Serdica
Bulgariacae mathematicae publicationes

Сердика
Българско математическо списание

COMPUTATIONAL COMPARISONS FOR METHOD OF CENTERS OF GRAVITY OF VERTICES

TATIANA I. ENCHEVA

When solving nonlinear programming problems with objective functions whose evaluation is rather difficult and complex the question of minimizing the step number comes to the fore. Among methods optimal with respect to the number of iterations there are some methods of central sections. The method of centers of gravity of vertices has by now no theoretical justification of its convergence. Thus the experimental testing becomes rather important. In the present paper some conclusions are made based on extensive experimental material. They reveal the practical efficiency of the method and its closeness to optimal methods. Moreover, this method and some other known methods of central sections are considered and compared to a certain extent.

1. Introduction. The methods of central sections (MCS) were introduced in [1] as a tool for solving convex programming problems with small dimension and labour consuming evaluation of the objective function. The interest in these methods has grown considerably in the recent years. On the one hand, the scope of applicability of MCS has been extended over the class of quasi-convex objective functions [2, 3]. On the other hand, there are some very important practical problems (e. g. the block convex programming problem and the linear time-optimal control problem [4]) for which the use of MCS might be the most advisable.

The great variety of MCS can be separated into two groups — methods with inclusions and methods without inclusions. To the first group belong the ellipsoid method (EM) [5, 6], the simplex method (SM) [7, 8] and their numerous modifications. Among the methods of the second group is the first MCS — the method of centers of gravity (MCG) [1], the method of inscribed ellipsoids (MIE) [9], the stochastic MCS, the method of centers of gravity of vertices (MCGV) [10]. While the methods with inclusions have been tested even experimentally (see e. g. [11, 12]), the methods without inclusions have not been thoroughly studied in this respect. This is due to the fact that they comprise difficult geometrical constructions which hamper their computer realization. However, most of these methods are optimal with respect to the number of iterations (see [13, 9]). Hence, they play an important role when solving problems with an objective function being hard for computation.

2. The scheme of the MCGV. First let us describe the general iteration of all MCS. Consider the problem of minimizing a continuous quasi-convex function f over a convex compact $M_0 \subset \mathbf{R}^n$ ($\operatorname{int} M_0 \neq \emptyset$). Denote by $x_{\min}$ the arbitrary minimum point and by $\square f(x)$ the quasi-gradient [3] of f at the arbitrary point $x \notin M_0$. Let M_k, $k=0, 1, \ldots$, be the localizer (i. e. the domain, which contains the point $x_{\min}$) after k iterations. On the recurrent $k+1$ iteration a certain "central" point $x_k \in M_k$ (more often the center of gravity of M_k) is selected. Next $f(x_k)$ and $\square f(x_k)$ are computed. If $\square f(x_k)=0$, then $x_k=x_{\min}$, i. e. the problem is solved. Otherwise, a part of M_k lying in the half-space $\langle \square f(x_k), x-x_k \rangle > 0$ and not containing minimum points is cut off.

For the deterministic MCS (in particular for MCG, MIE, EM, SM) it is proved that drawing cutting hyperplanes through some localizer points being central in a certain respect one can achieve that the volumes of the localizers decrease exponentially. Furthermore, small localizer volumes imply small errors with respect to the functional [13, 3].

SERDICA Bulgaricae mathematicae publicationes. Vol. 16, 1990, p. 188—193.

The MCGV is intended to solve quasi-convex programming problems for which the initial localizer M_0 (and therefore the subsequent localizers M_k) is a convex polytope. As a central point of the polytope M we choose either the center of gravity of its vertices $v_1, v_2, \ldots, v_m$ $z(M)=m^{-1}\sum_{i=1}^{m} v_i$ or its stochastic image $z(M, \alpha)=\sum_{i=1}^{m}\alpha_i v_i$, where $\alpha=(\alpha_1, \alpha_2, \ldots, \alpha_m)$ is a random point uniformly distributed in the simplex $\sum_{i=1}^{m}\alpha_i=1$, $\alpha_i\geq 0$, $i=1, 2, \ldots m$. We shall speak of a deterministic ($z(M)$) and a randomized ($z, M(\alpha)$) versions of the MCGV.

