Consistency Constraint Allocation in Augmented Lagrangian Coordination

James T. Allison[1]
Senior Applications Engineer
The Mathworks, Inc.,
Natick, MA 01760
e-mail: james.allison@mathworks.com

Panos Y. Papalambros
Professor
Department of Mechanical Engineering,
University of Michigan,
G.G. Brown Building,
Ann Arbor, MI 48109
e-mail: pyp@umich.edu

Many engineering systems are too complex to design as a single entity. Decomposition-based design optimization methods partition a system design problem into subproblems, and coordinate subproblem solutions toward an optimal system design. Recent work has addressed formal methods for determining an ideal system partition and coordination strategy, but coordination decisions have been limited to subproblem sequencing. An additional element in a coordination strategy is the linking structure of the partitioned problem, i.e., the allocation of constraints that guarantee that the linking variables among subproblems are consistent. There may exist many alternative linking structures for a decomposition-based strategy that can be selected for a given partition, and this selection should be part of an optimal simultaneous partitioning and coordination scheme. This article develops a linking structure theory for a particular class of decomposition-based optimization algorithms, augmented Lagrangian coordination (ALC). A new formulation and coordination technique for parallel ALC implementations is introduced along with a specific linking structure theory, yielding a partitioning and coordination selection method for ALC that includes consistency constraint allocation. This method is demonstrated using an electric water pump design problem.
[DOI: 10.1115/1.4001525]

1 Introduction

Many engineering systems are too complex to design as a single entity, but can be divided into smaller and more manageable subproblems. Each subproblem is formulated as an optimization problem, and a coordination strategy is used to guide repeated subproblem solutions toward a consistent and optimal system design. This decomposition-based design optimization approach requires an understanding of links between the analysis functions used to compute the responses of the various parts of the system for a given design. The analysis models are represented here using a set of m inter-related analysis functions, as shown in Fig. 1.

Each analysis function may be an objective, constraint, or intermediate function. We assume here that the input and output requirements of these functions are known precisely; this is the case when the analysis functions correspond to computer simulations. The ith analysis function $\mathbf{a}_i(\mathbf{x}_i, \mathbf{y}_i)$ depends on $\mathbf{x}_i$, which is a subset of the system design variable vector $\mathbf{x}$, and on $\mathbf{y}_i$, which is composed of the components of the system coupling variable vector $\mathbf{y}$ that are input to $\mathbf{a}_i$. Coupling variables are analysis function outputs that are required as inputs to other analysis functions; $\mathbf{y}_{ij}$ is the vector of quantities passed from $\mathbf{a}_j$ to $\mathbf{a}_i$. Design variables input to $\mathbf{a}_i$ only are the local design variables $\mathbf{x}_{\ell i}$, and design variables input to $\mathbf{a}_i$ and at least one other analysis function are the shared design variables $\mathbf{x}_{si}$. The set of all shared design variables is $\mathbf{x}_s$. Coupling variables and shared design variables together comprise a system's set of linking variables $\mathbf{z}$.

Application of decomposition-based design optimization to solve a system design problem requires a priori definition of a system partition and coordination strategy. A restricted growth string (RGS) $\mathbf{p}$ of length m can be used to specify a partition [1]; the value of p_i is the subproblem that analysis function i belongs to. Numerous subjective, as well as formal, partitioning methods have been proposed; see Ref. [2] for a review. Available coordination strategy options depend on the type of system optimization formulation used. Collaborative optimization (CO) uses a master optimization problem to drive subproblems toward system optimality and consistency [3], while formulations such as analytical target cascading (ATC) [4] and augmented Lagrangian coordination (ALC) [5,6] use penalty relaxation methods in tandem with algorithms for solving systems of equations. In the latter class of formulations, subproblem solution sequence in the coordination algorithm influences computational expense. Allison et al. [2] showed that partitioning and subproblem sequence decisions are coupled, and proposed a combined partitioning and coordination (P/C) decision method for reducing problem complexity and computational expense. Coordination decisions in this method were limited to subproblem sequence. This article describes another important component of coordination decisions, consistency constraint allocation, and shows how to incorporate it into a combined partitioning and coordination decision method for ALC in an automated way. Section 2 reviews the ALC method to motivate the need for proper consistency constraint allocation. Section 3 introduces a new parallel ALC to enable parallel solution of all subproblems when the number of subproblems exceeds the number of processors. Section 4 introduces the theory for analyzing linking structures using graph theory and constraint satisfaction programming. Section 5 describes how linking structure decisions can be included in developing a partitioning and coordination strategy for parallel ALC. Section 6 demonstrates the method using an electric water pump problem, followed by a conclusion in Sec. 7.

2 Augmented Lagrangian Coordination

When a system is partitioned, some design variables may be shared across subproblems, and some coupling variable relationships may cross subproblem boundaries. These variables are termed external linking variables. ALC requires that subproblems are solved independently of each other. This is accomplished by

[1]Corresponding author.

Contributed by the Design Automation Committee of ASME for publication in the JOURNAL OF MECHANICAL DESIGN. Manuscript received September 3, 2008; final manuscript received February 24, 2010; published online June 17, 2010. Assoc. Editor: Timothy W. Simpson.

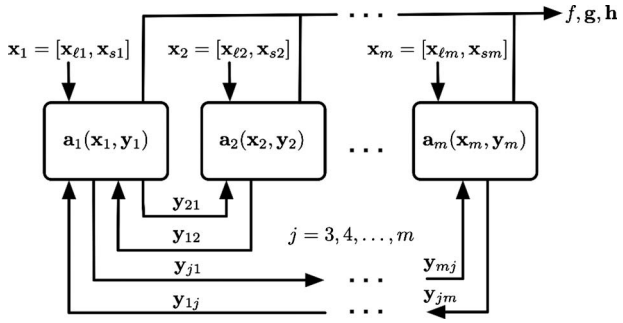

Fig. 1 Input and output relationships for a system of analysis functions

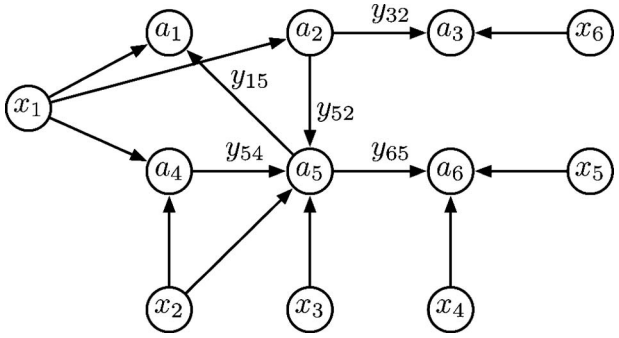

Fig. 2 Analysis function digraph for example system

using separate copies of external linking variables in each subproblem. The ALC algorithm must ensure that these copies match at convergence to guarantee system consistency.

