A Heuristic Optimization Algorithm in System Structure Optimization

Derong Zhu
Mechanical and Electronic Engineering Department, Luoyang Institute of Science and Technology, Luoyang, China
Email: zhuderongnetwork@126.com

***Abstract*—Systems structure optimization is a multi-objective or a combinatorial optimization problem, which should consider the comprehensive influence of cost, time and resource etc. This paper firstly describes the system structure optimization problem and gives the mathematical model. Then a heuristic optimization algorithm is proposed to analyze the execution time, the success rate and the cost in system structure optimization. And the efficiency of the heuristic optimization algorithm in searching for the optimal system structure solution is compared with genetic algorithm and particle swarm optimization method. At last, the experiment simulation verifies the validity and the correctness of the proposed algorithm.**

***Index Terms*—system structure optimization, cost, mathematical model, algorithm**

I. Introduction

In the past 20 years, a number of researchers have developed structural techniques and processes. These studies focused on efforts to address structural issues such as description, analysis, validation and evaluation [1-3]. The ultimate goal of performing structure analysis and evaluation is to optimize program designs. However, structure is inherently multi-dimensional and must be optimized using a comprehensive and dynamic method, guided by various optimal goals. Therefore, determining an optimal program or methodology in structure design remains a significant challenge [4].

Numerous studies on optimization of system and structure design have been conducted. Considering this approach, Kim and Hidalgo [5] helped an airline company introduce new planes to meet the dynamic requirements of consumers. Meanwhile, Wolf [6] developed a model of a dispatching operation force to an objective area, with response time and cost as the objective function. He used cooperative optimization and investigated the optimization problems of marines designing at the structural level. The use of the simulation-based method [7, 8] to improve structure design can be regarded as two distinct but closely related optimization techniques. Shen et al. [9] developed a mathematical model for the optimal allocation of large complex system and demonstrated the rational allocation of resources and capability to optimize the performance of the overall system structure. Despite the contribution of these studies to system structure optimization (system structure optimization) research, the effects of changes on business processes or activities during system configuration are rarely reported.

Available system is considered as resources to execute an activity. Cost, time, and success rate are among the factors affecting system structure and thus, must be considered in solving a system structure optimization problem. Previous studies neither considered these factors nor examined them comparatively.

The system structure optimization [10-13] problem should be viewed as a multi-objective or a combinatorial optimization problem. Software modularity also comes as a necessity when complex optimization problems in complex materials need to be tackled with an array of different methods and techniques. The computer software available for analyzing and interpreting the experimental data is a heterogeneous mixture of often incompatible programs. With regard to the details of the algorithms they use, most programs are poorly documented. At the same time, the complexity of some programs has grown so dramatically that even the most dedicated researcher must spend months. To meet these goals, algorithms were moved from the time, cost, resource balance, etc.

Many studies have been reported, with time, cost, and resource balance viewed as multiple optimization objectives. These studies use genetic algorithms [14] and particle swarm optimization algorithm [15-17] to solve comprehensive optimization under multi-objective conditions. In this paper, a heuristic optimization algorithm is proposed to solve the system structure optimization problem. The proposed algorithm mainly considered three main factors: success rate, cost, and execution time of activities.

The rest of the paper is organized as follows. Section II gives the problem description. The algorithm of system structure optimization is analyzed in Section III. The experiment simulation verifies the validity and the correctness of the proposed algorithm in Section IV. Section V gives the conclusions.

II. Problem Descriptions

Systems structure optimization is concerned with the structure of a system to complete all activities. This section presents the development of a mathematical model to quantify the relationship among all decision variables of the system structure optimization problem.

The system structure optimization problem mainly

doi:10.4304/jsw.9.3.743-747

includes three kinds of constraints: constraints on the sequence of activities that have to be executed, constraints on the success rate, and constraints on the cost of changes in the system structure.

