Parallel and Distributed Controlled Random Search Algorithms; Case Study Results

Michał Marks[1] and Ewa Niewiadomska-Szynkiewicz[2]

[1] Warsaw University of Technology, Institute of Control and Computation Engineering, Warsaw, Poland, email: mmarks@elka.pw.edu.pl
[2] Research Academic Computer Network (NASK), Warsaw, Poland, e-mail: ewan@nask.pl

Abstract. CRS (Controlled Random Search) algorithms for global optimization are considered. The main objective is to present the advantages of developing the parallel and distributed random search algorithms to search for the global solution. A practical example, application of parallel CRS2, CRS4, CRS6, CRSI algorithms and distributed CRS2 algorithm to calculate the optimal prices of products that are sold in the market, are presented. In the final part of the paper the results of numerical experiments performed on the historical data are described and discussed.

1 Introduction

Random search methods have proved to be a robust way to find the global solution of optimization problems defined as $\min f(x)$ for $x \in X \subset \Re^n, X = \{x : a_i \leq x_i \leq b_i\}, i = 1, \ldots, n$ with nonlinear, non-differentiable and also non-convex performances f, [2, 5, 9]. This class of methods is based on global exploration of the domain and localization of search. The algorithms are easy to implement, do not require any properties of the function being optimized and use the constant computer memory storage. They can be applied with success to solving problems with noisy objective function values. All these features result in the random search being widely used in applied science and engineering. Unfortunately, random search algorithms are usually slow because they require a large number of iterations and function evaluations to calculate the global solution, so acceleration is worthwhile. The direction, which should bring benefits, is parallel or distributed implementation where the whole task is partitioned between several processors or machines.

In this paper the robustness and efficiency of parallel and distributed versions of controlled random search algorithms are discussed. The numerical results of their application to optimal price calculations are presented. The paper is organized as follows. First it provides a short overview of several versions of controlled random search methods. Second parallel and distributed implementations are described. Finally, the price management optimization problem is formulated and the results of optimizations performed for historical data concerned with sales in the supermarket are discussed.

2 Controlled Random Search - Algorithms Overview

The Controlled Random Search methods (CRS) are population set based random search algorithms. The basic random search consists of three main steps: 1) generate the initial set of points $P \subset X$, 2) transform the population P, 3) check the assumed stopping condition. In principle, CRS methods were designed as a combination of a local optimization algorithm with a global search procedure. All CRS algorithms start from the creation of the set of points P (much more than $n+1$ points in n-dimensional space), selected randomly from the domain X. Then the best x_L (i.e., that of the minimal value of the performance index) and the worst x_H (i.e., that of the maximal value of the performance index) points are determined. Next, new trial point x_Q is calculated. Then finally, if this point is admissible and better than x_H it replaces the worst point x_H in the set P, so the initial population of trial points is modified.

Several versions of CRS methods related to different strategies of new trial points calculations are available. They were developed by Price [7], Ali and Storey [1], Mohan and Shanker [3]. Three versions CRS1 – CRS3 use the approach with nonlinear simplex formulation and transformation. In CRS2 the nonlinear simplex is formed with the best point x_L and n points $(x_2, \ldots, x_{n+1})$ randomly chosen from P. Afterwards, the centroid x_G of points $x_L, x_2, \ldots, x_n$ is determined. The next trial point x_Q is calculated, $x_Q = 2x_G - x_{n+1}$. Then, if the point x_Q is admissible and better (i.e. $f(x_Q) \leq f(x_H)$), x_Q replaces the worst point x_H in the set P. Otherwise, a new simplex is randomly formed and so on until the stop criterion is met, i.e. $f(x_H) - f(x_L) \leq \varepsilon$, where ε denotes the assumed accuracy.

The CRS3 algorithm is a combination of the CRS2 procedure with the local optimization procedure based on the Downhill Simplex Algorithm of Nelder and Mead described in [4]. The local algorithm is switched when a newly generated point in CRS2 fell within the bottom one-tenth of the ordered array P. After completing the local search the global search is continued. The CRS3 method tends to speed the convergence of the algorithm with respect to CRS2. The local optimization procedure operates only on a small part of set P and thus has a minimal effect on the global search performance of the CRS2 phase. The local procedure can operate at any stage of CRS3. It is triggered automaticly but it can be modified to permit the user to switch the local procedure in or out as required.

