Swarm-based Evaluation of Nonparametric SysML Mechatronics System Design

Mohammad Chami*, Haitham Bou Ammar†, Holger Voos*, Karl Tuyls† and Gerhard Weiss†
*Research Unit in Engineering Science, University of Luxembourg, Luxembourg
†Department of Knowledge Engineering, Maastricht University, The Netherlands

Abstract—**The design of a mechatronics system is considered one of the hardest challenges in industry. This is mainly due to the multidisciplinary nature of the design process that requires the knowledge integration of the participating disciplines. Previously, we have proposed SysDICE a framework that is capable of: (1) modeling the multidisciplinary information of mechatronics systems using SysML and (2) adopting a nonparametric technique for evaluating such a SysML model. In SysDICE the optimization that led to the determination of the best alternative combinations for satisfying the requirements was time-costly and discarded prohibited combinations. This paper contributes by: (1) proposing an effective method for restricting the set of possible alternative combinations and (2) employing a swarm intelligence based optimization scheme which significantly reduces the computational cost of SysDICE.**

I. Introduction

Mechatronics system design exhibits a multidisciplinary nature by aggregating various engineering disciplines (i.e., mechanical, electrical, software and control), project and business management fields. This nature imposes a substantial challenge that deals with integrating the involved human factors with their methodologies, modeling languages and software tools for the aim of attaining an efficient system design.

In theory, the course of system design from idea creation to product disposal has been successfully proposed (e.g. [1], [2]). However, the industrial development techniques are still mono-disciplinary [3]. Particularly, the integration phase is handled at later stages, which makes the procedure expensive, cost and time in-efficient. Therefore, an early integrated evaluation approach of the whole system design is still required.

So far, little attention has been given to the collaborative work for evaluating designs in a sequel of making the procedure adaptable, efficient, and intelligent. Recently, SysDICE [4], a framework that makes use of the System Modeling Language (SysML) [5] to model a mechatronics system in order to evaluate its conceptual design, was proposed. SysDICE further contributed by adopting Gaussian Processes (GPs) for a non-parametric execution of the SysML model. These were later used with the help of a mathematically formalized optimization problem to attain a combination of components that best satisfy a set of prioritized requirements. Though SysDICE has shown to be successful, it suffered from two problems. First, it does not take into account the filtering of prohibited alternative combinations. Second, the optimization method, namely, Conjugate Gradient Descent (CGD), suffers from complexity problems especially once it comes to dealing with complex systems, i.e. large number of requirements, alternatives and components.

The work presented in this paper contributes by solving the above problems. Namely, the actual SysML model is extended by modeling the components interfaces using the internal block definition diagram (ibd) in order to restrict the number of alternative combinations to the possible ones only. Moreover, a gradient-free approach is developed to deal with the evaluation of the SysML model. Particularly, biologically inspired algorithms from artificial intelligence are used to aid in the evaluation and execution of the SysML model. These algorithms rely on Particle Swarm Optimization (PSO) and are used to reduce the computational complexity of SysDICE. We further, show the applicability and the advantage of using such algorithms by conducting design experiments to model a differential drive robot.

II. Background

A. SysML for Mechatronics System Design

From a mechatronics engineering perspective, the methodology for designing and developing mechatronic products differs mainly due to the nature of the system under consideration and to the variety between its disciplines. Although a wide range of similar design approaches have been proposed, in literature, it is agreed that there is no one accepted methodology for mechatronics [6] and in practice companies are individually developing their own techniques. Hereby, SysDICE is based on one of the popular mechatronics design frameworks, the VDI2206 guideline [2].

Following the Object Management Group [5], the Systems Modeling Language (SysML) is defined as *"a general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems that may include hardware, software, information, personnel, procedures, and facilities"*. SysML is constructed to be a software engineering extension of a customized subset of the *Unified Modeling Language* (UML) for system engineering applications.

A generalized common language for modeling the multidisciplinary information in mechatronics design is still missing. SysML with its diagrams deals with this problem, while offering a general-purpose approach, and has been already adopted successfully during the last few years for modeling mechatronics systems as in [7], [8], [9].

B. Swarm Intelligence and Particle Swarm Optimization

In this paper, a widely adopted and mostly understood population based optimization algorithm, the Particle Swarm Optimizer (PSO), is applied. PSO was first proposed by [10] as a technique to simulate the social behavior of bird flocking. The idea behind PSO is having a population of candidate solutions, referred to as particles, where they are moved around in the search space according to simple formulae over the particle's positions and velocities. PSO is a gradient-free approach where on the contrary to gradient algorithms and quasi-newton methods it does not assume the optimization problem to be differentiable. Algorithm 1 represents a general description of the main steps in PSO. The procedure starts by initializing the particles randomly in each of the problem dimensions. The goodness of the fit solutions is then evaluated as seen in lines $3-9$ of the algorithm. Later, in lines $11-15$ a neighborhood search for the best performing particle is conducted and the goal state is updated accordingly. Finally, the particles are moved around in the search space according to the velocity and position update functions.

