Automated Structure Detection for Distributed Process Optimization

Ehecatl Antonio del Rio-Chanona, Fabio Fiorelli, Vassilios S. Vassiliadis*

Department of Chemical Engineering and Biotechnology, University of Cambridge, Pembroke Street, Cambridge CB2 3RA, United Kingdom

Abstract

The design and control of large-scale engineering systems, consisting of a number of interacting subsystems, is a heavily researched topic with relevance both for industry and academia. This paper presents two methodologies for optimal model-based decomposition, where an optimization problem is decomposed into several smaller sub-problems and subsequently solved by augmented Lagrangian decomposition methods. Large-scale and highly nonlinear problems commonly arise in process optimization, and could greatly benefit from these approaches, as they reduce the storage requirements and computational costs for global optimization. The strategy presented translates the problem into a constraint graph. The first approach uses a heuristic community detection algorithm to identify highly connected clusters in the optimization problem graph representation. The second approach uses a multilevel graph bisection algorithm to find the optimal partition, given a desired number of sub-problems. The partitioned graphs are translated back into decomposed sets of sub-problems with a minimal number of coupling constraints. Results show both of these methods can be used as efficient frameworks to decompose optimization problems in linear time, in comparison to traditional methods which require polynomial time.

Keywords: distributed optimization, process optimization, design reformulation, community detection, multilevel graph bisection, augmented Lagrangian decomposition

1. Introduction

Decomposition is a general approach to solve problems by breaking them into smaller ones and solving each of the sub-problems separately, either in parallel or sequentially (Boyd et al., 2012). In the field of Process Systems Engineering (PSE) there is a need to accurately model real systems that are computationally intensive, both due to increasing scale of the models, *e.g.* large-scale plant-wide control and optimization, and the increased complexity of the reactions involved, such as for pharmaceutical products or nanotechnological applications (Stephanopoulos and Reklaitis, 2011; Rivera-Toledo et al., 2014). With increasing computing power of microprocessors and the ability to share the load across several of them with parallel architectures, decomposition methods offer a solution to many of the performance and flexibility requirements of the industry (Hamdi and Al-saud, 2008).

The storage requirements in optimization problems generally grow in proportion to $\max\{n \cdot m,\ n^2\}$ for dense systems (where n is the number of variables and m is the number of constraints), and become excessive for a large number of variables and constraints. As a result, in many cases problems cannot be solved in the

*Corresponding author: vsv20@cam.ac.uk

desired detail because of their large size. However, by using methods that decompose the original problem into a set of smaller ones these complex systems can be successfully optimized (Sagastizábal, 2012).

Computational savings may be a second motivation for distributed optimization. Benefits for local optimization are commonly perceived to be small, if present at all (Alexandrov and Lewis, 1999). On the contrary, global optimization may benefit substantially from distributed optimization because computational costs for global optimization algorithms often increase rapidly with the number of design variables (Tosserams et al., 2009, 2010). Therefore, solving a number of smaller sub-problems is expected to be preferable over solving a single large problem if a good degree of separability can be achieved, provided that the coordination overhead introduced by distributed optimization is relatively small (Haftka et al., 1992; Haftka and Watson, 2005). Moreover, decomposition algorithms have been used as initialization strategies for nonlinear optimization when good initial estimates are hard to find (Safdarnejad et al., 2015). Additional benefits are expected when these sub-problems can be solved in a parallel computing environment, thus allowing larger problems to be solved.

The design and control of large-scale engineering systems that consist of a number of interacting subsystems is very common and is heavily researched in areas such as multidisciplinary design optimization (MDO) and PSE. Common engineering examples of these systems are process plants which are integrated and consist of a variety of processing units, where distributed optimization could be a natural way to optimize them.

Algorithms that are used to solve such decomposed problems are generally referred to as coordination algorithms. Augmented Lagrangian decomposition methods, also known as augmented Lagrangian coordination (ALC) (Allison et al., 2009), combine the idea of an augmented Lagrangian penalty method and a decomposition scheme. These methods optimize large systems by optimizing each sub-problem separately and coordinating the solution through a master problem.

The emphasis of this work is on presenting a strategy that efficiently decomposes problems into an optimal partition, minimizing the coupling between subsystems. The algorithm in this work optimally decomposes optimization problems with linear time complexity, while previous strategies present at least a polynomial time complexity. For completeness, coordination algorithms must be considered, and in this work the augmented Lagrangian method has been selected to demonstrate the applicability of the automated model decomposition algorithms presented.

Distributed optimization poses a coordination overhead which is related to the coupling of the subsystems (Allison et al., 2009). Ideally a partition should be found such that the numbers of variables and constraints that crossover between subsystems are minimized. By translating the problem to an equivalent graph representation, it is possible to take advantage of graph theory algorithms to find a structure in the underlying optimization problems.

A heuristic community detection algorithm is used to identify highly connected clusters in the optimization problem, or alternatively a multilevel graph bisection algorithm to find the optimal partition given a desired number of sub-problems to suit users preferences or constraints.

This work is organized as follows: section 2 presents the structure detection strategy proposed, section 3 presents augmented Lagrangian decomposition methods, section 4 shows the results of computational experiments, and finally section 5 presents conclusions and further work recommendation.

2. Structure Detection

The partitioning problem is the decision of how many sub-problems to decompose the original problem and which variables and constraints should be clustered into each of the sub-problems. The way the original problem is partitioned can have a profound effect on the computational solution requirements. Ideally, the partitioning decisions should minimize the complexity of the resulting distributed optimization problem (Allison et al., 2009). It is a logical consequence that the resulting system should consist of highly coupled sub-problems, with weak coupling across them.

Spectral partitioning methods are known to produce good partitions for a wide class of problems, and they are used quite extensively in literature (Pothen et al., 1990, 1992; Hendrickson and Leland, 1995). However, these methods are expensive since they require the computation of the eigenvector corresponding to the second smallest eigenvalue of the adjacency matrix (Fiedler vector)(Pothen et al., 1990, 1992; Hendrickson and Leland, 1995).

In general, partitioning methods based on spectral analysis have Singular Value Decomposition (SVD) at their core. This has the advantage of being a technique from linear algebra, which is less computationally demanding than other methods, especially for sparsely coupled systems, and is able to compute a partition in polynomial time. However, its computational cost is still of a magnitude of $\mathcal{O}\left(mn^2 + nm^2\right)$ (Drineas et al., 2004) with n, m in this case being respectively the number of variables and equations. For large problems this gives a high computational cost. Still, it has been used successfully in Sarkar et al. (2009) to automatically decompose optimization problems.

The procedures presented in this work automatically detect structure in optimization problems so as to determine the optimal number of sub-problems, and to allocate variables and constraints to each sub-problem practically in linear time. This procedure is described in the subsequent sections of this paper.

As a first step, the problem is translated into a graph. Optimization models are stated in the common form:

$$\min_{x} \quad f(x) \tag{1a}$$

subject to:

$$h(x) = 0 \tag{1b}$$

$$g(x) \leq 0 \tag{1c}$$

where $x \in \mathbb{R}^n$ is a vector of variables over which the system is optimized. The optimization problem in Equations 1a - 1c is mapped onto a constraint graph, where nodes correspond to variables and edges correspond to constraints on variables whose nodes they connect (Montanari, 1974), or when variables are coupled by a nonlinear function in the objective. In summary, an edge is added between two nodes when these two are coupled either by a constraint or the objective function.

2.1. Different Approaches to Problem Decomposition

When attempting to decompose an optimization problem there can be two general scenarios. In the first scenario a problem is to be decomposed and has a fixed known number of sub-problems, because of hardware constraints (*e.g.* number of microprocessors in a computing infrastructure or available memory) or other

constraints intrinsic to particular problems (*e.g.* geographical regions or other physical boundaries). In the second scenario the mentioned constraints might not exist and the user is free to decide the number of sub-problems. The fundamental difference is that the objective function of the partitioning methodology changes. In the first scenario the interest lies in finding the optimal partition for the given number of sub-problems. In the second scenario the interest is in the best possible sub-problem decomposition (including number of sub-problems), or even in testing whether the problem has a structure suitable for the chosen optimization strategy at all. Past work on methods for discovering groups in graphs is also divided along these two principal lines of research (Newman, 2006).

The first method examined here is called graph partitioning. Developed especially in computer science and related fields, it has mainly found application in parallel computing and integrated circuit design (Elsner, 1997). The second method considered is alternatively called community structure detection, block modeling or hierarchical clustering. This has been implemented in many fields, such as physics, applied mathematics, sociology and biology, with a particular focus on application in the latter two (Newman, 2004; White et al., 1976; Wasserman and Faust, 1994). While intuitively it could be said that they pursue similar objectives, we must stress the differences in the basic objectives of the two methods as they determine the desirability of their technical features.

As an example, with graph partitioning we have the typical problem of allocating tasks in a cluster of parallel processors with the objective of minimizing the communication load between them. Both the number of processors and the approximate magnitude of the capacity for each individual processor are known in advance, hence predetermining size and number of groups in the network. In this case the objective is to find the optimal network configuration. However, without a clear indication on whether such a configuration actually exists, a possibility that the method might fail remains.

In this case, we emphasize the comparison with the technique of community structure detection, which instead analyzes the structure of a large scale network such as internet data, social networks or biochemical reactions networks. The assumption is that, instead of having arbitrary divisions imposed upon it, the main network naturally divides into smaller subgroups of varying size and configuration that can be identified by the investigator. Moreover, community structure methods may explicitly admit the possibility that no good division of the network exists, an outcome that is itself considered to be of interest for the information it provides on the problem structure.

2.2. Community Detection and Modularity

A property that seems ideally suited for the purpose of decomposing an optimization problem is *community structure*, the division of network nodes into clustered groups within which the graph connections are dense, but between which they are sparser (Newman and Girvan, 2004). This translates into an optimization problem where each sub-problem has highly coupled internal variables and constraints, but sub-problems are loosely coupled amongst themselves.

Suppose that the structure of a graph is known, either because it is artificial or its natural structure has been discovered. We wish to determine whether its nodes can be divided into distinct non-overlapping communities of any size. Since it is based on sensible statistical principles and has proven effective in practice, maximization of modularity can be considered as the most desirable method to be adopted in current research for community detection (Newman, 2006).

Consider a particular division of a network into M communities. Define an $M \times M$ symmetric matrix E whose element E_{ij} is the fraction of all edges in the network that link nodes in community i to nodes in

community j. The trace of this matrix $\mathrm{Tr}E = \sum_i E_{ii}$ indicates what fraction of the network edges leads to nodes in the same community. This can be used as a measure of "how much" any node belongs to a certain community. However we should not consider the trace on its own, otherwise the system will degenerate into one single community thus making $\mathrm{Tr}E \to 1$, giving no information on the structure of the problem. To partially protect the structure of the system we add the row/column sums $a_i = \sum_j E_{ij}$, representing the fraction of edges that connect to nodes in community i. If we take any random sub-graph of the main network the expected value would be $E_{ij} = a_i a_j$. Previous research defined a modularity measure Q by

$$Q = \sum_i (E_{ii} - a_i^2) = \mathrm{Tr}E - \left\|E^2\right\| \tag{2}$$

where $\|\cdot\|$ indicates the sum of the elements of a matrix . This formula measures the fraction of the edges in the network that connect vertices of the same type (i.e. within-community edges), minus the expected value of the same quantity in a network with the same community divisions but random connections between the nodes.

The complexity of computing modularity exactly is NP-complete (Brandes et al., 2006), and calculating the best partition of the original problem into sub-problems could result in a difficult and computationally demanding problem in itself. Fortunately, several heuristic algorithms have been presented over the years, with the specific implementation in this work following from Blondel et al. (2008). This is an efficient algorithm with a linear complexity on typical and sparse data.

2.3. Multilevel Algorithm for Partitioning Graphs

Following the second approach mentioned in Section 2.1, given a constraint on the number of sub-problems we would like to divide the nodes of a graph into groups of roughly equal size, such that the number of edges connecting nodes in different groups is minimized. Hence this results in a graph partitioning problem (LaSalle and Karypis, 2013). Because the problem of graph partitioning is known to be NP-Complete (LaSalle and Karypis, 2013), directly solving it is not an option. Moreover, algorithms have been developed to find a reasonably good partition with low computational requirements.

