ELSEVIER

Computers and Chemical Engineering 26 (2002) 617–630

Computers & Chemical Engineering

www.elsevier.com/locate/compchemeng

An accelerated Branch-and-Bound algorithm for assignment problems of utility systems

Alexandros M. Strouvalis [a], Istvan Heckl [b], Ferenc Friedler [b], Antonis C. Kokossis [c,*]

[a] *Department of Process Integration, University of Manchester, Institute of Science and Technology, PO Box 88, Manchester M60 1QD, UK*
[b] *Department of Computer Science, University of Veszprém, Egyetem u.10, Veszprém H-8200, Hungary*
[c] *Department of Chemical and Process Engineering, School of Engineering in the Environment, University of Surrey, Guildford GU2 7XH, UK*

Abstract

A methodology is proposed for implementing logic and engineering knowledge within a Branch-and-Bound algorithm and with a purpose to accelerate convergence. The development addresses assignment problems of utility networks with an emphasis on the optimal allocation of units for maintenance problems. Proposed criteria are presented to automatically tailor the solution strategy and fully customise the optimisation solver. Model and problem properties are exploited to reduce the solution space, prioritise the branching of nodes, calculate lower bounds, and prune inferior parts of the binary tree. Comparisons with commercial MILP solvers demonstrate the significant merits of customising the solution search engine to the particular solution space. Extraction of knowledge and analysis of operations is conceptually supported by the graphical environment of the Hardware Composites.

Keywords: Turbine networks; Maintenance scheduling; Operational planning; MILP solvers; Hardware composites

1. Introduction

The impact of Mathematical Programming proves significant through a wide range of applications. Operations Research groups developed and contributed considerable part of the available optimisation tools. At their best, they epitomise general theoretical, computational and numerical knowledge in relevance to the different classes of problems. Past methods have proved unable to capture application features automatically. In the absence of specific knowledge, the use of general-purpose heuristics remains the only venue to accelerate the search and/or reduce the solution space. Considering that optimisation technology is a natural extension of simulation—now widely accepted in industry—it might be instructive to recall that simulation earned acceptance and credit only with the use of customised algorithms (c.f. the inside-out algorithm for distillation). In a similar vane, optimisation solvers should make an effort to systematically incorporate problem information. In the case of MILP's, the efficiency of a Branch-and-Bound algorithm is affected by the node branching criteria (Geoffrion & Marsten, 1972; Parker & Rardin, 1988; Floudas, 1995). In the absence of specific knowledge, the use of general search rules can not guarantee performance tantamount to the difficulty or the actual size of the problem.

A first major contribution to exploit problem logic involved the modelling stage. Floudas and Grossmann (1995) reviewed methods for reducing the combinatorial complexity of discrete problems using logic-based models. Raman and Grossmann (1992) reported improvements in solving MINLP's with a combined use of logic and heuristics. They illustrated their ideas with inference logic for the branching of the decision variables (1993), and studied the use of logical disjunctions as mixed-integer constraints (1994). Turkay and Grossmann (1996) extended the application of logic disjunctions to MINLP's using logic-based versions of OA and GBD algorithms. Hooker, Yan, Grossmann and Raman (1994) applied logic cuts to decrease the number of nodes in MILP process networks models. Friedler, Tarjan, Huang and Fan (1992) illustrated the potential for dramatic reductions in the solution space. Friedler,

* Corresponding author. Tel.: +44-1483-300800; fax: +44-1483-686581.
E-mail address: a.kokossis@surrey.ac.uk (A.C. Kokossis).

0098-1354/02/$ - see front matter © 2002 Published by Elsevier Science Ltd.
PII: S0098-1354(01)00788-8

Varga and Fan (1995) later introduced a decision mapping approach that is particularly efficient for process synthesis applications. Vecchietti and Grossmann (1999) discussed the development of a solver (LOGMIP) for solving disjunctive, algebraic MINLP or hybrid non-linear optimisation problems. They later, (2000) presented a modelling language to set up the disjunctions and logical constraints. Lee and Grossmann (2000) proposed a non-linear relaxation of the Generalised Disjunctive Programming and developed a special Branch-and-Bound algorithm that makes use of information of the relaxed problem to decide on the branching of terms of disjunctions.

The present work explains the integration of knowledge at a more advanced level: the solution search engine. The problem structure is exploited to prioritise and co-ordinate the optimisation search. A Customised Branch-and-Bound Solver (B&B) is developed to incorporate conceptual information. The approach reports significant reductions in the computational effort and supplements the work of Strouvalis, Heckl, Friedler and Kokossis (2000). The methodology makes use of Hardware Composites, (Mavromatis & Kokossis, 1998a,b,c; Strouvalis, Mavromatis & Kokossis, 1998). The tool supports the customisation of the algorithm, provides insights of utility operations and promotes the understanding of results.

2. Assignment problems and optimisation challenges

The problem taken into account deals with the assignment of units to shut-down periods. The appropriate sequence of switching off turbines and boilers for preventive maintenance contributes to the reliability, availability and profitability of the entire system. Each period relates to either a single or multiple sets of constant demands in power and process heat. Optimal scheduling assumes meeting demands and maintenance needs while minimising the operational cost. Switching off units imposes penalties to the economic performance of the network. As less efficient units are loaded or power is purchased from grid to compensate for the ones maintained, optimisation has to consider demand variations over time, differences in the efficiencies of units and feasibility aspects.

Formulations yield MILP problems with a considerable number of variables. Constraints are linear and binary variables are assigned for the ON/OFF status of units. Planning and Scheduling Problems of large-scale networks are computationally expensive or even intractable. This is due to the exponential growth of combinatorial load with the number of periods and units. Conventional approaches suggest the formulation of efficient models solved by standard commercial optimisation platforms. Even for moderate networks, however, the problem assumes prohibitive dimensions. Furthermore, due to minor differences in nearby solutions, the Branch-and-Bound trees are generally flat and difficult to fathom. General-purpose solvers are unable to capitalise on available information out of the general-purpose tuning, bounding and pruning framework they employ. Therefore, there is scope for efficient optimisation tools customised to the particular application.

3. Development of the Branch-and-Bound algorithm

The problem domain consists of three major components: utility network, demands, and maintenance requirements. Efficiencies and layout of units, operational costs, and constraints set up the network component. The fluctuating pattern of demands is a parameter of the problem along with the maintenance needs of each turbine and boiler. The problem domain is researched to define and prioritise the solution space and thus customise the algorithm. Conceptual analysis, feasibility assessment of maintenance scenarios, and cost assessment of basic maintenance scenarios are employed to preprocess the space. Conceptual analysis is achieved by plotting demands on the Hardware Composites and understanding the role of units over time. A set of LP's are solved to find infeasible options and objective function values of modes corresponding to the shut-down of individual units in all possible periods.

A standard B&B algorithm (i.e. Dakin, 1965; Floudas, 1995) is shown in Fig. 1(a) and consists of six major steps: (i) initialisation, (ii) termination, (iii) selection of candidate subproblem, (iv) relaxation (bounding), (v) pruning and (vi) separation (branching). The proposed customisation spans over the four main stages (highlighted boxes in Fig. 1(b)): candidate problem selection, bounding, branching and pruning. Special additional functions identify (mark) and exclude (delete) inferior nodes that should not be enumerated.

The contributions are at two levels.

(i) The conceptual analysis of the problem and the assignment of priorities (period and unit).
(ii) The tuning and customisation of the solver to the particular application.

The reduction of the solution space is accomplished by screening out infeasible and redundant combinations of variables. This first level makes simple and straightforward use of the engineering information and is referred as the Preprocessing Stage. The second level constitutes a more refined analysis of the knowledge. The basic idea is to replace general B&B rules by a number of criteria and functions to co-ordinate the branching and enhance the pruning.

3.1. Solution space preprocessing

3.1.1. Conceptual tools

The Hardware Composites have been developed for representing the feasible and optimal space of steam turbine networks. Visualised solution spaces function as 'maps' where demands are traced and corresponding turbine modes identified. The tool is based on the well-established Willans line (Church, 1950; Kearton, 1958) which models the expansion of steam through cylinders. For a single turbine t (Fig. 2(a)), inlet steam flowrate MS_t is given by:

$$MS_t = m_t E + c_t \quad (1)$$

where E is the turbine power output, m_t the slope of the line associated with the expansion path efficiency and c_t the no-load constant. Feasible modes locate on the Willans line.