The copies of design variables shared between subproblems i and j, local to subproblem i, are $\bar{\mathbf{x}}_s^{ij}$. The external coupling variables that passed from subproblem j to i are $\bar{\mathbf{y}}_{ij}$, and the corresponding analysis functions are $\bar{\mathbf{a}}_{ij}(\bar{\mathbf{x}}_j, \hat{\mathbf{y}}_j, \bar{\mathbf{y}}_j)$, where $\bar{\mathbf{y}}_j$ are the external coupling variables input to subproblem j, and $\bar{\mathbf{x}}_j$ are the design variables for subproblem j. Coupling variables that link analysis functions within subproblem j are the internal coupling variables $\hat{\mathbf{y}}_j$. The external linking variables between subproblems i and j are $\bar{\mathbf{z}}_{ij} = [\bar{\mathbf{x}}_s^{ij}, \bar{\mathbf{y}}_{ij}]$. ALC uses consistency constraints on external linking variables in the subproblem formulations to ensure consistency between subproblems. The external consistency constraints between subproblems i and j are:

$$\bar{\mathbf{c}}_{ij}(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j, \hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j, \bar{\mathbf{y}}_i, \bar{\mathbf{y}}_j) = [\bar{\mathbf{y}}_{ij} - \bar{\mathbf{a}}_{ij}(\bar{\mathbf{x}}_j, \hat{\mathbf{y}}_j, \bar{\mathbf{y}}_j), \bar{\mathbf{y}}_{ji} - \bar{\mathbf{a}}_{ji}(\bar{\mathbf{x}}_i, \hat{\mathbf{y}}_i, \bar{\mathbf{y}}_i), \bar{\mathbf{x}}_s^{ij} - \bar{\mathbf{x}}_s^{ji}] \tag{1}$$

Note that the components of $\bar{\mathbf{x}}_s^{ij}$ are part of the vector $\bar{\mathbf{x}}_i$, and $\bar{\mathbf{y}}_{ij}$ is part of the vector $\bar{\mathbf{y}}_i$.

Equation (1) specifies a very large number of consistency constraints; only a subset is actually required to ensure consistency. Requirements for a valid subset are provided in Sec. 4. The number of possible ways to choose (i.e., allocate) consistency constraints is very large, and is a task beyond intuition for all but the smallest system design problems. Allocation guidelines have been proposed for constructing bilevel or hierarchical consistency constraint structures for ALC implementations [5,6]. These recommendations are helpful, but do not capitalize on the potential benefit realized through tailoring ALC structure to a specific system. This article proposes an automated technique for ALC consistency constraint allocation.

After consistency constraints are selected, an augmented Lagrangian penalty function is used to relax them as follows:

$$\phi_{ij}(\bar{\mathbf{c}}_{ij}) = \mathbf{v}_{ij}\bar{\mathbf{c}}_{ij}^T + \|\mathbf{w}_{ij} \circ \bar{\mathbf{c}}_{ij}\|_2^2 \tag{2}$$

where $\mathbf{v}_{ij}$ and $\mathbf{w}_{ij}$ are vectors of penalty weights on the linear and quadratic terms, respectively, and $\circ$ indicates the Hadamard product (i.e., element-by-element multiplication).

Internal coupling variable consistency is fulfilled using auxiliary equality constraints in subproblem formulations. The analysis functions that correspond to $\hat{\mathbf{y}}_i$ are $\hat{\mathbf{a}}_i(\bar{\mathbf{x}}_i, \hat{\mathbf{y}}_i, \bar{\mathbf{y}}_i)$. The internal consistency constraints for subproblem i are:

$$\mathbf{c}_i(\bar{\mathbf{x}}_i, \hat{\mathbf{y}}_i, \bar{\mathbf{y}}_i) = \hat{\mathbf{y}}_i - \hat{\mathbf{a}}_i(\bar{\mathbf{x}}_i, \hat{\mathbf{y}}_i, \bar{\mathbf{y}}_i) = \mathbf{0} \tag{3}$$

The set of indices for subproblems with external linking variables common to subproblem i is $\mathcal{N}_i$. The design inequality and equality constraints computed by analysis functions in subproblem i are $\mathbf{g}_i$ and $\mathbf{h}_i$, respectively. The set of decision variables for subproblem i includes $\bar{\mathbf{x}}_i$, $\hat{\mathbf{y}}_i$, and $\bar{\mathbf{y}}_i$. The ALC formulation of the optimization problem for subproblem i is:

$$\begin{aligned}
&\min_{\bar{\mathbf{x}}_i, \hat{\mathbf{y}}_i, \bar{\mathbf{y}}_i} f_i(\bar{\mathbf{x}}_i, \hat{\mathbf{y}}_i, \bar{\mathbf{y}}_i) + \sum_{j \in \mathcal{N}_i | j > i} \phi_{ij}(\bar{\mathbf{c}}_{ij}(\bar{\mathbf{x}}_i, \bar{\mathbf{y}}_{ij}, \hat{\mathbf{y}}_i)) \\
&\qquad + \sum_{j \in \mathcal{N}_i | j < i} \phi_{ji}(\bar{\mathbf{c}}_{ji}(\bar{\mathbf{x}}_i, \bar{\mathbf{y}}_{ij}, \hat{\mathbf{y}}_i)) \\
&\text{subject to} \quad \mathbf{g}_i(\bar{\mathbf{x}}_i, \hat{\mathbf{y}}_i, \bar{\mathbf{y}}_i) \leq \mathbf{0} \\
&\qquad \mathbf{h}_i(\bar{\mathbf{x}}_i, \hat{\mathbf{y}}_i, \bar{\mathbf{y}}_i) = \mathbf{0} \\
&\qquad \mathbf{c}_i(\bar{\mathbf{x}}_i, \hat{\mathbf{y}}_i, \bar{\mathbf{y}}_i) = \hat{\mathbf{y}}_i - \hat{\mathbf{a}}(\bar{\mathbf{x}}_i, \hat{\mathbf{y}}_i, \bar{\mathbf{y}}_i) = \mathbf{0}
\end{aligned} \tag{4}$$

A parallel coordination strategy for ALC is described in Sec. 3. The formulation in Eq. (4) makes a distinction between shared and coupling variables, in contrast to the original ALC formulations [5,6]. This distinction allows analysis function outputs to be used directly in consistency constraints, as shown in Eq. (1). Earlier ALC formulations use an additional linking variable copy in place of analysis function outputs. This extra variable is a subproblem decision variable, and requires an additional equality constraint to ensure it matches the corresponding analysis function output. While avoiding the distinction between coupling and shared variables simplifies formulation representation, it increases the number of decision variables and constraints. The formulation presented here applies only to quasiseparable problems, which are problems that have linking variables but not linking functions. Simulation-based design problems frequently are quasiseparable. Recent ALC formulations apply also to problems with linking functions [6].