Another local techniques are introduced in versions CRS4 and CRS5. CRS4 evaluates the value of performance f in m points from the β distribution, assuming the mean value equal to the current best point x_L and standard deviation equal to $x_H - x_L$. CRS5 uses a gradient local search starting from x_L. The quadratic interpolation (1) is used in CRSI to generate the new trial point x^{trial}:

$$x_i{}^{trial} = \frac{1}{2}\frac{(x_{1_i}^2 - x_{2_i}^2)f(x_L) + (x_{2_i}^2 - x_{L_i}^2)f(x_1) + (x_{L_i}^2 - x_{1_i}^2)f(x_2)}{(x_{1_i} - x_{2_i})f(x_L) + (x_{2_i} - x_{L_i})f(x_1) + (x_{L_i} - x_{1_i})f(x_2)}, \quad i = 1 \ldots n \qquad (1)$$

where x_1 and x_2 are two points randomly selected from the set P, i denotes the i-th coordinate of each point. CRS6 uses quadratic interpolation (1) and random generation from the β distribution as in CRS4.

3 Parallel and Distributed Implementation of CRS Algorithms

3.1 Parallel Version

The CRS algorithms are not only quite simple and efficient, but also easily adaptable to a parallel environment. The parallel versions of CRS2 and CRS6 were developed and tested. The goal was to speed up the calculations. The numerical experiments were carried out on the parallel machine with the help of the POSIX threads library. Several calculation threads were executed. All threads operating in parallel transformed the same, globally available population of points P. The current best point x_L was global for all calculation threads, while the worst points x_H were local, one for each thread. The additional thread (the main thread) was responsible for calculation threads initialization, synchronization, communication and results presentation. The calculation structure is presented in Figure 1.

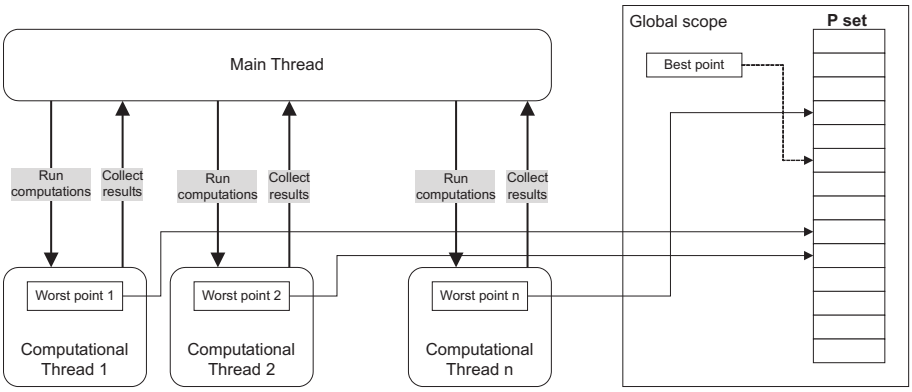

Figure 1. CRS - parallel implementation.

3.2 Distributed Version

The distributed version of the CRS2 method was developed to perform calculations in computer networks. The goal was to improve the accuracy of the solution. The numerical experiments were carried out on the network of Sun workstations. The implementation was based on MPI (*Message Passing Interface*) library. Master-slave architecture was applied. In the distributed versions of all CRS methods considered, several independent instances (calculation processes) of algorithms were executed, each on a separate processor. The additional process (coordinator) was responsible for calculation processes initialization, inter-processes communication and calculations termination. In contrast with parallel implementation each calculation process transformed its individual population of points P. After each assumed number of iterations set by the user the calculation process sents its local best point to the coordinator and took the best point currently available in the coordinator process. Each process stopped the calculations after stopping condition was met and sent its results together with the adequate message to the coordinator process. The communication between coordinator and calculation processes was asynchronous. The distributed implementation is presented in Figure 2.

