Detecting promising areas by evolutionary clustering search

Alexandre C. M. Oliveira[1],[2] and Luiz A. N. Lorena[2]

[1] Universidade Federal do Maranhão - UFMA, Departamento de Informática, S. Luís MA, Brasil.
acmo@deinf.ufma.br

[2] Instituto Nacional de Pesquisas Espaciais - INPE, Laboratório Associado de Computação e Matemática Aplicada, S. José dos Campos SP, Brasil.
lorena@lac.inpe.br

Abstract. A challenge in hybrid evolutionary algorithms is to define efficient strategies to cover all search space, applying local search only in actually promising search areas. This paper proposes a way of detecting promising search areas based on clustering. In this approach, an iterative clustering works simultaneously to an evolutionary algorithm accounting the activity (selections or updatings) in search areas and identifying which of them deserves a special interest. The search strategy becomes more aggressive in such detected areas by applying local search. A first application to unconstrained numerical optimization is developed, showing the competitiveness of the method.

Keyword: Evolutionary algorithms; unconstrained numerical optimization.

1 Introduction

In the hybrid evolutionary algorithm scenario, the inspiration in nature have been pursued to design flexible, coherent and efficient computational models. The main focus of such models are real-world problems, considering the known little effectiveness of canonical genetic algorithms (GAs) in dealing with them. Investments have been made in new methods, which the evolutionary process is only part of the whole search process. Due to their intrinsic features, GAs are employed as a generator of promising search areas (search subspaces), which are more intensively inspected by a heuristic component. This scenario comes to reinforce the parallelism of evolutionary algorithms.

Promising search areas can be detected by fit or frequency merits. By fit merits, the fitness of the solutions can be used to say how good their neighborhood are. On other hand, in frequency merits, the evolutionary process naturally privileges the good search areas by a more intensive sampling in them. Figure 1 shows the 2-dimensional contour map of a test function known as *Langerman*. The points are candidate solutions over fitness surface in a typical simulation. One can note their agglomeration over the promising search areas.

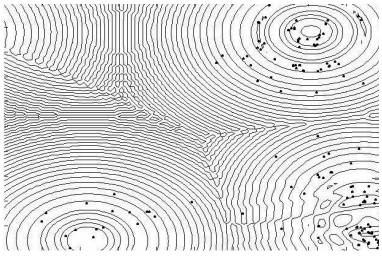

Fig. 1. Convergence of typical GA into fitter areas

The main difficulty of GAs is a lack of exploitation moves. Some promising search areas are not found, or even being found, such areas are not consistently exploited. The natural convergence of GAs also contributes for losing the reference to all promising search areas, implicating in poor performance.

Local search methods have been combined with GAs in different ways to solve multimodal numerical functions more efficiently. Gradient as well as direct search methods have been employed as exploitation tool. In the Simplex Genetic Algorithm Hybrid [1], a probabilistic version of Nelder and Mead Simplex [2] is applied in the elite of population. In [3], good results are obtained just by applying a conjugate gradient method as mutation operator, with certain probability. In the Population-Training Algorithm [4], improvement heuristics are employed in fitness definition, guiding the population to settle down in search areas where the individuals can not be improved by such heuristics. All those approaches report an increase in function calls that can be prohibitive in optimization of complex computational functions.

The main challenge in such hybrid methods is the definition of efficient strategies to cover all search space, applying local search only in actually promising areas. Elitism plays an important role towards achieving this goal, once the best individuals represent such promising search area, *a priori*. But the elite of population can be concentrated in few areas and thus the exploitation moves are not rationally applied.

More recently, a different strategy was proposed to employ local search more rationally: the Continuous Hybrid Algorithm (CHA) [5]. The evolutionary process run normally until be detected a promising search area. The promising area is detected when the highest distance between the best individual and other individuals of the population is smaller than a given radius, i.e., when population diversity is lost. Thereafter, the search domain is reduced, an initial simplex is built inside the area around the best found individual, and a local search based upon Nelder and Mead Simplex is started. With respect to detection of promising areas, the CHA has a limitation. The exploitation is started once, after diversity loss, and the evolutionary process can not be continued afterwards, unless a new population takes place.

