X Congresso Brasileiro de Inteligência Computacional (CBIC'2011), 8 a 11 de Novembro de 2011, Fortaleza, Ceará

DIGITAL DATA NETWORKS DESIGN USING HYBRID METAHEURISTIC

Gilberto F. de Sousa Filho
Departamento de Ciências Exatas, Universidade Federal da Paraíba
gilberto@dce.ufpb.br

Lucidio dos Anjos F. Cabral
Departamento de Informática, Universidade Federal da Paraíba
lucidio@di.ufpb.br

Abstract – This work emphasises the contruction of two strategies for refinement phase used in the hybrid metaheuristic *GILS* for configuration of a service, called Dynavideo, applied in the video distribution. This problem can be formulated as Steiner Tree Star Problem (STSP). At first strategy is employed the Descent Method in its Local Phase, at second is used a hybridization of *GILS* with a restrict formulation of the mathematical model for the STSP. In following, we describe some computational experiments compared with the instances of literature. The results indicate that the new strategies discussed were able to improve some results better than previous solutions.
Keywords – Steiner Tree Star Problem, Metaheuristics, GRASP, ILS.

1 INTRODUCTION

Servers that deal with video distribution, such as digital TV broadcasting and transmission of live events over the Internet, usually work with abrupt changes in their demand. The number, type and location of service clients can vary greatly in a short period of time. For example, this occurs every time a program premium starts to be displayed.

Currently many systems are able to distribute video in digital networks. These systems are capable of transmitting video streams in various formats. However, once configured, if it occurs some variation in demand during the broadcast of a video, these distribution services are unable to automatically adjust your settings.

In order to circumvent the difficulty of the foregoing, was designed plataform Dynamic Video Distribution Service(Dynavideo) exposed by [1]. The main feature of DynaVideo is the ability to dynamically adjust the configuration for a given demand. The costs involved with the design of Dynavideo's reconfiguration are: activation cost of the servers; cost of connection between the servers; cost allocation of clients to servers. Therefore, the configuration of backbone of video distribution must meet all clients at a minimal cost.

The main feature of DynaVideo is the ability to dynamically adjust the configuration for a given demand. The adjustment is based on the concept of replication of mobile video servers described by [2]. These replicas can be created in real time and moved to locations on the network enabling us to serve a given demand.

The current work proposes the contruction of two strategies for refinement phase used in the hybrid metaheuristic called *GILS*, presented in [3], at first strategy is employed the Descent Method in its Local Phase, at second is used a hybridization of *GILS* with a restrict formulation of the mathematical model for the STSP. To check the efficiency of this used strategies, were set two parameters: computacional time and quality of the solution.

This article is organized into seven sections. In Section 2, we present a brief description of the system DynaVideo, the problem of dynamic reconfiguration of service delivery of video and its relationship with the Steiner Tree Problem. Section 3, we present a mathematical formulation for the Steiner Tree Star Problem. In Section 4, we describe the hybrid metaheuristic *GILS* found in the literature and the use of a Descent Method, section 5, present the hybridization of *GILS* with a reformulation of mathematical model for STSP. In Section 6, we present the computational results. Finally, section 7, some conclusions are made.

2 SYSTEM ARCHITECTURE DYNAVIDEO

The service DynaVideo as described by [4] is dynamically configured. This flexibility allows the service to automatically adjust for variations in demand, the idea is that the service continually try to find an optimal configuration to meet a given demand. In DynaVideo, demand is defined by the number and location of customers. Figure 1 shows the main components of DynaVideo through a component diagram.

The *DynaVideo Manager* (DM) controls the assignment. When a client requests a video stream, this module looks for a server with capacity to meet it. If found, the DM client associates with that server. Otherwise, a server containing a replica will be activated to meet the same.

The *Distribution Server* (DS) is the component that works in a distributed having an instance on each server and having direct access to video sources, which can be encoded in real time or video file servers. When the DS does not make direct access to video, he works as a reflector receiving the stream from a server and passed to clients associated with it or to another server.

