6

A Distributed Algorithm for Reconfigurable Process Control

6.1 Introduction

We now develop a strategy to effectively coordinate the operation of the DRPC system. Predictably, this is in the form of a distributed algorithm for guiding and managing the distributed interactions of process elements to achieve suitable control settings. We focus in particular on the interactions between unit elements during synthesis phase of the reconfiguration process (Fig. 5.1) where the settings of each element are established. The proposed algorithm is used by unit elements to find control settings for their local and other network parameters once the physical layout of the specific process scheme to be used for production has been established.

6.1.1 Overview

Before beginning with the strategy development, we position the proposed approach in the context of existing distributed approaches.

As reviewed in Section 2.4, previous research on distributed approaches in control, both in manufacturing and other domains, has used distribution to solve large control problems by breaking them into multiple smaller problems referring to individual subsystems and then solving them either independently (Šiljak 1991) or via iterative coordination based on hierarchical (Mesarovic *et al.* 1970) or distributed (Bertsekas & Tsitsiklis 1989) techniques. Alternatively, when problems are already distributed, the question of problem solving is to coordinate the local solutions so as to ensure a global objective or constraint is satisfied. Problems of these nature arise in numerous large-scale domains as explored in Section 2.5 which just looks at few.

Previous research in distributed manufacturing paradigms of holonic and agent-based manufacturing control have taken a view of separating the control architecture from control algorithms to enhance reconfigurability of the architecture, *i.e.,* the desire for reconfigurability of the architecture and software drives distribution rather than computational simplification. Bongaerts

(Bongaerts *et al.* 2000, Bongaerts 1998) in this sense used a mix of hierarchical hierarchical (fully distributed) control to switch between proactive (when conditions are planned) and reactive behaviour (when disturbances arise) of distributed holons in so-called PROSA architecture (van Brussel *et al.* 1998). More advanced work on distributed scheduling has used lagrangian decomposition (Liu & Sycara 1997, Gou *et al.* 1998) and market programming approaches (Váncza & Márkus 1998, Tharumarajah 2001, Shen, Wang & Hao 2006) to define the methods for resource allocation, *i.e.,* assignment of tasks to machines and scheduling of their start and end times.

In this chapter we use a distributed coordination technique of *nested decomposition*, studied previously for multi-stage optimisation problems (Ho & Manne 1974, O'Neill 1976, Wittrock 1985), to define the coordination strategy for process elements. Our rationale for using this approach is two-fold:

a. The process units in a continuous process remain tightly interconnected, therefore the coordination of their distributed settings should occur via direct interactions between them instead of achieved by separate product elements as in previous holonic and agent research. It is likely that if the latter approach is used the amount of coordination effort required could become excessive;
b. The approach of nested decomposition provides an economic interpretation that can be linked to the price and demand guided interactions between companies in a virtual enterprise, and so, to the use of this analogy in defining the protocol for material exchange in the interaction model.

The previous techniques in nested decomposition are not immediately applicable to process control problems though because they can only be linked to multi-stage process networks of series-connected form. Instead, we seek an extension which can be applied to process networks of arbitrary form.

6.1.2 Requirements for a Distributed Coordination Strategy

The coordination approach is expected to support the distributed nature of control architecture and interaction model in previous two chapters. This distribution was considered essential to promote a maximum level of reconfigurability in the design and interactions of process elements. Any numerical technique used as part of coordination must not disturb the reconfigurability, *i.e.,* the use of a centralised entity or constraint must be avoided. We also aim that the elemental sub-problems (as obtained after distribution) adhere to a common production objective which in this chapter is considered as the sum of all local costs. This is to ensure the distributed solution meets the optimality and global coherence of hierarchical control where possible. With regards to the four requirements in Fig. 2.7, the strategy is also expected to provide a level of responsiveness to variations in local problem formulations and other disturbances.

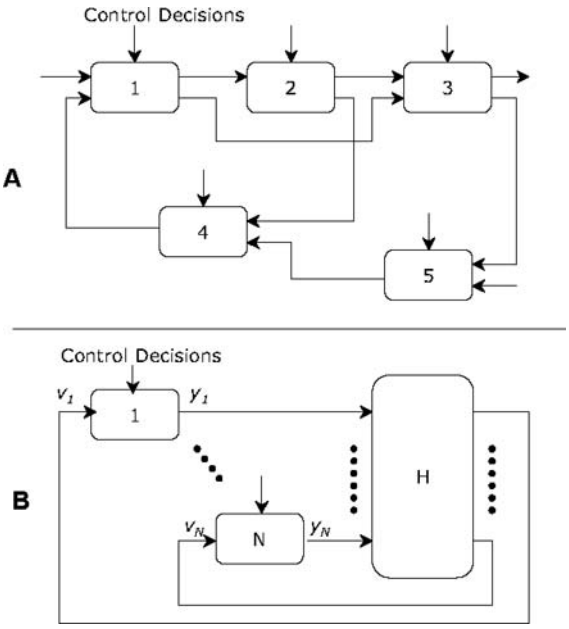

Fig. 6.1. Schematic of a large-Scale or complex system

This chapter is structured as follows. The next section introduces the formulation of distributed control problem to be used in this chapter as the basis of analysis. Section 6.3 characterises the technique behind overall solution strategy. Sections 6.4 to 6.6 then develop the distributed algorithm in a constructive manner. Illustrative examples and the potential future extensions of the approach are discussed in Sections 6.7 and 6.8.

6.2 Distributed Control Problem

We start by mathematically formalising the distributed control problem to be used in this chapter.

A chemical process can be seen as a large-scale system comprised of multiple subsystems or process units as shown in Fig. 6.1(a). In this network form, the outputs of each unit become the inputs to its downstream units and so on. By imposing an orderly input-output matrix H, this network form can be converted into a generic form shown in Fig. 6.1(b) where H now represents the network structure of the process, with each row in H referring to an input and each column to an output of associated process unit.

Assuming there are N units in the process, we then use the following model structure to define the dynamics of the whole process.

$$\begin{aligned}
&\dot{x}_i(t) = h_i(x_i(t), u_i(t), v_i(t), t) \quad i = 1, \ldots, N \\
&y_i(t) = g_i(x_i(t), u_i(t), v_i(t), t) \\
&v_i(t) = \sum_{\substack{j=1,\ldots,N \\ j \neq i}} H_{ij} y_j(t) \\
&G_i(x_i(t), u_i(t), v_i(t), t) \in S_i \\
&r(x(t), u(t), v(t), t) \in R
\end{aligned} \tag{6.1}$$

where $x_i \in X_i$ is the vector of states, $u_i \in U_i$ is the vector of manipulated variables, $y_i \in Y_i$ is the output vector and $v_i \in V_i$ is the interaction vector associated with unit i. The vectors x, u and v are the aggregate vectors of x_i, u_i and v_i, $i = 1, \ldots, N$ respectively. The constraints h_i and g_i represent the state and output equations, G_i are the local constraints, and r is the shared constraint coupling one or more process units. The matrix H_{ij} then aggregates the effects of all units $j \neq i$ on unit i. □

Assuming process units are connected via piping streams only, we can derive a specific formulation of matrix H. To do so, we adopt the so-called *P-Graph* model proposed by Friedler, Tarján, Huang & Fan (1992).

Omitting services (*e.g.*, steam, cooling water), in a P-Graph form the process is represented as a finite set of materials M being transformed by a finite set of process units O available in the process. Each process unit i in this sense is written as a material tuple $(\mathrm{mat}_i^{in}, \mathrm{mat}_i^{out})$ where mat_i^{in} and $\mathrm{mat}_i^{out} \in M$ are the sets of incoming and outgoing materials. If $\wp(M)$ is the set of all possible subsets of M, then we get the following two relationships:

$$O \subset \wp(M) \times \wp(M) \quad O \neq \emptyset,$$

and

$$M = \left\{ \bigcup_{i \in O} \mathrm{mat}_i^{in} \right\} \bigcup \left\{ \bigcup_{i \in O} \mathrm{mat}_i^{out} \right\}$$

We can then use the P-Graph (M, O) to represent the network structure of the process as a directed, bipartite graph comprising material nodes (as elements of M) and unit nodes (as elements of O). Fig. 6.2 shows an example P-Graph comprising three unit nodes, where M and O can be written as $M = \{A, B, C, D, E, F\}$ and $O = \{\mathsf{U1}, \mathsf{U2}, \mathsf{U3}\}$.

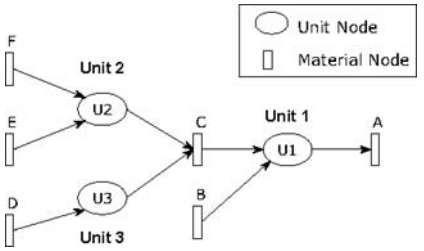

Fig. 6.2. An example of P-Graph model

In what follows, we consider a specific interpretation of vectors v_i, u_i, x_i and y_i in Eq. 6.1 to relate the model equations with demand-pull type interactions between unit elements in the interaction model.

- *Interactions v_i:* The interactions v_i are taken as the flow rate demands for outgoing materials mat_i^{out} of unit i, *i.e.*, the flow-rate demands act as a form of disturbances that the unit cannot control on its own but are set by the units in its downstream. Other types of interactions, *e.g.*, due to pressure and temperature variables, are omitted here.
- *Manipulated Inputs u_i:* The manipulated inputs u_i are divided into three types: $u_{i,in}$, $u_{i,util}$ and $u_{i,loc}$. The $u_{i,in}$ are set as the input flow-rates for materials mat_i^{in}, $u_{i,util}$ as the input flow-rates of utilities, and $u_{i,loc}$ as other local variables (*e.g.*, agitator speed of a reactor) associated with unit i. $u_{ij,in}$ as the jth element of $u_{i,in}$ then refers to unit i's input demand to unit j in its upstream.
- *States x_i:* The state variables x_i refer to various local properties of unit i, such as level, volume or material concentrations.
- *Outputs y_i:* The outputs y_i are taken as input demands $u_{i,in}$, *i.e.*, y_i for materials mat_i^{in} of unit i.

Using the above assignment of variables and P-graph model (M, O), we get at the following equivalent form of Eq. 6.1 where P-graph (M, O) is now used to replace matrix H.

$$\begin{aligned}
&\dot{x}_i(t) = h_i(x_i(t), u_i(t), v_i(t), t) \quad i = 1, \ldots, N \\
&y_i(t) = \{u_{ij,in}(t)\} \quad \text{for all } j \in \mathbf{S}_i^q, q \in \text{mat}_i^{in} \\
&v_i(t) = \left\{ \sum_{j \in \mathbf{M}_i^d} u_{ji,in}(t) \right\} \quad \text{for all } d \in \text{mat}_i^{out} \\
&G_i(x_i(t), u_i(t), v_i(t), t) \in S_i \\
&r(x(t), u(t), v(t), t) \in R
\end{aligned} \tag{6.2}$$

where, in reference to P-Graph (M, O), $\mathbf{M}_i^d$ and $\mathbf{S}_i^q$ are:

$\mathbf{M}_i^d$: indices of units $j \in [1, \ldots, N]$ connected to unit i through a material stream $d \in \text{mat}_i^{out}$

$\mathbf{S}_i^q$: indices of units $j \in [1, \ldots, N]$ connected to unit i through a material stream $q \in \text{mat}_i^{in}$. □

In this chapter, to simplify the discussion and limit the amount of mathematical rigor involved, we restrict ourselves to a linear, steady-state form of Eq. 6.2 as follows.

$$\begin{aligned}
&0 = A_i x_i + B_i u_i - E_i v_i \\
&y_i = \{u_{ij,in}\} \quad \text{for all } j \in \mathbf{S}_i^q, q \in \text{mat}_i^{in} \\
&v_i = \left\{ \sum_{j \in \mathbf{M}_i^d} u_{ji,in} \right\} \quad \text{for all } d \in \text{mat}_i^{out} \\
&x_i \in X_i, \\
&u_i \in U_i
\end{aligned} \tag{6.3}$$

where x_i, u_i, v_i and y_i now all refer to their steady-state values. Matrices A_i and B_i in Eq. 6.3 are assumed to be of appropriate dimensions, while C_i, D_i, E_i and H_{ij} are assumed to possess a special form as discussed later in this section. Note that we have omitted the shared constraint $r(x, u, v) \in R$ in Eq. 6.2.

