Many objective optimization and complex network analysis

Maulana, A.

PART I

3

Community Detection for Reducing Complexity of Many Objective Optimization

3.1 · Introduction

In the last decade, the modern science of networks was probably the most active field within the interdisciplinary research field of complex systems. Many complex systems can be represented as networks, where the elementary parts of a system and their mutual interactions are nodes and links, respectively. Study and analysis of networks is a challenging research topic.

Social network analysis as a subfield of network science that is concerned with the classification and clustering of networks. As such it is powerful in defining the community and correlation among nodes (representing individuals) in the context of social networks. However, community detection in large networks needs a special technique that can help to optimize results. In recent years multi-objective optimization for community detection has been developed by many researchers. Here, multi-objective optimization seeks to find an optimal clustering with respect to criteria such as *modularity* and *parsimony*.

In our research, however, we turn the question around and seek to apply social network analysis, and in particular community analysis, to analyze multi-objective optimization problems with a large number of objectives, as they are, for instance, given in problems of facility location problems, multi-class classification problems, nurse scheduling, or multidisciplinary design.

This chapter starts with a definition of the problem (Section 3.2) and a brief summary of related work (Section 3.3). The approach of mathematical statistics has been described in Chapter 2. Furthermore, a workflow of the decomposition and the applied algorithms will be discussed in Section 3.4. Empirical proof of concept studies illustrate and validate the methodology in Section 3.5. Finally, Section 3.6 closes the chapter with a summary of the main results and a discussion of current limitations of the method.

3.2 · Problem definition

Recall from Chapter 2, that a multi-objective optimization problem (MOP) is defined by a number of objective functions $f_i : X \to \mathbb{R}, i = 1, \ldots, m$ to be minimized (or maximized) for some search space X. Without loss of generality we consider only maximization as a goal. Moreover, we will not consider constrained problems in the first place, that is we assume that X consists only of feasible solutions.

A prevalent problem in multi-objective optimization is that for a large number of objective functions it is not possible anymore to visualize Pareto fronts in a Cartesian coordinate system. The trade-off analysis of conflicts between objective functions can become very intransparent. Moreover, a high number of objectives can yield to a significant increase in the computational time complexity required to compute Pareto fronts.

The aim of our research is to *reduce complexity in many-objective optimization by aggregating complementary objective functions or decomposing the problem into independent subproblems*.

A novel approach to achieving this is studied. This approach seeks to apply *community detection in networks* in order to form groups of objectives that support each other, and to separate groups of objectives that are independent of each other. This will form clusters of objective functions: $\{C_1, \ldots, C_k\}$ with $k \leq m$ forming a partitioning of the set $\{f_1, \ldots, f_m\}$. Moreover, we will establish dependence and independence relations between these clusters, based on whether or not two clusters are conflicting or neutral with respect to each other.

The analysis will be based on an interpretation of objective functions and their correlation with random variables. Positive correlation will be interpreted as a positive link, negative correlation as a negative link, and, finally, zero correlation as no link in the network. In the following, we will detail the components of the approach and

discuss its scope (types of MOPs where it is suitable) and limitations. Moreover, we will provide a benchmark study in order to prove the concept empirically.

3.3 · Related Work

Many-objective optimization was introduced as a class of problems with more than four objective functions [48]. It has been observed that classical evolutionary multi-objective optimization methods (EMOA), such as NSGA-II [17] and SPEA2 [65], typically work well only for a few objective functions (ca. 2 – 4). Firstly, the results of optimization are more difficult to interpret than for problems with, say, only two or three objectives. Moreover, hypervolume-based methods such as SMS-EMOA [8] that perform very well for small numbers of objective functions become computationally infeasible for a higher number of objective functions [55].

This is the reason why algorithms have been suggested that are especially well suited for many-objective optimizations. Examples of algorithms that perform well in many-objective optimization are HyPE [5] and MOEA/D [64]. However, they do not make use of special properties of the problem such as a correlation between objective functions. Here, it shall be noted that in the general case the Pareto front of a m dimensional problem can be a $m - 1$ dimensional manifold, which in case of a large number of objectives leads to a reduction of the space of alternatives that can be considered as marginal. Moreover, it is very difficult to get any intuition about trade-offs from the data on such high dimensional spaces.

The situation changes if some of the objective functions are correlated or anti-correlated with each other. On the one hand, some of the aforementioned algorithms perform better in such cases [26] and on the other hand, if correlations are suspected a-priori, they can be exploited actively by the algorithm to reduce the problem difficulty. For instance, in [51] it is suggested to use Principal Component Analysis (PCA) for finding a reduced set of objectives. The research presented in this thesis goes in a similar direction, that is it seeks to exploit the correlation between objective functions. In [28] the idea that correlated objective functions cause redundancy was exploited to reduce the number of objective functions during a run.

However, the proposed method will differ from existing approaches in three important aspects:

- We will take a new perspective on how to view the problem of aggregation and decomposition by *viewing objective functions as entities of a community*, and use

community detection algorithms. These methods not only can help to simplify the problem of finding the Pareto front but also provide interesting graph based visualizations of the problem's inherent structure to the decision maker.

- As in [28, 51], we will use correlation, but treat correlation, anti-correlation, and neutrality differently, thereby exploiting certain features that the social community based perspective offers. Informally speaking, objective functions can be friendly, neutral or enemies with each other.