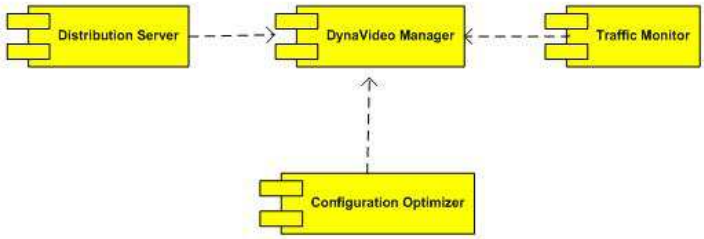

Figura 1: Major components of DynaVideo.

The arrival of a customer requires a change in service configuration. This event activates the *Traffic Monitor* (TM) described by [5], so this set servers to which the customer is closer, by tracing the routes of active servers to the client. Thus, the role of TM is to create and update a data structure, the routing graph, which stores the routes of active servers to clients.

The Component *Configuration Optimizer* (CO) was initially designed to work in a centralized manner, see [4], but with the parallelism of the optimization algorithm, this module now has an instance on each server system, and they are activated by DM for these to communicate and compute an optimal configuration for the service. CO is run in background, looking for a better setting for service delivery, considering the current demand represented by the graph of routes and the cost of processing and communication involved in reconfiguration. Being evaluated the new configuration, the CO requests to the DM that:

- Move a client from one server to another;
- Enable or disable a server;
- Move a server from one location to another;

2.1 OPTIMIZING THE SERVICE CONFIGURATION

In DynaVideo, demand is defined by the number and location of customers. The costs involved with the design of automatic reconfiguration are:

- Activation cost (or configuration) of the servers (DS);
- Cost of connection between the servers (DS);
- Cost allocation (or connection) of clients to servers (DS);

Connection costs and allocation can be assessed, respectivamante through a function that indicates quantitatively the quality of the connection between servers and the quality of the connection with the client being serviced by a server. This function can take as parameters, for example, the number of hops, which gives an idea of distance between client and server. The design of the backbone infrastructure must meet the requirements of customers, at minimal cost. Since the costs of activation can be evaluated using a function that would present the following parameters: the waiting time to enable the application server and the time to load configuration data from the server.

The problem of finding a minimum cost tree, connecting a set of nodes, possibly using auxiliary nodes, is classically known as Steiner Tree Problem (STP), described by Winter [6]. Thus, the problem of network design discussed above is a special case of the STP, called Steiner Tree-Star Problem (STSP), presented by [7] and [8], since each destination node is connected only one active node (server) in a star topology. This problem was addressed in [4] using the metaheuristic *GRASP* and [9] using genetic and transgenetic algorithms. In [3] were proposed three hybrid metaheuristic parallelization for the STSP.

3 MATHEMATICAL FORMULATION

The STS network consists of m client nodes that are interconnected through n Steiner nodes; each Steiner node j that is used to connect to at least one client node or other Steiner nodes is called an active Steiner node and will incur a fixed cost B_j; each client node i must be connected to exactly one active Steiner node, incurring a connection cost C_{ij}; two distinct Steiner nodes j and k that are directly connected incur a connection cost D_{jk}. The design problem is to find the minimum cost tree that spans all client nodes through selected active Steiner nodes. The STS problem can be formulated as an integer-programming model presented in [8].

The following notation is used in the formulation:

Indices:

i :index of client nodes; $i = 1, 2, \ldots, M$;
j, k : index of Steiner nodes; $j, k = 1, 2, \ldots, N$;

M : set of client nodes;
N : set of Steiner nodes.

Parameters:

C_{ij} : cost of connecting target node *i* to Steiner node *j*;
D_{jk} : cost of connecting Steiner nodes *j* and *k*;
B_j : cost of activating Steiner node *j*;
S : subset of N;
W : node of set S.

Decision variables:

X_{ij} := 1 if and only if client node *i* is linked to Steiner node *j*;
otherwise $X_{ij} = 0$;
Y_{jk} := 1 if and only if Steiner node *j* is linked to Steiner node *k*;
otherwise $Y_{jk} = 0$;
Z_j := 1 if and only if Steiner node *j* is selected to be active;
otherwise $Z_j = 0$.

