THE PSEUDO-BOOLEAN OPTIMIZATION APPROACH TO FORM THE N-VERSION SOFTWARE STRUCTURE[*]

I V Kovalev, D I Kovalev, P V Zelenkov, A A Voroshilova
Siberian State Aerospace University named after Academician M.F. Reshetnev
31 “KrasnoyarskiyRabochiy” prospect, Krasnoyarsk, 660037, Russia.

E-mail:zelenkov@sibsau.ru

Abstract. The problem of developing an optimal structure of N-version software system presents a kind of very complex optimization problem. This causes the use of deterministic optimization methods inappropriate for solving the stated problem. In this view, exploiting heuristic strategies looks more rational. In the field of pseudo-Boolean optimization theory, the so called method of varied probabilities (MVP) has been developed to solve problems with a large dimensionality. Some additional modifications of MVP have been made to solve the problem of N-version systems design. Those algorithms take into account the discovered specific features of the objective function. The practical experiments have shown the advantage of using these algorithm modifications because of reducing a search space.

1. Introduction

The use of N-version programming approach turns out to be effective, since the system is constructed out of several parallel executed versions of some software module [1,2]. Those versions are written to meet the same specification but by different programmers. Where, the writing process of each version of concrete software module in any way must not intersect with or depend on another version code writing.

The problem of developing the optimal structure of an N-version software system (NVS) is the following [3]: to choose a set of software modules, so as to provide the highest reliability for the system subject to the budget constraint. Since a description of any possible system configuration is made through such the positioning of its components [4,5], we can say that an observed problem has the binary essence [6]. Moreover, the existing theory of pseudo-Boolean functions and their optimization contains strong tools for solving problems of this kind [7]. And that fact makes the use of binarization algorithms more affordable.

The process of a problem binarization consists in setting relationships between the system states and the binary space elements. In the case of our system model, we need to determine some Boolean vector the elements of which will characterize the system structure. Each element of such the Boolean vector will signify either presence or absence of corresponding system component [8].

To derive an optimal dependability solution by means of an systematic, the exhaustive comparison algorithm would mean that all potential system configurations have to be tentatively generated, checked for the fulfillment of the side conditions and processed to

[*]The research was done with the financial support of the Ministry of Education and Science of the Russian Federation in accordance with the agreement № 14.574.21.0041, unique identifier RFMEFI57414X0041

compute the corresponding overall system reliability. This usually would cause a computing complexity that is untractable even for the most modern high speed computers: if, e.g., we consider a system consisting 64 modules, all of which are to be triplicated, thereby selecting each of the module versions from 5 different candidate modules, we would have to consider $[5!/3!\cdot 2!]^{64} = 10^{64}$ different system configurations! Assuming e.g. 1 nsec for processing each system configuration (of course, a value by far too optimistic!), the resulting 10^{55} sec of needed computation time would exceed the estimated age of the universe of about 10^{17} sec by many orders of magnitude! Therefore, here only stochastic search methods appear possible to provide, in a heuristic way, an optimal solution [9-12].

2. The Method of Varying Probabilities

In the field of pseudo-Boolean optimization theory, the so called *method of varied probabilities* has been developed to solve complicated problems, especially those ones with a large dimensionality [6-8]. The method of variable probabilities (MVP) presents a family of heuristic algorithms based on the common scheme: in order to find an extremal solution of a pseudo-Boolean optimization problem, a *probability vector* of dimensionality of sought solution vector is formed. Each component of the probability vector presents a probability of assigning a *one* value to the correspondent component of a Boolean vector. In the terms of developing NVS structure, it looks like a probability to include a version-candidate into the system structure.

The initial values of the probability vector components describe a situation when every software version has the equal probability to be included into the system structure. Then, at a computational phase, random decisions are generated according to the probability distribution specified by means of the probability vector. Each time the objective function is calculated in several random points, the values of the probability vector components are updated, so changing a probability distribution form. The way of changing these values defines a separate algorithms of MVP scheme. The common approach for updating a probability vector can be characterized by the rule: the better result received with a one-valued binary vector component the bigger probability is assigned for it to get the value of one in the final solution.

This scheme can be augmented whether by some special methods for updating the probability vector or through involving the peculiar procedures of generating random solutions at a computational phase of an algorithm. This paper discusses the two methods for updating the probability vector (ARSA and Modified ARSA ver. 1) and the two procedures of generating random solutions (the independent generation of random points and the generation of non-zero solutions) giving thus as a result four different realizations (algorithms) of MVP.