Algorithm 1 Particle Swarm Optimization

```
Require: Number of particle swarms $N$, optimization problem (e.g., Cost
    function $J$), number of dimensions $D$, index of neighbors to be consid-
    ered $k$, minimum and maximum velocity ranges $(V_{min}, V_{max})$
 1: Initialize each swarm in each dimension randomly
 2: repeat
 3:     for $i = 1$ to $N$ do
 4:         if $J(\mathbf{x}^{(i)}) > J(\mathbf{p}^{(i)})$ then
 5:             for $d = 1$ to $D$ do
 6:                 $p_d^{(i)} = x_d^{(i)}$
 7:             end for
 8:         end if
 9:     end for
10:     Set the index of the best performing neighbor arbitrary (e.g., $g = i$)
11:     for $j = 1$ to $k$ do
12:         if $J(\mathbf{p}^{(j)}) > J(\mathbf{p}^{(g)})$ then
13:             $g = j$
14:         end if
15:     end for
16:     for $d = 1$ to $D$ do
17:         Update the velocity and position according to,
                $v_d^{(i)}(t) = f(x_d^{(i)}(t-1), v_d^{(i)}(t-1), p_d^{(i)}, p_d^{(g)})$
                $v_d^{(i)} \in (V_{min}, V_{max})$
                $x_d^{(i)}(t) = g(v_d^{(i)}(t), x_d^{(i)}(t))$
18:     end for
19: until goal solution is fixed
```

After it was proposed [10], different PSO algorithms were developed in literature. The difference between these algorithms are the update functions of the particle's velocities and positions (lines $16-18$ of Algorithm 1). In this paper four different types of PSO algorithms are applied. Namely, we use: (1) Common PSO with inertia [10], (2) Trelea model one PSO [11], (3) Trelea model two PSO [11], and (4) Clerc model PSO with constriction coefficent [12]. The technicalities of the update functions for the velocities and positions are explained in Section V as needed.

III. Related Work

AI methods have been proposed to aid the mechatronic design process. For instance, in [13] the design activity optimization was solved using a heuristic-based hybrid search algorithm and in [14] a maximum likelihood estimation method for determining the unknown design parameters based on given information was conducted. The application of ACO for combinational optimization and PSO for continuous optimization is described in [15]. An efficient SI based algorithm for multi-objective optimization is presented in [16] where the corporation of a Pareto dominance relation into PSO was proposed. The main problem in these existing approaches relates to the high effort in capturing the interdisciplinary information to be used in AI. Although others, as [17], contribute by proposing an integrated design evaluation, with graph based models, and using PSO for encoding such models, these are considered non-generalizable. Several other approaches have solved the integration issues in mechatronics, as with a central high-level model framework among the different tools [18] or a constraint classification of mechanical and electrical domains [19]. These exhibit similar problems to the former as such proposed methods are also non-generalizable, where previously unconsidered disciplines can hardly be integrated later. One of the solutions to solve the generalization problem used in recent reasearch work is SysML. For instance in [7], SysML was used to specify the central view-model of the mechatronics system. In [8], the system-level modeling with SysML was adopted to support mechatronic design. Although SysML supports in modeling mechatronics systems, its execution and evaluation is still an open topic and an intelligent, adaptable and efficient execution is still demanded.

Formalization of SysML has been recently considered. For instance, Petri nets and temporal logic LTL are used in [20] to formalize the system behavior and requirements, and [21] encoded some SysML diagrams with description logic for formal semantics. Compared to these approaches our framework take a step further in incorporating noisy models and uncertainties that are typically not available once adopting logical descriptions.

IV. SysDICE Preliminaries and Contributions

A. SysML Model Generation

SysDICE [4] uses a three layer hierarchical architecture to model the design of a mechatronics system. Namely, SysDICE starts with modeling the requirements of a mechatronics system using the ≪*requirement*≫ block within the req diagram of SysML. In this paper we focus on numerical requirements where each is specified by a desired value $v_d^{(i)}$ and a priority $w^{(i)}$ with $i = \{1, 2, \dots, k\}$ where k describes the number of requirements. Following a similar trend to industry, the system is then decomposed into its constituent subsystems and their corresponding components. This is achieved through the SysML ≪*block*≫ element and the ≪*composition*≫ association within the bdd diagram. Each component of the system has various alternatives that are modeled with a stereotyped

≪*block*≫ in order to represent their uniqueness in a possible conceptual design solution. These are specified by their corresponding properties such as the weight, the price, the power consumption and so forth. The relations among the latter properties are modeled using the ≪*constraintProperty*≫ within the par diagram.