The mathematical model can be given by the combination of three constraints:

$$\text{Minimize } TC = \frac{[C(A)+R(A)]*T_\varepsilon(A)}{V_\delta(A)} \tag{1}$$

$$T(A) < T_{max} \tag{2}$$

$$V(A) \geq V_{min} \tag{3}$$

$$C(A) + R(A) < C_{\max} \tag{4}$$

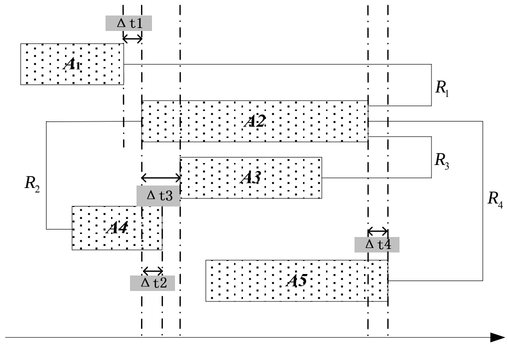
In (2), *T(A)* is the time that the task is completed and T_{max} is the upper limit of time. In (3), *V(A)* is the success rate that the task is completed and $V_{\min}$ is the lower limit of success rate. In (4), the sum of the cost of changes in the physical structure of the system *C(A)* and that in the logical structure of the system *R(A)* must be less than the upper limit. In (1), $T_\varepsilon(A)=\varepsilon * T(A)$, $V_\delta(A)=\delta * V(A)$, and ε are the coefficients of *T(A)* to the total cost *TC*, and δ is the coefficient of *V(A)* to the total cost *TC*. The objective function *TC* of the optimal system structure program has three components: the cost of changes in the system structure [*C*(*A*)+*R*(*A*)], which involve changes in both the physical structure (*C*(*A*)) and the logical structure (*R*(*A*)); the total execution time [$T_\varepsilon(A)$]; and the success rate [$V_\delta(A)$]. Each member in an optional system set can be deployed to perform an activity and has its own execution time and success rate (the execution time and success rate can be specific values or a certain distribution of the probability distribution function).

III. Algorithm of the System Structure Optimization

The algorithm of the system structure optimization problem can be divided into the following: execution time, success rate, and cost of changes.

A. Computation of the Execution Time

Fig. 1 shows the relationship between the execution of activities, which can be divided into four types. First, activity A_2 can only begin if activity A_1 ends (R_1); the time difference between the start time of A_2 and the end time of A_1 is Δt1. Second, A_4 can only end if A_2 begins (R_2); the time difference between the start time of A_2 and the end time of A_4 is Δt2. Third, A_3 can only begin if A_2 begins (R_3); the time difference between the start times of A_2 and A_3 is Δt3. Finally, A_5 can only end if A_2 ends (R_4); the time difference between the end times of A_2 and A_5 is Δt4.



Time line of task executing process

Relationship	*Execution time*
$R_1(A_1)=\{A_2\}$	$T(A_1)+T(A_2)+\Delta t1$
$R_2(A_2)=\{A_4\}$	$T(A_2)+T(A_4)-\Delta t2$
$R_3(A_2)=\{A_3\}$	$Max(T(A_3),T(A_2)+\Delta t3)$
$R_4(A_2)=\{A_5\}$	$Max(T(A_2)+\Delta t4,T(A_5))$

Figure 1. Four relationship types of activity execution

For the relationships R_1, R_2, R_3, and R_4, the minimum total execution time of two activities is shown in the right place in Fig. 1. Generally, activities related to A_i can be divided into four types: $R_1(A_i)$, $R_2(A_i)$, $R_3(A_i)$, and $R_4(A_i)$. The task completion time can be calculated based on relationship type between activities. For convenience, we assume that Δt = Δt1 = Δt2 = Δt3 = Δt4, where Δt is a known variable.

The calculation of the execution time of the task is presented in Algorithm 1. The first activity to be executed is A_1, with the corresponding start time $T0$. The completion time of all activities is then calculated. Finally, the completion times of all activities are compared to derive the updated time, where A_{next} is the next activity to calculate, and $T_{A_{next}}$ is the execution time of A_{next}.