The Adaptive Random Search Algorithm (ARSA) plays a role of the background for the rest of the algorithms of MVP scheme [10]. Initially, ARSA has been developed for the problem of pattern recognition to select an informative subsystem of attributes. The main disadvantage of this algorithm is a potential problem of updating values of probability vector components. Namely, in some cases it is possible to get the values of intermediate solutions which do not let the probability vector components to be changed. To correct the defect, the modification of ARSA has been developed (Modified ARSA ver.1). The statistical data of applying the modified version of ARSA display the better behavior of the algorithm when solving problems of developing a structure of NVS.

Next, applying to the stated optimization problem, ARSA doesn't provide a technique of avoiding zero-solutions when solving the problem of designing NVS structure. To protect an algorithm against spending both computational and time resources for calculating the objective function values in the points of this kind, the particular technique of generating random non-zero solutions has been developed. This technique is utilized in the MVP based algorithm named NVS MVP (mentioning the strict field of using the algorithm).

Making use of both of the mentioned enhancements gave a great raise in the efficiency of applying the MVP based algorithms to the problem of NVS structure development. The statistical results presented in the final part of the paper show it. Different algorithms have been tasted on the same optimization problem with the same quantity of objective function calls.

The objective function of the presented optimization problem has several specific features which can assist to reduce a search domain, thus allowing to decrease the searching time. The objective function as a function of the whole system reliability represents the product of reliabilities of separate software modules. Consequently, when a reliability of any of the modules is equal to zero the overall system reliability turns into zero value also. Physically, it represents a case when there are no versions chosen for (at least) some of the software modules. The implication vector components corresponding to such the software modules will be assigned zeroes as well. Obviously, it is necessary to avoid computing the objective function in such the points.

The number of system structures having at least one software module without versions-candidates assigned can be determined as the difference of the number of all the possible structures and the quantity of the structures which provide every software module with at least one candidate, i.e. $N_0 = N_{all} - N_{R>0}$. The number of all possible structures is determined through the dimensionality of an implication vector *n* as follows $N_{all} = 2^n$. The second intermediate value is found basing on the multiplication principle from combinatorics as a number of all possible structures with software module combinations each without one of them (that with no versions assigned). Formally, it is described in the following way: $N_{R>0} = \prod_{i=1}^{I}(2^{k_i} - 1)$, where *I* is the number of software modules, k_i represents a number of versions for the *i*-th software module.

Then, the final expression determining a sought value looks like this:

$$N_0 = 2^n - \prod_{i=1}^{I}(2^{k_i} - 1).$$

The value of this expression depends on an overall number of candidates (a dimensionality of the optimization problem), a number of software modules *I* and the numbers of versions for each of the software module (k_i, $i = \overline{1,I}$). In general case, this expression takes grand values counting up to $0.9N_{all}$, i.e. 90% of all the possible solutions. This means in this case that in order to find a solution it is sufficient to search through only 10% of the definitional domain of the objective function.

Unfortunately, there is the other side of the question making this result not so optimistic. Namely, for the problems of large dimensionalities reducing the search domain to 10% means diminishing the dimensionality of a problem by very small value. For instance, for a test problem of dimensionality n=117, avoiding all the null-valued points lowered the problem dimensionality only down to $n_{R>0} = 116$.

Nevertheless, exploiting this feature of the objective function has given satisfactory results when applying the algorithms of the method of varied probabilities (MVP). The modification of the MVP with the ability of avoiding null-valued points called NVS MVP has its own way of generating random points at iterational steps of the algorithms. In NVS MPV, random points are generated so that to provide each software module with at least one version.

At every iterational step, the whole implication vector generated is concerned as consisting of parts each describing the structure of a separate software module. Thus indeed, random vector generating consists of generating of random structures of modules. This approach allows having only non-zero solutions in result.

3. The Random Search of Boundary Points

Another stochastic algorithm to optimize the structure of NVS is the algorithm of *random search of boundary points* [8]. It is based on the proved fact that a solution of the stated optimization problem is a so called *boundary point*. Or in terms of binary space topology, a point neighboring to the set of infeasible solutions. Such a point describes a system structure which can not be updated through including a software versions additionally without violating the resource conditions, i.e. no version can not be added to a system structure of this kind paying attention to restrictions. The algorithm of random search of boundary points constitutes a generating of multiple boundary solutions and comparing the objective functions values in them.