B. Mathematical Formalization and Evaluation

Given a set of k requirements with their desired values and priorities, w, we define, $\mathbf{v}_d = [v_d^{(1)}, \ldots, v_d^{(k)}] \in \mathbb{R}^{k\times 1}$ and $\mathbf{W}_{k,k} = diag(\mathbf{w})$ to represent the vector of desired values and the diagonal matrix of priorities respectively. We further define $\mathbf{v} = [v^{(1)}, \ldots, v^{(k)}]$, to represent the output of the constraint equations. We assume these values to be uncertain with a Gaussian noise and that the priorities weight the requirements in each of the k dimensions. Therefore, the likelihood for a desired value to occur is defined by:

$$p(v_d^{(i)}|v^{(i)};\sigma^2, w^{(i)}) = \prod_{i=1}^{k} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2} w_{i,i}(v_d^{(i)} - v^{(i)})^2\right) \tag{1}$$

The maximization of the logarithm of Equation 1 can be written in a matrix form as the minimization of the following:

$$\min_{\mathbf{v}} \frac{1}{2}[\mathbf{v} - \mathbf{v}_d]^T \mathbf{W}[\mathbf{v} - \mathbf{v}_d] \tag{2}$$

Equation 2 represents the weighted requirement satisfaction problem. In other words, the solution of the minimization problem is seeking the optimal value $\mathbf{v}^\star$ that minimizes the error with respect to the prioritized requirements (i.e., $\mathbf{v}^\star = \arg\min_{\mathbf{v}} \frac{1}{2}[\mathbf{v} - \mathbf{v}_d]^T \mathbf{W}[\mathbf{v} - \mathbf{v}_d]$).

The solution at this point represents the optimal values that need to be related to the properties level. Therefore, to approximate the values of the corresponding combinations of the properties we resorted to GPs. In other words, we approximate the relations between the attained values $\mathbf{v}$ and their corresponding inputs (i.e., properties $\mathbf{P}$) using GPs. These are then substituted in Equation 2 to generate the following new optimization problem defined by the cost function $J(\mathbf{P})$ as follows:

$$\min_{\mathbf{P}} J(\mathbf{P}) = \sum_{l=1}^{m} \left(\frac{1}{2} \sum_{i=1}^{k} w_{i,i} \left(\mathcal{GP}_i(\mathbf{P}^{(l)}) - v_d^{(k)} \right)^2 \right), \tag{3}$$

where $\mathbf{P}^{(l)} = p_1^{(l)} \otimes p_2^{(l)} \cdots \otimes p_c^{(l)}$, with c being the number of components and $l = \{1, 2, \ldots, m\}$ representing the number of available alternatives.

C. Filtering of Not Allowed Component Combinations

In SysDICE [4], the number of possible alternative combinations has been obtained from the number of all possible combinations of the properties. However, this was discovered to be unsuitable when two different alternatives cannot be interfaced together in a given system. This issue is solved by extracting the components interfaces information modeled within the internal block definition diagram (ibd) of SysML (i.e., which alternative component is connected with the other) and is used to filter and discard the impossible combinations on the properties levels.

Aiming at formalizing this scheme we define, $C = \{c_{ij}\}$ to be a finite set of all components where i is the number of components forming a system and j is the number of alternatives for each component c_i (e.g., c_{11}, c_{12}, c_{13} are 3 alternatives of c_1). Furthermore, R is defined as a relation on the set C which relates the elements of C to it self and simultaneously is a subset of the Cartesian product $C \times C$. For example, we write "$c_{12}\ R\ c_{31}$" that means c_{12} *"is interfaced to"* c_{31}. Therefore, a relation R is generated (from the ibd's) each time a SysML model is parsed to hold the information of the possible interfaced components's alternatives. Later this R is used to filter out the not allowed combinations.

V. Particle Swarm Optimization for SysDICE

Minimizing Equation 3 is computationally expensive due to the complexity of the problem. In the previous work [4], Conjugate Gradient Descent (CGD) was proposed to perform the optimization. CGD is a gradient-based approach that required the derivative of Equation 3. Since the latter included the derivative of a GP which was hard to attain symbolically, a first order Taylor approximation of the derivative was used. Seeking the improvement of the computational complexity we employ PSO.

