Proceedings of IDETC/CIE 2007
Proceedings of the ASME 2007 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
September 4-7, 2007, Las Vegas, Nevada, USA

IDETC2007-35566

DIAGONAL QUADRATIC APPROXIMATION FOR PARALLELIZATION OF ANALYTICAL TARGET CASCADING

Yanjing Li
Graduate Student
Department of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305
Email: yanjingl@stanford.edu

Zhaosong Lu
Assistant Professor
Department of Mathematics
Simon Fraser University
Burnaby, British Columbia V5A 1S6
Canada
Email: zhaosong@sfu.ca

Jeremy J. Michalek
Assistant Professor
Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: jmichalek@cmu.edu

ABSTRACT

Analytical Target Cascading (ATC) is an effective decomposition approach used for engineering design optimization problems that have hierarchical structures. With ATC, the overall system is split into subsystems, which are solved separately and coordinated via target/response consistency constraints. As parallel computing becomes more common, it is desirable to have separable subproblems in ATC so that each subproblem can be solved concurrently to increase computational throughput. In this paper, we first examine existing ATC methods, providing an alternative to existing nested coordination schemes by using the block coordinate descent method (BCD). Then we apply diagonal quadratic approximation (DQA) by linearizing the cross term of the augmented Lagrangian function to create separable subproblems. Local and global convergence proofs are described for this method. To further reduce overall computational cost, we introduce the truncated DQA (TDQA) method that limits the number of inner loop iterations of DQA. These two new methods are empirically compared to existing methods using test problems from the literature. Results show that computational cost of nested loop methods is reduced by using BCD and generally the computational cost of the truncated methods, TDQA and ALAD, are superior to other nested loop methods with lower overall computational cost than the best previously reported results.

INTRODUCTION

When designing complex systems, generally it is not possible or desirable to have a single decision-maker in charge of all decisions because of the need to manage problem complexity. Instead, such systems are routinely decomposed hierarchically into subsystems and components, and various design groups interact to coordinate their decisions and achieve a feasible and consistent system solution. For each system in such a hierarchy, target specifications are chosen for the subsystems below such that the system can meet targets set by the supersystem above. If targets cannot be met, negotiation and rebalancing is necessary to ensure that the final system solution is consistent and achieves system goals. Ford Motor Company refers to this process as target cascading, and the analogous model-based, computational process for such hierarchical systems has been termed analytical target cascading (ATC) [7]. In ATC, top level design targets are propagated to lower levels, which are optimized to meet the targets. The resulting responses are rebalanced at higher levels to achieve consistency. The optimal system solution is obtained through an iterative process until target/response consistency is achieved globally.

ATC approaches this target-setting and matching process through formal mathematical decomposition methods, and so it has similarities to many of the multidisciplinary design optimization (MDO) methods that have been developed to coordinate complex analysis models from various disciplines during optimization, such as collaborative optimization (CO), concurrent subspace optimization (CSSO), and bi-level integrated system synthesis (BLISS). In particular, Allison *et al.* [1] compare and contrast ATC and CO. Apart from the difference in initial motivation, the formulation of ATC also differs in that it is defined for an arbitrarily large hierarchy of subsystems, and formal convergence proofs ensure that the method will reach an optimal system solution under typical assumptions. More recently, methods for solving non-hierarchical quasiseparable, or dual block-angular, problems with proven convergence properties have also emerged [5] [6] [21]. In this paper, we will focus on hierarchical problems in the spirit of ATC; however, as we will discuss, ATC works by translating a description of a hierarchy of systems and subsystems linked by target-response interactions into a general quasiseparable problem[1], through relaxation of target-response relationships between systems and their subsystems. The methods we pose for solving hierarchical ATC systems could also be

[1]A quasiseparable problem is nearly separable except for a few linking variables that appear in multiple subsystems, as we will define rigorously later.

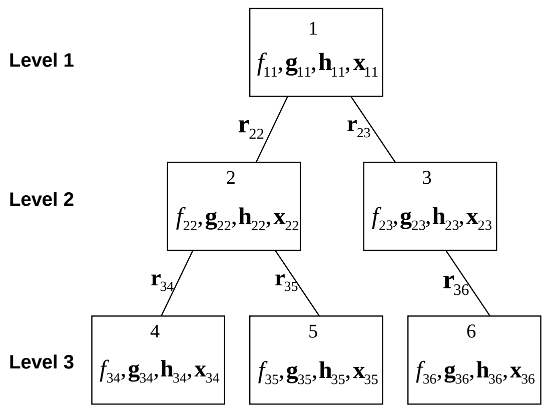
Level 1
Level 2
Level 3
1
$f_{11},\mathbf{g}_{11},\mathbf{h}_{11},\mathbf{x}_{11}$
$\mathbf{r}_{22}$
$\mathbf{r}_{23}$
2
$f_{22},\mathbf{g}_{22},\mathbf{h}_{22},\mathbf{x}_{22}$
3
$f_{23},\mathbf{g}_{23},\mathbf{h}_{23},\mathbf{x}_{23}$
$\mathbf{r}_{34}$
$\mathbf{r}_{35}$
$\mathbf{r}_{36}$
4
$f_{34},\mathbf{g}_{34},\mathbf{h}_{34},\mathbf{x}_{34}$
5
$f_{35},\mathbf{g}_{35},\mathbf{h}_{35},\mathbf{x}_{35}$
6
$f_{36},\mathbf{g}_{36},\mathbf{h}_{36},\mathbf{x}_{36}$

Figure 1. HIERARCHICAL PROBLEM STRUCTURE AND VARIABLE ALLOCATION FOR ATC

used to approach general quasiseparable systems. ATC has been applied to complex systems such as automotive design [8], architectural design [4], and multidisciplinary product development [12] [13], and these studies demonstrate scalability of ATC on large and computationally onerous case studies.

In this paper, we summarize existing methods for coordination of subsystems in ATC and present two new methods that enable parallelization and overcome some of the drawbacks of existing methods. We start by providing an overview of ATC in section 2. In section 3 we describe existing coordination methods, and in section 4 we describe two new methods, including a discussion of convergence properties. In section 5 we compare approaches empirically using test problems from the literature, and we discuss results in section 6 and conclude in section 7.

OVERVIEW OF ATC

System Structure

ATC is applicable for problems that have a hierarchical structure so that the top level design is a supersystem that consists of a number of systems, each of which may consist of its own subsystems. For example, an automobile may be composed of powertrain, body, and chassis, and the powertrain may be composed of engine and transmission, etc. This model is general enough to account for any number of levels in the hierarchy [16]. The objective function for the overall system can be described as a sum of the objective functions of its components; Typically the objective function is entirely at the system level. Moreover, the subproblems are nearly separable except for a few linking variables. Figure 1 shows an example. Specifically, a parent and a child are connected by a response variable, which represents a child's response to the design specification that its parent imposes. This "response variable" may or may not be a variable in the original pre-decomposition formulation: It is typically the output of the subsystem simulator and input to the system simulator and it may be treated as an intermediate calculation of the original formulation. The effects of subsystem response on system behavior is what prevents subsystems from being designed independently.

Notation

Different notations are used in describing and defining ATC, depending on the application [7] [15] [16] [20]. In this paper, we adopt the notational system of Tosserams *et al.* [20] for simplicity. Consider a system that can be decomposed into N levels and M elements. The subscript ij is used to denote the jth element of the system in the ith level. f_{ij} is the scalar objective function, and $\mathbf{g}_{ij} \leq \mathbf{0}$ and $\mathbf{h}_{ij} = \mathbf{0}$ are the inequality and equality constraints, respectively. Local variables of element j are denoted by $\mathbf{x}_{ij}$. $\mathbf{r}_{ij}$ is the response of element j calculated by analysis model $\mathbf{a}_{ij}$. Generally speaking, this model is an engineering simulation or a set of equations predicting the behavior of the subsystem. $\mathcal{E}_i$ is the set of elements at level i, and $\mathcal{C}_{ij}$ is the set of children of element j. The system in Figure 1 is shown corresponding to this notation.

Mathematical Formulation

By the assumption of the problem structure, and using the notation described above, the hierarchical problem before decomposition, also known as the all-in-one (*AIO*) formulation, can be described as:

$$\begin{aligned}
\min_{\mathbf{x}_{ij},\forall j\in\mathcal{E}_i,\, i=1,\ldots,N} & f(\mathbf{x}_{ij},\, \forall j \in \mathcal{E}_i,\, i=1,\ldots,N) = \sum_{i=1}^{N}\sum_{j\in\mathcal{E}_i} f_{ij}(\overline{\mathbf{x}}_{ij}) \\
\text{s.t.} \quad & \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) \leq \mathbf{0} \\
& \mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) = \mathbf{0} \\
\text{where} \quad & \overline{\mathbf{x}}_{ij} = [\mathbf{x}_{ij}, \mathbf{r}_{(i+1)k}\, \forall k \in \mathcal{C}_{ij}] \\
& \mathbf{r}_{ij} = \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\
& \forall j \in \mathcal{E}_i,\, i=1,\ldots,N.
\end{aligned} \tag{1}$$

Note that the response $\mathbf{r}_{ij}$ of each element j depends on the response of its children which prevents the objective function and the constraint sets from being separable. In order to separate the set of variables governed by each subsystem, target variables $\mathbf{t}_{ij}$ are created for each shared variable. And the *consistency constraint*:

$$\mathbf{t}_{ij} - \mathbf{r}_{ij} = \mathbf{0} \tag{2}$$

is added to ensure target/response consistency.