Spectral partitioning methods are known to produce good partitions for a wide class of problems, and they are extensively used in published works (Pothen et al., 1990, 1992; Hendrickson and Leland, 1995). However these methods are computationally expensive, since they require the computation of the eigenvector corresponding to the second smallest eigenvalue of the adjacency matrix (Fiedler vector) as already pointed out in previous references.

Geometric partitioning algorithms (Heath and Raghavan, 1995; Miller et al., 1993, 1991) tend to be fast, but often yield partitions that are worse than those obtained by spectral methods, with larger and less well-defined communities. Among the most prominent geometric partition schemes is the algorithm described in Miller et al. (1991, 1993). Another class of graph partitioning algorithms coarsens the graph (i.e. reduces the graph) by merging together nodes and edges, then partitions the smaller graph and uncoarsens it to construct a partition for the original graph. These are called multilevel graph partitioning schemes (Cheng and Wei, 1991; Hagen and Kahng, 1992b,a; Garbers et al., 1990). These schemes tend to give good partitions at a reasonable computational cost. Hendrickson and Leland (1995) enhanced this approach by using edge and node weights to capture the collapsing of the vertices and edges. In particular, this latter work showed that multilevel schemes can provide better partitions than spectral methods at lower computational cost for a variety of finite element problems.

In multilevel approaches, a series of successively smaller graphs that approximate the original graph are generated before a partitioning of the smallest graph is made. This is then applied to each successively larger graph with some optimization, until it is finally applied to the original graph (LaSalle and Karypis, 2013). These algorithms solve the underlying optimization problem using a methodology that follows a *"simplify and conquer"* approach, initially used by multi-grid methods for solving systems of partial differential equations (LaSalle and Karypis, 2013).

Multilevel partitioning methods consist of three distinct phases as outlined in Karypis and Kumar (1998) :

1. Coarsening phase: the original graph G_0 is used to generate a series of increasingly coarser graphs, $G_1, G_2, ..., G_m$.
2. Initial partitioning phase: a partitioning P_m of the much smaller graph G_m is generated using a more computationally expensive algorithm.
3. Uncoarsening phase: the initial partitioning is used to derive partitions of the successive finer graphs. This is done by first projecting the partition of G_{i+1} to G_i, followed by partitioning refinement whose goal is to reduce the edge-cut by moving nodes among the partitions. Since the successive finer graphs contain more degrees of freedom, such refinement is often feasible and leads to significant edge-cut reductions.

It is important to notice that although this method may not be able to give as much information about the structure of optimization problems as community detection, it has other significant benefits. Mainly it creates problems of similar sizes, a feature crucial to exploiting memory reduction and global optimization, advantages for which augmented Lagrangian decomposition and other distributed optimization methods are known.

For graph partitioning in this work a highly efficient package, Metis (Karypis and Kumar, 1998), was used in the implementation. Metis builds on the work of Hendrickson and Leland (1995), however it highly enhances all three phases (coarsening, partition of the coarsest graph and refinement), compared to other previous work. This is a state-of-the-art algorithm with a linear complexity. Full derivation and the algorithm can be reviewed in Karypis and Kumar (1998); LaSalle and Karypis (2013).

2.4. Linear Complexity of the Decomposition Algorithms

The algorithms presented in Sections 2.2 and 2.3 can find communities or partitions in linear time with respect to the number of nodes v and the number of edges e of a graph. However, this needs not be the same as for variables and constraints in the equivalent optimization problem.

In an optimization problem where the number of variables is n and the number of constraints is m, if we consider the problem mapped to the structure of a graph with constraints implemented as edges, then the number of edges mapped from the optimization problem to its equivalent graph would be

$$e = ((n-1) + (n-2) \dots (n-(n-2)) + 1) = \left(\frac{1}{2}n^2\right) \rightarrow \mathcal{O}\left(n^2\right)$$

per constraint in the worst case. If this is then multiplied by the number of constraints the total number of edges would be $\mathcal{O}\left(mn^2\right)$. This would not be computationally advantageous over traditional decomposition methods.

However, from the practical point of view, the number of variables n represented in these costs is not the same as all the number of variables in the optimization problem seen in previous complexity estimates. It

represents only the number of variables that are present in a specific constraint, n_{c_j} for $j = 1, ..., m$. In realistic large systems for which $m \sim 10^3 - 10^5$, it is highly unlikely that each of these constraints will involve all n variables and having $n >> n_{c_j}$ per constraint is often the norm. The method presented has to deal computationally only with a number of variables at a time that is less than the full set. The complexity is dependent in the number of variables n_{c_j} in each constraint j, adding one edge between each two variables in a constraint.

$$\mathcal{O}\left(\frac{1}{2}n_{c_0}^2 + \frac{1}{2}n_{c_1}^2 + \ldots \frac{1}{2}n_{c_m}^2\right) \tag{3}$$

Not knowing the distribution followed by the number of variables per constraints, the complexity on equation 3 is bounded from above by m times the constraint with the most variables n_{max}

$$\mathcal{O}\left(\frac{1}{2}n_{c_0}^2 + \frac{1}{2}n_{c_1}^2 + \ldots \frac{1}{2}n_{c_m}^2\right) \leq \tag{4}$$

$$\mathcal{O}\left(m\ \frac{1}{2}n_{max}^2\right) = \tag{5}$$

$$\mathcal{O}\left(m\ n_{max}^2\right) \tag{6}$$

As the distribution followed by the number of variables per constraints in unknown, Equation 6 is a loose upper bound for the complexity on the algorithms presented here. Furthermore, decomposition is generally practiced in sparse systems where the number of variables per constraint is small. In these cases the complexity will be solely due to the number of constraints accompanied by a constant term related to the number of variables in each constraint:

$$\mathcal{O}\left(m\ n_{max}^2\right) \rightarrow \mathcal{O}\left(m\right)$$

The pre-processing step of constructing the constraint graph can also be considered, although it is required only once per execution of the method. In the way that most software packages manage equations, each constraint is already parsed as a sub-tree. Exploration is thus aimed at establishing whether two variables x_i and x_k, with $i \neq k$, are linked within each constraint, which is necessary for the reformulation. In the worst case, a degenerate tree where all nodes are in a row is obtained. This is at most linear in the number of variables involved in the constraint as in $\mathcal{O}\left(n_c\right)$. In the best case of a perfectly balanced tree, this is at most equal to the height of the tree, so the cost will be $\mathcal{O}\left(\log n_c\right)$ (Bollobás, 1998). In many cases the complexity will be a value in between the two extreme cases.

Without knowing if the tree structures of the constraints follow a specific distribution, assuming the worst case, the computational complexity is $\mathcal{O}\left(n_c\right)$ per constraint. This is then multiplied by the number of constraints. If we call $E\left[n_c\right]$ the expected number of variables per constraint, which is a bounded constant unique to the problem being studied, then in a realistically large problem the complexity of this pre-processing step would be $\mathcal{O}\left(E\left[n_c\right]\ m\right) = \mathcal{O}\left(m\right)$.

Therefore, if the following assumptions are true:

- The original algorithms can decompose a graph in linear time.
- The average number of variables per constraint is a small constant when compared to the total number of variables.

This analysis and discussion shows that decomposing an optimization problem through one of the methods outlined in previous sections can be effectively done in linear time.

2.5. Modular simulation

The techniques presented in the course of this work can not only be applied to the decomposition of optimization problems, but also to the area of process simulation. This would result in the simulated system divided into discrete modules, referred to as modular simulation. In the present section, we define modular simulation as any form of simulation in which we have an approach where the equations and variables determining a system are partitioned into discrete sets (Hillestad and Hertzberg, 1986). These partitions are usually state phases or physical sub-divisions (Hillestad and Hertzberg, 1986). It should be clarified that the definition of modularity used in the current section is different (although possibly related) from that shown in section 2.2. The modularity defined in the current section is a qualitative assessment rather than a quantitative measure.

An example of modular simulation could be a distillation column with n stages which can be considered as an ensemble of n models of a tray, populated by identical equations, with a model for the boiler, a model for the condenser and other auxiliary items. This particular model could be further divided in gas, liquid and solid phases. In Hillestad and Hertzberg (1988) these models are defined as orthogonal equations, in opposition to normal equations which instead describe the "flows" of the system. In such an example, the rationale for the division in modules is based on roughly identifying process units and grouping them into modules (Aspentech, 2015; Process Systems Enterprise, 2015). However the above rationale is not the only possible method: the strategies presented in this work can be used to find automatically modules in a flowsheet, or other system comprised of several sub-systems.

Much of this procedure starts from manual identification of subdivisions in a system. There is available literature that details how the modules should be built mathematically, as in Chen and Adomaitis (2006), or what the optimal ordering is such as in Fagley and Carnahan (1990), or how to ensure that the modules produce stable and convergent solutions (Chen and Adomaitis, 2006). Many of these sources also consider making larger groupings of individual process units into modules (Pidd and Castro, 1998) to improve the speed and efficiency of dynamic simulation for the heavily interlinked parts of a process. Despite the comprehensive studies mentioned, as far as the authors are aware of, there is no research focusing on developing the methods that determine which variables and parts of a real system would be assigned in a certain module in the field of chemical engineering. Therefore, the current study proposes to use the techniques here presented as tools for the automation of modular decomposition.

To assess the feasibility of applying the current algorithms to modular simulation in chemical engineering, it is necessary to understand the requirements of modularity in the context of modular simulation. Two properties, named locality and encapsulation, have been previously proposed by Zeigler (1987) and Cota and Sargent (1992), respectively. Locality (Cota and Sargent, 1992) means that the values and structures necessary to solve a system are available locally. Encapsulation, in parallel to programming concepts, is defined as being able to hide from external manipulation internal variables outside of normal interactions through the stated inputs and outputs (Zeigler, 1987). It then follows that modularity must necessarily encompass both of these properties.

It can be argued that the methods presented in this work possess properties analogous to the above two.

1. (a) A community identified through community detection or multilevel partitioning possesses encapsulation. Most of the variables are hidden inside the community, and connected only with other variables

belonging to the same community. These variables can only be influenced indirectly from the outside, through the minority of variables that are directly linked to the members in others communities.

2. (b) The same reasoning also allows to argue for locality. Variables that have a strong connection to one another have a high probability that they will be assigned to the same sub-graph. If they are not in the same community, they can be considered either as inputs/outputs for the connection between the two communities they belong to. It could also be stated that locality is present by necessity, because the objective of the partitioning methods is to obtain subproblems that are able to be solved independently.

Consequently this approach could be applied to model simulation as it encompasses the necessary properties of modular decomposition.

3. Augmented Lagrangian Decomposition Methods

Augmented Lagrangian decomposition methods are effective algorithms for distributed optimization. Different algorithms have been developed throughout the years (Watanabe and Ntshimura, 1978; Tappenden et al., 2015; Powell, 1969; Hamdi and Al-saud, 2008; Hamdi, 2005; Hamdi et al., 1997), which present different approaches to decompose the original problem and coordinate the solution. In this paper the algorithm used follows from the Separable Augmented Lagrangian Algorithm (SALA) family of methods (Hamdi et al., 1997; Hamdi, 2005; Hamdi and Al-saud, 2008). Reviews on different augmented Lagrangian decomposition methods can be found in Hamdi et al. (1997); Tosserams et al. (2009).

The general augmented Lagrangian decomposition method is the following, where $\zeta = [y, x_1, ..., x_M]$:

$$\min_{\zeta} \quad \sum_{j=1}^{M} f_j(y, x_j) \tag{7a}$$

subject to:

$$h_j(y, x_j) = 0 \qquad j = 1, ..., M \tag{7b}$$

$$g_j(y, x_j) \leq 0 \qquad j = 1, ..., M \tag{7c}$$

where M is the number of subsystems, the variable vector ζ consists of the linking (global) variables y (those belonging to two or more subsystems), and the local (private) variables x_j belonging to subsystem j. The local objective function f_j and local constraints g_j and h_j belong only to subsystem j and may depend on the linking variables y as well as to the local variables x_j.