Algorithm 1: Compute the execution time

function execute time（）
//Input: relationship of activities and value of Δ t
//Output: the latest end time in all the activities *T(A)*
find first activity *A*1 to execute
$A_{next} = A_1$
set $A' = A$
while (collection $A' \neq \varnothing$)
{
    $T_s(A_{next})=T_0, T_e(A_{next})=T_0+T_{A_{next}}$
    $A' = A' - A_{next}$
    set sub collection $A_R=R_1(A_{next})\cup R_2(A_{next})\cup R_3(A_{next})\cup R_4(A_{next})$
    for each *a* in A_R //calculate each end time $Te(a)$;
        if $a \in R_1(A_{next})$ then $Te(a)=T(A_{next})+T(a)+\Delta t$ Add the result into *Te*;
        if $a \in R_2(A_{next})$ then $Te(a)=T(A_{next})+T(a)-\Delta t$ Add the result into *Te*;
        if $a \in R_3(A_{next})$ then $Te(a)=Max(T(a),T(A_{next})+\Delta t)$ Add the result into *Te*;
        if $a \in R_4(A_{next})$ then $Te(a)=Max(T(a)+\Delta t,T(A_{next}))$ Add the result into *Te*;
        traverse each activity $a \in A_R$
        $A_{next} = a$
}
search the maximum value $T_{\max}$ in *Te* ;
$T(A)=T0+T_{\max}$

B. Computation of the Success Rate

The success rate of a task is related to the structure of the activity relationship. The relationship between activities can be divided into two types: “OR” and “AND”. OR describes a relationship in which the task can be completed successfully even if only one activity is completed successfully. AND describes a relationship in which the task can be completed successfully only if all activities are completed successfully. Activities with AND or OR relationships can be divided into “OR activity groups” or “AND activity groups.” $Type(G_x)$ represents an OR activity group or an AND activity group. A member of one group can be an activity, an OR activity group, or an AND activity group. For instance, if A_2 and A_1 have an OR relationship, then A_2 and A_1 belong to the OR activity group $G_1 = \{A_1, A_2\}$, $Type(G_1) = OR$. If the activity group G_1 and A_3 have an AND relationship, then G_1 and A_3 combine into a new AND activity group $G_2 = \{G_1, A_3\}$, $Type(G_2) = AND$. The success rate of a group is calculated using the following methods:

(1) If A, B, C form an “AND activity group”, the composite success rate is $V(A)*V(B)*V(C)$.

(2) If A, B, C form an “OR activity group”, then the success rate is $1-(1-V(A))*(1-V(B))*(1-V(C))$.

(3) If a task with the determined relationship of success rate “AND” and “OR” is known, the method described above can be used to determine the suitable formula for computing the success rate of a task.

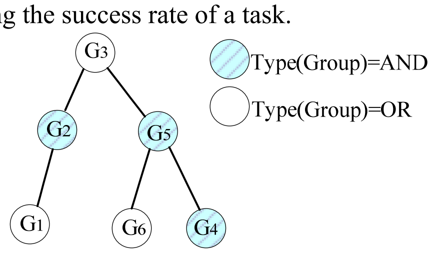


Figure 2. Relationship tree of the activity groups

To obtain the formula for the success rate, we define a tree composed by all activity groups. A group can be an AND activity group or an OR activity group. A group that contains a smaller group is considered the father of the smaller group. Similarly, all relationships in all groups can be represented by a tree. Fig. 2 is an example of a relationship tree of the activity groups. The group at the bottom of the tree is the group that contains no smaller group or that which is composed of activities. Algorithm 2 shows the method of calculating the success rate of a task. From bottom to top, the success rate of each group can be calculated.

Algorithm 2: Compute the success rate

function compute success rate()
//Input: relationships in terms with success rate;
//Output: the calculation formula of task;
if relationships of activities have conflicts
    return false ;
else
build the relation tree RT;
find Group sets $G_{bottom} = \{G_x, G_y, \cdots, G_z\}$ in the bottom of RT;
for each G_k in G_{bottom} {
    if ($Type(G_k) = AND$)
        $V(G_k) = \prod_{k=k_1}^{K} V(A_k)$
    else
        $V(G_k) = 1 - \prod_{k=1}^{K} (1 - V(A_k))$
}
find the father of G; $G_{father} = \{G_a, G_b, \cdots, G_c\}$
while (G_{father} !=NULL){
compute the success rate of G_k in G_{father} ;
    find the father of G_{father} , $G_{grandfather}$;
    $G_{father} = G_{grandfather}$;
}
return $V(G_{father})$;
end function