The formulation is presented as follows:

$$Minimize \sum_{i \in M} \sum_{j \in N} C_{ij} X_{ij} + \sum_{j \in N} \sum_{k > j, k \in N} D_{jk} Y_{jk} + \sum_{j \in N} B_j Z_j \tag{1}$$

Subject to

$$\sum_{j \in N} X_{ij} = 1, i \in M, \tag{2}$$

$$X_{ij} \leq Z_j, i \in M, j \in N, \tag{3}$$

$$Y_{jk} \leq (Z_j + Z_k)/2, j < k, j, k \in N, \tag{4}$$

$$\sum_{j \in N} \sum_{k > j, k \in N} Y_{jk} = \sum_{j \in N} Z_j - 1, \tag{5}$$

$$\sum_{j \in S} \sum_{k > j, k \in S} Y_{jk} \leq \sum_{j \in \{S - w\}} Z_j, w \in S, S \subset N, |S| \geq 3, \tag{6}$$

$$X_{ij} \in \{0,1\}, Y_{jk} \in \{0,1\}, Z_j \in \{0,1\}, i \in M, j > k, j, k \in N. \tag{7}$$

In this formulation, the objective function (1) is to minimize the total cost, which includes the connection cost between a client node and a Steiner node, the connection cost between Steiner nodes, and the fixed cost to activate Steiner nodes. Constraint set (2) indicates that each client node is connected to exactly one Steiner node. Constraint set (3) indicates that the Steiner node must be activated before client node can be connected. Constraint set (4) indicates that two Steiner nodes must be activated before they can be connected to each other. Constraint set (5) expresses the constraints of a spanning tree that the total connections among Steiner nodes must be equal to the number of activated Steiner nodes less one. Constraint set (6) is an anti-cycle constraint that ensures that the backbone network solution is a spanning tree without any cycles. Constraints (7) indicate that the three decision variables are binary. In the formulation, constraint set (2) contains M constraints. Constraint set (3) generates $M \times N$ constraints. Constraint set (4) provides (N^2 - N)/2 constraints. Constraint sets (5) and (6) sum up to $N(2^{N-1}$ - $N)$ + 1 constraints. The total constraints are $N(2^{N-1}$ + M - (N - 1)/2) + M + 1. Eqs. (7) contain 1/2(N^2 + N) + $M * N + N$ decision variables. Although, a number of studies have attempted to solve the problem, the computational complexity makes it difficult to solve large size problems. Therefore, researchers have been designing efficient heuristics to solve large scale problem.

4 HYBRID METAHEURISTIC GILS

In this section, we describe the hybrid metaheuristic *GILS* that combines the metaheuristics *GRASP* [10] and *ILS* [11], aside from using a Descent Method which replaces the local search phase of *GRASP*.

4.1 REPRESENTATION OF THE PROBLEM

The main factor in solving the STSP is to identify the best servers that can meet customer demand. We assume that the set *N*, with $|N| = n$, represent the set of servers while the set *M*, with $|M| = m$, represent the set of customers.

The objective function that evaluates each solution is set according to costs: activation of the selected servers (actives) $j \in N$; connection of active servers *j,* $k \in N$ in a minimum spanning tree; assignment of customers $i \in M$ for its active servers $j \in N$. Apart from these costs, we must note that the capacity of each server must not be exceeded.

To calculate the cost of assigning two greedy algorithms were implemented: one that uses the ordering of the values of the matrix C_{ij} of costs from customers *i* to *j* servers in ascending order, called $GreedyOrd$, and another does not make the ranking of these values, called $GreedyNOrd$. Both algorithms try to assign each customer to the best active server, respecting the capacity of the chosen server.

To determine the value of the connections between servers, in each solution *S*, used the Kruskal algorithm to find a minimum spanning tree.

4.2 CONSTRUCTION *GRASP*

The construction phase, start with an empty solution *S*, i.e., we don't have the presence of active servers. At each iteration is chosen, randomly, a server from the restricted candidate list (*RCL*). The *RCL* list corresponds to one of the best inactive server. This list is obtained with the help of a greedy function, called $g(j)$, set for each inactive server *j*:

$$g(j) = gc(j) + gl(j) + gs(j), \tag{8}$$

where:

- $gc(j)$: represents the arithmetic average of the costs of the customers to server *j*;
- $gl(j)$: matches the best connection cost of server *j* with the other servers already active;
- $gs(j)$: corresponds to the cost of activating the server *j*.