3.1. Decomposition of the original problem

The following steps are taken to decompose the original problem in Equations 7a-7c:

1. Coupling constraints through the linking variables y are removed and a separate copy y_j is introduced in each subsystem j, such that the local constraints g_j and h_j depend on the copy y_j. In this way for any pair of subsystems j and l, with $l \neq j$ the condition $y_j = y_l$ holds. This also implies that the local constraints are dependent on these linking variables.
   This concept is extended beyond pairing of variables into a generalized simultaneous neighborhood consistency check. The copies are enforced to be equal by consistency constraints H, which are defined

as a collection of constraints H_{jn} between subsystem j and its neighbors $n \in N_j$ where N_j is the set of neighbors:

$$H_{jn} = \sum_{i=1}^{2} v_i(y_i) = y_j - y_n \tag{8a}$$

where

$$H_{jn} = y_j - y_n = 0, \qquad n \in N_j | n > j, \qquad j = 1, ..., M \tag{8b}$$

The neighbors N_j are defined as the subsystems to which subsystem j is linked to through the consistency constraints. The condition $n > j$ ensures that only one of the equivalent pairs H_{jn} and H_{nj} is included in the consistency constraints as $H_{jn} = H_{nj}$ by definition.

2. Each sub-problem is allocated its own variables, and consistency constraints are introduced to enforce coupling of the sub-problems.

$$\min_{x_j, y_j} \quad f_j(x_j, y_j) \tag{9a}$$

subject to:

$$v_j(y_j) + w_j = 0 \tag{9b}$$

$$h_j(y_j, x_j) = 0 \tag{9c}$$

$$g_j(y_j, x_j) \leq 0 \tag{9d}$$

where the w_j variables enforce the link between the subsystems by taking on the value $v_n(y_n)$, which is the value the variable in the neighboring sub-problem on the previous iteration.

3. Next, relaxation of the consistency constraints is introduced as the vector of all linking constraints using an augmented Lagrangian penalty (Lenoir and Philippe, 2007; Hamdi et al., 1997; Hamdi and Mahey, 2000)

$$L(x, y, c, \lambda, w) = \sum_{j=1}^{M} f_j(x_j, y_j) + \sum_{j=1}^{M} \lambda_j (v_j(y_j) + w_j) + \frac{1}{2} c \sum_{j=1}^{M} (v_j(y_j) + w_j)^2 \tag{10}$$

where λ are Lagrange multipliers and c is a penalty parameter. After relaxation of the linking constraints, subsystems have separable constraint sets, and are no longer coupled.

4. The next step of decomposition is to define a sub-problem P_j for each subsystem. The augmented Lagrangian decomposition algorithms optimize for the variables of each subsystem separately. Sub-problem P_j therefore only includes the terms that depend on its own variables$\hat{x}_j = [x_j, y_j]$, and is given by Equation 11a

$$\min_{\hat{x}_j} \quad f_j(x_j, y_j) + \lambda_j (v_j(y_j) + w_j) + \frac{1}{2} c \left\| v_j(y_j) + w_j \right\|_2^2 \tag{11a}$$

subject to:

$$h_j(y_j, x_j) = 0 \tag{11b}$$

$$g_j(y_j, x_j) \leq 0 \tag{11c}$$

The solution to sub-problem P_j depends on the solution of the other sub-problems $P_i | i \neq j$. A coordination algorithm is necessary to account for this coupling.

There are other approaches to augmented Lagrangian decomposition that propose to have global constraints when these involve a large number of problem variables, *e.g.* Tosserams et al. (2010). In this case the decomposition strategy remains the same, except that it simply removes those constraints from the constraint graph. The global constraints are then added after allocating the variables and other constraints to the sub-problems. Examples are presented in Section 4.

3.2. Coordination algorithm

To coordinate the solution from the sub-problems the algorithm SALA (Hamdi and Mahey, 2000) has been implemented in this work. The algorithm is the following:

Algorithm 1 SALA

1. Initialize: $x^0, y^0, w^0 \in A$. with $A = \left\{(w_1, ..., w_M) \mid \sum_{i=1}^{M} w_i = 0\right\}$, $\lambda > 0, k = 0, \varepsilon > 0, \beta \geq 2$
2. Determine: $\forall\, j \in 1, \ldots, M$

$$\hat{x}_j^{k+1} = \arg\min_{\hat{x}_j} \quad \left(f_i(\hat{x}_j) + \lambda_j^k (v_j(y_j) + w_j^k) + \frac{1}{2} c^k \left\| v_j(y_j) + w_j^k \right\|_2^2 \right)$$

$$\hat{x}_j = [x_j, y_j] \in S_j = \mathbb{R}^{n_j} \left\{ \begin{array}{c} g_j(y_j, x_j) \leq 0 \\ h_j(y_j, x_j) = 0 \end{array} \right\}$$

3. Calculate $H \mid H = \sum_{j=1}^{M} H_{jn} \quad n \in N_j | n > j$.

   If: $\|H\| < \varepsilon$ stop.

   Else: go to step 4.

4. Update parameters w, c, λ

$$\left\{ \begin{array}{c} \lambda^{k+1} = \lambda^k + \frac{c^k}{M} H^{k+1}, \\ w_j^{k+1} = -v_j(y_j) + \frac{H^{k+1}}{M} \\ c^{k+1} = c^k \frac{\beta}{2} \end{array} \right. .$$

   Go to step 2.

Full algorithm derivation and convergence analysis can be found in Hamdi and Mahey (2000).

At present, decomposition algorithms have a similar complexity (polynomial) to that of coordination algorithms. This represents a serious overhead to distributed optimization and hence limits its applicability. As it is noted in subsection 2.1 current decomposition strategies make use of SVD, as a consequence, the resulting complexity is polynomial in both n and m. In contrast, the algorithm here presented maps the optimization problem onto a constraint graph. Due to this mapping, it is possible to use algorithms developed from graph theory to find structure in the original optimization problem. It is demonstrated in Section 4 of this work that it is possible to decompose an optimization problem in linear time through the use of this framework.

4. Results

4.1. Numerical experiments setup

We investigate seven test problems in this paper. The first five examples shown in subsection 4.2 were taken from Tosserams et al. (2010) and Mezura-Montes and Coello Coello (2005). They are chosen so as to exemplify how the methodology presented in this work decomposes problems into a convenient configuration for distributed optimization. In these cases we do not focus on the coordination of the solution, as very detailed solutions and their analysis by augmented Lagrangian decomposition methods are presented in Tosserams et al. (2010) and Mezura-Montes and Coello Coello (2005). The other two examples in subsection 4.3 aim to show the decomposition and distributed optimization of problems that could represent real chemical processes. These problems are solved by the coordination algorithm SALA (Hamdi and Mahey, 2000).

The implementation in this work is programmed in a Python environment. Sub-problems where solved in Pyomo (Hart et al., 2012), a tool package for modeling optimization applications in Python and which serves as an interface for the optimization solver IPOPT (Wächter and Biegler, 2006), used as a library in Pyomo.

4.2. Decomposition problems

In this subsection five problems are decomposed with the purpose of testing the efficacy and performance of the algorithms presented in this work.

4.2.1. Example 1

This problem, taken from Tosserams et al. (2010), consists of 14 variables, 4 inequality constraints, and 3 equality constraints. The problem is nonlinear and non-convex, given by:

$$\min_{z_1,\ldots,z_{14}} \quad \sum_{i=1}^{14}(z_i-3)^2 \tag{12a}$$

subject to:

$$g_1 = (\sin(z_3^{-2}) + z_4^2)z_5^{-2} - 1 \leq 0 \tag{12b}$$

$$g_2 = (z_8^2 + z_9^2)z_{11}^{-2} - 1 \leq 0 \tag{12c}$$

$$g_3 = (z_{11}^2 + \sin(z_{12}^{-2}))z_{13}^{-2} - 1 \leq 0 \tag{12d}$$

$$g_4 = (z_5^2 + z_6^{-2})z_7^{-2} + (z_8^{-2} + z_{10}^2)z_{11}^{-2} + (z_{11}^2 + z_{12}^2)z_{14}^{-2} - 3 \leq 0 \tag{12e}$$

$$h_1 = (z_3^2 + z_4^{-2} + z_5^2)z_1^2 - 1 = 0 \tag{12f}$$

$$h_2 = (\sin(z_{11}^2) + z_{12}^2 + z_{13}^2 + z_{14}^2)\sin(z_6^2) - 1 = 0 \tag{12g}$$

$$h_3 = (z_5^2 + z_6^2 + z_7^2)z_2^2 + (z_8^2 + z_9^{-2} + z_{10}^{-2} + z_{11}^2)z_3^{-2} - 2 = 0 \tag{12h}$$

$$0.1 \leq z_i \leq 5, \qquad i = 1, ..., 14 \tag{12i}$$

The problem represented in the equations 12a-12i is mapped into a constraint graph, and the community detection algorithm is used to find optimal partition strategies. Constraints g_4 and h_3 have been removed as in the example from Tosserams et al. (2010) so that an accurate comparison can be made. This is presented in Figure 1.

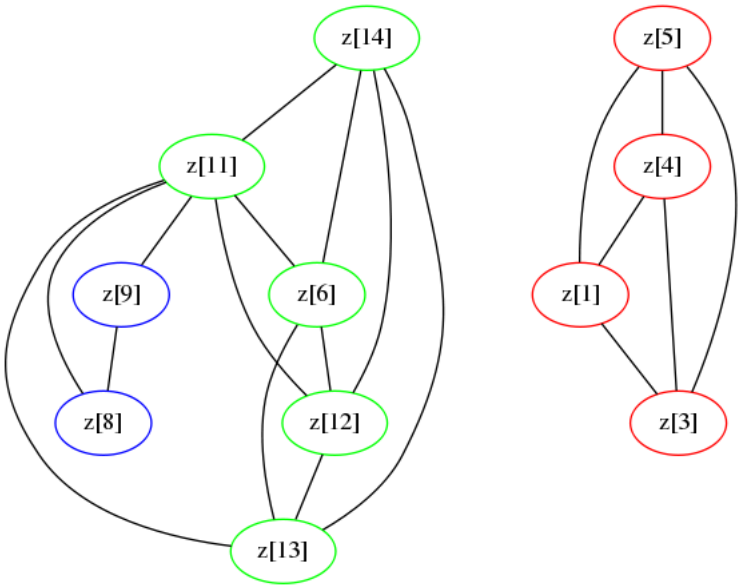


Figure 1: Example 1 partitioning

Given the partitioning by community detection in Figure 1 constraints and variables are allocated and yield the same partition as presented in Tosserams et al. (2010) shown in Figure 2. Note that variables z_2, z_7, z_{10} do not appear in the constraint graph in Figure 2, as they are not bound by any constraint. In graph theory terms, they are isolated vertices and effectively belong to any component graph independent from the measures of their distance to the other vertices.

When the multilevel partitioning algorithm is used to decompose the problem into three sub-problems, the partitions are not the same as those shown in Figure 1. This is because the partitioning algorithm creates sub-problems that have to be the same size, even at the cost of a worse partitioning, forcing the inclusion of non-optimal nodes into the residual sub-graphs. When two extra nodes are added, which we will call *slack nodes*, the resulting partitioned problem becomes the same as the one obtained through the community detection algorithm. From this observation we propose using a small number of *slack nodes* when partitioning systems, so that the best partition is obtained with similar sized sub-problems, instead of a bad partition with equally sized problems. In this way a relaxation of the system can also be obtained in the context of graphs. The number of slack nodes to obtain a better partition was chosen empirically and the criteria by which the algorithm should determine this number remains an issue that requires further work.

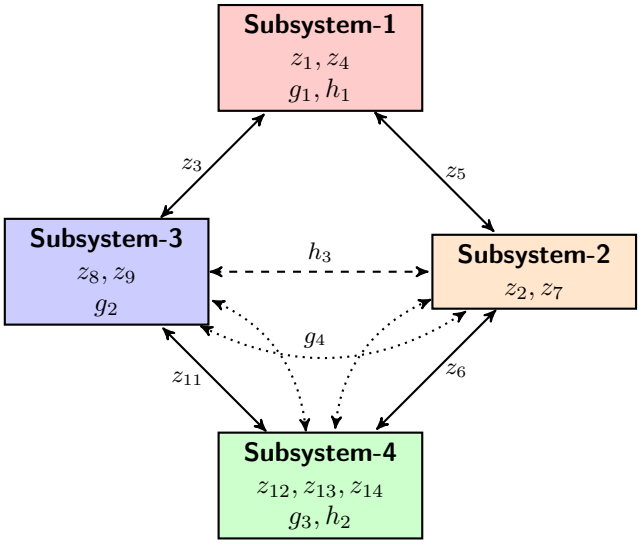


Figure 2: Decomposition of example 1(after Tosserams et al. (2010))

4.2.2. Example 2

The problem is nonlinear and non-convex, given by Tosserams et al. (2010):

$$\min_{z_1,\ldots,z_{2m}} \quad \sum_{i=1}^{m} 2z_i - \sum_{i=m+1}^{2m} z_i \tag{13a}$$

subject to:

$$g_i = -\left(z_i + a_i\sqrt{\frac{1}{m}}\right)\left(z_i + b_i\sqrt{\frac{1}{m}}\right) \leq 0, \quad i = 1, ..., 2m \tag{13b}$$

$$g_{2m+1} = \sum_{i=1}^{2m} z_i^2 - 2 \leq 0 \tag{13c}$$

$$h_i = z_i - z_{i+m} = 0, \qquad i = 1, ..., 2m \tag{13d}$$

$$-2 \leq z_i \leq 2, \qquad i = 1, ..., 2m \tag{13e}$$

where m can be adjusted to determine the number of variables and constraints. The parameters $0 \leq a_i,\ b_i \leq 1$ determine the feasible domain of the problem. For this example constraint g_{2m+1} was also removed so that decomposition results could be comparable with those in Tosserams et al. (2010).

