International Journal of Production Research
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713696255

A tabu search heuristic for scheduling the production processes at an oil refinery

Jan A. Persson [a]; Maud Göthe-Lundgren [b]; Jan T. Lundgren [c]; Bernard Gendron [d]
[a] Department of Software Engineering and Computer Science Blekinge Institute of Technology, [b] Department of Mathematics Linköpings universitet, Sweden [c] Department of Science and Technology, Linköpings Universitet, Sweden [d] Centre de recherche sur les transports, Université de Montréal, Québec, Canada

Online Publication Date: 01 February 2004

To cite this Article Persson, Jan A., Göthe-Lundgren, Maud, Lundgren, Jan T. and Gendron, Bernard(2004)'A tabu search heuristic for scheduling the production processes at an oil refinery',International Journal of Production Research,42:3,445 — 471
To link to this Article: DOI: 10.1080/00207540310001613656
URL: http://dx.doi.org/10.1080/00207540310001613656

INT. J. PROD. RES., 2004, VOL. 42, NO. 3, 445–471

Taylor & Francis
Taylor & Francis Group

A tabu search heuristic for scheduling the production processes at an oil refinery

JAN A. PERSSON†*, MAUD GÖTHE-LUNDGREN‡,
JAN T. LUNDGREN§ and BERNARD GENDRON¶

In this paper we present a tabu search heuristic which can be used for scheduling the production at an oil refinery. The scheduling problem is to decide which production modes to use at the different processing units at each point in time. The problem is a type of lot-sizing problem where costs of changeovers, inventories and production are considered. In the suggested tabu search heuristic we explore the use of variable neighbourhood, dynamic penalty and different tabu lists. Computational results are presented for different versions of the heuristic and the results are compared to the best-known lower bound for a set of scheduling scenarios.

1. Introduction

We consider the scheduling of a set of processing units at an oil refinery. The production at an oil refinery is carried out using a set of processing units, where the output from one unit may be used as input for another unit. Each processing unit can normally be operated in different modes of operation, or run-modes, where each mode is defined by the set of input/output products and the rate at which the products are produced and consumed. The change from one run-mode to another causes disturbances to the production process, which reduces the quality of the products produced. In order to avoid frequent changeovers, we introduce changeover costs. Further, we also consider inventory costs, and costs of using a run-mode. Consequently, there is a trade-off between how long to use a run-mode and the obtained inventory levels. The scheduling problem concerns the question of which mode of operation to use at a particular point in time in each processing unit, while meeting the demand. The scheduling is strongly related to the planning at other levels in the company, and it affects many types of decisions in the company. The ability to efficiently construct high-quality schedules is therefore crucial for the refinery in order to be competitive.

Not very much is reported in the literature regarding the scheduling of operation modes at refineries using optimization models. One example is given in Ballintijn (1993), which uses a mixed-integer linear programming model to minimize the number of changeovers between operation modes. More recently the minimization

Revision received July 2003.

†Department of Software Engineering and Computer Science Blekinge Institute of Technology.
‡Department of Mathematics Linköpings universitet, Sweden.
§Department of Science and Technology, Linköpings Universitet, Sweden.
¶Centre de recherche sur les transports, Université de Montréal, Québec, Canada.
*To whom correspondence should be addressed. email: jan.persson@bth.se

International Journal of Production Research ISSN 0020–7543 print/ISSN 1366–588X online
http://www.tandf.co.uk/journals
DOI: 10.1080/00207540310001613656

of changeover and inventory costs is considered in Göthe-Lundgren *et al.* (2002). In that paper, the focus was on the modelling of the refinery scheduling problem and the potential use of optimization-based solution methods and not on the solution methods. In areas related to the scheduling of operation modes, such as long-term aggregate production planning (see Coxhead 1994 and Reklaitis 1996), and blending problems (Deqitt *et al.* 1989, Rigby *et al.* 1995, Amos *et al.* 1997), optimization models have been used. Other works related to the scheduling of operation modes concern unloading and blending of crude oil, feed management, and to some extent tank and pipe management (see Lee *et al.* 1996 and Shah 1996). A review of the problems and the models within the area of refinery planning and scheduling is given in Göthe-Lundgren *et al.* and Persson (2002).