Another approach attempting to find out relevant areas for numerical optimization is called UEGO by its authors. UEGO is a parallel hill climber, not an

evolutionary algorithm. The separated hill climbers work in restricted search areas (or clusters) of the search space. The volume of the clusters decreases as the search proceeds, which results in a cooling effect similar to simulated annealing [6]. UEGO do not work so well as CHA for high dimensional functions.

Several evolutionary approaches have evoked the concept of species, when dealing with optimization of multimodal and multiobjective functions [6],[7]. The basic idea is to divide the population into several species according to their similarity. Each species is built around a dominating individual, staying in a delimited area.

This paper proposes an alternative way of detecting promising search areas based on clustering. This approach is called Evolutionary Clustering Search (ECS). In this scenario, groups of individuals (clusters) with some similarities (for example, individuals inside a neighborhood) are represented by a dominating individual. The interaction between inner individuals determines some kind of exploitation moves in the cluster. The clusters work as sliding windows, framing the search areas. Groups of mutually close points hopefully can correspond to relevant areas of attraction. Such areas are exploited as soon as they are discovered, not at the end the process. An improvement in convergence speed is expected, as well as a decrease in computational efforts, by applying local optimizers rationally.

The remainder of this paper is organized as follows. Section 2 describes the basic ideas and conceptual components of ECS. An application to unconstrained numerical optimization is presented in section 3, as well as the experiments performed to show the effectiveness of the method. The findings and conclusions are summarized in section 4.

2 Evolutionary Clustering Search

The Evolutionary Clustering Search (ECS) employs clustering for detecting promising areas of the search space. It is particularly interesting to find out such areas as soon as possible to change the search strategy over them. An area can be seen as an abstract search subspace defined by a neighborhood relationship in genotype space.

The ECS attempts to locate promising search areas by framing them by clusters. A cluster can be defined as a tuple $\mathcal{G} = \{c, r, s\}$, where c and r are the *center* and the *radius* of the area, respectively. There also exists a *search strategy* s associated to the cluster. The radius of a search area is the distance from its center to the edge.

Initially, the center c is obtained randomly and progressively it tends to slip along really promising points in the close subspace. The total cluster volume is defined by the radius r and can be calculated, considering the problem nature. The important is that r must define a search subspace suitable to be exploited by aggressive search strategies.

In numerical optimization, it is possible to define r in a way that all search space is covered depending on the maximum number of clusters. In combinatorial

optimization, r can be defined as a function of some distance metric, such as the number of movements needed to change a solution inside a neighborhood. Note that neighborhood, in this case, must also be related with the search strategy s of the cluster. The search strategy s is a kind of local search to be employed into the clusters and considering the parameters c and r. The appropriated conditions are related with the search area becoming promising.

2.1 Components

The main ECS components are conceptually described here. Details of implementation are left to be explained later. The ECS consist of four conceptually independent parts: (a) an evolutionary algorithm (EA); (b) an iterative clustering (IC); (c) an analyzer module (AM); and (d) a local searcher (LS). Figure 2 brings the ECS conceptual design.

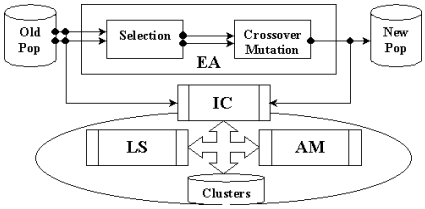

Fig. 2. ECS components

The EA works as a full-time solution generator. The population evolves independently of the remaining parts. Individuals are selected, crossed over, and updated for the next generations. This entire process works like an infinite loop, where the population is going to be modified along the generations.

The IC aims to gather similar *information* (solutions represented by individuals) into groups, maintaining a representative solution associated to this information, named the center of cluster. The term *information* is used here because the individuals are not directly grouped, but the similar information they represent. Any candidate solution that is not part of the population is called information. To avoid extra computational effort, IC is designed as an iterative process that forms groups by reading the individuals being selected or updated by EA. A similarity degree, based upon some distance metric, must be defined, *a priori*, to allow the clustering process.

The AM provides an analysis of each cluster, in regular intervals of generations, indicating a probable promising cluster. Typically, the *density* of the cluster is used in this analysis, that is, the number of selections or updatings recently happened. The AM is also responsible by eliminating the clusters with lower densities.

At last, the LS is an internal searcher module that provides the exploitation of a supposed promising search area, framed by cluster. This process can happen

after AM having discovered a target cluster or it can be a continuous process, inherent to the IC, being performed whenever a new point is grouped.