- We will propose a workflow-approach that provides feedback to users in different stages of the optimization process. The method can be successful (in case sufficient structure and modularity are found) or unsuccessful, in which case one has to resort to conventional methods. In case of success, it can provide the user with feedback on the essential structure of the Pareto front by means of low dimensional trade-off surfaces and also generate sets that cover the essential parts of it.

Related to coefficient correlation as explained in Chapter 2, in the context of decision making, we can interpret this correlation as follows:

- Positively correlated objective functions can be interpreted as objective functions that support each other. This means that optimizing one function will imply that also the other function will obtain good values.

- Uncorrelated objective functions are considered to be independent of each other. They can be optimized in isolation from each other. The MOP can be decomposed in MOPs that consist of partitions with less objective functions.

- Negatively correlated objective functions are in a strong conflict with each other. For conflicting objective functions trade-off analysis needs to be conducted, which in turn can be prepared by computing the Pareto front of the problem.

Here it is proposed to compute the correlation between all pairs of objective functions. The information is then interpreted in a relational context and we apply community detection in order to identify communities of objective functions. Communities can be opposed to each other (negative link), or neutral with respect to each other (no link). The fundamental idea of this chapter is how to use techniques from social network analysis in order to decompose and aggregate.

Once the correlation between the objective functions is found, one can make a grouping of the objective functions based on the correlation status of those objective functions either being correlated or conflicting. Moreover, each group that consists of both correlated and conflicting objective functions will pull/attract each other. This information can serve the needs of solving many objective optimization problems, by grouping objective functions and decomposing the problem into subproblems.

3.4 · Workflow and Algorithms

The workflow and algorithmic methods used in the community-detection for many-objective optimization (CoDeMO) approach will be used. As opposed to the term 'algorithm', by a 'workflow' we mean a step-wise, linear process in which the user interacts or inspects data or visualizations at several stages, and possibly is asked for feedback.

The CoDeMo workflow proceeds with the following steps:

1. **Problem sampling**: Generate n samples of decision variable vectors that are evenly distributed in the range of the decision variables.

2. **Evaluation**: Evaluate the m objective functions for each sample, which yields an $n \times m$ matrix.

3. **Correlation Analysis**: Compute the correlation coefficient matrix from the table of objective function values, which yields a $m \times m$ matrix of correlation coefficients in $[-1, 1]$.

4. **Construct Network**: Take each objective function as a node, and the correlation coefficient between two objective functions as a link, which means the correlation coefficient matrix will be taken as the matrix of the constructed signed network.

5. **Detect Communities**: Use a graph-theoretic algorithm to detect communities using the information of the correlation matrix and interpreting them as edge weights (values near zero indicate non-existence of edge).

6. **Decompose**: Based on the result of the community detection, decompose the problem into $k \leq m$ subproblems by partitioning the set $\{f_1, \ldots, f_m\}$ into communities of mutually non-independent objectives.

7. **Aggregate**: Within each community: Aggregate cliques of communities that support each other to a single objective function by summation, resulting for each subproblem i with, say, k_i objective functions, in a similar subproblem with $\ell < i$ pseudo-objective functions.

8. **Solve Subproblems**: Find an approximation and Pareto front for each of the subproblems with aggregated objectives.

9. **Merge**: Merge all results and output them in a Parallel coordinate diagram, first ordered by subproblem and within subproblems by aggregates.

The Algorithm of community detection used in step 5 (as explained in chapter 2), but with more specific measured weight, is called BGLL [29]. The algorithm is designed to use a greedy strategy to improve the modularity on a signed network [24], which is defined as

$$Q_{signed} = \frac{1}{2w} \sum_i \sum_j \left[w_{ij} - \left(\frac{w_i^+ w_j^+}{2w^+} - \frac{w_i^- w_j^-}{2w^-} \right) \right] \delta(C_i, C_j)$$

where w_{ij} is the weight of the adjacency matrix, $w_i^+(w_j^+)$ is the sum of all positive weights of node $i(j)$. $w^+(w^-)$ denotes the sum of absolutes of all positive(negative) weights of the signed network and $w = w^+ + w^-$. $C_i(C_j)$ is the community label of node $i(j)$ and $\delta(C_i, C_j)$ is 1 if $C_i = C_j$; otherwise 0.

BGLL greedily aggregates two nodes into one community if it causes a large modularity increment. The aggregation will repeat many times until the optimum community structure is obtained. The algorithm is implemented in Pajek, were it is called Louvain method [54]. Details can be found in [29] [54].

The introduced nine step workflow involves the user in several parts. First and foremost, the community structure visualization for the objective functions might provide valuable insights into the problem's structure to the user. As we will not find a clear-cut partitioning into cliques, in the current CoDeMO workflow the user needs to perform the merging and aggregation steps. It might be possible, however, to develop robust criteria for doing this automatically.

The visual output of the community detection process is an important result of the workflow, as it supports the decomposition process by providing essential insights into the problem structure. Its graphical elements are:

- Each objective function is a node in a circle.
- The graph visualization places anti-correlated nodes distant to each other and indicates their conflicting nature by blue connectors.
- It places objective functions that are positively correlated close to each other and indicates their supportive relation by red connectors.
- Elements that have close to zero correlation, meaning that they are independent (neutral) of each other (a threshold $1/e$ is used here, where e is the Euler number), and do not have a connecting line segment.