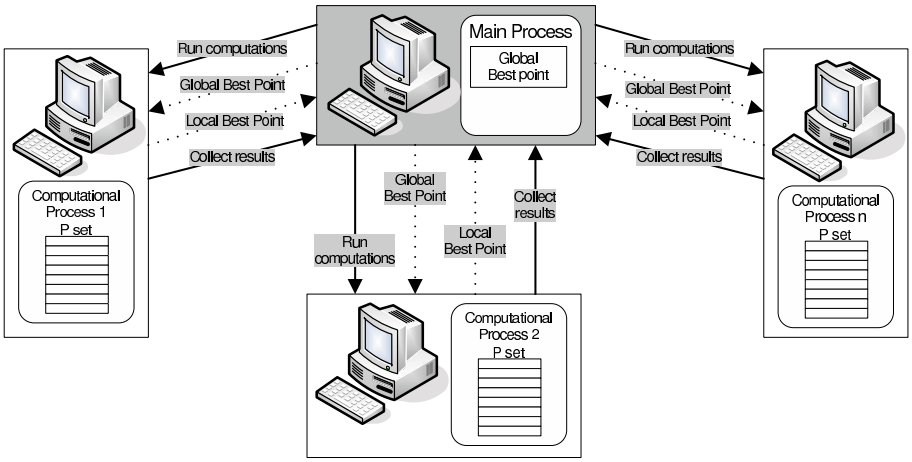

Figure 2. CRS - distributed implementation.

4 Case Study – Price Management Problem

The goal of price management is to determine the optimal prices or a pricing strategy that satisfies the firm's objectives. Let us consider n products that are sold in the market. Assume that $x = [x_1, x_2, \ldots, x_n]$ denotes a vector of prices of n products; x_i is the price of the i-th product, and $q = [q_1, q_2, \ldots, q_n]$ denotes a vector of expected sales within the considered period; q_i is the sale of the i-th product. The relationship between prices and sales is called *price response function* $S(x)$, defined as follows $q_i = S_i(x)$.

We can formulate a price management problem. The goal is to calculate the optimal prices for products that are sold in the market to maximize the long-term profit equal to revenue minus cost

$$\max_{x=[x_1,\ldots,x_n]} PR^T(x) \tag{2}$$

where PR^T is the total expected profit within the considered period;
$PR^T(x) = \sum_{i=1}^{n} PR_i(x)$ and PR_i profit calculated for product i and n number of products exist (corresponding to n price decisions x_i).

The value of profit strongly depends on the market response function. Several linear and nonlinear market response models can be found in the literature, [6, 8]. All these models describe market response for the price of the i-th product.

The popular measure of the impact of price on sales is the price elasticity. The elasticity refers to the relation of a percentage change in sales volume to the percentage change in price. Sales of a product depend on its own price and the prices of other products. The first dependency is measured by the direct elasticity, the second by the cross elasticity.

The elasticity is considered in multiplicative price response function that represents the sales q as a nonlinear function of price x:

$$S_i(x) = \alpha_i \prod_{j=1}^{n} x_j^{\beta^{ij}} \tag{3}$$

where x_j denotes the price of product j, α_i is the scaling factor for sales of product i, β^{ij} is the elasticity of sales of product i with respect to the price of product j (β^{ii} is referred to as the direct elasticity and β^{ij}, $i \neq j$ is the cross elasticity). The curve defined by the function x^β is convex for $\beta < 0$ or $\beta > 1$ and concave for $0 < \beta < 1$.

Taking into account linear cost function and VAT the performance index to be maximized in (2) is given as:

$$PR^T(x) = \sum_{i=1}^{n} \Big(\frac{x_i}{1+v_i} - d_i \Big) S_i(x) \tag{4}$$

where v_i and d_i are given constants corresponding to the market entities of VAT and cost per product, S_i are expected sales of the product i within the considered period defined in (3), assuming that prices of all products are fixed over this period.

The following constraints for price, sale and cash of each product and for total sale and cash can be considered:

$$x_i^{min} \leq x_i \leq x_i^{max} \quad i = 1, \ldots, n \tag{5}$$

$$S_i^{min} \leq S_i(x) \leq S_i^{max} \quad i = 1, \ldots, n \tag{6}$$

$$C_i^{min} \leq x_i S_i(x) \leq C_i^{max} \quad i = 1, \ldots, n \tag{7}$$

$$TS^{min} \leq \sum_{i=1}^{n} S_i(x) \leq TS^{max} \tag{8}$$

$$TC^{min} \leq \sum_{i=1}^{n} x_i S_i(x) \leq TC^{max} \tag{9}$$

In the above constraints x_i^{min} and x_i^{max} denote minimal and maximal prices of product i, C_i^{min}, C_i^{max} minimal and maximal cash, TS^{min}, TS^{max} minimal and maximal total sale, TC^{min}, TC^{max} minimal and maximal total cash.