The best candidates formed by inactive servers will be those that satisfy the condition:

$$g(j) \leq g_{min} + \alpha(g_{max} - g_{min}), \tag{9}$$

where $g_{min} = \min g(j)|jisinactive$, $g_{max} = \max g(j)|jisinactive$ and $\alpha \in (0, 1)$.

Finally, after obtaining *RLC* and choosing a random server $j \in RLC$, performed the process of adaptation, i.e., add the server *j* in the list of active solution *S*. The condition for stopping the construction phase was set so that the initial solution reach viability, i.e., the choice of inactives servers continues until the service capacity of active servers is greater than or equal to the total number of customers.

4.3 DESCENT METHOD

This work proposes the replacement of *GRASP*'s local search phase by a Descent Method, which uses three movements in following neighborhood used in literature [9]: ADD, $DROP$ and $SWAP$, iterating in a exhaustively way and moving forward with best improvement strategy, i.e., the current solution *S* is updated as you find a better solution.

These movements consist of:

- ADD: consider a server *j* that does not belong to the solution *S*. In this case, the server *j* will become active in solution *S*;
- $DROP$: consider a server *j* that belong to the solution *S*. In this case, the server *j* will become inactive in solution *S*;
- $SWAP$: corresponds to an *2-opt* exchange, i.e., server *j* that is active in the solution *S*, becomes inactive in *S* while the server *k* was inactive is replaced to active state.

These exchanges allow the formation of neighboring solutions to the current solution S.

4.4 *GILS* (*GRASP + ILS*)

The hybrid metaheuristic *GILS*, whose pseudocode was presented in [3] was modify with adition of Descent Method like is showed in Figure 2. It receives the following parameters: the evaluation function $f(.)$, constructor function greedy $g(.)$, neighborhood structure $N(.)$, number of movements for the disturbance $kmax$, and the percentage size of the list greedy α.

In *GILS* after *GRASP* construction phase and the Descent Method as intensification phase of the neighborhood of the solution built, will be applied the metaheuristic Iterated Local Search (*ILS*) as diversification phase. The objective of *ILS* is to improve upon stochastic Mutli-Restart Search by sampling in the broader neighborhood of candidate solutions and using a Local Search technique to refine solutions to their local optima.

procedure GILS(MaxIterGILS, $RandomSeed$, $f(.)$, $g(.)$, $N(.)$, $kmax$,α)
1. f(s^*) $\leftarrow \infty$;
2. *for i* $\leftarrow$ *1 to* MaxIterGILS do
3. s $\leftarrow$ ConstructGreedyRandomizedSolution($g(.)$,α);
4. $s_0 \leftarrow$ DescentMethod(s);
5. *if* $f(s_0) < f(s^*)$ then
6. $s^* \leftarrow s_0$;
7. *end-if*
8. $s_{ILS} \leftarrow$ ILS$(f, N, kmax, s_0, MaxIterILS)$;
9. *if* $f(s_{ILS}) < f(s^*)$ then
10. $s^* \leftarrow s_{ILS}$;
11. *end-if*
12. *end-for*
13. return(s^*).

end GILS.

Figura 2: Hybrid algorithm GILS (GRASP+ILS).

In each iteration of the loop of lines 2-12 is made *GRASP* iteration followed by an iteration *ILS*. Here neighborhoods ADD, $DROP$ and $SWAP$ are used. In the construction phase of *GRASP* just ADD neighborhood is used, while the local search phase all three neighborhoods are used exhaustively. For the *ILS* algorithm is passed as the initial solution the best solution obtained by local search and random moves are made $kmax$ type ADD or $DROP$, these movements after the local search procedure *GRASP* algorithm is triggered on the solution.

5 STRATEGY EXACT ASSIGNMENT (EA)

The mathematical model for the STSP, presented in section 2, need an exponential number of constraints to ensure no existence of cycle in the graph of active servers, so its execution is impractical for large instances, however, we noticed experimentally that the location of active servers and the generation of the minimum tree defined by the *GILS* metaheuristic obtained results, in average, close to those achieved by the exact model with runtimes computationally good for small instances, however, calculating the allocation of customers through greedy algorithms $GreedyOrd$ and $GreedyNOrd$ had poor results, due to its combinatorial nature, since each server has a limited capacity for serving clients.

