Genetic Local Search for the Servers Load Balancing Problem

Yuri Kochetov[1,2], Artem Panin[1,2], Alexander Plyasunov[1,2]

[1] Novosibirsk State University,
2 Pirogov St., 630090 Novosibirsk, Russia
[2] Sobolev Institute of Mathematics,
4 Acad. Koptyug avenue, 630090 Novosibirsk, Russia
jkochet@math.nsc.ru, aapanin1988@gmail.com

Abstract. In this work we consider the servers load balancing problem (SLBP) formulated as a mixed integer linear programming problem. As known, this problem is NP-hard in strong sense. We analyze the approximability of the problem and prove that SLBP is Log-APX-hard under PTAS-reducibility and cannot be NPO-complete unless P=NP. To solve the problem we develop an approximate method using the idea of genetic local search. Computational results are discussed.

Keywords: servers load balancing, mixed integer linear programming, approximability, genetic local search.

1 Introduction

We consider the following servers load balancing problem [1, 2]. Given a set of servers and set of disks (more precisely, disk images). Internet sites with heterogeneous data are stored on those disks. Users visit the sites and thus cause the load on the servers. This load varies in time and is characterized by several parameters (e.g., load of the processor, required memory, etc). The users activity for each disk over the planning period is known. This activity allows to determine the load of the servers at each moment within each of the parameters. The time is assumed to be discrete, i.e. the load is changing, for example, each minute or second. If the load by each of the parameters does not exceed the given threshold then the server is running in its normal working mode. Otherwise the server has an overload. In order to avoid overload, some disks can be moved from one server to another. This transfer requires certain computational costs. We call them *additional costs*. We assume that for each disk the additional costs for moving it from one server to another are known for each parameter. The initial distribution of the disks by servers is given. The goal is to reallocate disks to the servers in order to achieve the minimum total overload during the whole planning period subject to the additional cost constraints for each server.

It was proven in [1] that servers load balancing problem is NP-hard in strong sense. An approximation algorithm with a posteriori performance guarantee for solving this problem based on the problem representation as mixed integer linear programming problem was also proposed in this paper. Integer variables were substituted with continuous ones and the optimal solution of the corresponding linear programming problem was used to obtain an approximate solution and its error estimations. More precisely, a linear programming allows to fix a part of disks on servers and thus to reduce the dimension of the problem and to solve the obtained subproblem exactly with the branch-and-bound method (using CPLEX solver). This approach allows us to find quickly an optimal solution on the instances with zero overload and gives rather good results in case of big overload of servers. For small overload a linear programming approach gives almost no positive integer components; thus, the dimension of the problem is not reduced, and hence the algorithm does not work.

Another approach based on the local search principles was proposed in [2]. Together with the essential neighborhoods of small size (switch disk to another server or swap disk with another disk) authors considered a new original neighborhood of exponential size. One disk was chosen on each server. After that, selected disks were redistributed between the servers. Each server got one disk, while the total overload of servers was minimized. The solution of the assignment problem gives the best redistribution of disks. Authors studied different ways of choosing disks for redistribution and efficiency of the local search with such neighborhood.

In the second chapter we formulate the servers load balancing problem as a mixed integer linear programming problem. We prove some results on approximation guarantee in the third chapter. We propose a new approximate method which is based on the idea of genetic local search in the fourth chapter . Computational results are discussed in the fifth chapter.

2 Mathematical model

We use the following notation: S is the set of servers, D is the set of disks; T is the planning period; R is the set of load parameters (CPU, RAM, ...); c_{drt} is the load of the disk d at the moment t by the parameter r; $\bar{c}_{sr}$ is the threshold load of the server s by the parameter r; x^0_{ds} is the initial distribution of disks among servers; $b^w_{sdr}(b^e_{sdr})$ is the additional costs to move the disk d to (from) the server s for the parameter r, and $B^w_{sr}(B^e_{sr})$ is the maximum allowed additional costs for the parameter r to move disks to (from) the server s.

Variables of the problem are: $x_{ds} = 1$ if the disk d is moved to the server s, $x_{ds} = 0$ otherwise; y_{str} is the overload at the server s at the moment t by the parameter r.

