DIAGONAL QUADRATIC APPROXIMATION FOR PARALLEL COMPUTING WITH ANALYTICAL TARGET CASCADING

THESIS

Presented to the Faculty of the Department of Mathematical Sciences
of Carnegie Mellon University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science

Yanjing Li

August, 2006

DIAGONAL QUADRATIC APPROXIMATION FOR PARALLEL COMPUTING WITH ANALYTICAL TARGET CASCADING

Yanjing Li

Approved:

Dr. Jeremy Michalek Thesis Advisor Department of Mechanical Engineering	Date
Dr. Zhaosong Lu Thesis Advisor Department of Mathematical Sciences	Date
Dr. Lorenz Biegler Committee Member Department of Chemical Engineering	Date

Table of Contents

List of Figures

Abstract

Analytical Target Cascading (ATC) is an effective decomposition algorithm used for engineering design optimization problems that have hierarchical structures. The overall system is split into subsystems, which are solved separately and coordinated by 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. Among the existing implementation methods, the augmented Lagrangian approach is known to have the most stable convergence properties, however, the augmented Lagrangian function is not separable due to the quadratic penalty term. To overcome this drawback, we apply diagonal quadratic approximation (DQA) by linearizing the cross term of the augmented Lagrangian function. An inner loop is used to update the linearization by solving all subproblems in parallel, and an outer loop is used to implement the method of multipliers. 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. For both DQA and TDQA, to ensure that the linearization approximation is accurate, choosing an appropriate step size is the key, and we impose a trust region for this purpose. Empirical results show that both the computational throughput and the overall computational cost of DQA are superior to all nested loop methods, and that TDQA has lower overall computational cost and throughput than the best previously reported results.

DIAGONAL QUADRATIC APPROXIMATION FOR PARALLEL COMPUTING WITH ANALYTICAL TARGET CASCADING

I. Introduction

Engineering design optimization problems often involve large, complex mathematical formulations that are difficult to solve directly. Many methodologies have been developed to support the endeavor of solving these design problems. A common approach is to decompose the overall system into a number of interacting subsystems. By solving each subsystem, and using a coordination scheme to ensure consistency, it is possible to obtain the optimal solution for the overall system.

One category of engineering design optimization methods that employ decomposition technique is multidisciplinary design optimization (MDO). MDO methods are used to solve design problems incorporating a number of analysis models representing different disciplines. Popular MDO methods include collaborative optimization (CO), concurrent subspace optimization (CSSO) and bi-level integrated system synthesis (BLISS). The analytical target cascading (ATC) method differs from most MDO methods in that it is based on the hierarchical structure of a system by setting targets at the system level to be achieved by subsystem designs [6]. Readers are referred to [2] for a discussion on the comparison between ATC and CO.

Many of the MDO methods lack formal mathematical justifications and may experience convergence difficulties [1]. ATC is one of the few methods that has been proven to converge to the optimal system solution [12], [14], [18]. More recently, two methods used for solving non-hierarchical quasiseparable problems, or dual block-angular problems, have been proposed and theoretically and numerically justified. The dual block-angular problems consist of a number of subsystems that are almost separable except for a few linking variables. And these methods are the constraint margin method by Haftka and Watson [5] and the augmented Lagrangian approach by Tosserams *et al.* [19]. ATC involves the translation of hierarchical problems into dual block-angular structures. It is widely used in practical engineering problems, such as automotive design [7] and multidisciplinary product development [10], [11].

In ATC, top level design targets are propagated to lower levels, which are optimized to meet the targets. The resulting responses are reballanced at higher levels to achieve consistency. The optimal system solutions are obtained by an iterative process until target/response consistency is achieved globally. All variations of implementation of ATC make use of a relaxation of the consistency constraint, by treating it as part of the objective function using a quadratic penalty term, a Lagrangian term, or both. Whenever a quadratic term is present, subproblems are not separable, and sequential computation needs to be carried out throughout the hierarchical chain, thus introducing high computational cost. Although separability can be obtained when using the ordinary Lagrangian function, the method often encounters convergence difficulties due to the existence of duality gaps.

The goal of this thesis is to present two methods that overcome the drawbacks of the existing methods. First, we apply diagonal quadratic approximation (DQA) to the augmented Lagrangian function to achieve separability and support parallelization. Both global and local convergence properties of DQA are discussed. Second, a truncated DQA (TDQA) method is proposed to further reduce overall computational cost. The trust region technique can also be integrated into both DQA and TDQA. Empirical results of both methods are compared with prior methods to demonstrate improvement.

The thesis is organized as the follows: In section 2, we give an overview of the ATC decomposition method. In section 3, we summarize the existing methods for implementing consistency constraint relaxation in ATC. In section 4, we present the process of DQA, along with its theoretical justification, and introduce TDQA. The trust region technique that can be used in DQA and TDQA is also discussed. Computational cost, solution accuracies, and throughput performance are compared among different methods in section 5, followed by a discussion in section 6 and conclusion in section 7.

II. Overview of ATC

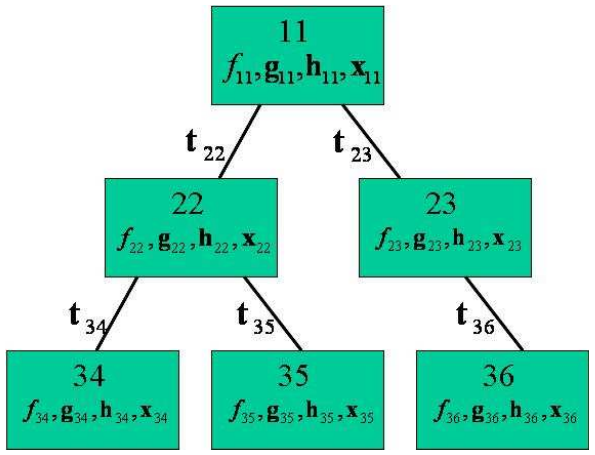
A general procedure of the ATC decomposition process is described in this section.

2.1 System Structure

ATC is applicable for problems that have the following properties: First, the problem has 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 [14]. 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. Second, the problem has a dual block angular structure, or is quasiseparable, meaning that the subproblems are nearly separable except for a few linking variables, as shown in Figure 1. Specifically, a parent and a child are connected by a target variable, which represents a design specification that the parent imposes on its child. This 'target variable' may or may not be a variable in the original pre-decomposition formulation. It is the response of the subsystem that has impact on the system performance and it is usually the output of the subsystem engineering simulation function. The effects of subsystem response on system behavior is what prevents subsystems from being designed independent.

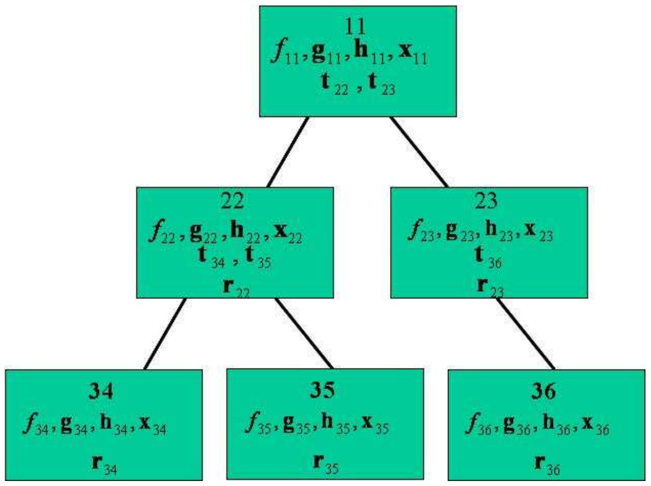
2.2 Notation

Different notations are used in describing and defining ATC, depending on the application [6], [13], [14], [18]. In this thesis, we adopt the notational system of Tosserams *et al.* [18] 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



11
$f_{11}, \mathbf{g}_{11}, \mathbf{h}_{11}, \mathbf{x}_{11}$
$\mathbf{t}_{22}$
$\mathbf{t}_{23}$
22
$f_{22}, \mathbf{g}_{22}, \mathbf{h}_{22}, \mathbf{x}_{22}$
23
$f_{23}, \mathbf{g}_{23}, \mathbf{h}_{23}, \mathbf{x}_{23}$
$\mathbf{t}_{34}$
$\mathbf{t}_{35}$
$\mathbf{t}_{36}$
34
$f_{34}, \mathbf{g}_{34}, \mathbf{h}_{34}, \mathbf{x}_{34}$
35
$f_{35}, \mathbf{g}_{35}, \mathbf{h}_{35}, \mathbf{x}_{35}$
36
$f_{36}, \mathbf{g}_{36}, \mathbf{h}_{36}, \mathbf{x}_{36}$

Figure 1 Hierarchical problem structure and variable allocation for ATC



11
$f_{11}, \mathbf{g}_{11}, \mathbf{h}_{11}, \mathbf{x}_{11}$
$\mathbf{t}_{22}, \mathbf{t}_{23}$
22
$f_{22}, \mathbf{g}_{22}, \mathbf{h}_{22}, \mathbf{x}_{22}$
$\mathbf{t}_{34}, \mathbf{t}_{35}$
$\mathbf{r}_{22}$
23
$f_{23}, \mathbf{g}_{23}, \mathbf{h}_{23}, \mathbf{x}_{23}$
$\mathbf{t}_{36}$
$\mathbf{r}_{23}$
34
$f_{34}, \mathbf{g}_{34}, \mathbf{h}_{34}, \mathbf{x}_{34}$
$\mathbf{r}_{34}$
35
$f_{35}, \mathbf{g}_{35}, \mathbf{h}_{35}, \mathbf{x}_{35}$
$\mathbf{r}_{35}$
36
$f_{36}, \mathbf{g}_{36}, \mathbf{h}_{36}, \mathbf{x}_{36}$
$\mathbf{r}_{36}$

Figure 2 Variable allocation for ATC after introducing response copies

objective function, and $\mathbf{g}_{ij}$ and $\mathbf{h}_{ij}$ are the inequality and equality constraints, respectively. Local variables of element ij are denoted by $\mathbf{x}_{ij}$. $\mathbf{t}_{ij}$ is the target variable of element ij given by its parent, and it can be determined by an analysis model $\mathbf{a}_{ij}$. Generally speaking, this model is an engineering simulation or a set of equations predicting the behavior of the subsystem. ε_i is the set of elements at level i, and $\varsigma_{ij} = \{k_1, ..., k_{c_{ij}}\}$ is the set of children of element ij, where c_{ij} is the number of children of this element. The system in Figure 1 is shown corresponding to this notation.