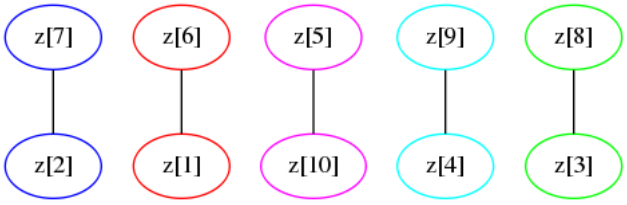


Figure 3: Example 2 partitioning

The community detection algorithm for $m = 5$ yielded the m-partite graph in Figure 3. In Tosserams et al. (2010) a decomposition with $2m$ sub-problems is carried out. This decomposition into a number of sub-problems can represent a decomposition caused by physical or other constraints, and the best partition is the one presented in Figure 3, with m sub-problems.

The multilevel partitioning yield the same graph as the one obtained by the community detection algorithm. Both of these methods yield the graph shown in Figure 3.

4.2.3. Example 3

The problem has 4 variables, 1 equality constraint, and 1 inequality constraint. The objective and equality constraint are both non-convex. The problem is given by Tosserams et al. (2010):

$$\min_{z_1,z_2,z_3,z_4} \quad 4\left(z_1 - \frac{3}{2}\right)\left(z_3 - \frac{1}{2}\right) - (z_2 - 1)^2 - 2(z_4 - 1)^2 \tag{14}$$

subject to:

$$g = -2z_1 - z_2 + \frac{3}{4} \leq 0 \tag{15a}$$

$$h = -z_3 z_4 + 1 = 0 \tag{15b}$$

$$0 \leq z_i \leq 2, \qquad i = 1, ..., 4 \tag{15c}$$

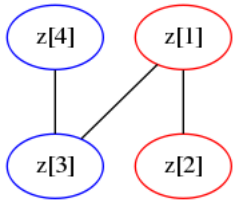


Figure 4: Example 3 partitioning

This simple example is decomposed by the community detection as shown in Figure 4 and results in the same decomposition as presented in Tosserams et al. (2010). The multilevel partitioning algorithm demonstrates the same performance for two sub-problems and yields the same graph as the community detection method.

In the following representation the objective function is decomposed into the parts

$$\begin{aligned} F_1 =& 4\left(z_1 - \frac{3}{2}\right)\left(z_3 - \frac{1}{2}\right) \\ F_2 =& -(z_2 - 1)^2 \\ F_3 =& -2(z_4 - 1)^2 \end{aligned}$$

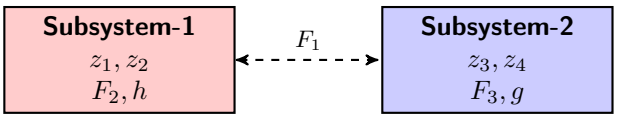


Figure 5: Decomposition of example 3(after Tosserams et al. (2010))

4.2.4. Example 4

This problem, taken from Tosserams et al. (2010), has 12 variables, a concave objective function, and 9 linear inequality constraints. This is a minimization, with a parabolic concave objective function and linear constraints.

$$\min_{z_1,\ldots,z_{12}} \quad \sum_{i=4}^{12}(5z_i - 5z_i^2) + \sum_{i=5}^{12}(-z_i) - 3 \tag{16a}$$

subject to:

$$g_1 = 2z_1 + 2z_2 + z_{10} + z_{11} - 10 \leq 0 \tag{16b}$$

$$g_2 = 2z_1 + 2z_3 + z_{10} + z_{12} - 10 \leq 0 \tag{16c}$$

$$g_3 = 2z_2 + 2z_3 + z_{11} + z_{12} - 10 \leq 0 \tag{16d}$$

$$g_4 = -8z_1 + z_{10} \leq 0 \tag{16e}$$

$$g_5 = -8z_2 + z_{11} \leq 0 \tag{16f}$$

$$g_6 = -8z_3 + z_{12} \leq 0 \tag{16g}$$

$$g_7 = -2z_4 - z_5 + z_{10} \leq 0 \tag{16h}$$

$$g_8 = -2z_6 - z_7 + z_{11} \leq 0 \tag{16i}$$

$$g_9 = -2z_8 - z_9 + z_{12} \leq 0 \tag{16j}$$

$$0 \leq z_i \leq 3, \quad i = 1, ..., 12 \tag{16k}$$

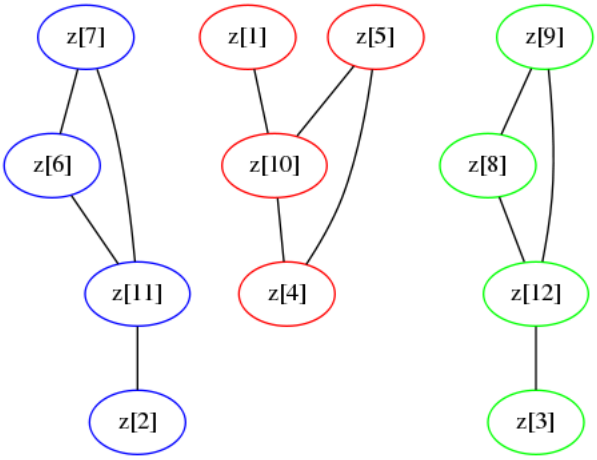


Figure 6: Example 4 partitioning

This example is decomposed by the community detection as show in Figure 6 and schematically in Figure 7. It results in the same decomposition as presented in Tosserams et al. (2010).

The multilevel partitioning algorithm for 3 sub-problems yields the same graph as the one shown in Figure 6, the same as the community detection method. Components in the resulting graph are not actually connected so it is easy to verify that the correct partition of the system is obtained.

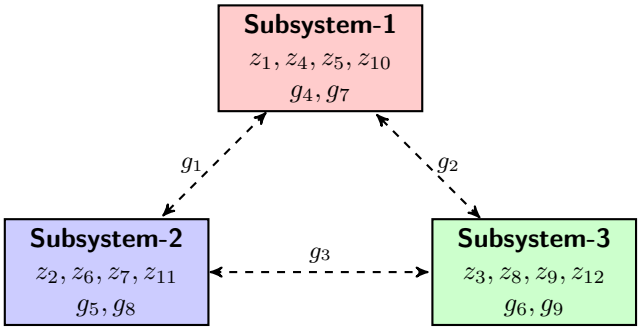


Figure 7: Decomposition of example 4 after Tosserams et al. (2010)

4.2.5. Example 5

This is a 10 variable nonlinear constrained optimization problem from Mezura-Montes and Coello Coello (2005).

$$\min_{z_1,\ldots,z_{10}} \quad z_1^2 + z_2^2 + z_1 z_2 - 14z_1 - 16z_2 + 45 + (z_3 - 10)^2 + 4(z_4 - 5)^2 + (z_5 - 3)^2 + 2(z_6 - 1)^2 + 5z_7^2 + 7(z_8 - 11)^2 + 2(z_9 - 10)^2 + (z_{10} - 7)^2 \tag{17a}$$

subject to:

$$g_1 = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0 \tag{17b}$$

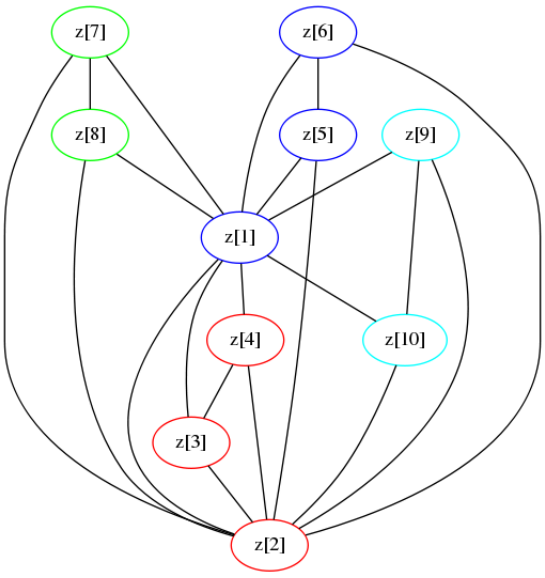


Figure 8: Example 5 partitioning

$$g_2 = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0 \tag{17c}$$

$$g_3 = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0 \tag{17d}$$

$$g_4 = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0 \tag{17e}$$

$$g_5 = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0 \tag{17f}$$

$$g_6 = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0 \tag{17g}$$

$$g_7 = \frac{1}{2}(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0 \tag{17h}$$

$$g_8 = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0 \tag{17i}$$

$$-10 \leq x_j \leq 10,\ i = 1, 2, \ldots, 10 \tag{17j}$$

This example is decomposed by the community detection algorithm as show in Figure 8 and results in the same decomposition as presented in Dormohammadi and Rais-Rohani (2013).

The multilevel partitioning algorithm for four sub-problems yields the same graph as the one shown in Figure 8. The most interesting feature here is the correct identification of the community for the node representing variable z_1, which has an edge toward every other node in the system, as it can be understood from the

structure of the problem. In the schematic presented in Figure 9 the decomposition of the objective function is the following

$$
\begin{aligned}
f_1 =& z_1^2 + z_2^2 + z_1 z_2 - 14 z_1 - 16 z_2 + 45 \\
f_2 =& (z_3 - 10)^2 + 4(z_4 - 5)^2 \\
f_3 =& (z_5 - 3)^2 + 2(z_6 - 1)^2 \\
f_4 =& 5 z_7^2 + 7(z_8 - 11)^2 \\
f_5 =& 2(z_9 - 10)^2 + (z_{10} - 7)^2
\end{aligned}
$$

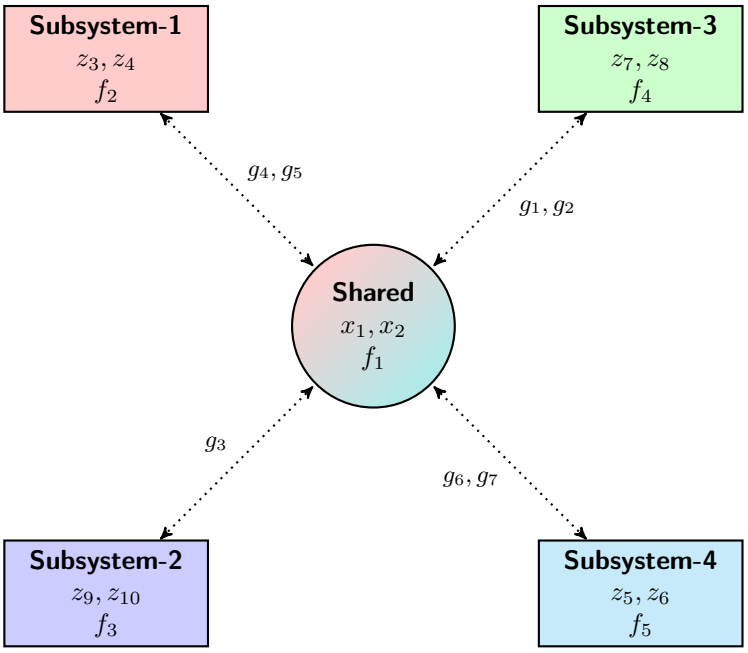


Figure 9: Decomposition of example 5

4.3. Decomposition-Optimization problems

In this subsection larger problems than those previously presented are investigated. The size of the systems tested is closer to realistic engineering problems. These examples are decomposed and later optimized through the Augmented Lagrangian decomposition algorithms.