A. Particle Swarm Optimization Models

We have conducted experiments with four different PSO algorithms. In the following, each of these algorithms is described.

Common Particle Swarm Optimization with Inertia: Shi and Eberhart [10] proposed PSO with inertia coefficient. This value is multiplied by the velocity and is linearly decreased through the run. This decrease specifies whether the algorithm is exploring the search space (i.e., in the global search mode) or exploiting the current solution (i.e., local search). The update equations for the velocity and position in this algorithm is given by:

$$\begin{aligned} v_d^{(i)}(t) &= \Xi(t)(v_d^{(i)}(t-1) + c_1\phi_1(p_d^{(i)}(t-1) - x_d^{(i)}(t-1)) \\ &\quad + c_2\phi_2(p_d^{(g)}(t-1) - x_d^{(i)}(t-1)) \\ x_d^{(i)}(t) &= x_d^{(i)}(t-1) + v_d^{(i)}(t)) \end{aligned} \tag{4}$$

where c_1 represents the first social parameter, c_2 is the second social parameter, $\phi_1 \in [0,1]$ is a uniform random number, $\phi_2 \in [0,1]$ is the second random number and $\Xi(t)$ is the *weight inertia*. The value of Ξ is reduced linearly through time from $\Xi(0)$ to $\Xi(T)$ where T is the maximum allowed time for a run. This linear reduction is performed according to the following equation,

$$\Xi(t) = \frac{(T-t)(\Xi(0) - \Xi(T))}{T} + \Xi(T)$$

Terelea PSO models: The next variant of PSO algorithms that we have used are Terelea models one and two. These models are based on a dynamical analysis of the original PSO algorithm. Here the update functions of the particles' velocities and positions are expressed as follows:

$$
\begin{aligned}
v_d^{(i)}(t) &= av_d^{(i)}(t-1) + b(p_d^{(i)}(t-1)) \\
&\quad - x_d^{(i)}(t-1)) + b(p_d^{g}(t-1) - x_d^{(g)}(t-1)) \\
x_d^{(i)}(t) &= cx_d^{(i)}(t-1) + dv_d^{(i)}(t)
\end{aligned} \tag{5}
$$

After conducting a dynamical and experimental analysis of the algorithm, Trelea concluded two main parametric settings. The first, which was called Trela PSO model 1 for $a = 0.6$ and $b = 1.7$ and the second was Trelea PSO model 2 where $a = 0.729$ and $b = 1.494$. The values of c and d in both cases were set to 1.

Particle Swarm Optimization with Constriction Coefficient: To guarantee convergence of PSO, Clerc [12] suggested the usage of a constriction coefficient that bounds the dynamics of PSO. Using this method the particle's oscillations decreased as PSO in this case focuses more on the locality and the neighborhood of the past best solutions. In other words, the addition of this new constriction coefficient balances the problem of global and local search. The update equations in PSO with the a constriction coefficient are:

$$
\begin{aligned}
v_d^{(i)}(t) &= \chi(v_d^{(i)}(t-1) + c_1\phi_1(p_d^{(i)}(t-1) - x_d^{(i)}(t-1)) \\
&\quad + c_2\phi_2(p_d^{(g)}(t-1) - x_d^{(i)}(t-1))) \\
x_d^{(i)}(t) &= x_d^{(i)}(t-1) + v_d^{(i)}(t),
\end{aligned} \tag{6}
$$

where $\chi = \frac{2k}{|2-\phi-\sqrt{\phi^2-4\phi}|}$ with $\phi = c_1 + c_2,\ \phi > 4$ being the constriction coefficient. After a deep analysis of the algorithm Clerc suggested the usage of $k = 1$ and $c_1 = c_2 = 2$.

B. Swarm SysDICE

Given a set of prioritized requirements start by modeling the mechatronics system using SysML as described in Section IV-A. Having all the alternatives for the different components as well as the requirements, the model is transformed into MATLAB where it is to be executed. The execution is the determination of the best alternative combination that solves the problem described in Equation 3. The solution of this problem is conducted using one of the SI algorithms. In Algorithm 2 the main contributions of this paper are summarized. Namely, after modeling the system using SysML and transforming it to MATLAB, the new framework filters out non allowed combinations as described in Section IV-C and then makes use of PSO to avoid the computational problems encountered once using CGD or any other gradient based approach. Lines 5 – 6 of Algorithm 2 reflect that the attained optimal combination by solving Equation 3 might not be present within the allowed properties data set. Therefore, a k-means search for the closest alternative combination in the components space is conducted.