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:

$$
\begin{aligned}
\min_{\overline{\mathbf{x}}_{11},\ldots,\overline{\mathbf{x}}_{NM}} f(\overline{\mathbf{x}}_{11},\ldots,\overline{\mathbf{x}}_{NM}) &= \sum_{i=1}^{N}\sum_{j\in\varepsilon_i} f_{ij}(\overline{\mathbf{x}}_{ij}) \\
\text{s.t.}\quad \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) &\leq \mathbf{0} \\
\mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) &= \mathbf{0} \\
\mathbf{t}_{ij} &= \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\
\text{where}\quad \overline{\mathbf{x}}_{ij} &= [\mathbf{x}_{ij}, \mathbf{t}_{(i+1)k_1}, \ldots, \mathbf{t}_{(i+1)k_{c_{ij}}}], \quad \forall j \in \varepsilon_i\,, i = 1, \ldots, N
\end{aligned}
\tag{1}
$$

As discussed above, a parent element shares target variables with each of its children, which prevents the objective function and the constraint sets to be fully separable. In order to separate the set of variables governed by each subsystem, response variable copies $\mathbf{r}_{ij}$, corresponding to the target variables $\mathbf{t}_{ij}$ and governed by the analysis function $\mathbf{a}_{ij}$, are introduced. Figure 2 shows the system structure and variable allocations after introducing the response variables. In addition, to ensure target/response consistency, we need the following *consistency constraint*:

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

We rewrite the problem as:

$$
\begin{aligned}
\min_{\overline{\mathbf{x}}_{11},\ldots,\overline{\mathbf{x}}_{NM}} \sum_{i=1}^{N}\sum_{j\in\varepsilon_i} f_{ij}(\overline{\mathbf{x}}_{ij}) & \\
\text{s.t.} \quad \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) &\leq \mathbf{0} \\
\mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) &= \mathbf{0} \\
\theta_{ij} &= \mathbf{t}_{ij} - \mathbf{r}_{ij} = \mathbf{0} \\
\mathbf{r}_{ij} &= \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\
\text{where} \quad \overline{\mathbf{x}}_{ij} &= [\mathbf{x}_{ij}, \mathbf{t}_{(i+1)k_1}, \ldots, \mathbf{t}_{(i+1)k_{c_{ij}}}], \quad \forall j \in \varepsilon_i\,, i = 1, \ldots, N \qquad (3)
\end{aligned}
$$

Note that the problem is almost separable except for the *consistency constraints* θ_{ij}. For decomposition purposes, the constraint is relaxed by an *consistency constraint relaxation function* π, resulting in the *relaxed AIO problem*:

$$
\begin{aligned}
\min_{\overline{\mathbf{x}}_{11},\ldots,\overline{\mathbf{x}}_{NM}} \sum_{i=1}^{N}\sum_{j\in\varepsilon_i} f_{ij}(\overline{\mathbf{x}}_{ij}) &+ \sum_{i=2}^{N}\sum_{j\in\varepsilon_i} \pi(\mathbf{t}_{ij} - \mathbf{r}_{ij}) \\
\text{s.t.} \quad \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) &\leq \mathbf{0} \\
\mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) &= \mathbf{0} \\
\mathbf{r}_{ij} &= \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\
\text{where} \quad \overline{\mathbf{x}}_{ij} &= [\mathbf{x}_{ij}, \mathbf{t}_{(i+1)k_1}, \ldots, \mathbf{t}_{(i+1)k_{c_{ij}}}], \quad \forall j \in \varepsilon_i\,, i = 1, \ldots, N \qquad (4)
\end{aligned}
$$

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

$$\begin{aligned}
\min_{\overline{\mathbf{x}}_{ij}} f_{ij}(\overline{\mathbf{x}}_{ij}) \quad &+ \quad \pi(\mathbf{t}_{ij} - \mathbf{r}_{ij}) + \sum_{k=1}^{c_{ij}} \pi(\mathbf{t}_{(i+1)k} - \mathbf{r}_{(i+1)k}) \\
\text{s.t.} \quad \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) \quad &\leq \quad \mathbf{0} \\
\mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) \quad &= \quad \mathbf{0} \\
\mathbf{r}_{ij} \quad &= \quad \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\
\text{where} \quad \overline{\mathbf{x}}_{ij} \quad &= \quad [\mathbf{x}_{ij}, \mathbf{t}_{(i+1)k_1}, ..., \mathbf{t}_{(i+1)k_{c_{ij}}}]
\end{aligned} \tag{5}$$

Note that in the above formulation, the variables $\mathbf{t}_{ij}$ and $\mathbf{r}_{(i+1)k}$ for $k = 1, ..., c_{ij}$ are constants with respect to element ij. 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 sequencially and iteratively solving each subproblem as specified in (5) in any cyclic order, convergence is guaranteed. This algorithm is called the *block coordinate descent*, and the convergence result is applicable for any general relaxation consistency function π. The following theorem summarizes the convergence result of block coordinate descent and the proof is given in [3].

Theorem 1 *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 $k = 1, ..., c_{ij}$. Supposed that for all i and j, f_{ij} is continuously differentiable and the constraint sets are independent, closed and convex. Furthermore, supposed that $\overline{\mathbf{x}}_{ij}^{\kappa}$ is uniquely attained for all iterations. Let $\{\overline{\mathbf{x}}_{ij}^{\kappa}\}$ be the sequence generated by the block coordinate descent method. Then, every limit point of $\{\overline{\mathbf{x}}_{ij}^{\kappa}\}$ is a stationary point.*

In the theorem above, independence of the constraint sets is defined as the following:

Definition 1 *Consider the following problem:*

$$\min_{\mathbf{x}} f(\mathbf{x}) = \sum_{i=1}^{M} f_i(\mathbf{x}_i) \text{ s.t. } \mathbf{x} \in \mathbf{X}$$

where $\mathbf{X}$ *is closed and convex. The constraint sets of the problem is independent if* $\mathbf{X}$ *is a Cartesian product of closed convex sets* $\mathbf{X}_i$, $\forall i = 1, ..., M$*:* $\mathbf{X} = \mathbf{X}_1 \times \mathbf{X}_2 \times ... \times \mathbf{X}_M$ *where* $\mathbf{x}_i \in \mathbf{X}_i \forall i = 1, ..., M$*.*

In ATC literature, non-separable subproblems are handled in nested loop coordination schemes that are carried out in a bi-level fasion, and the convergence proof for ATC [14] is based on the nested schemes. Figure 3 shows two possible 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 relaxation function π has been approximated in three ways: a quadratic penalty function [12], an ordinary Lagrangian function [9], or an augmented Lagrangian function [8], [18]. 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 [17]. These methods will be discussed in more detail in the following sections.

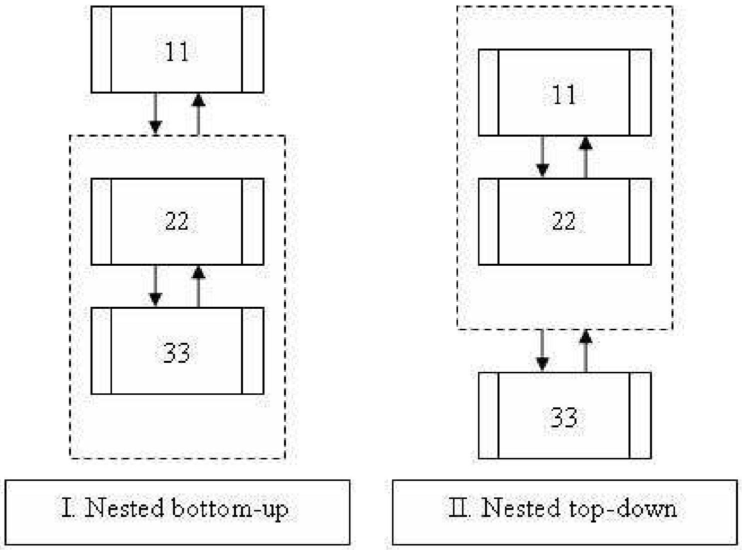


Figure 3   Nested coordination schemes for convergence of the ATC inner loop

III. Prior Consistency Constraint Relaxation Methods for ATC

3.1 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 relaxation constraint and add to the objective function a penalty term that prescribes a high cost to infeasible points. Associated with this term is a penalty parameter $\mathbf{w} = [\mathbf{w}_{ij}, \forall i, j]$ that determines 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:

$$\pi_{\mathbf{Q}}(\theta_{ij}) = ||\mathbf{w}_{ij} \circ \theta_{ij}||_2^2 \tag{6}$$

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

$$\begin{aligned} \min_{\overline{\mathbf{x}}_{11},\ldots,\overline{\mathbf{x}}_{NM}} \sum_{i=1}^{N}\sum_{j\in\varepsilon_i} f_{ij}(\overline{\mathbf{x}}_{ij}) \quad &+ \quad \sum_{i=2}^{N}\sum_{j\in\varepsilon_i} ||\mathbf{w}_{ij} \circ (\mathbf{t}_{ij} - \mathbf{r}_{ij})||_2^2 \\ \text{s.t.} \quad \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) \quad &\leq \quad \mathbf{0} \\ \mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) \quad &= \quad \mathbf{0} \\ \mathbf{r}_{ij} \quad &= \quad \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\ \text{where} \quad \overline{\mathbf{x}}_{ij} \quad &= \quad [\mathbf{x}_{ij}, \mathbf{t}_{(i+1)k_1}, \ldots, \mathbf{t}_{(i+1)k_{c_{ij}}}], \quad \forall j \in \varepsilon_i\,, i = 1, \ldots, N \end{aligned} \tag{7}$$

Proper selection of weighting coefficients heavily impacts the performance of this method. Generally speaking, as $\mathbf{w}_{ij}$ becomes larger the resulting inconsistency between $\mathbf{t}_{ij}$ and $\mathbf{r}_{ij}$

pairs decrease. Convergence properties of the quadratic penalty method has been studied thoroughly and can be shown as the following theorem. Readers are referred to [15] for more details on proofs.

Theorem 2 *Suppose that* $\{\overline{\mathbf{x}}_{ij}^{\kappa}\}$ *for all* i *and* j *is the sequence of exact global minimizers of (7) for each fixed* $\mathbf{w}^{\kappa}$*. Then as* $\mathbf{w}^{\kappa} \rightarrow \infty$*, every limit point of* $\{\overline{\mathbf{x}}^{\kappa}\}$ *is a solution of the problem (3).*

For each subproblem, we have:

$$\begin{aligned}
\min_{\overline{\mathbf{x}}_{ij}} f_{ij}(\overline{\mathbf{x}}_{ij}) \quad &+ \quad ||\mathbf{w}_{ij} \circ (\mathbf{t}_{ij} - \mathbf{r}_{ij})||_2^2 + \sum_{k=1}^{c_{ij}} ||\mathbf{w}_{(i+1)k} \circ (\mathbf{t}_{(i+1)k} - \mathbf{r}_{(i+1)k})||_2^2 \\
\text{s.t.} \quad \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) \quad &\leq \quad \mathbf{0} \\
\mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) \quad &= \quad \mathbf{0} \\
\mathbf{r}_{ij} \quad &= \quad \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\
\text{where} \quad \overline{\mathbf{x}}_{ij} \quad &= \quad [\mathbf{x}_{ij}, \mathbf{t}_{(i+1)k_1}, ..., \mathbf{t}_{(i+1)k_{c_{ij}}}] \qquad (8)
\end{aligned}$$

With this formulation, constraint sets are separable, but the objective function is not. So we need to make use of several inner loops that implement a nested block coordinate descent method, iteratively carried out in a bi-level fashion throughout the hierarchical chain for fixed $\mathbf{w}$. Upon convergence of all inner loops, an outer loop is used to update $\mathbf{w}$. It is noted in [12] that while any positive $\mathbf{w}_{ij}$ can ensure convergence to a consistent solution for problems that have attainable targets, no finite $\mathbf{w}_{ij}$ will lead to perfect consistency for problems with unattainable targets. Prior approaches are either setting $\mathbf{w}_{ij}$ through trial and error, or initializing it to a small value then increasing its value by a linear update scheme, i.e., multiplying the current weight by a constant. However, the trial and error approach can be difficult for large-scale problems. Michalek *et al.* [12] proposed an efficient weight update method (WUM) for finding values of $\mathbf{w}_{ij}$ that achieve solutions within user-specified inconsistency tolerances. It is demonstrated that WUM achieves better performance than a constant $\mathbf{w}_{ij}$