C. Computation of the Cost of Changes in the System Structure

Algorithm 3 illustrates the method of calculating the cost of changes in the system structure. λ indicates the influence parameters from the information exchange relationship to the structural distance, and the value is specified by experience. *M(W)* is the correlation matrix corresponding to the original structure, *M(W´)* is the correlation matrix corresponding to the new structure, and *TC* represents the total cost of changes in the system structure.

Algorithm 3: Compute cost of changes in systems’structure

function cost()
    compute the cost C1;
    compute the cost C2;
    return C=C1+C2;
end function
else
function ComputeLogicalCost()
load the correlation matrixes *M(W)* and *M(W´)*;
    $D = M(W') - M(W)$ // compute the difference of two matrixes
    $d = \sum |d_i| \mid d_i < 0$ //compute the structure distance *d*, where d_i is the item of matix *D*;
    compute the influence parameter of information exchange relationship to structure distance λ;
    $R = \lambda \times d$ // compute the changes of systems’logical structure *R*:
    return R
end function
else
function ComputeCost()
    return *TC*=*C*+*R*;
end function

IV. Experiment Simulations

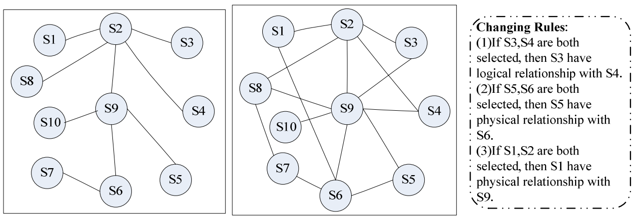
The experiments include the assumption that the execution time of the activities deployed to the available system is exponentially distributed, with an average execution time that is equal to those indicated in Table I for each available system. Table II shows the relationships of the activities in terms of execution time and success rate. The system structure and the assumed changing rules are shown in Fig. 3.

TABLE I
PARAMETERS OF SYSTEM ACTIVITIES

A_i	S_k	$E(T(A_i))$	$V(A_i)$	A_i	S_k	$E(T(A_i))$	$V(A_i)$	A_i	S_k	$E(T(A_i))$	$V(A_i)$
1	1	2	0.96	2	4	6	0.93	3	4	4	0.90
1	2	3	0.95	2	5	7	0.94	3	5	4	0.91
1	3	3	0.98	2	6	5	0.94	3	6	3	0.94
1	4	2.8	0.88	2	7	7	0.82	3	9	5	0.83
1	5	4	0.91	2	8	6	0.87	3	8	3	0.81
4	2	5	0.92	5	4	3	0.92	6	1	5	0.90
4	3	4	0.97	5	5	2	0.93	6	3	6	0.87
4	4	5	0.75	5	7	3	0.78	6	4	5	0.78
4	6	3	0.85	5	8	3	0.85	6	6	4	0.85
4	5	4	0.75	5	9	4	0.73	6	7	5	0.79
7	1	7	0.88	8	7	5	0.78	9	7	3	0.81
7	2	6	0.96	8	8	6	0.86	9	8	4	0.92
7	4	6	0.95	8	6	5	0.75	9	10	3	0.91
7	5	7	0.98	8	9	4	0.88	9	5	5	0.88
10	1	4	0.77	—	—	——	——	—	—	——	——
10	4	5	0.57	—	—	——	——	—	—	——	——
10	6	6	0.88	—	—	——	——	—	—	——	——
10	7	4	0.79	—	—	——	——	—	—	——	——