The constraint in this optimization problem partitions the whole binary space into two domains – the domain of solution satisfying the constraint function and a set of points not satisfying to the constraint. It is shown that these domains represent the connected sets and that a solution of correspondent optimization problem is a point neighboring to the set of infeasible solutions. This kind of solution can be called a *boundary point*.

Basing on the results stated above, it is clear that it is sufficient to search among only boundary points in order to find the best value of the objective function. Thus, the problem of finding a best solution becomes a problem of an exhaustive search on the boundary points set.

The following is the algorithm of generating boundary point for the problem of developing the optimal structure of NVS (Fig. 1).

Different boundary points can be reached using this algorithm when different combinations of ways to choose i at the second step of the algorithm will be followed.

Hence, the algorithm of searching boundary points will have the following scheme.

1. The initializing step: i=0.
2. Determine a boundary point $\mathbf{X}_{\mathrm{b}i}$ (b – as an index means “boundary”).
3. Calculate the objective function value $F_i=F(\mathbf{X}_{\mathrm{b}i})$.
4. If the stopping condition is satisfied go to p. 5, otherwise $i=i+1$ and go to p. 2.
5. The solution is $F^* = \max_i F_i$.

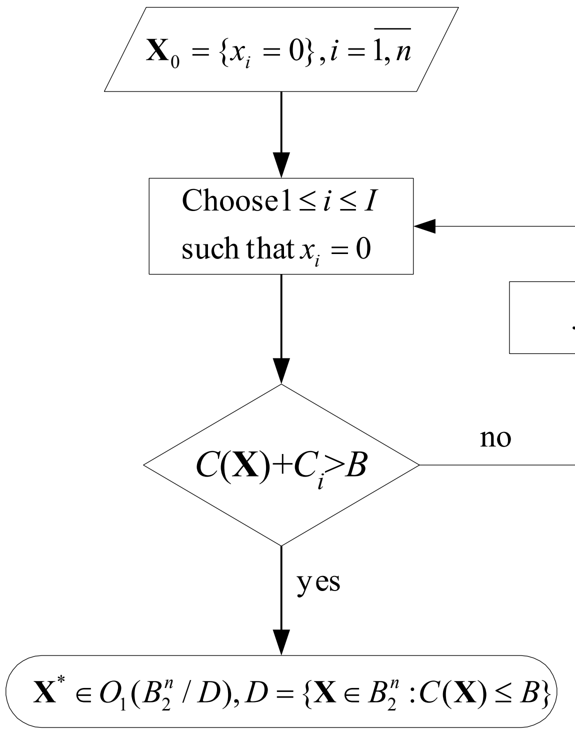
$\mathbf{X}_0 = \{x_i = 0\}, i = \overline{1,n}$
Choose $1 \le i \le I$ such that $x_i = 0$
$x_i=1$
$C(\mathbf{X})+C_i>B$
no
yes
$\mathbf{X}^* \in O_1(B_2^n / D), D = \{\mathbf{X} \in B_2^n : C(\mathbf{X}) \le B\}$

Figure 1. Generating a boundary point.

Separate variations of the *algorithm of boundary points search* may differ from each other in a stopping condition and in ways of reaching boundary when generating boundary points. For the optimization problems of high complexity it is more rational to use stochastic version of the algorithm when boundary points are reached in a random way and this process is executed repeatedly.

4. The Comparison of the Random Optimization Procedures

Concluding the paper, let us cite the comparative data of the computational results for different random optimization procedures. To gather such the information, the presented algorithms have been used to solve the test NVS structure optimization problems [9-12]. The efficiency of the random search algorithms has been judged by the values of the objective and constraining functions.

The problem of dimensionality 117 has been chosen as the test problem, i.e. the developed software system included 117 software versions. It's worth mentioning that every of the random search algorithms needed approximately same period of time for calculating under equal conditions. That's why the time has not been set as an efficiency characteristic.

Table 1 contains the computational results of algorithms testing [13,14]. The best searching capabilities have been revealed with the use of NVS MVP algorithm and the algorithm of boundary points search. The latter displayed the highest stability of the solutions found, although using NVS MVP it is sometimes possible to find more reliable system configurations.

Table 1. The results of random search algorithms working.