4.3.1. Example 6

A chemical reactor network problem from Floudas et al. (1989) is considered. 5 different sub-reactor networks are used to create a larger reactor network in an effort to simulate an industrial system. The general formulation below has been taken from Schweiger and Floudas (1997). It is generalized for any number of the following items

1. CSTR units
2. Components
3. Reaction paths

4. Feed streams
5. Product streams

This problem deals with determining the types, sizes, and interconnections of reactors which optimize a desired performance objective. The superstructure-based approaches propose a network of reactors which contain all the possible design alternatives of interest. The solution of the resulting optimization problem then determines the optimal structure of the flowsheet. The reactor network consists of mixers, splitters and CSTRs. Splitters are used to split the feed and the outlet of each reactor. Mixers are situated at the inlet to each reactor to mix the streams which come from the feed splitter and the splitters after each reactor. A final mixer is used to mix the streams which go from the splitter after each reactor unit to the product stream. A flowsheet which has two CSTRs is shown in Figure 10. The formulation, slightly adapted from the original source, can be found in the supplementary materials.

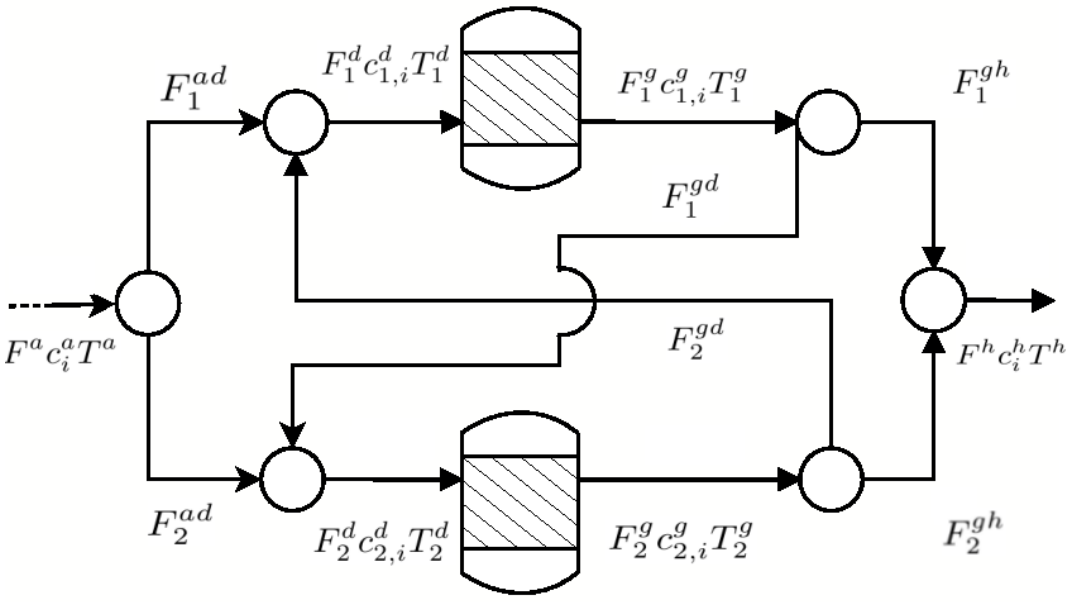


Figure 10: Reactor network consisting of two CSTRs

The reactor network superstructure considered in this work is shown in Figure 11.

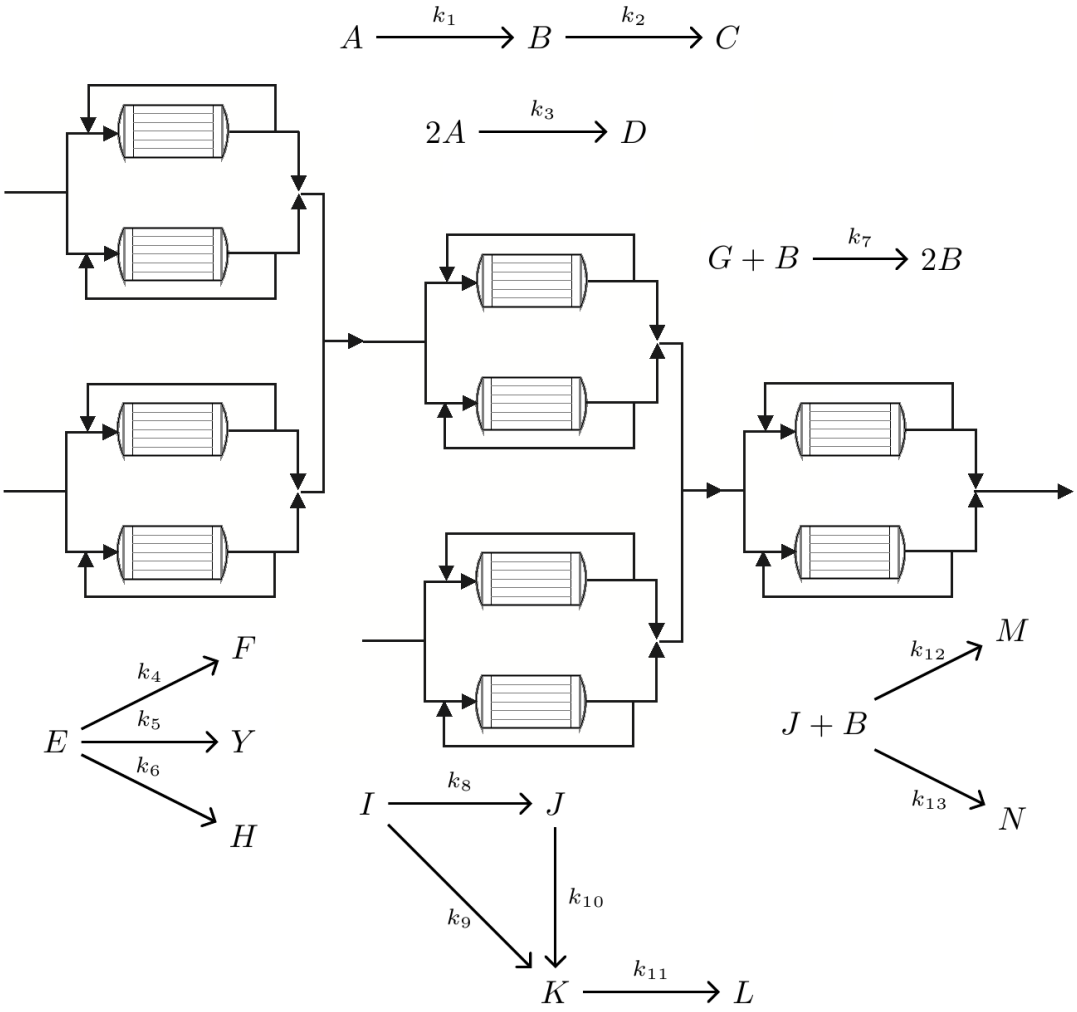


Figure 11: Reactor network superstructure; each sub-reactor network consists of more than 2 reactors in parallel, with only 2 shown for each block for clarity.

The reactor network problem represented by the equations in the supplementary materials is mapped into a constraint graph. The community detection and multilevel graph partitioning algorithms found the same optimal partition strategies shown in Figure 12. As expected, the algorithms found that each sub-reactor network should be grouped into a separate sub-problem, and then the solution of the whole reactor network system should be coordinated to reach the optimal solution of the original problem.

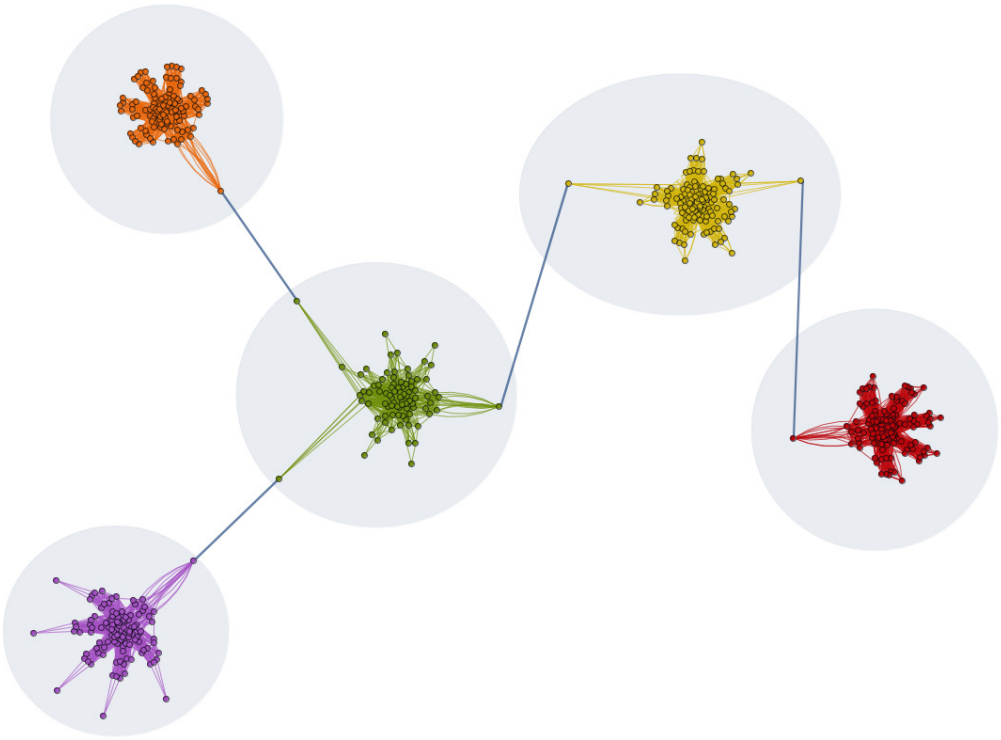


Figure 12: Partition of reactor network

SALA was used as the coordination algorithm to solve the resulting decomposed problem, this is shown in Table 1.

Table 1: SALA algorithm on the Reactor Network

Decomposition Statistics	
Method	CPU Decomposition time (s)
Multilevel Graph Partitioning	0.003
Community Detection	0.006
Solution Statistics	
Number of iterations	76
final objective value	8.09
Number of subproblems	5
CPU solution time(s)	26.70

From Table 1 it can be observed that the decomposition time is negligible compared to the solution time. Furthermore, although the multilevel graph partitioning method seems to outperform the community detection algorithm in terms of CPU time, this difference is insignificant and might be more due to the implementation of the packages used than to the actual algorithms. The objective value obtained is the same as the best one reported in the literature.

4.3.2. Example 7

In this problem an incoming stream of nitrogen and hydrogen is converted to ammonia (Rase, 1977; Passas-Garrido, 2012). There are a number of different commercial reactor designs in use, but the one used on the

plant that was the basis for this work is the Kellogg reactor. One specific implementation of the Kellogg reactor is described in Noe (1987). The reaction is undertaken in a number of reactors in series since the maximum conversion in any given reactor is equilibrium-limited. Inter-coolers are strategically placed between the reactors to shift the equilibrium temperature and hence to give further opportunity to increase overall conversion. One possible process flowsheet for the ammonia reactor is shown in Figure 13.

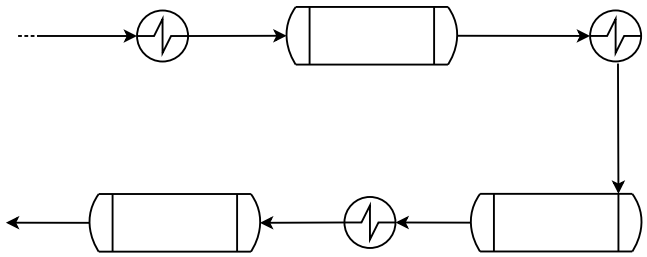


Figure 13: Process flowsheet for the Kellogg reactor

The objective of the optimization is to maximize the production of ammonia, given the existing constraints. A full problem description is provided in the supplementary material. The model divides each reactor into 10 volume elements using orthogonal collocation. According to source literature (Rase, 1977), the kinetic expressions calculate the rate of production of ammonia per unit volume of packed bed; this is assumed to mean the total bed volume as opposed to the catalyst volume. It is assumed that the energy released during the reaction only heats the gas phase, which differs from the more complex assumption that it also includes other contributions such as the latency caused by the liquid phase. Since the reaction kinetics were highly sensitive to temperature, the molar heat capacity used is tuned to minimize the error in temperature prediction between the two models (Rase, 1977; Passas-Garrido, 2012). The pressure drop across each packed bed is interpolated from UniSim's™ predictions of pressure drop through a packed bed calculated from the Ergun equation. It is assumed that a 40 kPa pressure drop occurs over each heat exchanger in the flowsheet. The data for the ammonia reactors is shown in the supplementary materials.