VI. Experiments and Results

To test the efficacy of the proposed approach, four different experiments are conducted on the design of a differential drive robot.The *e-puck*, top-right of Figure 1, is an example of such

Algorithm 2 Swarm SysDICE

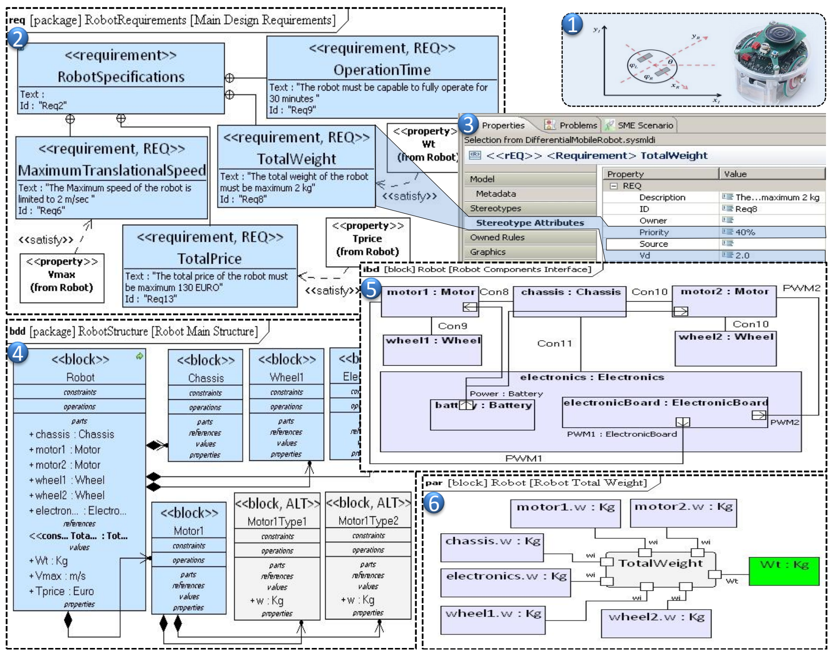
```
Require: Set of numerical requirements $\mathbf{v}_d \in \mathbb{R}^{k\times 1}$, set of priorities $\mathbf{w} \in \mathbb{R}^{k\times 1}$, set of components with their properties $\mathbf{P} = \{p_1^{(l)}, \dots, p_c^{(l)}\}$, possible component alternatives $m$, tolerance error $\epsilon$
1: Use SysML to model the mechatronics system
2: Transform the obtained SysML model to MATLAB
3: Filter the not allowed alternatives combinations
4: Determine $\mathbf{P}^\star$ by solving Equation 3 using either:
     • Conjugate Gradient Descent
     • PSO with Inertia:
         – Use Algorithm 1 with Equation 4
     • PSO with Terlea models (one and two)
         – Use Algorithm 1 with Equation 5
     • PSO with Constriction Coefficients
         – Use Algorithm 1 with Equation 6
5: Attain the closest possible combination of components to $\mathbf{P}^\star$ by solving,
     $\min_{\mathbf{P}} \sum_{l=1}^{m} \left(||\mathbf{P}^{(l)} - \mathbf{P}^\star||_2^2\right)$
6: return $\arg\min_{\mathbf{P}} \sum_{l=1}^{m} \left(||\mathbf{P}^{(l)} - \mathbf{P}^\star||_2^2\right)$
```

a robot. The application of the proposed approach is detailed in: (1) modeling the robot using SysML, (2) using the mathematical formulation and GPs to find the optimal combination of component alternatives to satisfy different requirements' configurations, and (3) adopting the four PSO algorithms to compare the computational efficiency of SysDICE.

A. SysML Model Generation

During the early stages system engineers transform stakeholders' objectives to its engineering requirements' representation in order to start analyzing the most suitable conceptual solution. Here, SysML is used to model the requirements, components structure and interfaces, and constraint interrelationships between properties. SysML modeling was done using the open source tool TOPCASED-SysML. Figure 1 shows the four types of SysML diagrams: req, bdd, ibd and par diagrams used to model the mobile robot. Figure 1(2) shows a part of the main design requirements: the *TotalWeight*, the *TotalPrice*, the *MaximumTranslationalVelocity*, and the *OperationTime*. Each is stereotyped as *"REQ"* to allow for the addition of the requirements' properties (i.e., v_d and w). Similarly all other requirements were modeled. Each *REQ* must be satisfied by a value of a design entity (i.e. component, property or even a system) and this is done using the ≪*satisfy*≫ association.