An example of a community detection result for 30 objective functions and subsequent problem decomposition and aggregation is provided in Fig. 3.5. From this graph, the grouping of the objective functions is as follows: Firstly, we identify three independent subproblems. The community detection places the nodes in such a way that the connected components of the graph can be easily identified:
$C_1 = \{1, 2, 3, 4, 6, 7, 8, 5, 9, 10\}$,
$C_2 = \{20, 18, 17, 16, 14, 13, 12, 19, 15, 11\}$,
$C_3 = \{21, 22, 23, 24, 25, 26, 27, 28, 29, 30\}$.
In the aggregation step of the workflow we aggregate the objective functions, which yields three subproblems:
Subproblem 1:
$\Phi_1^{C1} := f_1 + f_2 + f_3 + f_4 + f_6 + f_7 + f_8 \rightarrow \min$,
$\Phi_2^{C1} := f_5 + f_9 + f_{10} \rightarrow \min$.
Subproblem 2:
$\Phi_1^{C2} := f_{11} + f_{15} + f_{19}$,
$\Phi_2^{C2} := f_{20} + f_{18} + f_{17} + f_{16} + f_{14} + f_{13} + f_{12} \rightarrow \min$.
and Subproblem 3:
$\Phi_1^{C3} := f_{24} + f_{25} + f_{30}$,
$\Phi_2^{C3} := f_{12} + f_{13} + f_{14} + f_{16} + f_{17} + f_{19} + f_{20} \rightarrow \min$.
Now, instead of solving a 30-objective problem, three bicriteria optimization problems can be solved instead without losing essential information.

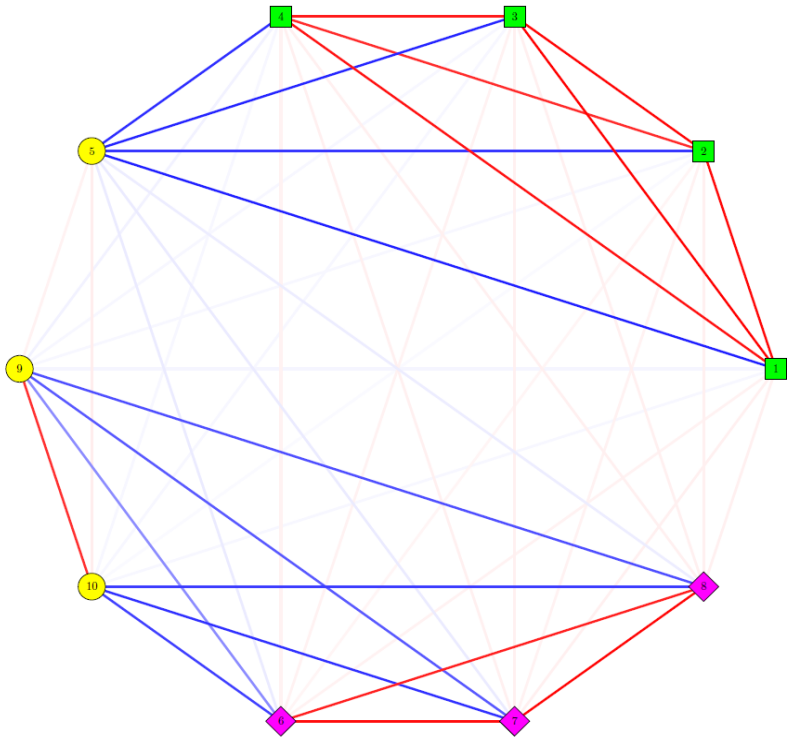

Figure 3.1 Community detection for ten objective function

3.5 · Experimental Analysis

For testing the approach in a proof-of-concept study we test it on a problem for which we already have an idea of the inherent structure of communities, but the algorithm is not provided with this information. Moreover, the problem is scalable and has a practical background, being a special case of a multi-facility location problem.

In the following, we only consider one facility of each type, which provides a benchmark with a known solution in terms of decomposition into subproblems and aggregation of objective functions. The decision vector is then given by $(x_1, x_2, \ldots, x_{2N-1}, x_{2N})$, where (x_1, x_2) is the coordinate of the first facility of type 1, (x_3, x_4) is the coordinate of the second facility of type 2, and so on. We assume the position of the citizens is fixed and each one of the citizens forms a separate objective function. More concretely citizen's needs are of the kind that a citizen with coordinates $(c_1^{(.)}, c_2^{(.)})$ wants to be close to a facility of type 1, a citizen with coordinates $(c_3^{(.)}, c_4^{(.)})$

wants to be close to a facility of type 2, a citizen with coordinates $(c_5^{(\cdot)}, c_6^{(\cdot)})$ want to be close to facility 3, and so on.

Moreover, in order to test the capability of the workflow to detect independent clusters we make the additional assumption that each citizen has only one facility that he or she wants to be located close to.