In these terms we can write the servers load balancing problem (SLBP) as a mixed integer linear programming problem [1, 2]:

$$\min \sum_{s\in S} \sum_{t\in T} \sum_{r\in R} y_{str}$$

subject to

$$y_{str} \geq \sum_{d\in D} c_{drt} x_{ds} - \bar{c}_{sr}, \qquad s \in S, \quad t \in T, \quad r \in R,$$

$$\sum_{s \in S} x_{ds} = 1, \qquad d \in D,$$

$$\sum_{d \in D} b^w_{sdr} x_{ds} \big(1 - x^0_{ds}\big) \le B^w_{sr}, \qquad s \in S, \quad r \in R,$$

$$\sum_{d \in D} b^e_{sdr} x^0_{ds} (1 - x_{ds}) \le B^e_{sr}, \qquad s \in S, \quad r \in R,$$

$$y_{str} \ge 0, \qquad x_{ds} \in \{0, 1\}, \qquad d \in D, \quad s \in S, \quad t \in T, \quad r \in R.$$

The objective function defines the total overload of servers during the planning period. The first set of inequalities defines the overload for each server at each moment of time for each parameter. The second set of equalities ensures that each disk will be put exactly at one server. The third and fourth sets of inequalities bound the additional cost for each server and each parameter when ejecting and inserting disks.

Formulating SLBP as an integer linear programming gives us a possibility to solve it using commercial software such as IBM ILOG CPLEX, GUROBI, AMPL, etc. Unfortunately, because of huge integrality gap, transition to continuous variable x_{ds} does not give us a possibility to find an exact solution even for instances of average size. For example, for

$$|S| = 20, \qquad |D| = 200, \qquad |T| = 150, \qquad |R| = 2,$$

calculations can take more than one day without any guarantee to find an exact solution. It can be partly explained by the fact that the problem is NP-hard in strong sense even for $|T| = |R| = 1$ and for zero additional costs [1, 2].

3 Approximation complexity

In this chapter we describe the place of SLBP in approximation hierarchy [3]. The first level of approximation hierarchy has the following structure:

$$PO \subseteq FPTAS \subseteq PTAS \subseteq APX \subseteq Log - APX \subseteq$$

$$\subseteq Poly - APX \subseteq Exp - APX \subseteq NPO.$$

Each class consists of optimization problems from the class NPO and describes certain quality of approximation, i.e. the first level is used to describe properties of the problems for which the corresponding decision version belongs to the class NP. The class PO consists of the problems for which there exists an exact polynomial algorithm. The class FPTAS consists of the problems for which there exists fully polynomial approximate scheme and the class PTAS is formed by the problems for which there exists polynomial approximate scheme. Classes APX, Log-APX, Poly-APX and Exp-APX consist of the problems for which there exists polynomial r-approximate algorithm, where r is a constant, logarithmic, polynomial, and exponential estimates for the accuracy of error, respectively. In the last three cases values of the above-mentioned functions depend on the length of input data (instance of the problem) [3, 4]. As known, inclusions of these classes are proper unless P=NP [3–5].

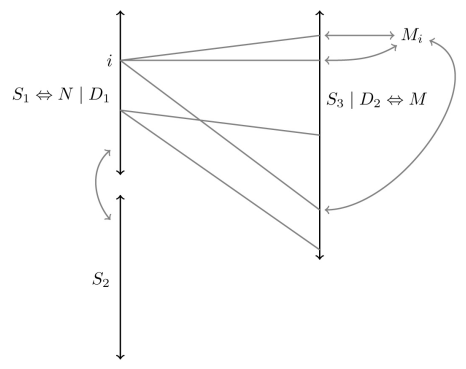
M_i
i
$S_1 \Leftrightarrow N \mid D_1$
$S_3 \mid D_2 \Leftrightarrow M$
S_2

Fig. 1. The constructed instance of SLBP.