We rewrite the problem as:

$$\begin{aligned}
\min_{\overline{\mathbf{x}}_{11},\ldots,\overline{\mathbf{x}}_{NM}} & \sum_{i=1}^{N}\sum_{j\in\mathcal{E}_i} f_{ij}(\overline{\mathbf{x}}_{ij}) \\
\text{s.t.} \quad & \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) \leq \mathbf{0} \\
& \mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) = \mathbf{0} \\
& \mathbf{t}_{ij} - \mathbf{r}_{ij} = \mathbf{0} \\
\text{where} \quad & \overline{\mathbf{x}}_{ij} = [\mathbf{x}_{ij}, \mathbf{t}_{(i+1)k}\, \forall k \in \mathcal{C}_{ij}] \\
& \mathbf{r}_{ij} = \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\
& \forall j \in \mathcal{E}_i,\, i=1,\ldots,N.
\end{aligned} \tag{3}$$

Figure 2 shows the system structure and variable allocations after introducing the target variables. Note that the problem is almost separable except for the consistency constraint $\mathbf{t}_{ij} - \mathbf{r}_{ij} = \mathbf{0}$. In order to make the constraint sets separable, the consistency constraint can be relaxed using penalty functions or Lagrangian relaxation. In general, the problem can be relaxed via a *consistency constraint relaxation function* π. Alternate methods for consistency constraint relaxation are discussed in section 3. For a

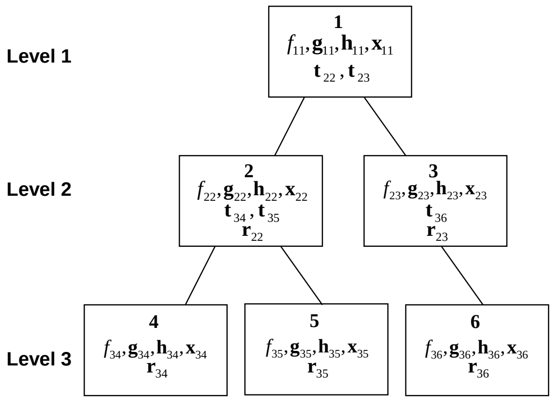
Level 1
1
$f_{11}, \mathbf{g}_{11}, \mathbf{h}_{11}, \mathbf{x}_{11}$
$\mathbf{t}_{22}, \mathbf{t}_{23}$
Level 2
2
$f_{22}, \mathbf{g}_{22}, \mathbf{h}_{22}, \mathbf{x}_{22}$
$\mathbf{t}_{34}, \mathbf{t}_{35}$
$\mathbf{r}_{22}$
3
$f_{23}, \mathbf{g}_{23}, \mathbf{h}_{23}, \mathbf{x}_{23}$
$\mathbf{t}_{36}$
$\mathbf{r}_{23}$
Level 3
4
$f_{34}, \mathbf{g}_{34}, \mathbf{h}_{34}, \mathbf{x}_{34}$
$\mathbf{r}_{34}$
5
$f_{35}, \mathbf{g}_{35}, \mathbf{h}_{35}, \mathbf{x}_{35}$
$\mathbf{r}_{35}$
6
$f_{36}, \mathbf{g}_{36}, \mathbf{h}_{36}, \mathbf{x}_{36}$
$\mathbf{r}_{36}$

Figure 2. VARIABLE ALLOCATION FOR ATC AFTER INTRODUCING TARGET COPIES

general π, the resulting formulation is the *relaxed AIO problem*:

$$\begin{aligned}
\min_{\overline{\mathbf{x}}_{11},\ldots,\overline{\mathbf{x}}_{NM}} & \sum_{i=1}^{N}\sum_{j\in\mathscr{E}_i} f_{ij}(\overline{\mathbf{x}}_{ij}) + \sum_{i=2}^{N}\sum_{j\in\mathscr{E}_i} \pi(\mathbf{t}_{ij}-\mathbf{r}_{ij}) \\
\text{s.t.} \quad & \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) \le \mathbf{0} \\
& \mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) = \mathbf{0} \\
\text{where} \quad & \overline{\mathbf{x}}_{ij} = [\mathbf{x}_{ij}, \mathbf{t}_{(i+1)k} \forall k \in \mathscr{C}_{ij}] \\
& \mathbf{r}_{ij} = \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\
& \forall j \in \mathscr{E}_i, i = 1, \ldots, N. \qquad (4)
\end{aligned}$$

For a general π, consider only the subset of the decision variables that are non-constant in subsystem j to obtain the general subproblem corresponding to each element:

$$\begin{aligned}
\min_{\overline{\mathbf{x}}_{ij}} \ & f_{ij}(\overline{\mathbf{x}}_{ij}) + \pi(\mathbf{t}_{ij}-\mathbf{r}_{ij}) + \sum_{k\in\mathscr{C}_{ij}} \pi(\mathbf{t}_{(i+1)k}-\mathbf{r}_{(i+1)k}) \\
\text{s.t.} \quad & \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) \le \mathbf{0} \\
& \mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) = \mathbf{0} \\
\text{where} \quad & \overline{\mathbf{x}}_{ij} = [\mathbf{x}_{ij}, \mathbf{t}_{(i+1)k} \forall k \in \mathscr{C}_{ij}] \\
& \mathbf{r}_{ij} = \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\
& \forall j \in \mathscr{E}_i, i = 1, \ldots, N. \qquad (5)
\end{aligned}$$

Note that in the above formulation, the variables $\mathbf{t}_{ij}$ and $\mathbf{r}_{(i+1)k}$ for $k \in \mathscr{C}_{ij}$ are constants with respect to element j. The constraint sets are now separable, and depending on the consistency relaxation function π, the subproblems may or may not be separable. If subproblems are separable, they can be solved in parallel. Otherwise, sequential computation of each subproblem is required.

It can be shown that by sequentially and iteratively solving each subproblem as specified in (5) in any cyclic order, convergence is guaranteed. This algorithm is called the *block coordinate descent*, or BCD, and the convergence result is applicable for any general relaxation consistency function π because the constraint sets are independent. The following theorem summarizes the convergence result of block coordinate descent and the proof is given in Proposition 2.7.1 of [2], p. 268.

Theorem 1. *Suppose that for all i and j, f_{ij} is continuously differentiable and the constraint sets are independent, closed and convex. Let $\overline{\mathbf{x}}_{ij}^{\kappa}$ denote the solution of (5) in the κth iteration with $\mathbf{t}_{ij} = \mathbf{t}_{ij}^{\kappa}$ and $\mathbf{r}_{(i+1)k} = \mathbf{r}_{(i+1)k}^{\kappa-1}$ for $k \in \mathscr{C}_{ij}$. Furthermore, suppose that $\overline{\mathbf{x}}_{ij}^{\kappa}$ is uniquely attained for all iterations. Then $\{\overline{\mathbf{x}}_{ij}^{\kappa}\}$ is the sequence generated by the block coordinate descent method, and every limit point of $\{\overline{\mathbf{x}}_{ij}^{\kappa}\}$ is a stationary point.*

In the ATC literature, non-separable subproblems are handled in nested loop coordination schemes that are carried out in a bi-level fashion, and the original convergence proof for ATC [16] is based on the nested schemes. However, by the theorem above, the nested schemes are not necessary, and the block coordinate descent method alone can produce convergence for any sequence of subproblem solutions.

The inconsistency constraint relaxation function π has been implemented in three ways: a quadratic penalty function [7] [14] [16], an ordinary Lagrangian function [10], or an augmented Lagrangian function [9] [20]. Both the quadratic penalty and augmented Lagrangian approaches do not allow separability of subproblems, and block coordinate descent is required to achieve convergence, which limits efficiency. The ordinary Lagrangian approach does produce separable subproblems. However, the method is not robust when duality gaps exist [19]. These methods will be discussed in more detail in the following sections.

PRIOR CONSISTENCY CONSTRAINT RELAXATION METHODS FOR ATC

The Quadratic Penalty Method

In early ATC literature, a quadratic penalty term was used as the constraint relaxation function π. The basic idea in penalty methods is to eliminate the consistency constraint and add to the objective function a penalty term that prescribes a high cost to infeasible points. Associated with this term are the penalty parameters $\mathbf{w} = [\mathbf{w}_{ij}, \forall i, j]$ that determine the severity of the penalty and, as a consequence, the extent to which the resulting unconstrained problem approximates the original constrained problem.

The general quadratic penalty function is defined as:

$$\pi_{\mathbf{Q}}(\mathbf{t}_{ij}-\mathbf{r}_{ij}) = ||\mathbf{w}_{ij} \circ (\mathbf{t}_{ij}-\mathbf{r}_{ij})||_2^2 \qquad (6)$$

where $\circ$ is used to denote the Hadamard product, so that $(\mathbf{A} \circ \mathbf{B})_i = \mathbf{A}_i \mathbf{B}_i$. With π_Q, (4) can be rewritten as:

$$\begin{aligned}
\min_{\overline{\mathbf{x}}_{11},\ldots,\overline{\mathbf{x}}_{NM}} & \sum_{i=1}^{N}\sum_{j\in\mathscr{E}_i} f_{ij}(\overline{\mathbf{x}}_{ij}) + \sum_{i=2}^{N}\sum_{j\in\mathscr{E}_i} ||\mathbf{w}_{ij} \circ (\mathbf{t}_{ij}-\mathbf{r}_{ij})||_2^2 \\
\text{s.t.} \quad & \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) \le \mathbf{0} \\
& \mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) = \mathbf{0} \\
\text{where} \quad & \overline{\mathbf{x}}_{ij} = [\mathbf{x}_{ij}, \mathbf{t}_{(i+1)k} \forall k \in \mathscr{C}_{ij}] \\
& \mathbf{r}_{ij} = \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\
& \forall j \in \mathscr{E}_i, i = 1, \ldots, N. \qquad (7)
\end{aligned}$$

Proper selection of weighting coefficients heavily impacts the performance of this method. As $\mathbf{w}_{ij}$ becomes larger, the resulting inconsistency between $\mathbf{t}_{ij}$ and $\mathbf{r}_{ij}$ pairs decreases. Convergence properties of the quadratic penalty method have been studied thoroughly and can be summarized in the following theorem.

Theorem 2. *Assume that the functions f_{ij}, g_{ij}, and h_{ij} for all i, j are continuous. Suppose that $\{\overline{\mathbf{x}}_{ij}^{\kappa}\}$ for all i and j is an exact global minimizer of (7) for each fixed $\mathbf{w}^{\kappa}$ and that $\mathbf{w}^{\kappa} \uparrow \infty$. Then every limit point of $\{\overline{\mathbf{x}}^{\kappa}\}$ is a solution of the problem (3).*

Its proof is similar to that of Theorem 17.1 of [17], p. 496. Indeed, by letting $X = \{\overline{x} \in \Re^n : \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) \leq \mathbf{0},\ \mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) = \mathbf{0}, \forall j \in \mathscr{E}_i, i = 1, ..., N.\}$, problem (3) can be generalized to the problem

$$\begin{aligned} &\min_x f(x) \\ s.t.\ &c_i(x) = 0,\ i \in \mathcal{E}, \\ &x \in X \end{aligned}$$

Under the assumption that $\mathbf{g}_{ij}$ and $\mathbf{h}_{ij}$ for all i, j are continuous on $\Re^n$, the set X is closed. Thus, replacing $\Re^n$ by X in the proof of Theorem 17.1 of [17] yeilds a valid proof for Theorem 2. Theorem 2 is also a special case of Proposition 4.2.1 of [2], p. 391 with $\lambda^{\kappa} = 0$ for all κ.