In practice, usually prices of only some products are changed at anyone time. The following constraint restricts the number of prices, which can be modified

$$\sum_{i=1}^{n} \frac{\gamma (x_i - x_i^0)^2}{1 + \gamma (x_i - x_i^0)^2} \leq w \tag{10}$$

where γ and w are assumed parameters, x_i^0 the current price of the product i.

5 Numerical Results

Multiple numerical experiments were performed for several sets of historical data obtained from an IT company in Manchester, containing various groups of products offered in supermarkets. The goal of optimization was to maximize the global profit as described in (4) subject to constraints (5) – (10), with the sales response market model given by (3). Controlled random search algorithms – CRS2, CRS4, CRS6 and CRSI (sequential, parallel and distributed versions) were used to calculate the optimal prices. The size of the considered optimization problems was restricted by the available sets of historical data.

5.1 Sequential Version

The weakness of all CRS methods is the way in which the constraints of a type $g_j(x) \leq 0$ are handled. The infeasible points are simply rejected from further consideration. The suggested approach is to account for these constraints in the objective function using simple penalty terms for constraints violation. The reformulation of optimization problem $\min f(x)$ for $x \in X \subset \Re^n, X = \{x : a_i \leq x_i \leq b_i\}, i = 1, \ldots, n, g_i(x) \leq 0, j = 1, \ldots, m$, is as follows:

$$\min_{x \in X}[f(x) + \Phi(x)], \quad \Phi(x) = \mu \sum_{j=1}^{m} \max\Big(0, g_i(x)\Big)^p \tag{11}$$

where μ and p are parameters.

The optimization results of price management problem (2), considering two approaches to infeasible points, and various sizes of the initial population P of points are presented in Tables 1, 2 and Figures 3, 4. The assumed population size was defined as $\|P\| = NP \cdot (n + 1)$, where n denotes the problem dimension and the NP parameter is defined by the user. The calculations were performed for $NP = 10, 15, 20$ and 25. The assumed accuracy of optimal the point calculation was 1E-4.

Figures 3, 4 and Table 1 present the average results when prices for fifteen products were calculated using CRS2, CRS4, CRS6 and CRSI methods, obtained during 5 runs of each optimization algorithm. The bar diagrams in Figure 3 and Figure 4 show the average number of performance function (4) evaluations and average calculation time.

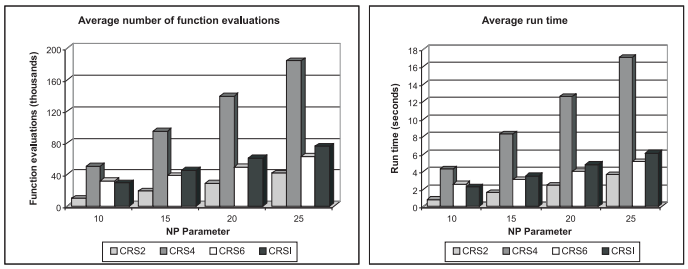

Figure 3. Number of function evaluations and run time with infeasible points discarding.

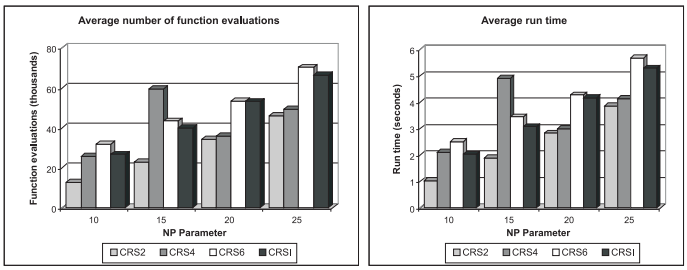

Figure 4. Number of function evaluations and run time with penalty for constraints violation.

The results obtained for modified objective function (11), Figure 4, are compared with those obtained for the standard approach that discards infeasible, Figure 3. The values of the objective function (4) obtained for four methods and two approaches to constraints implementations are collected in Table 1.

The presented results indicate that the trade off between the size of the initial population that influences the time of calculation, and the solution accuracy will be necessary. The suggested values of NP parameter are $10 - 15$. The CRS2 algorithm is very fast but only gives an approximate solution, even in the case when the penalty function is used. The CRS4 method provides better results with respect to CRS2. In our tests the best results were obtained by CRS6 and CRSI methods but the time required to compute a solution was longer than the CRS2 method.

Table 1. Function and penalty values with different approach to infeasible points.