Fig. 6.3 depicts the example from Fig. 6.2 with the dynamics of individual units and their interactions. Note that process units are connected through relationship $v_i = \left\{\sum_{j \in \mathbf{M}_i^d} u_{ji,in}\right\}$ for all $d \in \text{mat}_i^{out}$. For units 2 and 3 the set $\mathbf{M}_i^d$ refers to unit 1 as the only customer unit for material C.

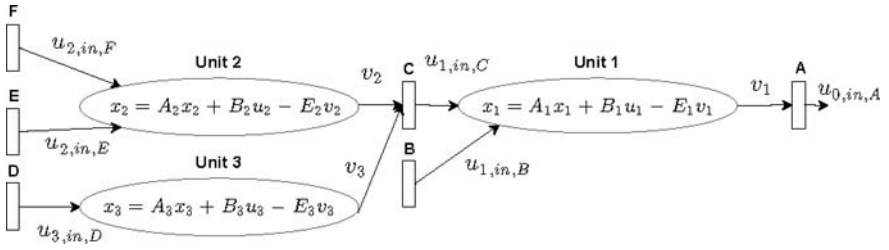

Fig. 6.3. Local unit dynamics for P-Graph in Fig. 6.2

In what follows we make a further assumption that the interaction vector v_i for all units $i = 1, \ldots, N$ is of the dimension equal to number of materials in mat_i^{out}, *i.e.,* each element in v_i refers to a demand for a material in mat_i^{out}. If unit i supplies the same material to multiple customer units, then the total sum of all demands is used as the value for respective element in v_i.

With this assumption, the formulation of matrices C_i, D_i and E_i in Eq. 6.3 can be simplified. In particular, all entries in E_i are set to 0 except where a 1 appears when an element in v_i is connected to an element in state vector x_i via the state equation. The matrix C_i similarly becomes a zero matrix with all entries 0, while D_i becomes a matrix with all entries 0 except where a 1 appears when any element of $u_{i,in}$ is connected to an element of y_i. Note that each element in $u_{i,in}$ in this form is connected to only one element in y_i.

Using the revised form of E_i we can also simplify Eq. 6.3 by separating the rows in A_i and B_i which do not contain any element of v_i into separate local matrices $A_{i,loc}$ and $B_{i,loc}$. The constraints involving $A_{i,loc}$ and $B_{i,loc}$ then become the local constraints of unit i. For the sake of simplicity, we separate these constraints from remaining constraints and assume that A_i and B_i now only refer to those rows that correspond to an element of v_i. For simplicity, we also assume that each row in A_i and B_i is associated with only one element in v_i, *i.e.,* the number of rows in A_i and B_i equal the length of v_i, which, in turn, equals the number of outgoing material streams $d \in \text{mat}_i^{out}$. We can then eliminate E_i from Eq. 6.3 altogether.

We now define the distributed control problem used in this chapter. Formally, the problem assigned to each unit $i = 1, \ldots, N$ is to find an optimal, steady-state deviation in its variables x_i and u_i from a nominal operating point $\bar{x}_i$ and $\bar{u}_i$ for a given demand deviation of v_i from the nominal demand $\bar{v}_i$. The nominal point for all three vectors may refer to a target set point supplied by the higher-level optimiser.

Problem 6.1 (Distributed Control Problem).

$$\begin{aligned}
\underset{x_i,u_i}{\text{minimise}} \; & \sum_{i=1}^{N} f_i(x_i,u_i) \quad i=1,\ldots,N \\
\text{s.t.} \quad & A_i x_i + B_i u_i = v_i, \\
& A_{i,loc} x_i + B_{i,loc} u_i = 0, \\
& y_i = \{u_{ij,in}\}, \quad \text{for all } j \in \mathbf{S}_i^q, q \in \text{mat}_i^{in} \\
& v_i = \left\{ \sum_{j \in \mathbf{M}_i^d} u_{ji,in} \right\}, \quad \text{for all } d \in \text{mat}_i^{out}, \\
& x_i \in X_i, u_i \in U_i \qquad \square
\end{aligned} \tag{6.4}$$

The objective function $f_i(\cdot)$ is assumed to be strictly convex jointly on its constituent variables x_i and u_i. For simplicity, we assume that f_i is linear-quadratic, *i.e.*, $x_i^T Q_i x_i + u_i^T R_i u_i + c_i^T [x_i^T, u_i^T]^T$ where $[]^T$ represents the transpose operator. This together with the affine nature of the constraint equations guarantees that the dual problem of Prob. 6.1 is differentiable (Rockafellar 1970).

In summary, the distributed control problem to be solved for the overall process is to minimise the joint total cost (as the sum of individual costs) of all units in the P-Graph subject to a constraint that all material flow interactions are satisfied between units. The solution of this problem then defines the material flow-rates $u_{i,in}$ in the network.

Note that although the local costs $f_i(x_i, u_i)$ of all units $i = 1, \ldots, N$ are separable, the individual sub-problems are not, because the constraints $v_i = \{\sum_{j \in \mathbf{M}_i^d} u_{ji,in}\}, \forall d \in \text{mat}_i^{out}$ link them. As a result the overall problem cannot simply be decomposed into sub-problems and solved independently. A distributed approach to solving Prob. 6.1 must be able to coordinate these linking constraints via distributed interactions.

Prob. 6.1 is general enough to be applied to a process network of any arbitrary nature. However, in this work, we limit ourselves to processes of *acyclic* nature only (*i.e.*, processes that do not contain material or energy recycles) and having no by-products. Recycles or by-products play an important role in modern process plants, however the developments made in this chapter cannot support such process forms at present.

We observe that this and other assumptions made in this section regarding problem formulation and network structure help us to simplify the solution strategy described next. These can be relaxed as appropriate by generalising the approach discussed here.

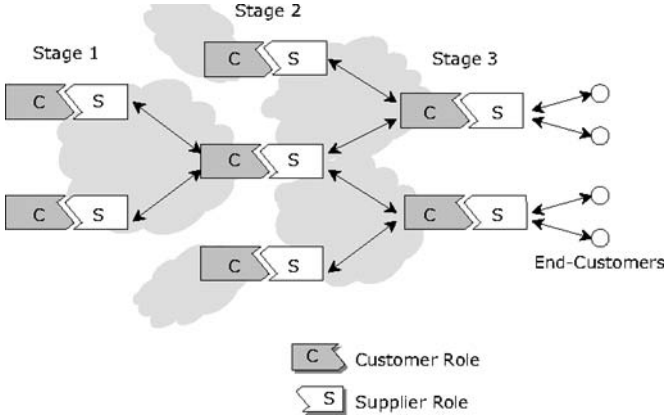

Fig. 6.4. Supplier-customer relationships between unit elements

6.3 Distributed Coordination Approach

Having defined the problem formulation, we now develop the distributed coordination approach used in this chapter for solving Prob. 6.1. We use the concept of so-called *nested decomposition* from optimisation and operations research literature (Ho & Manne 1974, O'Neill 1976, Wittrock 1985) to develop the approach. In simple terms, it refers to solving a multi-stage optimisation problem (*e.g.*, in a staircase type linear program) by solving multiple, smaller coordination problems, each associated with a linking constraint connecting two successive stages.

We split the development of the proposed approach into three main steps:

i. *Problem decomposition:* Develop a method for decomposing the overall process network into multiple two-stage problems;
ii. *Solution of two-stage problems:* Develop a general method for solving customer-supplier coordination (the two-stage problems);
iii. *Solution of multi-stage problems:* Develop an algorithm for solving the two-stage problems in a nested sequence.

Before describing each of these in detail (Sections 6.4-6.6), below we provide in this section an outline of the overall approach.

Fig. 6.4 repeated here from Chapter 5 depicts the form of supplier-customer type relationships between unit elements in the interaction model – the unit elements act as the suppliers for their outgoing products and customers for their incoming feedstocks. In order to formalise this relationship mathematically, we consider a further analogy of price-demand type interactions between companies in a supply chain, or specifically a virtual enterprise, when operating under a make-to-order environment.

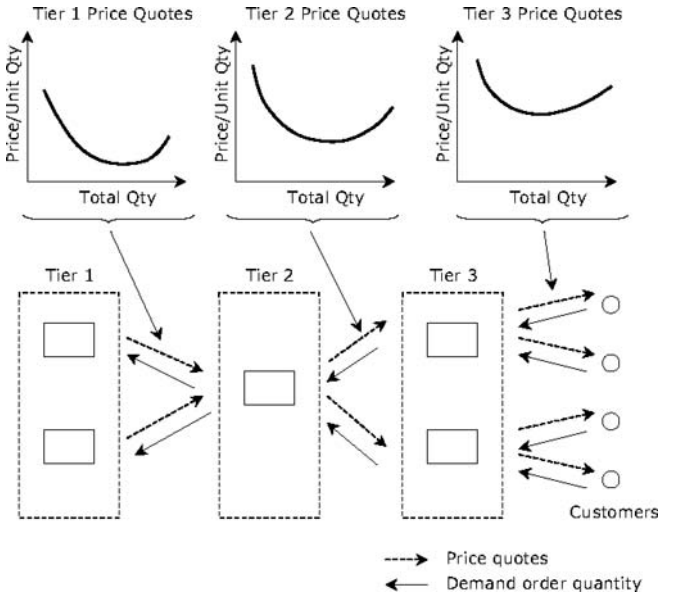

Fig. 6.5. Price-demand oriented interactions between companies in a supply chain

In a supply chain, as shown in Fig. 6.5, companies share price quotes and demand quantities to make decisions on which suppliers to select and how much demand is allocated to each supplier. The customer companies along each tier request potential supplier companies upstream for price quotes for expected demand quantities. The price quotes then flow forwards. The customer companies use these price quotes to select suppliers and allocate demands. The interactions thus remain bidirectional and may repeat until all companies settle on a price-demand contract, at which point the operations can commence.

When viewed in an analogous manner, we can formalise the interactions between unit elements by attaching a price quote to a supply proposal as the variation in local cost of the supplier unit element for a unit change in the product flow rate. The interactions would then proceed as follows.

Starting from the terminal stage in the process, each unit element in its role as a customer together with all unit elements acting as its suppliers attempt to form a two-stage control problem of the form of Prob. 6.1 involving only these unit elements. The objective is to distribute the customer's feedstock demands among suppliers in a manner that the total cost of suppliers and of the customer is minimised. The same principle is then applied to these supplier elements who attempt to distribute their demands by forming appropriate two-stage control problems involving unit elements further upstream. The process is repeated until unit elements in the first stage of the process are reached.

At this stage, we obtain multiple two-stage problems involving unit elements from successive stages of the process. The overall solution approach then operates by solving these two-stage problems sequentially in a nested, iterative fashion.

Each individual two-stage problem is itself associated with a two-level coordination algorithm based on the so-called *primal decomposition* concept (Geoffrion 1970). A detailed explanation of primal decomposition approach is given in Appendix A. Within the two-stage problem, each unit element has two roles. In its role as a customer, it becomes the *coordinator* of its demand distribution, and as a supplier a *sub-problem* where its role is supplying the price quotes. We next exploit an economic interpretation of the primal decomposition algorithm to link the solution of two-stage problems with the price-demand type interactions in Fig. 6.5.

At first, each unit element in its role as a customer (called the customer unit) selects a distribution of its feedstock demands and passes that as *coordination variables* to all unit elements acting as its suppliers (called supplier units). For a given demand, the supplier units attempt to solve their local problems to find the optimum supply costs and a solution to other local variables. The supplier units return back to customer units their supply proposals comprising: (a) the supply cost for the specified demand, and (b) the *marginal cost* as an indication of the variation in supply cost for a unit change in the demand flow rate. In the language of primal decomposition, this supply proposal refers to so-called *optimality cut* that is included in customer unit's local problem at the next iteration. Subsequently, at each iteration, the customer unit adjusts its demand, taking into account the previous supply proposals, such that the total cost of all suppliers plus its own is minimised.