For each subproblem, we have:

$$\begin{aligned} \min_{\overline{\mathbf{x}}_{ij}} \ & f_{ij}(\overline{\mathbf{x}}_{ij}) + ||\mathbf{w}_{ij} \circ (\mathbf{t}_{ij} - \mathbf{r}_{ij})||_2^2 \\ & + \sum_{k \in \mathscr{C}_{ij}} ||\mathbf{w}_{(i+1)k} \circ (\mathbf{t}_{(i+1)k} - \mathbf{r}_{(i+1)k})||_2^2 \\ \text{s.t.} \ & \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) \leq \mathbf{0} \\ & \mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) = \mathbf{0} \\ \text{where} \ & \overline{\mathbf{x}}_{ij} = [\mathbf{x}_{ij}, \mathbf{t}_{(i+1)k} \forall k \in \mathscr{C}_{ij}] \\ & \mathbf{r}_{ij} = \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\ & \forall j \in \mathscr{E}_i, i = 1, ..., N. \end{aligned} \tag{8}$$

With this formulation, constraint sets are separable, but the objective function is not. Subproblems are solved sequentially throughout the hierarchical chain for fixed **w** as an inner loop. As noted before, either the nested coordination scheme or the block coordinate descent method (BCD) can be used as the inner loop. Upon convergence of all inner loops, an outer loop is used to update **w**. Figure 8 shows the procedure using nested and BCD schemes. Prior approaches to update **w** are either setting the penalty weights through trial and error, or initializing them to a small value and then increasing their value by a linear update method, i.e., multiplying the current weights by a constant. However, the trial and error approach can be difficult for large-scale problems. Michalek *et al.* [14] observed that while any positive **w** can ensure convergence to a consistent solution for problems that have attainable targets, no finite **w** will lead to perfect consistency for problems with unattainable targets. They proposed an efficient weight update method (WUM) for finding values of **w** that achieve solutions within user-specified inconsistency tolerances. It is demonstrated that WUM achieves better performance than a constant **w**.

In practice, large penalty weights can cause ill-conditioning [2]. Moreover, the quadratic term prevents each subproblem from being separable, preventing the use of parallel computing.

The Ordinary Lagrangian Method

An alternative choice for the constraint relaxation function π is an ordinary Lagrangian function [10]. This method (OL) is based on Lagrangian duality theory [2] [17]. Let $\lambda = [\lambda_{ij}, \forall i, j]$ be the vector of Lagrangian multipliers. The consistency constraint relaxation function is specified as:

$$\pi_L(\mathbf{t}_{ij} - \mathbf{r}_{ij}) = \lambda_{ij}^T(\mathbf{t}_{ij} - \mathbf{r}_{ij}) \tag{9}$$

With π_L, (4) can be rewritten as:

$$\begin{aligned} \Lambda(\lambda) = \min_{\overline{\mathbf{x}}_{11}, ..., \overline{\mathbf{x}}_{NM}} & \sum_{i=1}^{N} \sum_{j \in \mathscr{E}_i} f_{ij}(\overline{\mathbf{x}}_{ij}) + \sum_{i=2}^{N} \sum_{j \in \mathscr{E}_i} \lambda_{ij}^T(\mathbf{t}_{ij} - \mathbf{r}_{ij}) \\ \text{s.t.} \ & \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) \leq \mathbf{0} \\ & \mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) = \mathbf{0} \\ \text{where} \ & \overline{\mathbf{x}}_{ij} = [\mathbf{x}_{ij}, \mathbf{t}_{(i+1)k} \forall k \in \mathscr{C}_{ij}] \\ & \mathbf{r}_{ij} = \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\ & \forall j \in \mathscr{E}_i, i = 1, ..., N. \end{aligned} \tag{10}$$

And the dual problem for (10) is:

$$\max_{\lambda} \Lambda(\lambda) \tag{11}$$

Under convexity and constraint qualification assumptions, the strong duality theorem holds and there is no duality gap. As a result, an iterative process can be carried out by first solving (10) with a fixed λ, then updating λ via the subgradient method until convergence. Note that this process is based on solving the dual problem (11), with (10) as its nested problem, to obtain the overall solution of (3). This procedure is theoretically justified by the following theorems, which are presented in Proposition 5.3.2 of [2], p. 514. Again, readers are referred to the text for proofs.

Theorem 3 (Strong Duality Theorem). *Assume that f_{ij} and $\mathbf{g}_{ij}$ are convex, and $\mathbf{h}_{ij}$ is linear for $\forall j \in \mathscr{E}_i, i = 1, ..., N$. Suppose that $\{\overline{\mathbf{x}} : \mathbf{t} - \mathbf{r} = \mathbf{0}, \mathbf{g}(\overline{\mathbf{x}}) < \mathbf{0}, \mathbf{h}(\overline{\mathbf{x}}) = \mathbf{0}\}$ is non-empty. Assume that the optimal value of problem (3) is finite. Then there is no duality gap and there exists at least one Lagrange multiplier.*

Theorem 4. *Assume that the Strong Duality Theorem is satisfied for problem (3). If λ^* is the optimal solution to problem (11), and if $\overline{\mathbf{x}}^* = [\overline{\mathbf{x}}_{ij}^* \forall i, j]$ solves problem (10) with λ=λ^*, then $\overline{\mathbf{x}}^*$ also solves problem (3).*

For each subsystem, consider a fixed λ_{ij}. Also note that since $\mathbf{t}_{ij}$ and $\mathbf{r}_{(i+1)k}$ are constants with respect to element j, the terms $\lambda_{ij}\mathbf{t}_{ij}$ and $\lambda_{(i+1)k}^T\mathbf{r}_{(i+1)k}$ can be eliminated from the objective function, resulting in the following subproblem:

$$
\begin{aligned}
\min_{\overline{\mathbf{x}}_{ij}} \; & f_{ij}(\overline{\mathbf{x}}_{ij}) + \lambda_{ij}^T(-\mathbf{r}_{ij}) + \sum_{k\in\mathscr{C}_{ij}} \lambda_{(i+1)k}^T \mathbf{t}_{(i+1)k} \\
\text{s.t.} \quad & \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) \le \mathbf{0} \\
& \mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) = \mathbf{0} \\
\text{where} \quad & \overline{\mathbf{x}}_{ij} = [\mathbf{x}_{ij}, \mathbf{t}_{(i+1)k} \forall k \in \mathscr{C}_{ij}] \\
& \mathbf{r}_{ij} = \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\
& \forall j \in \mathscr{E}_i, i = 1, \ldots, N.
\end{aligned} \tag{12}
$$

In this method, each subproblem is separable and can be solved in parallel, with fixed Lagrange multipliers. Starting with arbitrary Lagrange multiplier estimates, we update λ using the subgradient method, defined as the following:

$$
\lambda^{\kappa+1} = \lambda^{\kappa} + \tau^{\kappa}(\mathbf{t}^{\kappa} - \mathbf{r}^{\kappa}) \tag{13}
$$

The superscript κ denotes the number of iteration, and τ is a scalar representing the step size. The subgradient method is proven to converge, under assumptions shown in the following theorem [10].

Theorem 5. *If the feasible set of the solution of (10) is bounded and a sequence of step size τ^{κ} satisfies the conditions:*

1. $\tau^{\kappa} \to 0$ *as* $\kappa \to \infty$
2. $\sum_{\kappa=1}^{\infty} \tau^{\kappa} = \infty$

then either the iterative process of the subgradient method terminates finitely at $\kappa = \kappa^*$ with $\Lambda(\lambda^{\kappa^*}) = \Lambda(\lambda^*)$, or else an infinite sequence of iterates is generated such that $\Lambda(\lambda^{\kappa^*}) \to \Lambda(\lambda^*)$ as $\kappa \to \infty$.

Figure 8 shows the OL procedure. By the theorems above, it is necessary that the strong duality condition holds in order to obtain global convergence. This is the main drawback of this method since duality gaps may exist [19], which can cause instability and limit application of the method for practical problems.

The Augmented Lagrangian Method

An improved method, the augmented Lagrangian method (AL) applied to ATC formulations by Tosserams *et al.* [20] combines both the quadratic penalty term and the Lagrangian term, which overcomes some drawbacks of both QP and OL. The Lagrangian term is used to avoid ill-conditioning. The quadratic term also reduces duality gaps. The augmented Lagrangian function is defined as the following:

$$
\pi_{\mathbf{AL}}(\mathbf{t}_{ij} - \mathbf{r}_{ij}) = \lambda_{ij}^T(\mathbf{t}_{ij} - \mathbf{r}_{ij}) + ||\mathbf{w}_{ij} \circ (\mathbf{t}_{ij} - \mathbf{r}_{ij})||_2^2 \tag{14}
$$

With π_{AL}, the general problem is:

$$
\begin{aligned}
\hat{\Lambda}(\lambda) = \min_{\overline{\mathbf{x}}_{11},\ldots,\overline{\mathbf{x}}_{NM}} & \sum_{i=1}^{N} \sum_{j\in\mathscr{E}_i} f_{ij}(\overline{\mathbf{x}}_{ij}) + \sum_{i=2}^{N} \sum_{j\in\mathscr{E}_i} [\lambda_{ij}^T(\mathbf{t}_{ij} - \mathbf{r}_{ij}) \\
& + ||\mathbf{w}_{ij} \circ (\mathbf{t}_{ij} - \mathbf{r}_{ij})||_2^2] \\
\text{s.t.} \quad & \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) \le \mathbf{0} \\
& \mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) = \mathbf{0} \\
\text{where} \quad & \overline{\mathbf{x}}_{ij} = [\mathbf{x}_{ij}, \mathbf{t}_{(i+1)k} \forall k \in \mathscr{C}_{ij}] \\
& \mathbf{r}_{ij} = \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\
& \forall j \in \mathscr{E}_i, i = 1, \ldots, N.
\end{aligned} \tag{15}
$$

The dual problem for (15) is:

$$
\max_{\lambda} \hat{\Lambda}(\lambda) \tag{16}
$$

The following theorem is the basic convergence result of the augmented Lagrangian function. This theorem is a specialization of Proposition 4.2.1 of [2], p. 391. let $X = \{\overline{x} \in \Re^n : \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) \le \mathbf{0},\ \mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) = \mathbf{0}, \forall j \in \mathscr{E}_i, i = 1, \ldots, N.\}$. Under the assumption that $\mathbf{g}_{ij}$ and $\mathbf{h}_{ij}$ for all i, j are continuous on $\Re^n$, the set X is closed. Thus, problem (3) satisfies all assumptions of Proposition 4.2.1 of [2]. Readers are referred to the text for the proof.