3.5.1 · Problem with 10 Objectives

As a first example we choose ten nodes as 10 objective functions. The functions have the signatures $f_1 : \mathbb{R}^4 \to \mathbb{R}$, ..., $f_{10} : \mathbb{R}^4 \to \mathbb{R}$. The functions f_1, f_2, ..., f_5 are sensitive to changes of x_1 and x_2 (first facility type) and the functions f_6, ..., f_{10} are sensitive to changes of x_3 and x_4 (second facility type). These functions are distances to centers of facility type 2 ((x_3, x_4)-space). Table 3.1 lists the locations of $c^{(i)}$ (the citizens) for each objective function and a graphical representation of them is found in Fig. 3.2 (for facilities of type 1) and in Fig 3.3 (for facilities of type 2).

Table 3.1 Facility locations and facility types.

10 Objective Functions				
	$c_1^{(i)}$	$c_2^{(i)}$	j_1	j_2
f1	1.1	1.2	1	2
f2	0.9	0.8	1	2
f3	1	1	1	2
f4	0.7	1.1	1	2
f5	0.2	0.1	1	2
f6	0.7	1	3	4
f7	1	1.1	3	4
f8	1	0.9	3	4
f9	0.2	0.4	3	4
f10	0.3	0.2	3	4

In Step 1 of the workflow, we sample the functions randomly within the $[0, 1]-$ range 500 times. Then, in Step 2, evaluate the objective functions for each point, yielding a 500×10 matrix of function values. From this matrix – in Step 3 of the workflow – we determine correlation coefficients.

The full matrix with one correlation coefficient for each pair of objective functions is shown in Table 3.1.

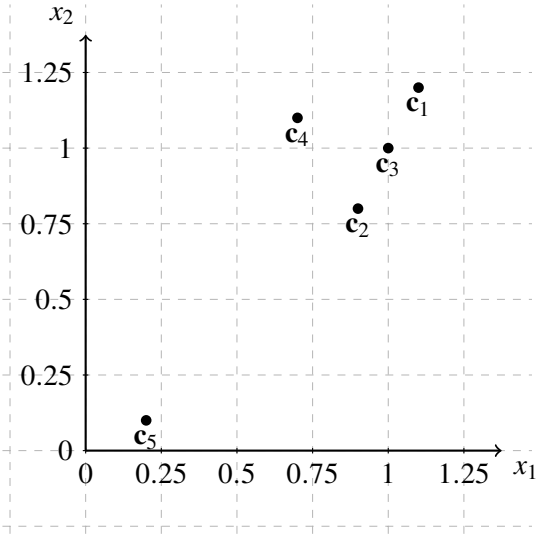

Figure 3.2 The centers in the coordinate space spanned by x_1 and x_2. In a facility location problem, the centers stand for citizens who demand facility of type 1 in their proximity.

Next, in Step 4 of the CoDeMO workflow, we apply the community detection method, which yields Fig. 3.1. Based on the clear structure of the graph, Step 4 (decomposition) and Step 5 are straightforward and we identify two independent subproblems, one involving the set $C_1 = \{1, 2, 3, 4, 5\}$, and the other involving the set $C_2 = \{6, 7, 8, 9\}$. This result, which we obtained from the graphics, is also plausible in the context of the facility location problem because the set of objective functions is partitioned into sets of functions for the same type of facility. The aggregation is done according to the color given to the nodes by the community detection algorithms, yielding Subproblem 1:

$$\Phi_1^{C1}(x) = (f_1 + f_2 + f_3 + f_4)(x) \to \min \tag{3.1}$$

$$\Phi_2^{C1}(x) = f_5(x) \to \min \tag{3.2}$$

$$x \in [0,1]^4, \tag{3.3}$$

and Subproblem 2:

$$\Phi_1^{C2}(x) = (f_6 + f_7 + f_8)(x) \to \min \tag{3.4}$$

$$\Phi_2^{C2}(x) = (f_9 + f_{10})(x) \to \min \tag{3.5}$$

$$x \in [0,1]^4 \tag{3.6}$$

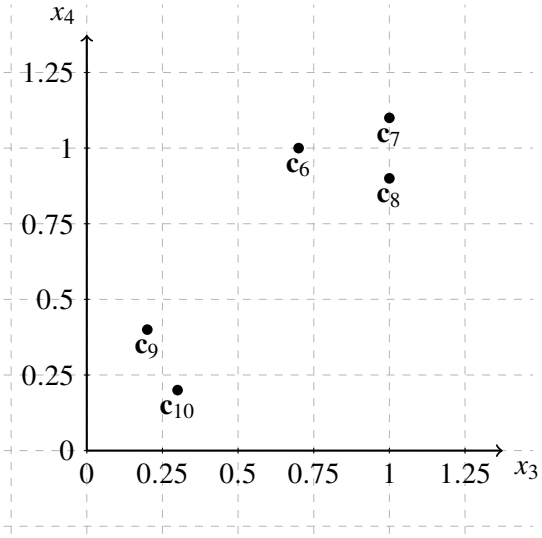

Figure 3.3 The centers in the coordinate space spanned by x_3 and x_4. In a facility location problem, the centers stand for citizens who demand facility of type 2 in their proximity.

