Design Optimization of Cyber-Physical Systems by Partitioning and Coordination: A Study on Mechatronic Systems

Pouya Mahdavipour Vahdati, *Graduate Student Member, IEEE*, Lei Feng, *Member, IEEE*, Martin Törngren, *Senior Member, IEEE*

***Abstract*—Cyber-Physical Systems are inherently complex. Reducing the design complexity of such systems, is beneficial for scalability of the design. In this paper, a method for design optimization of these systems is proposed to facilitate the decomposition of the optimization problem for the whole system into smaller sub-problems and coordination of modular solutions to reach the desired optimum. The coordinated solutions are either identical to the optimal solution of the complex optimization problem for the system as a whole or within an acceptable error margin of it. To demonstrate the efficacy of the method, it is applied to a mechatronic case study. The results provide evidence for the potential feasibility of the methodology in terms of meeting the requirements on the solutions, while reducing the computational demand of the design process.**

***Index Terms*—Cyber-Physical Systems, Mechatronic Systems, Design Optimization, Design Structure Matrix, Complexity.**

I. Introduction

The design and development of systems is becoming an increasingly complex task due to the numerous disciplines involved, rapidly increasing scales, and also the integration issues of different domains and technologies. The complexities increase especially in case of Cyber-Physical Systems (CPS). These systems are integrations of computation, communication and physical elements, and of the multiple concerns of interests including dimension, power, weight, performance, safety, modularity, sustainability, etc. Since the design of CPS is inherently a multi-view problem [1], [2], it raises the requirement for more efficient algorithms to execute the design task. Increasing the design process efficiency serves the purpose of reduction in usage of allocated resources.

There has been numerous studies in design optimization of complex systems. One of the major fields of study has been the field of Multidisciplinary Design Optimization (MDO), which essentially is the coupling of several disciplines and their respective analysis in the context of an optimization problem [3]. The basics of MDO methodologies rely on the fact that the system should be decomposed into several parts based on the disciplines involved; in other words, the system is decomposed into a set of several smaller partitions. There has been extensive studies to improve the performance of the MDO methodologies [4]–[8] and thorough benchmarking and reviews on the methods [9]–[11]. However, since the partitioning in MDO methodologies is only based on the subjective decisions of disciplines, it is not guaranteed that the resulted partitions are the best for reducing computational overheads and also the optimality of the solution [12]. Hence, there is a need to use algorithmic approaches to find the most effective system partitions for reducing complexities. There are numerous algorithms based on graph theory and Design Structure Matrix (DSM) representation to address the partitioning problem; however, to the best of authors' knowledge, they have not been directly applied to design optimization problems [13]–[15].

In addition, in decomposition-based design optimization methods, management of the connections between the partitions is of crucial importance for consistency of the design. This management of connections is done by coordinating the solutions of the partitions towards the optimal results for the design problem as a whole. In other words, the coordination of solutions keeps the decomposed system consistent [16].

Methodologies other than MDO tend to solve the optimization problem as a whole without decomposing the problem into smaller problems. This causes obstacles in terms of scalability of the methodologies; if the systems are complex, the number of domains involved would increase and consequently the number of variables. Therefore, the robustness of the optimization in terms of convergence would be highly dependent on the utilized algorithms and the computational demands would increase.

In this paper, the partitioning is based on the dependencies between the design variables (DVs) and the functions of the system [17]. Utilizing the dependencies between variables and functions across the whole system for partitioning enables the partitions to be formed based on the nature of the system rather than subjective organizations of disciplines. Partitioning the system in this manner has the potential to make the design process more efficient since the number of DVs and functions in each partition is much less than the total number. This way of partitioning differs from partitioning methods used in MDO, and is not subjective. Next, a coordination strategy as a high-level entity is introduced, which guides the solution of partitions towards the optimum of the complex system as a whole, or within an acceptable margin of the objective function's optimal value. The coordination strategy only uses the interacting design variables between the partitions. This

P. Mahdavipour Vahdati is supported by the European Unions Horizon 2020 Framework Programme for Research and Innovation under grant agreement no 674875 (oCPS Marie Curie Network).

P. Mahdavipour Vahdati, L. Feng, M. Törngren are with the Department of Machine Design, KTH Royal Institute of Technology, Stockholm, Sweden.
E-mail: pouyamv@kth.se.

means that the optimization of partitions is over their local design variables, and the interacting variables keep the partitions consistent through the coordination strategy.

Section II describes the optimization approach and the generic framework and its criteria for partitioning and coordination of the partitions' solutions. In Section III, the methodology is applied to a mechatronic system and the results and discussions are provided. Section IV concludes the paper and provides insight into future work.

II. Optimization Approach

This section first introduces the fundamental concepts of the framework. It is assumed that a mathematical and explicit representation of the system is present, based on which the DVs, the functions, and the general objective of the design optimization can be identified. Under this assumption, the framework targets to utilize formalized dependencies to decompose the complex optimization problem into smaller sub-problems (i.e. partitions), proposes a high-level entity as a coordination strategy for integrating optimal solutions of smaller sub-problems, and investigates the effect of such decomposition and coordination on computational overhead of the design process and also the optimality of the resulted design. As mentioned earlier, results should be either identical to the optimum derived from the solution of the complex problem without partitioning, or within an acceptable margin of the objective function's optimal value. In this work, the acceptable margin is difference of less than 5% between the optimal value of the objective function when the problem is solved with partitioning, and the optimal values of the objective function while the system is optimized without partitioning. The aforementioned high-level entity can also be an optimizer.