For a backpressure passout turbine (Fig. 2(b)) the second cylinder gives rise to an additional term accounting for the passout flowrate and the associated power generation. The inlet steam consumption is given by the equation:

$$MS_t = m_t E + \left(1 - \frac{m_t}{g_t}\right) S + c_t \quad (2)$$

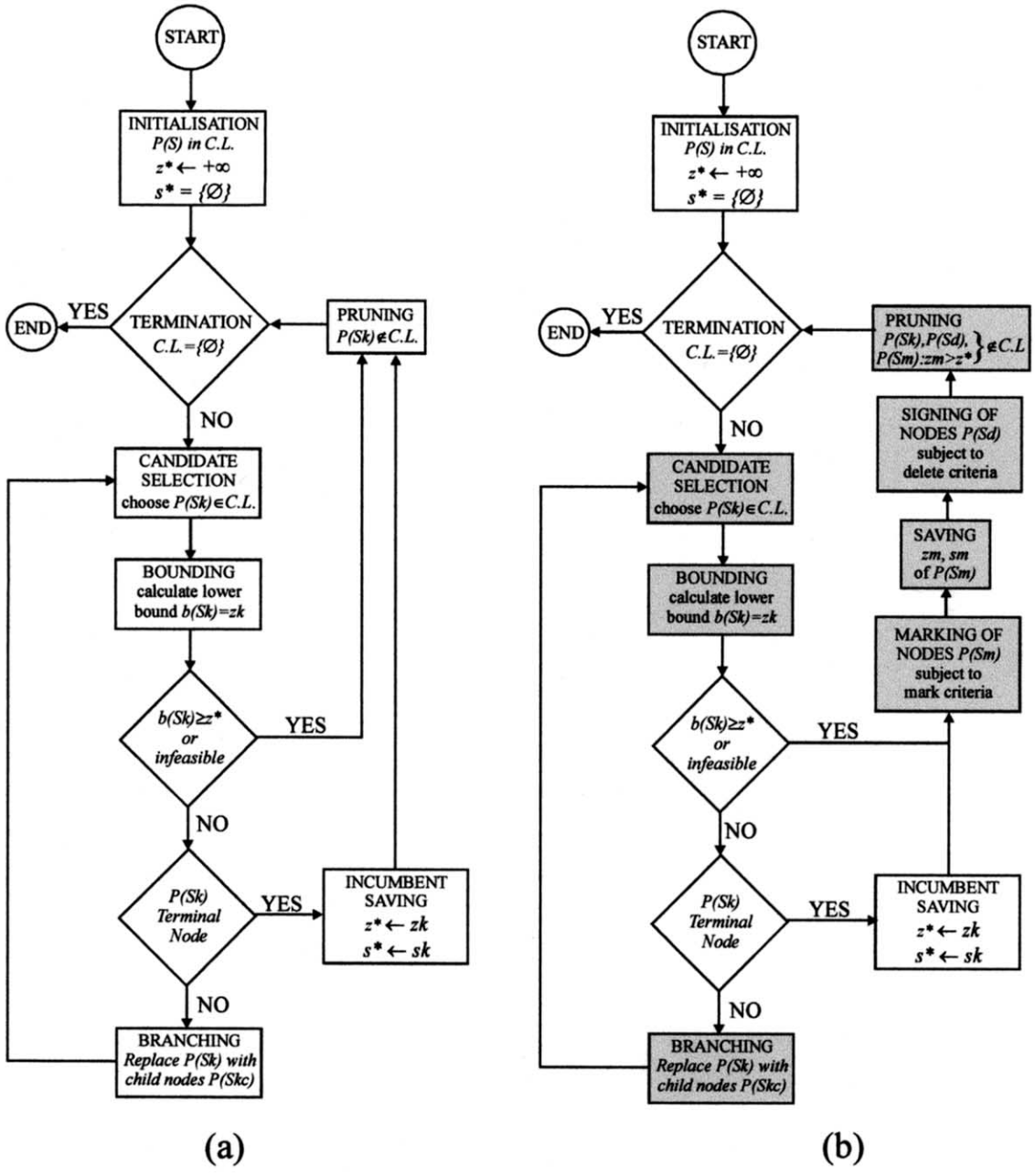

Fig. 1. (a) Standard and (b) customised Branch-and-Bound algorithm.

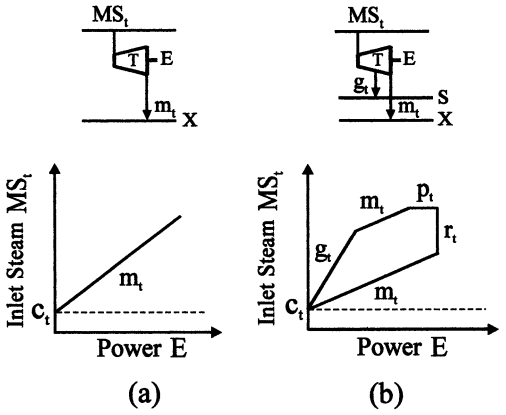

Fig. 2. Willans lines for (a) single and (b) backpressure passout turbines.

where S stands for the steam passout demand and g_t is the Willans line slope associated with the high-pressure expansion path. A turbine capability diagram may be restrained by lines $r_t = (1 - m_t/g_t)$ and $p_t = 1/(1/m_t - 1/g_t)$ corresponding to power and inlet steam capacities, respectively. Feasible operational modes are all inside the capability diagram. Operation of boilers and gas turbines is similarly defined by linear models presented in Appendix A.

The application of Construction Rules of Mavromatis and Kokossis (1998b) resolves efficiency conflicts and structures the optimal solution space of the entire turbine network. The rules impose full loading of the most efficient expansion paths prior to less efficient ones. Slopes m and g (incremental efficiencies) of the Willans lines define the line sequence on the Hardware Composites. Objective is minimisation of the inlet steam consumption, or equivalently the cost of fuel burnt in the boiler house.

Given the efficiencies and capacities of turbines (capability diagrams) and boilers, the Hardware Composites of a network featuring three steam turbines and two boilers are shown in Fig. 3(a). Any point on the Composites corresponds to unique set of demands and optimal mode. P, for example, designates the network operation for meeting power e_p and passout s_p demands with the least steam requirement MS_p. The path from the graph origin to P reveals the load of cylinders and depicts the role of units. The high-pressure cylinders of T_2 (at full load) and T_1 (supplementary) supply passout s_p. The T_1 low-pressure cylinder generates the remaining power output requirement of P. Dashed zones over the Hardware Composites denote operation of boilers and their steam generation capacities. Less fuel-consuming boilers first switch on to meet turbine inlet steam requirements. For greater flowrates (upper-right part of solution space) additional boilers accommodate the demands. At P, inlet steam MS_p is entirely provided by Boiler 1.

An alternative and equivalent representation is shown in Fig. 3(b); the Hardware Composites are drawn with the power and passout steam as co-ordinates. This version allows for directly spotting sets of demands on the solution space and is mainly adopted in the present work.

3.1.2. Assignment of priorities

The relative position of the demands and the units on the Hardware Composites determines favourable, unfavourable and impossible maintenance combinations. As units are switched off, the penalties imposed to the