NP	CRS2		CRS4		CRS6		CRSI	
	Solution	Penalty	Solution	Penalty	Solution	Penalty	Solution	Penalty
Discarding infeasible points								
10	1212.899	0.00000	1241.029	0.00000	1241.271	0.00000	1241.271	0.00000
15	1216.683	0.00000	1241.130	0.00000	1241.272	0.00000	1241.272	0.00000
20	1215.797	0.00000	1241.142	0.00000	1241.272	0.00000	1241.272	0.00000
25	1215.951	0.00000	1241.119	0.00000	1241.272	0.00000	1241.272	0.00000
Penalty for constraints violation								
10	1229.209	0.65939	1235.638	0.64042	1241.272	0.00002	1241.272	0.00003
15	1232.280	1.35452	1238.941	0.21826	1241.272	0.00001	1241.272	0.00003
20	1236.550	1.93125	1235.862	1.07984	1241.272	0.00002	1241.272	0.00002
25	1237.452	1.12425	1236.818	0.98825	1241.272	0.00004	1241.272	0.00002

The numerical results of three series of experiments (prices of 15, 31 and 53 products) and CRS2, CRS6, CRSI methods are presented in Table 2. All obtained results were compared to the best.

Table 2. Comparison of the fastest and the most accurate methods.

Task size	The best solution	Algorithm	Function evaluations	Solution	Penalty	Run Time (in seconds)	Relative error
15	1241.27	CRS2	24002	1235.41	0.63	1.98	0.47%
		CRSI	45180	1241.27	0	3.51	0.00%
		CRS6	39392	1241.27	0	3.16	0.00%
31	830.747	CRS2	69562	805.76	0	23.39	3.01%
		CRSI	155452	830.75	0	45.00	0.00%
		CRS6	122298	830.71	0	33.64	0.00%
53	544.649	CRS2	76169	526.03	0	23.85	3.42%
		CRSI	389933	544.65	0	101.07	0.00%
		CRS6	285849	544.65	0	75.99	0.00%

As a conclusion the following strategy is proposed: in cases when accuracy of the solution is the crucial the CRSI or CRS6 methods with the discarding of infeasible points are suggested; when it is crucial that the problem is solved quickly the CRS2 method with the penalty function should be used.

5.2 Parallel Version

The multithread versions of CRS2 and CRS6 algorithms were applied to solve the optimization problem (2) for 15 products. Many calculations were performed using both methods and different number of threads. The numerical experiments were carried out on a Sun Fire V440 equipped with four processors. The goal was to speed up calculations. The question was how parallelization influence computing time. The results of the experiments are collected in Table 3 (CRS2 method) and Table 4 (CRS6 method). The tables present the average solution, number of function evaluations and calculation time for 5 runs of each optimization algorithm.

Table 3. Parallel version of CRS2 with penalty for constraints violation.

Threads	Function evaluations	Solution	Penalty	Run time
1	35264	1236.557	1.024	2.93
2	34238	1236.529	1.532	1.47
3	35442	1235.396	1.507	1.06
4	34844	1236.061	1.588	0.82
5	38543	1237.319	1.703	0.92
6	48936	1236.764	1.856	1.16
7	66681	1237.217	1.815	1.55
8	79864	1237.247	0.763	1.86

Table 4. Parallel version of CRS6 with infeasible points discarded.

Threads	Function evaluations	Solution	Penalty	Run time
1	59693	1241.27	0	4.74
2	61091	1241.27	0	2.49
3	62375	1241.27	0	1.73
4	61274	1241.27	0	1.37
5	61346	1241.27	0	1.35
6	60321	1241.27	0	1.33
7	61036	1241.27	0	1.32
8	60836	1241.27	0	1.34

The results presented in Figure 5 show that the number of threads influences the computation time: the bigger the number of threads (but less or equal the number of processors) - the better results. The obtained values of performance were similar to the sequential version.

The acceleration factors for the 15-dimension price management problem and four threads were calculated for CRS2 and CRS6 algorithms:

$$\frac{1\_thread\_time_{CRS2}}{4\_threads\_time_{CRS2}} = \frac{2.93}{0.82} = 3.57, \qquad \frac{1\_thread\_time_{CRS6}}{4\_threads\_time_{CRS6}} = \frac{4.74}{1.37} = 3.46$$

The values of acceleration factors show the effectiveness of a parallel implementation, both for CRS2 and CRS6 algorithms.