A. Dependencies, Graph and DSM

Dependency in this work is determined by DVs and functions, and is considered to be binary; in other words, the strength of the dependencies is not taken into consideration at this point [18]. Based on the mathematical representation of the system, the DVs and system functions are represented by a digraph to identify dependencies. In the digraph of the system, each vertex represents a DV or a function, and the directed edges are dependencies from DVs to functions. The direction of the edges represents the function-variable relations in the graph; if the edge points from vertex i to vertex j, it means that vertex j is dependent on vertex i and hence it is a function of vertex i. This digraph is then used to construct the DSM of the system. DSM is a square matrix, whose elements can represent both functions and variables [19]. In other words, if the system has n variables and m functions, then the DSM would be a matrix of dimension $(n+m) \times (n+m)$. Binary dependencies are represented in the following manner; let the DSM be a matrix D of $(n+m) \times (n+m)$. $D(i,j) = 1$ if and only if there exists an edge between vertex i and vertex j in the graph of the system. Otherwise, $D(i,j) = 0$. This DSM, will be then decomposed into partitions.

B. Partitioning and Complexity Measure

There are several interpretations of complexity in CPS, [20]. In this work, complexity is defined according to the dependencies between DVs and functions. Additionally, the computational overhead scales with the number of variables and functions included in each optimization process. To reduce the computational overhead for the design process, the primary idea is that the complex system should not be optimized altogether. Initially, the system should be decomposed into several partitions, each of which contains a subset of DVs and functions. To this end, the dependencies which define the complexity within partition and also in between the partitions will be utilized to partition the system optimally. In other words, to reduce the computational overhead, the closely dependent DVs and functions should be clustered together and the interactions between the partitions should be minimized. This partitioning should be formulated as a multi-objective optimization problem, in which the cost functions are the complexity within and between partitions (i.e. intensity of dependencies); The objective functions are the following:

- The cost of every subgroup determined by the dependencies of the enclosed functions and design variables.
- The cost of interactions between the partitions.

Since at this stage these dependencies are considered to be binary, the complexity measure can be represented as the number of directed edges in the graph of the system. These definitions represent the objectives of decomposition, and the aforementioned multi-objective optimization problem can be formulated such that the number of edges within each partition would be balanced while the number of edges between the partitions would be minimized. In a way, this is distributing the complexity of the system among the partitions. This would lead to reduction in optimization overhead of each partition which would in turn lead to less computational demand for the design process. The mathematical formalization of optimal partitioning problem is within the scope of future work.

This paper does not determine the partitions by the multi-objective optimization approach. A heuristic DSM manipulation method [21], [22] is applied to find a good-enough decomposition.

Partitioning Algorithm: This section applies the DSM organization algorithm in [21], [22] to determine the system decomposition. The objective of this algorithm is to come up with clusters which contain elements with high inter-dependencies, while the dependencies between the clusters is minimized. The general approach of the algorithm is that, initially all elements are considered as a partition and then they bid on other elements to form a bigger partition by having that element in their partition. The bidding function, cost functions, general flow of the algorithm and its parameters are introduced below. Note that Simulated Annealing (SA) [23] is used in the algorithm to find the best partitions according to the cost functions. The random search operation of SA may prevent the algorithm from reaching local minimum.

- DSM_{size}: Number of the rows/columns of the DSM.
- pow_{cc}: The parameter to control the type of penalty assigned to the size of the partition in the coordination

cost. For example, pow_{cc} implies a quadratic type of penalization.

- pow_{bid}: The parameter to control the penalty assigned to the size of the partition in the bidding function.
- pow_{dep}: The parameter to increase the effect of DSM interactions in bidding function.
- max_Cl_size: The maximal number of elements in each partition.
- $times$: The parameter for the algorithm to attempt $times \times DSM_{size}$ to pick an element and form a cluster before checking the stability.
- $stable\_limit$: The parameter for the algorithm to loop $stable\_limit \times times \times DSM_{size}$ without any change in coordination cost before ending.
- $rand\_accept$: The parameter for the algorithm to limit the number of times that it is allowed to accept the changes in partitions even if the coordination cost is not improved. Setting it to N, makes the algorithm to proceed with changes 1 out of N times, despite the fact that the coordination cost is not improved. This parameter is used for SA.
- $rand\_bid$: The parameter for the algorithm to limit the number of times that it is allowed to choose the second highest bidder instead of the highest bidder. Setting it to N, makes the algorithm to choose the second highest bidder instead of the highest one 1 out of N times. This parameter is used for SA.

The algorithm with corresponding functions is provided below.

1) Consider each element of DSM to be a partition initially.
2) Calculate the Coordination Cost of the partitioned matrix.
3) Choose an element randomly.
4) Calculate the bid for each partition for the selected element to become a member of their partition.
5) Choose a number between 1 and $rand\_bid$ randomly.
6) Calculate the Total Coordination Cost of the selected element after it becomes a member of the partition with the highest bid (use the second highest bid if Step 5 is equal to $rand\_bid$).
7) Choose a number between 1 and $rand\_accept$ randomly.
8) If the new Coordination Cost is lower than the previous Coordination Cost or the number chosen in Step 7 is equal to $rand\_accept$, make the change permanent otherwise make no changes.
9) Go back to Step 3 until the iteration limit is reached.

