Metaheuristics optimisation techniques for software structure optimisation

J. Balicki & Z. Kitowski
The Navy Academy of Poland

Abstract

In this paper, a study of two modern heuristics techniques is carried out for finding optimal structure of software modules. The tabu search algorithm and an evolutionary algorithm are compared. An optimisation task of program module allocation is treated as a benchmark problem for an assessment of the algorithm capabilities in the structural optimisation of complex systems, which are models of smart structures. Finally, some numerical results are presented.

1 Introduction

Designing of several structures for complex systems, modelled and simulated by computer means, requires efficient optimisation techniques. But the common formulated optimisation problems for above field - incorporating the smart structure design - are recognized as NP-hard task of combinatorial computations. So, there is no chance for preparing efficient algorithms for large instances of designed structures. But, we can apply approximation algorithms for finding good, suboptimal solutions, if standard method failed.

In this paper, a study of two modern heuristics techniques is carried out. The first approach is called a tabu search, because special areas are forbidden during the seeking in a search space of all possible structures. These fields are tabu in any time, and the algorithm usually explores the permitted structures. The second approach is called an evolutionary search, because it includes genetic algorithms, evolutionary algorithms, evolutionary strategies, and genetic programming for finding optimal structures of complex systems. This set of techniques assumes that the next generations are better suited to the environment than the previous populations. A selection, a crossover and a mutation can obtain the better structures, if individuals with better fitness are preferred.
To comparing of tabu search and evolutionary search some combinatorial optimisation problems are formulated for finding optimal structures. We chose the structures of software in distributed computer systems, but this study can be carried out for any smart structure designing. For assumed optimisation problems two the best representative techniques are discussed. Theoretical aspects and results of numerical experiments are presented, and final theses are deduced.

2 Genetic algorithm for decision-making aid

The standard genetic algorithm GA is applied for solving several optimisation problems with a scalar criterion with small modifications [9]. But, optimisation of complex structures is often related with using some scalar criteria for structure evaluation. These criteria represent time, cost or safe measures and are in conflict each to other. So, a compromise has to be found to respect different preferences.

Schaffer [11] considered the GA for solving multiobjective optimisation problems by the vector evaluated genetic algorithm VEGA. The VEGA divides the population on \( N \) subpopulations, where \( N \) is the number of criteria. For each \( n \)th subpopulation the criterion \( F_n \) is a fitness function. After partial selection, crossoverring, and mutation are carried out for whole population. This selection discriminates of Pareto solutions situated in the interior of the Pareto frontier. Indeed, mainly lexicographic solutions are preferred.

Fourmann considered selection with using hierarchical tournaments, where two randomly chosen solutions are compared, and a hierarchical solution is a winner in this competition, and it is included to a mating pool of potential parent [5]. A selection probability is calculated for the most important goal. A random choice is carried out twice according to the roulette rule. But similarly to VEGA approach, hierarchical tournaments for target preferences set a priori support the solution migration towards lexicographical solutions.

\[ F_2 \]
\[ F_1 \]

Figure 1. Ranking of individuals from the current population in the evaluation space
To avoid the discrimination of the interior Pareto solutions Goldberg introduced the ranking system for non-dominated individuals [12]. If there are some feasible solutions in a population, then the Pareto-optimal individuals are sought, and they get the rank 1 (Fig. 1). Then they are temporarily eliminated from the population. From reduced population the new Pareto-optimal trajectories are found and get the rank 2. This procedure with increasing of the rank is repeated until the set of feasible solutions will be exhausted. That is why, all non-dominated solutions have the same rank and the same fitness to reproduction. They have the same chance to produce their offspring.

If x is non-feasible, it gets the minimal fitness function value $f(x)=1$. If x is feasible, then the fitness function value is calculated, as below:

$$f(x) = L - r(x) + 1,$$

where

$r(x)$ - the rank of a feasible solution $x$,
$L$ – the size of a population (number of individuals).