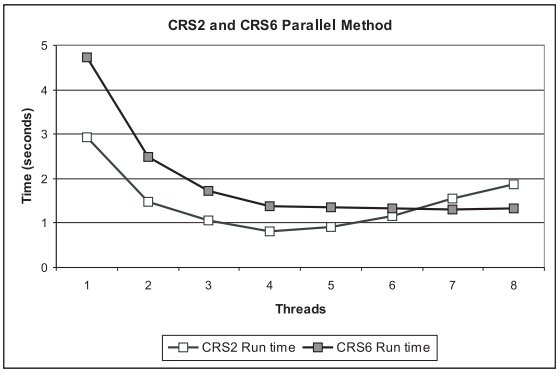

Figure 5. Run times for CRS2 and CRS6 parallel method.

The numerical results of four series of experiments (prices of 15, 31, 53 and 76 products) are presented in Table 5. The solutions provided by CRS2 and CRS6 methods are compared with the best.

Table 5. Comparison of the fastest and the most accurate methods in multithreads mode.

Task size	The best solution	Algorithm	Function evaluations	Solution	Penalty	Run Time (in seconds)	Relative error
15	1241.27	CRS2	34844	1236.061	1.588	0.82	0.42%
		CRS6	61274	1241.271	0	1.37	0.00%
31	830.747	CRS2	99541	807.809	0	10.11	2.76%
		CRS6	202326	830.747	0	15.77	0.00%
53	544.649	CRS2	130232	525.634	0	12.31	3.49%
		CRS6	536782	544.649	0	36.40	0.00%
76	1898.658	CRS2	149499	1833.475	0	29.33	3.43%
		CRS6	1350278	1898.656	0	180.95	0.00%

5.3 Distributed Version

The distributed version of the CRS2 algorithm was applied to solve the optimization problem (2) for 15, 31 and 53 products. The goal of distributed implementation, as described in section 3.2, was to improve the accuracy of the solution by sequential CRS2 rather than speed up the calculations. The numerical experiments were carried out on a network of five Sun workstations. The average values of performance (4) provided by parallel and distributed CRS2 implementations and obtained during 5 runs of each task are compared in Table 6.

The numerical results show that the distributed version of CRS2 provides better solutions with respect to sequential (see Table 2) and parallel realizations. The time required to obtain a solution was similar for both distributed and sequential implementations and longer than in the parallel version. The number of processors influences the effectiveness of optimization method, that is presented in Table 7. The bigger the number of processors - the better the results.

Table 6. Comparison of distributed and parallel CRS2 implementation.

Task size	The best solution	Parallel implementation		Distributed implementation	
		Solution	Relative error	Solution	Relative error
15	1241.27	1236.061	0.42%	1239.527	0.14%
31	830.747	807.809	2.76%	812.743	2.17%
53	544.649	525.634	3.49%	530.738	2.55%

Table 7. Solution accuracy.

Distributed implementation			The best solution
Computational Processes	Solution	Relative error	
1	1233.128	0.66%	1241.27
3	1237.772	0.28%	
5	1240.441	0.07%	

6 Conclusions

In this paper the application of the random search CRS methods to optimal pricing calculations was discussed. Three versions of CRS algorithms were compared: sequential, parallel and distributed. The conclusion to be drawn is that parallel/distributed calculations provides the better solution for the optimization process and speeds up the calculations.

Bibliography

[1] M.M. Ali and C. Storey. Modified controlled random search algorithms. *International Journal of Computer Mathematics*, 54:229–235, 1995.

[2] M.M. Ali, A. Törn, and S. Vittanen. A numerical comparison of some modified controlled random search algorithms. *Journal of Global Optimization*, 11:377–385, 1997.

[3] C. Mohan and K. Shanker. A controlled random search technique for global optimization using quadratic approximation. *Asia-Pacific Journal of Operational Research*, 11:93–101, 1994.

[4] J.A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[5] E. Niewiadomska-Szynkiewicz. Parallel global optimization for optimal flood control. *Acta Geophysica Polonica*, 47:93–109, 1999.

[6] E. Niewiadomska-Szynkiewicz. Application of evolutionary strategy to price management problem. In *Proceedings of VIII Conference on Evolution Algorithms and Global Optimization, KAEiOG*, 2005.

[7] W.L. Price. Global optimization by controlled random search. *Journal of Optimization Theory and Applications*, 40:333–348, 1983.

[8] H. Simon. *Price management*. North-Holland, 1989.

[9] A. Törn and A. Žilinskas. *Global optimization*. Springer-Verlag, 1989.