2.2 The clustering process

The clustering process described here is based upon Yager's work, which says that a system can learn about an external environment with the participation of previously learned beliefs of the own system [8],[9].

The IC is the ECS's core, working as an information classifier, keeping in the system only relevant information, and driving a search intensification in the promising search areas. To avoid propagation of unnecessary information, the local search is performed without generating other points, keeping the population diversified. In other words, clusters concentrate all information necessary to exploit framed search areas.

All information generated by EA (individuals) passes by IC that attempts to group as known information, according to a distance metric. If the information is considered sufficiently new, it is kept as a center in a new cluster. Otherwise, redundant information activates a cluster, causing some kind of perturbation in it. This perturbation means an assimilation process, where the knowledge (center of the cluster) is updated by the innovative received information.

The assimilation process is applied over the center c, considering the new generated individual p. It can be done by: (a) a random recombination process between c and p, (b) deterministic move of c in the direction of p, or (c) samples taken between c and p. Assimilation types (a) and (b) generate only one internal point to be evaluated afterwards. Assimilation type (c), instead, can generate several internal points or even external ones, holding the best evaluated one to be the new center, for example. It seems to be advantageous, but clearly costly. These exploratory moves are commonly referred in path relinking theory [10].

Whenever a cluster reaches a certain density, meaning that some information template becomes predominantly generated by the evolutionary process, such information cluster must be better investigated to accelerate the convergence process in it. The cluster activity is measured in regular intervals of generations. Clusters with lower density are eliminated, as part of a mechanism that will allow to create other centers of information, keeping framed the most active of them. The cluster elimination does not affect the population. Only the center of information is considered irrelevant for the process.

3 ECS for unconstrained numerical optimization

A sequential real-coded version of ECS for unconstrained numerical optimization is presented in this section. Several test functions can be found in literature related to unconstrained numerical optimization (or function parameter optimization). The general presentation of such problems is:

$$\min \ / \ \max f(x), x = (x_1, x_2, x_3, \ldots, x_n)^T \in R^n \tag{1}$$

$$\text{where } L_i < x_i < U_i$$

In test functions, the upper U_i and lower L_i bounds are defined *a priori* and they are part of the problem, bounding the search space over the challenger areas in function surface. This work uses some of well-known test functions, such as *Michalewicz*, *Langerman*, *Shekel* [11], *Rosenbrock*, *Sphere* [12], *Schwefel*, *Griewank*, and *Rastrigin* [13]. Table 1 shows all test functions, their respective known optimal solution and bounds.

Table 1. Test functions

Function	var	opt	$L_i; U_i$	Function	var	opt	$L_i; U_i$	Function	var	opt	$L_i; U_i$
Ackley	n	0	-15 ; 30	Goldstein	2	3	-2 ; 2	Zakharov	n	0	-5 ; 10
Sphere	n	0	-5.12;5.12	Griewank	n	0	-600; 600	Rastrigin	n	0	-5.12;5.12
Easom	2	-1	-100;100	Hartman	6	-3.322	0 ; 1	Rosenbrock	n	0	-5.12;5.12
Michalewicz	5	-4.687	0; π	Langerman	5	-1.4	0;10	Schwefel	n	0	-500;500
Michalewicz	10	-9,66	0; π	Langerman	10	-1.4	0;10	Shekel 10	4	-10.536	-10 ; 10

3.1 Implementation

The application details are now described, clarifying the approach. The component EA is a steady-state real-coded GA employing well-known genetic operators as roulette wheel selection [14], blend crossover ($BLX0.25$) [15], and non-uniform mutation [16]. Briefly explaining, in each generation, a fixed number of individuals $\mathcal{NS}$ are selected, crossed over, mutated and updated in the same original population, replacing the worst individual (steady-state updating). Parents and offspring are always competing against each other and the entire population tends to converge quickly.

The component IC performs an iterative clustering of each selected individual. A maximum number of clusters, $\mathcal{MC}$, must be defined *a priori*. The i^{th} cluster has its own center c_i, but a common radius r_t, in each generation t, is calculated for all clusters by:

$$r_t = \frac{x_{sup} - x_{inf}}{2 \cdot \sqrt[n]{|C_t|}} \tag{2}$$

where $|C_t|$ is the current number of clusters (initially, $|C_t| = \mathcal{MC}$), x_{sup} and x_{inf} are, respectively, the known upper and lower bounds of the domain of variable x, considering that all variables x_i have the same domain.