The SA part of the algorithm utilizes parameters $rand\_bid$ and $rand\_accept$ to accept the bid of the second highest bidder instead of the highest bidder. The first one affects the probability of choosing either the first or second best bid. The second one, effects the probability of cluster changes while total coordination cost is not reduced. In both cases simulated annealing introduces random changes in the steps taken by the algorithm, changes that would not have taken place otherwise. By doing so, the algorithm tries to look for all possible options of clustering.

In addition, the bid is calculated using the following formula. Note that the bid is calculated for the randomly chosen element in Step 3. Assuming that the bidding partition is the j^{th} partition,

$$PartitionBid_j = \frac{inout^{pow_{dep}}}{PartitionSize_j^{pow_{bid}}} \tag{1}$$

where j is the partition number, $PartitionBid_j$ is the bid from partition j for the chosen element, and $inout$ is the sum of DSM interactions of the chosen element with each of the elements in partition j.

$$\begin{aligned} IntraPartitionCost = (DSM(j,k)+ \\ DSM(k,j)) \times PartitionSize(y)^{pow_{cc}} \end{aligned} \tag{2}$$

where $PartitionSize(y)$ represents the number of elements in partition y. For interactions between element j and k that occur outside of a partition,

$$\begin{aligned} InterPartitionCost = (DSM(j,k) + DSM(k,j)) \times \\ DSM_{size}^{pow_{cc}} \end{aligned} \tag{3}$$

Using (2) and (3) the Total Coordination Cost can be formulated as follows,

$$\begin{aligned} TotalCoordinationCost = \sum_{DSM_{size}} IntraPartitionCost + \\ \sum_{DSM_{size}} InterPartitionCost \end{aligned} \tag{4}$$

The pseudo-code of the algorithm is provided below and its flowchart is shown in Figure 1.

Partitioning Algorithm

Initialize: Read data and parameters. Form initial clusters. Calculate total coordination cost. Set the system to be unstable: $system = 0$

```
1:  while system < stable_limit do
2:      repeat
3:          Choose Element
4:          Calculate bids (random uniform distribution)
5:          Select the best bid (the second best bid 1 out of
            rand_bid times)
6:          Calculate new total coordination cost;
7:          if Improvement in total coordination cost then
8:              Update clusters
9:          else if Still update (1 out of rand_accept times)
            then
10:             Update clusters
11:         end if
12:     until Repetitions = DSM_size × times
13:     system = system + 1
14: end while
15: Output clusters
```

The process of selecting an element is repeated in the inner $repeat$ loop $DSM_{size} \times times$ times. The outside $while$ loop is repeated for as long as the inner loop achieves decreases in the total coordination cost. Once no decreases are found, it attempts to improve the cost $stable\_limit$ times more. Therefore, there will be at least $DSM_{size} \times times \times stable\_limit$ attempts to improve the coordination cost before the algorithm finishes. The penalization of inter and intra partition costs, is