we introduce notation for an arbitrary optimization problem A with minimization criterion: $L(A)$ is the set of inputs (refer to an arbitrary input $t \in L(A)$ as a problem t); $OPT_A(t)$ is the optimal value of $t \in L(A)$. $D_A(t)$ is the set of feasible solutions of $t \in L(A)$; $F_A(t,s)$ is the value of the objective function of $t \in L(A)$ on the solution $s \in D_A(t)$. The error of the solution $s \in D_A(t)$ of $t \in L(A)$ is defined as

$$R_A(t,s) = \max\{\frac{F_A(t,s)}{OPT_A(t)}, \frac{OPT_A(t)}{F_A(t,s)}\} \leq 1.$$

Clearly, if A is the optimization problem with minimization criterion then

$$R_A(t,s) = \frac{F_A(t,s)}{OPT_A(t)}.$$

Here, we deal with rational numbers only. We remind the definition of PTAS-reducibility [6]. Let A and B be problems from the class NPO. We say that the problem A PTAS-reduces to the problem B if and only if there exist three functions φ, ρ and γ such that

- $\varphi(t,\varepsilon) \in L(B)$ for any $t \in L(A)$, and any ε from the interval $(0,1)$; φ is polynomially computable with respect to the length $|t|$ of the input t;
- $\rho(t,s,\varepsilon) \in D_A(t)$ for any $t \in L(A)$, any $\varepsilon \in (0,1)$, and any $s \in D_B(\varphi(t,\varepsilon))$; ρ is polynomially computable with respect to $|t|$ and the length $|s|$ of the solution s;
- $\gamma : (0,1) \to (0,1)$; if $R_B(\varphi(t,\varepsilon),s) \leq 1/\varepsilon$ then $R_A(t,\rho(t,s,\,\varepsilon)) \leq 1/\gamma(\varepsilon)$; $t \in L(A)$, $\varepsilon \in (0,1)$, $s \in D_B(\varphi(t,\varepsilon)))$.

The following statement holds:

Theorem 1. *The problem SLBP is Log-APX-hard under PTAS-reducibility.*

Proof. We consider the minimum set cover problem [7] that can be described as follows. Given a set M and its subsets $M_1,...,M_n$ such that $\bigcup_{i=1}^{n} M_i = M$, a set $\tilde{N} \subseteq N =$

$\{1, ..., n\}$ is the *cover* of M if and only if $\bigcup_{i \in \tilde{N}} M_i = M$. The *weight* of each set M_i equals 1. The weight of the cover N is the sum of weights of all $M_i, i \in N$. The objective is to find the cover with minimal weight. As known, the minimum set cover problem is Log-APX-complete under PTAS-reducibility [8].

We reduce the minimum set cover problem (MSCP) to SLBP. To define the function φ we consider the following input of SLBP. Let $S = S_1 \cup S_2 \cup S_3$, where $|S_1| = |S_2| = |N|$ (for ease of description, let $S_1 = N$) and $|S_3| = |M|$, $D = D_1 \cup D_2$, where $|D_1| = |N|$ and $|D_2| = |M|$ (for ease of description, let $D_2 = M$). The planning period T consists of a single moment, i.e. $|T| = 1$. The set R consists of $|M|$ load parameters. Each disk from D_1 locates in a server from S_1 (one disk in each server). By analogy, each disk from D_2 are in a server from S_3. The load of each disk $d \in D_1$ by each load parameter r equals 1. We associate each disk $d \in D_2$ with a certain load parameter r. Then, the load of disk $d \in D_2$ by the association load parameter r equals 1. For leftover load parameters the load is 0. The threshold load of any server $s \in S_1$ by any load parameter equals 1. The threshold load of any server from S_2 by the first load parameter equals $1 - \delta$ for some small positive δ. For another load parameters the threshold load equals 1. The threshold load of the servers S_3 by each load parameter is 0. Let the additional costs to move a disk from a server be 0. We associate each disk $d \in D_1$ with two certain servers from S_1 and S_2, for example, the first disk is associated with the first server from S_1 and the first server from S_2, the second disk is associated with the second server from S_1 and the second server from S_2, i.e. the additional costs to move a disk $d \in D_1$ to the associated server $s \in S_2$ equal 0 and to another server equal 1. The additional costs to move the disk $d \in M_s$ to the server $s \in S_1$ are equal to 0 and to another server are equal to 1. All maximum allowed additional costs are equal to 0. Scheme of the reduction is shown in figure 1.