Step 7 is to perform optimization on the subproblem and then to visualize results for the objective functions. In Fig. 3.6 the results for Φ_1^{C1} and Φ_2^{C1} using NSGA-II and MOEA/D are depicted. For the optimization, we used JMetal with a population size of 300 and a budget of 15000 evaluations. On the plain many-objective optimization problem MOEA/D achieved better results than NSGA-II (Fig. 3.6, top). For this reason, we show the result of a comparison only between MOEA/D and CoDeMO, using MOEA/D for the lower dimensional subproblem. As expected, the accuracy of the front is much better when the subproblem is optimized directly. This holds for both problems, but for subproblem 1 the difference was most significant. For subproblem 1 (Fig 3.6) the range of Φ_2 is much better captured when CoDeMO is used.

3.5.2 · Problem with 30 and 50 objectives

In a similar manner we created a MOP with 30 objective functions and 50 objective functions. For these problems we will only demonstrate the decomposition process. This time three facility types are featured. Coordinates are given by (x_1, x_2), (x_3, x_4), and, respectively, (x_5, x_6). Table 3.3 shows the centers: By a given correlation matrix, the visualization of the community structure can be seen in Fig. 3.5, and the scatter plot in Fig. 3.7. The three clusters are clearly separated from each other and conflicting

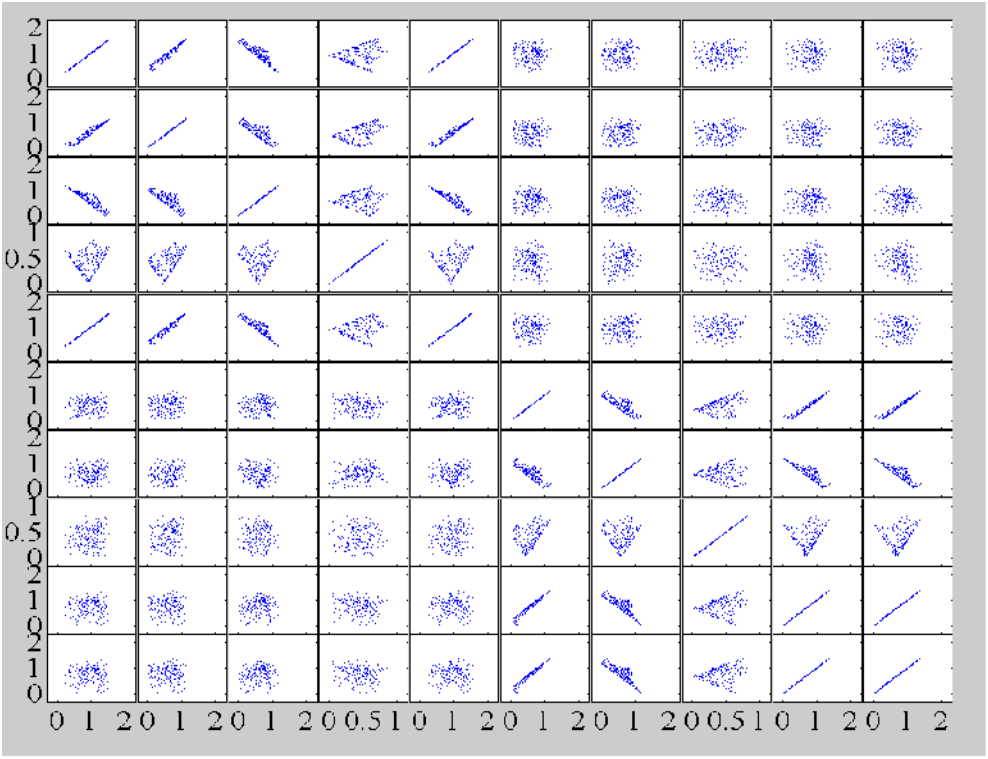

Figure 3.4 Scatter plot matrix for the 10 objective problem.

objectives placed opposite to each other. We can decompose this problem as described in Section 3.4.

The table of the 50 citizen problem exceeds the size of the paper and is made available as support material. We only display the scatter plot matrix of this problem in Fig. 3.9. We used a similar structure than in the 30-dimensional problem but added citizens to the clusters. The resulting community graph is depicted in Fig. 3.8. This scatter plot matrix is almost unreadable and we only show it to contrast it to the clearer picture we obtain from graphs provided by community detection. Again, the community detection algorithm manages to visualize the clusters in a way that allows for simple decomposition and aggregation. The result shows that the community detection also can work for larger size problems if the underlying structure is sufficiently simple.

A note of caution is to be made here: Nodes 37 and 4 in the lower left of the biggest cluster are not easy to place. They are close to two nodes in terms of positive correlation but conflict with all other nodes in the cluster. This 'in-between-ness' makes it difficult to

Table 3.2 Correlation Coeffecients for 10 objective function with 500 inputs.

1.00	0.97	1.00	0.93	-0.88	0.05	0.06	0.06	-0.04	-0.03
0.97	1.00	0.98	0.83	-0.79	0.04	0.05	0.06	-0.03	-0.02
1.00	0.98	1.00	0.90	-0.86	0.05	0.06	0.06	-0.03	-0.03
0.93	0.83	0.90	1.00	-0.82	0.05	0.06	0.06	-0.05	-0.03
-0.88	-0.79	-0.86	-0.82	1.00	-0.08	-0.08	-0.07	0.05	0.07
0.05	0.04	0.05	0.05	-0.08	1.00	0.95	0.88	-0.45	-0.75
0.06	0.05	0.06	0.06	-0.08	0.95	1.00	0.98	-0.65	-0.81
0.06	0.06	0.06	0.06	-0.07	0.88	0.98	1.00	-0.69	-0.76
-0.04	-0.03	-0.03	-0.05	0.05	-0.45	-0.65	-0.69	1.00	0.81
-0.03	-0.02	-0.03	-0.03	0.07	-0.75	-0.81	-0.76	0.81	1.00

decide to which aggregate objective function they belong. In CoDeMO the assignment to clusters is done manually and the user could aggregate them as a new (third) objective function thereby making the visualization and analysis of the subproblem more difficult. Alternatively, (s)he could place them in the lower community – ignoring their positive correlation with two objective functions. The latter approach has the advantage that the subproblem can be solved as a problem with only two objectives. To avoid loss of essential information for decision making, we recommend using the first approach.