3 Parallel ALC

This section introduces a new parallel coordination approach for ALC where the number of subproblems exceeds the number of processors. An example system with six analysis functions is used to illustrate the following concepts:

$$a_1(x_1, y_{15}), \quad a_2(x_1), \quad a_3(x_6, y_{32}), \quad a_4(x_1, x_2),$$

$$a_5(x_2, x_3, y_{52}, y_{54}), \quad a_6(x_4, x_5, y_{65})$$

The structure of this system can be visualized using a directed graph representation (Fig. 2), and is represented compactly with its reduced adjacency matrix [2]:

$$\mathbf{A} = \begin{array}{c|cccccccccccc}
 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
\hline
a_1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
a_2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
a_3 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
a_4 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
a_5 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
a_6 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0
\end{array}$$

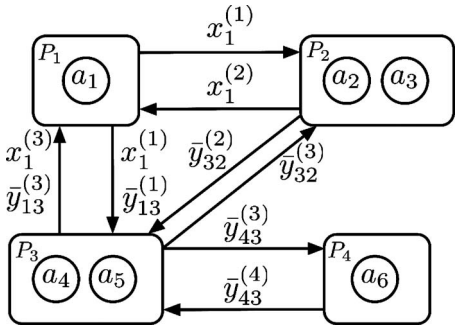

Fig. 3 Subproblem graph

The ALC algorithm specifies when each subproblem is to be solved, communicates values between subproblems, and updates penalty weights as needed. Coordination difficulty typically increases with the number of external linking variables [5]. The coordination of ALC subproblems can be viewed as the solution to a system of nonlinear equations where subproblems are optimal value functions and external linking variable copies are the unknown quantities. The subproblem i input arguments are $\bar{\mathbf{z}}_i = [\bar{\mathbf{x}}_{si}, \bar{\mathbf{y}}_i]$, and the outputs include updated values for $\bar{\mathbf{x}}_{si}$ and external coupling variable copies passed from subproblem i to other subproblems ($\bar{\mathbf{y}}_{\bullet i}$). The optimal value function for subproblem i is

$$\bar{\mathbf{z}}_{\bullet i} = [\bar{\mathbf{x}}_{si}, \bar{\mathbf{y}}_{\bullet i}] = \boldsymbol{\pi}_i(\bar{\mathbf{z}}_i) \tag{5}$$

The structure of the coordination problem can be analyzed using a directed graph where subproblems are represented by vertices, and the linking variable copies passed between subproblems correspond to arcs. Partitioning the example system from Fig. 2 using $\mathbf{p}=[1,2,2,3,3,4]$ results in the subproblem graph depicted in Fig. 3, where the boxes correspond to vertices that represent subproblems, and analysis functions that are contained within each subproblem are displayed.

The linking variable superscripts indicate subproblem of origin. Figure 3 illustrates the external linking variable copies that must be passed between subproblems. Note that the external coupling variables $\bar{y}_{13}$, $\bar{y}_{32}$, and $\bar{y}_{43}$ correspond to the coupling variables y_{15}, y_{52}, and y_{65}, respectively. While subproblems 2 and 3 share x_1, copies of x_1 are not communicated between them; Sec. 4 will explain the validity of this structure. Figure 4 illustrates the subproblem graph in more compact form.

The ALC algorithm requires an inner and an outer loop. The inner loop solves the system of equations formed by subproblem optimal value functions for the external linking variable values. The system of equations to be solved is $\bar{\mathbf{z}} = \boldsymbol{\pi}(\bar{\mathbf{z}})\mathbf{S}$, where $\bar{\mathbf{z}}$ is the set of all external linking variable copies; $\boldsymbol{\pi} = [\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots, \boldsymbol{\pi}_N]$ is the optimal value function for all subproblems; and $\mathbf{S}$ is a selection matrix that matches the outputs of $\boldsymbol{\pi}$ to the components of $\bar{\mathbf{z}}$. Note that inner loop solution alone does not ensure consistency,

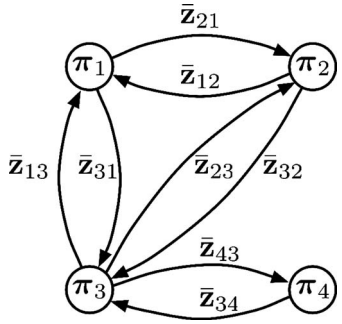

Fig. 4 Condensed subproblem graph

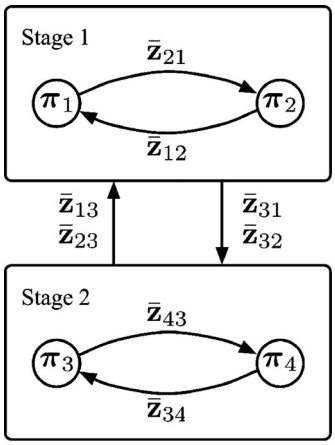

Fig. 5 Stage graph

only a fixed point. The inner loop may be solved exactly, inexactly, or even executed with a single iteration. The outer loop computes new penalty weight values using inner loop results and the method of multipliers, guiding the system toward consistency [7].

An algorithm for solving systems of nonlinear equations is used for the inner loop problem. A typical approach is to apply fixed point iteration (FPI), also known as nonlinear Gauss–Seidel, by solving each subproblem in sequence, providing the most recent linking variable information for each subproblem solution. Jacobi iteration may also be used to enable parallel solution of all subproblems. If the number of processors available is insufficient for complete parallel execution, block parallel Gauss–Seidel may be applied to blocks of subproblems sequenced into stages. The assignment of subproblems into stages is specified by the stage assignment vector $\mathbf{s}$, where the value of s_i is the stage that subproblem i belongs to. The inner loop stages for the running example system correspond to Fig. 5 if $\mathbf{s}=[1,1,2,2]$. At each inner loop iteration, subproblems 1 and 2 are solved in parallel using values for $\bar{\mathbf{z}}_{12}$, $\bar{\mathbf{z}}_{21}$, $\bar{\mathbf{z}}_{23}$, and $\bar{\mathbf{z}}_{13}$ from the previous inner loop iteration. Subproblems 3 and 4 are solved in parallel using $\bar{\mathbf{z}}_{31}$ and $\bar{\mathbf{z}}_{32}$ computed during stage 1, and $\bar{\mathbf{z}}_{43}$ and $\bar{\mathbf{z}}_{34}$ from the previous inner loop iteration. Using a stage assignment that reduces the number of values obtained from the previous iteration can help speed inner loop convergence. Global convergence proofs for the Gauss–Seidel and Jacobi approaches are available under certain conditions; see Ref. [8]. These conditions are more restrictive for Jacobi iteration than for Gauss–Seidel; thorough convergence analysis, however, is beyond the scope of the present article and is an appropriate topic for further research.

4 Linking Structure Analysis

One distinguishing characteristic of formulations for decomposition-based design optimization is linking structure; i.e., different formulations allow specific approaches to structuring consistency constraints. Most methods require a bilevel or a multilevel hierarchical constraint structure. ALC is unique in the flexibility it provides for consistency constraint structure, which enables potentially more efficient implementations where linking structure is tailored to the problem at hand. While flexibility is a beneficial feature, it may be difficult to manage. Early ALC approaches rely on bilevel or multilevel hierarchical structures to guide linking structure decisions. Deciding between the numerous nonhierachical possibilities is a task beyond intuition for all but the simplest systems. Optimization techniques can be applied effectively to this task, resulting in superior ALC implementations. A deeper understanding of consistency constraint structure is developed in this section using techniques from constraint satisfaction programming. The theory required to provably identify the

set of valid consistency constraint allocation options for ALC is developed, and Sec. 5 uses these results to define an optimal partitioning and coordination decision problem for ALC with linking structure considerations.