To solve the constructed input we need to move disks from D_2 to servers from S_1. Then, we have to move disks from D_1 that are in servers whose number of disks is more than 1. We move it to the servers from S_2 (one disk to its distinct server). Thus, the optimal objective value is equal to $|N^*|\delta$, where N^* is an optimal cover.

Let (x, y) be the feasible solution of the considered input. To define ρ construct a feasible solution $\rho(x, y, \varepsilon)$. First, using (x, y), construct the following solution $(\tilde{x}, \tilde{y})$. In x if the disk $d \in M_s$ from some $s \in N$ is located in a server from S_3 then we move d to the server s. We consider consequently all disks $\tilde{d} \in D_1$, which is located in a server from S_1. If there is another disk in this server then we move $\tilde{d}$ to the appropriate server from S_2. Derived distribution is $\tilde{x}$. Thus, the objective function value does not grow up, i.e. $\tilde{y} \leq y$. Moreover, all disks in S_2 associate with some cover $\tilde{N}$. Finally, we have:

$$R_{MSC}(t, \rho(x, y, \varepsilon)) = \frac{|\tilde{N}|}{|N^*|} = \frac{|\tilde{N}|\delta}{|N^*|\delta} = \frac{\tilde{y}}{|N^*|\delta} \leq$$

$$\leq \frac{y}{|N^*|\delta} = R_{SLB}(\varphi(t, \varepsilon), x, y) \leq \frac{1}{\varepsilon}.$$

Thus, we could take the function γ as the identity function. □

In previous theorem we obtained a lower bound on the location of SLBP in the approximation hierarchy. In addition, we derive an upper bound. Let us consider the

	LP		ILP		MLS	
$\overline{c_{sr}}$	time	value	time	value	time	value
850	2	0	8	0	1	0
840	2	0	21	0	1	0
830	2	0	11	0	1	0
820	2	0	3000	4	2	0
810	2	0	2100	0	2	0
800	2	0	3000	1370	3	0
750	200	0	—	—	1200	8424
700	734	87700	—	—	1200	120862
650	182	382399	3000	469294	1200	382560
600	62	682399	3000	761107	1	682399

Table 1. Computational result for the case $B^w_{sr} = B^e_{sr} = 150$.

maximum weighted satisfiability problem (MWSP) described in [3]. In this problem there is the boolean formula φ in variables x_1, x_2, ..., x_n with nonnegative weights w_1, w_2, ..., w_n. Solution is the truth assignment τ to the variables that satisfies φ. The measure is $\max\{1, \sum_{i=1}^n w_i\tau(x_i)\}$, where Boolean values "true" and "false" are identified with 1 and 0, respectively. As known, MWSP is NPO-complete under AP-reducibility[1] [3]. If SLBP is NPO-complete under AP-reducibility, then MWSP AP-reduces to SLBP. Then we can obtain a feasible solution of MWSP in polynomial time using, for example, the initial distribution of SLBP. Thus, we can solve NP-complete satisfiability problem in polynomial time. Therefore, SLBP cannot be NPO-complete under AP-reducibility unless P=NP.

High complexity of the problem gives rise to development of new approximation algorithms. One of the priorities in the area is development of genetic local search methods that are known to perform well while solving many NP-hard discrete optimization problems [9, 10].

4 Genetic local search

To solve SLBP we developed a hybrid algorithm based on the genetic algorithm and local search [9,10]. First, we determine all parameters of this hybrid algorithm (genetic local search or GLS). Let I_{max} be the iteration number of the algorithm. P is the size of population (problem's solutions). C_{max} is the maximal number of possible crosses of parents. The population is the set of feasible solutions of SLBP. The parent is the population's element. Then, the genetic local search scheme can be described as follows:

– *Step 0*: Create the initial population consisting of P feasible solutions. Let $i := 0$;
– *Step 1*: Take an arbitrary manner two parents from the population and cross them as follows: each disk places randomly with probability 0.5 to the first parent server

[1] AP-reducibility is a special case of PTAS-reducibility.