The process scheduling problem we study can be formulated as a mixed integer linear programming problem, where the decisions concern which mode of operation to use in each processing unit during a set of time periods. The optimization model can be regarded as a capacitated lot-sizing problem with start-up costs, where more than one product is obtained for some modes of operation and where inventory capacities are explicitly considered. For an extensive review of lot-sizing problems and solution methods, see Pochet and Wolsey (1995), Constantino (1995) and Drexl and Kimms (1997).

We develop a tabu search heuristic for finding good schedules for the process scheduling problem. The origin of tabu search dates back to the early years of the 1970s, and the tabu search of the form we know today was introduced in the late 1980s. For a general description of tabu search, see Glover (1989, 1990). Tabu search has been applied with success to various optimization problems (see the reports presented in Volume 41 of the *Annals of Operations Research* (1993), the application survey in Glover and Laguna (1993), and Volume 106 of the *European Journal of Operational Research* (1998)). Applications on various scheduling problems exist, but we have found only one, Kimms (1996), which deals with a lot-sizing problem similar to the one we study. Compared to Kimms (1996), our process scheduling problem is more complex, since we have to deal with the simultaneous production of different products.

We consider real-life scheduling scenarios obtained from the Nynäashamn refinery in Sweden, owned by the Nynas oil company. According to our experience, guaranteed optimal schedules cannot be found in reasonable time for these scenarios using exact methods. This is due to the high complexity of the type of lot-sizing problem we are studying, a fact which is well known (see Constantino (1995), for a review of the time complexity of solving related lot-sizing problems). A tabu search heuristic is thus particularly indicated to address the problem.

The contribution of this paper is the introduction of a tabu search heuristic which can be used for finding good process schedules for refineries and possibly for other processing industries. We explore several neighbourhood structures, which can be used in the presented tabu search heuristic and possibly also in other types of heuristics (e.g. simulated annealing). We investigate what features are important for the performance of the tabu search heuristic on problem scenarios of practical size.

In section 2, we introduce the process scheduling problem, by describing the production facility and the planning situation at an oil refinery. The problem is mathematically formulated in section 3 and the tabu search heuristic is presented in section 4. In section 5, computational experiments are analysed for different

versions of the heuristic. Finally, conclusions and future research are outlined in section 6.

2. Problem description

In this section, we give a description of the production process and the associated scheduling problem at the Nynäshamn refinery. At the refinery, two major product groups are produced, bitumen products and naphthenic special oils. Additionally, some naphtha is produced, but this quantity is not important for the planning due to its low relative value.

The production process at the Nynäshamn refinery consists of three processing units; the central distillation unit (CDU), which transforms crude oil into bitumen, distillates, and naphtha, and two hydro-treatment processes (HF and HT) which transform distillates into naphthenic special oils. The production process is illustrated in figure 1.

In the CDU one bitumen product, four different distillates, and some naphtha are concurrently produced. At full production capacity, the rate of production corresponds to using approximately 4500 tonnes of crude oil per day. Different modes of operation can be used in the CDU, where the mode of operation, or run-mode, defines which type of crude oil that is used and at which rate the CDU is fed (which may be less than 4500 tonnes per day but not less than approximately 2000 tonnes). The run-mode also specifies which particular distillate products and bitumen product are concurrently produced. For example, a particular run-mode represents the continuous production at a rate per day corresponding to 45 tonnes of nahphta; 315, 279, 423, 545 tonnes of the different distillate products A, B, C, and D, respectively; and 2893 tonnes of a bitumen product E, when using 4500 tonnes of crude oil F. There are about 10 different run-modes normally used in the CDU during a limited time interval (of approximately six months).