The overall interactions across the process then operate in a chain whereby all these two-stage problems are solved iteratively in a nested, iterative sequence. Starting from terminal stage, each individual customer unit allocates its feedstock demands to its supplier units. These supplier units then further propagate the demands upstream until the first stage is reached. Next, starting from the first stage, the supplier units return their supply proposals (as optimality cuts) back to customer units downstream. The customer units in each stage then include the new supply proposals to solve their local problems and alter the demands allocated. This multi-level chain of interactions thus repeat, first backwards and then forwards, until all interim demand flow rates converge to an optimum value.

The solution strategy described above supports the reconfigurability of the interaction model in following ways. Firstly, the above solution approach operates in a completely distributed form, *i.e.,* the overall problem is solved by direct interactions between unit elements themselves without referral to a centralised coordinator. Secondly, because all unit element problems are made independent via distributed interactions, the approach allows adding or removing unit elements from problem formulation without having to reformulate the dynamics model in Prob. 6.1 which otherwise would be necessary

in a centralised implementation. Thirdly, under certain conditions it can be shown that the solution obtained for interim flow rates converges to the same optimal solution as that obtained by solving a centralised problem. Finally, if there are bounds on supply capacities of the supplier units that can be modelled in the formulation of customer problems, then it is possible to restrict the interim demands of customer units to be within these bounds to ensure feasibility. The above solution process, hence, can be interrupted at any stage in the sequence to use a suboptimal but immediately usable solution. This might be desirable, for example, in a constantly changing environment. The proposed approach thus retains distributed character of the interaction model whilst also maintaining optimality of the solution.

In the next three sections we now address the three steps of the proposed distributed coordination approach in detail.

6.4 Problem Decomposition

In the first step to solving Prob. 6.1, the overall problem is decomposed into a set of two-stage problems, each referring to a network junction between two or more process streams. We refer to such a junction as a *Junction Block*. Each junction block is thus a two-stage process consisting all unit elements and process streams associated with that junction.

Fig. 6.6 shows the four different types of junction blocks that can be found in any acyclic process network. The MIXER and SPLITTER blocks represent the junctions associated with process units such as mixer, splitter or a piping header where multiple material streams of identical properties are mixed together or a single stream split into multiple such streams. The MULTIFEED and MULTIPROD represent the junctions associated with process units such as feed preparation, reactor, distillation column *etc.*, where material streams of non-identical properties are merged together or are produced as outcomes of the processing task. Note that a more complex junction with multiple incoming and outgoing streams can be always represented by superimposing MIXER or MULTIFEED blocks on top of SPLITTER or MULTIPROD blocks.

Fig. 6.7 shows an example of problem decomposition in which an arbitrary process network is decomposed into its constituent junction blocks. The junction blocks are interconnected via process units common between them.

6.5 Solution of Two-Stage Problems

In the next step to solving Prob. 6.1, each junction block, being a two-stage process, is associated with a two-level coordination algorithm based on primal decomposition concept of the type introduced in Section 6.3. This algorithm is used to solve the associated mini control problem of junction block involving local problems of unit elements in both stages in a distributed manner. The

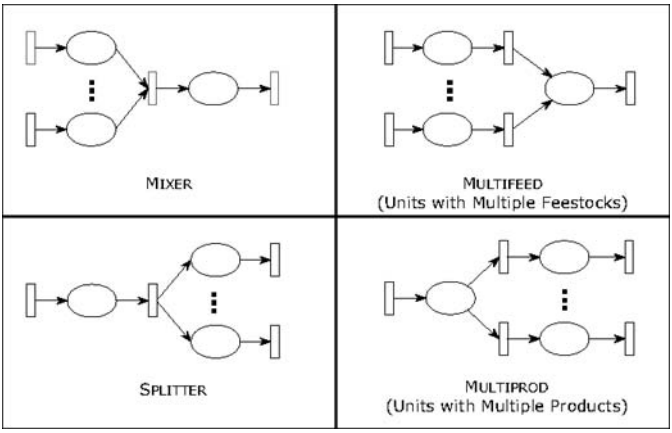

Fig. 6.6. Four types of 'junction blocks' in an acyclic process network

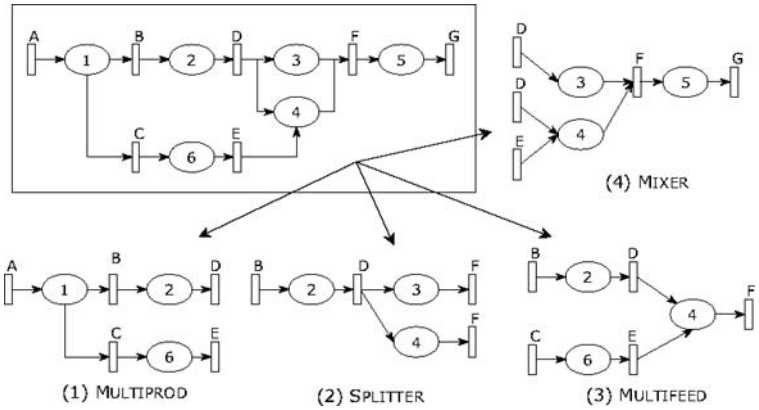

Fig. 6.7. Decomposition of a process network into junction blocks

unit elements in the second stage become the *coordinators* or so-called *master* problems and those in the first stage as so-called *sub-problems*. In economic terms, the first and second stage elements also act as suppliers and customers of materials through which they are connected in the P-Graph.

Since all four junction blocks in Fig. 6.6 have different network structures, *i.e., joins* in MIXER and MULTIFEED blocks and *forks* in SPLITTER and MULTIPROD blocks, they need different coordination techniques. The problems for MIXER and MULTIFEED blocks involve a single coordinator and can be solved based on primal decomposition technique as described in Appendix A. However, for SPLITTER and MULTIPROD blocks, this is not directly possible as they involve multiple customers and hence multiple coordinators. Therefore, we develop a variant of primal decomposition using the techniques from parametric programming field (Fiacco 1983) to accommodate this variation.

In what follows, we develop a single algorithm that can be applied to all four types of junction blocks with a facility to tailor the algorithm for individual type of block. To develop the necessary algorithm, we consider solving three problems of increasing complexity:

i. Two-Units problem involving two, series-connected units (Section 6.5.1)
ii. Two-Units problem with an uncontrolled parameter (Section 6.5.2)
iii. Multi-unit problem, so-called *superset block* problem, which enables the solution to all four junction blocks (Section 6.5.3).

6.5.1 Solution of Two-Units Problem

We consider first a process network comprising two unit elements connected in series, the so-called STAIRCASE block. Based on Prob. 6.1, the associated control problem for this block can be written as:

Problem 6.2 (Two-Units Problem).

$$\begin{aligned} \underset{\substack{x_1,u_1\\x_2,u_2}}{\text{minimise}}\ & f_1(x_1,u_1)+f_2(x_2,u_2) \\ \text{s.t.}\quad & A_1x_1+B_1u_1=v_1 \\ & A_2x_2+B_2u_2=v_2 \\ & x_1\in X_1, u_1\in U_1 \\ & x_2\in X_2, u_2\in U_2 \end{aligned} \tag{6.5}$$

where $v_1 \equiv y_2 = u_{21,in}$ represents the demand from unit 2 to unit 1.□

In the proposed use of primal decomposition, unit 2 becomes the coordinator or master problem (denoted by SP_2), unit 1 as the only sub-problem (denoted by SP_1), and v_1 is the interaction variable linking them. The solution process then operates iteratively. For an initial value of $\hat{v}_1$, the problem SP_1 is solved first to find the optimal values of the value function α_1 and the Lagrange multiplier λ_1 for linking constraint $A_1x_1 + B_1u_1 = \hat{v}_1$. This information is passed as an optimality cut to the master problem SP_2. With including this new cut, the master problem SP_2 is solved to find a revised value of $\hat{v}_1$ that minimises the total cost of both problems. The process thus repeats between solving SP_1 and SP_2 until a form of convergence is achieved. Algorithm 6.1, based on the description in Section A.2, Appendix A, formally defines this solution procedure.

Fig. 6.8 outlines the information exchange between SP_1 and SP_2 problems. Note that ν_2 in SP_2 represents the supremum of piecewise-linear approximations of SP_1's optimal value function $\alpha_1(\hat{v}_1)$. This together with $f_2(x_2, u_2)$ equals the approximate total cost of both unit elements at any one iteration. Given that the optimality cuts do not overestimate $\alpha_1(\hat{v}_1)$, the optimal cost obtained from solving SP_2 provides a lower-bound on the total cost of both units. Since each new optimality cut should improve upon the linear approximation of $\alpha_1(\hat{v}_1)$, the lower-bound obtained should improve at each iteration until SP_2 converges to an optimal solution.

Algorithm 6.1 (Two-Units Problem)

Step 0: Initialise: *Set* $K := 1$. *Assume an initial value of* $x_2^{(0)} \in X_2$ *and* $u_2^{(0)} \in U_2$. *Set* $\hat{v}_1^{(0)} \equiv y_2^{(0)} = C_2 x_2^{(0)} + D_2 u_2^{(0)} = u_{21,in}^{(0)}$.

Step 1: Sub-problem $SP_1^{(K)}$*:* *At any iteration* K, *fixing* $\hat{v}_1^{(K-1)} \equiv y_2^{(K-1)}$, *solve unit* 1*'s problem* $SP_1^{(K)}$ *as:*

$$SP_1^{(K)} \qquad \begin{cases} \underset{x_1,u_1}{minimise}\ \alpha_1 \triangleq f_1(x_1,u_1) \\ s.t. \quad A_1x_1 + B_1u_1 = \hat{v}_1^{(K-1)} \\ \qquad\quad x_1 \in X_1, u_1 \in U_1 \end{cases} \tag{6.6}$$

Set $z_1^{(K)} \equiv A_1 x_1^{(K)} + B_1 u_1^{(K)}$. *Pass* $\alpha_1^{(K)}$, $\lambda_1^{(K)}$ *and* $z_1^{(K)}$ *to unit* 2.

Step 2: Master Problem $SP_2^{(K)}$*:* *Use* $\alpha_1^{(K)}$, $\lambda_1^{(K)}$ *and* $z_1^{(K)}$ *to construct a new optimality cut in unit* 2*'s problem* $SP_2^{(K)}$. *Solve the resulting problem as:*

$$SP_2^{(K)} \qquad \begin{cases} \underset{x_2,u_2,\nu_2}{minimise}\ \alpha_2 \triangleq \nu_2 + f_2(x_2,u_2) \\ s.t. \quad \nu_2 \geq \alpha_1^{(k)} + \lambda_1^{(k)}(z_1^{(k)} - u_{21,in}), \quad k \in \boldsymbol{K} \\ \qquad\quad A_2x_2 + B_2u_2 = \hat{v}_2^{(K-1)}, \\ \qquad\quad x_2 \in X_2, u_2 \in U_2, \\ where \quad \boldsymbol{K} \triangleq set\ of\ iterations \end{cases} \tag{6.7}$$

Step 3: Terminate/Iterate: *Terminate if the convergence criteria is satisfied, which is considered to be as*

$$\left\| \left\{ x_1^{(K)}, u_1^{(K)}, x_2^{(K)}, u_2^{(K)} \right\} - \left\{ x_1^{(K-1)}, u_1^{(K-1)}, x_2^{(K-1)}, u_2^{(K-1)} \right\} \right\| \leq \epsilon. \tag{6.8}$$

Else, set $K := K + 1$ *and return to step 1.* □

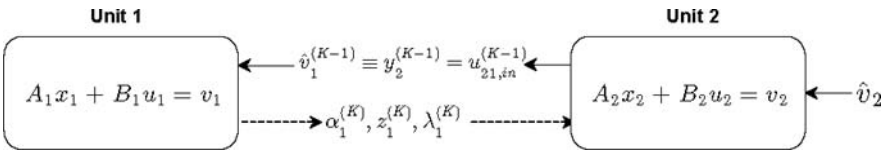

Fig. 6.8. Information exchange between units 1 and 2 sub-problems in Two-Units problem

6.5.2 Solution of Two-Units Parametric Problem

We consider next an extension of the Two-Units problem (Prob. 6.2) where we now assume that SP_1 contains an uncontrolled parameter vector θ which it cannot alter.