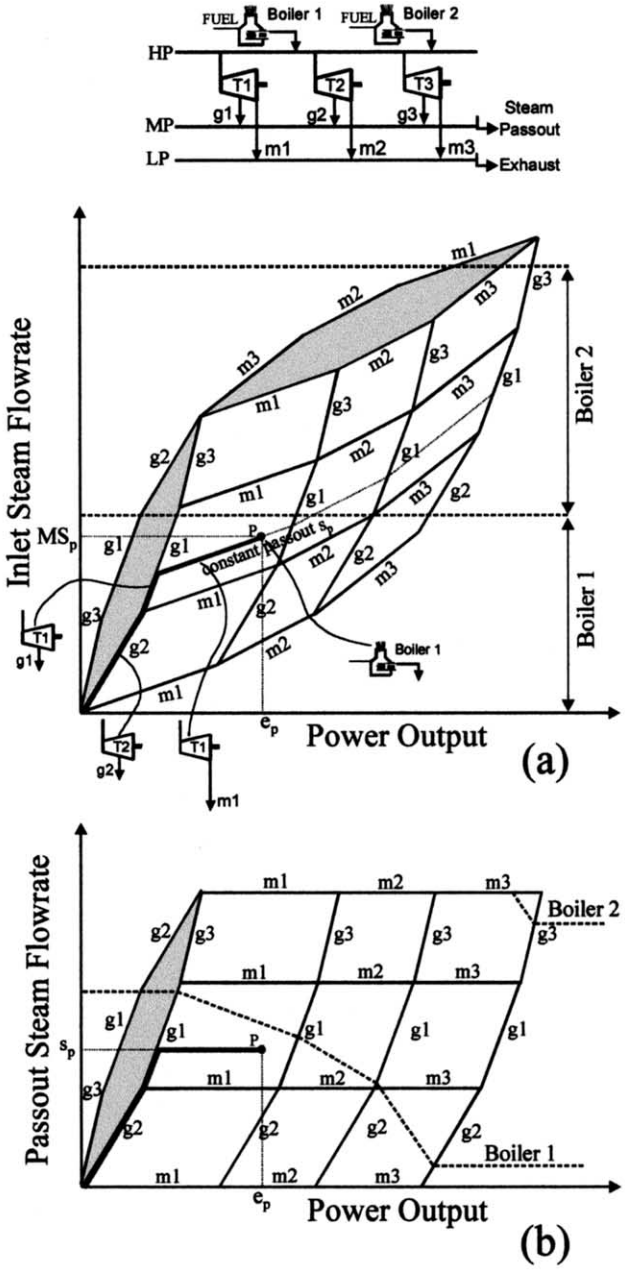

Fig. 3. Alternative Hardware Composites representations: (a) inlet steam vs. power output and (b) passout steam vs. power output.

objective function vary with the (i) unit efficiency, (ii) the efficiency of the units available to replace them and (iii) the demands of the particular period. Each period is affected to a different extent and priorities are strong functions of the layout of demands. High priorities for maintenance options relate to minor alterations in the operation of the utility network. Conceptual analysis (Strouvalis et al. (2000)), based on the Hardware Composites, provides knowledge and insights of the space and prioritises maintenance options.

In addition to conceptual preprocessing, priorities are established computationally as well for the sake of automation. Associated with each period $t \in T$ are lower bounds corresponding to the (lowest) operating cost (objective function) that employs all units $u \in U$. Let the set of lower bounds, $\text{Cost}^{\text{oper}} = \{c_{T_t}\}$, over all periods n of the time horizon be:

$$\text{Cost}^{\text{oper}} = \{c_{T_t} | t = 1, \ldots, n\} \tag{3}$$

For an idle u in period k a penalty $p(u, T_k)$ is assigned calculated by shutting down exactly u and evaluating the new objective value, C_{uT_k}. The penalty is given by:

$$p(u, T_k) = (C_{uT_k} - c_{T_k}) \tag{4}$$

Let OS_u^{pen} be the ordered set of penalties in ascending order:

$$\text{OS}_u^{\text{pen}} = \{(C_{uT_t} - c_{T_t}) | u \in U, t \in T\} \tag{5}$$

Let OS_u^{per} define the prioritised periods for u from the penalty sequence in OS_u^{pen}:

$$\text{OS}_u^{\text{per}} = \{T_u^i | i = 1, \ldots, n\} \tag{6}$$

where:

$$T_u^1 = T_t(C_{uT_t} - c_{T_t}) = \min_{t \in T} \text{OS}_u^{\text{pen}} \tag{7}$$

Period T_u^1 has the highest priority, followed in OS_u^{per} by periods of decreasing priority. The periods for each u in OS_u^{per} are alternatively represented as $T_u^{k(u)}$; T_u^1 corresponds to $k(u) = 1$, T_u^2 to $k(u) = 2$, etc.

Priorities over units are developed by calculating arithmetic averages P_u^{aver} over the first $N(u)$ entries of the OS_u^{pen}. $N(u)$ is defined by the following heuristic:

Let:

$$a = \frac{N_T}{A} \tag{8}$$

(non-integer values are rounded to previous integer, with $a \geq 1$))

$$b(u) = \text{number of periods in } \text{OS}_u^{\text{per}} \tag{9}$$

$$N(u) = \min\{a, b(u)\} \tag{10}$$

A is the weight parameter associated with the length of operating horizon.

The first $N(u)$ penalties of OS_u^{pen} are used to provide the arithmetic average P_u^{aver}:

$$P_u^{\text{aver}} = \frac{1}{N(u)} \sum_{n=1}^{N(u)} p(u, T_u^n) \tag{11}$$

Penalties P_u^{aver} for units u sorted in descending order provide ordered set OS_U:

$$\text{OS}_U = \{u_i : P_{u_i}^{\text{aver}} \geq P_{u_j}^{\text{aver}}, \forall j > i\} \tag{12}$$

Let us represent the integer solution space by a matrix with rows SMR_i that correspond to units and columns SMC_j that relate to periods. The elements of the matrix are binary variables allocating maintenance periods. The prioritisation of periods and units (OS_u^{per} and OS_U) defines the sequences SMR_i over the columns and rows, respectively. The Solver Matrix of the solution space is thus obtained by:

$$\text{SMR}_i = \text{OS}_{u_j}^{\text{per}} : \text{rank}(\text{OS}_U, u_j) = i \tag{13}$$

Rows follow the sequence of units in OS_U with periods ordered according to OS_u^{per} per row.

3.2. Tuning of the solver

3.2.1. Customised branching

Established priorities for periods and units encompass the knowledge of the particular utility network and variation of demands. Let us consider as Candidate List (CL) the running list of candidates for enumeration. The list includes nodes $v = (T_{u_1}^{k(u_1)}, T_{u_2}^{k(u_2)}, \ldots, T_{u_n}^{k(u_n)})$ that define specific subproblems (combination of binary variables assigning shut-down of units). Period $T_{u_1}^{k(u_1)}$ corresponds to the maintenance of the first unit of $\text{OS}_U[Y_{u_1 T_{u_1}^{k(u_1)}} = 0]$, $T_{u_2}^{k(u_2)}$ is the maintenance period of $u_2[Y_{u_2 T_{u_2}^{k(u_2)}} = 0]$, etc. If a node does not involve a decision for the maintenance of a unit, symbol (X) defines the corresponding relaxation. In the customised solver (X) represents the exclusion of the maintenance requirements (constraints) of u at the specific enumerated node. Parent node $v_o = (X, X, \ldots, X)$ represents the relaxed subproblem where no unit maintenance is assigned to any period. Node $(T_{u_1}^{k(u_1)}, T_{u_2}^{k(u_2)}, X, \ldots, X)$ represents the relaxed subproblem where the first two units in priority are allocated to maintenance periods with the rest assigned to no period. The bound value of node v is represented by $B(v)$.

Assume node $v = (T_{u_1}^{k(u_1)}, T_{u_2}^{k(u_2)}, \ldots, T_{u_m}^{k(u_m)}, X, \ldots, X)$ with the units in priority list OS_U up to m switched off in periods $T_{u_1}^{k(u_1)}, T_{u_2}^{k(u_2)}, \ldots, T_{u_m}^{k(u_m)}$, respectively.

- If $T_{u_1}^{k(u_1)} \neq T_{u_2}^{k(u_2)} \neq \ldots \neq T_{u_m}^{k(u_m)}$, node v is called independent. An independent node includes independent periods.
- If at least two out of periods $T_{u_1}^{k(u_1)}, T_{u_2}^{k(u_2)}, \ldots, T_{u_m}^{k(u_m)}$ are identical, node v is called dependent. The identical periods are called dependent periods.

- If node v is infeasible, it is classified as dependent with infinite bound.

Based on the properties of dependent and independent nodes, bounds are determined by penalty values $p(u_n, T_{u_n}^{k(u_n)})$ according to the following Lemma.

Lemma. *Let a parent node:*

$$v = (T_{u_1}^{k(u_1)}, T_{u_2}^{k(u_2)}, \ldots, T_{u_m}^{k(u_m)}, X, \ldots, X)$$

and a child node:

$$v' = (T_{u_1}^{k(u_1)}, T_{u_2}^{k(u_2)}, \ldots, T_{u_m}^{k(u_m)}, T_{u_n}^{k(u_n)}, X, \ldots, X)$$

(i) *if v' is independent then*

$$B(v') = B(v) + p(u_n, T_{u_n}^{k(u_n)}) \tag{14}$$

(ii) *if v' is dependent then*

$$B(v') \geq B(v) + p(u_n, T_{u_n}^{k(u_n)}) \tag{15}$$

If node v is terminal:

$$B(v) = B(v_0) + \sum_{i=1}^{U_M} p(u_i, T_{u_i}^{k(u_i)}) \quad (v \text{ independent}) \tag{16}$$

$$B(v) \geq B(v_0) + \sum_{i=1}^{U_M} p(u_i, T_{u_i}^{k(u_i)}) \quad (v \text{ dependent}) \tag{17}$$

where U_M is the total number of units subject to maintenance and $v_0 = (X, X, \ldots, X)$.