Whenever the run-mode of the CDU is changed, it needs time to stabilize under the new operating conditions. The characteristics of the products obtained are

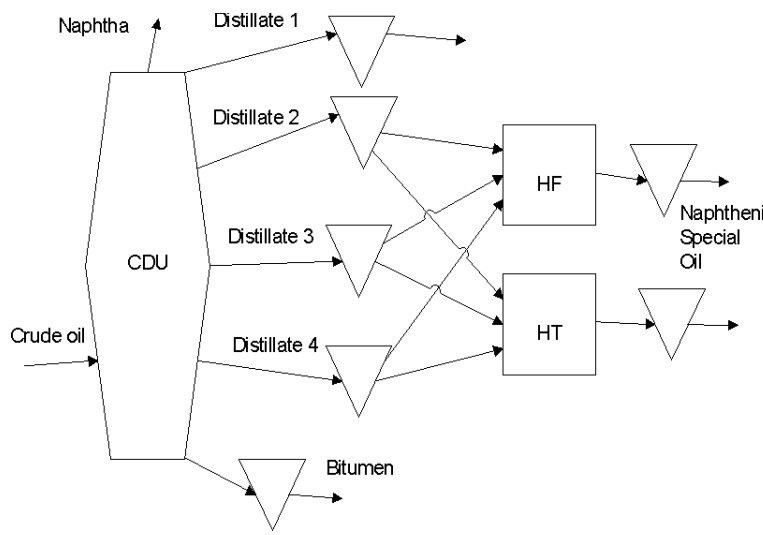

Figure 1. A schematic picture of the production process at the Nyäshamn refinery.

therefore uncertain and fluctuate for 1–2 h after a changeover. During this period, the distillates may have to be degraded to fuels of lower sales values. The degree of downgrading depends on several factors (e.g. the exact characteristic of the products in the tanks, the status of the production process, and the total quantity of the product produced by using the run-mode). In general, the production capacity of high-value distillates is reduced by frequent changeovers due to downgrading.

A run-mode used in the hydro-treatment processes represents the consumption of a particular distillate and the production of a particular naphthenic special oil, i.e. the run-mode defines the rate and the input and output of the process. The choice of run-modes in the two hydro-treatment units is restricted by the limited capacity of the hydrogen generating unit, and by the capacity of removing sulphur from the hydro-treatment units. We will refer to these limitations as resource constraints. Each processing unit can be operated using 5–7 different run-modes. After changing the run-mode used in a hydro-treatment process, the product produced during the first hour normally needs to be downgraded. The extent of downgrading is partly sequence dependent, i.e. the required time for downgrading depends on which run-modes the production is switched between. We will model the complex effects of changeovers in the processing units by simply including costs for changing run-modes.

Normally, there are two base bitumen products produced in the CDU. These are blended into about five different bitumen products, before being shipped with one of the two tankers owned by Nynas to a depot located in northern Europe. The naphthenic special oil products may also be blended, before being shipped with tankers to customers located world-wide. Currently, at the refinery, the planners construct a process schedule, which specifies which run-mode to use during each 24-h period in each processing unit, in order to meet the planned shipments during a planning horizon of one month.

The decision in the process scheduling problem is to decide which run-mode to use in a particular point in time, or equivalently, when to change between run-modes, in order to meet the planned deliveries from the refinery. There are safety stock levels and storage capacities, which limit the inventory levels which must be accounted for when planning the production. Further, there are additional resources limiting the production, which may prohibit the use of certain combinations of run-modes. However, most of these constraints are somewhat soft, meaning that solutions slightly violating them may be of interest. For example, a production schedule almost satisfying the planned deliveries may be of interest since some shipments can be re-scheduled without any greater impacts on the refinery's costs. Further, violation of resource constraints may be ignored if a lower quality or a lower production rate of the products is acceptable for a short period of time.

3. Problem formulation

In this section we present a mathematical formulation of the process scheduling problem at the Nynäshamn refinery. We formulate the model using general notation, which shows the possibility of using the model for other refinery production scheduling problems. The optimization model describing the process scheduling problem together with the notation is summarized in Appendix A.