Problem 6.3 (Parametric Two-Units Problem).

$$\begin{aligned} \underset{\substack{x_1,u_1\\x_2,u_2}}{\text{minimise}}\ & f_1(x_1,u_1)+f_2(x_2,u_2) \\ \text{s.t.}\quad & A_1x_1+B_1u_1=v_1 \\ & A_{1,\theta}x_1+B_{1,\theta}u_1=\theta \\ & A_2x_2+B_2u_2=v_2 \\ & x_1\in X_1, u_1\in U_1 \\ & x_2\in X_2, u_2\in U_2 \end{aligned} \tag{6.9}$$

where $\theta \in \Theta$ is a vector of additional parameters local to unit 1. Assume that Θ is a convex, compact sub-set of $\mathbb{R}^{p_\theta}$ where p_θ denotes the dimension of θ. □

If Algorithm 6.1 for Two-Units problem is applied to the above problem, then sub-problem $SP_{1,\theta}$ would have the following formulation at an iteration K.

$$SP_{1,\theta}^{(K)}\left\{\begin{array}{ll} \underset{x_1,u_1}{\text{minimise}}\ \alpha_{1,\theta}\triangleq f_1(x_1,u_1) & \\ \text{s.t.} & A_1x_1+B_1u_1=\hat{v}_1^{(K-1)} \rightsquigarrow \lambda_1^{(K)} \\ & A_{1,\theta}x_1+B_{1,\theta}u_1=\theta^{(K)} \rightsquigarrow \lambda_{1,\theta}^{(K)} \\ & x_1\in X_1, u_1\in U_1 \end{array}\right. \tag{6.10}$$

where $\theta^{(K)}$ denotes the value of θ at iteration K.

Considering that the second constraint in $SP_{1,\theta}^{(K)}$ is a function of x_1, u_1 and θ, any change in θ from $\theta^{(K)}$ would change the feasible region of (x_1, u_1) and hence their optimal values for a given $\hat{v}_1^{(K-1)}$. Any such variation therefore would lead to a non-unique response from $SP_{1,\theta}$ to master problem $SP_2^{(K)}$ for a given $\hat{v}_1^{(K-1)}$. A significant variation in θ may also invalidate the optimality cuts passed to master problem in the previous iterations.

The situation can be recovered if the Algorithm 6.1 is restarted at every instance when θ changes. However, this is undesirable if θ is likely to change frequently (as in the case considered in the next sub-section). Instead, we propose a simple alteration to Algorithm 6.1 which interprets the change in θ and updates the previous optimality cuts passed to the master problem. The proposed technique is based on so-called *basic sensitivity theorem* from sensitivity analysis studies (Fiacco 1983) and is referred to as the *approximate optimality cut update technique*. See Section A.1 in Appendix A for a brief discussion of the relevant concepts from sensitivity analysis.

We first consider the following first-order approximation result based on basic sensitivity theorem (Theorem 3.2.2 in Fiacco 1983).

Lemma 6.4. *Consider the optimal solution* $\alpha_{1,\theta}^{(K)}$ *of the value function* $\alpha_{1,\theta}$ *in Eq. 6.10. Assume the constraint matrix* $\begin{bmatrix} A_1 & B_1 \\ A_{1,\theta} & B_{1,\theta} \end{bmatrix}$ *has a full row rank, i.e., the rows are linearly independent, then*

$$\frac{\partial \alpha_{1,\theta}^{(K)}}{\partial \theta} = -\lambda_{1,\theta}^{(K)} \tag{6.11}$$

where $\lambda_{1,\theta} \in \Lambda_{1,\theta} \subseteq \mathbb{R}_{+}^{p_\lambda}$ *is the vector of Lagrange multipliers for the second constraint* $A_{1,\theta}x_1 + B_{1,\theta}u_1 = \theta$, *and* $\lambda_{1,\theta}^{(K)}$ *denotes its value for a given* $\theta = \theta^{(K)}$. □

The lemma suggests that for a change of $\Delta\theta$ in θ from $\theta^{(K)}$, the optimal value of the value function α_1 can be expected to change at least by $\Delta\alpha_{1,\theta} \equiv -\lambda_{1,\theta}^{(K)}$. Note that this is still a first-order approximation, and cannot be used to find the exact change in $\alpha_{1,\theta}^{(K)}$. However, even when $\Delta\theta$ is large, the following is true.

Lemma 6.5. *Consider the value function* $\alpha_{1,\theta}$ *in Eq. 6.10. If the function* f_1 *is convex in* (x_1, u_1) *and the sets* X_1 *and* U_1 *in Prob. 6.3 are convex, then the optimal value function* $\alpha_{1,\theta}$ *is convex on* Θ. *Furthermore, for a change in* θ *from any* $\theta^{(j)}$ *to* $\theta^{(k)}, k \geq j, \theta^{(j)}, \theta^{(k)} \in \Theta$,

$$\alpha_{1,\theta}^{(k)} \geq \alpha_{1,\theta}^{(j)} - \lambda_{1,\theta}^{(j)^T}(\theta^{(k)} - \theta^{(j)}), \tag{6.12}$$

where $\alpha_{1,\theta}^{(j)}$ *and* $\alpha_{1,\theta}^{(k)}$ *are the optimal values of* $\alpha_{1,\theta}$ *obtained by solving Eq. 6.10 for value of* θ *being* $\theta^{(j)}$ *and* $\theta^{(k)}$. *In other words, the change in optimal value of* $\alpha_{1,\theta}$ *as obtained from solving Eq. 6.11 for a change in* θ *from* $\theta^{(j)}$ *to* $\theta^{(k)}$ *is always an underestimation of the optimal value of* $\alpha_{1,\theta}^{(k)}$ *that results by solving the sub-problem in Eq. 6.10 again at* $\theta^{(k)}$.

Proof. The first part of the statement follows from standard convexity results in parametric nonlinear programming (see *e.g.*, Proposition 2.1 in Fiacco & Kyparisis 1986) while considering in addition that both constraints in Eq. 6.10 are linear in (x_1, u_1). The second part follows due to convexity of $\alpha_{1,\theta}^{(j)}$ in θ and noting that $\alpha_{1,\theta}^{(j)} - \lambda_{1,\theta}^{(j)^T}(\theta - \theta^{(j)})$ is a linear support to optimal $\alpha_{1,\theta}^{(j)}$ at $\theta = \theta^{(j)}$. □

Fig. 6.9 illustrates the intention behind considering above lemma. The bold curve therein shows the variation in optimal value function $\alpha_{1,\theta}^{*}$ as a function of θ with $\hat{v}_1^{(K-1)}$ being constant, while the straight line shows the gradient to $\alpha_{1,\theta}^{*}$ at $\theta = \theta^{(j)}$. By using this gradient, we can obtain an approximate value of $\alpha_{1,\theta}^{*}$ for $\theta = \theta^{(k)}$ to update the optimality cuts in the master problem from previous iterations.

The modified procedure then operates exactly the same as Algorithm 6.1 except the following. At Step 1, the values of parameter θ and the Lagrange multiplier $\lambda_{1,\theta}$ for constraint $A_{1,\theta}x_1 + B_{1,\theta}u_1 = \theta$ from the previous iterations and the current iteration (respectively as $\theta^{(k)}$ and $\lambda_{1,\theta}^{(k)}$, $k \in \mathbf{K}$) are used to construct an update vector $\alpha_{1,updt}^{(k)}, k \in \mathbf{K}$, where $\alpha_{1,updt}^{(k)}$ is calculated as:

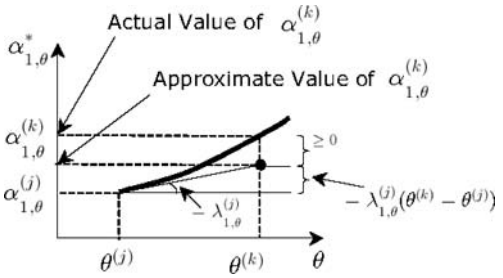

Fig. 6.9. Linear approximation of $\alpha^{*}_{1,\theta}$ as a function of variation in θ

$$\alpha^{(k)}_{1,updt} \triangleq -\lambda^{(k)^T}_{1,\theta}\left(\theta^{(K)} - \theta^{(k)}\right), \quad k \in \mathbf{K}. \tag{6.13}$$

At Step 2, $\alpha^{(k)}_{1,updt}, k \in \mathbf{K}$ is then used to update the optimality cuts in the master problem SP_2 before solving the revised SP_2 as:

$$SP_2^{(K)} \begin{cases} \underset{x_2,u_2,\nu_2}{\text{minimise}} \ \alpha_2 \triangleq \nu_2 + f_2(x_2,u_2) \\ \quad \text{s.t.} \quad \nu_2 \geq \alpha^{(k)}_{1,\theta} + \alpha^{(k)}_{1,updt} + \lambda^{(k)}_1(z^{(k)}_1 - u_{21,in}), \quad k \in \mathbf{K} \\ \qquad\quad A_2x_2 + B_2u_2 = \hat{v}_2, \\ \qquad\quad x_2 \in X_2, u_2 \in U_2, \\ \quad \text{where} \quad \mathbf{K} \triangleq \text{set of iteration indices} \end{cases} \tag{6.14}$$

Compared to Algorithm 6.1, the above modification thus requires calculating $\alpha^{(k)}_{1,updt}, k \in \mathbf{K}$ as a reflection of the change in optimal value function $\alpha_{1,\theta}$ for a change in θ. The modified Algorithm 6.1 is not described here for brevity.

Note that in the above modification we still retain the same multipliers $\lambda^{(k)}_1$, $k \in \mathbf{K}$ in the master problem as before. In fact, the use of sensitivity analysis suggests that $\lambda^{(k)}_1$ also vary with a change in θ. Arguments similar to Lemma 6.4 can be used to obtain a first-order approximation of multipliers (Fiacco 1983). However, our numerical experience (see Section 6.7) indicates that using the same $\lambda^{(k)}_1$ do not lead to a significant problem considering that $-\lambda^{(k)}_1$ denote the sub-gradient of value function $\alpha^{(k)}_{1,\theta}$ for a unit change in $\hat{v}^{(k-1)}_1$. As a result, unless $\lambda^{(k)}_1$ changes widely, the hyperplane $\nu_2 \geq \alpha^{(k)}_{1,\theta} + \alpha^{(k)}_{1,updt} + \lambda^{(k)}_1(A_1x^{(k)}_1 + B_1u^{(k)}_1 - u_{21,in})$ still underestimates the value function $\alpha^{(k)}_{1,\theta}$ as required by the approximate cut update technique.

6.5.3 Solution of Superset Block Problem

The approximate cut update technique can be used to develop a solution algorithm for SPLITTER and MULTIPROD blocks. In particular, for each cus-

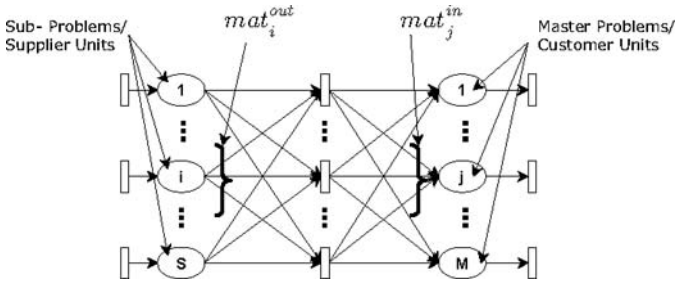

Fig. 6.10. Superset junction block

tomer element j in the second stage of either of these junction blocks, the demands $u_{mi,in}$ from all other customer elements $m \neq j$ can be treated as an uncontrollable parameter vector θ in the sub-problem of the supplier unit. The procedure in the modified Algorithm 6.1 can hence be repeated for all customer elements separately to coordinate the parametric effect of their demand changes onto the supplier element's problem.