Branching of nodes is handled by two criteria: the Branching Variable Selection [BVSC] and Candidate Problem Selection Criterion [CPSC]. The scope of [BVSC] is to define the membership of CL and the ordering of the subproblems in it. The [CPSC] is responsible for executing the enumeration of the next node from the ones included in CL.

3.2.1.1. Branching variable selection criterion [BVSC]. The [BVSC] determines (i) the membership in the CL and (ii) the ordering of nodes in CL. This criterion uses sets OS_U and OS_u^{per}.

3.2.1.2. Candidate problem-node selection criterion [CPSC]. The Candidate Problem Selection Criterion [CPSC] has the task to select the next node for branching out of the CL of nodes. The criterion adopts a Depth-First strategy with backtracking; the Last-In-First-Out (LIFO) selects a child from parent node, whereby, the last added to CL is the first one to select. When a child node is pruned, backtracking of the parent node identifies other child nodes not yet enumerated. The (LIFO) criterion is chosen on the merits of requiring less candidate storage and calculation restarts than the Best-Bound alternative, as Parker and Rardin (1988) mention. The (LIFO) strategy implemented in a standard B&B, however, involves the solution of greater numbers of candidate subproblems. The customised pruning and bounding eliminate a significant portion of the explicitly visited nodes and overcome this drawback.

3.2.2. Customised pruning

Standard pruning disregards nodes with higher bounds (Fig. 1(a)). The proposed enhanced pruning consists of two additional criteria: (i) Mark and (ii) Delete Criterion (Fig. 1(b)). The first 'marks' subproblems in relation to dependent nodes, while the second eliminates from CL the qualified for pruning nodes.

Enhanced pruning exploits the properties of dependent and independent nodes. The methodology makes use of sets OS_u^{per} to select combinations and exclude redundant enumerations; nodes that follow independent nodes are pruned. For dependent nodes further branching is required.

3.2.2.1. Mark criterion [MC]. The Mark Criterion [MC] imposes the enumeration of nodes due to conflicts in priorities. [MC] applies on any node v retrieved from the CL. For a general dependent node $v = (T_{u_1}^{k(u_1)}, \ldots, T_{u_n}^{k(u_n)}, X, \ldots, X)$, where periods $T_{u_1}^{k(u_1)} \equiv T_{u_n}^{k(u_n)}$, [MC] marks the nodes $v_1 = (T_{u_1}^{k(u_1)+1}, X, \ldots, X)$, and $v_2 = (T_{u_1}^{k(u_1)}, \ldots, T_{u_n}^{k(u_n)+1}, X, \ldots, X)$ for enumeration. The number of marked nodes equals the number of dependent periods present in the dependent node. The same procedure is applied to units and periods of infeasible nodes.

3.2.2.2. Delete criterion [DC]. The Delete Criterion is applied on terminal or pruned nodes and it removes from CL unmarked nodes. [DC] eliminates nodes starting from the end of the CL up to the point a marked node is found. The elimination then halts and enumeration of marked nodes is enforced. If no marked node is present [DC] eliminates the entire list.

3.2.2.3. Theorems of customised pruning. Nodes v' qualified for enhanced pruning are defined by Theorem 1 and Theorem 2. The first defines pruned subproblems associated to independent nodes while the second to dependent ones.

Theorem 1. *Let a terminal independent node:*

$$v_1 = (T_{u_1}^{k(u_1)}, T_{u_2}^{k(u_2)}, \ldots, T_{u_n}^{k(u_n)})$$

or a pruned independent node:

$$v_2 = (T_{u_1}^{k(u_1)}, T_{u_2}^{k(u_2)}, \ldots, T_{u_n}^{k(u_n)}, X, \ldots, X)$$

For every node:

$$v_1' = (T_{u_1}^{k'(u_1)}, T_{u_2}^{k'(u_2)}, \ldots, T_{u_n}^{k'(u_n)})$$

or

$$v'_2 = (T^{k'(u_1)}_{u_1}, T^{k'(u_2)}_{u_2}, \ldots, T^{k'(u_n)}_{u_n}, X, \ldots, X)$$

with $k'(u_i) \geq k(u_i)$, $\forall i \in \{1, \ldots, n\}$ *it holds:*

$$B(v_1) \leq B(v'_1) \text{ and } B(v_2) \leq B(v'_2) \quad (18)$$

Theorem 2. *Let* $T^{k(u_k)}_{u_k} \equiv T^{k(u_j)}_{u_j}$ *on a terminal dependent node:*

$$v_1 = (T^{k(u_1)}_{u_1}, T^{k(u_2)}_{u_2}, \ldots, T^{k(u_k)}_{u_k}, \ldots, T^{k(u_j)}_{u_j}, \ldots, T^{k(u_n)}_{u_n})$$

or a pruned dependent node:

$$v_2 = (T^{k(u_1)}_{u_1}, T^{k(u_2)}_{u_2}, \ldots, T^{k(u_k)}_{u_k}, \ldots, T^{k(u_j)}_{u_j}, X, \ldots, X)$$

For every node:

$$v'_1 = (T^{k'(u_1)}_{u_1}, T^{k'(u_2)}_{u_2}, \ldots, T^{k(u_k)}_{u_k}, \ldots, T^{k(u_j)}_{u_j}, \ldots, T^{k'(u_n)}_{u_n})$$

or

$$v'_2 = (T^{k'(u_1)}_{u_1}, T^{k'(u_2)}_{u_2}, \ldots, T^{k(u_k)}_{u_k}, \ldots, T^{k(u_j)}_{u_j}, X, \ldots, X)$$

with $k'(u_i) \geq k(u_i)$, $\forall i \in \{1, \ldots, n\}$ *it holds:*

$$B(v_1) \leq B(v'_1) \text{ and } B(v_2) \leq B(v'_2) \quad (19)$$

Proofs of Theorem 1 and Theorem 2 are presented in Appendix B.

3.2.3. Customised bounding

The accelerated algorithm bounds nodes by capitalising on information of the Preprocessing Stage. Rather than calling the LP solver in every node for estimation of bounds (by fixing branched variables to integer values and relaxing the remaining ones), a more refined policy of solving LP's is adopted. Independent node combinations relate to decoupled maintenance (maintenance of units in non-identical periods). These nodes are appropriate for having their bounds defined by already available values. Nodes, on the contrary, associated to coupled combinations need separate bounding and LP solution. As such the intelligent and restrained use of the LP solver reduces the resources spent on bounding.

The approach includes:

1. Selective processing of nodes,
2. Selective processing of constraints.

3.2.3.1. Selective processing of nodes. Customised bounding applies only to independent nodes. In this case the solution of separate LP's to develop bounds becomes redundant. Indeed, bounding only requires updates of bounds through straightforward calculations. The approach makes use of the penalty values $p(u_i, T^{k(u_i)}_{u_i})$ to define the bound increase during branching. For dependent nodes, bounds follow conventional procedures. These nodes are bounded using the LP solver.

Overall, bounding makes a restricted use of the LP solver. The tightness of bounds is though relaxed, because, some of the maintenance constraints are disregarded; i.e. for the bound of $v_1 = (T_1, X)$, maintenance constraints of the second unit are not considered. As the remaining (relaxed) maintenance constraints are not taken under consideration less tight lower bounds are obtained.

3.2.3.2. Selective processing of constraints. Development of customised bounds is a combined application of a customised use of the LP solver and preprocessing information. In that perspective every subproblem must be examined disregarding all maintenance constraints of child nodes. This allows for decoupling decisions-constraints between parent and child nodes or equivalently between periods. In that manner calculation of bounds is permitted using values of the preprocessing.

The LP solver addresses subproblems where all binary variables are set to fixed values. Maintenance constraints coupling periods are introduced at each branching level. Nodes not already branched have the corresponding unit maintenance constraints (and hence binary variables) excluded from the bound calculation of their parent nodes. This allows the examination of decoupled scenarios using already available bounds. Only coupled operations, where shut-down of units affects more than one periods, are examined by separate LP's.