We first introduce some notation. A time discretization of the planning period is made, and the resulting set of time periods is denoted by T. In the Nynas case, the length of one period is 24 hours, and the planning period is either one or two

months (31 days or 60 days). The lengths of the time periods are equal during the whole planning horizon, i.e. the time discretization is uniform. The decision variables y_{mt} are used to represent the process schedule and they are defined to assume the value 1 if run-mode m is used at time period t, else they assume the value 0.

The set of processing units is denoted by Q and at the Nynäashamn refinery, this set includes the CDU and the two (or three) hydro-treatment units. The set of possible run-modes at processing unit q is denoted by M_q, and the union of all possible run-modes at all processing units is denoted by M, i.e. $M = \cup_{q \in Q} M_q$. It is also assumed that a run-mode is dedicated for a particular processing unit, i.e. $M_q \cap M_{\bar{q}} = \varnothing, q \in Q$ and $\bar{q} \in Q : \bar{q} \neq q$. Let P denote the set of all types of products including input products (components) and end products. Further, we let R denote the set of resources, which in the Nynas case is associated with hydrogen generation and sulphur removal.

Since only one run-mode can be used in each processing unit at each time period, the equality

$$\sum_{m \in M_q} y_{mt} = 1, \tag{1}$$

must hold for all $q \in Q$ and $t \in T$. Without loss of generality we can enforce that one run-mode must be used in each processing unit in every time period, since one run-mode always represents zero production (the stop-mode).

In order to consider changeover costs we define a set of changeover variables, $s^{\text{seq}}_{m\hat{m}t}$, each of which assumes the value 1 if a changeover is performed from run-mode m to $\hat{m}$ between time periods $t-1$ and t, and 0 otherwise. Further, we use start-up variables, s_{mt}, each of which is equal to 1 if run-mode m is started at period t (i.e. it was not used at the previous time period), and 0 otherwise. The start-up variables are introduced since they are convenient to use in our tabu search heuristic. The constraints

$$s^{\text{seq}}_{m\hat{m}t} + 1 \geq y_{m,t-1} + y_{\hat{m}t}, \tag{2}$$

for all $q \in Q, m \in M_q, \hat{m} \in M_q : \hat{m} \neq m, t \in T$, and

$$y_{m,t-1} + s_{mt} \geq y_{mt}, \tag{3}$$

for all $m \in M, t \in T$, enforce these two types of variables to assume value 1 when a changeover is performed. We do not include any constraints enforcing the variables s_{mt} and $s^{\text{seq}}_{m\hat{m}t}$ to become zero when no changeover is performed, since this will be the case in an optimal solution due to non-negative changeover and start-up costs.

A process schedule (solution) can, besides being represented by variables y_{mt}, also be represented by the start-up variables s_{mt}. If a certain run-mode $m \in M_q$ of a processing unit q is utilized at time $t = t_1$ ($y_{mt_1} = 1$), then we know that run-mode m is used until a new start-up is performed of that processing unit at time $t = t_2 > t_1$, i.e. $y_{mt} = 1, \forall t \in \{t_1, \ldots, t_2 - 1\}$. Consequently, it is possible to translate a solution expressed in the variables y_{mt} into the variables s_{mt} and vice versa.

Given the values of the decision variables, y_{mt}, the quantities of the input (consumption) and output (production) follow at each time period. Let x_{pt} and z_{pt} denote the variables for production and consumption, respectively, of product p at period t. These variables are related to the run-mode variables by the production and consumption yield of product p when operating run-mode m. Let a_{pm} denote the

production yield and let b_{pm} denote the consumption yield. Then the variables x_{pt} and z_{pt} are related to y_{mt} according to

$$x_{pt} = \sum_{m \in M} a_{pm} y_{mt}, \qquad z_{pt} = \sum_{m \in M} b_{pm} y_{mt}, \tag{4}$$

for all $p \in P$ and $t \in T$. The production and consumption yields are by definition non-negative.