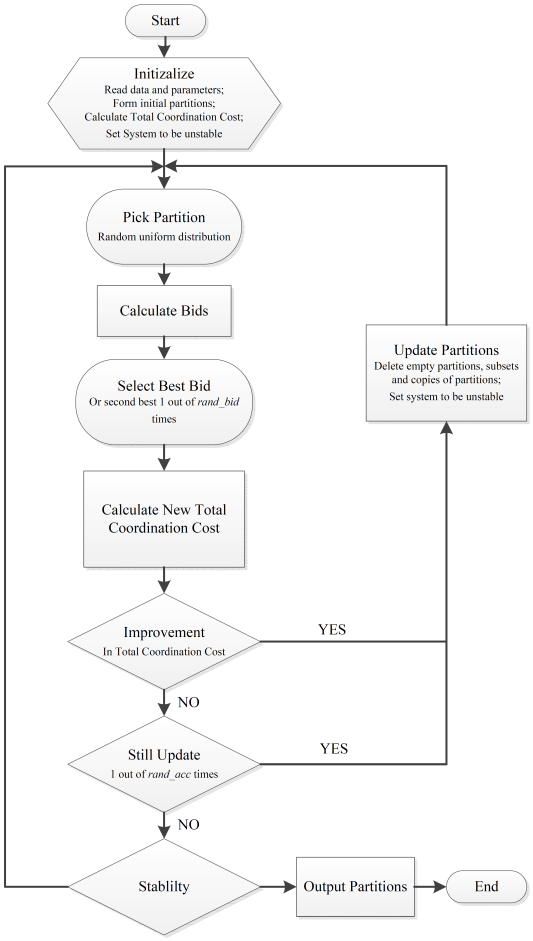

Fig. 1: Partitioning algorithm

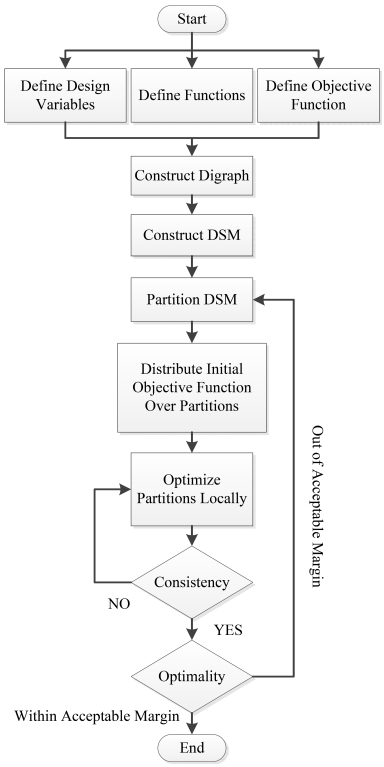

Fig. 2: General framework of the proposed methodology

done through the penalization of cluster size with respect to the size of the DSM in the bidding function. Additionally, the tuning of the parameters was done by trying different values and choosing the best result. It should be noted that this partitioning is not optimal, and optimal partitioning of the system is withing the scope of future work.

C. Coordination of Partitions

In the partitioning process, it is implied that the resulting partitions are derived such that their interactions and coordination would represent the system as a whole. This notion is the *consistency* of the partitions. The consistency of the systems is kept through the interacting variables of partitions. Each partition has a subset of DVs and/or functions, but some DVs/functions might appear in more than one partition, which is the underlying reason of intra-partition interactions. This means that the optimal values for the interacting variables should be in accordance with the optimal values of the local partition variables. Hereby, the interacting variables between partitions are referred to as coordinating variables.

In this work, partitions are individually optimized over their respective local variables (i.e. independent from the coordinating variables), and then a high-level entity (which can be another layer of optimization) is introduced for coordination of the partitions with the consistency constraint of the system; In other words, the values for coordinating variables which guarantee the consistency of the system, are determined by the high-level entity, and then are given as inputs to the local optimizers for each partition. This can be interpreted as introducing a *new* layer in the system which is a function of the coordinating variables between any two individual but interacting partitions.

It should be noted that in this approach, the objective function of the high-level entity would be a function of objective functions of the partition optimizers, and the constraint would be consistency of coordinating variables. In addition, this coordination strategy can be formulated as an iterative method such as Fixed-Point Iteration, or can be formulated as an optimization problem with the coordinating variables' consistency constraint. The generic flow of the framework is presented in Figure 2.

III. Case Study

This section evaluates the proposed framework with a case study, which is a mechatronic servo system consisting of a motor, a shaft, two planetary gears and a rotational load. The optimization objective is to minimize the volume of the system under the condition that the system must satisfy the given load profile. The system is shown in Figure 3. The system formulation consists of three constraints for a motor, a shaft (which serves as the dynamic behavior constraint) and two planetary gears. In [24], the authors have reformulated these constraints from a nonlinear form to geometric programming form to utilize the benefits of convex optimization algorithms and also have used Disciplined Convex Programming tool

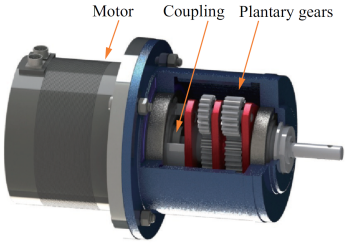

Fig. 3: System under study [7]

TABLE I: Design Variables

r_m	Motor radius
l_m	Motor length
r_g	Planetary gear radius
b_{gi}	Width of the i^{th} stage gear
n	Total gear ratio
n_i	i^{th} stage gear ratio
r_s	Shaft radius
l_s	Shaft length

to solve the optimization problem. Here, the same formulae have been used. This example is used to demonstrate the applicability of the methodology. In general, this framework is not limited to convex problems.

A. Optimization Problem

In this section, the optimization problem and the DVs are introduced. As mentioned earlier, the geometric programming formulation is used here. The DVs of the system are enumerated in Table I. All the formulations, derivation of DVs, specifications of the system and dynamic criteria of the system are obtained from [24]–[26]. The optimization problem is formulated as follows:

$$
\begin{aligned}
min \quad & f = \pi r_m^2 l_m + \pi r_s^2 l_s + \tfrac{3}{2}\pi r_g^2 (b_{g1} + b_{g2}) \\
S.t. \quad & \tfrac{n_j^2 n^2 C_{mj}^2 T_{rms}^2 r_m^3}{J_{load}^2 C_m^2} + \tfrac{2 n_j C_{mj} T_{rms}^2}{J_{load}^2 r_m l_m C_m^2} + \tfrac{T_{rms}^2}{n^2 l_m^2 r_m^5 C_m^2} \leq 1 \\
& 4 \times 10^4 \tfrac{C_{gr}^2}{\sigma_{H,max}^2 r_g^2 b} (n + \tfrac{1}{n} \textstyle\sum_{k=1}^{7} (\tfrac{2}{n})^{k-1}) \leq 1 \\
& \tfrac{40.5316}{2.09689 k_l} + \tfrac{(0.027864 - ISE_0^2) k_l}{2.09689} \leq 1 \\
& 0.03 \leq r_m \leq 0.07 \\
& 0.01 \leq l_m \leq 0.18 \\
& l_{bm} \leq \tfrac{l_m}{r_m} \leq u_{bm} \\
& l_{bg} \leq \tfrac{b_{g1,2}}{r_g} \leq u_{bg} \\
& 3 \leq n_{1,2} \leq 10 \\
& 0.05 \leq l_s \leq 0.1 \\
& l_{bs} \leq r_s \leq 0.1 \\
& 9 \leq n \leq 100 \\
& 3 \leq n_1 \leq 10 \\
& 3 \leq n_2 \leq 10 \\
& n = n_1 . n_2
\end{aligned}
\tag{5}
$$

In (5), the first three constraints are for the motor, the planetary gear and the dynamics of the shaft, respectively. The rest are the boundary constraints of the DVs. This problem has been solved as a monolithic optimization problem in [24]. In the next section, the results obtained by application of the methodology presented in Section II will be presented.