Theorem 6. *Assume that the objective and constraint functions of problem (3) are continuous. For $\kappa = 0, 1, \ldots$, let $\overline{\mathbf{x}}^{\kappa}$ be the global minimum of the problem $\hat{\Lambda}(\lambda^{\kappa})$ with $\mathbf{w}^{\kappa}$, where λ^{κ} is bounded, $0 < \mathbf{w}^{\kappa} < \mathbf{w}^{(\kappa+1)}$ for all κ, and $\mathbf{w}^{\kappa} \to \infty$. Then every limit point of the sequence $\{\overline{\mathbf{x}}^{\kappa}\}$ is a global minimum of problem (3).*

For each subproblem, we have:

$$
\begin{aligned}
\min_{\overline{\mathbf{x}}_{ij}} \; & f_{ij}(\overline{\mathbf{x}}_{ij}) + \lambda_{ij}^T(-\mathbf{r}_{ij}) + ||\mathbf{w}_{ij} \circ (\mathbf{t}_{ij} - \mathbf{r}_{ij})||_2^2 \\
& + \sum_{k\in\mathscr{C}_{ij}} [\lambda_{(i+1)k}^T \mathbf{t}_{(i+1)k} \\
& + ||\mathbf{w}_{(i+1)k} \circ (\mathbf{t}_{(i+1)k} - \mathbf{r}_{(i+1)k})||_2^2] \\
\text{s.t.} \quad & \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) \le \mathbf{0} \\
& \mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) = \mathbf{0} \\
\text{where} \quad & \overline{\mathbf{x}}_{ij} = [\mathbf{x}_{ij}, \mathbf{t}_{(i+1)k} \forall k \in \mathscr{C}_{ij}] \\
& \mathbf{r}_{ij} = \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\
& \forall j \in \mathscr{E}_i, i = 1, \ldots, N.
\end{aligned} \tag{17}
$$

Duality theory also applies to the augmented Lagrangian formulation so we can use the same procedure as OL. However, the subproblems as shown in (17) are not separable due to the quadratic penalty term. As a result, instead of solving all subproblems in parallel as in OL, either the nested coordination scheme or the block coordinate descent method is carried out as an inner loop for the AL relaxation formulation, similar to the QP approach. In order to achieve convergence, we update the Lagrange multipliers so they approach to the optimal values. The augmented Lagrangian function allows the use of *method of multipliers*. The scheme for selecting new terms λ from loop iterate

κ to $(\kappa+1)$ is given by the following formula[2]:

$$\lambda^{(\kappa+1)} = \lambda^{\kappa} + 2\mathbf{w}^{\kappa} \circ \mathbf{w}^{\kappa} \circ (\mathbf{t}_{ij}^{\kappa} - \mathbf{r}_{ij}^{\kappa}) \tag{18}$$

Convergence properties of the method of multipliers applied to problem (3) are presented as the following theorem. The proof of this theorem is presented in [11].

Theorem 7. *Assume that the objective function of problem (3) is continuously differentiable, and the set X defined as $X = \{\overline{x} \in \Re^n : \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) \leq \mathbf{0},\ \mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) = \mathbf{0}, \forall j \in \mathscr{E}_i, i = 1, ..., N.\}$ is closed convex. Let*

$$f_{AL}(\overline{\mathbf{x}}, \lambda, \mathbf{w})$$
$$= \sum_{i=1}^{N} \sum_{j \in \mathscr{E}_i} f_{ij}(\overline{\mathbf{x}}_{ij}) + \sum_{i=2}^{N} \sum_{j \in \mathscr{E}_i} [\lambda_{ij}^T (\mathbf{t}_{ij} - \mathbf{r}_{ij}) + ||\mathbf{w}_{ij} \circ (\mathbf{t}_{ij} - \mathbf{r}_{ij})||_2^2]$$

where $\overline{\mathbf{x}} = [\overline{\mathbf{x}}_{11}, ..., \overline{\mathbf{x}}_{NM}]$ and $\lambda = [\lambda_{11}, ..., \lambda_{NM}]$. Let $\overline{\mathbf{x}}^{\kappa}$ be a minimum of problem (15), and suppose that $\{\lambda^{\kappa}\}$ is bounded, and $\{\mathbf{w}^{\kappa}\}$ satisfies:

$$0 < \mathbf{w}^{\kappa} < \mathbf{w}^{\kappa+1}\ \forall \kappa,\ \mathbf{w}^{\kappa} \to \infty$$

Assume that a subsequence $\{\overline{\mathbf{x}}^{\kappa}\}_K$ converges to a vector $\overline{\mathbf{x}}^*$ and $\{\lambda^{\kappa} + 2\mathbf{w}^{\kappa} \circ (\mathbf{t}^{\kappa} - \mathbf{r}^{\kappa})\}_K \to \lambda^*$. Then $\overline{\mathbf{x}}^*$ is a minimum of problem (3) and λ^* is a generalized Lagrange multiplier corresponding to the constraints $\mathbf{t}_{ij} - \mathbf{r}_{ij} = \mathbf{0}$ at $\overline{\mathbf{x}} = \overline{\mathbf{x}}^*$ for all $j \in \mathscr{E}_i$, $i = 1, ..., N$.

There are other ways of updating λ when using the augmented Lagrangian method. For example, the subgradient update scheme is used in [9]. In AL, the penalty weight $\mathbf{w}$ is also updated. A linear scheme with constant scale β for updating $\mathbf{w}$ is used in [20]:

$$\mathbf{w}^{(\kappa+1)} = \beta \mathbf{w}^{\kappa}, \beta \geq 1 \tag{19}$$

Because the quadratic term prevents subproblems from being separable, the nested coordination scheme or BCD is used in the inner loop for finding the optimal solution with respect to fixed λ and $\mathbf{w}$. The outer loop applies (18) and (19) for updating the Lagrange multipliers and penalty weights. In implementing the AL method, the update of $\mathbf{w}$ can also be separate from the update of λ. $\mathbf{w}$ can be updated only if the improvement of the current iteration is not large enough [21]. Figure 8 shows the procedure using nested and BCD schemes. Although the augmented Lagrangian approach shows stable convergence properties and a superior performance compared to QP, the inner loop can still induce large computational cost. Moreover, throughput is low due to non-separability of subproblems.

The Augmented Lagrangian with Alternating Direction Method of Multipliers

To reduce computational cost of the augmented Lagrangian approach, Tosserams *et al.* [20] applied the augmented Lagrangian with alternating direction method of multipliers (ALAD). The key observation is that all elements in the odd levels of the hierarchy only depend on the elements in the even levels and vice versa, so it is possible to first solve all odd-level elements in parallel, then all even-level elements, for a fixed number of iterations. The Lagrange multipliers are then updated using the method of multipliers. The penalty weight can also be updated. Contrary to QP or AL, a large penalty weight may have negative effect on convergence, but a small penalty weight may result in unbounded subproblems [3]. In [20], Tosserams *et. al* keep a constant $\mathbf{w}$. Furthermore, the inner loop is solved for only one iteration to minimize computational effort. This method is demonstrated to be effective on all example problems with superior computational properties compared to QP and AL. It has been proven to converge under the assumption that the problem is convex, the feasibility sets are nonempty, and the constraint sets are bounded [3]. In practice, however, ALAD shows good numerical convergence behavior for non-convex problems as well [21]. Figure 8 shows the ALAD procedure. This method is promising since it demonstrates both good convergence properties and low computational cost through the truncated inner loops and partial parallelization [20].

Summary

Except for the ordinary Lagrangian approach, all other prior methods for consistency constraint relaxation prevent fully parallel computation of subproblems. The ALAD method requires sequential computation of odd levels and even levels, and the QP and AL method require sequential computation of all levels. As parallel and distributed computing power becomes more popular, it is desirable to have fully separable subproblems so that each subproblem can be solved concurrently, and computational throughput can be improved. Since the ordinary Lagrangian approach has convergence difficulties, and other approaches prevent separability, we are motivated to explore alternate approaches.

PROPOSED CONSISTENCY CONSTRAINT RELAXATION METHODS FOR ATC

The Diagonal Quadratic Approximation Method

In order for the subproblems of the augmented Lagrangian approach to be separable, we apply the diagonal quadratic approximation (DQA) originally proposed in [18], where it is used to solve block-angular structure problems. Proofs on convergence and convergence rate are available for this method [18]. There are other variations of solving similar problems using approximation techniques, for example, the method discussed in [19].

In the original paper, DQA is used to solve problems in the following form:

$$\min_{\mathbf{x}} [f(\mathbf{x}) = \sum_{i=1}^{L} f_i(\mathbf{x}_i)] \tag{20}$$

$$\sum_{i=1}^{L} \mathbf{A}_i \mathbf{x}_i = \mathbf{b} \tag{21}$$

$$\mathbf{x}_i \in X_i, i = 1, 2, ..., L \tag{22}$$

where f is convex, and X_1, X_2, ..., X_L are nonempty closed convex sets. Our ATC formulation as in (3) falls into this form for convex problems. First, the objective function is separable with respect to individual subproblems. Second, the consistency constraint function is in the form (21). Third, the inequality and equality constraints $\mathbf{g}$ and $\mathbf{h}$, now fully separable, form sets X_i. In

[2]Note that the penalty weights are squared in the definition of multipliers in this case because they are squared in the definition of the quadratic penalty term.