4. Implementation

The customised Branch-and-Bound solver is implemented in C++ and uses LINX (Fabian, 1992) as the LP solver. LINX is a simplex-based routine collection to facilitate special-purpose solvers and support the successive use of LP's in relevant sub-problems. The B&B solver features options for the implemented level of customisation. Options account for conventional B&B pruning or pruning based on [DC] and [MC]. As usual, priorities can alternatively be set by users.

5. Illustration

5.1. Problem description

The problem considers four steam turbines with the efficiencies of Table A1 (Appendix A). The time horizon consists of four time periods. Maintenance requires one shut-down period per unit. Sets of data in power and heat (A, B, C, D) are plotted on the Hardware Composites of the network (Fig. 4).

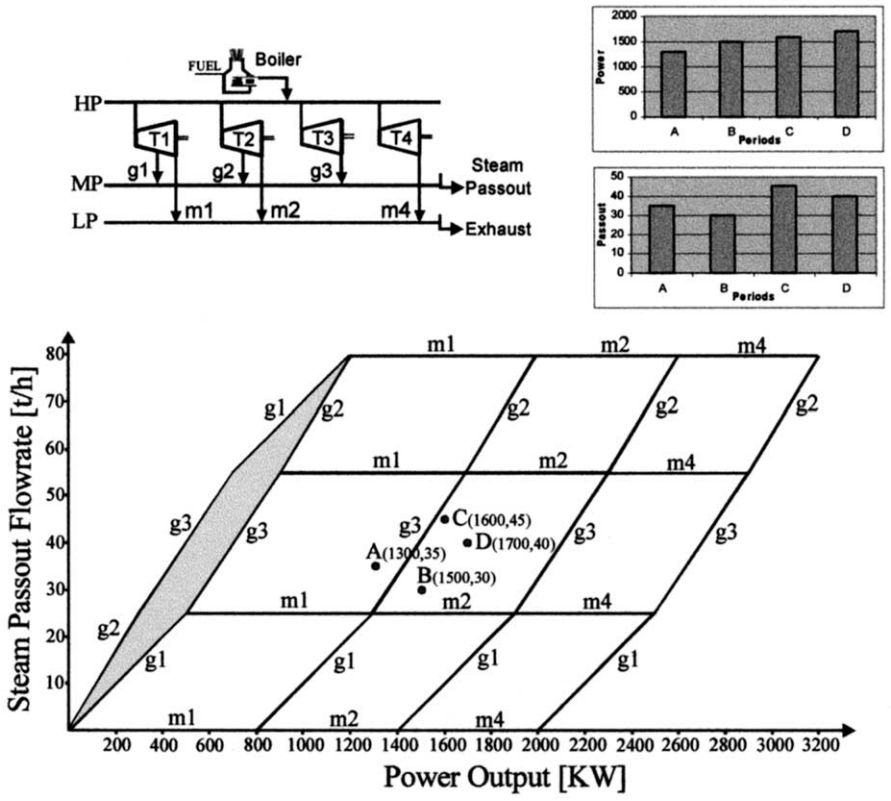

Fig. 4. The utility network and the hardware composites for illustration.

Step I: Preprocessing of Solution Space

(i) Preliminary Reduction of Solution Space
The sets of maintenance periods are:

$M(\mathrm{TU}_1) = \{A, B, C\}$

$M(\mathrm{TU}_2) = \{A, B, C, D\}$

$M(\mathrm{TU}_3) = \{A, B, C, D\}$

$M(\mathrm{TU}_4) = \{A, B, C, D\}$

Operation of turbine TU_1 is compulsory in D due to the high power demand. The large capacity of TU_1 safeguards the feasible operation and excludes period D from the set of possible maintenance periods $M(\mathrm{TU}_1)$ of TU_1.

(ii) Period prioritisation: Values c_{T_t} are calculated for each T_t and shown in Table 1:
Solution of $4 \times 4 = 16$ LP's yields the operational costs C_{uT_t} of Table 2.Penalties $p(u, T_t) = (C_{uT_t} - C_{T_t})$ are then defined (each considering one unit off) as shown in Table 3:Sorted in ascending order penalties $p(u, T_t)$ determine sets $\mathrm{OS}_u^{\mathrm{pen}}$:

$\mathrm{OS}_{\mathrm{TU}_1}^{\mathrm{pen}} = \{4433.3, 6022.2, 6388.9\}$

$\mathrm{OS}_{\mathrm{TU}_2}^{\mathrm{pen}} = \{0, 222.2, 888.9, 1333.3\}$

$\mathrm{OS}_{\mathrm{TU}_3}^{\mathrm{pen}} = \{55.6, 83.3, 166.6, 222.2\}$

$\mathrm{OS}_{\mathrm{TU}_4}^{\mathrm{pen}} = \{0, 0, 0, 0\}$

Next, associated periods T_t to penalties of $\mathrm{OS}_u^{\mathrm{pen}}$ provide the priority lists $\mathrm{OS}_u^{\mathrm{per}}$:

$\mathrm{OS}_{\mathrm{TU}_1}^{\mathrm{per}} = \{A, C, B\}$

$\mathrm{OS}_{\mathrm{TU}_2}^{\mathrm{per}} = \{A, C, B, D\}$

$\mathrm{OS}_{\mathrm{TU}_3}^{\mathrm{per}} = \{B, A, D, C\}$

$\mathrm{OS}_{\mathrm{TU}_4}^{\mathrm{per}} = \{D, C, B, A\}$

Table 1
Minimum operational cost c_{T_t} per T_t for illustration

	Period			
	A	*B*	*C*	*D*
c_{T_t} (\$/period)	21 666.7	21 111.1	27 777.8	26 666.7

Table 2
Operational costs (\$/period) of maintenance scenarios for illustration

	Period			
	A	*B*	*C*	*D*
TU_1	26 100	27 500	33 800	Infeasible
TU_2	21 666.7	22 000	28 000	28 000
TU_3	21 750	2166.7	28 000	26 833.3
TU_4	21 666.7	21 111.1	27 777.8	26 666.7

Table 3
Penalties ($/period) of maintenance scenarios for illustration

	Period			
	A	*B*	*C*	*D*
TU_1	4433.3	6388.9	6022.2	–
TU_2	0	888.9	222.2	1333.3
TU_3	83.3	55.6	222.2	166.6
TU_4	0	0	0	0

Especially for OS^{per}_{TU4} the priorities between zero-penalty periods are defined by insights of the problem (period *D* has higher priority over the rest since it is the least preferable for the maintenance of units TU_1, TU_2 and TU_3).

(iii) Unit Prioritisation: The number of periods to consider for calculations of P^{aver}_u uses the heuristic rule of Eqs. (8)–(11):

$$N(TU_1) = \min\left(\frac{4}{4}, 3\right) = 1$$

$$N(TU_2) = \min\left(\frac{4}{4}, 4\right) = 1$$

$$N(TU_3) = \min\left(\frac{4}{4}, 4\right) = 1$$

$$N(TU_4) = \min\left(\frac{4}{4}, 4\right) = 1$$

(1 period is chosen per unit).

Penalties are applied for the first element of OS^{pen}_u: TU_1 is assigned to $P^{aver}_{TU1} = 4433.3$, followed by TU_3 with $P^{aver}_{TU3} = 83.3$. TU_2 and TU_4 have zero first-period penalties. The priority list (first iteration) is $OS_U = \{TU_1,\ TU_3,\ TU_2–TU_4\}$, where $TU_2–TU_4$ represents units with identical P^{aver}_u. An additional iteration considers the second elements of OS^{pen}_{TU2} and OS^{pen}_{TU4}. TU_3 is now assigned to value $(888.9 + 0)/2$ and TU_4 to $(0 + 0)/2$. The unit priority list is finally defined as:

$$OS_U = \{TU_1, TU_3, TU_2, TU_4\} \quad (20)$$

From priority lists OS^{per}_u and OS_U the Solver Matrix is formulated:

$$\begin{bmatrix} TU_1 & A & C & B & \\ TU_3 & B & A & D & C \\ TU_2 & A & C & B & D \\ TU_4 & D & C & B & A \end{bmatrix}$$

Step II: Application of the B&B solver:

The application of the customised solver selects candidate problems and variables for branching, marks nodes for enumeration and deletes inferior nodes. Unless differently stated (dependent nodes), all lower bounds are calculated by adding to the lower bound of the parent node the penalty for shutting down a unit (associated to the branching level) in a specific period.