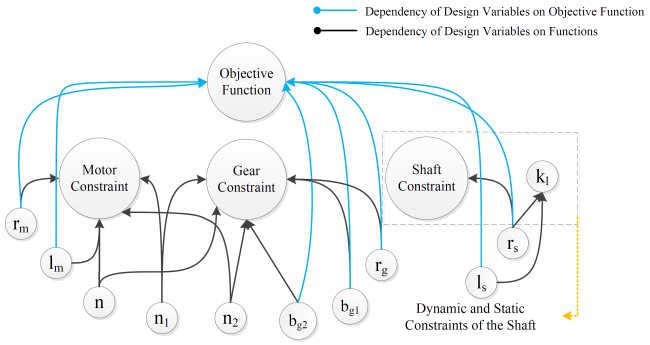

Fig. 4: Graph representation of the system

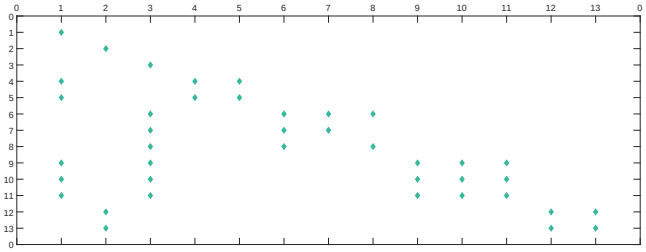

Fig. 5: Design Structure Matrix of the system before partitioning

B. Graph, DSM and Partitioning

This section first constructs the dependency graph of the case study. The graph of the system is shown in Figure 4. In the graph above, the dependencies are illustrated. Based on these dependencies the DSM of the system is constructed in Figure 5. Based on Section II-A, the motor constraint, the gear constraints and the shaft constraint are considered as functions, hence represented by vertices. The DVs are also represented by vertices. Hence the DSM consists of ten DVs and three functions. The elements of the DSM are shown in Table II. This matrix represents dependencies in a binary manner. Elements in the rows are identical to the elements of the columns, hence the matrix is a square matrix. Adopting the partitioning algorithm of Section II-B, the DSM is partitioned into three groups as shown in Figure 6. As observed in Figure 6, there are three partitions in the system. The first and second partitions, which contain the motor and gear constraints respectively, are coupled through n, n_1 and n_2. In other words, these variables connect the first and second partitions.

TABLE II: Elements of DSM

Motor Constraint	1
Shaft Constraint	2
Gear Constraint	3
r_m	4
l_m	5
r_g	6
b_{g1}	7
b_{g2}	8
n	9
n_1	10
n_2	11
r_s	12
l_s	13

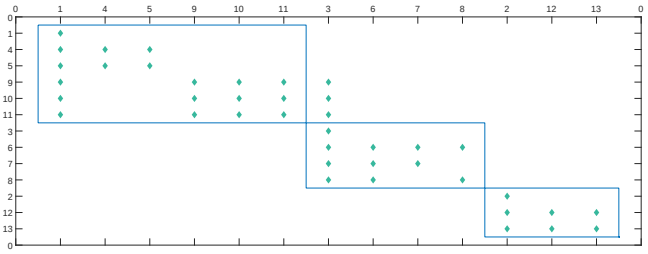

Fig. 6: Design Structure Matrix of the system after partitioning

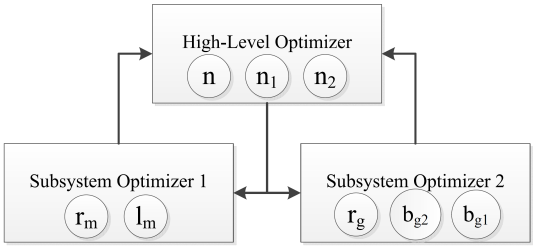

Fig. 7: Optimization based on the first approach.

If there was no coupling between the partitions, the DSM after partitioning would consist of three completely decoupled partitions with all the functions and variables included in their respective partitions. The third partition, which is the dynamic performance constraint of the system, is decoupled and can be optimized autonomously. he coordinating variables between two of the partitions are n, n_1 and n_2, and the third partition is completely decoupled which means that there are no interactions between this partition and the others. Hence, the coordinating variables between two of the partitions are n, n_1 and n_2.

These observations imply that the three above-mentioned variables should be used to keep the system consistent. There can be two different approaches for keeping the system consistent:

- n, n_1 and n_2 are all considered as coordination variables to keep the system consistent.
- n is considered as a coordination variable to keep the system consistent, while n_1 and n_2 are optimized in their corresponding partition through the constraints.