3.5.3 • Limitations of the Approach

In cases where much less community structure is available than in our examples, our approach is unfortunately only of limited use. In such cases, it at most provides negative results, that is the insight that the problem's correlation structure is inherently complex. There might be approaches that are able to exploit types of structures based on correlation other than those exploited in CoDeMO, but this falls beyond the scope of this study.

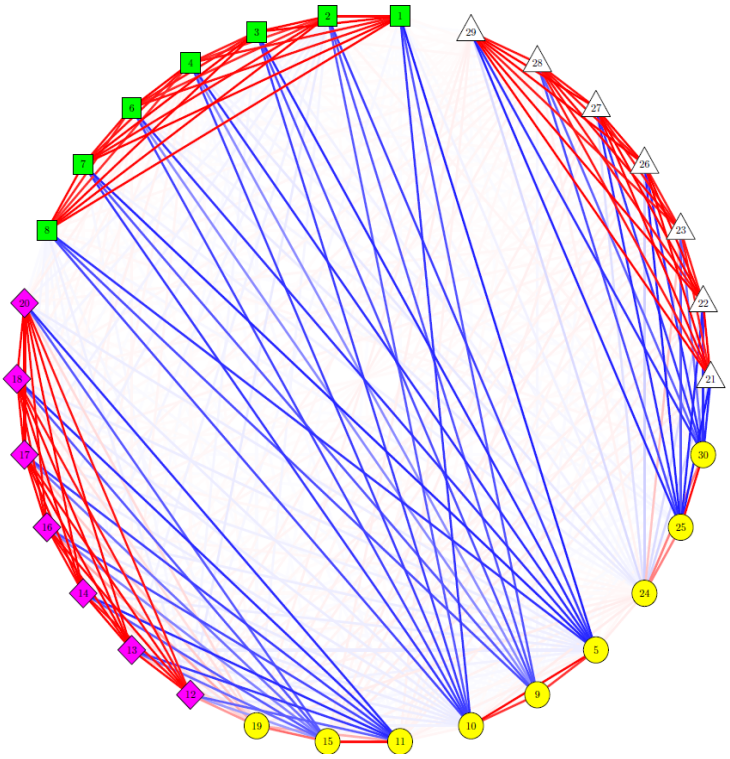

Figure 3.5 Example of a community detection result for 30 objective functions.

An even more subtle case of a difficult problem structure occurs in case of a phenomenon we will refer to as *higher order conflict*: It is possible to construct problems where every pair of objective function is non-conflicting but there is no solution that satisfies all objective functions. For instance the problem $min\{||\mathbf{x},(0,0,1)||,||\mathbf{x}-(0,1,0)||\} \rightarrow \min$, $min\{||\mathbf{x}-(1,0,0)||,||\mathbf{x}-(0,1,0)||\} \rightarrow \min$, and $min\{||\mathbf{x}-(1,0,0)||,||\mathbf{x}-(0,1,0)||\} \rightarrow \min, \mathbf{x} \in [0,1]^3 \subset \mathbb{R}^3$ has this characteristic. An appropriate technique to visualize this information would be to use community detection in hypergraphs.

3.6 · Summary

In this chapter, we applied a community detection technique to reduce the number of the objective functions in many-objective optimization. A workflow called Community Detection for Many-objective Optimization (CoDeMO) was discussed that uses graph-theoretic community detection to reveal the structure of many-objective black-box