DQA, a linear approximation is applied on the cross term $\mathbf{t}_{ij}\circ\mathbf{r}_{ij}$ included in the quadratic penalty term $||\mathbf{t_{ij}}-\mathbf{r_{ij}}||_2^2$ in (15).

$$||\mathbf{t}_{ij}-\mathbf{r}_{ij}||_2^2 = ||\mathbf{t}_{ij}\circ\mathbf{t}_{ij}+\mathbf{r}_{ij}\circ\mathbf{r}_{ij}-2(\mathbf{t}_{ij}\circ\mathbf{r}_{ij})||_1 \tag{23}$$

Using the Taylor expansion for multiple variable scalar functions up to the first order, a linearization at the point $\mathbf{t}_{ij}^{\kappa}$, $\mathbf{r}_{ij}^{\kappa}$ gives

$$\begin{aligned}\mathbf{t}_{ij}\circ\mathbf{r}_{ij} &\cong \mathbf{t}_{ij}^{\kappa}\circ\mathbf{r}_{ij}^{\kappa}+\mathbf{r}_{ij}^{\kappa}\circ(\mathbf{t}_{ij}-\mathbf{t}_{ij}^{\kappa})+\mathbf{t}_{ij}^{\kappa}\circ(\mathbf{r}_{ij}-\mathbf{r}_{ij}^{\kappa})\\ &= \mathbf{r}_{ij}^{\kappa}\circ\mathbf{t}_{ij}+\mathbf{t}_{ij}^{\kappa}\circ\mathbf{r}_{ij}-\mathbf{t}_{ij}^{\kappa}\circ\mathbf{r}_{ij}^{\kappa}\end{aligned} \tag{24}$$

Combining (23) and (24), we have:

$$\begin{aligned}&||\mathbf{t}_{ij}-\mathbf{r}_{ij}||_2^2\\ \cong\ & ||\mathbf{t}_{ij}\circ\mathbf{t}_{ij}+\mathbf{r}_{ij}\circ\mathbf{r}_{ij}-2(\mathbf{r}_{ij}^{\kappa}\circ\mathbf{t}_{ij}+\mathbf{t}_{ij}^{\kappa}\circ\mathbf{r}_{ij}-\mathbf{t}_{ij}^{\kappa}\circ\mathbf{r}_{ij}^{\kappa})||_1\\ =\ & ||[\mathbf{t}_{ij}^{\kappa}\circ\mathbf{t}_{ij}^{\kappa}+\mathbf{r}_{ij}\circ\mathbf{r}_{ij}-2(\mathbf{t}_{ij}^{\kappa}\circ\mathbf{r}_{ij})]||_1+\\ & ||[\mathbf{t}_{ij}\circ\mathbf{t}_{ij}+\mathbf{r}_{ij}^{\kappa}\circ\mathbf{r}_{ij}^{\kappa}-2(\mathbf{t}_{ij}\circ\mathbf{r}_{ij}^{\kappa})]||_1+\\ & ||2(\mathbf{t}_{ij}^{\kappa}\circ\mathbf{r}_{ij}^{\kappa})-\mathbf{t}_{ij}^{\kappa}\circ\mathbf{t}_{ij}^{\kappa}-\mathbf{r}_{ij}^{\kappa}\circ\mathbf{r}_{ij}^{\kappa}||_1\\ =\ & ||\mathbf{t}_{ij}^{\kappa}-\mathbf{r}_{ij}||_2^2+||\mathbf{t}_{ij}-\mathbf{r}_{ij}^{\kappa}||_2^2+\mathit{Constant}\end{aligned} \tag{25}$$

By substituting (25) into (15) where $\mathbf{t}_{ij}^{\kappa}$ and $\mathbf{r}_{ij}^{\kappa}$ are solutions obtained from the previous iterations and are constant with respect to the problem of the current iteration, we have derived the overall problem after applying DQA. Note that we have omitted the constant term since the solution of the objective function will not be affected by a constant.

$$\begin{aligned}\min_{\overline{\mathbf{x}}_{11},\ldots,\overline{\mathbf{x}}_{NM}} & \sum_{i=1}^{N}\sum_{j\in\mathscr{E}_i} f_{ij}(\overline{\mathbf{x}}_{ij})+\sum_{i=2}^{N}\sum_{j\in\mathscr{E}_i}[\lambda_{ij}^T(\mathbf{t}_{ij}-\mathbf{r}_{ij})\\ &+||\mathbf{w}_{ij}\circ(\mathbf{t}_{ij}^{\kappa}-\mathbf{r}_{ij})||_2^2\\ &+||\mathbf{w}_{ij}\circ(\mathbf{t}_{ij}-\mathbf{r}_{ij}^{\kappa})||_2^2]\\ \text{s.t.}\quad & \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij})\leq\mathbf{0}\\ & \mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij})=\mathbf{0}\\ \text{where}\quad & \overline{\mathbf{x}}_{ij}=[\mathbf{x}_{ij},\mathbf{t}_{(i+1)k}\,\forall k\in\mathscr{C}_{ij}]\\ & \mathbf{r}_{ij}=\mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij})\\ & \forall j\in\mathscr{E}_i, i=1,\ldots,N.\end{aligned} \tag{26}$$

And for each subproblem:

$$\begin{aligned}\min_{\overline{\mathbf{x}}_{ij}}\ & f_{ij}(\overline{\mathbf{x}}_{ij})+\lambda_{ij}^T(-\mathbf{r}_{ij})+||\mathbf{w}_{ij}\circ(\mathbf{t}_{ij}^{\kappa}-\mathbf{r}_{ij})||_2^2\\ &+\sum_{k\in\mathscr{C}_{ij}}[\lambda_{(i+1)k}^T\mathbf{t}_{(i+1)k}\\ &+||\mathbf{w}_{(i+1)k}\circ(\mathbf{t}_{(i+1)k}-\mathbf{r}_{(i+1)k}^{\kappa})||_2^2]\\ \text{s.t.}\quad & \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij})\leq\mathbf{0}\\ & \mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij})=\mathbf{0}\\ \text{where}\quad & \overline{\mathbf{x}}_{ij}=[\mathbf{x}_{ij},\mathbf{t}_{(i+1)k}\,\forall k\in\mathscr{C}_{ij}]\\ & \mathbf{r}_{ij}=\mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij})\\ & \forall j\in\mathscr{E}_i, i=1,\ldots,N.\end{aligned} \tag{27}$$

The DQA approach consists of an inner loop and an outer loop. The inner loop is used to improve linearization while the outer loop is used to implement the method of multipliers. We concurrently update the penalty weight using the linear update method, similar to the AL approach. The general DQA algorithm consists of the following steps.

1. initialize $\overline{\mathbf{x}}$, λ, and $\mathbf{w}$, and set $\kappa=0$, where κ denotes the number of outer loop iteration.
2. given $\overline{\mathbf{x}}^{\kappa}$, the final solution of the previous $(\kappa-1)$st outer loop iteration upon inner loop convergence, set $s=0$, where s is the inner loop iteration, and $\overline{\mathbf{x}}^{\kappa+1,0}=\overline{\mathbf{x}}^{\kappa}$, where $\overline{\mathbf{x}}^{\kappa+1,s}$ is the solution of the sth inner loop iteration and the current outer loop iteration.
3. for each element, solve for $\overline{\mathbf{x}}_{ij}$ in (27) in parallel, and obtain $\overline{\mathbf{x}}_{ij}^{\kappa+1,s+1}$.
4. if $max(||\mathbf{t}^{\kappa+1,s+1}-\mathbf{t}^{\kappa+1,s}||,||\mathbf{r}^{\kappa+1,s+1}-\mathbf{r}^{\kappa+1,s}||)\leq\sigma_{inner}$, where σ_{inner} is the inner loop consistency deviation tolerance, set $\overline{\mathbf{x}}^{\kappa+1}=\overline{\mathbf{x}}^{\kappa+1,s+1}$, and go to step 5. Otherwise, set $\overline{\mathbf{x}}^{\kappa+1,s+1}=\overline{\mathbf{x}}^{\kappa+1,s}+\tau(\overline{\mathbf{x}}^{\kappa+1,s+1}-\overline{\mathbf{x}}^{\kappa+1,s})$, where τ is the step size, set $s=s+1$, and go to step 3.
5. if $max(||\mathbf{t}^{\kappa+1}-\mathbf{t}^{\kappa}||,||\mathbf{r}^{\kappa+1}-\mathbf{r}^{\kappa}||)\leq\sigma_{outer}$, where σ_{outer} is the outer loop consistency deviation tolerance, then stop, and set the optimal solution $\overline{\mathbf{x}}^{*}$ to be $\overline{\mathbf{x}}^{\kappa+1}$; otherwise, set $\kappa=\kappa+1$, and update the Lagrange multipliers using the method of multipliers by setting $\lambda^{\kappa+1}=\lambda^{\kappa}+\mathbf{w}^{\kappa}\circ(\mathbf{t}^{\kappa}-\mathbf{r}^{\kappa})$, $\mathbf{w}^{(\kappa+1)}=\beta\mathbf{w}^{\kappa}$, and go to step 2.

This procedure is shown in Figure 8. Note that the consistency deviation tolerance for both the inner loop σ_{inner} and the outer loop σ_{outer} should be significantly smaller than the step size τ to prevent premature convergence.

Convergence is proven for the above algorithm with $\sigma_{inner}=0$, as stated in [18]. For practical purposes, however, we allow σ_{inner} to be small but non-zero.

Theorem 8. *Suppose that the constraint sets are bounded, $\sigma_{inner}=0$, and the step size $\tau\in(0,1)$ is significantly small. The following statements hold:*

1. *For all i,j,* $\lim_{s\to\infty}(\mathbf{t}^{\kappa,s+1}-\mathbf{t}^{\kappa,s})=\mathbf{0}$, $\lim_{s\to\infty}(\mathbf{r}^{\kappa,s+1}-\mathbf{r}^{\kappa,s})=\mathbf{0}$
2. *Each limit point of the sequence* $\{\overline{\mathbf{x}}^{\kappa,s}\}_{s=0}^{\infty}$ *is a solution of* $\hat{\Lambda}(\lambda^{\kappa})$.

The proof of the theorem is based on the estimation of the difference between the approximation and the augmented Lagrangian function. This estimation shows that a progress in the former also introduces progress in the later. The step size τ plays an important role in the DQA method. Since linearization is only accurate in a neighborhood of the point at which the linearization takes place, we use a small step size to obtain a good approximation. As noted in [18] the step size τ is related to the number of linking variables. Proof of the above theorem requires smaller τ for problems with a larger number of linking variables. However, empirical results indicate that a step size close to 1 can bring convergence to all example problems presented in this paper.