The first approach, has three variables assigned to the high-level entity to keep the system consistent. The second approach assigns only one coordination variable to the high-level entity. It should be noted that the high-level entity in this case, is also an optimizer. This means that in the first approach, the high-level entity would give optimal values for all the coordinating variables in accordance with the optimal values of each partition. Hence, if the algorithm converges, the results would be identical to the results of optimizing the system as a whole because there is no prioritization in terms of compliance with one of the partitions for the high-level entity and all the partitions are optimized over their local variables while the high-level entity handles all the coordinating variables. However, in the second approach, only one of the coordinating variables is handled by the high-level entity and the other two are handled by their respective local optimizer; this means that the chosen coordinating variable for the high-level entity would be optimized in accordance to one of the partitions. Then the other partition would comply with the results of high-level entity based on the prioritization which was introduced by choosing only a subset of coordinating variables for the high-level entity. The results of each approach will be discussed in the following sections.

1) High-Level Optimizer with n, n_1 and n_2 as Variables: As mentioned in Section III-B, for keeping the consistency of the system, an optimization problem is formulated to solve the three coordination variables. The optimization problems for each partition are formulated separately. For the first partition, the formulation is as follows,

$$\begin{aligned} \underset{r_m,l_m}{min} \quad & \pi r_m^2 l_m \\ S.t. \quad & \tfrac{n_j^2 n^2 C_{mj}^2 T_{rms}^2 r_m^3}{J_{load}^2 C_m^2} + \tfrac{2 n_j C_{mj} T_{rms}^2}{J_{load}^2 r_m l_m C_m^2} + \tfrac{T_{rms}^2}{n^2 l_m^2 r_m^5 C_m^2} \leq 1 \\ & 0.03 \leq r_m \leq 0.07 \\ & 0.01 \leq l_m \leq 0.18 \\ & l_{bm} \leq \tfrac{l_m}{r_m} \leq u_{bm} \\ & 9 \leq n \leq 100 \end{aligned} \tag{6}$$

Similarly, for the second partition the optimization problem is formulated as,

$$\begin{aligned} \underset{r_g,b_{g1},b_{g2}}{min} \quad & \tfrac{3}{2}\pi r_g^2 (b_{g1} + b_{g2}) \\ S.t. \quad & 4 \times 10^4 \tfrac{C_{gr}^2}{\sigma_{H,max}^2 r_g^2 b} (n + \tfrac{1}{n} \textstyle\sum_{k=1}^{7} (\tfrac{2}{n})^{k-1}) \leq 1 \\ & l_{bg} \leq \tfrac{b_{g1,2}}{r_g} \leq u_{bg} \\ & 3 \leq n_1 \leq 10 \\ & 3 \leq n_2 \leq 10 \end{aligned} \tag{7}$$

And for the third partition, the optimization problem is formulated as,

$$\begin{aligned} \underset{r_s,l_s}{min} \quad & \pi r_s^2 l_s \\ S.t. \quad & \tfrac{40.5316}{2.09689 k_l} + \tfrac{(0.027864 - ISE_0^2) k_l}{2.09689} \leq 1 \\ & 0.05 \leq l_s \leq 0.1 \\ & l_{bs} \leq r_s \leq 0.1 \end{aligned} \tag{8}$$

However, for the high-level optimizer, which keeps the first and second partitions consistent, the following optimization problem is formulated,

$$\begin{aligned} \underset{n,n_1,n_2}{min} \quad & \pi r_m^2 l_m + \tfrac{3}{2}\pi r_g^2 (b_{g1} + b_{g2}) \\ S.t. \quad & n = n_1 \times n_2 \end{aligned} \tag{9}$$

The constraint of the high-level optimizer is the system consistency condition. It should be noted that the variables for the high-level optimizer are n, n_1 and n_2, and it receives r_m, l_m, r_g, b_{g1} and b_{g2} from the first and second partition optimizers as inputs. In other words, the variables of the first partition optimizer in (6) are r_m, l_m and the variables of the second partition optimizer in (7) are r_g, b_{g1} and b_{g2}. This has been illustrated in Figure 7. The results obtained by application of this approach, which is based on decomposition and coordination, are listed in Table III at which t_{ex} represents the convergence time of the methods and Er represents the

TABLE III: Results of optimization based on different approaches.

Parameters \ Approaches	Monolithic Approach	First Approach	Second Approach
r_m	$0.0300\ m$	$0.0300\ m$	$0.0300\ m$
l_m	$0.795\ m$	$0.795\ m$	$0.737\ m$
r_s	$0.0059\ m$	$0.0059\ m$	$0.0059\ m$
l_s	$0.0500\ m$	$0.0500\ m$	$0.0500\ m$
r_g	$0.0453\ m$	$0.0453\ m$	$0.0442\ m$
$b_{g1,2}$	$0.0271\ m, 0.0120\ m$	$0.0271\ m, 0.0120\ m$	$0.0298\ m, 0.0135\ m$
$n_{1,2}$	$5.6682, 14.8815$	$5.6682, 14.8815$	$5.9759, 16.7338$
n	$84, 3518$	$84, 3518$	100
f^*	$6.08 \times 10^{-4}\ m^3$	$6.08 \times 10^{-4}\ m^3$	$6.1195 \times 10^{-4}\ m^3$
Er	0%	0%	0.655%
t_{ex}	9.5 sec	12 sec	6.3 sec