The genetic algorithm for solving multicriteria optimisation problems can be presented in Fig. 2.

```
BEGIN
  $t=0$, set the size of population $L$
  randomly generate initial population $P(t)$
  calculate ranks $r(x)$ and fitnesses $f(x), x \in P(t)$
  $\text{finish} := \text{FALSE}$
  WHILE NOT $\text{finish}$ DO
    BEGIN /* new population */
      $t := t+1$, $P(t) := \emptyset$
      calculate selection probabilities $p_x(x), x \in P(t-1)$
      FOR $L/2$ DO
        BEGIN /* reproduction cycle */
          ♦ proportional selection of a potential parent pair $(a,b)$ from the population $P(t-1)$
          ♦ simple crossovering of a parent pair $(a,b)$ with assumed crossover probability $p_c$
          ♦ bit mutation of an offspring pair $(a',b')$ with assumed mutation probability $p_m$
          END
        $P(t) := P(t) \cup (a',b')$
      END
    calculate ranks $r(x)$ and fitnesses $f(x), x \in P(t)$
    IF ($P(t)$ converges OR $t \geq T_{\text{max}}$) THEN $\text{finish} := \text{TRUE}$
  END
BEGIN
```

Figure 2. Genetic algorithm for solving a wide class of multicriteria optimisation problems
3 Tabu search algorithm

Tabu search is the meta-heuristic approach which vitally important applications in engineering, economics and science have been the predominant focus of research throughout the past three decades and still the focus of many academic works [6, 10]. Tabu search algorithms have been applied for solving several optimisation problems in scheduling, computer-aided design, quadratic assignment, training and designing of neural networks. Moreover, the best results have been obtained by development of tabu search in telecommunication call routing, volume discount acquisition in production, and vehicle routing. Its good capabilities have been confirmed during solving standard optimisation problems such as graph partitioning, graph colouring, clique partitioning. So tabu search can be treated as a general combinatorial optimisation techniques for using in zero-one programming, nonconvex nonlinear programming, bilevel programming and general mixed integer optimisation.

Tabu search uses memory structures by reference to four principal dimensions, consisting of recency, frequency, quality and influence. Tabu search algorithm inherits from a simple descent method an idea of a neighbourhood $N(x^{\text{now}})$ of a current solution $x^{\text{now}}$. From this neighbourhood we can choose the next solution to a search trajectory $x^{\text{now}}$. The accepted alternative should have the best value of an objective function among the current neighbourhood. But, the descent method terminates its searching, when the chosen candidate is worse than the best one from the searching trajectory.

In the tabu search based on the short term memory, a basic neighbourhood $N(x^{\text{now}})$ of a current solution may be reduced to a considered neighbourhood $M(x^{\text{now}})$ because of the maintaining a selective history of the states encountered during the search. Some solutions, which were visited during the given last term, are excluded from the basic neighbourhood according to the tabo classification of movements. If any solutions performs aspiration criterion, then it can be included to the considered neighbourhood, only.

A recency-based memory keeps track of solutions attributes that have changed the recent past. Selected attributes that occur in solutions recently visited are labelled tabu-active, and solutions that contain tabu-active attributes or particular combinations of these attributes became tabu, too. This prevents certain solutions from the recent part of a trajectory from belonging to a considered neighbourhood and hence from being revisited. Furthermore, other solutions with tabu-active attributes are similarly prevented being visited. While the tabu classification strictly refers to solutions that are forbidden to be visited, we also often refer to moves that lead to such solutions as being tabu.

In the tabu search based on the long term memory, a considered neighbourhood may also be expanded to include solutions not ordinary found in a basic neighbourhood. During long exhausting searching, there is an opportunity to count frequency measures of selected attributes. Often performed movements should be forbidden to take a chance rarely performed movements after long observation. Frequency measures of selected attributes are respected in the selecting function of a next solution from a current neighbourhood.
The quality memory refers to the ability to differentiate the merit of solutions visited during search. A memory can be used to identify elements that are common to good solutions or to paths that lead to such solutions. A quality becomes a foundation for incentive-based learning, where inducements are provided to reinforce actions that lead to good solutions. Penalties are provided to discourage actions that lead to poor solutions. The flexibility of these memory allows to guide in multi-objective space, where search directions are determined by more than one function.

The influence memory considers the impact of the chosen solutions, not only on quality but also on structure. Information about the influence of trajectory solutions incorporates additional level of learning. The assessment and exploiting of influence by a memory is an important feature of tabu search algorithms.

4 Model for cost minimization

A reasonable program module allocation in multiple computer systems can decrease the total time of program execution by taking advantage of the specific efficiencies of some computers or advantage of load computer states [2]. In a distributed computer system, another way for the minimization of the total time of program execution is to change the computers for module processing. So, the supercomputer with a power floating point unit can be dedicated for performing program modules with several numerical procedures, and the database server is suitable for processing program modules with great number of queries to a database [3].