Whenever a selected individual p_k is *far away* from all centers (a distance above r_t), then a new cluster must be created. Evidently, $\mathcal{MC}$ is a bound value that prevents a unlimited cluster creation, but this is not a problem because the clusters can slip along the search space.

The cluster assimilation is a foreseen step that can be implemented by different techniques. The selected individual p_k and the center c_i, which it belongs to, are participants of the assimilation process by some operation that uses new information to cause some changing in the cluster location. In this work, the cluster assimilation is given by:

$$c_i' = c_i + \alpha \cdot (p_k - c_i) \tag{3}$$

where α is called disorder degree associated with assimilation process. In this application, the center are kept more conservative to new information ($\alpha = 0.05$).

These choices are due to computational requests. Complex clustering algorithms could make ECS a slow solver for high dimensional problems. Considering the euclidean distance calculated for each cluster, for a n-dimensional problem, the IC complexity is about $O(\mathcal{MC} \cdot n)$.

At the end of each generation, the component AM performs the *cooling* of all clusters, i.e., they have their accounting of density, δ_i, reset. Eventually some (or all) clusters can be *re-heated* by selections or become inactive, being eliminated thereafter by AM. A cluster is considered inactive when no selection has occurred in the last generation. This mechanism is used to eliminate clusters that have lost the importance along the generations, allowing that other search areas can be framed. The AM is also evoked whenever a cluster is activated. It starts the component LS, at once, if

$$\delta_i \geq \mathcal{PD} \cdot \frac{\mathcal{NS}}{|C_t|} \tag{4}$$

The pressure of density, $\mathcal{PD}$, allows to control the sensibility of the component AM. The meaning of $\mathcal{PD}$ is the desirable cluster density beyond the normal density, obtained if $\mathcal{NS}$ was equally divided to all clusters. In this application, satisfactory behavior has been obtained setting $\mathcal{NS} = 200$ and $\mathcal{PD} = 2.5$.

The component LS was implemented by a Hooke and Jeeves direct search (HJD) [17]. The HJD is an early 60's method that presents some interesting features: excellent convergence characteristics, low memory storage, and requiring only basic mathematical calculations. The method works by two types of move. At each iteration there is an exploratory move with one discrete step size per coordinate direction. Supposing that the line gathering the first and last points of the exploratory move represents an especially favorable direction, an extrapolation is made along it before the variables are varied again individually. Its efficiency decisively depends on the choice of the initial step sizes $\mathcal{SS}$. In this application, $\mathcal{SS}$ was set to 5% of initial radius.

The Nelder and Mead Simplex (NMS) has been more widely used as a numerical parameter optimization procedure. For few variables the simplex method is known to be robust and reliable. But the main drawback is its cost. Moreover, there are n parameter vectors to be stored. According to the authors, the number of function calls increases approximately as $O(n^{2.11})$, but these numbers were obtained only for few variables ($n \leq 10$) [2]. On the other hand, the HJD is less expensive. Hooke and Jeeves found empirically that the number of function evaluations increase only linearly, i.e., $O(n)$ [17].

3.2 Computational experiments

The ECS was coded in ANSI C and it was run on Intel AMD (1.33 GHz) platform. The population size was varied in $\{10, 30, 100\}$, depending upon the problem size.

The parameter $\mathcal{NC}$ was set to 20 for all test functions. In the first experiment, ECS is compared against two other approaches well-known in the literature: Genocop III [16] and the OptQuest Callable Library (OCL) [18]. Genocop III is the third version of a genetic algorithm designed to search for optimal solutions in optimization problems with real-coded variables and linear and nonlinear constraints. The OCL is a commercial software designed for optimizing complex systems based upon metaheuristic framework known as scatter search [10]. Both approaches were run using the default values that the systems recommend and the results showed in this work were taken from [18].

The results in Table 2 were obtained, in 20 trials, allowing ECS to perform $10,000$ function evaluations, at the same way that Genocop III and OCL are tested. The average of the best solutions found (FS) and the average of function calls (FC) were considered to compare the algorithm performances. The average of execution time in seconds (ET) is only illustrative, since the used platforms are not the same. The values in bold indicate which procedure yields the solution with better objective function value for each problem. Note that ECS has found better solutions in two test functions, while both OCL and Genocop III have better results in one function.