The B&B tree and the branching of nodes are presented in Fig. 5. Node numbers reflect the branching sequence. Information is also given for bound and penalty values.

The optimum maintenance vector is:

$$(TU_1, TU_3, TU_2, TU_4) = (A, B, C, D) \quad (21)$$

associating units and maintenance periods. While branching and pruning, the LP solver was called once, in dependent v_3, to provide a lower bound. All other enumerated nodes are independent and as such penalty values $p(u, T_t)$, available from the Preprocessing Stage, were used for bounding.

6. Example problem

The utility system of Fig. 6(a) includes nine units (steam turbines $T_1–T_5$, gas turbine GT and boilers $B_1–B_3$) with efficiencies and capacities presented in Table A2. The operating horizon consists of 24 equal-length periods. Each period relates to a pair of constant demands in power and process steam as shown in Table A3. Preventive maintenance imposes the shut-down of all units for one period. Optimal scheduling is expected to meet all demands and maintenance needs in the most economic way. Model properties are presented in Table 4.

The solution of the problem is addressed in two steps: I) preprocessing of the solution space and II) B&B customisation.

6.1. Step I

The solution space is analysed to reveal feasible and prioritised options. Preprocessing is applied in conceptual and computational terms. The location of demands on the solution space (Fig. 7) in relevance to hardware limits identifies infeasible scenarios, which are excluded from optimisation. LP's are solved to calculate penalties associated with the shut-down of units in feasible periods. Penalties in increasing sequence define priority lists OS^{per}_u. Penalties are also used for the unit prioritisation list OS_U. Ordered sets OS^{per}_u and OS_U formulate the Solver Matrix (Fig. 8) which encompasses the prioritised solution space. The first column includes all units subject to maintenance arranged according to OS_U (upper elements—GT, T_2, T_1, T_3,…—relate to higher priority units). Each element-unit of the first column associates to the corresponding period priority list OS^{per}_u defining the rows of the matrix.

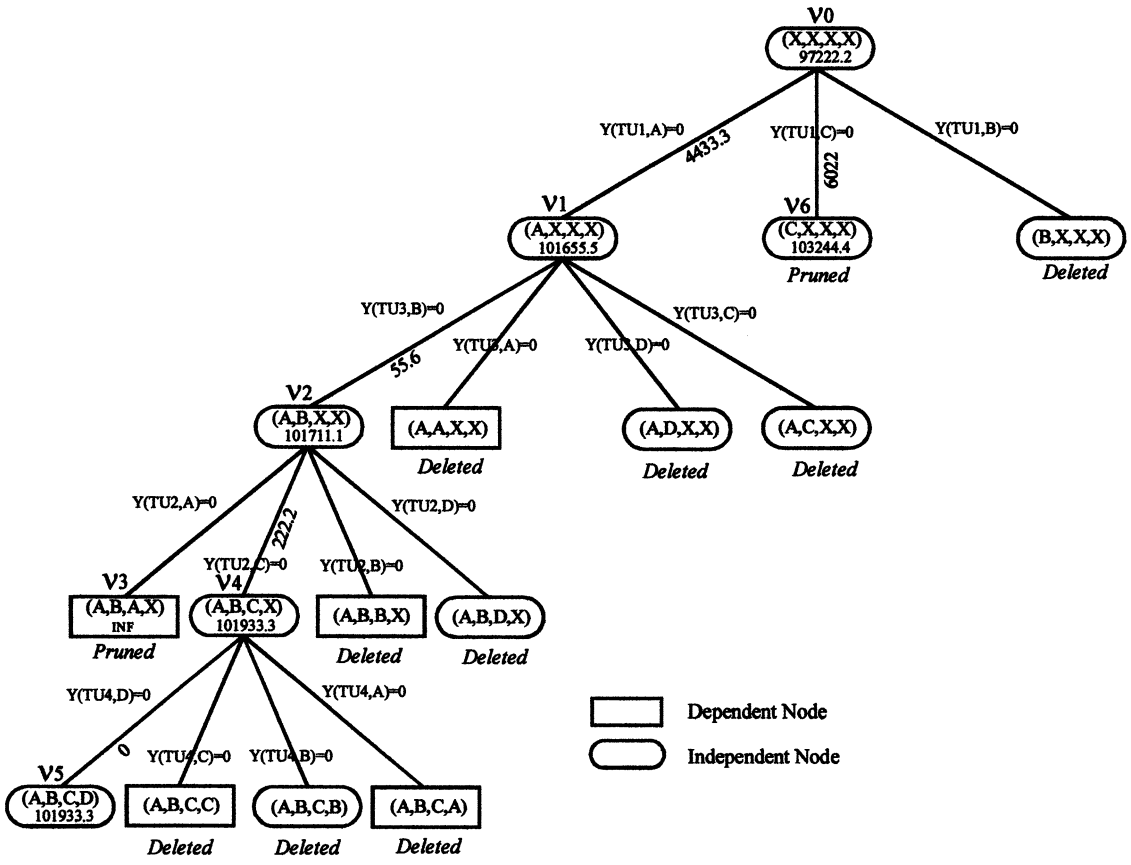

Fig. 5. The development of the B&B tree for illustration.

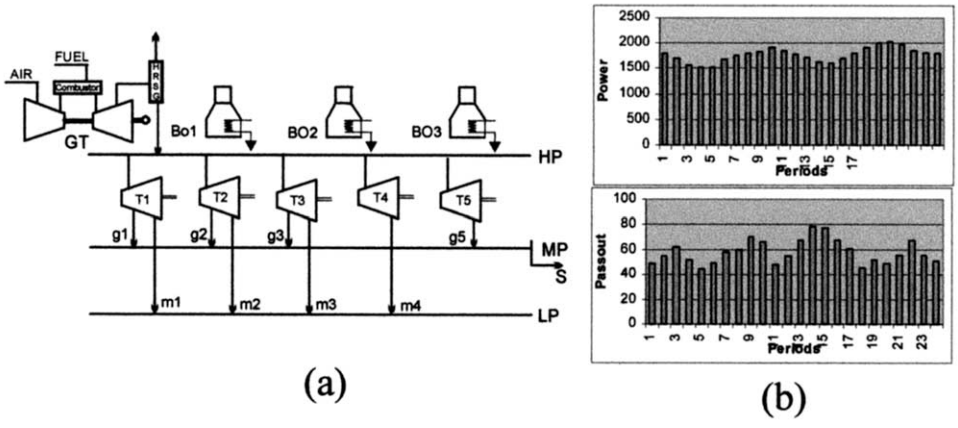

Fig. 6. (a) Utility network and (b) sets of demands in power and heat for example problem.

6.2. Step II

The customised B&B searches the solution space according to the structure of the Solver Matrix. The branching of the binary tree initiates from the first elements of the upper rows and proceeds to deeper options only if specific properties hold. The enhanced pruning effectively disregards inferior parts of the tree from enumeration. The result is acceleration of the B&B algorithm compared with the solution of the same

Table 4
Model size for example problem

Continuous variables	Binary variables	Constraints	Non-zero elements
457	216	754	2257