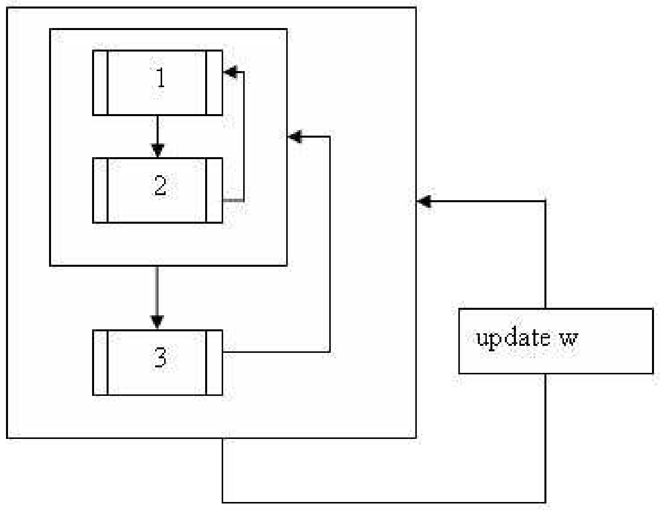


Figure 4 Quadratic penalty method flow chart

Figure 4 depicts the flow chart of the quadratic penalty (QP) method using the top-down scheme for a 3-level system. The first two levels are solved until convergence as an inner loop, then inner loop and third level are iterated until convergence. If the consistency deviation limit is not met, **w** is updated using WUM and the process is repeated. Although we follow the earlier ATC literature and use a nested coordination scheme, block coordinate descent can also be used.

In practice, large penalty weight can cause ill-conditioning and the quadratic term prevents each subproblem from being separable, preventing the sue of parallel computing.

3.2 The Ordinary Lagrangian Method

An alternative choice for the constraint relaxation function π, proposed by Lassiter *et. al*, is an ordinary Lagrangian function [9]. This method (OL) is based on Lagrangian duality theory [3], [15]. Let $\lambda = [\lambda_{ij}, \forall i, j]$ be the vector of Lagrangian multipliers. The consistency constraint relaxation function is specified as:

$$\pi_L(\theta_{ij}) = \lambda_{ij}^T \theta_{ij} \tag{9}$$

With π_L, (4) can be rewritten as:

$$\begin{aligned}
\mathbf{\Lambda}(\lambda) = \min_{\overline{\mathbf{x}}_{11},\ldots,\overline{\mathbf{x}}_{NM}} \sum_{i=1}^{N}\sum_{j\in\varepsilon_i} f_{ij}(\overline{\mathbf{x}}_{ij}) &\;+\; \sum_{i=2}^{N}\sum_{j\in\varepsilon_i} \lambda_{ij}^T(\mathbf{t}_{ij} - \mathbf{r}_{ij}) \\
\text{s.t.}\quad \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) &\leq \mathbf{0} \\
\mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) &= \mathbf{0} \\
\mathbf{r}_{ij} &= \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\
\text{where}\quad \overline{\mathbf{x}}_{ij} &= [\mathbf{x}_{ij}, \mathbf{t}_{(i+1)k_1}, \ldots, \mathbf{t}_{(i+1)k_{c_{ij}}}], \quad \forall j \in \varepsilon_i\,, i = 1, \ldots, N
\end{aligned} \tag{10}$$

And the dual problem for (10) is:

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

Under convexity assumptions, the strong duality theorem holds and so there is no duality gap. As a result, an iterative process can be carried out by first solving (10) with a fixed λ, then update λ until convergence. In [3], the following theorems are presented.

Theorem 3 (Strong Duality Theorem for Linear Constraints) *Let the objective functions f_{ij} of problem (10) be convex and let the constraint sets be polyhedral. Assume that the optimal value*

$$f^* = \sum_{i=1}^{N}\sum_{j\in\varepsilon_i} f_{ij}^*$$

is finite. Then there is no duality gap and there exists at least one Lagrangian multiplier.

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

For each subsystem, consider a fixed λ_{ij}, we have:

$$\begin{aligned}
\min_{\overline{\mathbf{x}}_{ij}} f_{ij}(\overline{\mathbf{x}}_{ij}) &+ \lambda_{ij}^T(\mathbf{t}_{ij} - \mathbf{r}_{ij}) + \sum_{k=1}^{c_{ij}} \lambda_{(i+1)k}^T(\mathbf{t}_{(i+1)k} - \mathbf{r}_{(i+1)k}) \\
\text{s.t.} \quad \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) &\leq \mathbf{0} \\
\mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) &= \mathbf{0} \\
\mathbf{r}_{ij} &= \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\
\text{where} \quad \overline{\mathbf{x}}_{ij} &= [\mathbf{x}_{ij}, \mathbf{t}_{(i+1)k_1}, ..., \mathbf{t}_{(i+1)k_{c_{ij}}}]
\end{aligned} \tag{12}$$

Note that since $\lambda_{ij}^T \mathbf{t}_{ij}$ and $\lambda_{(i+1)k}^T \mathbf{r}_{(i+1)k}$ for all k are constants, we can drop them from the objective function, resulting in the following subproblem:

$$\begin{aligned}
\min_{\overline{\mathbf{x}}_{ij}} f_{ij}(\overline{\mathbf{x}}_{ij}) &+ \lambda_{ij}^T(-\mathbf{r}_{ij}) + \sum_{k=1}^{c_{ij}} \lambda_{(i+1)k}^T \mathbf{t}_{(i+1)k} \\
\text{s.t.} \quad \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) &\leq \mathbf{0} \\
\mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) &= \mathbf{0} \\
\mathbf{r}_{ij} &= \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\
\text{where} \quad \overline{\mathbf{x}}_{ij} &= [\mathbf{x}_{ij}, \mathbf{t}_{(i+1)k_1}, ..., \mathbf{t}_{(i+1)k_{c_{ij}}}]
\end{aligned} \tag{13}$$

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

$$\lambda^{\kappa+1} = \lambda^{\kappa} + \tau^{\kappa}\theta^{\kappa} = \lambda^{\kappa} + \tau^{\kappa}(\mathbf{t}^{\kappa} - \mathbf{r}^{\kappa}) \tag{14}$$

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 some assumptions shown in the following theorem [9].

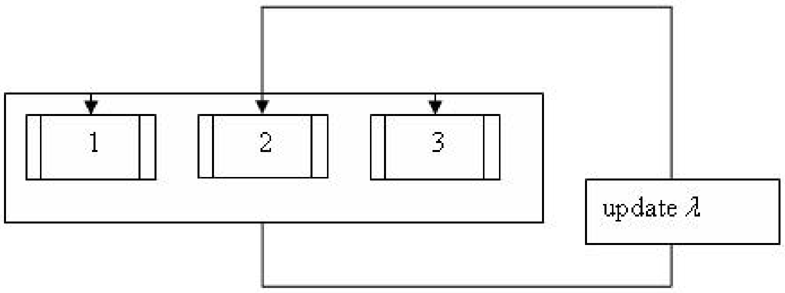


Figure 5 Ordinary Lagrangian method flow chart

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 $\mathbf{\Lambda}(\lambda^{\kappa^*}) = \mathbf{\Lambda}(\lambda^*)$, or else an infinite sequence of iterates is generated such that $\mathbf{\Lambda}(\lambda^{\kappa^*}) \to \mathbf{\Lambda}(\lambda^*)$ as $\kappa \to \infty$.

The iterative process using the ordinary Lagrangian function as the consistency constraint relaxation function is depicted in figure 5.

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 [17], which limits application of the method for practical problems.

3.3 The Augmented Lagrangian Method

An improved method, the augmented Lagrangian method (AL) applied to ATC formulations by Tosserams *et. al* [18] combines both the quadratic penalty term and the Lagrangian term, which overcomes the drawbacks of both QP and OL. The Lagrangian term is used to avoid ill-conditioning, since optimal solution can be obtained for smaller weights. The quadratic term convexifies the problem to avoid duality gaps. The augmented Lagrangian function is defined as the following:

$$\pi_{\mathbf{AL}}(\theta_{ij}) = \lambda_{ij}^T\theta_{ij} + ||\mathbf{w}_{ij} \circ \theta_{ij}||_2^2 \tag{15}$$

With π_{AL}, the general problem is:

$$\begin{aligned}
\hat{\mathbf{\Lambda}}(\lambda) = \min_{\overline{\mathbf{x}}_{11},\ldots,\overline{\mathbf{x}}_{NM}} \sum_{i=1}^{N}\sum_{j\in\varepsilon_i} f_{ij}(\overline{\mathbf{x}}_{ij}) \quad &+ \quad \sum_{i=2}^{N}\sum_{j\in\varepsilon_i}[\lambda_{ij}^T(\mathbf{t}_{ij}-\mathbf{r}_{ij}) + ||\mathbf{w}_{ij}\circ(\mathbf{t}_{ij}-\mathbf{r}_{ij})||_2^2] \\
\text{s.t.} \quad \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) \quad &\leq \quad \mathbf{0} \\
\mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) \quad &= \quad \mathbf{0} \\
\mathbf{r}_{ij} \quad &= \quad \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\
\text{where} \quad \overline{\mathbf{x}}_{ij} \quad &= \quad [\mathbf{x}_{ij}, \mathbf{t}_{(i+1)k_1}, \ldots, \mathbf{t}_{(i+1)k_{c_{ij}}}], \quad \forall j\in\varepsilon_i\,, i=1,\ldots,N
\end{aligned} \tag{16}$$

The dual problem for (16) is:

$$\max_{\lambda}\hat{\mathbf{\Lambda}}(\lambda) \tag{17}$$

The following theorem is the basic convergence result of the augmented Lagrangian function.