The robot components are modeled in the bdds with blocks and components hierarchy, using the SysML ≪*composition*≫ association. In Figure 1(4) the details of modeling these components are presented. Each component is described using its own block that holds certain properties typically needed in the design phase. In this example the robot consisted of 7 components, each having its own alternatives. These alternatives are modeled with blocks that are stereotyped as *"ALT"* so to indicate the multi-alternatives for each component during the transformation (e.g., Motor1Type1, Motor1Type2). Furthermore, the interfaces between these components are modeled with the ibd's, see Figure 1(5).


req [package] RobotRequirements [Main Design Requirements]
<<requirement>> RobotSpecifications
Text : Id : "Req2"
<<requirement, REQ>> OperationTime
Text : "The robot must be capable to fully operate for 30 minutes" Id : "Req9"
<<requirement, REQ>> MaximumTranslationalSpeed
Text : "The Maximum speed of the robot is limited to 2 m/sec" Id : "Req6"
<<requirement, REQ>> TotalWeight
Text : "The total weight of the robot must be maximum 2 kg" Id : "Req8"
<<property>> Wt (from Robot)
<<satisfy>>
<<property>> Vmax (from Robot)
<<requirement, REQ>> TotalPrice
Text : "The total price of the robot must be maximum 130 EURO" Id : "Req13"
<<property>> Tprice (from Robot)
Properties
Problems
SME Scenario
Selection from DifferentialMobileRobot.sysmldi
<<rEQ>> <Requirement> TotalWeight
Model
Metadata
Stereotypes
Stereotype Attributes
Owned Rules
Graphics
Property
Value
REQ
Description
The…maximum 2 kg
ID
Req8
Owner
Priority
40%
Source
Vd
2.0
bdd [package] RobotStructure [Robot Main Structure]
<<block>> Robot
constraints
operations
parts
+chassis : Chassis
+motor1 : Motor
+motor2 : Motor
+wheel1 : Wheel
+wheel2 : Wheel
+electron... : Electro...
references
<<cons... Tota... : Tot...
values
+Wt : Kg
+Vmax : m/s
+Tprice : Euro
properties
<<block>> Chassis
<<block>> Wheel1
<<block>> Motor1
<<block, ALT>> Motor1Type1
<<block, ALT>> Motor1Type2
+w : Kg
ibd [block] Robot [Robot Components Interface]
motor1 : Motor
chassis : Chassis
motor2 : Motor
Con8
Con9
Con10
Con11
PWM2
wheel1 : Wheel
wheel2 : Wheel
electronics : Electronics
Power : Battery
battery : Battery
electronicBoard : ElectronicBoard
PWM1 : ElectronicBoard
PWM1
par [block] Robot [Robot Total Weight]
motor1.w : Kg
motor2.w : Kg
chassis.w : Kg
electronics.w : Kg
TotalWeight
Wt : Kg
wheel1.w : Kg
wheel2.w : Kg

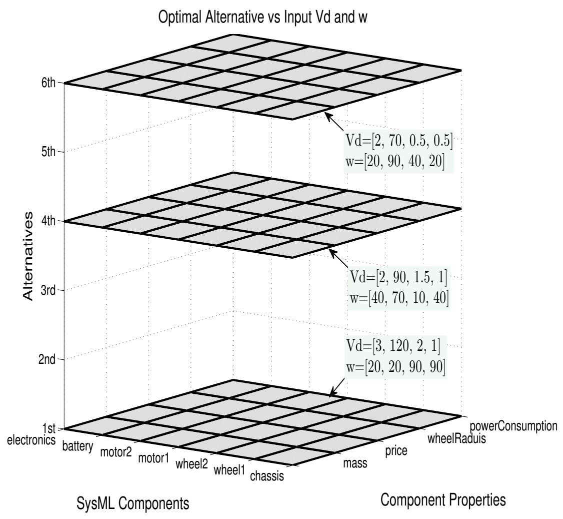


Fig. 1. SysML diagrams of a two-wheeled differential drive robot: (1) e.g. the *e-puck*, (2) req diagram with (3) the properties of the *TotalWeight* requirement (i.e., v_d and w), (4) bdd for components' structure and alternatives, (5) ibd for interfaces and (6) par diagram for the *TotalWeight* constraint property.

Fig. 2. The attained results of three different design configurations. Planes represent the best alternatives combination that satisfy the prioritized requirements. The values for these alternatives are determined by solving Equation 3.