We will focus on consistency with respect to a single linking variable z that in general could be external or internal. The language below is appropriate for the external case. A system is consistent with respect to a linking variable when all pairs of linking variable copies are consistent as follows:

$$z^{(i)} = z^{(j)}, \quad \forall\, i \neq j, \quad i,j \in \{1,2,\ldots,n^z\} \tag{6}$$

Here $z^{(i)}$ is the copy of z associated with subproblem i, and n^z is the number of subproblems that share z. The above statement implies that $n^z(n^z-1)$ constraints are required to assure consistency with respect to z. Since $z^{(i)}=z^{(j)}$ is equivalent to $z^{(j)}=z^{(i)}$, the number of constraints can be reduced to $n^z(n^z-1)/2$ by adopting the convention that the terms in the constraint $z^{(i)}=z^{(j)}$ are ordered such that $i<j$. It will be shown that certain subsets of consistency constraints can ensure consistency of a linking variable, and that the minimum number of constraints required to ensure consistency is n^z-1. It will be demonstrated that these minimal constraint sets are linearly independent, which is a requirement of the augmented Lagrangian penalty method used in ALC.

4.1 Consistency Constraint Graphs. Montanari [9] introduced the concept of using graphs to represent constraint sets, where vertices correspond to variables and edges correspond to constraints on variables whose vertices they connect. These constraint graphs are helpful in analyzing constraint set structure and developing solutions for constraint satisfaction problems [10]; along with results from constraint programming, they provide a framework for understanding consistency constraints in system optimization. Applications of constraint satisfaction theory in engineering design have included constraint based design [11], ensuring geometric feasibility of assemblies [12] and high-speed machinery design [13].

A binary constraint is a constraint on at most two variables, and a binary constraint graph corresponds to a set of binary constraints [14]. The set of $n^z(n^z-1)/2$ binary consistency constraints on a linking variable can be represented by the complete undirected graph K_{n^z}. An edge $\{i,j\}$ represents the constraint $z^{(i)}=z^{(j)}$, which can be expressed in a negative null form as $z^{(i)}-z^{(j)}=0$. A convenient representation of this constraint is:

$$\boldsymbol{\theta}_{ij}\tilde{\mathbf{z}}^T = 0 \tag{7}$$

where $\boldsymbol{\theta}_{ij}$ is the constraint vector that corresponds to edge $\{i,j\}$, and $\tilde{\mathbf{z}}$ is the vector of all n^z copies of the linking variable z. More precisely,

$$\boldsymbol{\theta}_{ij} = \mathbf{e}_i - \mathbf{e}_j \tag{8}$$

$$\tilde{\mathbf{z}} = [z^{(1)}, z^{(2)}, \ldots, z^{(n^z)}] \tag{9}$$

where $\mathbf{e}_i$ is the ith unit vector of length n^z. Two constraints are adjacent if their corresponding constraint graph edges are adjacent (i.e., they share a common variable). A consistency constraint graph G_c is defined as a subgraph of K_{n^z} that corresponds to a subset of the $n^z(n^z-1)/2$ consistency constraints. The consistency constraint matrix $\boldsymbol{\Theta}$ for G_c is composed of all constraint (row) vectors $\boldsymbol{\theta}_{ij}$ that correspond to edges in G_c. The edges in G_c specify which consistency constraints are to be used in an ALC solution process.

4.2 Valid Consistency Constraint Graphs. Not every possible consistency constraint graph is valid for use with ALC. A consistency constraint graph is valid if its associated constraints are equivalent to the constraints specified by K_{n^z}, and if the rows of the corresponding $\boldsymbol{\Theta}$ are linearly independent. The first requirement ensures complete consistency of the associated linking variables, and the second is necessary for the success of the augmented Lagrangian penalty method used in ALC. After the development of preliminary concepts, necessary and sufficient conditions for the validity of constraint graphs will be given.

Two sets of constraints are equivalent if their feasible domains are equal. The task of finding reduced sets of constraints equivalent to some original set is known as problem reduction. A constraint is redundant if its removal does not change the feasible domain of a constraint set. The composition of adjacent constraints can induce implicit constraints. For example, if the constraints $z^{(2)}=z^{(5)}$ and $z^{(5)}=z^{(7)}$ are specified explicitly in the problem linking structure, the constraint $z^{(2)}=z^{(7)}$ will be satisfied implicitly if the two explicit constraints are met. A constraint is said to be explicit if its corresponding edge exists in G_c, and implicit if it does not. A constraint is redundant if it is both explicit and implicit [10]. The properties of consistency constraint graphs enable easy identification of implicit and redundant constraints for the purpose of problem reduction. A consistency constraint graph is minimal if it specifies the fewest number of constraints required to ensure consistency.

Identification of implicit constraints requires application of a binary operator called constraint composition that generates a new constraint from two adjacent constraints [14].

DEFINITION. *Let* $\gamma_1(i,j)$ and $\gamma_2(j,k)$ be two binary constraints with a common variable $(z^{(j)})$ corresponding to vertex j, and let their composition be $\gamma_c(i,k)$. A binary constraint composition is valid if values for $z^{(i)}$ and $z^{(k)}$ satisfy $\gamma_c(i,k)$ if and only if there exists a value of $z^{(j)}$ such that $\gamma_1(i,j)$ and $\gamma_2(j,k)$ are satisfied.

In a consistency constraint graph two constraints with a common variable can be composed to form an implicit constraint by taking the vector sum of the corresponding constraint vectors.

PROPOSITION 4.1. *The composition of the consistency constraints defined by* $\boldsymbol{\theta}_{ij}$ *and* $\boldsymbol{\theta}_{jk}$ *with the common variable* $z^{(j)}$ *is* $\boldsymbol{\theta}_{ik}=\boldsymbol{\theta}_{ij}+\boldsymbol{\theta}_{jk}=\mathbf{e}_i-\mathbf{e}_j+\mathbf{e}_j-\mathbf{e}_k=\mathbf{e}_i-\mathbf{e}_k$.

Proof. Let a_i and a_k be values for $z^{(i)}$ and $z^{(k)}$, respectively, such that $\boldsymbol{\theta}_{ik}\tilde{\mathbf{z}}^T=0$ is satisfied. By definition of $\boldsymbol{\theta}_{ik}$, $a_i=a_k$. By selecting a value a_j for $z^{(j)}$ such that $a_i=a_j=a_k$, the constraints $\boldsymbol{\theta}_{ij}\tilde{\mathbf{z}}^T=0$ and $\boldsymbol{\theta}_{jk}\tilde{\mathbf{z}}^T=0$ consequently are satisfied. Let b_i, b_j, and b_k be values for $z^{(i)}$, $z^{(j)}$, and $z^{(k)}$, respectively, that satisfy $\boldsymbol{\theta}_{ij}\tilde{\mathbf{z}}^T=0$ and $\boldsymbol{\theta}_{jk}\tilde{\mathbf{z}}^T=0$. Since this satisfaction implies $b_i=b_j$ and $b_j=b_k$, $b_i=b_k$ and the composed constraint $\boldsymbol{\theta}_{ik}\tilde{\mathbf{z}}^T=0$ is satisfied. Therefore, $\boldsymbol{\theta}_{ik}=\boldsymbol{\theta}_{ij}+\boldsymbol{\theta}_{jk}$ is a valid constraint composition. ■

A higher than binary constraint composition is defined by the recursive application of a binary constraint composition. Binary consistency constraints that share a common variable have corresponding edges that are incident to the common variable vertex. At each stage of recursive composition a new edge can be included in the composition if it has a common vertex with the implicit edge generated by the intermediate composition. This occurs when all edges in a set to be composed lie in a connected path on G_c. Suppose p_{ij} is a connected path of length δ between the vertices i and j defined by the sequence of unique vertices $\langle v_1, v_2, \ldots, v_\delta, v_{\delta+1}\rangle$ where $v_1=i$ and $v_{\delta+1}=j$. The constraint vector resulting from the extended composition of edges in p_{ij} is $\boldsymbol{\theta}_{ij}=\sum_{\{k,l\}\in p_{ij}|k<l}\boldsymbol{\theta}_{kl}=\mathbf{e}_i-\mathbf{e}_j$.