One of the aims of the randomized version of the MCGV is to obtain a flexible scheme, which can be stochastically varied (at the expense of randomness of α). In particular, this version can be used to minimize functions which are not quasi-convex. In this case the point $x_{\min}$ can be found after a sequence of "trials". Here the MCGV behaves to a certain extent like the methods of random search which mainly use a gradient descent instead of central sections.

The main advantage of the MCGV is the simple way of finding a central point if the vertices of the polytope are known. However, from a practical point of view the problem of finding these vertices is difficult enough in itself (though it is easier than the problem of finding the exact center of gravity of the polytope). On the other hand, it is impossible to obtain a theoretical estimate of the volume contraction of the localizers (if $n>2$) for the MCGV. This fact might be illustrated if a pyramid with a great number of base vertices and a section parallel to the base is considered. Thus, the results of numerical experiments gain great significance.

3. Results of the computational experiments. The computational realization of the MCGV is accomplished by using the organization of the data base proposed in [14]. It is intended to be used for solving a sequence of standard problems referring to convex polytopes in a mode being quite rapid and efficient with respect to the memory capacity used. The programs are written in FORTRAN. All the experiments have been performed on an IBM 4341 model M2 computer. The computer executes about 1.5 million machine instructions per second.

Seventeen well known test functions [15, 16] have been used for the experiments. The problems of minimizing f on certain "parallelepipeds" $M_0\subset R^n$ have been solved. The number of problems solved has been 335. The computational procedure terminates when the inequality $f(x_k)-f(x_{\min})<10^{-11}$ has been satisfied for the current point x_k. The computational results for all the test problems are given in Tables 1 and 2, where the notations are as follows:

m — number of solved problems;
kmin — minimal number of iterations;
ks — average number of iterations;
ts — average time (in seconds) for solving one problem;
tsi — average time (in seconds) for performing one iteration.

With the two versions of the MCGV it turned out that for the group of problems under consideration the average number of iterations required to reach the error level of 10^{-11} showed an increase with respect to n, which is close to the linear one (especially for the version $z(M)$). In Fig. 1 the values $ks=ks(n)$ for the two versions and the approximating straight lines constructed by the method of least squares are given. The respective equations are as follows:

$$k(n)=31.349n-23.682 \text{ (for } z(M)\text{)};$$

$$k(n)=36.160n-14.875 \text{ (for } z(M, \alpha)\text{)}.$$

Fig. 1 shows that the linear approximation is very good for the version $z(M)$; the quality of the linear approximation is a bit worse for the version $z(M, \alpha)$.

Table 1

Average results of solving the problems to within 10^{-11} by the version z (M)

n	*m*	*kmin*	*ks*	*ts*	*tsi*
2	30	25	39	0.192	0.00493
3	30	42	71	1.021	0.01431
4	25	56	100	4.131	0.04126
5	10	81	134	25.971	0.19410

Table 2

Average results of solving the problems to within 10^{-11} by the version z (M, α)

n	*m*	*kmin*	*ks*	*ts*	*tsi*
2	80	21	53	0.335	0.00636
3	80	67	105	1.981	0.01893
4	50	79	122	9.497	0.07769
5	30	123	167	46.751	0.27950

Let us assume that the linearly increasing rate of *ks* as a function of *n* obtained empirically for a comparatively not large range of dimensions reflects the real increase of $ks(n)$ on the whole. This fact can be interpreted as follows: with the MCGV the volumes of the localizers decrease (on the average) geometrically with a ratio independent of *n*. This shows that in practice the MCGV, like the MCG, behaves as an optimal method with respect to the number of iterations. Recall that, for example, with the EM the volumes of the localizers decrease geometrically with a ratio approximately equal to $1-1/2n$; respectively, the number of the iterations increases not linearly with respect to *n*, but as const. n^2.