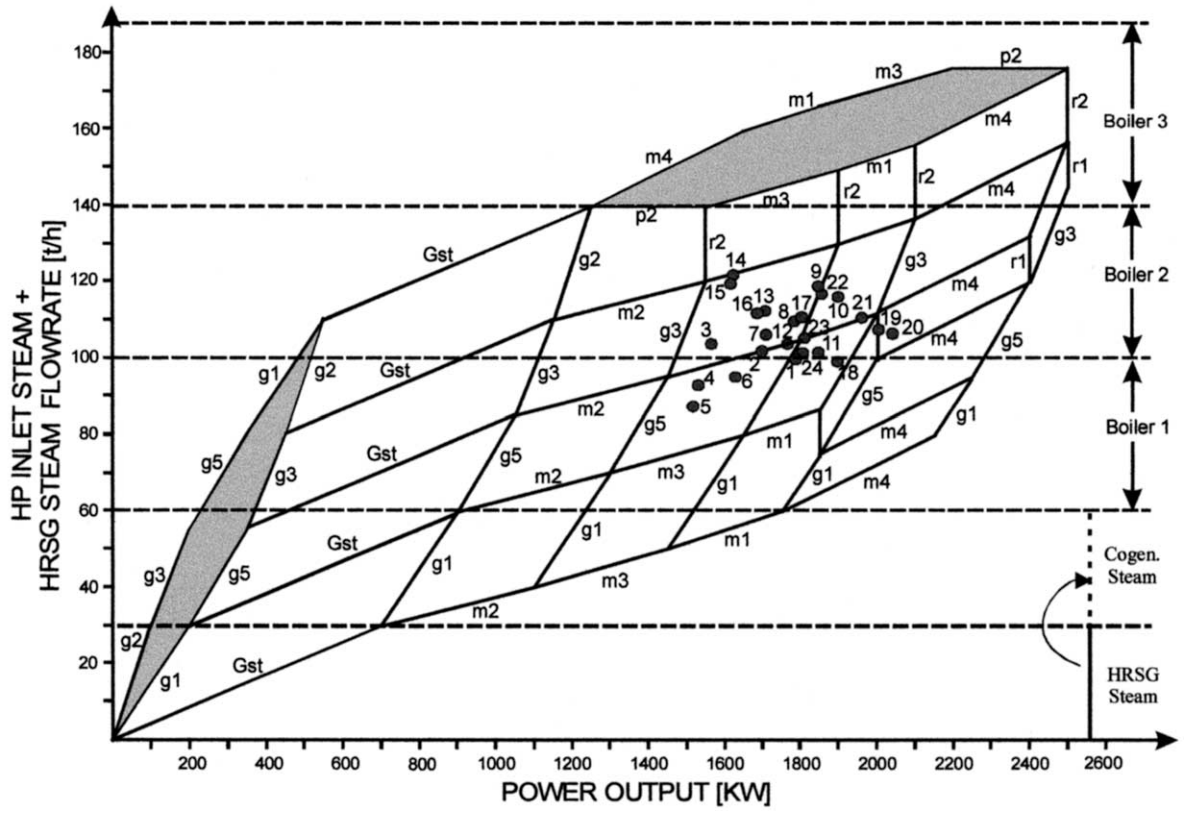

Fig. 7. Hardware composites depict the optimum solution space.

model by OSL implemented in GAMS (Brooke, Kendrick & Meeraus, 1992) as shown in Table 5.

OSL invested significant computational effort in searching the solution space. Even at the iteration limit of 150 000 optimality had not been reached due to the suboptimal flat profile the solver was trapped in. Alternatively, the customised B&B performed better in all aspects and identified the optimal schedule after solving (217 + 76) LP's during Preprocessing and Branching-and-Bounding, respectively. Optimal maintenance is performed according to vector:

$$(\mathrm{GT}, T_2, T_1, T_3, T_5, B_1, B_2, B_3, T_4) = (5, 15, 6, 4, 18, 4, 1, 19, 10) \quad (22)$$

7. Conclusions

This work reports on the acceleration observed by customising the solution search to the special structure of problems. While the basic notions of the Branch-and-Bound, a widely applied algorithm for solving MILP's, remain the same over the many contributions, computational experience has shown that general-purpose solvers can not capture special problem properties and are doomed to inferior performances. The design of inexpensive customised algorithms enhances computational efficiency and proves the superiority of such solvers over expensive commercial packages. It should be noted that the proposed methodology is generic

GT	5	3	15	14	16	13	6	2	7						
T2	15	14	4	3	5	16	13	6	2	7	8	17	12	9	1
T1	5	6	4	2	3	16	13	1	7	24	12	8	17		
T3	4	5	3	15	6	2	16	13	14	7	12	8	1		
T5	5	18	6	11	4	1	24	2	12	7	3	23	8		
B1	5	4	6	1	11	18	24	2	12	3	23	7	20		
B2	1	6	4	5	11	18	24	2	12	3	23	7	20		
B3	19	20	10	9	22	21	14	17	23	8	18	15			
T4	10	21	22	9	17	23	18	14	8	11	24	12			

Fig. 8. The solver matrix encompasses the B&B customisation.

Table 5
Computational results for example problem

	Customised B&B	OSL(GAMS)
Nodes	188	50,402
Iterations	150 000	(Interrupted)
Solved LP's	76	50 404
CPU(s)-333 MHz	3.2	1339
Objective-($/oper. horizon)	1 021 133.4	1 022 899.6
(Relaxed objective)	(1 004 690)	(1 005 439.7)

Preprocessing Stage, 217 LP's—26.2 CPU(s).

within the context of the application, namely for every problem within the specification in the outline and problem description. The proposed approach carries intelligence to fully automate the customisation of the optimisation solver. Solution search engines with built-in intelligence and search technology perform better orders of magnitude. The ability to apply key B&B functions (branching, pruning) tailored to particular solution spaces, speeds-up convergence and reduces computational costs.

Appendix A

Boiler and Gas turbine models:

The cost of fuel burnt in a boiler (CF_b), for producing steam flowrate MS_b is:

$$CF_b = B_f MS_b \tag{A.1}$$

Slope (B_f) represents the incremental boiler efficiency.

The gas turbine is modelled by two interrelated linear models that describe the relationship between the steam (MS_{HRSG}) in the Heat Recovery Steam Generator and the power output of the turbine (E_g):

$$MS_{HRSG} = G_{st} E_g \tag{A.2}$$

and a relation between the flowrate of the fuel burnt in the combustion chamber and the steam (MS_{HRSG}) raised in the HRSG:

$$CF_g = G_f MS_{HRSG} \tag{A.3}$$

G_{st} denotes the efficiency of raising HRSG steam and G_f the turbine efficiency.

Problem Data:

Table A1: Operational parameters of turbines and boilers for illustration.

m_{tu} [t/(hkW)]	g_{tu} [t/(hkW)]	B_f [$h/(t(time horizon)]	Turbine capacity: power (kW)
$m_{tu1} = 0.0125$	$g_{tu1} = 0.05$	$B_{fb} = 2000$	Turbine 3: 400
$m_{tu2} = 0.0166$	$g_{tu2} = 0.0833$		Turbine 4: 450
$m_{tu4} = 0.03$	$g_{tu3} = 0.075$		

Table A2: Operational parameters of turbines and boilers for example problem.

m_{tu} [t/(hkW)]	g_{tu} [t/(hkW)]	B_f [$h/(t(time horizon)]	Boiler capacity [t/h]
$m_{tu1} = 0.0333$	$g_{tu1} = 0.15$	$B_{fb1} = 523.3$	Boiler 1: 40
$m_{tu2} = 0.025$	$g_{tu2} = 0.3$	$B_{fb2} = 534.16$	Boiler 2: 40
$m_{tu3} = 0.02857$	$g_{tu3} = 0.25$	$B_{fb3} = 555$	Boiler 3: 50
$m_{tu5} = 0.05$	$g_{tu5} = 0.1667$	$G_f = 562.5$	

$G_{st} = 0.042857$ (t/hkW).

Table A3: Sets of demands in power and heat for example problem.

	Demands	
	Power (kW)	Process Steam [t/h]
Periods		
1	1790	49
2	1700	55
3	1572	62
4	1535	52
5	1520	45
6	1682	50
7	1765	58
8	1790	60
9	1840	70
10	1910	66
11	1850	48
12	1780	55
13	1710	68
14	1630	79
15	1610	77
16	1690	68
17	1800	61
18	1900	46
19	2003	52
20	2030	49
21	1960	56
22	1860	68
23	1820	56
24	1800	51

Appendix B

Proof of Theorem 1: v_1 is independent node $\Rightarrow$

$$B(v_1) = B(v_0) + p(u_1, T_{u_1}^{k(u_1)}) + p(u_2, T_{u_2}^{k(u_2)}) + \ldots + p(u_n, T_{u_n}^{k(u_n)}) \tag{B.1}$$

v_1' can be either independent node $\Rightarrow$

$$B(v_1') = B(v_0) + p(u_1, T_{u_1}^{k'(u_1)}) + p(u_2, T_{u_2}^{k'(u_2)}) + \ldots + p(u_n, T_{u_n}^{k'(u_n)}) \tag{B.2}$$

or dependent node ⇒

$$B(v_1') \geq B(v_0) + p(u_1, T_{u_1}^{k'(u_1)}) + p(u_2, T_{u_2}^{k'(u_2)}) + \ldots + p(u_n, T_{u_n}^{k'(u_n)}) \quad \text{(B.3)}$$