Table 2. Comparison against OCL and GENOCOP-3

		ECS		OCL		GENOCOP III	
Function	var	FS	ET	FS	ET	FS	ET
Ackley	50	0.000	0.181	0.000	16.800	0.000	13.400
Ackley	100	0.000	0.374	0.000	103.600	0.000	46.600
Sphere	100	**0.000**	0.128	2.419	60.300	1114.451	43.700
Griewank	20	0.000	0.123	0.000	3.800	1.076	2.600
Rastrigin	10	1.087	0.036	**0.000**	4.500	1.026	0.900
Rastrigin	20	10.129	0.063	**0.000**	6.300	10.508	2.500
Rosenbrock	6	**0.002**	0.065	5.950	6.300	273.309	0.800
Rosenbrock	8	**0.000**	0.077	0.484	3.200	5.601	0.800
Rosenbrock	20	**0.003**	0.022	5.600	6.900	7.685	2.800
Schwefel	10	118.160	0.042	844.069	1.800	**1.387**	0.800
Schwefel	20	1360.397	0.047	1506.067	2.400	**134.491**	2.100

In the second experiment, ECS is compared against other approach found in literature that works with the same idea of detecting promising search areas: the Continuous Hybrid Algorithm (CHA), briefly described in the introduction. The CHA results were taken from [5], where the authors worked with several n dimensional test functions. The most challenging of them are used for comparison in this work. The results in Table 3 were obtained allowing ECS to perform up to $100,000$ function evaluations in each one of the 20 trials. There is no information about the corresponding CHA bound. The average of the gaps between the solution found and the best known one (GAP) and the average of function calls (FC) were considered to compare the algorithm performances, besides the success rate (SR) obtained. In the **ECS** experiments, the SR reflects the percentage of trials that have reached at least a gap of 0.001. The SR obtained in CHA experiments is not a classical one, according the authors, because it considers the actual landscape of the function at hand [5].

One can observe that ECS seems to better than CHA in all test functions showed in Table 3, except for the *Zakharov*, which ECS has not found the best known solution. It is known that the 2-dimensional *Zakharov*'s function is a monomodal one with the minimum lying at a corner of a wide plain. Nevertheless, there was not found any reason for such poor performance. In function *Shekel*, although ECS have found better gaps, the success rate is not as good as CHA. The values in bold indicate in which aspects ECS was worse than CHA.

Table 3. Comparison against CHA

				ECS			CHA	
Function	var	ET	GAP	FC	SR	GAP	FC	SR
Eason	2	0.002	0.00060	593.5	100	0.001	952.0	100
Goldstein	2	0.001	0.00060	**345.4**	100	0.001	259.0	100
Hartman	6	0.003	0.00000	633.9	100	0.008	930.0	100
Rosenbrock	5	0.007	0.00040	2561.7	100	0.018	3290.0	100
Rosenbrock	10	0.023	0.00005	8979.5	100	0.008	14563.0	83
Rosenbrock	50	0.049	0.00015	32780.6	100	0.005	55356.0	79
Rosenbrock	100	0.286	0.00444	85821.0	80	0.008	124302.0	72
Shekel	4	0.003	0.00007	506.8	75	0.015	635.0	**85**
Zakharov	10	0.004	0.00050	2328.6	100	**1e-6**	4291.0	100
Zakharov	50	0.153	33.75020	100040.6	0	**1e-5**	75520.0	100

Other results obtained by ECS are showed in Table 4. The gap of 0.001 was reached a certain number of times for all these functions. The worst performance was in *Michalewicz* and *Langerman*'s functions (SR about 65%).

Table 4. ECS results for other test functions

Function	var	ET	GAP	FC	SR	Function	var	ET	GAP	FC	SR
Griewank	50	0.053	0.00010	5024.550	100.00	Rastrigin	10	0.100	0.00060	26379.950	100.00
Griewank	100	0.432	0.00000	24344.450	100.00	Rastrigin	20	0.339	0.00078	71952.667	90.00
Langerman	5	0.023	0.00000	5047.684	95.00	Schwefel	20	0.211	0.00035	39987.950	100.00
Langerman	10	0.075	0.00000	17686.692	65.00	Schwefel	30	0.591	0.00029	90853.429	70.00
Michalewicz	5	0.054	0.00035	12869.550	100.00	Michalewicz	10	0.222	0.00038	37671.923	65.00

4 Conclusion

This paper proposes a new way of detecting promising search areas based upon clustering. The approach is called Evolutionary Clustering Search (ECS). The ECS attempts to locate promising search areas by framing them by clusters. Whenever a cluster reaches a certain density, its center is used as start point of some aggressive search strategy.