The inventory levels, I_{pt} , of product p at the end of period t, can be computed by using standard inventory balancing constraints including production, consumption and demand. The demand is given by the shipment plan and denoted d_{pt} for product p at period t. The parameter d_{pt} may be negative if it represents a delivery of the product to the refinery (crude oil or imported components). Hence, for a given schedule, the inventory levels for a product p and a time period t can be computed in the order $t = 1, \ldots, |T|$, by using the equations

$$I_{pt} = I_{p,t-1} + x_{pt} - z_{pt} - d_{pt}. \tag{5}$$

The initial inventory levels at the beginning of the planning horizon, I_{p0}, are assumed to be given.

For each product p and time period t, we specify the available storage capacity, denoted by $\bar{I}_{pt}$. This capacity is defined as the total capacity of the tanks normally used for the product. In addition to the upper bounds on the inventory levels, we have lower bounds denoted by $\underline{I}_{pt}$, representing requirements on safety stock levels for product p at period t. Requirements on the inventory level variables I_{pt} can then be formulated as

$$\underline{I}_{pt} \leq I_{pt} \leq \bar{I}_{pt}, \tag{6}$$

for all $p \in P$ and $t \in T$.

In order to ensure that the created schedules do not require too much of the available resources, we introduce resource constraints. For example, the run-modes of the hydro-treatment processing units have different needs for hydrogen, and the capacity of generating hydrogen is not large enough to support all combinations of run-modes. The same problem also occurs with respect to the capacity of taking care of the sulphur, which is an output from the hydro-treatment processes. Therefore, the inequalities

$$\sum_{m \in M} g_{mr} y_{mt} \leq \bar{E}_r, \tag{7}$$

for all $r \in R$ and $t \in T$, must be satisfied, where g_{mr} represents the use of resource r when using run-mode m, and $\bar{E}_r$ denotes the available resource of type r.

There is a number of different cost components, which all depend on the chosen process schedule. Since we are using a tabu search heuristic, we have great flexibility when choosing which cost components to consider. The costs considered here are inventory costs, changeover costs, start-up costs, and production costs, and can be expressed mathematically as

$$\sum_{t \in T} \sum_{p \in P} c^I_{pt} I_{pt} + \sum_{t \in T} \sum_{m \in M} \sum_{\hat{m} \in M : \hat{m} \neq m} c^{\text{seq}}_{m\hat{m}t} s^{\text{seq}}_{m\hat{m}t} + \sum_{t \in T} \sum_{m \in M} c^s_{mt} s_{mt} + \sum_{t \in T} \sum_{m \in M} c^y_{mt} y_{mt}. \tag{8}$$

The inventory cost for product p at period t is denoted by c^I_{pt}, and this cost is associated with the capital cost for the product and the cost of keeping the product

warm. The inventory cost of a product is assumed to be proportional to the inventory level.

The changeover cost is associated with the negative effects of changing modes (e.g. the possible necessity to downgrade the products after a changeover). Sequence dependent changeover costs are considered by specifying a cost parameter $c^{\text{seq}}_{m\hat{m}t}$ of changing from run-mode m to $\hat{m}$ between time periods $t-1$ and t. We also define c^{s}_{mt} as the start-up cost of run-mode m in period t. Note that a large portion of the changeover costs can be viewed as sequence independent, i.e. the cost of starting a run-mode does not depend on which run-mode it is switched from. Hence, start-up costs allow one to represent changeover costs when sequence dependency can be ignored.

The total production capacity might exceed the total demand during the planning horizon. Then it might be economical to produce products for adding to the inventory levels. This is considered in the model by associating a value of using a particular run-mode. This value of using a run-mode is based on the values of its input and output products, which in turn can be estimated by using information from different planning steps at the refinery. The overall objective is to minimize the total cost in our model, and therefore, the estimated value of using a run-mode is given as a negative cost to the model, represented by the coefficient c^{y}_{mt}. Positive values of c^{y}_{mt} are often referred to as set-up costs.