Below we build upon this logic by developing a generic algorithm which can be applied to all four junction blocks, in particular the SPLITTER and MULTIPROD blocks. To do so, we consider a *superset* junction block as shown in Fig. 6.10 which captures within it all four types of blocks in Fig. 6.6, *i.e.,* the configuration of any block Fig. 6.6 can be obtained by selecting the appropriate edges and nodes in Fig. 6.10 while deleting the rest. Table 6.1 shows the notation we use to describe the superset block.

Based on the framework of Prob. 6.1, the control problem for superset block can be written as follows:

Problem 6.6 (Superset Junction Block). For $i = 1, \ldots, S$ and $j = 1, \ldots, M$,

$$\begin{aligned}
\underset{\substack{x_i, u_i \\ x_j, u_j}}{\text{minimise}} \; & \sum_{i=1}^{S} f_i(x_i, u_i) + \sum_{j=1}^{M} f_j(x_j, u_j) \\
\text{s.t.} \quad & A_{di} x_i + B_{di} u_i = \sum_{j \in \mathbf{M}_i^d} u_{ji,in}, \quad d \in \text{mat}_i^{out} \\
& A_j x_j + B_j u_j = v_j \\
& A_{i,loc} x_i + B_{i,loc} u_i = 0 \\
& A_{j,loc} x_j + B_{j,loc} u_j = 0 \\
& x_i \in X_i, u_i \in U_i \\
& x_j \in X_j, u_j \in U_j
\end{aligned} \tag{6.15}$$

where $i = 1, \ldots, S$ and $j = 1, \ldots, M$ respectively are the indices of supplier and customer elements. The first constraint represents the links between supplier and customer elements through material streams $d \in \text{mat}_i^{out}$, where A_{di}

Table 6.1. Notation for superset junction block

M	: Number of customer units in the second stage (units indexed by $j \in [1, \ldots, M]$)
S	: Number of supplier units in the first stage (units indexed by $i \in [1, \ldots, S]$)
mat_i^{out}	: Set of outgoing material streams of supplier unit i (streams indexed by $d \in \text{mat}_i^{out}$)
mat_j^{in}	: Set of incoming material streams of customer unit j (streams indexed by $q \in \text{mat}_j^{in}$)
$\mathbf{M}_i$	: Indices of customer units associated with supplier unit i
$\mathbf{S}_j$	: Indices of supplier units associated with customer unit j
$\mathbf{M}_i^d$	: Indices of customer units connected with material stream d in supplier unit i ($\mathbf{M}_i^d \subseteq \mathbf{M}_i$)
$\mathbf{S}_j^q$	: Indices of supplier units connected with material stream q in customer unit j ($\mathbf{S}_j^q \subseteq \mathbf{S}_j$)
K	: Current iteration index in the algorithm
$\mathbf{K}$	: Set of iterations $\{1, \ldots, K\}$
k	: Iteration index, $k \in \{1, \ldots, K\}$

and B_{di} represent the d^{th} row of A_i and B_i. As discussed previously, the variable $u_{ji,in}$ represents the input demand from customer unit j to supplier unit i, while $\hat{v}_j$ is the demand that customer unit j receives from its customer units in the further downstream.□

In what follows, we assume for simplicity that the local constraints $A_{i,loc}x_i + B_{i,loc}u_i = 0$ and $A_{j,loc}x_j + B_{j,loc}u_j = 0$ together with $x_i \in X_i, u_i \in U_i, i = \{1, \ldots, S\}$ and $x_j \in X_j$, $u_j \in U_j, j \in \{1, \ldots, M\}$ in Prob. 6.6 are sufficiently relaxed to absorb any demand imposed on respective unit element, *i.e.,* the sub- or master problems for supplier or customer units do not become infeasible for any v_i and v_j. The assumption, in turn, eliminates the need for generating *feasibility cuts* in the primal decomposition algorithm (see Section A.2, Appendix A). Consequently, to reduce the complexity of the description, we also omit these local constraints and implicitly assume that they always exist.

Algorithm 6.2 describes the generic procedure used for solving the superset block problem. The important step in the algorithm is the calculation of the update terms $\alpha_{ij,updt}(k)$ for individual junction blocks as listed in Table 6.2. For MIXER and MULTIFEED the algorithm operates exactly as per primal decomposition, with a single customer element, therefore the updates are set to 0. For SPLITTER block, the sum of demands $\hat{v}_{mS}$ for all customer elements $j \in [1, \ldots, M], m \neq j$ are treated as the parametric vector θ when referring to a customer element $m \in [1, \ldots, M]$. For MULTIPROD block, the vector of demands $\hat{v}_{1S}, \cdots, \hat{v}_{j-1S}, \hat{v}_{j+1S}, \cdots, \hat{v}_{MS}$ for all customer elements except m is treated as the parameter vector for element m. Thus, for both SPLITTER

Algorithm 6.2 (Superset Block Problem)

Step 0: Initialise *Set* $K := 1$. *Given* $v_j = \hat{v}_j, j = 1, \ldots, M$, *the demands for all second stage units. Assume an initial value of* $u_{ji,in} \equiv u_{ji,in}^{(0)}$. *Set* $\hat{v}_{ji}^{(0)} = u_{ji,in}^{(0)}$ *for all* $i \in \boldsymbol{S}_j^q$, $q \in mat_j^{in}$, $j = 1, \ldots, M$.

Step 1: Sub-problem $SP_i^{(K)}, i = 1, \ldots, S$: *At any iteration* K, *solve unit* i*'s problem*

$$SP_i^{(K)} \quad \left\{ \begin{array}{ll} \underset{x_i, u_i}{minimise}\ \alpha_i \triangleq f_i(x_i, u_i) & \\ s.t. \quad A_{di}x_i + B_{di}u_i = \sum\limits_{j \in \boldsymbol{M}_i^d} \hat{v}_{ji}^{(K-1)}, & d \in mat_i^{out} \end{array} \right. \tag{6.16}$$

to obtain the optimal values of $x_i^{(K)}$, $u_{i,in}^{(K)}$, *Lagrange multipliers* $\lambda_i^{(K)}$ *for linking equality constraints, and the objective function* $\alpha_i^{(K)}$.

Set $z_{ij}^{(K)} \equiv \hat{v}_{ji}^{(K-1)}$, $j \in \boldsymbol{M}_i^d$, $d \in mat_i^{out}$, *i.e., assume the demands from all master problems for all material streams* $d \in mat_i^{out}$ *is satisfied.*

Step 2: Master Problem $SP_j^{(K)}, j = 1, \ldots, M$: *Use* $\alpha_i^{(K)}, \lambda_{ij}^{(K)}$ *and* $z_{ij}^{(K)}$ *to form a new optimality cut in unit* j*'s problem* $SP_j^{(K)}$. *Solve the resulting problem*

$$SP_j^{(K)} \quad \left\{ \begin{array}{ll} \underset{x_j, u_j, \nu_j}{minimise}\ \alpha_j \triangleq \nu_j + f_j(x_j, u_j) & \\ s.t. \quad \nu_j \geq \sum\limits_{i \in \boldsymbol{S}_j} \left\{ \alpha_{ij}^{(k)} + \alpha_{ij,updt}^{(k)} + \lambda_{ij}^{(k)}(z_{ij}^{(k)} - u_{ji,in}) \right\}, & k \in \boldsymbol{K} \\ \qquad A_j x_j + B_j u_j = \hat{v}_j & \end{array} \right. \tag{6.17}$$

to obtain the solution $u_{ji,in}^{(K)}$ *and* $x_j^{(K)}$. *The* $\alpha_{ij,updt}^{(k)}$ *are the approximate optimality cut updates used for updating the master problem* j *for perturbations in the demands from remaining other master problems* $m \in [1, \ldots, M], m \neq j$. *The updates are calculated by the first stage units* $i = 1, \ldots, S$ *and passed to the second-stage units* $j = 1, \ldots, M$. *Table 6.2 describes the specific formulation of these updates for all four junction blocks.*

Step 3: Iterate/Terminate: *Terminate if the convergence criteria is satisfied, which is considered here as, for a given* $\epsilon > 0$ *and* $i = 1, \ldots, S, j = 1, \ldots, M$,

$$\left\| \left\{ x_i^{(K)}, u_i^{(K)}, x_j^{(K)}, u_j^{(K)} \right\} - \left\{ x_i^{(K-1)}, u_i^{(K-1)}, x_j^{(K-1)}, u_j^{(K-1)} \right\} \right\| \leq \epsilon \tag{6.18}$$

i.e., the solutions of sub-problems $SP_i^{(K)}$, $i = 1, \ldots, S$ *and master problems* $SP_j^{(K)}$, $j = 1, \ldots, M$ *converge to a fixed point with tolerance* ϵ. *Else, set* $\hat{v}_{ji}^{(K)} = u_{ji,in}^{(K)}$, $K := K + 1$, *and return to Step 1.* □

---

and MULTIPROD blocks, a parametric update vector $\alpha_{Sj,updt}^{(k)}$ is calculated at each iteration for each of the customer elements $m = 1, \ldots, M$. In summary, Algorithm 6.2 provides a solution strategy for solving the two-stage control problems for each of the junction blocks in a distributed manner.

Table 6.2. Approximate optimality cut updates for junction blocks

MIXER	$\alpha_{iM,updt}^{(k)} = 0$
MULTIFEED	$\alpha_{iM,updt}^{(k)} = 0$
SPLITTER	$\alpha_{Sj,updt}^{(k)} \triangleq \lambda_S^{(k)} \left(\sum_{\substack{m\in[1,\ldots,M] \\ m\neq j}} \hat{v}_{mS}^{(k-1)} - \sum_{\substack{m\in[1,\ldots,M] \\ m\neq j}} \hat{v}_{mS}^{(K-1)} \right), \quad k \in \mathbf{K}$ (6.19) where S is the single supplier unit, j is the customer unit for which the update is being calculated, and M is the last customer unit.
MULTIPROD	$\alpha_{Sj,updt}^{(k)} \triangleq \left[\lambda_{S1}^{(k)}, \cdots, \lambda_{Sj-1}^{(k)}, \lambda_{Sj+1}^{(k)}, \cdots, \lambda_{SM}^{(k)}\right] \cdot \left[\left(\hat{v}_{1S}^{(k-1)}, \cdots, \hat{v}_{j-1S}^{(k-1)}, \hat{v}_{j+1S}^{(k-1)}, \cdots, \hat{v}_{MS}^{(k-1)}\right)^T - \left(\hat{v}_{1S}^{(K-1)}, \cdots, \hat{v}_{j-1S}^{(K-1)}, \hat{v}_{j+1S}^{(K-1)}, \cdots, \hat{v}_{MS}^{(K-1)}\right)^T\right], \quad k \in \mathbf{K}$ (6.20) where S is the single supplier unit, j is the customer unit for which the update is being calculated, and M is the last customer unit.

6.6 Solution of the Multi-Stage Problem

As the final step to solving Prob. 6.1, we need to ensure that all of the interconnected two-level junction block problems are solved iteratively in an appropriate sequence.

In what follows, we consider solving a sequence of three problems to develop the strategy in a constructive manner.

i. N-Units problem involving N series-connected units (Section 6.6.1)
ii. N-Units problem with an uncontrolled parameter in one or more of unit problems (Section 6.6.2)
iii. Main distributed control problem – Prob. 6.1 (Section 6.6.3)

6.6.1 Solution of N-Units Problem

We consider first an extension of the Two-Units problem to an N-Units problem as comprising N series-connected unit elements.