An ECS application to unconstrained numerical optimization is presented employing a steady-state genetic algorithm, an iterative clustering algorithm and a local search based upon Hooke and Jeeves direct search. Some experiments are presented, showing the competitiveness of the method. The ECS was compared with other approaches, taken from the literature, including the well-known Genocop III and the OptQuest Callable Library.

For further work, it is intended to perform more tests on other bench-mark functions. Moreover, heuristics and distance metrics for discrete search spaces are being studied aiming to build applications in combinatorial optimization.

References

1. Yen, J., Lee, B.: A Simplex Genetic Algorithm Hybrid, In: IEEE International Conference on Evolutionary Computation -ICEC97, (1997)175-180.
2. Nelder, J.A., Mead, R.: A simplex method for function minimization. Computer Journal. (1956) 7(23):308-313.
3. Birru, H.K., Chellapilla, K., Rao, S.S.: Local search operators in fast evolutionary programming. Congress on Evolutionary Computation,(1999)2:1506-1513.
4. Oliveira A.C.M.; Lorena L.A.N. Real-Coded Evolutionary Approaches to Unconstrained Numerical Optimization. Advances in Logic, Artificial Intelligence and Robotics. Jair Minoro Abe and João I. da Silva Filho (Eds). Plêiade, ISBN: 8585795778. (2002)2.
5. Chelouah, R., Siarry, P.: Genetic and Nelder-Mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. Euro. Journal of Operational Research, (2003)148(2):335-348.
6. Jelasity, M., Ortigosa, P., García, I.: UEGO, an Abstract Clustering Technique for Multimodal Global Optimization, Journal of Heuristics (2001)7(3):215-233.
7. Li, J.P., Balazs, M.E., Parks, G.T., Clarkson, P.J.: A species conserving genetic algorithm for multimodal function optimization, Evolutionary Computation, (2002)10(3):207-234.
8. Yager, R.R.: A model of participatory learning. IEEE Trans. on Systems, Man and Cybernetics, (1990)20(5)1229-1234.
9. Silva, L.R.S. Aprendizagem Participativa em Agrupamento Nebuloso de Dados, Dissertation, Faculdade de Engenharia Elétrica e de Computação, Unicamp, Campinas SP, Brasil (2003).
10. Glover, F., Laguna, M., Martí, R.: Fundamentals of scatter search and path relinking. Control and Cybernetics, (2000) 39:653-684.
11. Bersini, H., Dorigo, M., Langerman, S., Seront G., Gambardella, L.M.: Results of the first international contest on evolutionary optimisation - 1st ICEO. In: Proc. IEEE-EC96. (1996)611-615.
12. Holland, J.H. Adaptation in Natural and Artificial Systems *University of Michigan Press*, Ann Arbor, 1975.
13. Digalakis, J., Margaritis, K.: An experimental study of benchmarking functions for Genetic Algorithms. IEEE Systems Transactions,(2000)3810-3815.
14. Goldberg, D.E.: Genetic algorithms in search, optimisation and machine learning. Addison-Wesley, (1989).
15. Eshelman, L.J., Schawer, J.D.: Real-coded genetic algorithms and interval-schemata, In: Foundation of Genetic Algorithms-2, L. Darrell Whitley (Eds.), Morgan Kaufmann Pub. San Mateo (1993) 187-202.
16. Michalewicz, Z.: $GeneticAlgorithms + DataStructures = EvolutionPrograms$. Springer-Verlag, New York (1996).
17. Hooke, R., Jeeves, T.A.: "Direct search" solution of numerical and statistical problems. Journal of the ACM, (1961)8(2):212-229.
18. Laguna, M., Martí, R.: The OptQuest Callable Library In Optimization Software Class Libraries, Stefan Voss and David L. Woodruff (Eds.), Kluwer Academic Pub., (2002)193-218.