Theorem 6 *Assume that the objective and constraint functions are continuous, the constraint sets are closed, and that the set* $\{\overline{\mathbf{x}} : \mathbf{h}(\overline{\mathbf{x}}) = 0\}$ *is nonempty. For* $\kappa = 0, 1, \ldots$, *let* $\overline{\mathbf{x}}^\kappa$ *be the local minimum of the problem* $\hat{\mathbf{\Lambda}}(\lambda^\kappa)$ *with* $\mathbf{w}^\kappa$, *where* λ^κ *is bounded,* $0 \leq \mathbf{w}^\kappa \leq \mathbf{w}^{(\kappa+1)}$ *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 the original problem.*

For each subproblem, we have:

$$
\begin{aligned}
\min_{\overline{\mathbf{x}}_{ij}} f_{ij}(\overline{\mathbf{x}}_{ij}) \quad &+ \quad \lambda_{ij}^T(-\mathbf{r}_{ij}) + ||\mathbf{w}_{ij} \circ (\mathbf{t}_{ij} - \mathbf{r}_{ij})||_2^2 + \sum_{k=1}^{c_{ij}} [\lambda_{(i+1)k}^T \mathbf{t}_{(i+1)k} + ||\mathbf{w}_{(i+1)k} \circ (\mathbf{t}_{(i+1)k} - \mathbf{r}_{(i+1)k})||_2^2] \\
\text{s.t.} \quad \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) \quad &\leq \quad \mathbf{0} \\
\mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) \quad &= \quad \mathbf{0} \\
\mathbf{r}_{ij} \quad &= \quad \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\
\text{where} \quad \overline{\mathbf{x}}_{ij} \quad &= \quad [\mathbf{x}_{ij}, \mathbf{t}_{(i+1)k_1}, ..., \mathbf{t}_{(i+1)k_{c_{ij}}}], \quad \forall j \in \varepsilon_i \, , i = 1, ..., N \qquad (18)
\end{aligned}
$$

Duality theory also applies to the augmented Lagrangian formulation so we can use the same procedure as OL. However, the subproblems as shown in (18) are not separable due to the quadratic penalty term. As a result, instead of solving all subproblems in parallel as in OL, the nested block coordinate descent method is carried out as an inner loop for the AL relaxation formulation, similar to the QP approach[1]. In order to achieve convergence, we must 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:

$$\lambda^{(\kappa+1)} = \lambda^{\kappa} + 2\mathbf{w}^{\kappa} \circ \mathbf{w}^{\kappa} \circ \theta^{\kappa} \qquad (19)$$

Convergence properties of the method of multipliers are presented as the following theorem [3]:

Theorem 7 *Assume that the objective function $f : \Re^n \rightarrow \Re$ and equality constraint functions $\mathbf{h} : \Re^n \rightarrow \Re^m$ are continously differentiable. For $\kappa = 0, 1, ...,$ let*

[1]Note that block coordinate descent method can also be used, but we follow the prior ATC literature to use the nested scheme.

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

where $\overline{\mathbf{x}} = [\overline{\mathbf{x}}_{11}, ..., \overline{\mathbf{x}}_{NM}]$ *and* $\lambda = [\lambda_{11}, ..., \lambda_{NM}]$,

and let

$\overline{\mathbf{x}}$ *satisfy*

$$||\nabla_x f_{AL}(\overline{\mathbf{x}}^{\kappa}, \lambda^{\kappa})|| \leq \sigma^{\kappa}$$

with penalty weight $\mathbf{w}^{\kappa}$, *where* $\{\lambda^{\kappa}\}$ *is bounded, and* $\{\sigma^{\kappa}\}$ *and* $\{\mathbf{w}^{\kappa}\}$ *satisfy:*

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

$$0 \leq \sigma^{\kappa} \; \forall \kappa, \; \sigma^{\kappa} \to 0$$

Assume that a subsequence $\{\overline{\mathbf{x}}^{\kappa}\}$ *converges to a vector* $\overline{\mathbf{x}}^*$ *such that* $\nabla \mathbf{h}(\overline{\mathbf{x}}^*)$ *has rank* m, *then*

$$\{\lambda^{\kappa} + \mathbf{w}^{\kappa} \mathbf{h}(\overline{\mathbf{x}}^{\kappa})\} \to \lambda^*$$

where λ^* *is a vector satisfying, together with* $\overline{\mathbf{x}}^*$, *the first necessary conditions*

$$\nabla f(\overline{\mathbf{x}}^*) + \nabla \mathbf{h}(\overline{\mathbf{x}}^*)\lambda^* = 0$$

,

$$\mathbf{h}(\overline{\mathbf{x}}^*) = \mathbf{0}$$

.

Note that in ATC, the equality constraint in the above theorem $\mathbf{h}$ cooresponds to the consistency constraint function $\theta = \mathbf{t} - \mathbf{r}$.

There are other ways of updating λ when using the augmented Lagrangian method. For example, the subgradient update scheme is used in [8].

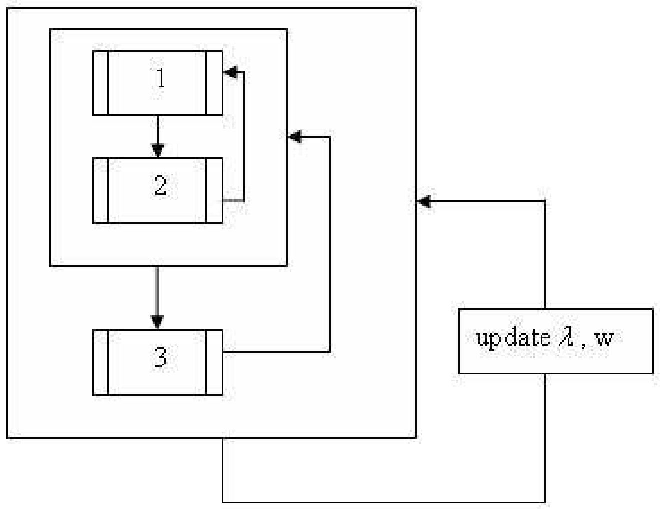


Figure 6 Augmented Lagrangian method flow chart

In AL, the penalty weight $\mathbf{w}$ is also updated. A linear scheme with constant scale β for updating $\mathbf{w}$ is used in [18]:

$$\mathbf{w}^{(\kappa+\mathbf{1})} = \beta\mathbf{w}^{\kappa}, \beta \geq 1 \tag{20}$$

The augmented Lagrangian function approach is depicted in figure 6. Notice that the quadratic term prevents subproblems from being separable, so the top-down coordination scheme is used in the inner loop for finding the optimal solution with respect to fixed λ and $\mathbf{w}$, and the outer loop applies (19) and (20) for updating the Lagrangian 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.

Although the augmented Lagrangian approach shows very stable convergence property, the inner loop coordination scheme still induces large computational cost.

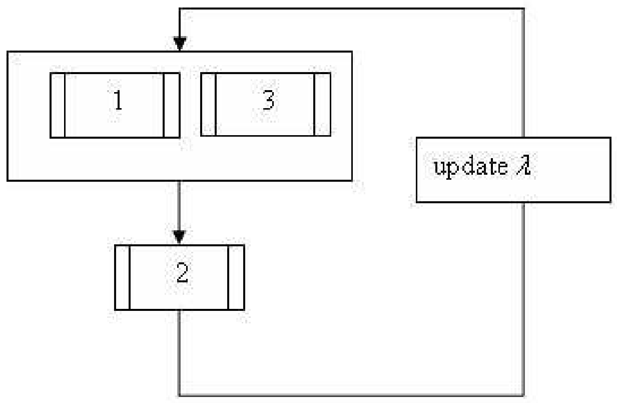


Figure 7 Augmented Lagrangian with alternating directions method flow chart

3.4 The Augmented Lagrangian with Alternating Direction Method of Multipliers

When using the augmented Lagrangian function, subproblems are not separable, thus introducing the large computational cost of nested loops. To reduce computational cost, Tosserams *et. al* [18] 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 [4]. In [18], Tosserams *et. al* keep a constant $\mathbf{w}$. Furthermore, the inner loop is solved for only one iteration and the process is shown in Figure 7. 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 feasibility sets are nonempty and the constraint sets are bounded [4]. This method is promising since it demonstrates both good convergence properties and low computational cost [18].

Besides the ordinary Lagrangian approach, all other prior methods for consistency constraint relaxation prevent 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.

IV. Proposed Consistency Constraint Relaxation Methods for ATC

4.1 The Diagonal Quadratic Approximation Method

As discussed previously, all existing methods have drawbacks. To overcome these drawbacks, we apply the diagonal quadratic approximation (DQA) originally proposed in [16], where it is used to solve block-angular structure problems. Proofs on convergence and convergence rate are available for this method [16]. There are other variations of solving similar problems using approximation techniques, for example, the method discussed in [17].

In the original paper, DQA is used to solve general dual block-angular structured problems in the following form:

$$\min_{\mathbf{x}}[f(\mathbf{x}) = \sum_{i=1}^{L} f_i(\mathbf{x}_i)] \tag{21}$$

$$\sum_{i=1}^{L} \mathbf{A}_i\mathbf{x}_i = \mathbf{b} \tag{22}$$

$$\mathbf{x}_i \in X_i, i = 1, 2, ..., L \tag{23}$$

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 (22). Third, the inequality and equality constraints $\mathbf{g}$ and $\mathbf{h}$, now fully separable, forms sets X_i. The 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 (16).

$$||\mathbf{t}_{ij} - \mathbf{r}_{ij}||_2^2 = \mathbf{t}_{ij} \circ \mathbf{t}_{ij} + \mathbf{r}_{ij} \circ \mathbf{r}_{ij} - 2(\mathbf{t}_{ij} \circ \mathbf{r}_{ij}) \tag{24}$$

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

$$\begin{aligned} \mathbf{t}_{ij} \circ \mathbf{r}_{ij} &\cong \mathbf{t}^*_{ij} \circ \mathbf{r}^*_{ij} + \mathbf{r}^*_{ij} \circ (\mathbf{t}_{ij} - \mathbf{t}^*_{ij}) + \mathbf{t}^*_{ij} \circ (\mathbf{r}_{ij} - \mathbf{r}^*_{ij}) \\ &= \mathbf{r}^*_{ij} \circ \mathbf{t}_{ij} + \mathbf{t}^*_{ij} \circ \mathbf{r}_{ij} - \mathbf{t}^*_{ij} \circ \mathbf{r}^*_{ij} \qquad (25) \end{aligned}$$

Combining (24) and (25), we have:

$$\begin{aligned} ||\mathbf{t}_{ij} - \mathbf{r}_{ij}||_2^2 &\cong \mathbf{t}_{ij} \circ \mathbf{t}_{ij} + \mathbf{r}_{ij} \circ \mathbf{r}_{ij} - 2(\mathbf{r}^*_{ij} \circ \mathbf{t}_{ij} + \mathbf{t}^*_{ij} \circ \mathbf{r}_{ij} - \mathbf{t}^*_{ij} \circ \mathbf{r}^*_{ij}) \\ &= [\mathbf{t}^*_{ij} \circ \mathbf{t}^*_{ij} + \mathbf{r}_{ij} \circ \mathbf{r}_{ij} - 2(\mathbf{t}^*_{ij} \circ \mathbf{r}_{ij})] + [\mathbf{t}_{ij} \circ \mathbf{t}_{ij} + \mathbf{r}^*_{ij} \circ \mathbf{r}^*_{ij} - 2(\mathbf{t}_{ij} \circ \mathbf{r}^*_{ij})] \\ &= ||\mathbf{t}^*_{ij} - \mathbf{r}_{ij}||_2^2 + ||\mathbf{t}_{ij} - \mathbf{r}^*_{ij}||_2^2 \qquad (26) \end{aligned}$$

By substituting (26) into (16) at $\mathbf{t}^*_{ij} = \mathbf{t}^\kappa_{ij}$ and $\mathbf{r}^*_{ij} = \mathbf{r}^\kappa_{ij}$, where $\mathbf{t}^\kappa_{ij}$ and $\mathbf{r}^\kappa_{ij}$ 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:

$$\begin{aligned} \min_{\overline{\mathbf{x}}_{11},\ldots,\overline{\mathbf{x}}_{NM}} \sum_{i=1}^{N} \sum_{j\in\varepsilon_i} f_{ij}(\overline{\mathbf{x}}_{ij}) \quad &+ \quad \sum_{i=2}^{N} \sum_{j\in\varepsilon_i} [\lambda_{ij}^T(\mathbf{t}_{ij} - \mathbf{r}_{ij}) + ||\mathbf{w}_{ij} \circ (\mathbf{t}^\kappa_{ij} - \mathbf{r}_{ij})||_2^2 + ||\mathbf{w}_{ij} \circ (\mathbf{t}_{ij} - \mathbf{r}^\kappa_{ij})||_2^2] \\ \text{s.t.} \quad \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) \quad &\leq \quad \mathbf{0} \\ \mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) \quad &= \quad \mathbf{0} \\ \mathbf{r}_{ij} \quad &= \quad \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\ \text{where} \quad \overline{\mathbf{x}}_{ij} \quad &= \quad [\mathbf{x}_{ij}, \mathbf{t}_{(i+1)k_1}, ..., \mathbf{t}_{(i+1)k_{c_{ij}}}], \quad \forall j \in \varepsilon_i \,, i = 1, ..., N \qquad (27) \end{aligned}$$