	LS(move)		LS(LK)		LS(rand)		GLS(move)		GLS(LK)	
$\overline{c_{sr}}$	time	value	time	value	time	value	time	value	time	value
850	59	0	65	0	<1	0	<1	0	<1	0
840	61	0	65	0	<1	0	<1	0	<1	0
830	61	0	64	0	1	0	1	0	1	0
820	61	0	65	0	1	0	1	0	1	0
810	74	0	76	0	2	0	2	0	2	0
800	88	0	90	0	20	0	20	0	20	0
750	237	8667	833	9199	284	8687	336	8362	1845	8427
700	285	121062	854	120121	143	120685	199	119840	1349	120073
650	174	382847	820	382807	73	382802	422	382673	774	382670
600	62	682399	522	682399	<1	682399	<1	682399	<1	682399

Table 2. Computational results for the case $B^w_{sr} = B^e_{sr} = 150$ (extention).

or to the second parent's server. If the resulting solution (descendant) is infeasible, because restrictions on the ejection and insertion disks were failed, then repeat crossing process. If the feasible solution is not found after C_{max} crosses, then $i := i + 1$ and if $i < I_{max}$, then repeat *step 1*, else *STOP*;

– *Step 2*: Apply the local search described below to the resulting solution from the previous step. Put $i := i + 1$. If i is not more than given iteration number I_{max}, then go to *step 1*. Otherwise *STOP*.

Apply the local search algorithm with randomized neighbourhood (neighbourhood *rand* described below) P times to the initial distribution x^0 to derive the initial population. Let $\tilde{x}$ be some feasible distribution of the problem. We use f to denote the objective function of SLBP. We describe the local search (LS) as follows:

– *Step 0*: $x^* := \tilde{x}$ and $f^* := f(\tilde{x})$;
– *Step 1*: Find the best neighbour x in the neighbourhood of x^*;
– *Step 2*: If $f(x) < f^*$, then $x^* := x$, $f^* := f(x)$, and go to *step 1*. Else *STOP*.

In the local search algorithm we consider three neighbourhood types: *move*, KL (Kernighan-Lin) [9] and *rand*. Neighbour of a feasible solution x in a neighbourhood *move* is arbitrary feasible solution which is received from x by switching a disk to another server or swapping a disk with another disk between two servers. Let $\tilde{D}$ be a set of disks. Neighbour of a feasible solution x in a neighbourhood *move*$(\tilde{D})$ is arbitrary feasible solution which is received from x as follows. We take a disk (or two disks) which does not belong to the set $\tilde{D}$ and move the disk(s) as in the neighbourhood *move*. We use L_{max} to denote the number transformations the solution which was moved in the neighbourhood *move*$(\tilde{D})$. Neighbour of some feasible solution x in a neighbourhood KL is arbitrary feasible solution which is received from x as the result of implementation of the following algorithm:

– *Step 0*: Given $\tilde{x} := x$, $x^* := x$, $\tilde{f} := f(x)$, and $D := \emptyset$;

	LP		ILP		MLS	
$\overline{c_{sr}}$	time	value	time	value	time	value
850	2	0	6	0	1	0
840	2	0	7	0	1	0
830	2	0	17	0	1	0
820	2	0	70	0	2	0
810	2	0	3000	1834	6	0
800	2	0	3000	2480	14	0
750	198	0	—	—	1200	13965
700	423	96911	3000	188306	1200	129511
650	160	382399	607	407792	1200	383448
600	58	682399	85	702039	1	682399

Table 3. Computational results for the case $B^w_{sr} = B^e_{sr} = 50$.

– *Step 1*: Repeat the following procedure L_{max} times. Find the best neighbour $x^{'}$ of x^* in $move(D)$. If solution $x^{'}$ differs from x^* only transferring some disk d to another server, then $D := D \cup d$. Else, for definiteness, let two disks d and $d^{'}$ were swaped, than $D := D \cup d \cup d^{'}$. Given $x^* := x^{'}$. If $f(x^*) < \tilde{f}$, then $\tilde{x} := x^*$ and $\tilde{f} := f(x^*)$.