This problem is mapped into its equivalent constraint graph, the community detection and multilevel graph partitioning algorithms (for the same number of divisions) find the same optimal partition strategy, shown in Figure 14.

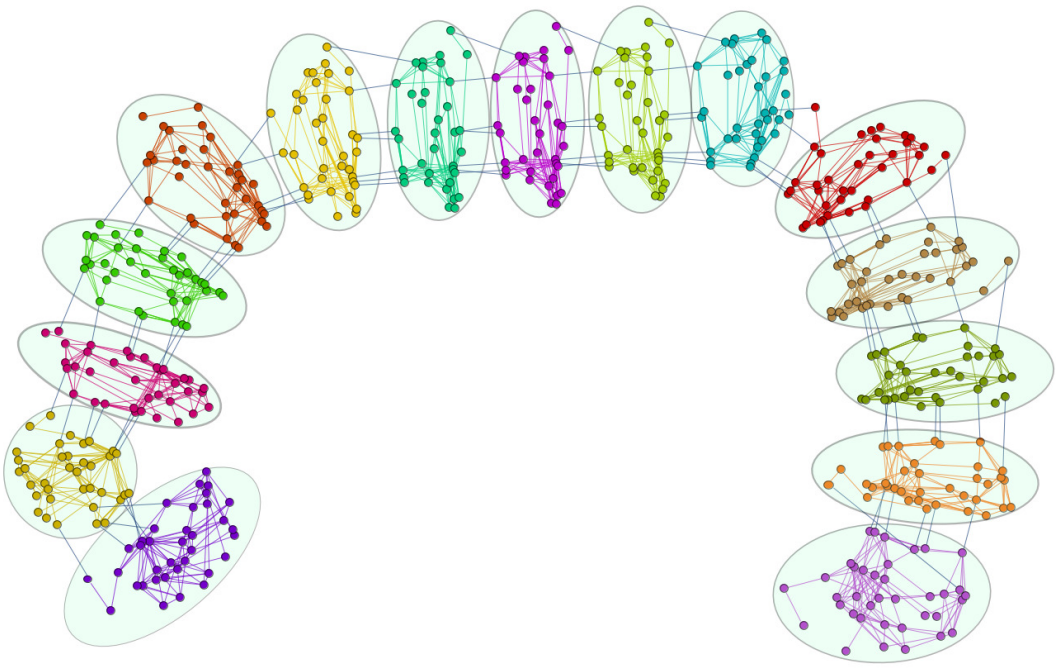


Figure 14: Community detection partition of ammonia reactor network

Figure 14 shows 15 partitions, representing the components, temperature and enthalpy for each of the reactors. The results from using SALA as the coordination algorithm are shown in table 2.

Table 2: SALA algorithm on the ammonia reactor network

Decomposition Statistics	
Method	CPU Decomposition time (s)
Multilevel Graph Partitioning	0.009
Community Detection	0.017
Solution Statistics	
Number of iterations	1078
Number of subproblems	15
final objective value	209.038
CPU time(s)	4,394.98

The optimal partition found by the community detection algorithm is 15 communities. When the multilevel graph partitioning algorithm is performed for 15 subproblems, the performance is very similar to the community detection approach. As mentioned in subsection 4.3.2 the small CPU time difference is likely caused by the implementations.

There is a trade-off between a large reduction in memory consumption and solution space and an increase in the number of iterations of the master problem. A reduction in solution space is ideal if global optimization were to be required as the solution space decreases combinatorially. On the other hand, it would be beneficial to decrease the number of sub problems to increase the convergence rate of the master problem.

If needed it is possible to use the multilevel graph partitioning algorithm to find a smaller number of partitions such as 3 in the example shown in Figure 15.

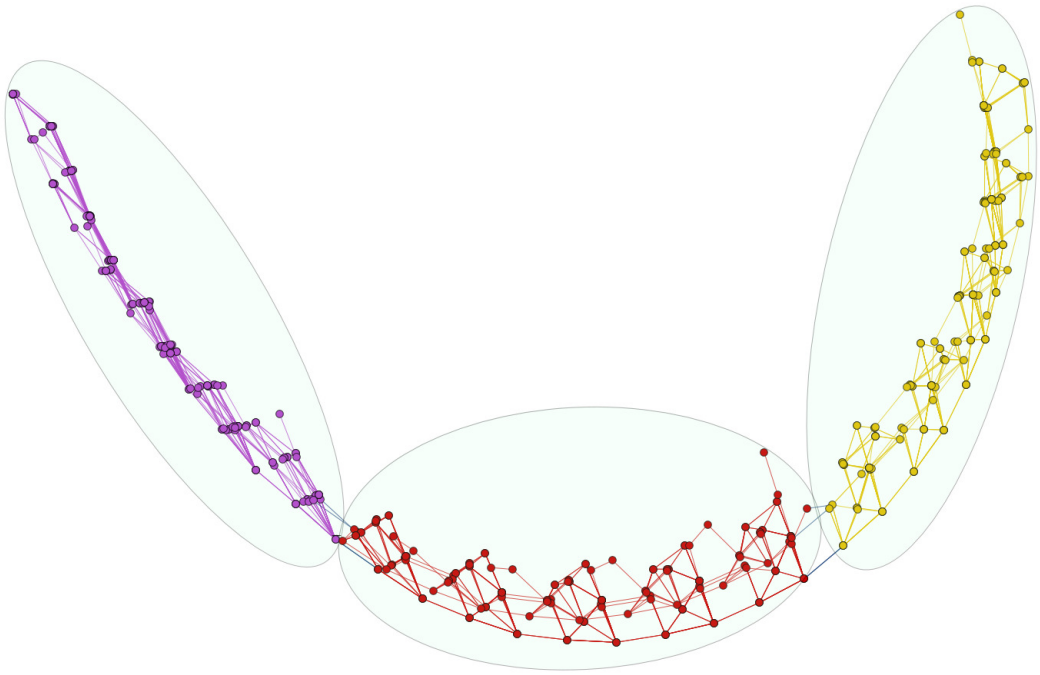


Figure 15: Three way partition of ammonia reactor network

Figure 15 shows the ammonia reactor network partitioned into 3 subgroups, each corresponding to a reactor in the network. The results from using SALA as the coordination algorithm for this problem are shown in Table 3.

Table 3: SALA algorithm on the ammonia reactor network

Decomposition Statistics	
Method	CPU Decomposition time (s)
Multilevel Graph Partitioning	0.007
Community Detection	-
Solution Statistics	
Number of iterations	32
Number of subproblems	3
final objective value	208.927
CPU time(s)	89.39

It is clear that 3 partitions as opposed to 15 would not have the same effect in memory and space reductions, however it results in having a much faster convergence.

In summary, with the first five sample problems the techniques presented in this work are demonstrated to be able to partition problems optimally. The last two examples aim to demonstrate that this can be applied to large systems like those that must be solved in engineering applications. Though it cannot be said that partitions resulting from this framework would always be optimal, from the examples presented it can be concluded that they will generally be of high quality.

From examples 6 and 7 it can be recognized that the identified communities mostly tend to reflect the natural subdivision of the physical units of the system. It can then be stated that community detection already empirically shows the capacity to reproduce known results, of least coupling between sub-systems

with a mathematical guarantee. Furthermore, this framework can be executed in linear time, which is faster than any other known technique. This is especially advantageous when compared to the time complexity of optimization procedures whose complexity is at least polynomial.

5. Conclusions

In this work we show the effectiveness of mapping an optimization problem into an equivalent constraint graph, applying a community detection or multilevel graph partitioning algorithm to find clusters in this graph which is mapped back into a decomposed optimization problem consisting on different subsystems. This results in a fast and robust methodology to find automatically high quality partitions (or even identify if they exist) in an optimization problem, which is a novel contribution in the field of chemical engineering as far as the authors are aware.

A first set of sample problems was used to evaluate the quality of the solution with the partitions found by the methods being optimal. A second set of problems was used to test more complex problems like those common to engineering applications. These last two examples portray the different applications of each approach. For the ammonia conversion process, the community detection method yields 15 sub-problems. These sub-problems reveal the inner structure of the problem: the three components, temperature and enthalpy balances for each reactor. The multilevel graph partitioning approach cannot yield such information about the system. However, when this algorithm is specified to compute a partition for 15 sub-problems the partition is identical to that of the community detection method, and the computational time is of the same order of magnitude.

The quality of the solution, as well as the efficiency of both methods is equivalent when the same number of sub-problems is assigned. However, the usefulness of each approach depends on the situation at hand. If the problem has a restriction on the number of sub-problems, then multilevel graph partitioning is the best option, as the user can directly specify the number of sub-systems. If the problem is to yield information on the current model, the community detection approach is of more use. As it enables the user to identify highly clustered sub-systems without the need to specify a number. In cases where community detection yields a single sub-system, or a very large number, the method shows that the current problem would not benefit from a distributed decomposition approach. This would result from a model having a structure that is not easily divided into subsystems and might present a large overhead for a coordination algorithm.

Focus in this work has been given to augmented Lagrangian decomposition methods as the coordination algorithm, however other strategies from MDO or similar fields could also be employed in future research.

Future work will explore other coordination algorithms as well as a theoretical analysis of the effect of the partition strategies presented on the decomposition optimization problems. It would be an interesting opportunity to explore the ability of this method to split a physical system into a series of modules, maximizing properties that are desirable in the context of simulation, by weighting connections, nonlinearity, and partitioning objective functions. Further investigation on the numerical quality of the solution in comparison to a non-decomposed problem is also warranted. Finally it is important to stress that the practicality of the implementation will be tested against larger realistic problems, and also to model more finely the community interaction of sub-components.

Acknowledgements

Author E. A. del Rio-Chanona would like to acknowledge CONACyT scholarship No. 522530 for funding this project. Author F. Fiorelli gratefully acknowledges the support from his family. The authors would also

like to thank Dr Bart Hallmark, University of Cambridge, for suggesting to employ as a demonstration the chemical system in Example 7.

References

Alexandrov, N. M., Lewis, R. M., 1999. Comparative Properties of Collaborative Optimization and other approaches to MDO. Tech. rep., NASA.

Allison, J. T., Kokkolaras, M., Papalambros, P. Y., 2009. Optimal Partitioning and Coordination Decisions in Decomposition-Based Design Optimization. Journal of Mechanical Design 131–138, 081008.

Aspentech, 2015. Aspen plus.
URL http://www.aspentech.com/products/aspen-plus.aspx

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E., 2008. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment 2008 (10), P10008.

Bollobás, B., 1998. Modern Graph Theory. Vol. 184 of Graduate Texts in Mathematics. Springer New York, New York, NY.
URL http://link.springer.com/10.1007/978-1-4612-0619-4

Boyd, S., Xiao, L., Mutapcic, A., Mattingley, J., 2012. Notes on Decomposition Methods. Tech. rep., Stanford University.

Brandes, U., Delling, D., Gaertler, M., Goerke, R., Hoefer, M., Nikoloski, Z., Wagner, D., 2006. Maximizing Modularity is hard.
URL http://arxiv.org/abs/physics/0608255

Chen, J., Adomaitis, R. A., 2006. An object-oriented framework for modular chemical process simulation with semiconductor processing applications. Computers & Chemical Engineering 30, 1354–1380.

Cheng, C. K., Wei, Y. C. A., 1991. An improved two-way partitioning algorithm with stable performance. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 10, 1502–1511.

Cota, B. A., Sargent, R. G., 1992. A modification of the process interaction world view. ACM Transactions on Modeling and Computer Simulation 2, 109–129.

Dormohammadi, S., Rais-Rohani, M., 2013. Exponential penalty function formulation for multilevel optimization using the analytical target cascading framework. Structural and Multidisciplinary Optimization 47, 599–612.

Drineas, P., Frieze, A., Kannan, R., Vempala, S., Vinay, V., 2004. Clustering large graphs via the Singular Value Decomposition. Machine Learning 56, 9–33.

Elsner, U., 1997. Graph Partitioning - A Survey.
URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.54.2060

Fagley, J., Carnahan, B., 1990. The sequential-clustered method for dynamic chemical plant simulation. Computers & Chemical Engineering 14, 161–177.