We propose in this paper a new phase of refinement of the solutions generated by *GILS*, trying to get the best of both worlds, i.e., delegates the responsibility to creates the tree of active servers for metaheuristic *GILS* and responsibility to assignment customers to active servers for a restrict formulated model of STSP.

5.1 RESTRICT FORMULATED MODEL OF STSP

For the assignment of customers, the original model STSP had restrictions (4-6), responsible for creating the minimal tree, relaxed and the objective function of the model reduced. The set *M* of servers is changed by the set *Ma*, a subset of *M* formed by active servers defined by *GILS*.

The reformulation is presented as follows:

$$Minimize \sum_{i \in Ma} \sum_{j \in N} C_{ij} X_{ij} \tag{10}$$

Subject to

$$\sum_{j \in N} X_{ij} = 1, i \in Ma, \tag{11}$$

$$X_{ij} \leq Z_j, i \in Ma, j \in N, \tag{12}$$

$$Z_j = 1, i \in Ma, \tag{13}$$

$$X_{ij} \in \{0, 1\}, i \in Ma, j > k, j, k \in N. \tag{14}$$

As the metaheuristic locates the servers and sets their interconnection, there is a need to add the constraint set 13 that indicates for model which the servers are already considered to be active.

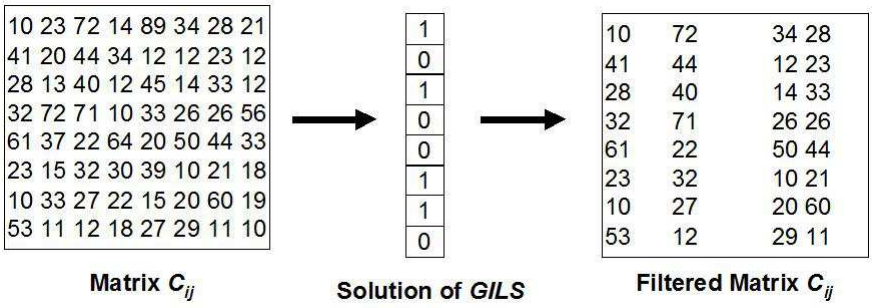

Figura 3: Filter matrix C_{ij}

5.2 FILTERING MATRIX C_{ij}

Because servers considered inactive staying out of the restrict formulated model, can decrease the size of the instance to this model, order to reduce its computational time, for this, after execution of *GILS* it will provide a boolean array of size *n*, indicating which servers should be active or inactive, the matrix C_{ij}, containing the cost of connecting customer *i* to server *j*, will be filtered, leaving only the columns corresponding to actives servers. The Figure 3 illustrates the filter C_{ij} guided by a solution of *GILS*.

This strategy that adds the execution of an exact model to solve the problem of assigning in a phase of refinement of solution generated by the metaheuristic *GILS* to the problem of the STSP, will be called *GILS+EA*.

6 COMPUTATIONAL RESULTS

The hybrid algorithm *GILS* proposed by [3] and its intensification strategies, *GILS+DM*, that change the local search phase of *GRASP* for a Descent Method, and *GILS+EA*, that make a hybridization of the algoritm GILS with a reformulation of the STSP model, described in the previous sections, were developed in C++ language and the aid of mathematical solver CPLEX [12]. All computational experiments were done on a machine consists of four Intel Core 2 Quad, each with the following specification: 4 processors 2.33 GHz with 2 GB of RAM and running the operating system Linux Ubuntu 9.04.

To investigate the performance of hybrid metaheuristic *GILS* and its intensification strategies were used 12 instances available in literature [4], with varying number of servers $N = \{10, 50, 100\}$ and varying number of customers in $M = \{10, 50, 100, 1000\}$. Each vertex (server or customer) had their coordinates in $\Re^2$, randomly generated with values between 0 and 100.