Local convergence is also discussed in [18], and this property of DQA is based on the quadratic growth condition of the augmented Lagrangian function, defined as the following:

Definition 1 (Quadratic growth condition). *let*

$$f_{AL} = \sum_{i=1}^{N}\sum_{j\in\mathscr{E}_i} f_{ij}(\overline{\mathbf{x}}_{ij}) + \sum_{i=2}^{N}\sum_{j\in\mathscr{E}_i}[\lambda_{ij}^T(\mathbf{t}_{ij}-\mathbf{r}_{ij}) + ||\mathbf{w}_{ij}\circ(\mathbf{t}_{ij}-\mathbf{r}_{ij})||_2^2],$$

and $\overline{\mathbf{x}}^*$ be the solution of (15), where $\overline{\mathbf{x}} = [\overline{\mathbf{x}}_{11},...,\overline{\mathbf{x}}_{NM}]$. The augmented Lagrangian function is said to satisfy the quadratic growth condition if there exists $\gamma > 0$ such that for every $\overline{\mathbf{x}}$,

$$f_{AL}(\overline{\mathbf{x}}) - f_{AL}(\overline{\mathbf{x}}^*) \geq \gamma||\overline{\mathbf{x}} - \overline{\mathbf{x}}^*||_2^2.$$

Under the quadratic growth rate of the augmented Lagrangian function, the rate of convergence for DQA is linear. However, other factors, such as a large number of linking variables can slow down the progress [18]. As a variation, it is possible to linearize at the mid-point of the solution from the previous iteration $(\frac{\mathbf{t}^\kappa+\mathbf{r}^\kappa}{2}, \frac{\mathbf{t}^\kappa+\mathbf{r}^\kappa}{2})$. We obtain similar results as linearization at end points in all of our test examples.

The Truncated Diagonal Quadratic Approximation Method

As will be discussed in the next section, DQA performs well on test problems in terms of throughput, but the overall computational cost is still high. From the experiments, we have found out that much of the computational effort is spent on the inner linearization loop. We have also observed that usually the inner loop progresses slowly and introduces a high cost to reach the desirable inner loop convergence tolerance. However, when the Lagrange multipliers are not optimal, high accuracy of the subproblem solutions is not necessary, and the computational effort is wasted. It is more desirable to quickly update the Lagrange multiplier to move towards its optimal value. This can be achieved by limiting the total number of inner loop iterations in DQA by treating it as a user-specified parameter in a way that is similar to the ALAD approach, which reduces the computational cost for solving the inner loop. Since some inner loop calculation is omitted in DQA, this method is called the truncated diagonal quadratic approximation method, or TDQA. The outer loop of TDQA implements the method of multipliers. Similar to ALAD, the penalty weight is held fixed.

In order to minimize overall cost, only one iteration of computation is carried out in the inner loop. The TDQA algorithm is given as the following:

1. initialize $\overline{\mathbf{x}}^0$, λ, and $\mathbf{w}$, and set $\kappa = 0$, where κ denotes the number of loop iterations.
2. for each element, solve for $\overline{\mathbf{x}}_{ij}$ in (27) in parallel, and obtain $\overline{\mathbf{x}}_{ij}^{\kappa+1} = \overline{\mathbf{x}}_{ij}^{\kappa} + \tau(\overline{\mathbf{x}}_{ij}^{\kappa+1} - \overline{\mathbf{x}}_{ij}^{\kappa})$ where τ is the step size
3. if $max(||\mathbf{t}^{\kappa+1} - \mathbf{t}^{\kappa}||, ||\mathbf{r}^{\kappa+1} - \mathbf{r}^{\kappa}||) \leq \sigma$, where σ is the outer loop consistency deviation tolerance, then stop, and set the optimal solution $\overline{\mathbf{x}}^*$ to be $\overline{\mathbf{x}}^{\kappa+1}$; otherwise, set $\kappa = \kappa + 1$, update the Lagrange multipliers using the method of multipliers by setting $\lambda^{\kappa+1} = \lambda^{\kappa} + \mathbf{w}^{\kappa}\circ(\mathbf{t}^{\kappa} - \mathbf{r}^{\kappa})$, $\mathbf{w}^{(\kappa+1)} = \beta\mathbf{w}^{\kappa}$, and go to step 2.

This procedure is shown in Figure 8. The outer loop toleranceσ should be significantly smaller than the step size τ to prevent premature convergence. Furthermore, we can also linearize at the mid-point of the solution obtained from the previous iteration. Again, empirical results show similar performance as linearization at end points.

Empirical results show promising results of the TDQA method. The intuition is that TDQA can be considered as an approximation of the ALAD method. As long as the approximation is accurate, solution of TDQA at each iteration is close to that of the ALAD method, and convergence follows by the convergence property of the ALAD method. We can also impose a strategy for increasing the total number of inner loop iterations when the improvement of the actual function is not large enough. In the extreme case when we allow the limit of the number of inner loop iterations to achieve infinity, the method becomes the DQA method, and convergence can be obtained. A proof of convergence for finite inner loop truncation is left for future work.

DQA and TDQA with the Trust Region Technique

From the discussion above, it is possible to have a linearization at the current point that is a poor approximation to the augmented Lagrangian function. A small step size τ can ensure accuracy of the approximation. However, if the step size is too small, convergence will be slow. As a result, some trial and error is required for finding a good value of the step size for practical applications.

An alternative approach for finding a good step size is to use a trust region. To determine whether the current linearization is accurate, we estimate the ratio ρ of the actual reduction (the reduction of the original augmented Lagrangian function) to the predicted reduction (the reduction of the diagonal quadratic approximation of the augmented Lagrangian function) between current solution $\overline{\mathbf{x}}_{ij}^{\kappa+1}$ and previous solution $\overline{\mathbf{x}}_{ij}^{\kappa}$. If ρ is close to one, the linearization is accurate. On the other hand, whenever the ratio ρ is far from one, it indicates that the step size for the current solution is too big, the linearization is not accurate, and it is desirable to use a smaller step size. The algorithm of DQA and TDQA can be modified according to this method in the following way. Instead of using a predetermined step size, we calculate the reduction ratio ρ after each iteration. If $\rho \geq \gamma$, where γ is the ratio threshold, accept the solution. Otherwise, shrink the step size by half. The recommended value of γ is 0.25 in the standard trust region method. This procedure can also be carried out iteratively.

RESULTS AND COMPARISONS

For the purpose of a clear comparison among all ATC methods, we use the four examples discussed in [20], in the same order. Readers are referred to [20] for a more in depth discussion on the structures and properties of each problem. To make sure that the comparison is fair, we use the same condition on terminating the update of the Lagrange multipliers for all methods, as shown in the following:

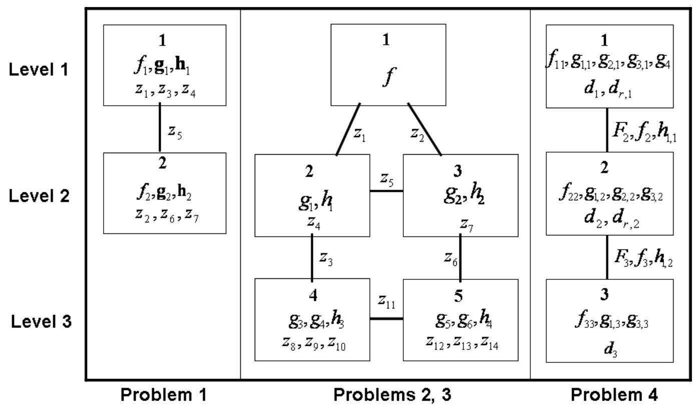

Figure 3. EXAMPLE PROBLEM STRUCTURES

$$max(||\mathbf{t}^{\kappa} - \mathbf{t}^{\kappa-1}||, ||\mathbf{r}^{\kappa} - \mathbf{r}^{\kappa-1}||) \leq \sigma_{outer} \tag{28}$$

This is different from [20], where the termination condition is:

$$max(||(\mathbf{t}^{\kappa} - \mathbf{r}^{\kappa}) - (\mathbf{t}^{\kappa-1} - \mathbf{r}^{\kappa-1})||) \leq \sigma_{outer} \tag{29}$$

We use a new criterion since (29) might result in premature convergence [3].

In this paper, three quantities are used to evaluate performance. The solution error, the total number of function evaluations, and the overall computational throughput. Solution error ξ of method C is defined as the following, the same as [20]:

$$\xi = ||\mathbf{x}^* - \mathbf{x}^{ATC}||_\infty$$

where $\mathbf{x}^*$ is the actual optimal solution and $\mathbf{x}^{ATC}$ is the solution found by ATC. The magnitude of the solution error is controlled by the convergence tolerance. The tighter the tolerance, the more accurate the solution. The total number of function evaluations, also defined in [20], is used to measure the overall computational cost, and is reported by the subproblem solver, in this case, Tomlab NPSol. Computational throughput is defined as the number of tasks that can be performed for a given time period. Since each problem is considered a task, for simplicity we use the inverse of throughput, or latency in CPU seconds, to compare the speed for solving each problem of each method. In other words, latency measures the total CPU time required to solve the overall problem, taking into account parallelization.