Various *par* diagrams were used to model the mathematical equations between the component properties. Each equation is represented with a ≪*constraintProperty*≫ with its own input and output properties. For instance, the constraint *"TotalWeight"* is used in the *par*, as shown in Figure 1(6), to relate all the components' weight properties (component.w) thus indicating the value of the actual total weight of the robot W_t. Here the *TotalWeight REQ* is satisfied by this property W_t that indicates the actual value v. The kinematical, dynamical as well as other related equations, have been also modeled similarly with other par diagrams. At this stage a SysML model incorporating all the disciplines is generated. Therefore, the necessary information for system engineers is ready for evaluation and the communication burden is solved.

B. SysML Model Evaluation: An Application Example

1) Determining the Optimal Alternative Combination: We have conducted various experiments with different priorities and desired values of the requirements. The system was provided with different alternatives having different properties, such as, the mass, the price and so forth as described above. The algorithm was provided with different $\mathbf{v}_d$'s and $\mathbf{w}$'s. After the GPs were approximated, conjugate gradient descent and the PSO algorithms as described in Algorithm 2 were applied to find the optimal alternative suiting the requirements.

Figure 2 shows the results obtained, where the three axis of the graph represent the components, properties and the alternatives respectively. The different planes are the optimal alternatives resulting from different requirement values and priority configurations. For instance, in the first plane (1st alternative) the focus was more towards having a high velocity robot (i.e., 2 m/s) with high operational time (i.e., 1 hour), where both requirements were given a priority of 90%. The second plane (4th alternative) represents a moderate robot while the third (6th alternative) correspond to having a cheap price robot of 70 with a high priority (i.e., 90%). It becomes obvious from Figure 2 that the platform captures different optimal alternatives suiting different design focuses and requirements and thus being adaptable and generalizable to different requirement and or priority values.

2) PSO Results and Computational Time: We have conducted different sets of experiments to determine the computational gain attained by using the PSO models. Namely, we have first considered the time for convergence using only the platform proposed in [4]. The convergence results are shown in Figure 3. More specifically, we have fixed the values of the requirements and priorities while varying the allowed alternative combinations. It is clear from Figure 3 that the attained time for convergence increases with the number of alternatives. For instance, it varies from 15 ticks on 20 alternatives to about 43 ticks once working with 100.

These results clearly reflect that CGD doesn't scale well once increasing the number of alternatives and/or components. To reduce these computational problems, we have conducted the same experiments using the proposed PSO models. Figure 4 summarizes the achieved results while varying the number of alternatives and using the different PSO algorithms. Firstly, it is clear that PSO and due to its gradient -free advantages requires less time to converge to the optimal solution. For instance, the "worst-performing" algorithm was the common PSO with inertia where it attains about 2.5 ticks for convergence at 100 alternatives. The performance of the different

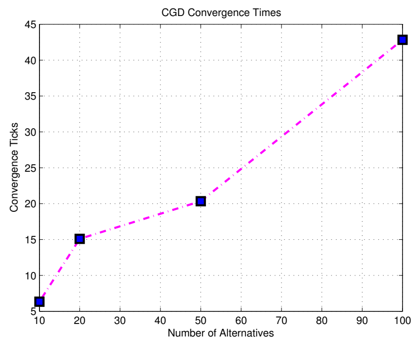


Fig. 3. Convergence results using conjugate gradient descent. The x-axis represents the variation in the possible alternatives while the y-axis represents the amount of time required for the algorithm to converge to the optimal alternative combination.

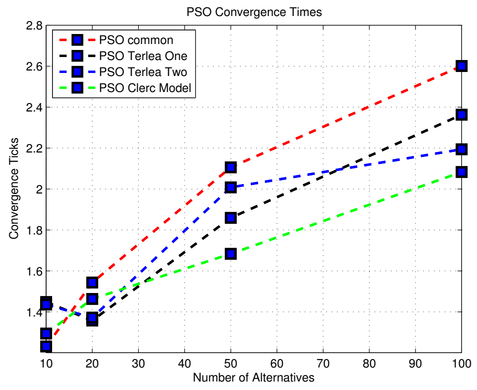


Fig. 4. Convergence results using the different proposed PSO models. The x-axis represents the variation in the possible alternatives while the y-axis represents the amount of time required for the algorithms to converge to the optimal alternative combination.

algorithms is relatively similar while the best performing one was that of Clerc with the constriction coefficient. Therefore, the presented results clearly manifest the improvements gained by adopting a population-based optimization scheme such as PSO.

VII. Conclusions and Future Work

SysDICE is a SysML-based nonparametric framework for the conceptual design evaluation of mechatronic systems. It was capable of attaining the optimal component alternative combination that best suits a set of prioritized requirements. This framework suffered from two problems. First it was not capable of filtering out non allowed combinations. Second, it adopted a gradient-based approach to solve the optimization problem. This paper targeted these two problems and proposed the usage of a gradient-free approach that was capable of reducing the computational time needed to attain the optimal behavior. Results clearly indicated the scalability of the approach and showed the improvements gained by such population based optimization algorithms.

