Context

The interest in multiobjective heuristic optimization is growing in the SBSE research community (2% of the papers in 2005 versus 20% of the papers in 2010).

Despite the advantages of MO algorithms, their application to SE problems raises new research questions, such as the number of objectives which should be explored by the search process.

Praditwong et al. observed that a search for solutions which maximize a given objective in the field of software module clustering might be improved by searching for solutions which also maximize the components of the desired objective. Similar results have been formerly observed in other research areas using MO search algorithms.

In this research we have designed and executed a set of experimental studies addressing the efficiency and effectiveness of adding an extra objective to a MO search process in the context of software module clustering.

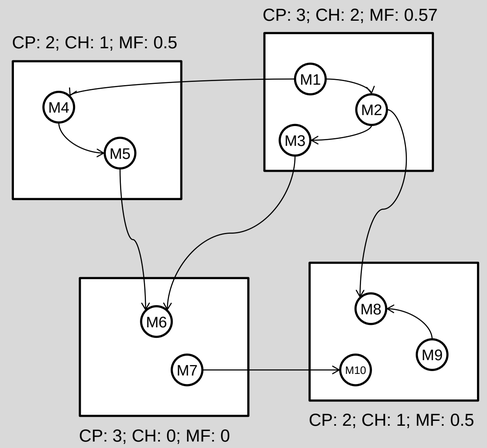
MO Software Module Clustering

Software module clustering addresses the distribution of modules representing the domain concepts and computational constructs composing a software system into larger, container-like structure.

We use a formulation based on structural, non-weighted dependencies among the software modules.

Coupling counts the number of inter-cluster edges.
Cohesion counts the number of intra-cluster edges.
Difference is the number of modules in the larger cluster minus the number of modules in the smaller one.
MQ is calculated according to the Bunch approach.
Number of clusters is fixed.

Given a set of modules, their dependencies, and the number of clusters, the MO search minimizes coupling (CP) and difference (DIF), while maximizing cohesion (CH) and MQ.



TOTALS: CP: 10; CH: 4; MQ: 1.57; DIF: 1

Results

We executed a MO algorithm (NSGA-II) both with 3 objectives (cohesion, coupling, and difference) and 4 objectives (the former plus MQ) and compared these pair wise results in a per instance basis.

We used a set of random instances to run the experiment for instances of varying size (number of modules) and complexity (number of clusters). We have also used a set of real world instances.

Spread and HV quality indicators were used to compare the joint best fronts of both simulations. The number of solutions and solutions in the joint best fronts were also counted.

INSTANCE	Solution Count		Best Solution		Spread		Hypervolume	
	W/out MQ	With MQ	W/out MQ	With MQ	W/out MQ	With MQ	W/out MQ	With MQ
JAVACC (154M, 6C)	27.6 ± 7.3	28.1 ± 5.2	1.7 ± 4.6	0.2 ± 0.8	0.59 ± 0.11	0.62 ± 0.08	**0.14 ± 0.08**	0.09 ± 0.05
SERVLET (100M, 6C)	**15.8 ± 6.9**	11.0 ± 6.7	0.5 ± 2.2	0.7 ± 2.3	0.67 ± 0.14	0.76 ± 0.16	0.09 ± 0.09	0.04 ± 0.06
XML DOM (63M, 4C)	28.0 ± 4.9	28.0 ± 5.6	1.4 ± 4.4	0.6 ± 2.2	0.56 ± 0.08	0.58 ± 0.11	**0.22 ± 0.10**	0.16 ± 0.08
JUNIT (119M, 9C)	**25.2 ± 8.8**	19.6 ± 8.1	1.2 ± 4.7	0.8 ± 1.3	0.64 ± 0.13	0.70 ± 0.14	0.18 ± 0.12	0.16 ± 0.13
JMETAL (190M, 46C)	40.8 ± 9.4	44.8 ± 9.3	**3.2 ± 5.1**	0.4 ± 0.9	0.50 ± 0.09	0.52 ± 0.08	0.44 ± 0.08	0.41 ± 0.04
XML API (184M, 17C)	26.0 ± 3.4	**31.1 ± 3.5**	1.0 ± 2.8	0.8 ± 2.5	0.51 ± 0.10	0.50 ± 0.12	0.34 ± 0.09	0.32 ± 0.06
DOM4J (195M, 616)	37.2 ± 9.5	35.2 ± 9.3	2.0 ± 4.4	1.4 ± 3.6	**0.58 ± 0.12**	0.66 ± 0.11	**0.44 ± 0.07**	0.37 ± 0.09
POORMAN (304M, 15C)	26.8 ± 5.4	26.4 ± 8.3	1.6 ± 4.1	0.2 ± 0.6	0.81 ± 0.19	0.81 ± 0.18	0.54 ± 0.09	0.52 ± 0.07
LOG4J (308M, 20C)	**30.9 ± 4.2**	27.2 ± 3.4	0.8 ± 3.1	1.2 ± 3.0	**0.54 ± 0.14**	0.63 ± 0.11	0.43 ± 0.06	0.42 ± 0.07
SEEMP (31M, 9C)	7.7 ± 0.7	7.7 ± 0.7	4.0 ± 1.8	4.6 ± 1.1	0.44 ± 0.04	0.41 ± 0.06	0.34 ± 0.06	0.34 ± 0.05

Contributions

The Pareto fronts generated by simulations without MQ are competitive with those generated by simulations considering MQ as an extra objective.

Since adding an extra-objective to the search process conducted by a MO algorithm increases the computer processing time required to execute the search, MQ might be suppressed.

Why is this relevant?

Praditwong et al. indicate that their MO software clustering approach can produce better solutions than the existing single objective approach, though with a increased computational cost.

The table on the right shows that the time required to account for MQ in a MO clustering search is significant.

Therefore, if the extra objective can be suppressed from the search without significant loss, the computational power required to run a MO software clustering will be closer to the existing SO solution.

INSTANCE	W/out MQ	With MQ	Increase
JAVACC	16	54	338%
SERVLET	2	4	188%
XML DOM	29	98	338%
JUNIT	10	26	274%
JMETAL	2,603	27,822	1069%
XML API	219	1,243	568%
DOM4J	211	1,231	583%
POORMAN	374	2,263	605%
LOG4J	679	5,236	771%
SEEMP	16	24	150%

PPGI - UNIRIO

Márcio Barros, PPGI / UNIRIO – Research project supported by FAPERJ and CNPq