And for each subproblem:

$$
\begin{aligned}
\min_{\overline{\mathbf{x}}_{ij}} f_{ij}(\overline{\mathbf{x}}_{ij}) \quad &+ \quad \lambda_{ij}^T(-\mathbf{r}_{ij}) + ||\mathbf{w}_{ij} \circ (\mathbf{t}_{ij}^{\kappa} - \mathbf{r}_{ij})||_2^2 + \sum_{k=1}^{c_{ij}} [\lambda_{(i+1)k}^T \mathbf{t}_{(i+1)k} + ||\mathbf{w}_{(i+1)k} \circ (\mathbf{t}_{(i+1)k} - \mathbf{r}_{(i+1)k}^{\kappa})||_2^2] \\
\text{s.t.} \quad \mathbf{g}_{ij}(\overline{\mathbf{x}}_{ij}) \quad &\leq \quad \mathbf{0} \\
\mathbf{h}_{ij}(\overline{\mathbf{x}}_{ij}) \quad &= \quad \mathbf{0} \\
\mathbf{r}_{ij} \quad &= \quad \mathbf{a}_{ij}(\overline{\mathbf{x}}_{ij}) \\
\text{where} \quad \overline{\mathbf{x}}_{ij} \quad &= \quad [\mathbf{x}_{ij}, \mathbf{t}_{(i+1)k_1}, ..., \mathbf{t}_{(i+1)k_{c_{ij}}}]
\end{aligned}
\tag{28}
$$

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. The general DQA method 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. 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, and $\overline{\mathbf{x}}^{\kappa}$ is the final solution of the previous $(\kappa - 1)$st outer loop iteration upon inner loop convergence.
3. for each element, solve for $\overline{\mathbf{x}}_{ij}$ in (28) 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 $\lambda^{\kappa+1} = \lambda^{\kappa} + \mathbf{w} \circ (\mathbf{t}^{\kappa} - \mathbf{r}^{\kappa})$, and go to step 2.

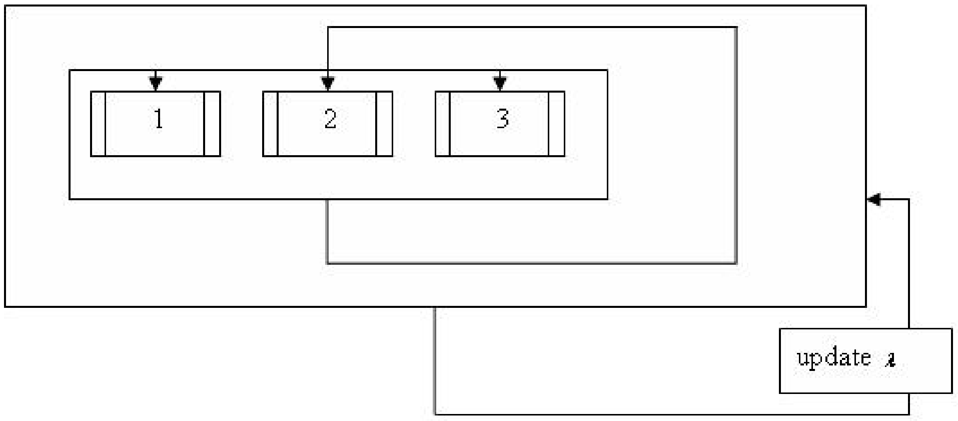


Figure 8 Diagonal quadratic approximation method flow chart

Figure 8 is the flow chart of the DQA method. An we use the same update scheme as AL to update the penalty weight $\mathbf{w}$. Also, 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 [16]. For practical purposes, however, we allow σ_{inner} to be very small but non-zero.

Theorem 8 *Suppose that the constraint sets are bounded, and that the step size τ 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* $\hat{\mathbf{\Lambda}}(\lambda^{\kappa})$.

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 rule 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 [16] 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 thesis.

Local convergence is also discussed in [16], and this property of DQA is based on the quadratic growth condition of the augmented Lagrangian function, defined as the following:

Definition 2 (Quadratic growth condition) *let*

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

,

and $\overline{\mathbf{x}}^*$ *be the solution of*

$$\min_{\overline{\mathbf{x}}} f_{AL}$$

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 is* $\gamma > 0$ *such that for every* $\overline{\mathbf{x}}$*,*

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

.

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 [16].

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.

As will be discussed in the next section, DQA performs well on test problems in terms of throughput vs solution error, 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 reach 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, same as DQA.

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 (28) 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 Lagrangian multipliers using the method of multipliers by setting $\lambda^{\kappa+1} = \lambda^{\kappa} + \mathbf{w} \circ (\mathbf{t}^{\kappa} - \mathbf{r}^{\kappa})$, and go to step 2.

Similar to DQA, we also fix the penalty weight $\mathbf{w}$ in TDQA, and the outer loop σ 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.

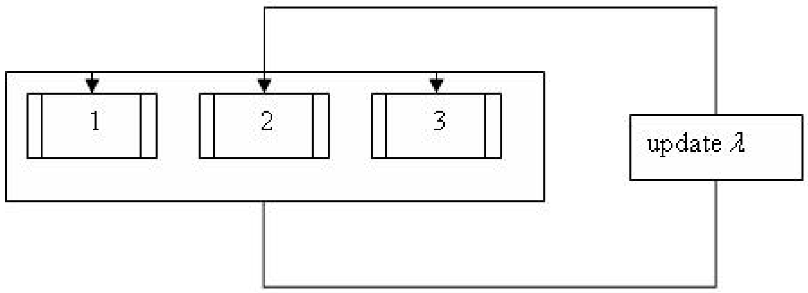


Figure 9 Truncated diagonal quadratic approximation method flow chart

Again, empirical results show similar performance as linearization at end points. Figure 9 shows the process of the TDQA method.

Empirical results show promising results of the TDQA method. However, formal theoretical justification is not available and needs future work. 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 improvement 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 turns into the DQA method and convergence can be obtained.

4.3 DQA and TDQA with the Trust Region Technique

From the discussion above, a drawback of DQA and TDQA method is that it is possible to have a linearization at the current point that is a poor approximation to the augmented Lagrangian function and convergence is not guaranteed. A small step size τ can ensure accuracy of the approximation. However, if the step size is too small, convergence will be significantly slowed down. As a result, some trial and error process is required for finding a good value of the step size for practical applications.

An alternative approach for ensuring good approximation is to use a trust region. The basic idea of trust region is simple. Let m be the original function and $\bar{m}$ be the approximation of m. To determine whether the current linearization is accurate, we estimate the ratio ρ of the actual reduction to the predicted reduction between the current solution $\mathbf{x}^{new}$ and previous solution $\mathbf{x}^{old}$:

$$\rho = \frac{m(\mathbf{x}^{new}) - m(\mathbf{x}^{old})}{\bar{m}(\mathbf{x}^{new}) - \bar{m}(\mathbf{x}^{old})} \tag{29}$$

If ρ is close to one, then 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 and the solution is not accurate, and it is necessary to impose a trust region to obtain a better move.

In the ATC context, let

$$\begin{aligned} P_{ij}^{\mathrm{AL}}(\overline{\mathbf{x}}_{ij}) = f_{ij}(\overline{\mathbf{x}}_{ij}) &+ ||\mathbf{w}_{ij} \circ (\mathbf{t}_{ij} - \mathbf{r}_{ij})||_2^2 + \sum_{k \in \varsigma_{ij}} ||\mathbf{w}_{(i+1)k} \circ (\mathbf{t}_{(i+1)k} - \mathbf{r}_{(i+1)k})||_2^2 \\ &+ \lambda_{ij}^T(\mathbf{t}_{ij} - \mathbf{r}_{ij}) + \sum_{k \in \varsigma_{ij}} \lambda_{(i+1)k}^T(\mathbf{t}_{(i+1)k} - \mathbf{r}_{(i+1)k}) \end{aligned} \tag{30}$$

$$\begin{aligned} P_{ij}^{\mathrm{DQA}}(\overline{\mathbf{x}}_{ij}) = f_{ij}(\overline{\mathbf{x}}_{ij}) &+ ||\mathbf{w}_{ij} \circ (\mathbf{t}_{ij}^{\kappa} - \mathbf{r}_{ij})||_2^2 + \sum_{k \in \varsigma_{ij}} ||\mathbf{w}_{(i+1)k} \circ (\mathbf{t}_{(i+1)k} - \mathbf{r}_{(i+1)k}^{\kappa})||_2^2 \\ &- \lambda_{ij}^T\mathbf{r}_{ij} + \sum_{k \in \varsigma_{ij}} \lambda_{(i+1)k}^T\mathbf{t}_{(i+1)k} \end{aligned} \tag{31}$$

and the reduction ratio is defined as

$$\rho = \frac{P_{ij}^{AL}(\overline{\mathbf{x}}_{ij}^{\kappa+1}) - P_{ij}^{AL}(\overline{\mathbf{x}}_{ij}^{\kappa})}{P_{ij}^{DQA}(\overline{\mathbf{x}}_{ij}^{\kappa+1}) - P_{ij}^{DQA}(\overline{\mathbf{x}}_{ij}^{\kappa})} \tag{32}$$

For DQA, a trust region can be used in solving the inner loop in that the DQA formulation of the augmented Lagrangian function is the approximation function for the augmented Lagrangian function. By global convergence of the trust region method [15], the inner loop solution obtained by using the trust region method converges to the solution obtained by fully solving the augmented Lagrangian function. As a result, using the method of multipliers in the outer loop, convergence for DQA with a trust region is attained.

The TDQA algorithm with trust region 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 (28) in parallel, and obtain $\overline{\mathbf{x}}_{ij}^{\kappa+1}$
3. let ρ be specified as (32). if $\rho \geq \gamma$, go to 4. Otherwise, shrink the step size by half and check the condition. again. Keep shrinking step size bt half until $\rho \geq \gamma$ is attained. γ = 0.25 is recommended in classical trust region strategies.
4. 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 Lagrangian multipliers using the method of multipliers by setting $\lambda^{\kappa+1} = \lambda^{\kappa} + \mathbf{w} \circ (\mathbf{t}^{\kappa} - \mathbf{r}^{\kappa})$, and go to step 2.

V. Results and Comparisons

For the purpose of a clear comparison among all ATC methods, we use the four examples discussed in [18], in the same order. Readers are referred to [18] 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:

$$max(||\mathbf{t}^{\kappa} - \mathbf{t}^{\kappa-\mathbf{1}}||, ||\mathbf{r}^{\kappa} - \mathbf{r}^{\kappa-\mathbf{1}}||) \leq \sigma_2 \tag{33}$$

This is different from [18], where the termination condition is:

$$max(||(\mathbf{t}^{\kappa} - \mathbf{r}^{\kappa}) - (\mathbf{t}^{\kappa-\mathbf{1}} - \mathbf{r}^{\kappa-\mathbf{1}})||) \leq \sigma_2 \tag{34}$$

We use a new criterion since (34) might result in premature convergence [1].