Now let us discuss the mean time *tsi* spent for one iteration with respect to the dimension *n* approximating $tsi(n)$ by an exponential curve. Respectively $\ln tsi(n)$ is to be approximated by a linear function of *n*. The results of the approximation obtained by the method of least squares are presented on Fig. 2 and show that the increase of $\ln tsi(n)$ is close enough to the linear one (naturally, in the examined range of values of *n*). The equations of the approximating straight lines are as follows:

$$t(n)=1.208n-7.824 \text{ (for } z(M)\text{)};$$

$$t(n)=1.276n-7.680 \text{ (for } z(M, \alpha)\text{)}.$$

For comparison note that with the MIE the time for one iteration grows as a polynomial in *n*. Thus, for great *n* the MIE will be less labour consuming than MCGV (if we assume that the exponential dependence remains valid for great *n* as well). In this connection the purely practical aspect becomes quite important. The high power (n^8) with the MIE and the exponential nature of the labour consumption increase for one iteration assumed with the MCGV show that the two methods would be realizable only for not large *n*. As shown by Tables 1 and 2 the average time for one iteration with the two MCGV versions grows 40 times when *n* increases from 2 to 5. On the other hand, n^8 grows more than 1500 times when *n* increases from 2 to 5. Thus, we can suppose that MCGV might work more rapidly than MIE. (Nevertheless, everything mentioned above does not mean that the theoretical significance of MIE is less important).

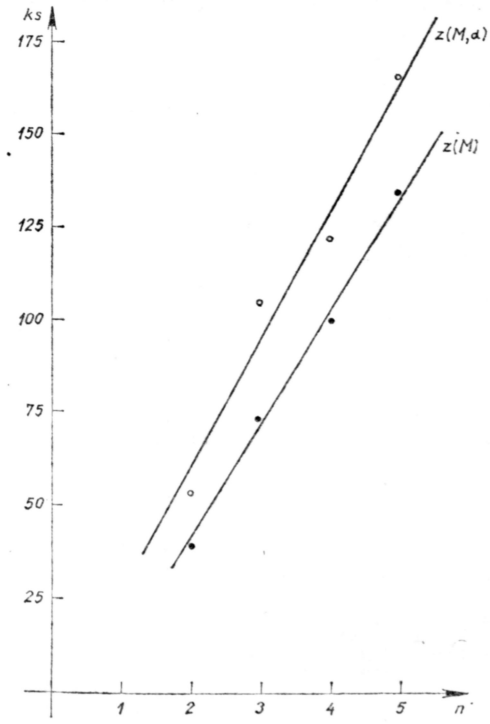

Fig. 1. Average number of iterations as a function of the problem dimension n

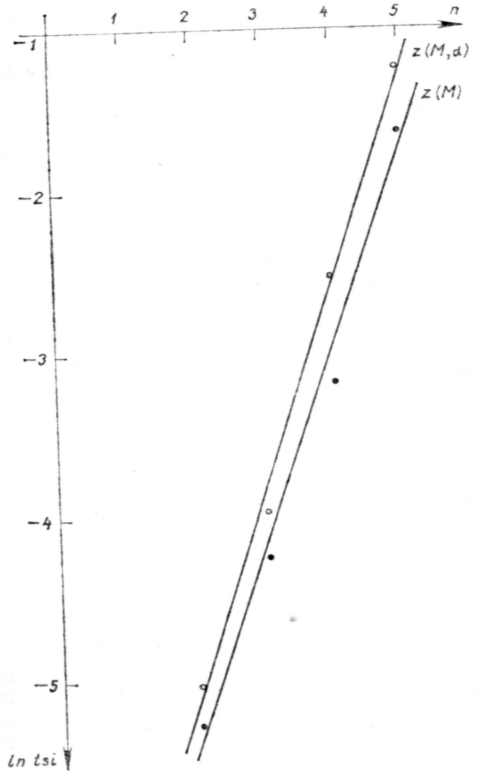

Fig. 2. Increase of the average time for one iteration as a function of the problem dimension n

