	-	ftsp_20_50_3	24	24	24	3.409	8.308
ftsp_10_50_4	17	20	20	82.919	212.336	ftsp_20_50_4	55	55	55	36.920	-
ftsp_10_50_5	32	32	32	-	1215.615	ftsp_20_50_5	46	54	-	-	-
ftsp_10_50_6	36	36	-	-	-	ftsp_20_50_6	41	41	41	-	774.470
ftsp_10_50_7	85	85	-	-	-	ftsp_20_50_7	21	24	24	262.765	376.478
ftsp_10_50_8	85	85	-	-	-	ftsp_20_50_8	15	22	22	95.926	220.422
ftsp_10_50_9	44	44	-	-	-	ftsp_20_50_9	23	23	23	3.066	93.374
ftsp_30_50_0	27	50	50	25.800	-	ftsp_40_50_0	8	25	25	12.949	41.350
ftsp_30_50_1	31	31	31	475.173	345.994	ftsp_40_50_1	17	36	36	11.325	159.683
ftsp_30_50_2	23	30	30	7.667	90.255	ftsp_40_50_2	23	24	24	50.267	62.041
ftsp_30_50_3	18	23	23	46.966	96.624	ftsp_40_50_3	12	28	28	7.594	189.292
ftsp_30_50_4	60	60	60	14.651	-	ftsp_40_50_4	24	24	24	3.088	21.925
ftsp_30_50_5	12	20	20	1.651	3.135	ftsp_40_50_5	29	38	38	15.521	408.155
ftsp_30_50_6	10	23	23	9.307	24.032	ftsp_40_50_6	46	46	-	-	-
ftsp_30_50_7	49	49	49	27.805	-	ftsp_40_50_7	53	53	53	32.547	-
ftsp_30_50_8	23	40	40	16.186	520.226	ftsp_40_50_8	27	39	39	16.813	167.999
ftsp_30_50_9	41	52	52	29.597	-	ftsp_40_50_9	19	23	23	26.519	106.560

Regarding the problem complexity, from the problem definition, it seems that the number of edges is more important than the number of vertices. This observation can be confirmed on tested instances. As can it be seen if the number of vertices is fixed and number of edges is varied following effects can be shown. For example, for all the instances with 10 vertices and 10 edges CPLEX reached optimal solutions in less than one second while Gurobi needed up to 11 seconds. On the other hand, for the instances with 10 vertices and 20 edges both solvers needed up to 450 seconds which is one order of magnitude longer. With increase of number of edges for instances with 10 vertices and 30 edges both solvers could not find optimal solutions for some of the instances in 7200 seconds.

Otherwise, if the number of edges if fixed and the number of vertices varies running times for various examples are in the same order of magnitude. For example instances with 20 edges and 10, 15 and 20 vertices in worst case finished in 419, 118 and 435 seconds, respectively.

Simple analysis of experimental results shows that the graph's density is the most important factor for problem complexity. For example, considering the instances with 10 vertices and 50 edges (density equal 5.0) 3 out of 10 optimal solutions were found. Although the instances with 40 vertices and 50 edges (density equal 1.25) got the largest dimensions, 9 out of 10 optimal solutions were obtained.

6 Conclusions

In this paper we considered the file transfer scheduling problem. An integer linear programming formulation for solving this problem is introduced for the first time. Two new reformulations of the problem were presented in the process of proving new ILP formulation's validity. Also, in this work it is proved that these two new reformulations are equivalent to the original problem.

Experiments were carried out on instances from the literature. CPLEX and Gurobi solvers, based on proposed problem ILP formulation, obtained results on extended set of instances. Numerical results show that both solvers are able to produce optimal solutions on instances up to 40 vertices and 50 edges in reasonable time. Moreover, it can be seen that there are cases where elementary lower bound is not tight.

This work can be extended by investigating the application of some exact method using this problem ILP formulation. Other possible direction can be using this methodology for solving similar problems.

References

1. Akbari M.K., Nezhad M.H., Kalantari M., Neural Network Realization of File Transfer Scheduling, The CSI Journal of Computer Science and Engineering, 2(2&4), 19–29 (2004)
2. Coffman E.G., Garey M.R., Johnson D.S., Lapaugh A.S., Scheduling File Transfers, SIAM Journal of Computing, 14(3), 744–765 (1985)
3. CPLEX solver, IBM-ILOG company, http://www.ibm.com
4. Dražić Z., Variable Neighbohood Search for the File Transfer Scheduling Problem, Serdica Journal of Computing, 3(6), 333–348 (2012)
5. Gouveia L., Moura P., Enhancing discretized formulations: the knapsack reformulation and the star reformulation, Top, 20(1) 52–74 (2012)
6. Gurobi Optimization, Gurobi optimizer reference manual, http://www.gurobi.com.
7. Havill J.T., Mao W., Greedy on-line file transfer routing, Proc IASTED International Conf on Parallel and Distributed Systems, 225–230 (1997)
8. Higuero D., Tirado J.M., Isaila F., Carretero J., Enhancing file transfer scheduling and server utilization in data distribution infrastructures, In Modeling, Analysis & Simulation

of Computer and Telecommunication Systems (MASCOTS), 2012 IEEE 20th International Symposium, 431–438 (2012)
9. Mao W., Directed File Transfer Scheduling, ACM 31st Annual Southeast Conference, 199–203 (1993)
10. Nakano S., Zhou X., Nishizeki T., Edge-coloring algorithms, Computer Science Today, 172–183 (1995)
11. Sherali H.D., Smith J.C., Dynamic Lagrangian dual and reduced RLT constructs for solving 01 mixed-integer programs, Top, 20(1), 173–189 (2012)