The structures of the four examples are shown in Figure 3. We compare the performance of each method in the following two ways: function evaluations vs solution error and computational latency vs solution error[4]. The methods being compared are the quadratic penalty method with nested coordination scheme and WUM (QP), the quadratic penalty method with BCD and WUM (QP-BCD), the augmented Lagrangian method with nested coordination scheme (AL), the augmented Lagrangian method with BCD (AL-BCD), ALAD, DQA, and TDQA. The value of β for updating penalty weights is set to 2.0 for AL and DQA, and 1.0 for ALAD and TDQA. **w** Step size for DQA is 0.9 for all problems. For TDQA, we invoke the trust region once to ensure the accuracy of the linearization, and the initial step size is set to 0.7. Note that OL is not included since it encounters convergence difficulties for the last three problems. All tests are carried out on the same machine with the same applications so comparison is fair.[5]

Example 1

The first example is a two-level decomposition of the geometric programming problem. It has a unique solution $\mathbf{z}^* = [2.15, 2.06, 1.32, 0.76, 1.07, 1.00, 1.47]$ with all constraints active.

$$\begin{aligned}
\min_{z_1,\ldots,z_7} \quad & f = f_1 + f_2 = z_1^2 + z_2^2 \\
\text{s.t.} \quad & g_1 = (z_3^{-2} + z_4^2) z_5^{-2} - 1 \leq 0 \\
& g_2 = (z_5^2 + z_6^{-2}) z_7^{-2} - 1 \leq 0 \\
& h_1 = (z_3^2 + z_4^{-2} + z_5^2) z_1^{-2} - 1 = 0 \\
& h_2 = (z_5^2 + z_6^2 + z_7^2) z_2^{-2} - 1 = 0 \\
& z_1, z_2, \ldots, z_7 \geq 0
\end{aligned} \tag{30}$$

Figure 4 shows results for each method for this example. Notice that since this is a two-level problem, QP and QP-BCD are identical, as are AL and AL-BCD. It is clear that the truncated methods ALAD and TDQA perform better than all the nested loop methods in terms of both function evaluations and latency. TDQA outperforms ALAD. DQA requires more function evaluations than AL but has a lower latency due to parallelization. QP does not perform as well as the other methods.

Example 2

The second example problem is a three-level decomposition of a posynomial geometric programming problem. Its unique solution is $\mathbf{z}^* = [2.84, 3.09, 2.36, 0.76, 0.87, 2.81, 0.94, 0.97, 0.87, 0.80, 1.30, 0.84, 1.76, 1.55]$ with all constraints active.

$$\begin{aligned}
\min_{z_1,\ldots,z_{14}} \quad & f = f_1 + f_2 = z_1^2 + z_2^2 \\
\text{s.t.} \quad & g_1 = (z_3^{-2} + z_4^2) z_5^{-2} - 1 \leq 0 \\
& g_2 = (z_5^2 + z_6^{-2}) z_7^{-2} - 1 \leq 0 \\
& g_3 = (z_8^2 + z_9^2) z_{11}^{-2} - 1 \leq 0 \\
& g_4 = (z_8^{-2} + z_{10}^2) z_{11}^{-2} - 1 \leq 0
\end{aligned}$$

[3]Generally speaking, three convergence criteria are widely used in the implementation of optimization methods: gradient of the Lagrangian function is close to 0, objective function value stops changing, and solution point stops changing. We are using the third method here, and for practical purposes, a small non-zero convergence tolerance is given to determine when the solution stops changing.

[4]We did not use the number of redesigns(i.e., the number of times each subproblem must be solved) as a metric since it can be misleading. In some iterations the subproblem may take many function evaluations to solve, while in many of the later iterations it will be very fast since the starting point is close to the solution. Thus, the redesigns metric does not seem to be an accurate or easy-to-interpret metric of computational cost.

[5]Application: Matlab Version 7.0 with Tomlab NPSol Solver Version 5.3; OS: SUSE Linux; Processor: Intel(R) Xeon(TM); CPU: 2.80GHz. Also, for the test results we used the time required to complete the longest running subproblem in each iteration to measure the effect of imperfect load balancing. We did not explicitly capture the communication overhead of multiprocessors, but this aspect will be very small in the examples compared to the computation time required at each iteration.

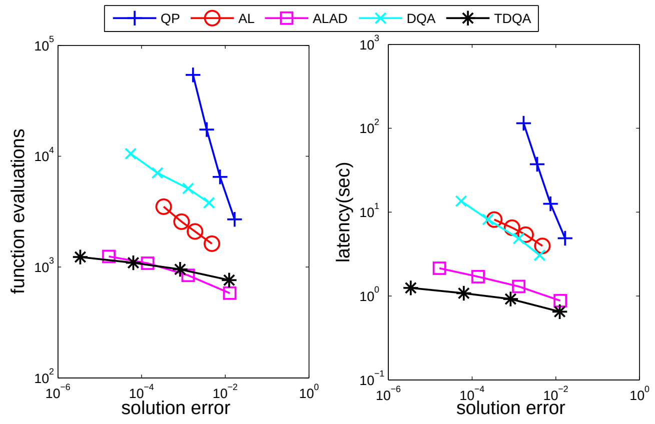

Figure 4. EXAMPLE 1 COMPUTATION COST AND THROUGH VS SOLUTION ACCURACY

$$
\begin{aligned}
g_5 &= (z_{11}^2 + z_{12}^{-2})z_{13}^{-2} - 1 \le 0\\
g_6 &= (z_{11}^2 + z_{12}^2)z_{14}^{-2} - 1 \le 0\\
h_1 &= (z_3^2 + z_4^{-2} + z_5^2)z_1^{-2} - 1 = 0\\
h_2 &= (z_5^2 + z_6^2 + z_7^2)z_2^{-2} - 1 = 0\\
h_3 &= (z_8^2 + z_9^{-2} + z_{10}^{-2} + z_{11}^2)z_3^{-2} - 1 = 0\\
h_4 &= (z_{11}^2 + z_{12}^2 + z_{13}^2 + z_{14}^2)z_6^{-2} - 1 = 0\\
& z_1, z_2, \ldots, z_{14} \ge 0 \qquad (31)
\end{aligned}
$$

Figure 5 shows the results of example 2. In this example, the advantage of BCD is clear since QP-BCD and AL-BCD outperform QP and AL, respectively. Moreover AL outperforms QP and QP-BCD. DQA outperforms AL and AL-BCD in computational latency due to parallelization. Similar to example 1, the truncated methods are superior to all nested loop methods. And TDQA outperforms DQA in terms of both function evaluations and latency.

Example 3

The third example is a slight variation of the second one, changing only the objective function so the targets are attainable. The objective function is now $f = f_1 + f_2 = (z_1 - 2.9)^2 + (z_2 - 3.1)^2$. In this example, only $z_1 = 2.9$ and $z_2 = 3.1$ are unique. The other variables can have different values as long as all constraints are satisfied.

This example is a special case. As discussed in [20], the Lagrange multipliers were originally set to zero, which happens to be the optimal value for this problem. As a result, the outer loops of all nested loop methods are theoretically unnecessary. However, due to numerical errors, the outer loops are updated several times in practice.

Similar to example 2, QP-BCD and AL-BCD outperform QP and AL, respectively. However, since λ is at its optimal value originally, the advantage of the Lagrangian term cannot be observed, and the QP methods outperform the AL methods in this example. For the truncated methods ALAD and TDQA, which outperform all nested loop methods, ALAD outperforms TDQA in terms of both function evaluations and latency. These results are shown in Figure 6. The total computational cost, and the variation among methods, is small for this problem because updating

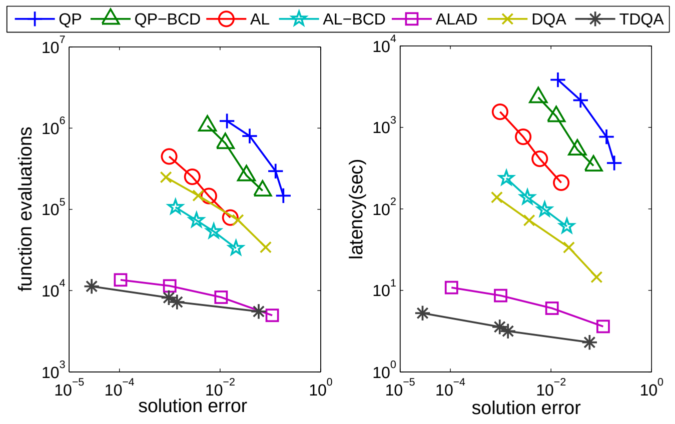

Figure 5. EXAMPLE 2 COMPUTATIONAL COST AND THROUGHPUT VS SOLUTION ACCURACY

λ and $\mathbf{w}$ is unnecessary.

Example 4

The fourth example is a structured optimization problem based on the analytical mass allocation problem. It has a unique solution $\mathbf{z}^* = [0.0346, 0.0349, 0.0294, 0.0046, 0.0028]$.

$$
\begin{aligned}
\min_{d_1,d_2,d_{r,1},d_{r,2}} & \sum_{i=1}^{3} m_i + \sum_{j=1}^{2} m_{r,j}\\
\text{s.t.}\quad & g_{1,i} = \sigma_{b,i} - \overline{\sigma} \le 0,\ i = 1,2,3\\
& g_{2,j} = \sigma_{a,j} - \overline{\sigma} \le 0,\ j = 1,2\\
& g_{3,i} = F_{t,i} - \overline{F_t} \le 0,\ i = 1,2,3\\
& g_4 = f_1 - \overline{f_1} \le 0\\
& h_1 = f_i - f_{i+1} - f_{r,i} = 0,\ i = 1,2\\
\text{where}\quad & m_i = \frac{\pi}{4} d_i^2 L\rho,\ i = 1,2,3\\
& m_{r,j} = \frac{\pi}{4} d_{r,j}^2 L\rho,\ j = 1,2\\
& \sigma_{b,i} = \frac{32L(F_i - F_{i+1})}{\pi d_i^3},\ i = 1,2,3\\
& f_i = \frac{64L^3(F_i - F_{i+1})}{3\pi E d_i^4},\ i = 1,2,3\\
& \sigma_{a,j} = \frac{4F_{j+1}}{\pi d_{rj}^2},\ j = 1,2\\
& f_{r,j} = \frac{4F_{j+1}L}{\pi E d_{rj}^2},\ j = 1,2\\
& \rho = 2700 kg/m^3\\
& E = 70GPa\\
& F_1 = 1000N \qquad (32)
\end{aligned}
$$

The results of example 4 shown in Figure 7 are very similar to those in example 2. QP-BCD and AL-BCD outperform QP and AL, respectively. the AL methods outperforms the QP methods. DQA outperforms AL: It has a larger number of function