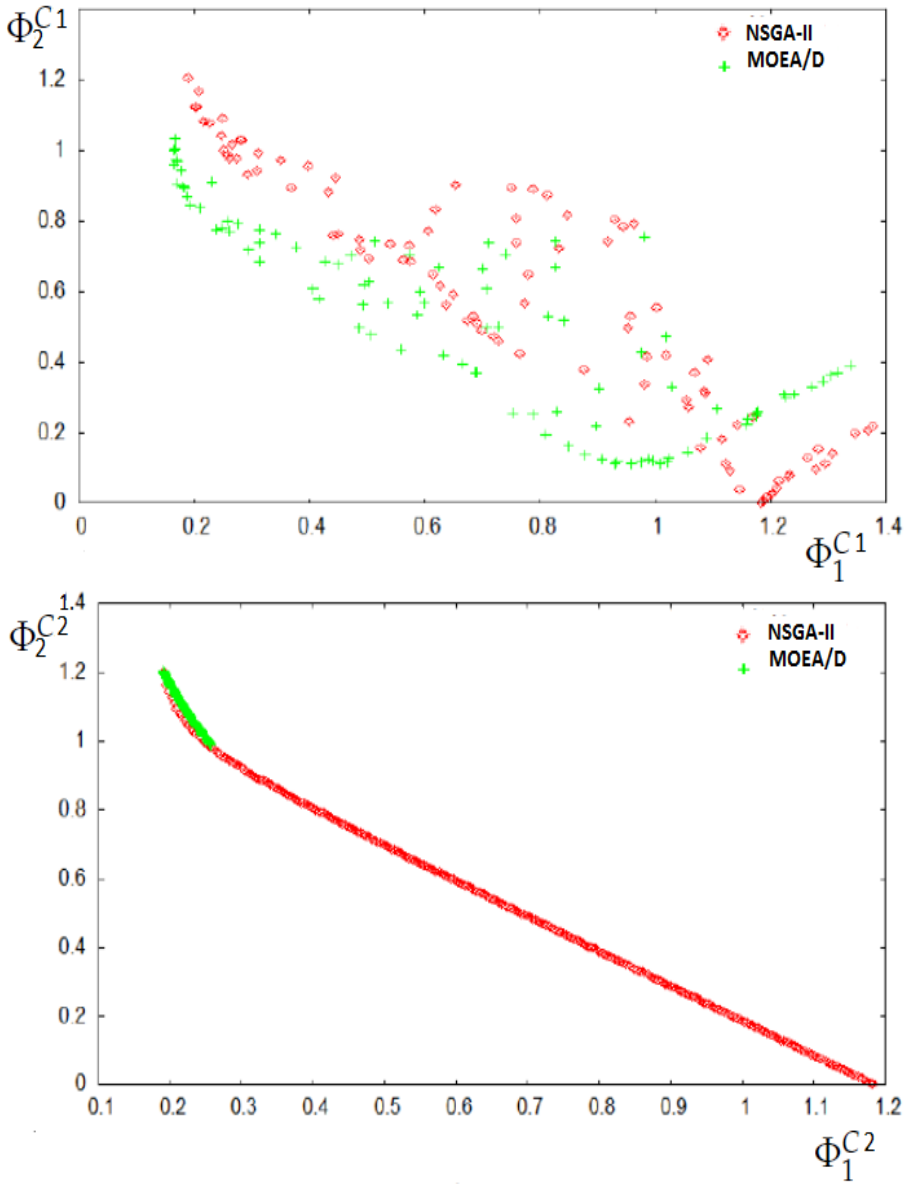

Figure 3.6 Comparison with CoDeMO optimization results (using MOEA/D to solve the subproblems compared to results from conventional many objective optimizations (NSGA-II, MOEA/D) for the facility location problem with 10 objective functions. Here,◇ stands for NSGA-II and + stands for MOEA/D.

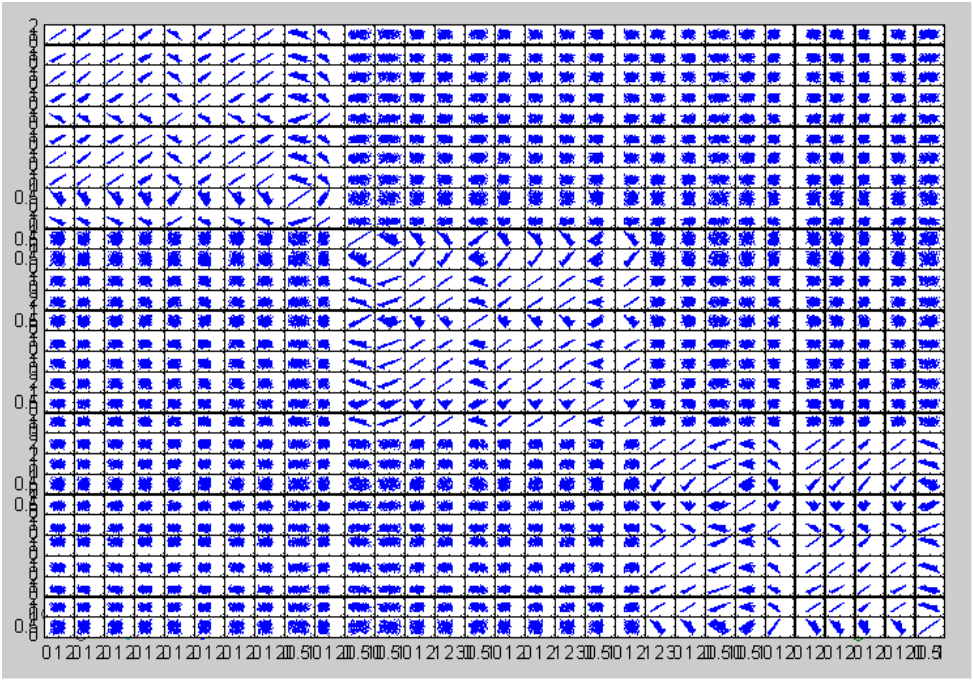

Figure 3.7 Scatter plot matrix for the 30 objective problem.

optimization problems. It interprets objective functions as actors in a social network (complimentary), which might be friends, enemies (conflicting) or neutral with respect to each other. The proof of concept study shows that for problems with a relatively simple underlying this approach works both to reveal the structure and to exploit it by providing more interpretable and also more accurate optimization results. The community detection works well for many-objective optimization and was tested with up to 50 objective functions.