There are a lot of interesting directions for future work. Here we mention two such directions. We plan on using transfer learning in order to transfer between different design models of various mechatronics systems as well as deal with non-numerical requirements.

References

[1] G. Pahl, W. Beitz, J. Fledhusen, and K.-H. Grote, *Engineering Design A Systematic Approach*, third edition ed., K. Wallace and L. Blessing, Eds. Springer, 2007.

[2] *VDI 2206 Design methodology for mechatronic systems*. Beuth Verlag GmbH, June 2004.

[3] F. P. Stappers, L. J. Somers, and M. A. Reniers, “Multidisciplinary Modeling - Current status and expectations in the Dutch TWINS consortium,” in *ICSSEA*, 2008.

[4] M. Chami, H. B. Ammar, H. Voss, K. Tuyls, and G. Weiss, “A Nonparametric Evaluation of SysML-based Mechatronic Conceptual Design,” in *Proceedings of the Benelux Conference on Artificial Intelligence (BNAIC)*, Maastricht, The Netherlands, 2012.

[5] “Object Management Group (OMG) Systems Modeling Language (OMG SysML^{TM}), available at http://www.omgsysml.org .” Nov 2008.

[6] T. Tomiyama, P. Gu, Y. Jin, D. Lutters, C. Kind, and F. Kimura, “Design methodologies: Industrial and educational applications,” pp. 543–565, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S000785060900170X

[7] K. Thramboulidis, “The 3+1 SysML View-Model in Model Integrated Mechatronics,” *Journal of Software Engineering and Applications (JSEA)*, vol. 3, no. 2, pp. 109–118, 2010.

[8] A. Qamar, J. Wikander, and C. During, “Designing Mechatronic Systems: A Model-Integration Approach,” in *Proceedings of the 18th International Conference on Engineering Design (ICED11)*, 2011.

[9] M. Chami, H. Seemller, and H. Voos, “A SysML-based Integration Framework for the Engineering of Mechatronic Systems,” IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications. IEEE, 2010.

[10] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4. IEEE, Nov. 1995, pp. 1942–1948 vol.4. [Online]. Available: http://dx.doi.org/10.1109/ICNN.1995.488968

[11] G. K. Jha, P. Thulasiraman, and R. K. Thulasiram, “Pso based neural network for time series forecasting,” in *Proceedings of the 2009 international joint conference on Neural Networks*, ser. IJCNN’09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 893–898.

[12] M. Clerc, “The swarm and the queen: towards a deterministic and adaptive particle swarm optimization,” in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 3, 1999. [Online]. Available: http://dx.doi.org/10.1109/CEC.1999.785513

[13] O. Mouelhi, P. Couturier, and T. Redarce, “An Artificial Intelligence Approach for the Multicriteria Optimization in Mechatronic Products Design,” in *Proceedings of the 2009 IEEE International Conference on Mechatronics and Automation*, 2009, pp. 1731–1736.

[14] X. Xu, L. Fu, and S. Fang, “Research on Product Variant Design with Uncertainty Information,” in *Proceedings of the 7th World Congress on Intelligent Control and Automation*, Chongqing, China, 2008.

[15] C. Blum and X. Li, *Swarm Intelligence in Optimization*. Springer, Natural Computing Series, Swarm Intelligence, Part I, 2008.

[16] M. J. Reddy and D. N. Kumar, “An efficient multi-objective optimization algorithm based on swarm intelligence for engineering design,” 2007.

[17] F.-Y. Huang and Y.-J. Tseng, “An integrated design evaluation and assembly sequence planning model using a particle swarm optimization approach,” 2011.

[18] A. A. A. Cabrera, M. S. Erden, M. J. Foeken, and T. Tomiyama, “High Level Model Integration for Design of Mechatronic Systems.” IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications, October 2008.

[19] K. Chen, J. Bankston, J. H. Panchal, and D. Schaefer, *A Framework for Integrated Design of Mechatronic Systems*. Springer, 2009, ch. 2, pp. 37–70.

[20] M. V. Linhares, R. S. de Oliveira, J.-M. Farines, and F. Vernadat, “Introducing the modeling and verification process in sysml,” in *IEEE International Conference. on. Emerging Technologies and Factory Automation (ETFA)*, 2007.

[21] H. Graves and Y. Bijan, “Modeling structure in description logic,” *DL2011*, 2011.