Floudas, C., Aggarwal, A., Ciric, A., Oct. 1989. Global optimum search for nonconvex NLP and MINLP problems. Computers & Chemical Engineering 13 (10), 1117–1132.

Garbers, J., Promel, H., Steger, A., Nov 1990. Finding clusters in vlsi circuits. In: Computer-Aided Design, 1990. ICCAD-90. Digest of Technical Papers., 1990 IEEE International Conference on. pp. 520–523.

Haftka, R. T., Sobieszczanski-Sobieski, J., Padula, S. L., 1992. On options for interdisciplinary analysis and design optimization. Structural optimization 4, 65–74.

Haftka, R. T., Watson, L. T., 2005. Multidisciplinary Design Optimization with Quasiseparable Subsystems. Optimization and Engineering 6, 9–20.

Hagen, L., Kahng, A., Nov 1992a. A new approach to effective circuit clustering. In: Computer-Aided Design, 1992. ICCAD-92. Digest of Technical Papers., 1992 IEEE/ACM International Conference on. pp. 422–427.

Hagen, L., Kahng, A. B., 1992b. New spectral methods for ratio cut partitioning and clustering. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 11, 1074–1085.

Hamdi, A., Jan. 2005. Two-level primal-dual proximal decomposition technique to solve large scale optimization problems. Applied Mathematics and Computation 160 (3), 921–938.

Hamdi, A., Al-saud, H., 2008. A re-scaled twin augmented Lagrangian algorithm for saddle point seeking. Nonlinear Analysis: Theory, Methods & Applications 69 (12), 4796–4802.
URL http://dx.doi.org/10.1016/j.na.2007.11.042

Hamdi, A., Mahey, P., 2000. Separable diagonalized multiplier method for decomposing nonlinear programs. Computational and Applied Mathematics 19 (1), 1–29.

Hamdi, A., Mahey, P., Dussault, J. P., 1997. A New Decomposition Method in Nonconvex Programming Via a Separable Augmented Lagrangian. In: Gritzmann, P., Horst, R., Sachs, E., Tichatschke, R. (Eds.), Recent Advances in Optimization. Vol. 452 of Lecture Notes in Economics and Mathematical Systems. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 90–104.

Hart, W. E., Laird, C., Watson, J.-P., Woodruff, D. L., 2012. Pyomo - Optimization Modeling in Python. Vol. 67 of Springer Optimization and Its Applications. Springer US, Boston, MA.
URL http://link.springer.com/10.1007/978-1-4614-3226-5

Heath, M. T., Raghavan, P., jan 1995. A Cartesian Parallel Nested Dissection Algorithm. SIAM Journal on Matrix Analysis and Applications 16 (1), 235–253.
URL http://epubs.siam.org/doi/abs/10.1137/S0895479892238270

Hendrickson, B., Leland, R., 1995. An Improved Spectral Graph Partitioning Algorithm for Mapping Parallel Computations. SIAM Journal on Scientific Computing 16, 452–469.

Hillestad, M., Hertzberg, T., 1986. Dynamic simulation of chemical engineering systems by the sequential modular approach. Computers & Chemical Engineering 10, 377–388.

Hillestad, M., Hertzberg, T., 1988. Convergence and stability of the sequential modular approach to dynamic process simulation. Computers & Chemical Engineering 12, 407–414.

Karypis, G., Kumar, V., 1998. METIS A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices.

LaSalle, D., Karypis, G., 2013. Multi-threaded graph partitioning. In: Proceedings - IEEE 27th International Parallel and Distributed Processing Symposium, IPDPS 2013. pp. 225–236.

Lenoir, A., Philippe, M., 2007. Accelerating convergence of a Separable Augmented Lagrangian Algorithm. Tech. rep., Laboratoire d'Informatique, de Modélisation et d'optimisation des Systèmes.

Mezura-Montes, E., Coello Coello, C. A., 2005. A simple multimembered evolution strategy to solve constrained optimization problems. IEEE Transactions on Evolutionary Computation 9, 1–17.

Miller, G., Teng, S.-H., Vavasis, S., Oct 1991. A unified geometric approach to graph separators. In: Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on. pp. 538–547.

Miller, G. L., Teng, S.-H., Thurston, W., Vavasis, S. A., 1993. Automatic mesh partitioning. In: George, A., Gilbert, J., Liu, J. (Eds.), Graph Theory and Sparse Matrix Computation. Vol. 56 of The IMA Volumes in Mathematics and its Applications. Springer New York, pp. 57–84.

Montanari, U., 1974. Networks of constraints: Fundamental properties and applications to picture processing. Information Sciences 7, 95–132.

Newman, M. E. J., 2004. Detecting community structure in networks. European Physical Journal B 38, 321–330.

Newman, M. E. J., 2006. Modularity and community structure in networks. Proceedings of the National Academy of Sciences of the United States of America 103, 8577.

Newman, M. E. J., Girvan, M., 2004. Finding and evaluating community structure in networks. Physical Review E - Statistical, Nonlinear, and Soft Matter Physics 69, 026113.

Noe, S., Sep. 29 1987. Ammonia synthesis converter. US Patent 4,696,799.
URL `http://www.google.com/patents/US4696799`

Passas-Garrido, J., 2012. Novel, Low Energy, Pre-Combustion Carbon Capture Feasibility Study.

Pidd, M., Castro, R., 1998. Hierarchical modular modelling in discrete simulation. In: 1998 Winter Simulation Conference. Proceedings (Cat. No.98CH36274). Vol. 1. pp. 383–389.

Pothen, A., Simon, H., Wang, L., Barnard, S., 1992. Towards a fast implementation of spectral nested dissection. In: Proceedings Supercomputing '92. pp. 42–51.

Pothen, A., Simon, H. D., Liu, K.-P. P., 1990. Partitioning Sparse Matrices with Eigenvectors of Graphs. SIAM Journal on Matrix Analysis and Applications 11, 430–452.
URL `http://hdl.handle.net/2060/19970011963`

Powell, M., 1969. A Method for Nonlinear Constraints in Minimization Problems. In: Optimization. Academic Press, pp. 283–298.

Process Systems Enterprise, 2015. gPROMS.
URL `www.psenterprise.com/gproms`

Rase, H. F., 1977. Chemical Reactor Design for Process Plants Volume two: Case studies and Design Data. John Wiley & Sons, Inc.

Rivera-Toledo, M., Del Río-Chanona, E. A., Flores-Tlacuahuac, A., Sep. 2014. Multiobjective Dynamic Optimization of the Cell-Cast Process for Poly(methyl methacrylate). Industrial & Engineering Chemistry Research 53 (37), 14351–14365.
URL http://pubs.acs.org/doi/abs/10.1021/ie5014162

Safdarnejad, S. M., Hedengren, J. D., Lewis, N. R., Haseltine, E., 2015. Initialization Strategies for Optimization of Dynamic Systems. Computers & Chemical Engineering 78, 39–50.
URL http://linkinghub.elsevier.com/retrieve/pii/S0098135415001179

Sagastizábal, C., 2012. Divide to conquer: Decomposition methods for energy optimization. Mathematical Programming 134 (1), 187–222.

Sarkar, S., Dong, A., Gero, J. S., 2009. Design Optimization Problem Reformulation Using Singular Value Decomposition. Journal of Mechanical Design 131 (August 2009), 081006.

Schweiger, C., Floudas, C., 1997. MINOPT: A Software Package for Mixed-Integer Nonlinear Optimization.

Stephanopoulos, G., Reklaitis, G. V., 2011. Process systems engineering: From Solvay to modern bio- and nanotechnology.. A history of development, successes and prospects for the future. Chemical Engineering Science 66 (19), 4272–4306.
URL http://dx.doi.org/10.1016/j.ces.2011.05.049

Tappenden, R., Richtarik, P., Buke, B., 2015. Separable Approximations and Decomposition Methods for the Augmented Lagrangian. Optimization Methods and Software 30.

Tosserams, S., Etman, L., Rooda, J., 2009. A classification of methods for distributed system optimization based on formulation structure. Structural and Multidisciplinary Optimization 39, 503–517.
URL http://dx.doi.org/10.1007/s00158-008-0347-z

Tosserams, S., Etman, L., Rooda, J., 2010. Multi-modality in augmented lagrangian coordination for distributed optimal design. Structural and Multidisciplinary Optimization 40, 329–352.
URL http://dx.doi.org/10.1007/s00158-009-0371-7

Wächter, A., Biegler, L. T., Apr. 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming 106 (1), 25–57.

Wasserman, S., Faust, K., 1994. Social Network Analysis in the Social and Behavioral Sciences. In: Social Network Analysis: Methods and Applications. Cambridge University Press, pp. 3–27.

Watanabe, N., Ntshimura, Y., 1978. Decomposition in Large System Optimization Using the Method of Multipliers I. Journal of Optimization Theory and Applications 25 (2), 181–193.

White, H. C., Boorman, S. A., Breiger, R. L., 1976. Social Structure from Multiple Networks. I. Blockmodels of Roles and Positions. American Journal of Sociology 81, 730–780.

Zeigler, B. P., 1987. Hierarchical, modular discrete-event modelling in an object-oriented environment. SIMULATION 49, 219–230.

AppendixA. Full Example Problem 6

The following is a complete description of the problem introduced in section 4.3.1.

Set	Description
I	Components
J	Reactions
L	CSTR units
R	Feeds
P	Products

Table A.4: Notation and subscripts in Example Problem 6 description

Constraints:

Feed splitter

$$F_r^a = \sum_{l \in L} F_{r,l}^{ad}, \qquad r \in R \tag{A.1a}$$

CSTR inlet mixer total balance

$$F_l^d = \sum_{r \in R} F_{r,l}^{ad} + \sum_{l' \in L} F_{l',l}^{gd}, \qquad l \in L \tag{A.1b}$$

CSTR inlet mixer component balance

$$F_l^d = \sum_{r \in R} c_{r,i}^a F_{r,l}^{ad} + \sum_{l' \in L} c_{l',i}^g F_{l',l}^{gb}, \qquad i \in I, \qquad l \in L \tag{A.1c}$$

CSTR total balance

$$F_l^g = F_l^d, \qquad l \in L \tag{A.1d}$$

CSTR component balance

$$c_{l,i}^g F_l^g = c_{l,i}^d F_l^d + V_l^m \sum_{j \in J} v_{i,j} r_{l,j}^m, \qquad i \in I, \qquad l \in L \tag{A.1e}$$

CSTR reaction rates

$$r_{l,j}^m = f_j^r \left(c_{l,i}^m, T_l^m \right), \qquad j \in J, \qquad l \in L \tag{A.1f}$$

CSTR outlet splitter

$$F_l^g = \sum_{l' \in L} F_{l,l'}^{gd} + \sum_{p \in P} F_{l,p}^{gh}, \qquad l \in L \tag{A.1g}$$

Product mixer total balance

$$F_p^h = \sum F_{l,p}^{gh}, \qquad p \in P \tag{A.1h}$$

Product mixer component balance

$$c_{p,i}^h F_p^h = \sum_{l \in L} c_{l,i}^g F_{l,p}^{gh}, \qquad i \in I, \qquad p \in P \tag{A.1i}$$

Variables:

Flowrates					
$F_{r,l}^{ad}$	$\forall r \in R \quad \forall l \in L$	Volumetric flowrates from feed splitters to CSTR mixers	F_l^d	$\forall l \in L$	Volumetric flowrates into CSTRs
$F_{l,p}^{gh}$	$\forall l \in L \quad \forall p \in P$	Volumetric flowrates from CSTR outlets to product mixer	F_l^g	$\forall l \in L$	Volumetric flowrates into CSTRs
$F_{l,l'}^{gd}$	$\forall l \in L \quad \forall l' \in L$	Volumetric flowrates from CSTR outlets to CSTR inlets	F_p^h	$\forall p \in P$	Volumetric flowrates of the product streams
Concentrations					
$c_{l,i}^d$	$\forall l \in L \quad \forall i \in I$	Concentrations of species in the CSTR inlet streams	$c_{l,i}^g$	$\forall l \in L \quad \forall i \in I$	Concentrations of species in the CSTR inlet streams
$c_{p,i}^h$	$\forall p \in P \quad \forall i \in I$	Concentrations of species in the product streams			
Temperatures			Reactor Volumes		
T_l^m	$\forall l \in L$	Temperatures in the CSTRs	V_l^m	$\forall l \in L$	Volume of the CSTRs
Reaction Rates					
$r_{l,j}^m$	$\forall l \in L \quad \forall j \in J$	Rate of reaction j in reactor l			

Table A.5: Super structure notation

Parameters:

The parameters for the problems are the stoichiometric coefficient matrix, the kinetic constants, and the feed conditions. The stoichiometric coefficient matrix, $\nu_{i,j}$, indicates the relative number of moles of products and reactants i that participate in a given reaction j.