Problem 6.7 (N-Units Problem). For $i = 1, \ldots, N$

$$\begin{array}{ll} \underset{\substack{u_i, x_i \\ i=1,\ldots,N}}{\text{minimise}} & \sum_{i=1}^{N} f_i(u_i, x_i) \\ \text{s.t.} & A_i x_i + B_i u_i = v_i \quad i = 1, \ldots, N \\ & x_i \in X_i, u_i \in U_i \quad i = 1, \ldots, N. \end{array} \tag{6.21}$$

where $v_i \equiv y_{i+1} = u_{i+1i,in}$ represents the demand to unit $i = 1, \ldots, N-1$. □

A nested decomposition algorithm based on primal decomposition simply extends Algorithm 6.1 to repeatedly solve a sequence of $N-1$ two-units problems, one corresponding to each link between two successive unit elements. The sub-problem for each unit $i = 2, \ldots, N$ is considered as a master problem for a composite problem comprising all predecessor units 1 to $i-1$. The combined problem of units 1 to $i-1$ and unit i then becomes the sub-problem for unit $i+1$, and so on, until unit N is reached. The sub-problems of units $i = 1, \ldots, N$ are thus solved sequentially to construct a new optimality cut in the immediate next master problem. Once a complete iteration is finished, the procedure repeats starting from unit 1. In reference to Algorithm 6.1, the formulation of sub-problem SP_i at iteration K for $i = 2, \ldots, N$ along the sequence becomes as follows. We do not describe the complete algorithm for brevity.

Step i: Sub-problem SP_i, $i = 2, \ldots, N$: Use $\alpha_{i-1}^{(K)}, \lambda_{i-1}^{(K)}$ and $z_{i-1}^{(K)}$ to construct a new optimality cut in unit i's problem $SP_i^{(K)}$. Solve the resulting problem as:

$$SP_i^{(K)} \quad \left\{ \begin{array}{ll} \underset{x_i, u_i, \nu_i}{\text{minimise}} & \alpha_i \triangleq \nu_i + f_i(x_i, u_i) \\ \text{s.t.} & \nu_i \geq \alpha_{i-1}^{(k)} + \lambda_{i-1}^{(k)}(z_{i-1}^{(k)} - u_{ii-1,in}), \quad k \in \mathbf{K} \\ & A_i x_i + B_i u_i = \hat{v}_i^{(K-1)}, \\ & x_i \in X_i, u_i \in U_i \\ \text{where} & \mathbf{K} \triangleq \text{set of iterations} \end{array} \right. \tag{6.22}$$

Set $z_i^{(K)} \equiv A_i x_i^{(K)} + B_i u_i^{(K)}$. Pass $\alpha_i^{(K)}, \lambda_i^{(K)}$ and $z_i^{(K)}$ to unit $i+1$.

6.6.2 Solution of N-Units Parametric Problem

We next consider an extension of the N-Units problem where the sub-problem of unit element 1 now contains an additional parameter vector $\theta \in \Theta$ while the

remaining sub-problems $SP_i, i = 2, \ldots, N$ are same as in Prob. 6.7. Formally, the modified problem is:

Problem 6.8 (N-Units Problem with Parametric $SP_{1,\theta}$).

$$\begin{array}{ll} \underset{\substack{x_i,u_i \\ i=1,\ldots,N}}{\text{minimise}} & \sum_{i=1}^{N} f_i(x_i, u_i) \\ \text{s.t.} & A_i x_i + B_i u_i = v_i \quad i = 1, \ldots, N \\ & A_{1,\theta} x_1 + B_{1,\theta} u_1 = \theta \quad i = 1 \\ & x_i \in X_i, u_i \in U_i \quad i = 1, \ldots, N. \end{array} \tag{6.23}$$

where $v_i \equiv y_{i+1} = C_{i+1} x_{i+1} + D_{i+1} u_{i+1}$ represents the interaction variable to unit i, $i = 1, \ldots, N-1$.

The solution of the above problem is faced with the same challenge because the parametric Two-Units problem (Prob. 6.3) as the changes in θ in SP_1 results in a non-unique response to SP_2, and therefore, the non-unique response of all subsequent sub-problems to their immediate next master problem. Fortunately, an extension of the approximate cut update technique provides a method to resolve this. In this extension, we simply pass the update vector $\alpha^{(\mathbf{K})}_{1,updt}$ (of dimension K) from unit 1 to all units $i = 2, \ldots, N$ to modify the optimality cuts in their problems SP_i in a similar manner to sub-problem SP_2 in solving the parametric Two-Units problem. In the modified form of N-units algorithm, the sub-problem SP_i at iteration K is solved as follows.

Step i: Sub-problem SP_i, $i = 2, \ldots, N$: For each $k \in \mathbf{K}$, use $\alpha^{(\mathbf{K})}_{1,updt}$ to calculate $\alpha^{(\mathbf{K})}_{i-1,mod} \triangleq \alpha^{(\mathbf{K})}_{i-1} + \alpha^{(\mathbf{K})}_{1,updt}$, and solve the modified SP_i in Eq. 6.22, $i = 2, \ldots, N$ as:

$$SP_i^{(K)} \begin{cases} \underset{u_i, x_i, \nu_i}{\text{minimise}} & \alpha_i \triangleq \nu_i + f_i(u_i, x_i) \\ \text{s.t.} & \nu_i \geq \alpha^{(k)}_{i-1,mod} + \lambda^{(k)}_{i-1}(z^{(k)}_{i-1} - y_i), k \in \mathbf{K} \\ & A_i x_i + B_i u_i = \hat{v}_i^{(K-1)} \rightsquigarrow \lambda_i^{(K)} \\ & x_i \in X_i, u_i \in U_i \\ \text{where} & y_i \equiv u_{ii-1,in} \end{cases} \tag{6.24}$$

Note that we use the same update vector $\alpha^{(\mathbf{K})}_{1,updt}$ to all update all unit sub-problems SP_i, $i = 2, \ldots, N$. This communication of update vector can be made symmetric by assigning $\alpha^{(\mathbf{K})}_{2,updt} \triangleq \alpha^{(\mathbf{K})}_{1,updt}$ and repeating $\alpha^{(\mathbf{K})}_{i,updt} \triangleq \alpha^{(\mathbf{K})}_{i-1,updt}$ for all $i = 3, \ldots, N$, and using $\alpha^{(\mathbf{K})}_{i-1,mod} \triangleq \alpha^{(\mathbf{K})}_{i-1} + \alpha^{(\mathbf{K})}_{i-1,updt}$.

We consider next a further extension where instead of just sub-problem SP_1, one or more of other sub-problems are also parameterized with parameter vectors $\theta_i \in \Theta_i$. The approximate cut update technique can as well be extended to solve this problem in an analogous manner.

For SP_{1,θ_1} we still continue to propagate the approximate cut update $\alpha^{(\mathbf{K})}_{1,updt}$ to all units $i = 2, \ldots, N$. In addition, for any sub-problem SP_{i,θ_i}, $i = 2, \ldots, N-1$, we also generate a separate approximate cut update that reflects the effects of parametric variations in its parameter vector θ_i. This can be written as: $\alpha^{(k)}_{i,loc,updt} \triangleq -\lambda^{(k)^T}_{i,\theta_i}(\theta^{(K)}_i - \theta^{(k)}_i)$, $k \in \mathbf{K}$ where λ_{i,θ_i} is the Lagrange multiplier associated with the constraint $A_{i,\theta_i}x_i + B_{i,\theta_i}u_i = \theta_i$.

We then simply add this update to the update received from previous sub-problem $SP_{i-1,\theta_{i-1}}$ and pass the composite update to the next sub-problem $SP_{i+1,\theta_{i+1}}$, *i.e.,* $\alpha^{(\mathbf{K})}_{i,updt} \triangleq \alpha^{(\mathbf{K})}_{i-1,updt} + \alpha^{(\mathbf{K})}_{i,loc,updt}$. With this modification the sub-problem $SP^{(K)}_{i,\theta_i}$, $i = 2, \ldots, N$ at iteration K is now solved as follows.

Calculate $\alpha^{(\mathbf{K})}_{i-1,mod} \triangleq \alpha^{(\mathbf{K})}_{i-1} + \alpha^{(\mathbf{K})}_{i-1,updt}$. Use $\alpha^{(\mathbf{K})}_{i-1,mod}$ in solving

$$SP^{(K)}_i \begin{cases} \underset{u_i,x_i,\nu_i}{\text{minimise}} \; \alpha_i \triangleq \nu_i + f_i(u_i, x_i) \\ \quad \text{s.t.} \quad \nu_i \geq \alpha^{(k)}_{i-1,mod} + \lambda^{(k)}_{i-1}(z^{(k)}_{i-1} - y_i), k \in \mathbf{K} \\ \qquad A_i x_i + B_i u_i = \hat{v}^{(K-1)}_i, \rightsquigarrow \lambda^{(K)}_i \\ \qquad A_{i,\theta_i}x_i + B_{i,\theta_i}u_i = \theta^{(K)}_i \rightsquigarrow \lambda^{(K)}_{i,\theta_i} \\ \qquad x_i \in X_i, u_i \in U_i \\ \quad \text{where} \quad y_i \equiv u_{ii-1,in} \end{cases} \tag{6.25}$$

Using $\lambda^{(K)}_{i,\theta_i}$, calculate the approximate cut update to be passed to the next unit $i+1$ as $\alpha^{(\mathbf{K})}_{i,updt} \triangleq \alpha^{(\mathbf{K})}_{i-1,updt} + \alpha^{(\mathbf{K})}_{i,loc,updt}$ where

$$\alpha^{(k)}_{i,loc,updt} \triangleq \lambda^{(K)^T}_{i,\theta_i}(\theta^{(K)}_i - \theta^{(k)}_i), k \in \mathbf{K}$$

6.6.3 Solution of Distributed Control Problem (Prob. 6.1)

We can now develop the distributed coordination algorithm for solving main distributed problem (Prob. 6.1) for an arbitrary, acyclic process network. To define the interactions between unit elements more systematically, we first develop an indexing of unit elements in the P-Graph.

Assume the process contains S different stages with each stage containing possibly one or more unit elements. The word *stage* refers to a typical processing task, such as the primary reaction, separation, *etc.* Each such stage may contain a number of unit elements of similar processing capabilities. We then use the following procedure to assign an index (n, s) to all unit elements.

We first assign the stage index 1 to all unit elements that use the main raw-materials as their feedstocks, *i.e.,* the input-set mat^{in} for these unit elements comprise only the main raw-materials. All unit elements connected to these first stage unit elements are then assigned the stage index 2. The assignment is thus repeated until the unit elements in the terminal stage reached whose output-set mat^{out} comprise only the end-products. These elements receive the stage index S. In this process of assigning stage indices, if an element receives

two or more different indices, because of it being connected to unit elements from two or more stages, then the highest stage index among all is used. Next, within each stage, the units are numbered from 1 to a maximum value (called N_s) that depends on the unit elements contained in that stage. We use a simple rule of progressing from top to bottom in the P-Graph to define this unit number. At the end of this indexing, each unit element thus receives an index (n, s) where s refers to the stage index and n as the number of the unit element within that stage.

We next use the notation described in Table 6.3 to define the solution procedure for Prob. 6.1. Note that for any stage s, S_s^- and S_s^+ denote the stages preceding and succeeding to stage s and comprise unit elements linked to any unit element in stage s. Note also that as per above indexing rules at least one unit in any stage s must be linked to stage $s-1$ as well as stage $s+1$. Unit elements in stage s may also be linked to other stages in S_s^- and S_s^+. The set $\mathbf{M}_{(n,s)}$ hence encompasses the indices of unit elements in stages S_s^+ which are connected to unit element (n, s) in stage s. The set $\mathbf{M}^d_{(n,s)}$ then denotes the subset of unit elements within $\mathbf{M}_{(n,s)}$ that are connected to element (n, s) through the material stream $d \in \mathrm{mat}^{out}_{(n,s)}$. Similar interpretation can be given for $\mathbf{S}_{(n,s)}$ and $\mathbf{S}^q_{(n,s)}$.

The nested solution procedure for Prob. 6.1 extends the algorithm for superset block (Algorithm 6.2) by using the results from parametric N-unit problems from the previous subsection. Algorithm 6.3 describes this distributed algorithm. As expected, the important step in the algorithm is to compute the approximate cut update terms $\alpha^{(K)}_{(n,s)j,updt}$ to be passed between stages while taking into account the specific type of junction block by which the associated unit element is connected to other unit elements.