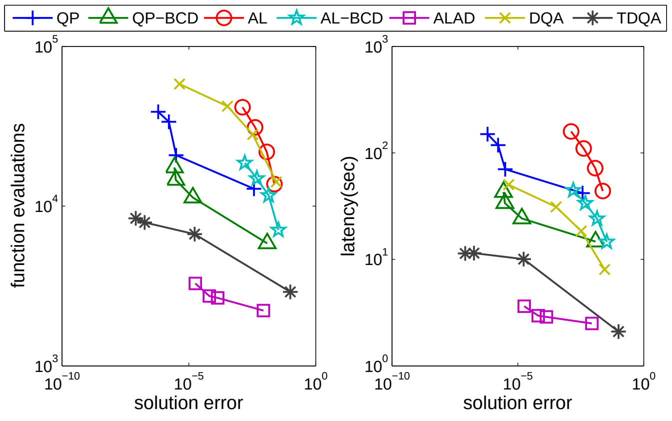

Figure 6. EXAMPLE 3 COMPUTATIONAL COST AND THROUGHPUT VS SOLUTION ACCURACY

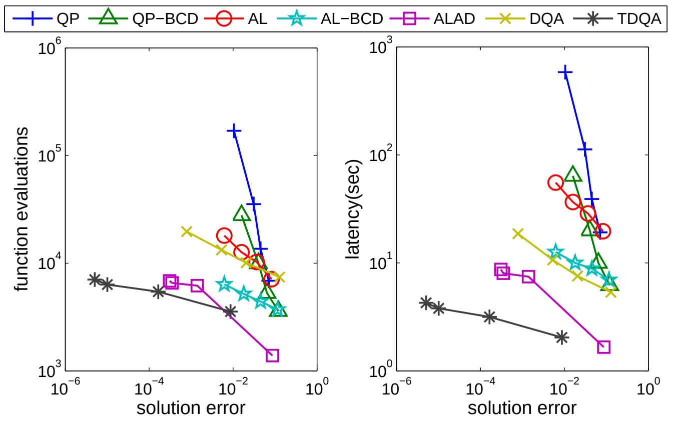

Figure 7. EXAMPLE 4 COMPUTATION COST AND THROUGH VS SOLUTION ACCURACY

evaluations than AL-BCD, but it has the smallest latency among all nested loop methods. The truncated methods are superior to all nested loop methods. And TDQA outperforms ALAD.

DISCUSSION

The biggest advantage of both DQA and TDQA is their ability to separate subproblems for parallel computation. This property is highly desirable, especially in large-scale problems: Since each subproblem is separable, nested loops are avoided. Moreover, since each problem can be solved in parallel, throughput is greatly increased. Although we only present DQA and TDQA in the context of hierarchical ATC problems, these methods can be used in a wide variety of decomposition methods. DQA was first proposed in [18] as a method solving general block-angular structured problems, and TDQA can be applied to any problems to which DQA can be applied. For example, in [21] an augmented Lagrangian approach is used in solving non-hierarchical dual-block angular problems, and DQA and TDQA can also be used for solving these problems. This is also an advantage of DQA and TDQA over ALAD. DQA and TDQA do not require problems to be hierarchical, while ALAD requires identification of independent levels in hierarchical problems. It is expected that DQA and TDQA will perform similarly in non-hierarchical decomposition schemes as in ATC.

One limitation of DQA and TDQA is that theoretically these methods achieve good performance only when the number of linking variables are small, as in the case of quasiseparable problems. Further investigation is needed to determine their performance in practice on problems that have large number of linking variables after decomposition. Future research is also needed to investigate the global and local convergence properties of the TDQA method.

CONCLUSION

In this paper, we have summarized the existing methods, compared the block coordinate descent method to nested schemes, and presented the diagonal quadratic approximation method and the truncated diagonal quadratic approximation method used in handling consistency constraint relaxation in ATC. The DQA method is supported by theoretical justification. Fundamental results show that by using BCD instead of the nested coordination scheme, computational cost and throughput are greatly improved. BCD is also theoretically justified. In general, DQA has lower computational cost than most nested loop methods and it has the highest throughput due to parallelization. TDQA has the least total cost and best throughput of all methods in all the test examples except for example 3, for which ALAD has achieved the best performance. In this degenerate case, updates of the penalty weights and Lagrange multipliers are unnecessary, and the total computational time is much smaller than the other non-degenerate cases, making distinctions less critical. As a result, we believe that TDQA is more preferable when parallel processing is available. DQA has proofs for local and global convergence, and proofs for TDQA are left for future work. The flow charts for the algorithms of all methods is shown in Figure 8 and a table summarizing methods and results can be found in [11].

The proposed methods overcome many of the concerns with prior approaches to ATC, such as convergence difficulties, ill-conditioning, and computational cost associated with the coordination strategies. DQA and TDQA enable parallelization of all subsystems in the ATC hierarchy, which can improve computational throughput when parallel processing is available. Given the theoretical benefits and promising empirical results of the DQA and TDQA approaches to ATC, we hope to see these methods utilized in future ATC studies and applications. Additionally, because DQA and TDQA do not require the assumption of hierarchical structures, extension to nonhierarchical structures has the potential to extend applicability to a wider range of systems optimization problems.

ACKNOWLEDGEMENTS

The authors wish to thank Simon Tosserams for sharing his code and both Simon Tosserams and Larry Biegler for their helpful feedback. This work was supported in part by the Pennsylvania Infrastructure Technology Alliance, a partnership of Carnegie Mellon, Lehigh University, and the Commonwealth of Pennsylvania's Department of Community and Economic Development (DICED).

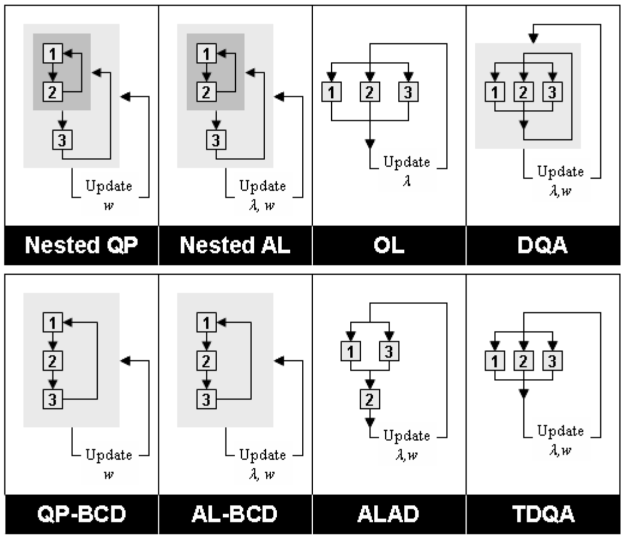

Figure 8. FLOW CHARTS OF METHODS

REFERENCES

[1] J. Allison, M. Kokkolaras, M. Zawislak, and P. Papalambros. "On the use of analytical target cascading and collaborative optimization for complex system design". In *Proceedings of the 6th World Congress on Structural and Multidisciplinary Optimization*, Rio de Janeiro, Brazil, 2005.

[2] D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, Massachusetts, 2 edition, 2003.

[3] D.P. Bertsekas and JN Tsitsiklis. *Parallel and Distributed Computation*. Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

[4] R. Choudhary, A. Malkawi, and P.Y. Papalambros. "Analytic target cascading in simulation-based building design". *Automation in Construction*, 14(4):551 – 568, 2005.

[5] A. DeMiguel and W. Murray. "A local convergence analysis of bilevel decomposition algorithms". *Optimization and Engineering*, 7:99 – 133, December 2006.

[6] R.T. Haftka and L.T. Watson. "Multidisciplinary design optimization with quasiseparable subsystems". *Optimization and Engineering*, 6(1):9 – 20, March 2005.

[7] H.M. Kim, N.F. Michelena, P.Y. Papalambros, and T.Jiang. "Target cascading in optimal system design". *Journal of Mechanical Design*, 125(3):474 – 480, September 2003.

[8] H.M. Kim, M.Kokkolaras, L.S. Louca, G.J. Delagrammatikas, N.F. Michelena, Z.S. Filipi, P.Y. Papalambros, J.L. Stein, and D.N. Assanis. "Target cascading in vehicle redesign: a class vi truck study". *International Journal of Vehicle Design*, 29(3):199 – 225, 2002.

[9] H.M. Kim, W.Chen, and M.M. Wiecek. "Lagrangian coordination for enhancing the convergence of analytical target cascading". *AIAA Journal*. To appear.

[10] J.B. Lassiter, M.M. Wiecek, and K.R. Andrighetti. "Lagrangian coordination and analytical target cascading: Solving ATC-decomposed problems with lagrangian duality". *Optimization and Engineering*, 6(3):361 – 381, September 2005.

[11] Y. Li, Z. Lu, and J. Michalek. "Diagonal quadratic approximation for parallelization of analytical target cascading". *ASME Journal of Mechanical Design*. In Press.

[12] J.J. Michalek, O. Ceryan, P.Y. Papalambros, and Y.Koren. "Balancing marketability and manufacturability in product line design decision-making". *ASME Journal of Mechanical Design*, 2006.

[13] J.J. Michalek, F.M. Feinberg, and P.Y. Papalambros. "Linking marketing and engineering product design decisions via analytical target cascading". *Journal of Product Innovation Management*, 22:42 – 62, 2005.

[14] J.J Michalek and P.Y. Papalambros. "An efficient weighting update method to achieve acceptable inconsistency deviation in analytical target cascading". *Journal of Mechanical Design*, 127(3):206 – 214, March 2005.

[15] J.J. Michalek and P.Y. Papalambros. "Weights, norms, and notation in analytical target cascading". *Journal of Mechanical Design*, 127(3):499 – 501, May 2005.

[16] N. Michelena, H. Park, and P.Y. Papalambros. "Convergence properties of analytical target cascading". *AIAA Journal*, 41(5):897 – 905, 2003.

[17] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer series in operations research. Springer, 1999.

[18] A. Ruszczynski. "On convergence of an augmented lagrangian decomposition method for sparse convex optimization". *Mathematics of Operations Research*, 20(3):634 – 656, August 1995.

[19] G. Stephanopoulos and A. W. Westerberg. "The use of hestenes' method of multipliers to resolve dual gaps in engineering system optimization". *Journal of Optimization Theory and Applications*, 15(3):285 – 309, March 1975.

[20] S. Tosserams, L.F.P. Etman, and J.E. Rooda. "An augmented Lagrangian relaxation for analytical target cascading using the alternating directions method of multipliers". *Structural and Multidisciplinary Optimization*, 31(3):176 – 189, March 2006.

[21] S. Tosserams, P. Etman, and J. Rooda. "An augmented lagrangian decomposition method for dual block-angular problems in MDO". In *Proceedings of the AIAA*, 2006.