error with respect to the global optima reported in [24]. All the computations have been performed on a computer with CPU frequency of 2.6 GHz and the memory of 16 GB. The results are identical to the results reported in [24], which means that the methodology has converged to the global optimum. In addition, this approach is computationally efficient since it has converged in approximately 12 seconds, averaged on 10 runs, which is a substantial improvement over the implementation of the same example with genetic algorithm which has reached the global optimum in 80 seconds based on [24]. However, compared to the geometric programming based solver, which converges in approximately 9.5 seconds on the same computer, averaged on 10 runs, it's slower in its convergence due to the nature of the methodology. In Section IV, the reasons for slower convergence will be discussed.

2) High-Level Optimizer with n as Variable: In this case, the formulations are identical to the ones presented in Section III-B1 except for the constraint of the high-level optimizer which is included in the constraints of the second partition. This is illustrated in Figure 8.

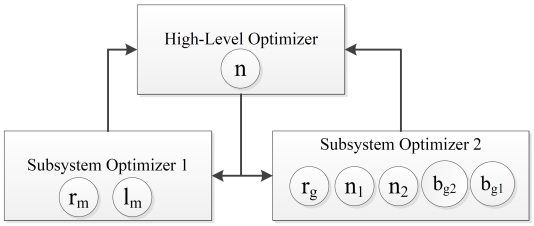

Fig. 8: Optimization based on the second approach.

The results obtained by application of this approach are provided in Table III. Note that this approach is also based on decomposition and coordination. In this case, the obtained results are not global optimum. As mentioned in II, for the purposes that this methodology serves, a small tolerance in results is acceptable. In this case, the error is between the global optimum and local optimum for the objective function of the optimization problem is 0.655%, which is considered to be an acceptable margin of error. On the other hand, the execution time and computational demand of this optimization approach is shown to be low; the algorithm converges in approximately 6.3 seconds, averaged on 10 runs.

IV. Discussions

In Section III, the methodology was applied to the mechatronic system and the results were demonstrated. In the first case, where all of the three coordinating variables were utilized to keep the system consistent through the high-level optimizer, it was shown that the methodology is slower than optimizing the system without partitioning. In previous studies such as [12], it has been shown that if the optimization problem of a complex system is solved as a whole (i.e. without decomposition and coordination) it will always converge faster (assuming that the problem is convergent). Nevertheless, neither robustness in terms of convergence nor computational efficiency is guaranteed if the problem scales to a larger number of variables. However, the decomposition and coordination based methodologies such as the one presented in this paper, are proven to be robust in terms of convergence [12]. Additionally, the proposed method is scalable because it decomposes a large optimization problems with many design variables to a hierarchy of several smaller optimization problems. Finally, the methodology presented in this paper was successful in converging to a local optimum within less than 1% margin of the global optimum, with more computational efficiency compared to [24]. In both approaches, since the optimization is based on decomposition and coordination, the methodology has the potential to be robust in terms of convergence and scalable for even higher scale problems.

The partitioning of the system in this paper has been done through decomposition of DSM. Alternatively, this decomposition can be formulated as a multi-objective optimization problem in accordance with the complexity measures described in Section II-B. However, a mathematical formalization for the complexity measures and system representation through graphs is required to formulate the aforementioned optimization problem. Since this problem formulation is under study, a partitioning algorithm for DSM is utilized to facilitate the application of the methodology [21], [22]. It should be mentioned that this partitioning is not optimal, and is merely used to demonstrate the potential benefits of the system optimization through partitioning and coordinating the partitions.

V. Conclusions and Future Work

In this paper, a method for design optimization of CPS was proposed that is based on decomposition of the complex problem into smaller sub-problems, local optimization of these

sub-problems and then coordination of the results with a higher level entity. This methodology serves to decrease the computational demand of the design process. The results are either identical to the optimum derived from the solution of the complex problem without partitioning, or within an acceptable margin of the objective function's optimal value. To demonstrate the efficiency of the methodology, it was applied to a mechatronic system. The results show that the methodology is able to reach the global optimum with a substantially decreased computational effort compared to global optimization methods such as genetic algorithm. In addition, the results demonstrate that the computational efficiency is even further improved if a margin of error in the neighborhood of the global optimum is defined for the final solution. In the case of the mechatronic system, the methodology proved to be faster than geometric programming based solver [24], at which the problem is not decomposed, while compromising less than 1% in optimality. Furthermore, the methodology proved to be robust in convergence if the global optimum is to be reached, which leads to scalability. It should be noted that this framework is not limited to convex problems

In the future, the complexity measures will be formalized. This formalization would yield to a mathematical definition of complexity by formal definition of dependencies. Accordingly, the multi-objective optimization problem for partitioning the system will be formulated using these formalizations. In addition, multi-level approaches will be investigated to further improve the coordination of the partitions, and the whole framework will be mathematically formalized. The efficiency of the methodology would be investigated on optimization problems of larger scale and more complex systems and comparative studies will be conducted.

ACKNOWLEDGMENT

This research was partially funded by the European Unions Horizon 2020 Framework Programme for Research and Innovation under grant agreement no 674875 (oCPS Marie Curie Network).