PROPOSITION 4.2. *A constraint defined by* $\boldsymbol{\theta}_{ij}$*, whether implicit or explicit, can be obtained through composition if and only if a path* p_{ij} *exists in* G_c.

Proof. If a path p_{ij} exists in G_c, extended constraint composition can be applied to obtain $\boldsymbol{\theta}_{ij}$ as follows:

$$\boldsymbol{\theta}_{ij} = \sum_{\{k,l\}\in p_{ij}|k<l} \boldsymbol{\theta}_{kl} = \mathbf{e}_{v_1} - \mathbf{e}_{v_2} + \mathbf{e}_{v_2} - \mathbf{e}_{v_3} + \ldots + \mathbf{e}_{v_\delta} - \mathbf{e}_{v_{\delta+1}} = \sum_{k=1}^{\delta} \mathbf{e}_{v_k} - \sum_{k=2}^{\delta+1} \mathbf{e}_{v_k} = \mathbf{e}_{v_1} + \sum_{k=2}^{\delta} \mathbf{e}_{v_k} - \sum_{k=2}^{\delta} \mathbf{e}_{v_k} - \mathbf{e}_{v_{\delta+1}} = \mathbf{e}_{v_1} - \mathbf{e}_{v_{\delta+1}} = \mathbf{e}_i - \mathbf{e}_j \tag{10}$$

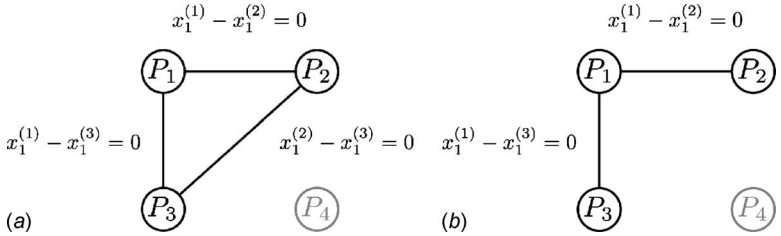

Fig. 6 Graph representation of consistency constraint options for x_1

If a path p_{ij} does not exist in G_c, then at least one edge $\{k,l\}$ in every possible set of constraint edges will be pendant, i.e., incident to a vertex of degree 1. If k is the pendant vertex, $\boldsymbol{\theta}_{kl}$ will contribute $\mathbf{e}_k$ to the constraint composition. Since only edge $\{k,l\}$ is adjacent to k, no constraint vector in the composition can annihilate $\mathbf{e}_k$. The case for l pendant is similar. Therefore, $\boldsymbol{\theta}_{ij}=\mathbf{e}_i-\mathbf{e}_j$ cannot be obtained if p_{ij} does not exist in G_c. ■

Extended constraint composition leads to a necessary condition for the equivalence of K_{n^z} and G_c. If a consistency constraint graph can be shown to be equivalent to K_{n^z}, its set of associated constraints will ensure complete consistency for the linking variable in consideration.

PROPOSITION 4.3. *A consistency constraint graph G_c is equivalent to K_{n^z} if and only if G_c is connected.*

Proof. If G_c is equivalent to K_{n^z}, G_c specifies either an explicit or an implicit edge for every constraint associated with K_{n^z}. Therefore, a path must exist between every pair of vertices, and G_c is connected. If G_c is connected, a path exists between every pair of vertices and a constraint exists between every pair of vertices in G_c, and the effective constraint sets and feasible domains of G_c and K_{n^z} are identical. ■

A consistency constraint graph is therefore minimal if it connects the required vertices using the fewest possible number of edges. By definition, a spanning tree uses the minimum number of edges (n^z-1) to ensure a graph is connected.

COROLLARY 4.4. *A consistency constraint graph is minimal if and only if it is a spanning tree of K_{n^z}.*

If G_c is connected and uses more than n^z-1 edges, then a cycle exists, and more than one path connects at least one pair of vertices. Such a graph is not minimal since at least one redundant constraint exists that could be removed. Since any consistency constraint can be composed through a composition of explicit constraints if G_c is connected, the set of explicit constraints corresponding to a minimally connected G_c can be viewed as a basis for the constraints in K_{n^z}. The constraint vectors in this set are, in fact, linearly independent, so indeed form a basis.

PROPOSITION 4.5. *The constraint vectors corresponding to explicit edges in G_c are linearly independent if and only if G_c is acyclic.*

Proof. If G_c is acyclic, at most one path exists between any pair of vertices. Therefore, if a constraint vector $\boldsymbol{\theta}_{ij}$ can be obtained, either $\boldsymbol{\theta}_{ij}$ is a row of $\boldsymbol{\Theta}$ and edge $\{i,j\}$ exists in G_c, or a unique path p_{ij} with length greater than 1 exists such that $\boldsymbol{\theta}_{ij}$ can be induced. If $\boldsymbol{\theta}_{ij}$ is a row of $\boldsymbol{\Theta}$, edge $\{i,j\}$ is the only path p_{ij}, and no composition of other constraints will yield $\boldsymbol{\theta}_{ij}$. Since this is true for all explicit constraints, the rows of $\boldsymbol{\Theta}$ are linearly independent. If G_c contains a cycle C, then two adjacent vertices on C (i and j) have at least two paths between them: the edge $\{i,j\}$ and $C\backslash\{i,j\}$. Therefore $\boldsymbol{\theta}_{ij}$ is an explicit constraint that can be obtained through composition of other explicit constraints, and the rows of $\boldsymbol{\Theta}$ are not linearly independent. ■

COROLLARY 4.6. *If G_c is minimal, it is an acyclic spanning tree and therefore has a linearly independent set of explicit consistency constraints.*

The independence properties of spanning trees are generalizable. If $\mathcal{I}$ is the set of all spanning trees of a graph G and their power sets, and E is the set of all edges of G, $(E,\mathcal{I})$ is the cycle matroid of G. The maximal sets in $\mathcal{I}$ are bases, and $\mathcal{I}$ coincides with the sets of linearly independent columns of the incidence matrix of G [15]. Another result of Proposition 4.5 is that the set of all constraint vectors on a linking variable and all linearly independent sets of these vectors form a vector matroid that corresponds to the cycle matroid of K_{n^z}. The favorable properties of binary consistency constraints enable not only the straightforward identification of valid constraint sets, but also open the door to increased understanding of consistency constraints due to their link to spanning trees and cycle matroids.

The foregoing propositions lead to the main result of this section.