Table 6.3. Notation for distributed coordination algorithm

S	: Number of process stages (stages indexed by $s \in [1, \ldots, S]$)
$\mathbf{S}_s^-$	: Indices of stages preceding to stage s
$\mathbf{S}_s^+$	: Indices of stages succeeding to stage s
N_s	: Number of units in stage n (units indexed by $n \in [1, \ldots, N_s]$)
(n, s)	: Index of unit n in stage s, $n \in [1, \ldots, N_s]$, $s \in [1, \ldots, S]$
$\mathrm{mat}^{out}_{(n,s)}$	: Set of outgoing material streams of unit (n, s) (indexed by $d \in \mathrm{mat}^{out}_{(n,s)}$)
$\mathrm{mat}^{in}_{(n,s)}$	: Set of incoming material streams of unit (n, s) (indexed by $q \in \mathrm{mat}^{in}_{(n,s)}$)
$\mathbf{M}_{(n,s)}$	: Indices of units in $\mathbf{S}_s^+$ connected with unit (n, s)
$\mathbf{M}^d_{(n,s)}$	: Indices of units in $\mathbf{S}_s^+$ connected with outgoing stream d of unit (n, s) ($\mathbf{M}^d_{(n,s)} \subseteq \mathbf{M}_{(n,s)}$)
$\mathbf{S}_{(n,s)}$	: Indices of units in $\mathbf{S}_s^-$ connected with unit (n, s)
$\mathbf{S}^q_{(n,s)}$	: Indices of units in $\mathbf{S}_s^-$ connected with incoming stream q of unit (n, s) ($\mathbf{S}^q_{(n,s)} \subseteq \mathbf{S}_{(n,s)}$)

Algorithm 6.3 (Distributed Coordination Algorithm)

Step 0: *Set* $K := 1$. *Given* $v_{(n,S)} = \hat{v}_{(n,S)}$, *the demands for unit elements* (n,S) , $n = 1,\ldots,N_S$ *in the terminal stage* S. *For all* $s = [1,\ldots,S]$ *and* $n = [1,\ldots,N_s]$, *assume an initial value of* $u_{(n,s)i,in} \equiv u^{(0)}_{(n,s)i,in}$, *and for all* $i \in \boldsymbol{S}_{(n,s)}$ *set* $\hat{v}^{(0)}_{(n,s)i} = u^{(0)}_{(n,s)i,in}$.

Step s, $s = 1,\ldots,S$: ***Sub-problem*** $SP^{(K)}_{(n,s)}, n = 1,\ldots,N_s$: *Use* $\alpha^{(K)}_{i(n,s)}$, $z^{(K)}_{i(n,s)}$, $i \in \boldsymbol{S}_{(n,s)}$ *to form a new optimality cut in unit* (n,s)*'s problem* $SP^{(K)}_{(n,s)}$. *Solve the resulting problem*

$$SP^{(K)}_{(n,s)} \left\{ \begin{array}{ll} \underset{x_{(n,s)},u_{(n,s)},}{\underset{\nu_{(n,s)}}{minimise}} & \alpha_{(n,s)} \triangleq \nu_{(n,s)} + f_{(n,s)}(x_{(n,s)},u_{(n,s)}) \\ s.t. & \nu_{(n,s)} \geq \sum\limits_{i \in \boldsymbol{S}_{(n,s)}} \Big\{ \alpha^{(k)}_{i(n,s)} + \alpha^{(k)}_{i(n,s),updt} + \\ & \qquad \lambda^{(k)}_{i(n,s)} (z^{(k)}_{i(n,s)} - u_{(n,s)i,in}) \Big\}, \quad k \in \boldsymbol{K} \\ & A_{d(n,s)} x_{(n,s)} + B_{d(n,s)} u_{(n,s)} = \sum\limits_{j \in \boldsymbol{M}^d_{(n,s)}} \hat{v}^{(K-1)}_{j(n,s)}, \quad d \in mat^{out}_{(n,s)} \end{array} \right. \tag{6.26}$$

to obtain the optimal values of $x^{(K)}_{(n,s)}, u^{(K)}_{(n,s)i,in}$, *optimal Lagrange multipliers* $\lambda^{(K)}_{(n,s)}$ *associated with the linking equality constraints and the optimal objective value* $\alpha^{(k)}_{(n,s)}$. *Note that* $A_{d(n,s)}$ *and* $B_{d(n,s)}$ *denote the* d^{th} *row in constraint matrices* $A_{(n,s)}$ *and* $B_{(n,s)}$ *associated with unit* (n,s). *The solution* $u^{(K)}_{(n,s)i,in}$ *defines the demands sent by unit* (n,s) *to all linked supplier elements* $\boldsymbol{S}_{(n,s)}$ *at the next iteration.*

Take $z^{(K)}_{(n,s)j} \equiv \hat{v}^{(K-1)}_{j(n,s)}$, $j \in \boldsymbol{M}^d_{(n,s)}$, $d \in mat^{out}_{(n,s)}$. *Set* $\lambda^{(K)}_{(n,s)j} \equiv \lambda^d_{(n,s)}, \forall j \in \boldsymbol{M}^d_{(n,s)}$ *where* $\lambda^d_{(n,s)}$ *denotes the* d^{th} *element in* $\lambda^{(K)}_{(n,s)}$. *Calculate the aggregate cut update* $\alpha_{(n,s)j,updt}$ *as,*

$$\alpha^{(\boldsymbol{K})}_{(n,s)j,updt} \triangleq \sum_{i \in \boldsymbol{S}^q_{(n,s)}} \Big\{ \alpha^{(\boldsymbol{K})}_{i(n,s),updt} \Big\} + \alpha^{(\boldsymbol{K})}_{(n,s)j,loc,updt} \tag{6.27}$$

where $\alpha^{(\boldsymbol{K})}_{(n,s)j,loc,updt}$ *denotes the approximate cut updates that unit element* (n,s) *generates locally for parametric demand variations from customer elements* $j \in \boldsymbol{M}_{(n,s)}$ *while* $\alpha^{(\boldsymbol{K})}_{i(n,s),updt}$ *are the approximate cut updates that it receives from supplier elements* $i \in \boldsymbol{S}_{(n,s)}$.

Step N+1: Iterate/Terminate: *Terminate if the specific convergence criteria is satisfied, which is considered as, for a given* $\epsilon > 0$ *and for all* $s = [1,\ldots,S]$ *and* $n = [1,\ldots,N_s]$,

$$\left\| \Big\{ x^{(K)}_{(n,s)}, u^{(K)}_{(n,s)} \Big\} - \Big\{ x^{(K-1)}_{(n,s)}, u^{(K-1)}_{(n,s)} \Big\} \right\| \leq \epsilon \tag{6.28}$$

Else, set $\hat{v}^{(K)}_{(n,s)i} = u^{(K)}_{(n,s)i,in}$, $K := K+1$ *and return to Step 1.* □

Note the bidirectional nature of information flow between unit elements in Algorithm 6.3 – the flowrate demands for feedstocks (in the form of $u^{(K)}_{(n,s)i,in}$) flow backwards and the resulting supply proposals (in the form of optimality cuts) flow forwards.

6.7 Implementation and Numerical Examples

The distributed coordination algorithm (Algorithm 6.3) developed in the previous section was implemented using MATLAB® software, the details of the implementation are provided in Appendix C. In what follows, we simply describe a few numerical examples to illustrate the different features of the algorithms discussed in the previous sections. An application of Algorithm 6.3 to an industrial-scale, multipurpose process problem is discussed in the next chapter.

Example 6.9. The first example illustrates the approximate optimality cut update technique described in Section 6.5.2. Consider the following problem.

$$\begin{array}{ll} \underset{x_{1,1},x_{1,2},x_{1,3},x_2}{\text{minimise}} & x_{1,1}^2 + x_{1,2}^2 + x_{1,3}^2 + x_2^2 + x_{1,1} + x_{1,2} + x_{1,3} \\ \text{s.t.} & x_{1,1} + x_{1,2} + x_{1,3} + x_2 = 3 \\ & 2.5x_{1,1} + 0.7x_{1,2} + 1.6x_{1,3} + \theta = 5 \end{array} \tag{6.29}$$

where $x_{1,\cdot}$ and x_2 refer to the local variables of unit elements 1 and 2, while θ is the parameter vector local to unit element 1. The formulation of the associated sub-problems at iteration K in the form of Algorithm 6.1 can be written as follows:

$$SP^{(K)}_{1,\theta} \left\{ \begin{array}{ll} \underset{x_{1,1},x_{1,2},x_{1,3}}{\text{minimise}} & \alpha_1 \triangleq x_{1,1}^2 + x_{1,2}^2 + x_{1,3}^2 + x_{1,1} + x_{1,2} + x_{1,3} \\ \text{s.t.} & x_{1,1} + x_{1,2} + x_{1,3} = 3 - \hat{x}_2^{(K-1)} \rightsquigarrow \lambda_1^{(K)} \\ & 2.5x_{1,1} + 0.7x_{1,2} + 1.6x_{1,3} + \theta = 5 \end{array} \right. \tag{6.30}$$

and

$$SP^{(K)}_{2} \left\{ \begin{array}{ll} \underset{\nu_2,x_2}{\text{minimise}} & \alpha_2 \triangleq \nu_2 + x_2^2 \\ \text{s.t.} & \nu_2 \geq \alpha_1^{(k)} + \lambda_1^{(k)}(x_{1,1}^{(k)} + x_{1,2}^{(k)} + x_2 - 3), k \in \mathbf{K} \end{array} \right. \tag{6.31}$$

where λ_1 is the Lagrange multiplier. Assume $\theta = 1$ when the algorithm is started and that it changes to $\theta = 5$ at iteration 5 and remains the same thereafter. If the approximate cut updates are not used when θ changes, then the master problem will retain the optimality cuts from previous iterations and result in an incorrect solution. Fig. 6.11 illustrates this effect by comparing the value function α_2 as a function of x_2 for three different scenarios: Case

A – when no updates are used; Case B – when the algorithm is restarted at iteration 6 after θ is changed; and, Case C – when the approximate cut updates are applied. Note that the parametric effect results in an incorrect solution in case A, while the use of approximate cut updates in case C gives the same result as case B where the algorithm is restarted after the change in θ.

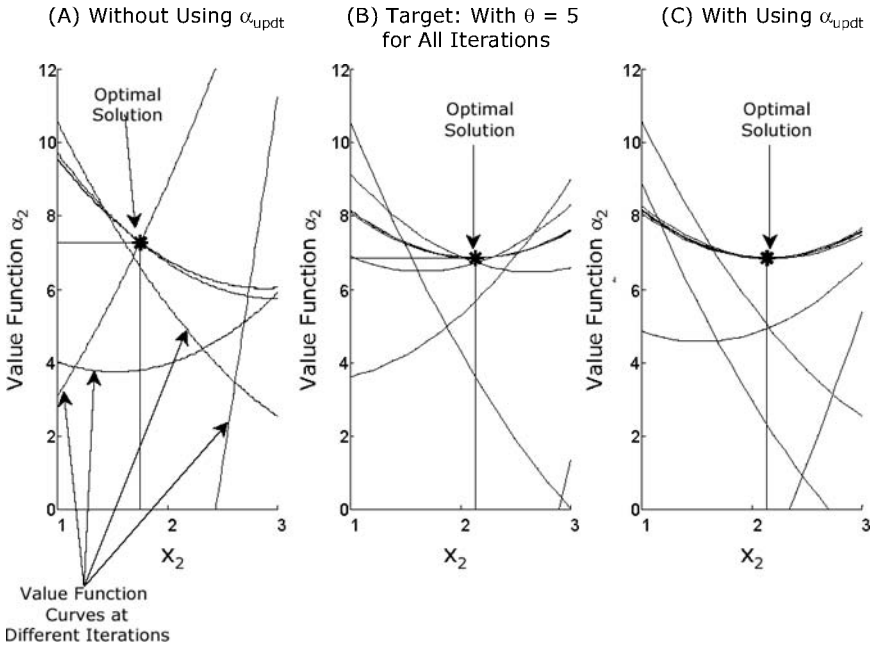

Fig. 6.11. Example 6.9: Effect of using approximate cut updates α_{updt} – (A): Without α_{updt} results in an incorrect solution, (B): Target solution – restarting the algorithm with $\theta = 5$, (C): With α_{updt} – results in a correct Solution

The next four examples refer to application of the Superset block algorithm (Algorithm 6.2) to four different process networks shown in Fig. 6.12. The data for matrices Q_i, A_i, $A_{i,loc}$ and vectors c_i, B_i, $B_{i,loc}$ for these examples are given in Appendix C.