Now let us compare the results obtained when using the two MCGV versions Tables 1 and 2 and the approximating functions given above show that the deterministic version is more efficient than the randomized one. The number of iterations required by the version $z(M)$ is about 1.3 times less than that required by the version $z(M, \alpha)$. The time spent for one iteration with the version $z(M)$ is about 1.5 times less than that with the version $z(M, \alpha)$. Thus, if it is known in advance that f is quasi-convex in M_0, in particular convex, then the use of the version $z(M, \alpha)$ is less preferable than that of the version $z(M)$. On the other hand, if there is lack of exact information about the required properties of f, then the randomized version, as mentioned above, might be useful for solving the problem.

Finally let us compare the number of iterations actually performed through the versions $z(M)$ and $z(M, \alpha)$ with the upper bounds for a number of iterations for the following MCS: MCG, MIE, EM, SM. We have considered test problems of one and the same type, namely the problems of minimizing the functions $\sum_{k=1}^{n} x_2$ and $\sum_{k=1}^{n} x_k^2/k!$, $n=3, 4, 5, 6$ over a certain "parallelepiped". The last two columns of Tables 3 and 4 comprise the numbers of iterations practically performed by the versions $z(M)$ and $z(M, \alpha)$ (the randomized version has been used twice and the average number of iterations is shown in the tables). Two upper estimates for the number of iterations obtained

Table 3

Number of iterations needed to minimize functions $\sum_{k=1}^{n} x_k^2$ *(n=3, 4, 5, 6) to within* 10^{-11}

n	MCG	MIE	EM	SM	MCGV	
					z (M)	z (M, α)
3	198	533	499	1591	65	103
	101	273	255	814		
4	264	710	908	3817	86	125
	137	367	469	1973		
5	329	883	1431	7463	115	149
	169	455	738	3848		
6	402	1079	2117	13172	135	146
	208	557	1094	6805		

Table 4

Number of iterations needed to minimize functions $\sum_{k=1}^{n} x_k^2/k!$ *(n=3, 4, 5, 6) to within* 10^{-11}

n	MCG	MIE	EM	SM	MCGV	
					z (M)	z (M, α)
3	194	522	489	1560	60	103
	101	273	255	814		
4	257	690	882	3710	84	126
	137	367	469	1973		
5	317	852	1380	7198	97	125
	169	455	738	3848		
6	385	1035	2021	12635	117	139
	208	557	1094	6805		

by the formulae from [13] and [3] respectively are given in columns II-V for each method. In all cases the estimate for the number of iterations obtained by the second formula is significantly smaller (approximately twice) than the estimate obtained by the first formula. This can be explained by the fact that in the test problems considered $x_{\min}$ has been a stationary point (see [3] for more details).

As it is seen from Tables 3 and 4 the number of iterations needed for the versions $z(M)$ and $z(M, \alpha)$ is essentially smaller than that for the other methods (even if we consider the better of the two estimates given). This can be explained by two reasons. First, we compare the actual number of iterations with the upper estimates. Probably that might be the explanation for the "success" of the MCGV compared with that of the MCG. The gap between the actual number of iterations and the upper estimates is remarkably great with SM. (Note that in practice the number of iterations for SM is usually less than that for EM). The other reason for the MCGV success is evidently its practical efficiency.

4. Conclusion. The experiments performed allow us to assume that the MCGV is practically efficient and surpasses the majority of known MCS for the problems with dimension $n \leq 6$. What concerns the practical efficiency we naturally have in view problems with values of f and $\Box f$ being hard to compute. Test functions defined by simple analytical expressions are not typical at all. Nevertheless, we have chosen

them so as to carry out a vast number of experiments within a reasonable computer time. An illustration of a problem being hard with respect to computing the values of f and $\square f$ and being important with regard to its application is the linear time-optimal control problem [4] mentioned above. It can be reduced to the problem of minimizing a certain quasi-convex function f. This function is described implicitly, but the value of $f(x)$ and the direction of $\square f(x)$ might be computed in a way which is quite labour consuming even for problems whose dimension is not large.