PROPOSITION 4.7. *G_c is a valid consistency constraint graph if and only if G_c is a spanning tree of K_n.*

Proof. If G_c is valid, the rows of $\boldsymbol{\Theta}$ are linearly independent, and G_c is acyclic by Proposition 4.5. It also follows from the validity of G_c that consistency is assured; i.e., G_c is equivalent to K_n. By Proposition 4.3, G_c is connected, and therefore G_c is a spanning tree of K_n. Conversely, if G_c is a spanning tree of K_n, G_c is connected and acyclic. It follows from Propositions 4.3 and 4.5 that G_c ensures consistency and linear independence of constraints. Therefore, G_c is valid. ■

This result means that the set of consistency constraint allocation options for a linking variable z associated with n^z subproblems is defined by the set of all possible spanning trees for the complete graph K_{n^z}. These trees may be represented easily and algorithms exist for their enumeration. This makes practical the inclusion of linking structure options in the optimal partitioning and coordination decision problem for ALC. Linking structures for other formulations, such as CO or ATC, have additional restrictions not present for ALC, and their analysis is left as future work.

4.3 Example Consistency Constraint Graph. The consistency constraint graph for x_1 from the example system of Fig. 2 is used to demonstrate valid consistency constraint options and their graph representations. When the partition $\mathbf{p}=[1,2,2,3,3,4]$ is used, x_1 is shared between subproblems 1, 2, and 3. The three available consistency constraints are displayed in Fig. 6(*a*) alongside graph edges that represent these constraints. One possible valid consistency constraint graph is shown in Fig. 6(*b*). The vector of x_1 copies is:

$$\tilde{\mathbf{z}} = [x_1^{(1)}, x_1^{(2)}, x_1^{(3)}] \tag{11}$$

and the linearly independent consistency constraint matrix for x_1 that corresponds to the edge set $\{\langle 1,2\rangle,\langle 1,3\rangle\}$ shown in Fig. 6(*b*) is:

$$\boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{\theta}_{12} \\ \boldsymbol{\theta}_{13} \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 \\ 1 & 0 & -1 \end{bmatrix} \tag{12}$$

5 Optimal Partitioning and Coordination Decisions for Parallel ALC

Section 4 demonstrated that the set of consistency constraints used for a linking variable must connect associated subproblems using a tree structure to meet ALC convergence and system consistency requirements. Determining consistency constraint structure for every linking variable is an important coordination decision, and influences the computational expense and reliability of an ALC implementation. This section extends the optimal partitioning and coordination method presented in Ref. [2] by including linking structure decisions along with partitioning and sequencing decisions.

If υ_i is the number of subproblems linked by the ith external linking variable, then the number of valid options for allocating consistency constraints for this variable is the number of unique spanning trees for a graph with υ_i vertices, or $\upsilon_i^{\upsilon_i-2}$. If n_z is the number of external linking variables in a problem, then $\upsilon_1^{\upsilon_1-2}\cdot\upsilon_2^{\upsilon_2-2}\cdot\ldots\cdot\upsilon_{n_z-1}^{\upsilon_{n_z-1}-2}\cdot\upsilon_{n_z}^{\upsilon_{n_z}-2}$ is the number of alternative linking structure options for a problem with a given system partition. The number of linking structure alternatives in a problem can be reduced by exploiting the natural structure present in coupling variable relationships. An analysis function output that is a coupling variable may be communicated to one or more analysis functions. All analysis functions receiving this coupling variable as an input link directly to the analysis function that computes the coupling variable; this structure forms a star graph, which is a spanning tree. While it is possible to use other trees for coupling variable consistency constraints, we assume here that the naturally occurring star graph is the consistency constraint graph used for each coupling variable. This reduces the number of trees that must be determined to the number of shared design variables.

Two important factors contribute to overall ALC computational expense: coordination problem difficulty and subproblem difficulty. An intrinsic tradeoff exists between these two factors; fine partitions may have lower subproblem expense, but can incur higher coordination expense due to more complicated external linking relationships. A metric for optimization problem size is used here to estimate subproblem expense. Coordination expense is approximated using a metric based on the assumption that block parallel Gauss–Seidel converges faster when linking variables input to subproblems are recently computed. Jacobi iteration is one extreme possibility where all input data are from the previous iteration, whereas sequential Gauss–Seidel (FPI) uses the most recently available data. Gauss–Seidel iteration is known to have better global and local convergence properties than the Jacobi iteration for linear systems [8], but Jacobi iteration offers advantages for parallelism. These arguments do not always extend to nonlinear systems, but are assumed to be reasonable in enabling a priori partitioning and coordination decisions based on a system's reduced adjacency matrix.

Once a system partition is defined, the subproblem graph can be constructed that describes external linking variable relationships, along with its associated adjacency matrix. $\bar{A}$ is defined to be the $N\times N$ valued adjacency matrix for a partitioned system's subproblem graph, where each entry indicates the dimension of the corresponding linking variable. For example, if $\bar{A}_{ij}=3$, then the dimension of $\bar{\mathbf{z}}_{ij}$ is 3. The coordination expense is estimated here using CS, a metric for coordination problem size that accounts for sequencing aspects of a coordination strategy:

$$\mathrm{CS}=\sum_{i=1}^{N}\sum_{j=1}^{N}\zeta_{ij}\bar{A}_{ij} \tag{13}$$

The value of ζ_{ij} quantifies in how many stages previous to the evaluation of subproblem i the linking variables $\bar{\mathbf{z}}_{ij}$ were computed. CS quantifies the number of linking variables in the coordination problem, and also accounts for the length of time between linking variable calculation and its use as an input. The metric ζ_{ij} is defined as follows:

$$\zeta_{ij}=\begin{cases}s_i-s_j & \text{if } s_i>s_j\\ n^s+s_i-s_j & \text{if } s_i\le s_j\end{cases} \tag{14}$$

where $n^s=\max(\mathbf{s})$ is the stage depth (i.e., the number of stages in the implementation).

A usual estimate for subproblem expense is subproblem size. Previous approaches for quantifying subproblem size were based only on the number of analysis functions or equations in each subproblem (e.g., Ref. [16]). The metric used here is somewhat more sophisticated, being based on optimization problem size. The size of the optimization problem for subproblem i is defined as:

$$SS_i=(n_{\bar{x}_s i}+n_{x_\ell i}+n_{yi}+n_{\bar{y}Ii})+(n_{\bar{x}_s ci}+n_{yi}+n_{\bar{y}i})+(n_{ai}) \tag{15}$$

The first four terms comprise the number of decision variables in subproblem i. The number of external shared variables associated with subproblem i is $n_{\bar{x}_s i}$, the number of local variables is $n_{x_\ell i}$, the number of internal coupling variables is n_{yi}, and the number of external input coupling variables is $n_{\bar{y}Ii}$. The next three terms express the number of consistency constraints in subproblem i. The number of consistency constraints for external shared variables is $n_{\bar{x}_s ci}$, the number of internal coupling variable consistency constraints is equal to n_{yi}, and the number of consistency constraints for external coupling variables is equal to $n_{\bar{y}i}$. The last term is the number of analysis functions (n_{ai}). The maximum subproblem size for each stage is computed, and $\overline{SS}_{\max}$ is the average of the maximum subproblem sizes.

The optimal P/C decision problem for parallel ALC with linking structure considerations is to minimize simultaneously CS and $\overline{SS}_{\max}$ by selecting a system partition $\mathbf{p}$, subproblem stage assignment $\mathbf{s}$, and a valid consistency constraint graph for each external shared design variable. The length of the vector $\mathbf{s}$ is N, which depends on $\mathbf{p}$. This complication is handled easily when the optimal P/C decision problem is solved with exhaustive enumeration. The linking structure decisions depend also on $\mathbf{p}$. System partition changes the set of external shared design variables, and the subproblems associated with each external shared design variable. As with stage assignment, linking structure can be handled with exhaustive enumeration. An evolutionary algorithm for making partitioning and coordination decisions was introduced in Ref. [17], and can handle this type of decision variable dependence.