To demonstrate the efficiency in terms of solution quality of *GILS* metaheuristic and its intensification strategies, we made comparisons with an exact procedure B&B. For each instance of Table 1 we have the first two columns representing the dimensions of the instances tested, and the remaining columns divided into four groups: B&B procedure, *GILS*, *GILS+DM* and *GILS+EA*. In the procedure B&B, column z^* denotes the optimal value and the column indicates the long computing time in seconds spent solving the instance. For the groups of columns *GILS*, *GILS+DM* and *GILS+EA* additional time beyond the column, are: z that indicates the value obtained by algoritm and Δ (gap) that indicates the percentage difference between the B&B solution:

$$\Delta = [(z - z^*)/z^*] * 100 \tag{15}$$

It was established that the number of iterations of algorithm GILS and its refinement strategies was equal to 120, for each algorithm and each instance there were five plays. The value chosen for α was equal to 0.5, for which it gave the best performance during the testing of algorithm *GILS*, see [3].

Instance		**B&B (CPLEX)**		**GILS**			**GILS+DM**			**GILS+EA**		
n	**m**	z*	**Time**	*Z*	Δ (%)	**Time**	*Z*	Δ (%)	**Time**	*Z*	Δ (%)	**Time**
10	10	587.08	0.63	587.08	0.00	0.04	587.08	0.00	0.058	587.08	0.00	0.068
10	50	1598.94	0.61	1662.32	3.96	0.06	1662.31	3.96	0.112	1602.97	**0.25**	0.116
10	100	2521.11	0.08	2593.73	2.88	0.082	2593.73	2.88	0.156	2521.11	**0.00**	0.156
10	1000	17873.46	0.390	17963	0.50	0.484	17963	0.50	1.094	17878.91	**0.03**	1.180

Tabela 1: Comparison of B&B results with intensifications strategies of *GILS*

We can see in Table 1 that *GILS+EA* find better results than the metaheuristic *GILS*, approaching of the result found by B&B, obtaining a maximum gap of **0.25**% compared to this. The procedure B&B had a good computational time for instance with n=10, i.e., with ten servers. But for instances with n = 50, B&B was unable to reach an optimal solution in the limit of six hours of processing, due to the exponential number of constraints required. Results for the B&B procedure were obtained using the

Instance		GILS		GILS+DM			GILS+EA		
n	**m**	**Z**	**Time**	**Z**	**Δ%**	**Time**	**Z**	**Δ%**	**Time**
10	10	587.08	0.04	587.08	0.00	0.04	587.08	0.00	0.068
10	50	1662.32	0.06	1662.31	0.00	0.06	1602.98	**-3.57**	0.116
10	100	2593.73	0.082	2593.73	0.00	0.082	2521.11	**-2.8**	0.156
10	1000	17963	0.484	17963	0.00	0.484	17878.91	**-0.47**	1.18
50	10	547.69	11.3	531.12	**-3.03**	11.3	531.12	**-3.03**	11.2
50	50	2215.98	42.9	2134.82	**-3.66**	42.9	2081.54	**-6.07**	72.7
50	100	2788.46	53.3	2716.59	**-2.58**	53.3	2670.03	**-4.25**	93.4
50	1000	10536.92	185.0	10443.05	**-0.89**	185.0	10443.05	**-0.89**	503.9
100	10	561.35	273.3	529.79	**-5.62**	273.3	529.79	**-5.62**	265.2
100	50	2134.28	586.1	1880.36	**-11.9**	586.1	1871.13	**-12.33**	626.3
100	100	3965.42	1393.5	3721.94	**-6.14**	1393.5	3676	**-7.3**	2328.2
100	1000	10145.93	3259.7	9888.27	**-2.54**	3259.7	9737.54	**-4.03**	6967.8

Tabela 2: Comparison of *GILS* with its hybridization strategies

software CPLEX, under academic license. For the process B&B were not included limits previously calculated, so that the tool use only the linear relaxation.

Table 2 presents a comparison between the avarage solutions, in five play, found by the metaheuristic *GILS* and the refinement strategies of the same. We can see this table that the strategy *GILS+DM* improves the outcome of the *GILS* in 8 of 12 instance tested, obtaining an average negative gap of **-2,56**%, i.e., improving the previous solution founded. The strategy *GILS+EA* improves the best result known in 11 of the 12 instances tested, obtaining an average negative gap of **-3,8**%. With regard to computacional time, was noticed a significant increase in the *GILS+DM* compared to *GILS*, due to exhaustively neighborhood. Comparing the times of *GILS+DM* and *GILS+EA*, was not noticed much variation.