In this thesis, we compare the performance of different methods in the following two ways: function evaluations vs solution error, and computational throughput vs solution error. Solution error of method C is defined as the the difference between the actual optimal solution of the problem and the solution we obtained by using C, and the magnitude of the solution error can be controlled by the convergence tolerance. The tighter the tolerance, the more accurate the solution. At any given solution error, consider the function evaluations of C, which indicates the overall computation cost for solving the problem using C, and the computational throughput, which indicates the CPU time spent on solving the problem using C

[1]Generally speaking, three convergence criteria are widely used in the implementation of optimization methods: gradient of the objective 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.

assuming available parallel computing power. All tests are done in the same machine with the same applications so comparison is fair[2].

5.1 Example 1

The first example is a two-level decomposition of the geometric programming problem.

$$\begin{aligned}
\min_{z_1,\ldots,z_7} f &= f_1 + f_2 = z_1^2 + z_2^2 \\
\text{s.t.}\quad g_1 &= (z_3^{-2} + z_4^2) z_5^{-2} - 1 \leq 0 \\
g_2 &= (z_5^2 + z_6^{-2}) z_7^{-2} - 1 \leq 0 \\
h_1 &= (z_3^2 + z_4^{-2} + z_5^2) z_1^{-2} - 1 = 0 \\
h_2 &= (z_5^2 + z_6^2 + z_7^2) z_2^{-2} - 1 = 0 \\
& \quad z_1, z_2, ..., z_7 \geq 0
\end{aligned} \tag{35}$$

Figure 10 shows the structure of the problem.

Figure 11 shows results among different methods. The left plot is function evaluations as a solution of solution accuracy. TDQA is close to but outperforms ALAD, since it uses fewer function evaluations to achieve a specific solution accuracy. DQA requires slightly larger function evaluations than AL when given a low solution accuracy, but less when given a high solution accuracy. The right plot is computational throughput as a function of solution accuracy. TDQA and DQA have better throughput compared to ALAD and AL, respectively. And QP does not perform as well as any of the other methods.

[2]Application: Matlab Version 7.0 with Tomlab NPSol Solver Version 5.3; OS: SUSE Linux; Processor: Intel(R) Xeon(TM) CPU 2.80GHz

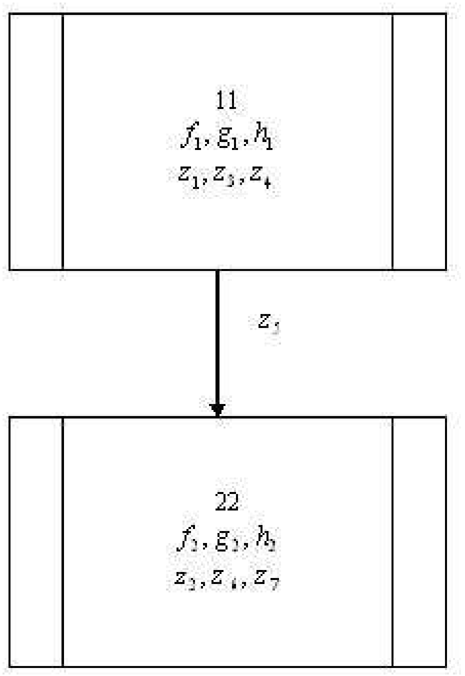


Figure 10 Example 1: Problem structure

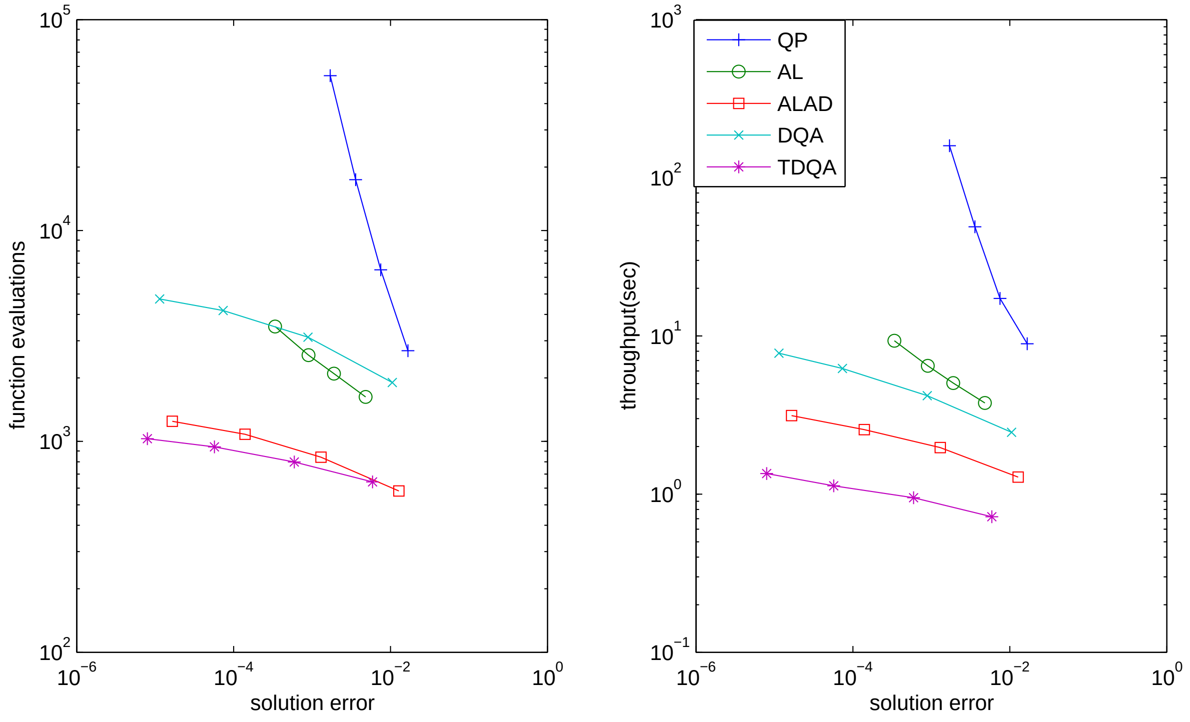


Figure 11 Example 1: Computational cost and throughput vs solution accuracy

5.2 Example 2

The second example problem is a three-level decomposition of posynomial geometric programming problem.

$$
\begin{aligned}
\min_{z_1,\ldots,z_{14}} f &= f_1 + f_2 = z_1^2 + z_2^2 \\
\text{s.t.} \quad g_1 &= (z_3^{-2} + z_4^2) z_5^{-2} - 1 \le 0 \\
g_2 &= (z_5^2 + z_6^{-2}) z_7^{-2} - 1 \le 0 \\
g_3 &= (z_8^2 + z_9^2) z_{11}^{-2} - 1 \le 0 \\
g_4 &= (z_8^{-2} + z_{10}^2) z_{11}^{-2} - 1 \le 0 \\
g_5 &= (z_{11}^2 + z_{12}^{-2}) z_{13}^{-2} - 1 \le 0 \\
g_6 &= (z_{11}^2 + z_{12}^2) z_{14}^{-2} - 1 \le 0 \\
h_1 &= (z_3^2 + z_4^{-2} + z_5^2) z_1^{-2} - 1 = 0 \\
h_2 &= (z_5^2 + z_6^2 + z_7^2) z_2^{-2} - 1 = 0 \\
h_3 &= (z_8^2 + z_9^{-2} + z_{10}^{-2} + z_{11}^2) z_3^{-2} - 1 = 0 \\
h_4 &= (z_{11}^2 + z_{12}^2 + z_{13}^2 + z_{14}^2) z_6^{-2} - 1 = 0 \\
z_1, z_2, \ldots, z_{14} &\ge 0 \qquad (36)
\end{aligned}
$$

Figure 12 shows the structure of the problem.

In this example, DQA and TDQA behave similarly as in Example 1. The differences are that in terms of function evaluations, DQA has a lower number than AL while obtaining a more accurate solution. And in terms of computational throughput, since each subproblem is more difficult than example 1, parallelization dramatically reduces throughput, so DQA and TDQA outperforms AL and ALAD in orders of magnitude. QP does not perform as well as any other methods.