Figure 3. An example of hierarchical program consisted of 14 modules assigned to 6 computers
The standard problem of program module allocations is a question how to find the allocation of program modules to minimize the program execution cost [1]. An objective of another optimisation problem can be the time of program performing, too. The other measure is the amount of computer resource reserved.

Fig. 3 shows an example of distributed hierarchical program in a computer network with 6 computers. The program was divided on 14 modules $m_1, m_2, ..., m_{14}$. Module $m_1$ is assigned to a computer with the module $m_2$. It is reasonable solution, if between above pairs of modules great number of interactions is required.

To reduce the communication time between modules, they should be allocated to a single computer. From the other hand, to reduce the module processing time each module should by assigned to the computer, where its the processing time is the lowest. There is a conflict between minimization of the communication time and the module processing time.

Let us assume that the module $m_v$ can be executed on several sorts of computers taken from the set $\Pi = \{ \pi_1, ..., \pi_j, ..., \pi_J \}$. Computers are able to operate in the fixed nodes included to the set $W = \{ w_1, ..., w_i, ..., w_J \}$, only.

Let us introduce the following decision variables:

$$ x^\pi_{ij} = \begin{cases} 1 & \text{if } \pi_j \text{ is assigned to the } w_i, \; i = 1, J, \; j = 1, J. \\ 0 & \text{in the other case,} \end{cases} $$

$$ x^m_{vi} = \begin{cases} 1 & \text{if } m_v \text{ is assigned to } w_i, \; v = 1, V, \; i = 1, I. \\ 0 & \text{in the other case,} \end{cases} $$

Now, the following decision vector can write the allocation of operations to computers:

$$ x = [x^m_{11}, ..., x^m_{1i}, ..., x^m_{IJ}, ..., x^m_{ji}, ..., x^m_{ji}, ..., x^m_{ji}, ..., x^m_{ji}, ... , x^\pi_{IJ}, ..., x^\pi_{ji}, ..., x^\pi_{ji}, ..., x^\pi_{ji}, ... ]^T. \; (2) $$

Above basic model can be easy modified to formulate the problem of minimization the cost of data processing in a distributed computer systems by reasonably allocation of program modules to computers.

5 Multiobjective optimisation problem

The first criterion used for an allocation evaluation is the cost of parallel program performing, which can be calculated, as below:

$$ F_1(x) = \sum_{j=1}^{J} \sum_{v=1}^{V} \sum_{i=1}^{I} t_{ij} x^m_{vi} x^\pi_{ij} + \sum_{v=1}^{V} \sum_{u=1}^{V} \sum_{i_1=1}^{I} \sum_{i_2=1}^{I} \tau_{uv} x^m_{vi_1} x^m_{ui_2}, \; x \in \mathcal{B}^M, \; (3) $$

where

- $t_{ij}$ - the cost of performing the module $m_v$ by the computer $\pi_j$,
- $\tau_{ij}$ - the cost of communications between the module $m_v$ and the module $m_u$,
- $\mathcal{B}$ - the set $\{0, 1\}$. 

An assessment performance of distributed computer system can be calculated, as follows:

\[ F_2(x^P) = \sum_{i=1}^{I} \sum_{j=1}^{J} p_j x_{ij}^P. \tag{4} \]

where \( p_j \) represents the performance of computer \( \pi_j \), which can be measured by the Linpack benchmark [Mflops] [3].

Let consider the case of multiobjective optimisation problem \((\mathcal{X}, F, P)\) for finding the Pareto-optimal solutions, which are some allocations of program modules and computer types in a distributed computer system. In this problem some denotations are used, as follows:

1) \( \mathcal{X} \) - a feasible solutions set

\[ \mathcal{X} = \{ x \in \mathbb{R}^{I(V+J)} \mid \sum_{i=1}^{I} x_{iI}^P = 1, \nu = \sum_{i=1}^{I} \nu_i V; \sum_{j=1}^{J} x_{ij}^P = 1, i = \sum_{i=1}^{I} i; \}. \]

2) \( F \) - a vector quality criterion

\[ F : \mathcal{X} \rightarrow \mathbb{R}^2, \]

\[ F(x) = [F_1(x), F_2(x)]^T \text{ for } x \in \mathcal{X} \]

where \( F_1(x) \) is calculated by (3),

\( F_2(x) \) is calculated by (4).