A set-valued decision variable $\mathcal{C}$ is defined for the purpose of representing problem linking structure. The cardinality of $\mathcal{C}$ is equal to the number of external shared design variables in a problem with a given partition. Each member of this set defines the consistency constraint graph for one of the shared variables. One approach to representing a consistency constraint graph, which must be a spanning tree, is with an edge set. For example, the variable x_1 in Fig. 3 is shared between P_1, P_2, and P_3, but the constraints on x_1 appear only in $\bar{\mathbf{c}}_{12}$ and $\bar{\mathbf{c}}_{13}$, which are the consistency constraints connecting P_1 with P_2 and P_1 with P_3, respectively. The edge set corresponding to these constraints for x_1 is $\{\langle 1,2\rangle,\langle 1,3\rangle\}$. By convention, edges are represented using ordered pairs $\langle i,j\rangle$ such that $i<j$. This way each edge has only one representation.

Now that we have defined the two objective functions and the P/C decision variables, we can state formally the optimal P/C problem:

$$\min_{\mathbf{p},\mathbf{s},\mathcal{C}}\{\mathrm{CS},\overline{SS}_{\max}\} \tag{16}$$

The solution to this problem is a set of Pareto-optimal P/C decision alternatives. This Pareto set helps assess the intrinsic

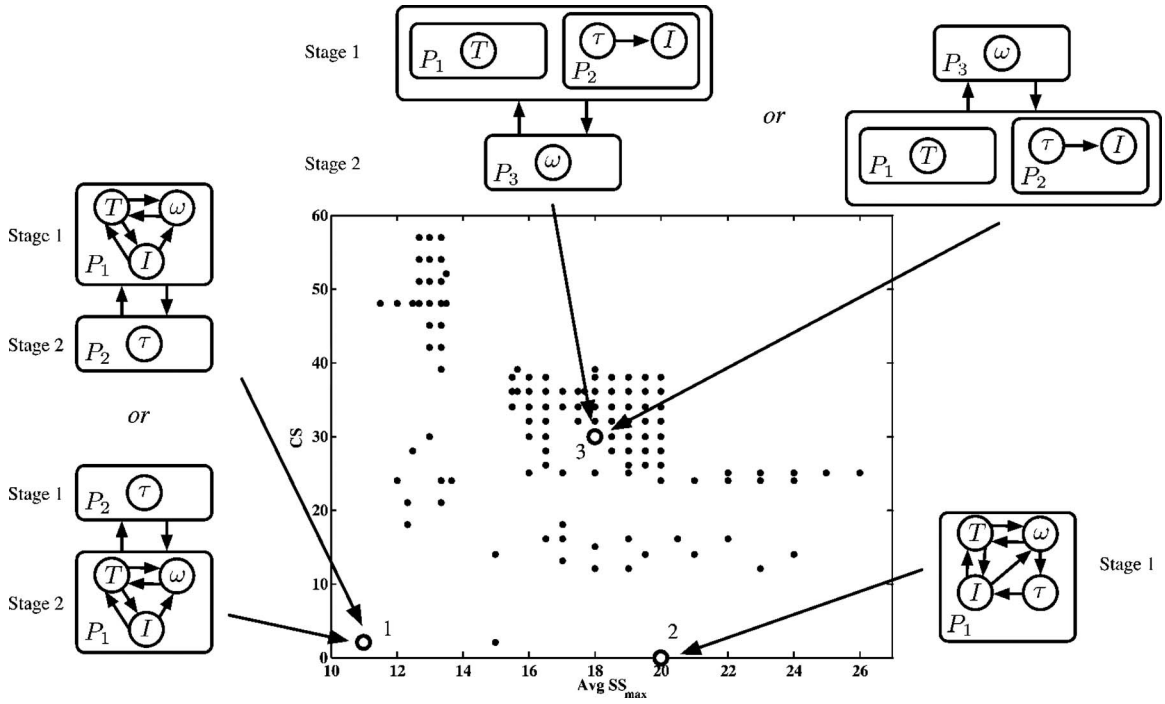

Fig. 7 ALC P/C results for electric water pump problem

tradeoffs in the optimal P/C problem. Note that specifying **p**, **s**, and $\mathcal{C}$ defines completely a parallel ALC partition, coordination algorithm, and set of subproblem formulations.

6 Example: Electric Water Pump Design Problem

The partitioning and coordination decision method for ALC described above was applied to the electric water pump design problem introduced in Ref. [2] and detailed in Ref. [18]. This example system is relatively small (large, nonproprietary models are difficult to obtain), but illustrates adeptly the proposed technique for P/C decision-making with linking structure considerations. This design problem involves a centrifugal pump for an automotive cooling system driven by an electric motor. The design objective is to reduce electric power consumption, subject to performance, thermal, and geometric constraints. The optimal pump consumes 140 W during operation, compared with 300 W consumed by a traditional belt-driven water pump. The reduced adjacency matrix for the problem is the only information needed to solve the problem in Eq. (16) above.

$$\mathbf{A} = \begin{array}{c|cccccccccccc} & a_1 & a_2 & a_3 & a_4 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} \\ a_1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ a_2 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_3 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ a_4 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{array} \tag{17}$$

The analysis functions a_{1-4} evaluate motor temperature T, motor current I, motor speed ω, and pump drive torque τ, respectively. Design variables x_{1-5} describe motor geometry, and x_{6-10} describe pump geometry. Using exhaustive enumeration of all **p**, **s**, and $\mathcal{C}$ combinations for this problem, 9295 unique partitioning and coordination alternatives were identified, and two Pareto-optimal points were found. All instances are displayed in the CS$-\overline{SS}_{\max}$ space in Fig. 7, and all partitioning and stage assignment options that correspond to three of these points are shown.

Point 1. Two P/C decision instances correspond to point 1 in Fig. 7, and all share the same partition and problem size metrics: CS=2, $\overline{SS}_{\max}=11$, and $\mathbf{p}=[1,1,1,2]$. Neither instance has any shared design variables, but can be distinguished by subproblem stage assignment: Instance 1: $\mathbf{s}=[1,2]$ and Instance 2: $\mathbf{s}=[2,1]$. These instances reflect a physical partition between the functions that depend on the motor design variables, and the function that depends on the pump design variables. This is an intuitive result, but partitioning by physical system may not always be preferred. In this case, there are no shared variables between the physical subsystems, so there is an obvious advantage to this partition. This may not always be true: Physical subsystems may have interfaces that result in shared design variables, complicating P/C decisions.

Point 2. The single subproblem case, where CS=0 and $\overline{SS}_{\max}=20$, is represented by point 2. Note that numerous P/C instances exist with larger subproblem sizes and nonzero coordination problem sizes. These points represent especially poor options for constructing an ALC formulation of the electric water pump problem. Moving from point 2 to point 1 reduces $\overline{SS}_{\max}$ from 20 to 11, and requires a coordination problem size of just 2. This indicates a problem formulation that is a good candidate for decomposition-based optimization because dividing the system into two subproblems stands to reduce subproblem computational expense substantially, while incurring only nominal coordination expense.