The statistical conclusions about the practical efficiency of the MCGV are based on experimental data and they should not be taken for granted. On the other hand, it is known that the practical efficiency of the algorithms is often shown more precisely through experimental data than through theoretical estimations (the simplex method in linear programming is the most typical example of this kind). Furthermore, we would like to point out that to prove theoretically the relations found empirically might be very difficult. It is impossible to obtain a deterministic estimate for the rate of reducing the localizers volumes with MCGV for $n>2$, which has been illustrated by the simple counter-example mentioned above. The rigorous proof of the problem is naturally connected with its probabilistic interpretation. Moreover, it would be necessary to put some additional restrictions on the objective function. Treating the problem in such a way seems to be rather problematic.

Acknowledgements. The author would like to thank A. Ju. Levin and D. L. Vandev for their help and guidance.

REFERENCES

1. А. Ю. Левин. Об одном алгоритме минимизации выпуклых функций. *Докл. АН СССР*, **160**, 1965, 6, 1241—1243.
2. А. С. Немировский, Д. Б. Юдин. Информационная сложность математического программирования. *Техническая кибернетика*, **1**, 1983, 88—117.
3. T. I. Encheva, A. Yu. Levin. Central sections in quasi-convex programming. *C. R. Acad. Bulg. Sci.*, **42**, 1989, 11, 39-42.
4. T. I. Encheva, A. Yu. Levin. Central sections in linear time-optimal control problems. *C. R. Acad. Bulg. Sci.*, **43**, 1990, 1, 33-37.
5. Д. Б. Юдин, А. С. Немировский. Информационная сложность и эффективные методы решения выпуклых экстремальных задач. *Экономика и матем. методы*, **12**, 1976, 2, 357—369.
6. Н. З. Шор. Методы отсечения с растяжением пространства для решения задач выпуклого программирования. *Кибернетика*, **1**, 1977, 94—95.
7. B. Yamnitsky, L. A. Levin. An old linear programming algorithm runs in polynomial time 23rd Ann. Symp. Found. Comput. Sci., Chicago, Ill., 3-5 Nov., 1982. Silver Spring, 1982, 327-328.
8. И. А. Александров, Е. Г. Анциферов, В. П. Булатов. К методам центрированных отсечений. Тезисы докл. конф. по матем. программированию. Свердловск, 1981, 162—163.
9. С. П. Тарасов, Л. Г. Хачиян, И. И. Эрлих. Метод вписанных эллипсоидов. *Докл. АН СССР*, 298, 1988, 5, 1081—1085.
10. T. I. Encheva, A. Ju. Levin, D. L. Vandev. On computationally implementable variants of the method of centers of gravity. *Mathematica Balkanica*, **2**, 1988, 2-3, 156-164.
11. J. G. Ecker, M. Kupferschmid. A computational comparison of several nonlinear programming algorithms. Report OR-S-82-2, New York, Rensselaer Polytechnic Institute. Troy, 1982.
12. P. F. Pickel. An update on the allipsoid algorithm. Discrete geometry and convexity. *Annals of the New York Acad. of Sci.*, **440**, 1985, 364-380.
13. Б. Т. Поляк. Введение в оптимизацию. Москва, 1983.
14. T. I. Encheva, D. L. Vandev. Maintenance of a polytope in the computer memory. Proceed. of XIII Nat. School on Appl. of Math. in Techn., Varna, 30 Aug.-6 Sept., 1987, 183-186.
15. Д. Химмельблау. Прикладное нелинейное программирование. Москва, 1975.
16. W. Hock, K. Schittkowski. Test examples for nonlinear programming codes. *Lecture Notes in Economics and Mathematical Systems*. Berlin, **187**, 1981.

Bulgarian Academy of Sciences
Institute of Mathematics
1090 Sofia, P. O. Box 373.

Received 27.09.1989