Example 6.10. (MIXER) The first example refers to a MIXER block with three unit elements. Table C.2 in Appendix C shows the progress of iterations during an execution of Algorithm 6.2. The variables $u_{(n,s,z)}$ and $x_{(n,s,z)}$ therein refer to the z^{th} element of the input and state vectors of sub-problem $SP_{(n,s)}$, while the last column shows the solution obtained for overall problem using a centralised algorithm. As can be verified, the solution obtained via distributed

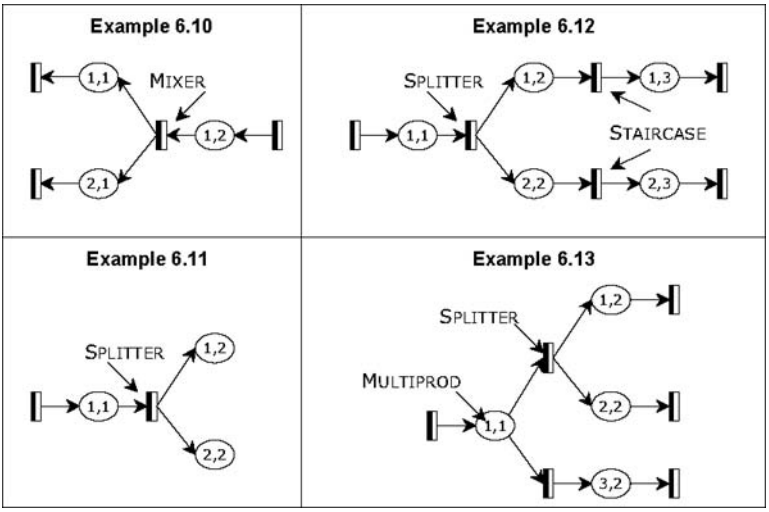

Fig. 6.12. Process configuration for Examples 6.10 to 6.13

algorithm matches with that from centralised algorithm within a predefined tolerance.

Example 6.11. (SPLITTER) The next problem refers to a SPLITTER block with three unit elements as shown in Fig. 6.12. Note that for unit $(1,1)$, the demand from unit $(2,2)$ becomes the uncontrollable parameter θ when referred to $(1,2)$. Similarly, the demand from unit $(1,2)$ becomes the parameter θ when referred to unit $(2,2)$. The approximate cut update technique resolves these parametric effects by updating the optimality cuts in the sub-problems of units $(1,2)$ and $(2,2)$. Table C.4 in Appendix C shows the progress of iterations for Algorithm 6.2, where again the last column confirms that the resulting solution matches with that from centralised algorithm within a predefined tolerance.

To demonstrate an additional feature of approximate cut updates, we change the terminal demands from 10 to 20 deviation units for units $(1,2)$ and $(2,2)$ while the execution of the algorithm is in progress. Fig. 6.13 shows that the algorithm is able to pick up the change and converge to a new optimum.

Example 6.12. (SPLITTER-STAIRCASE) The next example illustrates the nesting of junction blocks in a SPLITTER -STAIRCASE process, where the STAIRCASE block refers to a Two-Units process as a special case of the MIXER or MULTIFEED blocks.

As per the information flow described in Section 6.3, the demand $\hat{v}_{(1,3)(1,2)}$ from unit $(1,3)$ to $(1,2)$ propagates backwards to demand $\hat{v}_{(1,2)(1,1)}$ that unit $(1,2)$ sends to unit $(1,1)$. The same applies to units $(2,3)$ and $(2,2)$. The demands from terminal units $(1,3)$ and $(2,3)$ thus parameterize the sub-problem

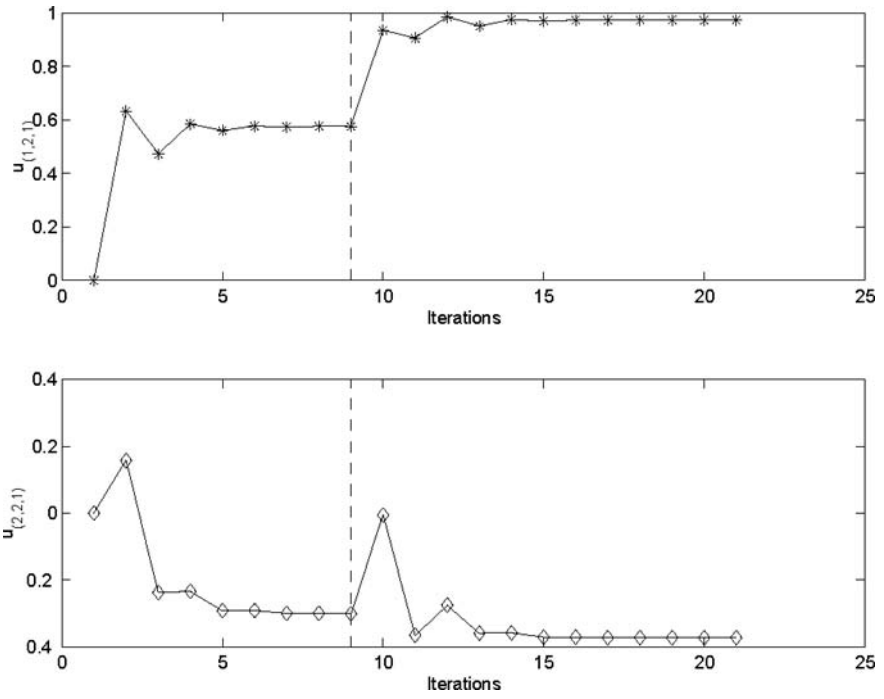

Fig. 6.13. Example 6.11: Effects of change in terminal demands for units $(1,2)$ and $(2,2)$

of unit $(1,1)$. The use of propagation of approximate cut updates along the network resolves these parametric effects by updating the optimality cuts in all four units $(1,2)$, $(1,3)$, $(2,2)$ and $(2,3)$ as described in Algorithm 6.3. Table C.6 in Appendix C summarises the progress of iterations and a comparison with the equivalent centralised solution.

Fig. 6.14, similar to Fig. 6.13, shows the ability of the algorithm to pick up a change in demands for units $(1,3)$ and $(2,3)$ from 10 to 20 deviation units. The plots refer to the input demands $u_{(1,2,1)}, u_{(2,2,1)}, u_{(1,3,1)}, u_{(2,3,1)}$ that the second and third stage units $(1,2), (2,2)$, $(1,3)$ and $(2,3)$ request from their supplier units. As can be seen, the algorithm converges to a new optimum.

Example 6.13. (MULTIPROD-SPLITTER) The final example illustrates the nesting of a MULTIPROD block with a SPLITTER block. The example shows the combined parametric effects from MULTIPROD and SPLITTER blocks within a single problem. The demand from unit $(3,2)$ becomes the parameter θ for unit $(1,1)$ when referring to a combined problem of units $(1,2)$ and $(2,2)$. Units $(1,2)$ and $(2,2)$ thus both receive the same approximate cut update of the MULTIPROD type (Eq. 6.20) for demand variations from unit $(1,3)$. Unit $(3,2)$ similarly also receives a cut update of MULTIPROD type (Eq. 6.20)

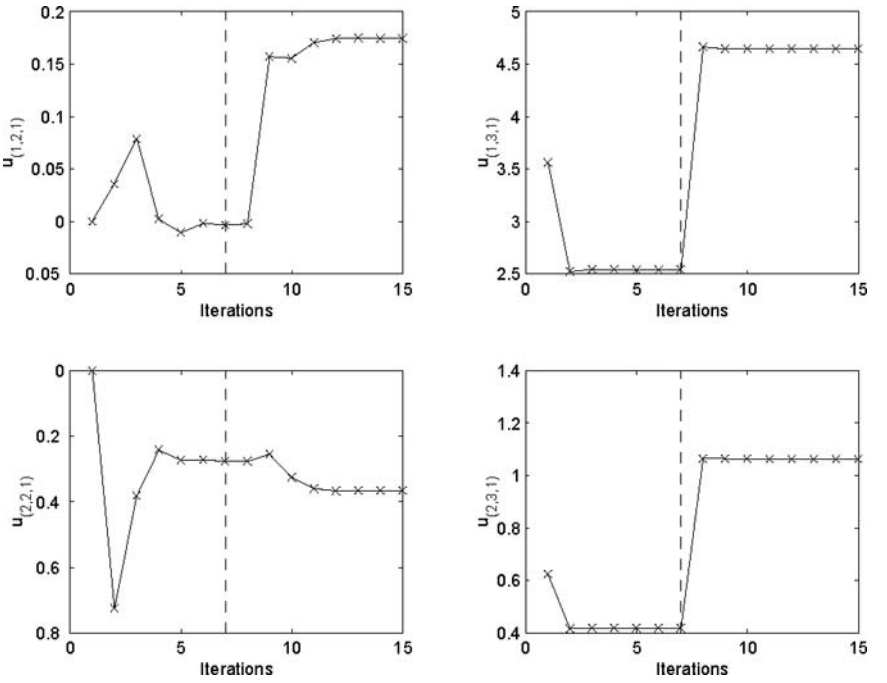

Fig. 6.14. Example 6.12: Effects of change in terminal demands for units $(1,3)$ and $(2,3)$

for a combined demand from units $(1,2)$ and $(2,2)$. In addition, the demands from units $(1,2)$ and $(2,2)$ also parameterize unit $(1,1)$'s sub-problem for each other's demand. Hence, these two units also receive an additional SPLITTER type cut update of the form Eq. 6.19.

Tables C.8 in Appendix C summarises the progress of iterations and the convergence to the optimal solution obtained by a centralised algorithm.

6.8 Future Extensions

In the course of developing the distributed algorithm, we made various assumptions that helped us simplify the discussions. The algorithm can be extended and generalised to a wider class of problems if one or more of these assumptions are relaxed. For example:

- *Infeasible Sub-problems:* The assumption that constraints $xi \in X_i$, $u_i \in U_i$ or $A_{i,loc}x_i + B_{i,loc}u_i = 0$ are sufficiently relaxed to allow unit i to accept any product demand $\hat{v}_i$ from downstream units can be relaxed by using so-called *feasibility restoration technique* for primal decomposition concept (Grothey, Leyffer & McKinnon 1999).

- *Incorporating Linking Inequality Constraints:* Apart from equality constraints $A_i x_i + B_i u_i = \hat{v}_i$, one can also include inequalities such as sharing of a limited quota of services (*e.g.*, energy flow) linking multiple units.
- *Multiple Demand Variables:* The case of singleton demand in $\hat{v}_i$ can be extended to a vector-valued demand, *e.g.*, a trajectory of demand variations in time domain in the context of an optimal control problem.
- *Recycle and By-products:* The case of recycle or by-products can be considered. This requires further analysis as the unit elements acting as the customers along recycle are now situated upstream in the process. An approach based on classical research in process flowsheeting (Westerberg *et al.* 1979) can be considered in which the interactions along recycles are coordinated separately by breaking the recycle loop and treating the unit elements on one side as the final customers and the other side as the main suppliers.

6.9 Summary

This chapter proposed a distributed coordination strategy for reconfigurable process control. The key to the approach is a modular, bottom-up type problem solving mechanism that solves the overall control problem by interactions between (distributed) unit elements. The decoupling between unit subproblems in the solution technique enables the introduction of new unit elements. The unit elements are also able to respond to local disturbances dynamically and adjust their settings (as shown via demand changes in Examples 6.11 to 6.13). We also note that – in line with typical supply chain behaviour – each unit element, acting as the customer of its feedstock demands, attempts to coordinate these demands among potential supplier elements. A propagation of this demand distributed across the process scheme ensures the process responds to changes in the demands in a dynamic and incremental manner.