	Budget constraint B	Number of iteraions	The random search algorithms			
			NVS MVP		Random search of boundary points	
			$R(X^*)$	$C(X^*)$	$R(X^*)$	$C(X^*)$
1.	00	15000	*0.7872*	789	*0.8074*	796
			0.7916	791	*0.7998*	797
			0.7907	791	*0.8177*	794
		30000	*0.8136*	786	*0.8318*	794
			0.8054	775	*0.8331*	797
			0.8118	784	*0.8377*	798
2.	00	15000	*0.9040*	850	*0.9149*	899
			0.9207	896	*0.9164*	899
			0.9039	887	*0.9148*	898
		30000	*0.9076*	867	*0.9192*	897
			0.9082	875	*0.9167*	897
			0.9155	890	*0.9177*	892
3.	000	15000	*0.9701*	995	*0.9622*	993
			0.9523	986	*0.9609*	998
			0.9546	989	*0.9635*	998
		30000	*0.9651*	994	*0.9652*	997
			0.9554	988	*0.9661*	995
			0.9712	997	*0.9631*	996

5. Conclusion

The problem of structuring an N-version software system is specified by the binary character, what made it plausible to apply the methods of pseudo-Boolean optimization. Within the limits of the discrete optimization a set of methods and algorithms has been proposed. The search capabilities of each of the algorithms realized have been investigated by solving the test problems. It was shown that the modification of the method of varying probabilities for NVS MVP together with the algorithm of boundary points search provide the best searching capabilities concerning the time efficiency and the solution quality.

References

[1] Laprie J-C et al 1987 Hardware- and Software-fault tolerance: definition and analysis of architectural solutions *Proceedings of the IEEE* pp. 116-121.

[2] Scheer S, Maier T 1997 Towards Dependable Software Requirement Specifications *In: Daniel, P. (ed.) Proceedings of SAFECOMP, New York (1997)*

[3] Kovalev I 1994 Optimization-based design of software of the spacecraft control systems *In: "Modelling, Measurement and Control, B*" vol. 56 №1 pp. 29-34.

[4] Kovalev I V, Engel E A, Tsarev R Ju 2007 Programmatic support of the analysis of cluster structures of failure-resistant information systems *Automatic Documentation and Mathematical Linguistics* vol.41**3** pp. 89.

[5] Avizienis A 1995 The methdology of N-version programming *In: Software fault tolerance* (edited by M.R. Lyu, Wiley) pp. 23-47.

[6] Antamoshkin A, Schwefel H P, Torn A, Yin G and Zilinskas A 1993 System Analysis, Design and Optimization. *Ofset Press, Krasnoyarsk* 312 p.

[7] Antamoshkin A N et al 2013 Random Search Algorithm for the p-Median Problem *Informatica* **3(37)** pp. 127–140.

[8] Kazakovtsev L, Stanimirovic P, Osinga I, Gudima M and Antamoshkin A 2014 Algorithms for location problems based on angular distances *Advances in Operations Research* vol. 2014. Articale ID 701267. 12 pages.
 - http://www.hindawi.com/journals/aor/raa/701267/

[9] Kovalev I. 1998 Optimization problems when realizing the spacecrafts control *In: Advances in Modeling and Analysis, C* vol. 52 **1-2** pp. 62-70.

[10] Kovalev I V, Dgioeva N N, Slobodin M Ju 2004 The mathematical system model for the problem of multi-version software design. *Proceedings of Modelling and Simulation, MS'2004 AMSE International Conference on Modelling and Simulation* (MS'2004. AMSE, French Research Council, CNRS, Rhone-Alpes Region, Hospitals of Lyon. Lyon-Villeurbanne).

[11] Kovalev I, Grosspietsch K-E 2000 Deriving the Optimal Structure of N-version Software under Resource Requirements and Cost *Timing Constraints* (Proc. Euromicro' 2000, Maastricht, 2000, IEEE CS Press) pp. 200-207.

[12] Kovalev I et al 2013 The Minimization of Inter-Module Interface for the Achievement of Reliability of Multi-Version Software *Proceedings of the 2013 International Conference on Systems, Control and Informatics (SCI 2013)* (Venice, Italy, September 28-30 2013) pp. 186-188.

[13] Kovalev I 1995 Optimal base software composition of the spacecrafts control system *In: "Advances in Modeling and Analysis, C"* (AMSE Periodicals) vol. 47 **3** pp. 17-26.

[14] Kovalev I V et al 2002 Fault-tolerant software architecture creation model based on reliability evaluation *Advanced in Modeling & Analysis* (Journal of AMSE Periodicals) vol. 48 **3-4** pp. 31-43.