7 CONCLUDING REMARKS

The use of the service reconfiguration of network servers, video distribution, as in the case of DynaVideo, reduces the impact of increased demand on the network, since this reconfiguration reduces the total traffic volume due to customer service. The replication tends to approach a video for a given client area, where it is having a high demand by placing a copy of the video on a server closer to these customers. Whenever we find a better solution to the problem addressed we are improving the capabilities of the distribution system, so finding better solutions of STSP is the main objective of this work.

The strategy of refinement *GILS+DM* achieved improvements in several solutions of metaheuristic *GILS*, reaching around 11.9% improvement, this proves that the refinement of Descent Method before the procedure *ILS* served for the algorithm to converge to better solutions, a side effect of this strategy was the growth of computational time, due to its exhaustively neighborhood.

The strategy *GILS+ED* used a hybridization of *GILS* with a restrict formulation of the mathematical model for the STSP, in an attempt to improve the allocation of customers, improved all the solutions of *GILS* that did not reach the optimal solution. The *GILS+EA* enough to get a gap of 12.33% compared to the *GILS*, in addition to approaching the results of the exact procedure B&B, demonstrating that the strategy of handing responsibility the location of active servers and building the distribution tree to metaheuristic *GILS* and the responsibility of customers's allocation to the reformulated model, obtained good results, without generating impact on the computational time.

REFERENCES

[1] L. L. E. C. de Souza Filho G. and B. T. "DynaVideo - A Dynamic Video Distribution Service". In *6th Eurographics Workshop on Multimedia*, pp. 95–106, 2001.

[2] K. F. et al. "Dynamic Reconfiguration of Scalable Internet Systems with Mobile Agents". Technical Report TD-5WYSEW, Department of Computer Science at the University of Illinois at Urbana-Champaign, 1999.

[3] S. G. C. L. M. E. and L. G. "Metaheurística Híbrida Paralela para a Configuração de um Serviço de Distribuição de Vídeo". In *XIII CLAIO*, Montevideo, Uruguai, September 2006.

[4] S. G. C. L. M. E. and L. G. "Uma metaheuristica GRASP para configuração de um serviço de distribuição de vídeo baseado em replicação móvel". In *SBPO*, Gramado-RS, Brazil, September 2005.

[5] M. M. B. T. and L. G. "SMTA: Um Sistema para Monitoramento de Tráfego em Aplicações Multimídia". In *CLEI'2000*, Cidade do México, México, September 2000.

[6] W. P. and Z. M. "Euclidian Steiner Minimum Trees: An Improved Exact Algorithm". *Networks*, vol. 30, no. 3, pp. 149–166, 1997.

[7] L. Y. C. S. and R. J. "A branch and cut algorithm for the Steiner tree-star problem". *Informs Journal on Computing*, vol. 8, no. 3, pp. 100–120, 1996.

[8] X. J. C. S. and G. F. "Using Tabu Search to solve the Steiner tree-star problem in telecommunication network design". *Telecommunications systems*, vol. 6, pp. 117–125, 1996.

[9] L. L. E. C. de Souza Filho G.; Goldbarg M. C. and G. E. F. G. "Comparando algoritmos genéticos e transgenéticos para otimizar a configuração de um serviço de distribuição de vídeo baseado em replicação móvel". In *XXII Simpósio Brasileiro de Redes de Computadores*, volume 1, pp. 129–132, 2004.

[10] P. Festa and M. Resende. "GRASP: an annotated bibliography", Essays and Surveys on Metaheuristics". In *Essays and Surveys on Metaheuristics*, edited by C. Ribeiro and P. Hansen, pp. 325–367. Kluwer Academic Publishers,, 2002.

[11] L. H. R. M. O. and S. T. "Iterated local search". In *Handbook of Metaheuristics*, volume 57 of *Series in Operations Research & Management Science*, pp. 321–353. Kluwer Academic Publishers,, 2002.

[12] CPLEX. "ILOG CPLEX 11.2 User's Manual and Reference Manual", 2009.