In *move* disks are viewed alternately and to any disk the best server or swapping disk are selected. In contrary, in the neighbourhood *rand* we select the disk randomly with uniform distribution. After that, we find the best server or swapping disk. *LS* algorithm with the neighbourhood *rand* stops after IR_{max} steps.

If we use *move* (KL or *rand*) neighbourhood, then call the local search algorithm and genetic local search as $LS(move)$ ($LS(KL)$ or $LS(rand)$) and $GLS(move)$ ($GLS(KL)$ or $GLS(rand)$), respectively.

5 Computational results

Presented algorithms LS and GLS were implemented in C++ and tested on PC Intel Core i7-3612QM with 4 GB RAM. We used known instances with random initial data from [1, 2] to explore our algorithms. At these instances there is following dimension: $|S| = 20$, $|D| = 200$, $|T| = 150$, $|R| = 2$. The values x^0_{ds} that of the initial distribution of disks on servers were generated randomly with uniform distribution. For each disk one server was chosen randomly. The load value and additional costs for each disk an average load value c_d, $d \in D$, was initially assigned. It was chosen uniformly on the interval $[50, 100]$. Then the following values were assigned: $c_{drt} = c_d/10 + \mu_{drt}$, $b^w_{sdr} = c_d/10 + \nu_{sdr}$, $b^e_{sdr} = c_d/10 + \theta_{sdr}, s \in S, d \in D, r \in R$. Here the values $\mu_{drt} \in [-20, 20]$ and $\nu_{sdr}, \theta_{sdr} \in [-3, 3]$ were also chosen as independent uniform random values from the specified intervals.

We divide computational experiments in two groups. In the first group (see tables 1 – 4) instances differ from each over only the choice of threshold loads and maximum

	LS(move)		LS(LK)		LS(rand)		GLS(move)	
$\overline{c_{sr}}$	time	value	time	value	time	value	time	value
850	57	0	52	0	1	0	1	0
840	63	0	52	0	1	0	1	0
830	62	0	53	0	3	0	3	0
820	63	0	54	0	2	0	2	0
810	76	0	62	0	3	0	3	0
800	92	0	75	0	3	0	3	0
750	169	10419	176	10485	29	11100	38	10522
700	211	124552	217	124703	24	123983	24	123983
650	156	383016	210	383073	21	383228	19	383068
600	62	682399	511	682399	<1	682399	<1	682399

Table 4. Computational results for the case $B^w_{sr} = B^e_{sr} = 50$ (extension).

allowed additional costs. In the tables 1 and 2 maximum allowed additional costs are equal to 150. In contrary, in the tables 3 and 4 it is equal to 50. Moreover, threshold loads are the same for all servers. Their value is specified in the first column of tables 1 – 4 and characterizes the number of instance. According to the computational experiment, instances with threshold loads equal to 700 were the most difficult for known algorithms and new methods. Therefore, the second group (see table 5) instances consist of 10 instances with maximum allowed additional costs equal to 150 and threshold loads equal to 700, i.e. we can say that these instances are the most difficult instances of servers load balancing problem for real dimension.

In the computational experiments we used the following empirical parameters of algorithms. We took $I_{max} = 200$, $P = 6$, $IR_{max} = 20000$, $C_{max} = L_{max} = 30$. We can increase IR_{max} to 100000 – 200000 and decrease I_{max} to 50 – 100 to reduce the computation time with small loss.

Algorithms $LS(move)$, $LS(KL)$, $LS(rand)$, $GLS(move)$, and $GLS(KL)$ are compared with servers load balancing problem linear relaxation (LP), algorithm ILP from [1] based on the transformation the solution of linear relaxation and local search with neighborhood of exponential size (algorithm MLS) described in [2].

Columns value and time correspond to objective function value and computational time of the best found solution, respectively. Experimental studies show the high efficiency of the developed genetic local search methods in comparison with known algorithms. Local search algorithm with randomized neighbourhood often obtains a good solution in short time. Therefore, we used this method to obtain a good initial population. Algorithms $LS(rand)$ and $GLS(move)$ were more efficient than method ILP and comparable with method MLS. Moreover, the relative deviation from the linear relaxation does not exceed 0,12 (12 percent). Finally, algorithm $GLS(move)$ does not significantly exceed in accuracy and significantly exceed in computational time $GLS(KL)$ (see table 2). Algorithms $LS(rand)$, $LS(move)$, and $LS(KL)$ are same in accuracy, but $LS(rand)$ significantly exceed $LS(move)$ and $LS(KL)$ in computational time (see tables 2 and 4).