Point 3. A third point, not in the Pareto set, is examined for illustrative purposes. Point 3 corresponds to 12 unique P/C instances, all with the same partition and problem size metrics: CS=30, $\overline{SS}_{\max}=18$, and $\mathbf{p}=[1,2,3,2]$. All 12 instances have the same set of external shared design variables: $\{x_1,x_2,x_3,x_4,x_5\}$. The first four are shared between three subproblems, so several consistency constraint allocation options exist. One possible set of valid consistency constraint graphs, with corresponding edge sets, is shown in Fig. 8.

The 12 instances that correspond to point 3 are distinguished by consistency constraint allocation and stage assignment. The two stage assignments that appear here are Instances 1–6: $\mathbf{s}=[1,1,2]$ and Instances 7–12: $\mathbf{s}=[2,2,1]$. These stage assignments are illustrated in Fig. 7, and both specify parallel solution of subproblems 1 and 2. No Pareto-optimal points specify parallel subproblem solution. This is due to both problem structure and the

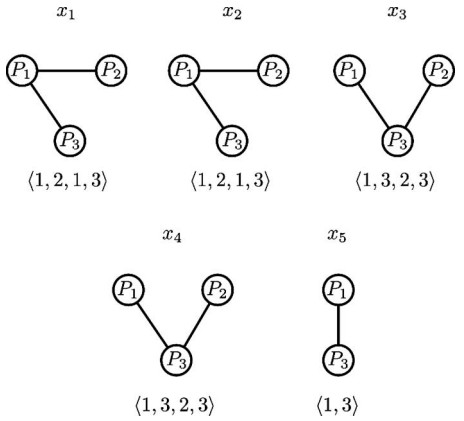

Fig. 8 Consistency constraint allocation option for point 3

problem size metrics selected. At point 1 we see that had we placed both subproblems in Stage 1 for a parallel solution, the time between calculation and use of some linking variables would have actually increased, increasing CS. Only CS penalizes stage depth (i.e., the number of stages in a parallel implementation). Other size metrics have been explored, such as the sum of all maximum subproblem sizes for each stage ($\Sigma SS_{\max}$). This metric penalizes stage depth, and when employed along with CS, the resulting Pareto set contains only single-stage P/C alternatives. An ideal metric would be an accurate estimate of computational expense, including speedup for parallelism. Since this is impractical to compute a priori for most problems, approximate metrics must be used.

7 Conclusion

This work established an approach for constructing problem formulations for decomposition-based design optimization, and a possible set of metrics was proposed (i.e., CS and $\overline{SS}_{\max}$). These metrics approximate two competing sources of computational expense: coordination problem and subproblem solution expenses. A new formulation technique for parallel ALC implementations was introduced, and used to study linking structure decisions. ALC linking structure is defined by the way consistency constraints on linking variables are allocated throughout a system design problem. Graph theory and constraint satisfaction techniques were used to identify valid consistency constraint allocation options for ALC. This development enables inclusion of linking structure decisions with the optimal partitioning and coordination decision problem for ALC; it extends previous P/C decision methods, which accounted only for partitioning and sequencing decisions; and it may help system designers take full advantage of ALC linking structure flexibility to tailor solution methods to system structure. Several open questions have been identified presenting opportunities for future work, including convergence analysis, handling of larger size problems, investigation of alternative problem size metrics, inclusion of coupling variable consistency constraint allocation in the coordination decision problem, analysis of linking structure for other system optimization formulations (such as ALC with linking functions), and demonstration on problems of sufficient complexity that would defy the designer's intuition.

Acknowledgment

This work was partially supported by a U.S. NSF Graduate Research Fellowship and by the Automotive Research Center, a U.S. Army Center of Excellence at the University of Michigan. This support is gratefully acknowledged. The work of J.T.A. was conducted while at the University of Michigan.

References

[1] Stanton, D., and White, D., 1986, *Constructive Combinatorics*, Springer-Verlag, New York.
[2] Allison, J. T., Kokkolaras, M., and Papalambros, P. Y., 2009, "Optimal Partitioning and Coordination Decisions in Decomposition-Based Design Optimization," ASME J. Mech. Des., **131**(8), p. 081008.
[3] Braun, R. D., 1996, "Collaborative Optimization: An Architecture for Large-Scale Distributed Design," Ph.D. thesis, Stanford University, Stanford, CA.
[4] Kim, H. M., Michelena, N. F., Papalambros, P. Y., and Jiang, T., 2003, "Target Cascading in Optimal System Design," ASME J. Mech. Des., **125**(3), pp. 474–480.
[5] Tosserams, S., Etman, L. F. P., and Rooda, J. E., 2007, "An Augmented Lagrangian Decomposition Method for Quasiseparable Problems in MDO," Struct. Multidiscip. Optim., **34**(3), pp. 211–227.
[6] Tosserams, S., Etman, L. F. P., and Rooda, J. E., 2008, "Augmented Lagrangian Coordination for Distributed Optimal Design in MDO," Int. J. Numer. Methods Eng., **73**(13), pp. 1885–1910.
[7] Bertsekas, D. P., 1999, *Nonlinear Programming*, 2nd ed., Athena Scientific, Belmont, MA.
[8] Bertsekas, D. P., and Tsitsiklis, J. N., 1997, *Parallel and Distributed Computation: Numerical Methods*, Athena Scientific, Belmont, MA.
[9] Montanari, U., 1974, "Networks of Constraints: Fundamental Properties and Applications to Picture Processing," Inf. Sci. (N.Y.), **7**, pp. 95–132.
[10] Tsang, E., 1993, *Foundations of Constraint Satisfaction*, Academic, San Diego, CA.
[11] Kusiak, A., Wang, J., and He, D. W., 1996, "Negotiation in Constraint-Based Design," ASME J. Mech. Des., **118**, pp. 470–477.
[12] Schmidt, L. C., Shi, H., and Kerkar, S., 2005, "A Constraint Satisfaction Problem Approach Linking Function and Grammar-Based Design Generation to Assembly," ASME J. Mech. Des., **127**, pp. 196–205.
[13] Hicks, B. J., Medland, A. J., and Mullineux, G., 2006, "The Representation and Handling of Constraints for the Design, Analysis, and Optimization of High Speed Machinery," Artif. Intell. Eng. Des. Anal. Manuf., **20**, pp. 131–328.
[14] Mackworth, A. K., 1977, "Consistency in Networks of Relations," Artif. Intell., **8**(1), pp. 99–118.
[15] Oxley, J., 2003, "What Is a Matroid?," Cubo Matemática Educacional, **5**(3), pp. 179–218.
[16] Michelena, N. F., and Papalambros, P. Y., 1997, "A Hypergraph Framework for Optimal Model-Based Decomposition of Design Problems," Comput. Optim. Appl., **8**(2), pp. 173–196.
[17] Allison, J. T., and Papalambros, P. Y., 2007, "Optimal Partitioning and Coordination Decisions in System Design Using an Evolutionary Algorithm," Proceedings of the Seventh World Conference on Structural and Multidisciplinary Optimization, Seoul, South Korea, May 21–25.
[18] Allison, J. T., 2008, "Optimal Partitioning and Coordination Decisions in Decomposition-Based Design Optimization," Ph.D. thesis, University of Michigan, Ann Arbor, MI.