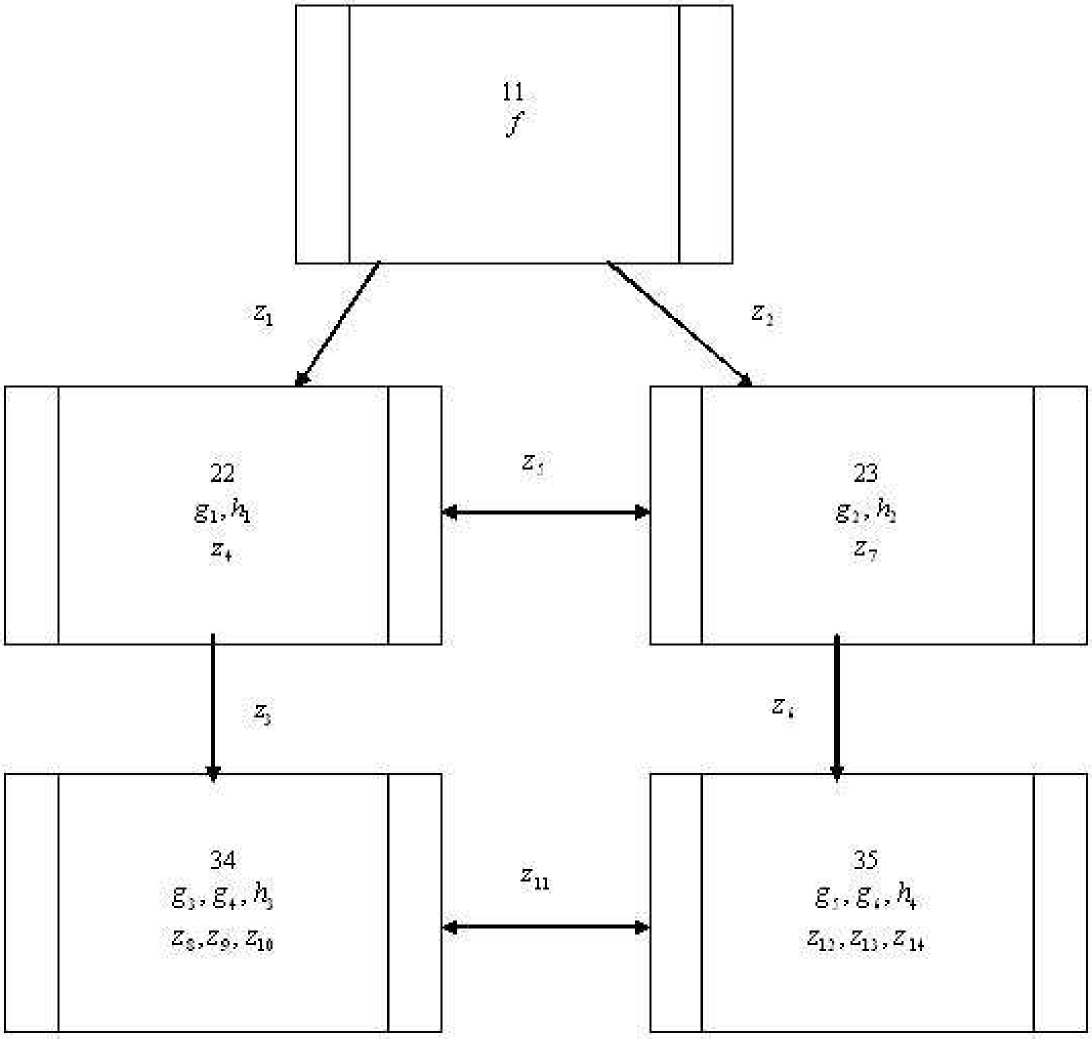


Figure 12 Example 2: Problem structure

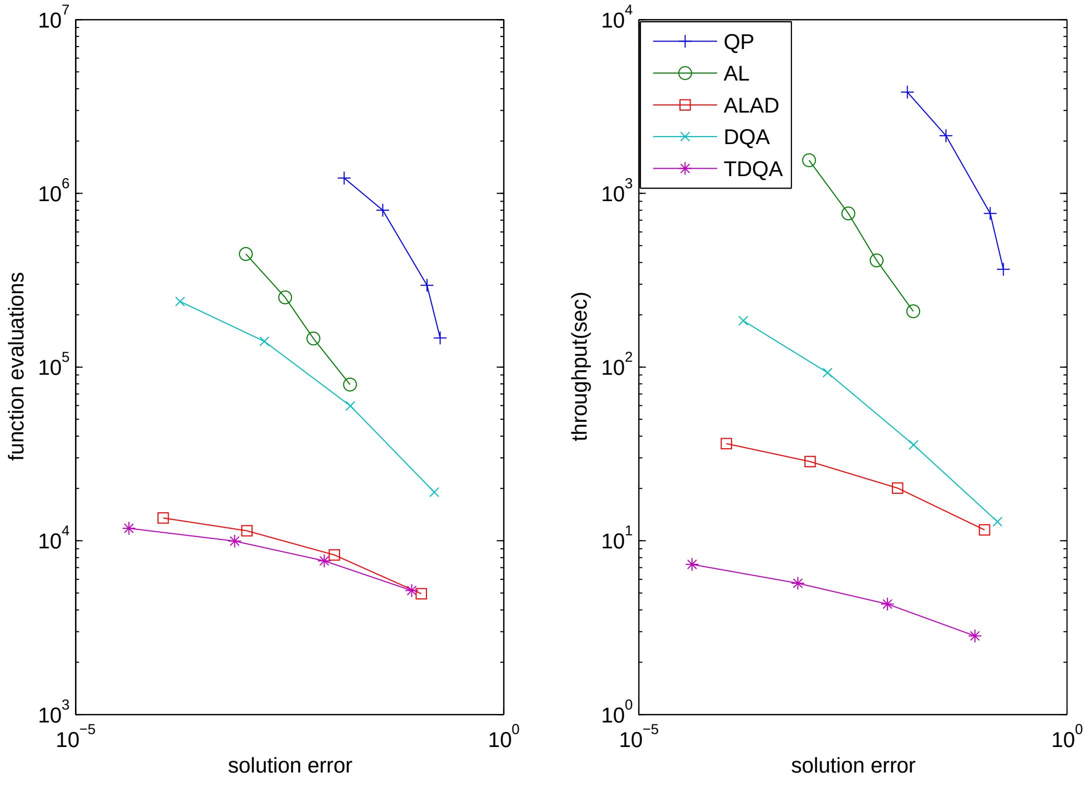


Figure 13 Example 2: Computational cost and throughput vs solution accuracy

5.3 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 $min_{z_1,\ldots,z_{14}} f = f_1 + f_2 = (z_1 - 2.9)^2 + (z_2 - 3.1)^2$.

In this problem, as discussed in [18], the Lagrange multipliers were originally set to 0, which happens to be the optimal value for this problem. In DQA, all computational effort is spent on improving the linearization, since the Lagrange multipliers do not need to be updated. This behavior is similar to AL and QP. In TDQA, since in the first iteration the linearization might not be perfect, the Lagrange multipliers will first be driven away from its optimal value, and eventually, as the solutions get closer to their optimal value, the Lagrange multipliers also tend to their optimal value. And this behavior is similar to ALAD. Since no update of Lagrange multipliers is necessary, AL or DQA do not have an advantage over QP any more. Indeed, in this example shows a better performance in QP. However, DQA still outperforms QP and AL in terms of throughput. Between TDQA and ALAD, TDQA has less total number of function evaluations and has significantly lower throughput than ALAD.

5.4 Example 4

The fourth example is a structured optimization problem based on the analytical mass allocation problem.

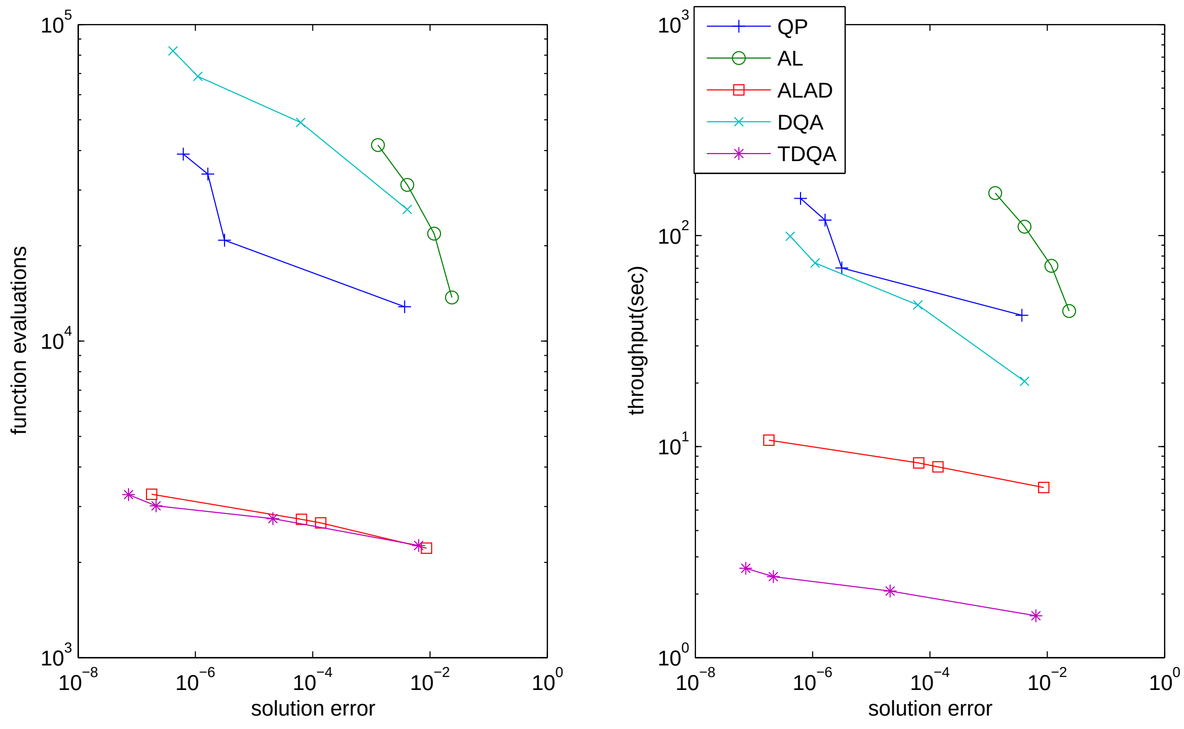


Figure 14 Example 3: Computational cost and throughput vs solution accuracy

$$
\begin{aligned}
\min_{d_1,d_2,d_{r,1},d_{r,2}} \sum_{i=1}^{3} m_i \quad &+ \quad \sum_{j=1}^{2} m_{r,j} \\
\text{s.t.} \quad g_{1,i} \quad &= \quad \sigma_{b,i} - \overline{\sigma} \le 0,\ i = 1,2,3 \\
g_{2,j} \quad &= \quad \sigma_{a,j} - \overline{\sigma} \le 0,\ j = 1,2 \\
g_{3,i} \quad &= \quad F_{t,i} - \overline{F_t} \le 0,\ i = 1,2,3 \\
g_4 \quad &= \quad f_1 - \overline{f_1} \le 0 \\
h_1 \quad &= \quad f_i - f_{i+1} - f_{r,i} = 0,\ i = 1,2 \\
\text{where} \quad m_i \quad &= \quad \frac{\pi}{4} d_i^2 L\rho,\ i = 1,2,3 \\
m_{r,j} \quad &= \quad \frac{\pi}{4} d_{r,j}^2 L\rho,\ j = 1,2 \\
\sigma_{b,i} \quad &= \quad \frac{32L(F_i - F_{i+1})}{\pi d_i^3},\ i = 1,2,3 \\
f_i \quad &= \quad \frac{64L^3(F_i - F_{i+1})}{3\pi E d_i^4},\ i = 1,2,3 \\
\sigma_{a,j} \quad &= \quad \frac{4F_{j+1}}{\pi d_{rj}^2},\ j = 1,2 \\
f_{r,j} \quad &= \quad \frac{4F_{j+1}L}{\pi E d_{rj}^2},\ j = 1,2 \\
\rho \quad &= \quad 2700 kg/m^3 \\
E \quad &= \quad 70 GPa \\
F_1 \quad &= \quad 1000 N
\end{aligned}
\tag{37}
$$

Figure 15 shows the structure of the problem.

The result is very similar to that in example 2. DQA and TDQA outperform AL and ALAD respectively in terms of function evaluations and throughput at any solution accuracy, while QP doesn't perform as well as any other methods.