	LP	ILP	MLS	LS(rand)		GLS(move)	
inst	value	value	value	time	value	time	value
1	234387	331819	246780	38	246599	1454	245225
2	327380	367169	329868	18	330406	1805	330231
3	486154	498036	531918	29	527598	1475	527904
4	404395	488143	417768	34	418865	1523	417525
5	346045	393182	353489	28	353449	1516	353340
6	341400	475071	384536	41	384845	1805	384134
7	407088	428234	407594	29	407722	1482	407648
8	256666	325084	281827	31	276887	1320	276634
9	322371	361414	325038	19	325544	1480	325091
10	549565	567503	561315	23	555645	23	555645

Table 5. Computational results for the case $\overline{c_{sr}} = 700$.

6 Conclusion

In this work the servers load balancing problem is considered. For solution of this problem new approximate approach based on the genetic local search is proposed. Experimental studies show high efficiency of the developed approximate methods. Approximability of the problem is analyzed. It is proven that the problem is Log-APX-hard under PTAS-reducibility and cannot be NPO-complete under AP-reducibility unless P=NP.

For the further research it is interesting to consider neighborhood of exponential size (see [2]) and another neighbourhoods, new crossing procedures, for example, procedure of optimal crossing [11], path relinking algorithm [9] and another heuristic methods. As the exact location of the problem in the approximation hierarchy did not established, it is planned to specify the upper and lower approximation bound.

Acknowledgments. This research was supported by the Ministry of Education and Science of the Republic of Kazakhstan (grant 0115PK00546).

References

1. Kochetov, Yu.A., Kochetova, N.A.: Problem of Load Balancing Servers. Vestnik Novosib. Gos. Univ. Ser. Inform. Tekhnol. 11(4), 71–76 (2013)
2. Davydov, I., Kochetov, Yu., Kononova, P.: Local Search with an Exponential Neighborhoodfor the Servers Load Balancing Problem. Journal of Applied and Industrial Mathematics. 9(1), 27-35 (2015)
3. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and approximation: combinatorial optimization problems and their aproximability properties. Berlin: Springer-Verlag, (1999)
4. Bazgan, C., Escoffer, B., Paschos, V.: Completeness in standard and differential approximation classes: Poly-(D)APX- and (D)PTAS-completeness. Theoret. Comput. Sci. 339, 272-292 (2005)

5. Crescenzi, P., Kann, V., Silvestri, R., Trevisan, L.: Structure in approximation classes. SIAM J. COMPUT. 28(5), 1759-1782 (1999)
6. Crescenzi, P., Trevisan, L.: On approximation scheme preserving reducibility and its application. Proc. 14th Annual Conference on Foundation of Software Technology and Teoretical Computer Science. Lecture Notes in Computer Science 880, Springer-Verlag, Belrin, 330-341 (1994)
7. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. San Francisco, Freeman, (1979); Mir, Moscow, (1982)
8. Escoffier, B., Paschos, V.Th.: Completeness in approximation classes beyond APX. Theoret. Comput. Sci. 359, 369-377 (2006)
9. Kochetov, Yu.A., Plyasunov, A.V.: Genetic local search the graph partitioning problem under cardinality constraints. Computational Mathematics and Mathematical Physics. 52(1), 157-167 (2012)
10. Kochetov, Yu., Panin, A., Plyasunov, A.: Comparison of metaheuristics for the bilevel facility location and mill pricing problem. Journal of Applied and Industrial Mathematics. 9(3), 392-401 (2015)
11. Eremeev, A.V., Kovalenko, Ju.V.: Optimal recombination in genetic algorithms for combinatorial optimization problems – part I. Yougoslav Journal of Operations Research. 24(1), 1-20 (2014)