TABLE II
RELATIONSHIPS OF ACTIVITIES IN TERMS OF EXECUTION TIME AND SUCCESS RATE

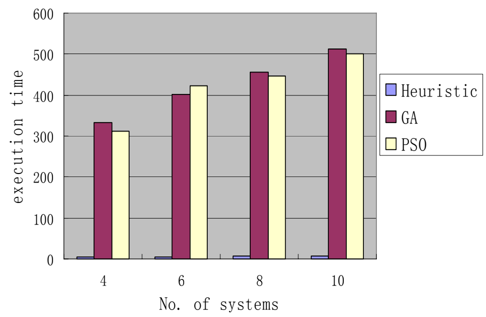
No. of Activities	*R*1	*R*2	*R*3	*R*4	*R*AND	*R*OR
4	R1(A1)={A2} R1(A3)={A4}	NULL	R3(A2)={A3}	R4(A2)={A4}	G1={A1,A2}	G2={G2,A3,A4}
6	R1(A1)={A2} R1(A3)={A4}	R2(A6)={A5}	R3(A2)={A3} R3(A4)={A5}	R4(A2)={A4}	G1={A1,A2} G3={G1,G2,A5,A6}	G2={ A3,A4}
8	R1(A1)={A2} R1(A3)={A4} R1(A3)={A4,A8}	R2(A5)={A6}	R3(A2)={A3} R3(A4)={A5} R3(A6)={A7}	R4(A2)={A4}	G1={A1,A2} G3={G1,G2,A5,A6}	G2={ A3,A4} G4={G3,A7,A8}
10	R1(A1)={A2} R1(A3)={A4} R1(A3)={A4,A8}	R2(A5)={A6} R2(A10)={A9}	R3(A2)={A3} R3(A4)={A5,A6} R3(A6)={A7,A9}	R4(A2)={A4}	G1={A1,A2} G3={G1,G2,A5,A6} G5={G4,G6}	G2={ A3,A4} G4={G3,A7,A8} G6={A9,A10}



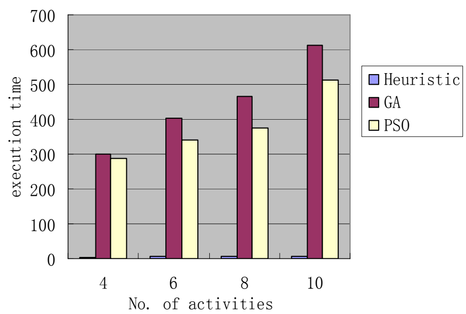
(a)systems' Physical relationship (b)systems' logical relationship

Figure 3. Systems' structure and their changing rules

All algorithms were tested on the same computer platform with fixed scenarios and number of iterations. The experiments are classified as follows: (1) when the number of activities is 6, the system that may assigned to each activity can be 4, 6, 8, or 10 and (2) when the number of system that may assigned to each activity is 6, the number of activities can be 4, 6, 8, or 10. The results of both are shown in Figs. 4(a) and 4(b). More than 300 experiments have demonstrated: with the increase in the number of system or activities, the execution time for the heuristic, genetic algorithm and particle swarm optimization methods also increases. However, the execution time of the heuristic method was only 1% to 5% of the average execution time of the genetic algorithm and particle swarm optimization methods as in Fig. 4.



(a) Average execution time for three algorithms (Activities=6)



(b) Average execution time for three algorithms (Systems=6)

Figure 4. Comparison of the genetic algorithm (GA), particle swarm optimization (PSO), and heuristic methods

All experiments show that the heuristic method is comparatively not as efficient as the genetic algorithm or particle swarm optimization methods in obtaining the optimal solution; however the heuristic approach performs efficiently in searching for available solutions within a short time. Considering that the heuristic approach requires a short search period, we combined the genetic algorithm or particle swarm optimization methods with the heuristic method. The purpose was to identify the primary available solution by using the heuristic method and then search for the reasonable solution by using the genetic algorithm or particle swarm optimization methods, with the primary solution as the input.

V. CONCLUSIONS

This study investigated how system can be optimal to support task completion. The system structure optimization problem in this case involves obtaining a solution that minimizes the execution time and maximizes the success rate under cost constraints. The study presented a highly efficient heuristic algorithm that can determine a solution that is less desirable than the optimal one. The experiment shows that the heuristic algorithm performs more efficiently compared with the genetic algorithm and particle swarm optimization algorithm in terms of the average execution time.