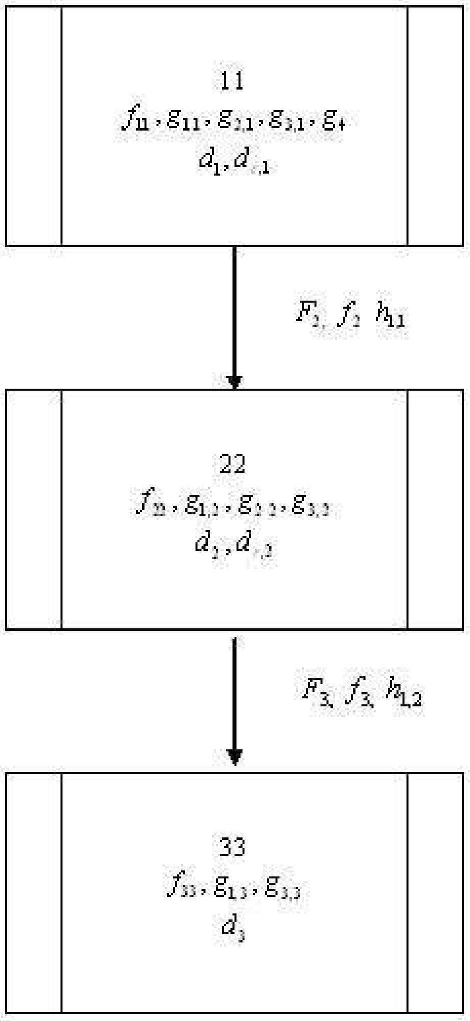

11
$f_{11}, g_{11}, g_{2,1}, g_{3,1}, g_4$
$d_1, d_{,1}$
$F_2, f_2, h_{1,1}$
22
$f_{22}, g_{1,2}, g_{2,2}, g_{3,2}$
$d_2, d_{,2}$
$F_3, f_3, h_{1,2}$
33
$f_{33}, g_{1,3}, g_{3,3}$
d_3

Figure 15 Example 4: Problem structure

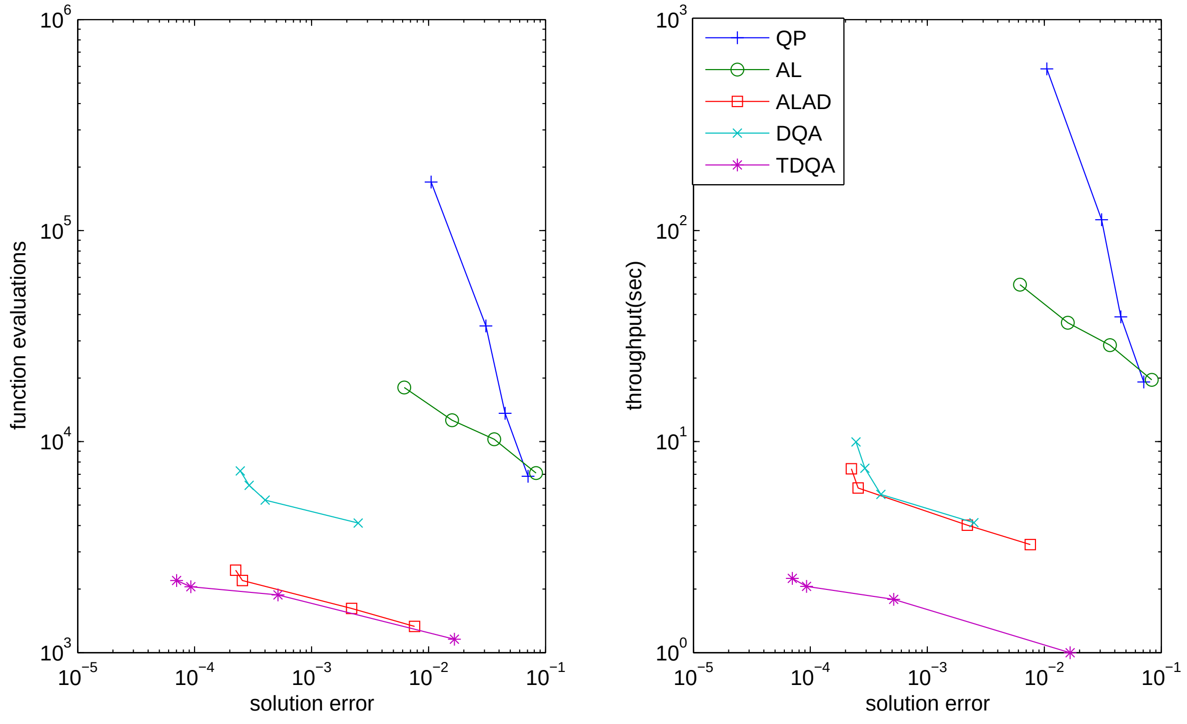


Figure 16 Example 4: Computational cost and throughput vs solution accuracy

VI. Discussion

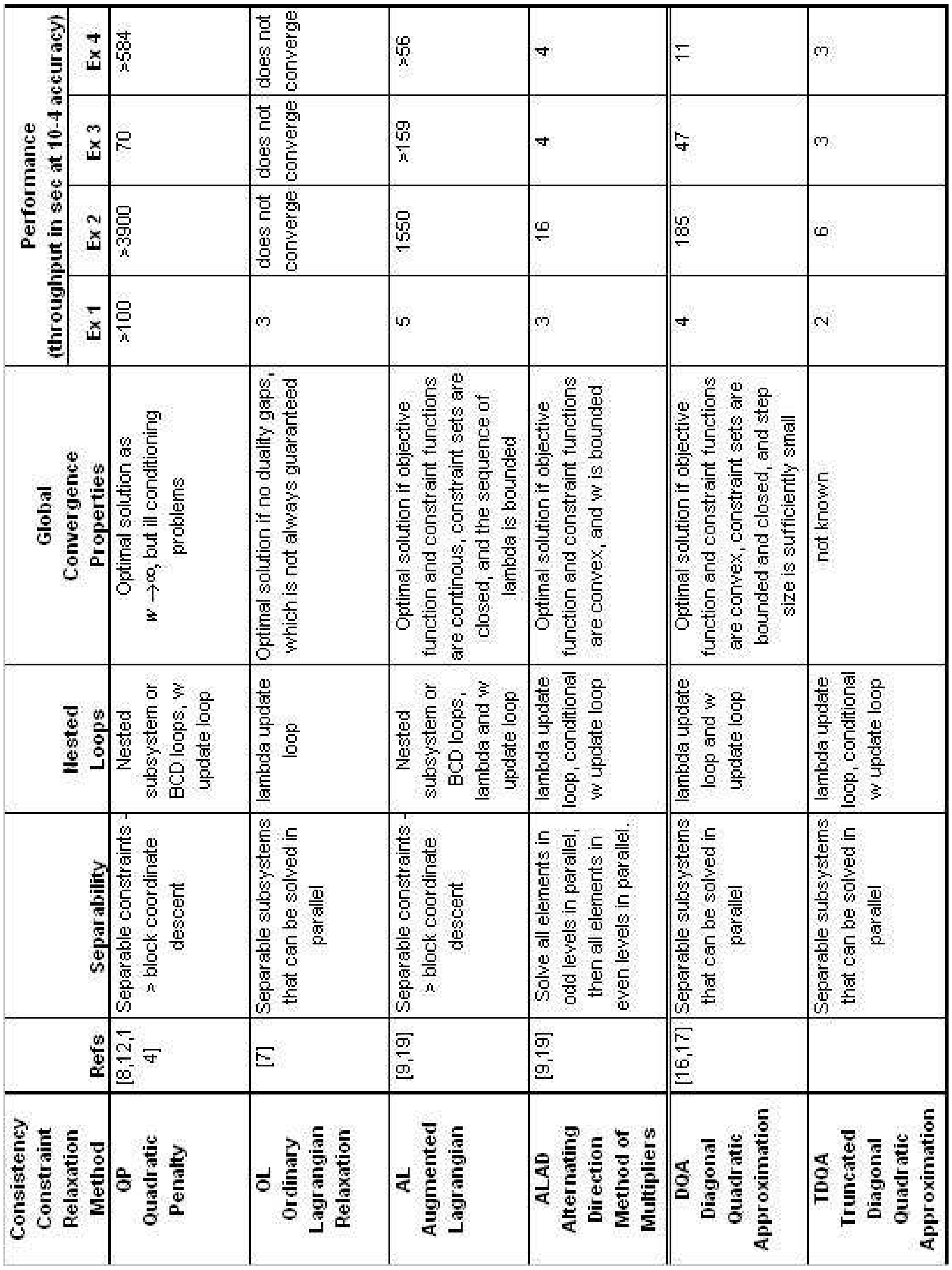
The biggest advantage of both DQA and TDQA is their ability to separate subproblems for parallel computation. This property is highly desirable, especially in large-scale problems: Since each subproblem is separable, nested loops of the iterative coordination process is avoided. Moreover, since each problem can be solved in parallel, throughput is greatly increased. Although we only present DQA and TDQA in solving ATC decomposition problems, these methods can be used in a wide variety of decomposition methods. When DQA was first proposed in [16], it was intended for solving general dual block-angular problems. And TDQA can be applied to any problems that DQA can apply to. For example, in [19], and 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. It is expected that DQA and TDQA will perform similarly in nonhierarchical decomposition schemes as in ATC.

Despite the good empirical results of TDQA, it lacks a formal convergence proof, and future work is required for obtaining proofs on convergence properties of TDQA.

VII. Conclusion

In this thesis, we have presented the diagonal quadratic approximation method (DQA) and the truncated diagonal quadratic approximation method (TDQA) used in handling consistency constraint relaxation in ATC. Moreover, the trust region method can be integrated into both the DQA and the TDQA method. The DQA method is supported by theoretical justification. Fundamental results show that DQA has comparable solution accuracy and total cost to the augmented Lagrangian method, but it greatly improves throughput because of parallelization, and that TDQA has the least total cost and best throughput at any solution accuracy of all methods presented. The DQA method has proofs for local and global convergence, and proofs for TDQA are left for future work. A table summarizing all methods is shown in Figure 17.

The proposed methods overcome many of the concerns with ATC and other decomposition methods, such as convergence, ill-conditioning, and computational cost associated with the coordination solution strategies. In the empirical examples, DQA and TDQA perform comparably or better than AL and ALAD respectively in terms of function evaluations, and the separability afforded by DQA and TDQA allows many of these function evaluations to be computed in parallel, dramatically improving throughput. The proposed methods are highly promising for solving general decomposition problems.



Consistency Constraint Relaxation Method	Refs	Separability	Nested Loops	Global Convergence Properties	Performance (throughput in sec at 10-4 accuracy)			
					Ex 1	Ex 2	Ex 3	Ex 4
QP Quadratic Penalty	[8,12,14]	Separable constraints -> block coordinate descent	Nested subsystem or BCD loops, w update loop	Optimal solution as $w \to \infty$, but ill conditioning problems	>100	>3900	70	>584
OL Ordinary Lagrangian Relaxation	[7]	Separable subsystems that can be solved in parallel	lambda update loop	Optimal solution if no duality gaps, which is not always guaranteed	3	does not converge	does not converge	does not converge
AL Augmented Lagrangian	[9,19]	Separable constraints -> block coordinate descent	Nested subsystem or BCD loops, lambda and w update loop	Optimal solution if objective function and constraint functions are continous, constraint sets are closed, and the sequence of lambda is bounded	5	1550	>159	>56
ALAD Alternating Direction Method of Multipliers	[9,19]	Solve all elements in odd levels in parallel, then all elements in even levels in parallel.	lambda update loop, conditional w update loop	Optimal solution if objective function and constraint functions are convex, and w is bounded	3	16	4	4
DQA Diagonal Quadratic Approximation	[16,17]	Separable subsystems that can be solved in parallel	lambda update loop and w update loop	Optimal solution if objective function and constraint functions are convex, constraint sets are bounded and closed, and step size is sufficiently small	4	185	47	11
TDQA Truncated Diagonal Quadratic Approximation		Separable subsystems that can be solved in parallel	lambda update loop, conditional w update loop	not known	2	6	3	3

Figure 17 Summary of methods

Bibliography

1. N.Y. Alexandrov and R.M Lewis. "Analytical and computational aspects of collaborative optimization for multidisciplinary design". *AIAA Journal*, 40(2):301 – 309, February 2002.

2. 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.

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

4. D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and Distributed Computation.* Prentice-Hall, 1989.

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

6. 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.

7. 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.

8. H.M. Kim, W.Chen, and M.M. Wiecek. "Lagrangian coordination for enhancing the convergence of analytical target cascading".

9. 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.

10. 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.

11. 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.

12. 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.

13. 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.

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

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

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

17. 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.

18. 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.

19. S. Tosserams, P. Etman, and J. Rooda. "An augmented lagrangian decomposition method for dual block-angular problems in MDO". In *AIAA*, 2006.