All periods T present in v_1' are of lower or equal priority than the corresponding ones of v_1. Due to priority sequence ($k'(u_i) \geq k(u_i)$), penalties:

$$p(u_1, T_{u_1}^{k'(u_1)}) \geq p(u_1, T_{u_1}^{k(u_1)}),\ p(u_2, T_{u_2}^{k'(u_2)}) \geq p(u_2, T_{u_2}^{k(u_2)}), \ldots, p(u_n, T_{u_n}^{k'(u_n)}) \geq p(u_n, T_{u_n}^{k(u_n)})$$

The lowest possible value for $B(v_1')$ (v_1' independent) is always higher than $B(v_1)$:

$$B(v_1) \leq B(v_1') \quad \text{(B.4)}$$

The same proof applies for the case of v_2. The lower bound $B(v_2)$ is given by Eq. (B.1), since the remaining non-branched levels (X) do not contribute penalty terms. It is similarly proven that:

$$B(v_2) \leq B(v_2') \quad \text{(B.5)}$$

Proof of Theorem 2

Node v_1 is dependent

$$(T_{u_k}^{k(u_k)} \equiv T_{u_j}^{k(u_j)}) \Rightarrow B(v_1) \geq B(v_0) + p(u_1, T_{u_1}^{k(u_1)}) + p(u_2, T_{u_2}^{k(u_2)}) + \ldots + p(u_k, T_{u_k}^{k(u_k)}) + \ldots + p(u_j, T_{u_j}^{k(u_j)}) + \ldots + p(u_n, T_{u_n}^{k(u_n)}) \quad \text{(B.6)}$$

$$\Rightarrow B(v_1) = B(v_0) + p(u_1, T_{u_1}^{k(u_j)}) + p(u_2, T_{u_2}^{k(u_2)}) + \ldots + p^{\text{ag}}((u_k, u_j), T_{u_k}^{k(u_k)}) + \ldots + p(u_n, T_{u_n}^{k(u_n)}) \quad \text{(B.7)}$$

where $p^{\text{ag}}((u_k, u_j), T_{u_k}^{k(u_k)}$ is the aggregated penalty when more than one units are shut down in the same period.

Node v_1' is dependent

$$(T_{u_k}^{k(u_k)} \equiv T_{u_j}^{k(u_j)}) \Rightarrow B(v_1') \geq B(v_0) + p(u_1, T_{u_1}^{k'(u_1)}) + p(u_2, T_{u_2}^{k'(u_2)}) + \ldots + p(u_k, T_{u_k}^{k(u_k)}) + \ldots + p(u_j, T_{u_j}^{k(u_j)}) + \ldots + p(u_n, T_{u_n}^{k'(u_n)}) \quad \text{(B.8)}$$

$$\Rightarrow B(v_1') = B(v_0) + p(u_1, T_{u_1}^{k'(u_1)}) + p(u_2, T_{u_2}^{k'(u_2)}) + \ldots + p^{\text{ag}}((u_k, u_j), T_{u_k}^{k(u_k)}) + \ldots + p(u_n, T_{u_n}^{k'(u_n)}) \quad \text{(B.9)}$$

All periods of node v_1' are of lower or equal priority than the corresponding ones of v_1 ($k'(u_i) \geq k(u_i)$) with the exception of dependent periods $T_{u_k}^{k(u_k)} \equiv T_{u_j}^{k(u_j)}$. As such, penalties $p(u_1, T_{u_1}^{k'(u_1)}) \geq p(u_1, T_{u_1}^{k(u_1)}), p(u_2, T_{u_2}^{k'(u_2)}) \geq p(u_2, T_{u_2}^{k(u_2)}), \ldots, p(u_n, T_{u_n}^{k'(u_n)}) \geq p(u_n, T_{u_n}^{k(u_n)})$, while ($p^{\text{ag}}((u_k, u_j), T_{u_k}^{k(u_k)}$) is the same on both nodes. Comparing the two bounds becomes evident that:

$$B(v_1) \leq B(v_1') \quad \text{(B.10)}$$

Similarly for a pruned dependent node v_2 where the non-branched levels the B&B tree (X) do not contribute penalty terms it is proven that:

$$B(v_2) \leq B(v_2') \quad \text{(B.11)}$$

References

Brooke, A., Kendrick, D., & Meeraus, A. (1992). *GAMS. A user guide*, *Release* 2.25. The Scientific Press.

Church, E. F. (1950). *Steam Turbines* (third ed.). New York: McGraw-Hill.

Dakin, R. J. (1965). A tree search algorithm for MILP problems. *Computational Journal*, *8*, 250.

Fabian, C.I. (1992). LINX: An interactive linear programming library.

Floudas, C. A. (1995). *Nonlinear and Mixed-Integer Optimisation*, *Fundamentals and Applications*. Oxford University Press.

Floudas, C.A. and Grossmann, I.E. (1995). Algorithmic approaches to process synthesis: logic and global optimisation, American Institute of Chemical Engineering Symposium Series 304, Fourth International Conference on Foundations of Computer-Aided Process Design, Vol. 91, 198.

Friedler, F., Tarjan, K., Huang, Y. W., & Fan, L. T. (1992). Graph-theoretic approach to process synthesis: Axioms and theorems. *Chemical Engineering Science*, *47*(8), 1973.

Friedler, F., Varga, J. B., & Fan, L. T. (1995). Decision-Mapping: a tool for consistent and complete decisions in process synthesis. *Chemical Engineering Science*, *50*(11), 1755.

Geoffrion, A. M., & Marsten, R. E. (1972). Integer programming algorithms: a framework and state-of-the-art survey. *Management Science*, *18*(9), 465.

Hooker, J. N., Yan, H., Grossmann, I. E., & Raman, R. (1994). Logic cuts for processing networks with fixed charges. *Computers Operations Research*, *21*(3), 265.

Kearton, W.J. Pitman, (1958). Steam turbine theory and practice, 7th ed., London: Pitman.

Lee, S., & Grossmann, I. E. (2000). New algorithms for nonlinear generalised disjunctive programming. *Computers and Chemical Engineering*, *24*, 2125.

Mavromatis, S. P., & Kokossis, A. C. (1998a). A logic based model for the analysis and optimisation of steam turbine networks. *Computers in Industry*, *36*(3), 165.

Mavromatis, S. P., & Kokossis, A. C. (1998b). Hardware Composites: A new conceptual tool for the analysis and optimisation of steam turbine networks in chemical process industries. Part I: principles and construction procedure. *Chemical Engineering Science*, *53*(7), 1405.

Mavromatis, S. P., & Kokossis, A. C. (1998c). Hardware Composites: a new conceptual tool for the analysis and optimisation of steam turbine networks in chemical process industries. Part II: application to operation and design. *Chemical Engineering Science*, *53*(7), 1435.

Parker, G., & Rardin, R. (1988). *Discrete Optimization*. Academic Press.

Raman, R., & Grossmann, I. E. (1992). Integration of logic and heuristic knowledge in MINLP optimisation for process synthesis. *Computers and Chemical Engineering*, *16*(3), 155.

Raman, R., & Grossmann, I. E. (1993). Symbolic Integration of logic in Mixed-Integer Linear Programming Techniques for process synthesis. *Computers and Chemical Engineering*, *17*(9), 909.

Raman, R., & Grossmann, I. E. (1994). Modelling and computational techniques for logic based integer programming. *Computers and Chemical Engineering*, *18*(7), 563.

Strouvalis, A. M., Mavromatis, S. P., & Kokossis, A. C. (1998). Conceptual optimisation of utility systems using hardware and comprehensive Hardware Composites. *Computers and Chemical Engineering*, *22*(Suppl.), 175.

Strouvalis, A. M., Heckl, I., Friedler, F., & Kokossis, A. C. (2000). Customised solvers for the Operational Planning and Scheduling of Utility Systems. *Computers and Chemical Engineering*, *24*, 487.

Turkay, M., & Grossmann, I. E. (1996). Logic-based MINLP algorithms for the optimal synthesis of process networks. *Computers and Chemical Engineering*, *20*(8), 959.

Vecchiety, A., & Grossmann, I. E. (2000). Modeling issues and implementation of language for disjunctive programming. *Computers and Chemical Engineering*, *24*, 2143.

Vecchietti, A., & Grossmann, I. E. (1999). LOGMIP: a disjunctive 0-1 non-linear optimizer for process system models. *Computers and Chemical Engineering*, *23*, 555.