ACKNOWLEDGEMENT

The authors gratefully acknowledge the Henan province science and technology research plan project (Grant No. 102102310372).

REFERENCES

[1] H. Levis, and L. W. Wagenhals, "C4ISR architectures: I. Developing a process for C4ISR architecture design," *Systems Engineering*, vol. 3, no.4, pp. 225-247, 2000.

[2] L. W. Wagenhals, I. Shin, D. Kim, and A. H. Levis, "C4ISR Structures II: A Structured Analysis Approach for Structure Design," *Systems Engineering*, vol. 3, no. 4, pp. 248-287, 2000.

[3] L. W. Wagenhals, S. Haider, and A. H. Levis, "Synthesizing executable models of object oriented architectures," *Systems Engineering,* vol. 6, no. 4, pp. 266-300, 2003.

[4] H. Philipp, and L. Pascal, "From integrated structure to integrated executable structure," *Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 148-153, 2010.

[5] H. M. Kim, and I. J. Hidalgo, "System of system optimization by pseudo-hierarchical mutistage model," *11th AIAA/ISSMO mutlidisplinary analysis and optimization conference*, 2006.

[6] R. A. Wolf, *Multiobjective collaborative optimization of system of system*. Cambridge: MIT, 2005.

[7] R. Z. Wang, and H. D. Cihan, "Executable System Architecting Using Systems Modeling Language in Conjunction with Colored Petri Nets," *Conference on Systems Engineering Research*, Los Angeles, CA, USA, 2008.

[8] T. V. Huynh, and J. S. Osmundson, "A Systems Engineering Methodology for Analyzing Systems of Systems Using the Systems Modeling Language," *2nd Annual System of Systems Engineering Conference*, 2006.

[9] X. J. Shen, K. S. Li, and H. H. Zhang, "Configuration for Complex Large-scale System Equipments," *Journal of Beijing University of Technology*, vol. 34, no. 4, pp. 347-352, 2008.

[10] C. Sun, Y. Tan, J. Zeng, J. Pan, Y. Tao, "The Structure Optimization of Main Beam for Bridge Crane Based on An Improved PSO," *Journal of Computers*, vol. 6, no. 8, pp. 1585-1590, 2011.

[11] D. Li, R. Jin, "Integrated Structure and Control Design for Servo System Based on Genetic Algorithm and Matlab," *Journal of Computers*, vol. 6, no. 9, pp. 1903-1912, 2011.

[12] N. Zhou, J. Zhang, F. Ju, S. Liu, "Genetic Algorithm Coupled with the Neural Network for Fatigue Properties of Welding Joints Predicting," *Journal of Computers*, vol. 7, no. 8, pp. 1887-1894, 2012.

[13] Y. Huang, J. Miao, "Efficiency Optimization of Asynchronism Motor Based on a Novel Reaching Law," *Journal of Computers*, vol. 8, no. 4, pp. 884-887, 2013.

[14] C. G. Xue, L. L. Dong, and J. J. Liu, "Enterprise information system structure optimization based on time property with improved immune algorithm and binary tree," *Computers and Mathematics with Applications*, vol. 63, no. 7, pp. 1155-1168, 2012.

[15] R. S. Kadadevaramath, J. C. H. Chen, B. L. Shankar, and K. Rameshkumar, "Application of particle swarm intelligence algorithms in supply chain network structure optimization," *Expert Systems with Applications*, vol. 39, no. 23, pp. 10160-10176, 2012.

[16] R. C. Eberhart, and J. Kennedy, "A new optimizer using particle swarm theory," *Proceedings of the sixth international symposium on micro machine and human science*, pp. 39–43, 1995.

[17] G. C. Luh, C. Y. Lin, "A Binary Particle Swarm Optimization for Structural Topology Optimization," *The Third International Joint Conference on Computational Science and Optimization*, 2010.

Derong Zhu was born in Sichuan, China, 1972. He is a associate professor at the mechanical and electronic engineering department in Luoyang Institute of Science and Technology, China. His research interests include computer technology and application, machine electro-hydraulic computer integration technology research, complex systems simulation, machinery design and manufacture.