In addition, we pointed to some limitations of the approach. We assume that in many cases the result of the community structure reveals whether or not a decomposition is possible. By pointing at the possibility of higher order interactions we also show that in some non-linear problems, simple structured problems have complex interactions. More examples for using the CoDeMO approach will be given in the subsequent chapters, namely for NK landscape analysis (Chapter 4) and for multilayer network analysis (Chapter 5 and Chapter 6).

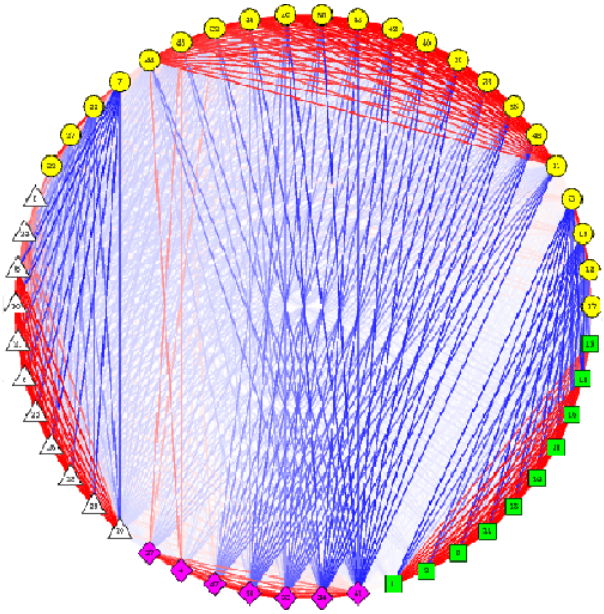

Figure 3.8 Example of a community detection result for 50 objective functions.

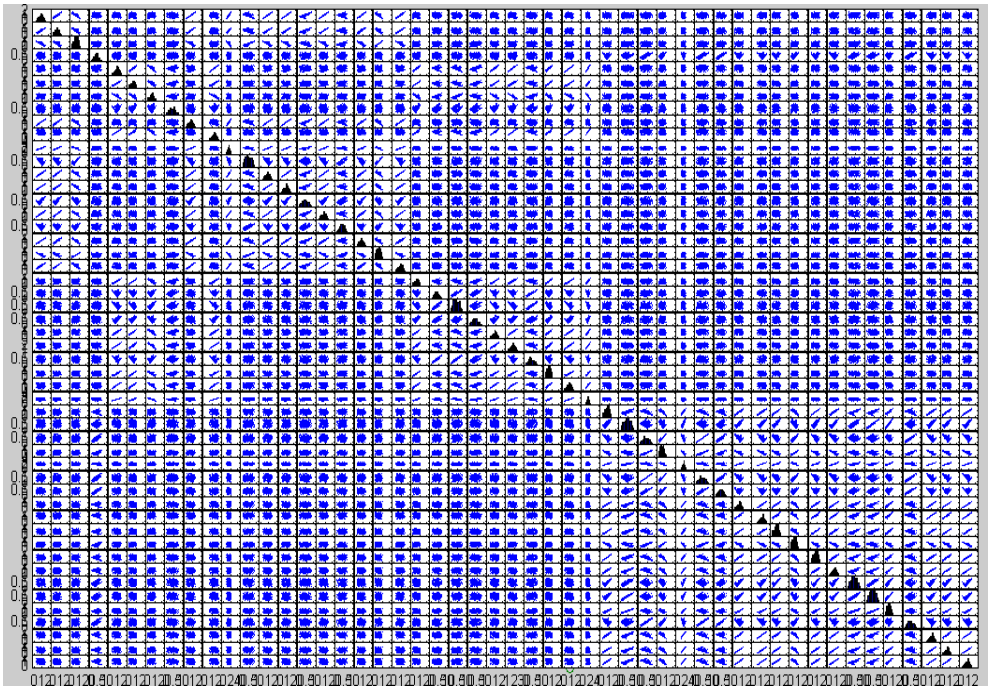

Figure 3.9 Scatter plot matrix for the 50 objective problem.

Table 3.3 Data on centers for the 30-Dimensional problem.

Data: 30-Objective Function	$c_1^{(i)}$	$c_2^{(i)}$	j_1	j_2
f1	1.1	1.2	1	2
f2	0.9	0.8	1	2
f3	1	1	1	2
f4	0.7	1.1	1	2
f5	0.2	0.1	1	2
f6	0.7	1	1	2
f7	1	1.1	1	2
f8	1	0.9	1	2
f9	0.2	0.4	1	2
f10	0.3	0.2	1	2
f11	0.2	0.3	3	4
f12	0.7	0.8	3	4
f13	1	1.1	3	4
f14	1.6	1.7	3	4
f15	0.3	0.4	3	4
f16	0.8	0.8	3	4
f17	0.9	1	3	4
f18	1.7	1.8	3	4
f19	0.5	0.5	3	4
f20	1.3	1.3	3	4
f21	1.7	1.8	5	6
f22	1.1	1.1	5	6
f23	0.7	0.7	5	6
f24	0.4	0.5	5	6
f25	0.2	0.2	5	6
f26	1.5	1.4	5	6
f27	1	1	5	6
f28	0.8	0.7	5	6
f29	1.1	1.1	5	6
f30	0.3	0.2	5	6