Reaction constants					
$\nu_{i,j}$	$\forall i \in I \quad \forall j \in J$	Stoichiometric coefficient matrix	k_j	$\forall j \in J$	Kinetic rate constant
$\hat{k}_j$	$\forall j \in J$	Pre-exponential factor	E_j	$\forall j \in J$	Activation Energy
ΔH_j	$\forall j \in J$	Heat of reaction			
Feed conditions					
F_r^a	$\forall r \in R$	Flowrate of feed streams	$c_{r,i}^a$	$\forall r \in R \quad \forall i \in I$	Concentrations of species in feed streams

Table A.6: Parameters of problem

Rate Expressions:

The rate expressions $f_j^r(c_i, T)$ are specified for each reaction. These expressions often involve the products of the concentrations of the species. The rate expressions have the form

Rate Constants			
reaction	$\hat{k}$	E	$\frac{\Delta H}{\rho Cp}$
1	5.4x10^9 h^{-1}	15.84 kcal/mol	84 K L/mol
2	1.6x10^{12} L/(mol h)	23.76 kcal/mol	108 K L/mol
3	3.6x10^5 h^{-1}	7.92 kcal/mol	60 K L/mol
Feed Conditions			
F_r^a	100 L/s		
$c_{r,i}^a$	1.0 mol/L A, 0.0 mol/L B, 0.0 mol/L C , 0.0 mol/L D		

Table A.7: Parameters of sub-reactor network 1

$$f_j^r(c_i, T) = k_j(T) \prod_{i \in I} c_i^{\nu_{i,j}^f} \tag{A.1j}$$

where $\nu_{i,j}^f$ correspond to the positive coefficients in the stoichiometric matrix representing the forward reactions. Note that for equilibrium reactions, both the forward and reverse reactions must appear separately and the rate constants for both must be provided.

For isothermal reactions, the values of k_j are fixed. For most non-isothermal problems, Arrhenius temperature dependence will be assumed:

$$k_j(T) = \hat{k}_j \exp\left(\frac{-E_j}{RT}\right) \tag{A.1k}$$

The resulting optimization problem is the following:

$$\max \quad 100 c_{1,M}^h - \sum_{l \in L} v_l^m \tag{A.1l}$$

Subject to the reactor network in figure 11 which can be represented by equations from A.1a to A.1l.

Sub-Reactor network 1

Isothermal Van de Vusse Reaction Case I

$$\mathrm{A} \xrightarrow{k_1} \mathrm{B} \xrightarrow{k_2} \mathrm{C}$$

$$2\mathrm{A} \xrightarrow{k_3} \mathrm{D}$$

Parameters

$$\nu_{i,j} = \begin{bmatrix} -1 & 0 & 2 \\ 1 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.2a}$$

Rate Expression

$$f_1^r = \hat{k}_1 \exp\left(\frac{-E_1}{RT}\right) C_A \tag{A.2b}$$

Rate Constants	
k_4	0.025 gmol/(L min) (first order)
k_5	0.2 min^{-1} (first order)
k_6	0.4 L/(mol min) (second order)
Feed Conditions	
F_r^a	100 L/min pure E
$c_{r,i}^a$	1.0 mol/L E, 0.0 mol/L F, 0.0 mol/L G, 0.0 mol/L H

Table A.8: Parameters of sub-reactor network 2

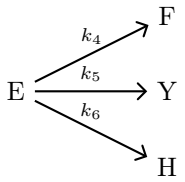
$$f_2^r = \hat{k}_2 \exp\left(\frac{-E_2}{RT}\right) C_B \tag{A.2c}$$

$$f_3^r = \hat{k}_3 \exp\left(\frac{-E_3}{RT}\right) C_A^2 \tag{A.2d}$$

The temperatures in the reactors are bounded between 300 K and 810 K.

Sub-Reactor network 2

Isothermal Trambouze Reaction



Parameters

$$\nu_{i,j} = \begin{bmatrix} -1 & -1 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.3a}$$

Rate Expressions

$$f_1^r = k_4 \tag{A.3b}$$

$$f_2^r = k_5 C_E \tag{A.3c}$$

$$f_3^r = k_6 C_E^2 \tag{A.3d}$$

Sub-Reactor network 3

Isothermal Levenspiel Reaction

$$\text{G+B} \xrightarrow{k_7} \text{2B}$$

Rate Constants	
k_7	1.0 L/(mol s) (second order)
Feed Conditions	
G outlet from Isothermal Trambouze Reaction	
B outlet from Isothermal Van de Vusse Reaction Case I	

Table A.9: Parameters of sub-reactor network 3

Rate Constants			
reaction	$\hat{k}$	E	$\frac{\Delta H}{\rho Cp}$
8	2.0x10^{13} h^{-1}	38.0 kcal/mol	364 K L/mol
9	2.0x10^{13} h^{-1}	38.0 kcal/mol	129 K L/mol
10	8.15x10^{17} h^{-1}	50.0 kcal/mol	108 K L/mol
11	2.1x10^{5} h^{-1}	20.0 kcal/mol	222 K L/mol
Feed Conditions			
F_r^a	100 L/s		
$c_{r,i}^a$	1.0 mol/L I, 0.0 mol/L J, 0.0 mol/L K, 0.0 mol/L L		

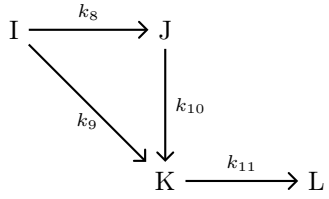
Table A.10: Parameters of sub-reactor network 4

Parameters

$$\nu_{i,j} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Sub-Reactor network 4

Nonisothermal Naphthalene Reaction



Parameters

$$\nu_{i,j} = \begin{bmatrix} -1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.4a}$$

Rate Expressions

$$f_1^r = \hat{k}_8 \exp\left(\frac{-E_8}{RT}\right) C_I \tag{A.4b}$$

Rate Constants			
reaction	$\hat{k}$	E	$\frac{\Delta H}{\rho Cp}$
12	5.4x10^{17} h^{-1}	19.138 kcal/mol	10 K L/mol
13	3.6x10^{5} h^{-1}	9.569 kcal/mol	20 K L/mol
Feed Conditions			
F_1^a	J outlet from Nonisothermal Naphthalene Reaction		
F_2^a	B outlet from Isothermal Levenspiel Reaction		
$c_{1,i}^a$	J outlet from Nonisothermal Naphthalene Reaction		
$c_{2,i}^a$	B outlet from Isothermal Levenspiel Reaction		

Table A.11: Parameters of sub-reactor network 5

$$f_2^r = \hat{k}_9 \exp\left(\frac{-E_9}{RT}\right) C_I \tag{A.4c}$$

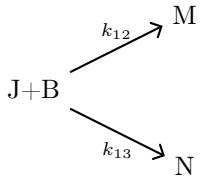
$$f_3^r = \hat{k}_{10} \exp\left(\frac{-E_{10}}{RT}\right) C_J \tag{A.4d}$$

$$f_4^r = \hat{k}_{11} \exp\left(\frac{-E_{11}}{RT}\right) C_K \tag{A.4e}$$

The temperatures in the reactors are bounded between 900 K and 1500 K

Sub-Reactor network 5

Nonisothermal Parallel Reactions



Parameters

$$\nu_{i,j} = \begin{bmatrix} -1 & -1 \\ -1 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{A.5}$$

AppendixB. Full Example Problem 7

The following is a full description of the problem presented in section 4.3.2.

The details of the stream entering the system are given in Table B.12.

Table B.12: Description of feed stream

Stream	Component mole fraction
Hydrogen	0.6989
Nitrogen	0.2501
Ammonia	0.0131
Conditions	
Temperature: (C)	61.6
Pressure: (kPa)	13600
Molar Flow (kgmol/h)	14760

Parameter	Value	Units
Number of packed beds	3	
Packed bed diameter	3	
Volume of packed bed 1	17.7	m^3
Volume of packed bed 2	17.7	m^3
Volume of packed bed 3	17.7	m^3
Packed bed void fraction	0.42	
Catalyst solid density	2500	kg/m^3
Enthalpy of reaction	-91420	kJ/kgmole nitrogen consumed
Average gas phase heat capacity(calculated by UniSim)	32.86	kJ/(kgmole K)
Corrected gas phase heat capacity	50.0	kJ/(kgmole K)

Table B.13: Data for ammonia reactors

The equilibrium describing ammonia formation from gaseous hydrogen and nitrogen is shown in Equation .

$$\mathrm{N_2} + 3\,\mathrm{H_2} \rightleftharpoons 2\,\mathrm{NH_3} \tag{B.1}$$

The rate of formation of ammonia, expressed in kilomoles of ammonia per cubic meter of catalyst bed per hour is shown below in Equation (B.2).

$$\mathrm{r_{NH_3}} = 2k\left[K_a^2 a_{N_2}\left(\frac{a_{H_2}^3}{a_{NH_3}^2}\right)^{\alpha} - \left(\frac{a_{NH_3}^2}{a_{H_2}^3}\right)^{1-\alpha}\right] \tag{B.2}$$

In Equation (B.2), k represents the rate constant for the decomposition of ammonia into hydrogen and nitrogen and is shown in Equation (B.3); the units of the activation energy and universal gas constant are Joule based, with the activation energy being 1.705×10^5 J/mol. K_a is the equilibrium constant, an expression for which is given in Equation (B.3). The component-specific parameters denoted by a_i in Equation (B.2) are the pure species activities. It is assumed that the reference fugacity in the system is one atmosphere, and hence it is possible to relate the pure species activity, a_i, to the gas phase mole fraction of a particular component, y_i, to the pure species fugacity coefficient, i, and to the total pressure in the system, P, measured in atmospheres. This relationship is shown in Equation (B.5) and literature expressions for the fugacity coefficients of hydrogen, nitrogen and ammonia are shown in Equations (B.8), (B.7) and (B.7) respectively. The parameter α is a constant typically ranging between 0.5 and 0.75. All temperatures are measured in K.

$$k = 8.89 \times 10^{14} \exp\left(\frac{-1.705 \times 10^5}{RT}\right) \tag{B.3}$$

$$\begin{aligned}\log_{10}(K_a) = &- 2.691122\ \log_{10}(T) - 5.519265 \times 10^{-5} T \\ &+ 1.848863 \times 10^{-7} T^2 + \frac{2001.6}{T} + 2.6899\end{aligned} \tag{B.4}$$

$$a_i = y_i \varphi_i P \tag{B.5}$$

with

$$\begin{aligned}\varphi_{N_2} =& 0.9343 + 3.101 \times 10^{-4} T + 2.959 \times 10^{-4} P \\ &- 2.707 \times 10^{-7} T^2 + 4.775 \times 10^{-7} P^2\end{aligned} \tag{B.6}$$

$$\begin{aligned}\varphi_{NH_3} =& 0.1439 + 2.029 \times 10^{-3} T - 4.488 \times 10^{-4} P \\ &- 1.143 \times 10^{-6} T^2 + 2.761 \times 10^{-7} P^2\end{aligned} \tag{B.7}$$

$$\begin{aligned}\varphi_{H_2} =& \exp\left[\left(-3.8402\, T^{0.125} + 0.541\right) P - \exp\left(-0.1263\, T^{0.5} - 15.98\right) P^2\right. \\ &\left. +300\ \exp\left(-0.011901\, T - 5.941\right) \exp\left(\frac{-P}{300} - 1\right)\right]\end{aligned} \tag{B.8}$$

Given the high nonlinearity of the equations (B.2 - B.8) the rate expression (B.2) was approximated by a multivariate Lagrange polynomial of second order for the optimization problem.

The objective of the above process is to maximize production of NH_3 while minimizing energy costs by the heat exchanger duty, this utility function is the following:

$$\max \quad 1000\ \mathrm{NH_3} - (T1_0 - 344.48)^2 - (T2_0 - T1_f)^2 - (T3_0 - T2_f)^2 \tag{B.9}$$

where Ti_j denotes the temperature on the i^{th} reactor, and $j = 0$ means the inlet temperature of the feed and $j = f$ denotes the outlet temperature of the reactor.