3) \( P \) - the Pareto relationship [12]

6 Multiobjective genetic algorithm

For solving multiobjective optimisation problem (5) with the considered three criteria an evolutionary algorithm can be used [8]. Let us consider the following numerical example. There are 3 nodes. Processing costs of modules on computers are assessed by a matrix

\[ T = \begin{bmatrix} 2 & 5 & 6 & 4 & 5 \\ 4 & 6 & 9 & 3 & 8 \end{bmatrix} \]

There are 2 computer sorts with performances 1 GFlops and 2 GFlops. If all modules are assigned to the first computer sort, then the global cost of processing is 22 [CU-cost units], and on the computer of a second sort it is 30 [CU]. So, the computer of a second sort is more expensive in performing, but it has a higher performance then the first one. The search space has \( 2^{31} \) binary alternatives, where 1944 solutions are admissible. The centralized assignments are eliminated.
The genetic algorithm with ranking procedures has different population queues for several trials from the same initial population. Above directional and probabilistic search can be analysed, where we focus on some representative trials or on average trial. Eventually, an optimistic trial or a pessimistic trial can be discussed.

The following parameters were taken $p_c=0.6$, $p_m=0.05$ and $L=20$. After 200 generations. Genetic algorithm found 32 evaluations, and, what is more important, it shows results close to Pareto frontier. An evaluation set has 52 elements. The greater number of the lack evaluations are from an anti-ideal point. In [1], a level of convergence to Pareto frontier is shown for different sizes of populations. The best performance was done for the largest populations according to the schema theorem [4, 7].

7 Evolutionary algorithm versus genetic algorithm

Evolutionary algorithms develop genetic algorithms for solving optimisation problems by another chromosome representation, more complex operators, and a specific knowledge related with the optimisation problem [9]. An overview of evolutionary algorithms for multiobjective optimisation problems is presented by Fonseca and Fleming [4]. GA can be used for solving the wide class of problems, and EA is rather focused on the special case of task, only. But usually, results obtained from EA are much better.

A logical scheme is similar to GA, but initial population is constructed to individuals satisfy the constraints

$$\sum_{i=1}^{I} x_{vi}^m = 1, \forall v = 1, V; \sum_{j=1}^{J} x_{ij}^\pi = 1, \forall i = 1, I$$

by introducing integer representation, as follows:

$$X = (X_1^m, ..., X_v^m, ..., X_1^\pi, ..., X_i^\pi, ..., X_j^\pi),$$

where $X_v^m = i$ for $x_{vi}^m = 1$ and $X_i^\pi = j$ for $x_{ij}^\pi = 1$

![Figure 4. Minimization of the average level of convergence to Pareto frontier by EA and GA](image-url)
Moreover, $0 < X^m_v \leq I$ and $0 < X^p_i \leq J$. A simple crossover operator is used, but a bit mutation is substituted by the random exchange of integer value by another from a feasible discrete set. If solution is non-feasible, then the penalty is calculated. The fitness for a non-feasible solution is equal to the difference between maximal penalty in population. Adding maximal penalty in population to a term $-r(x)+L+1$ creates the fitness for a feasible solution. This evolutionary algorithm gives much better results than GA (Fig. 4). For the GA the following parameters were taken $p_c=0.6, p_m=0.05$ and $L=20$.

8 Evolutionary approaches versus tabu search

An evolutionary algorithm $EA$ with an evolutionary strategy $ES$ are powerful tools for solving multicriteria optimisation problems. Now, we compare their calculation capabilities to tabu search approaches. Four variants of tabu search are considered with main kinds of memories: short-term $TS_1$, long term $TS_2$, quality $TS_3$ and influence $TS_4$. Fig. 5 shows minimization of the average level of convergence to Pareto frontier by above algorithms. The average level of convergence decreases with the number of choice function evaluations. The choice function is a fitness function in evolutionary algorithms or a selection function in tabu procedures.

![Figure 5. Minimization of the average level of convergence to Pareto frontier by evolutionary and tabu search algorithms](image)

The average level of convergence to Pareto frontier decreases faster for tabu search algorithms than evolutionary algorithms for this instance of a multiobjective optimisation problem. Also this general conclusion for this numerical example is representative for considered set of instances, but, for some instances the comparison results are reversed and we can not suggest using tabu search approach before evolutionary algorithms.
9 Concluding remarks

The tabu search algorithm and an evolutionary algorithm are compared according to the average level of convergence to Pareto frontier, and the preferences for tabu search approach can be emphasised. An optimisation task of program module allocation is treated as a benchmark problem for an assessment of the algorithm capabilities in the structural optimisation of complex systems, which are models of smart structures. Our future works will focused on the development tabu search algorithms for computer network design.

References