REFERENCES

[1] D. Broman, E. A. Lee, S. Tripakis, and M. Törngren, "Viewpoints, formalisms, languages, and tools for cyber-physical systems," in *Proceedings of the 6th International Workshop on Multi-Paradigm Modeling*. ACM, 2012, pp. 49–54.

[2] A. Qamar, M. Törngren, J. Wikander, and C. During, "Integrating multi-domain models for the design and development of mechatronic systems," in *7th European Systems Engineering Conference EuSEC 2010*. INCOSE, 2010.

[3] E. J. Cramer, J. E. Dennis, Jr, P. D. Frank, R. M. Lewis, and G. R. Shubin, "Problem formulation for multidisciplinary optimization," *SIAM Journal on Optimization*, vol. 4, no. 4, pp. 754–776, 1994.

[4] K. IM, "Development and application of the collaborative optimization architecture in a multidisciplinary design environment," 1995.

[5] J. F. Rodrıguez, J. E. Renaud, B. A. Wujek, and R. V. Tappeta, "Trust region model management in multidisciplinary design optimization," *Journal of Computational and Applied Mathematics*, vol. 124, no. 1-2, pp. 139–154, 2000.

[6] J. S.-S. Langley, J. S. Agte *et al.*, "Bi-level integrated system synthesis (bliss)," 1998.

[7] J. Sobieszczanski-Sobieski, T. D. Altus, M. Phillips, and R. Sandusky, "Bilevel integrated system synthesis for concurrent and distributed processing," *AIAA journal*, vol. 41, no. 10, pp. 1996–2003, 2003.

[8] M. Xiao, X. Shao, L. Gao, and Z. Luo, "A new methodology for multi-objective multidisciplinary design optimization problems based on game theory," *Expert Systems with Applications*, vol. 42, no. 3, pp. 1602–1612, 2015.

[9] A. de Wit and F. Van Keulen, "Overview of methods for multi-level and/or multi-disciplinary optimization," in *51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference 18th AIAA/ASME/AHS Adaptive Structures Conference 12th*, 2010, p. 2914.

[10] N. P. Tedford and J. R. Martins, "Benchmarking multidisciplinary design optimization algorithms," *Optimization and Engineering*, vol. 11, no. 1, pp. 159–183, 2010.

[11] J. R. Martins and A. B. Lambe, "Multidisciplinary design optimization: a survey of architectures," *AIAA journal*, vol. 51, no. 9, pp. 2049–2075, 2013.

[12] J. T. Allison, "Complex system optimization: A review of analytical target cascading, collaborative optimization, and other formulations," *MSs thesis, Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI*, 2004.

[13] L. Chen, Z. Ding, and S. Li, "A formal two-phase method for decomposition of complex design problems," *Journal of Mechanical Design*, vol. 127, no. 2, pp. 184–195, 2005.

[14] L. Chen and S. Li, "Analysis of decomposability and complexity for design problems in the context of decomposition," *Journal of Mechanical Design*, vol. 127, no. 4, pp. 545–557, 2005.

[15] S. Li, "A matrix-based clustering approach for the decomposition of design problems," *Research in Engineering Design*, vol. 22, no. 4, p. 263, 2011.

[16] J. T. Allison, M. Kokkolaras, and P. Y. Papalambros, "Optimal partitioning and coordination decisions in decomposition-based design optimization," *Journal of Mechanical Design*, vol. 131, no. 8, p. 081008, 2009.

[17] D. V. Steward, "The design structure system: A method for managing the design of complex systems," *IEEE transactions on Engineering Management*, no. 3, pp. 71–74, 1981.

[18] S. F. Alyaqout, D. L. Peters, P. Y. Papalambros, and A. G. Ulsoy, "Generalized coupling management in complex engineering systems optimization," *Journal of Mechanical Design*, vol. 133, no. 9, p. 091005, 2011.

[19] T. R. Browning, "Applying the design structure matrix to system decomposition and integration problems: a review and new directions," *IEEE Transactions on Engineering management*, vol. 48, no. 3, pp. 292–306, 2001.

[20] M. Törngren and U. Sellgren, "Complexity challenges in development of cyber-physical systems," *To appear in Principles of modeling, LNCS post-proceedings*, Essays dedicated to Edward Lee, 2018.

[21] C. I. Gutierrez, "Integration analysis of product architecture to support effective team co-location," Ph.D. dissertation, Massachusetts Institute of Technology, 1998.

[22] R. E. Thebeau, "Knowledge management of system interfaces and interactions from product development processes," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[23] K. A. Dowsland and J. M. Thompson, "Simulated annealing," in *Handbook of natural computing*. Springer, 2012, pp. 1623–1655.

[24] Y. Li, A. Duan, A. Gratner, and L. Feng, "A geometric programming approach to the optimization of mechatronic systems in early design stages," in *Advanced Intelligent Mechatronics (AIM), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1351–1656.

[25] F. Roos, "Towards a methodology for integrated design of mechatronic servo systems," Ph.D. dissertation, KTH, 2007.

[26] D. Malmquist, "A tool for holistic optimization of mechatronic design concepts," Ph.D. dissertation, KTH Royal Institute of Technology, 2015.