The complete optimization model, given by constraints (1)–(7) and by the cost function (8), is referred to as [SCH] and is summarized in Appendix A. For planning purpose, the constraints (6) and (7), and the non-negativity constraints of the variable I_{pt} may be regarded as soft and can accordingly be relaxed with a suitable cost of violation. The obtained problem is referred to as [R-SCH]. However, violation of any these constraints should only be allowed if no feasible solution can be found or if a competitive low-cost schedule can be found which only marginally violates these constraints. Hence, we allow the violation of these constraints at a rather high cost compared to the costs in the objective function (8). We are primarily interested in the best solution to [R-SCH].

In case that a schedule implies that a small deficit of a product will occur at a time period (i.e. a constraint of type (6) is violated), that deficit will persist during all the subsequent time periods and incur penalties, unless the deficit is eliminated by production. Since a deficit at the refinery can be dealt with, for example, by re-planning of the shipments from the refinery, we would like a deficit to incur a limited penalty, if the deficit is eliminated by production at a later point in time. Further, since violations of the maximal inventory levels can be dealt with, for example, by re-assigning tanks for storage, we would like the violations to incur a limited penalty. In order to model this situation, we modify the inventory levels whenever they are not within the specified limits in the previous time period. That is, we modify constraints (5) to

$$I_{pt} = I_{p,t-1} + x_{pt} - z_{pt} - d_{pt} + R^{f}[(\underline{I}_{p,t-1} - I_{p,t-1})^{+} - (I_{p,t-1} - \bar{I}_{p,t-1})^{+}], \qquad (9)$$

where R^{f} expresses the fraction of the deficit or of the excess to be eliminated. Following discussions with planners at Nynas, it was decided that a reasonable value is $R^{f} = 0.4$.

The problem [SCH] can be interpreted as a generalization of a lot-sizing problem. Since we consider *multiple products*, *production capacities* and *start-up costs*, our

problem [SCH] is similar to the multi-item capacitated lot-sizing problem with start-up costs, studied by Karmarkar and Schrage (1985). The proposed model, however, has a number of additional characteristics. It has an *all-or-nothing* characteristic, meaning that production (or consumption) can only be at zero-level or at the capacity level (a_{pm} or b_{pm}) given by the used run-mode. This is expressed by the equality requirements in constraints (4). Further, the products considered can be both produced and consumed, which makes it a *multi-level* (multi-stage) problem. We need to consider not only the product structure but also the structure of the production facility, i.e. which different run-modes can be utilized simultaneously. Additionally, in the model [SCH] the production of one product may 'force' the production of one (or several) other products. Upper limits on the inventory levels are often not included in lot-sizing models and the feature of forcing the production of multiple products appears not to have been previously accounted for in lot-sizing models.

It is known that the capacitated lot-sizing problem with start-up costs is hard to solve using exact methods, even for relatively small instances. For example, in Constantino (1996), problems with up to 180 binary set-up variables (corresponding to y_{mt}) are solved using valid inequalities and a branch-and-bound procedure. Heuristic methods can also be found in the literature (see the review in Drexl and Kimms (1997). For example, in Kimms (1996) a tabu search and a randomized regret search heuristic are used to find schedules to a capacitated lot-sizing problem with start-up costs, where production and consumption of only a single product was allowed at any time period. The tabu search is used to direct the alterations of a graph, which is used for creating schedules (solutions) to the problem. Also, the multiple machine capacitated lot-sizing problem is solved using randomized regrets in Kimms and Drexk (1998) and genetic algorithms in Kimms (1999).

4. A tabu search heuristic

In this section we first present a short general description of tabu search. Then we provide a detailed description of our tabu search heuristic for the process scheduling problem.

Tabu search can essentially be regarded as a refined neighbourhood (descent) search heuristic, in which the definition of the neighbourhood may vary and in which local deterioration of the objective function value is allowed. It iteratively moves from a current solution ξ^k by selecting the best solution ξ^{k+1} among those belonging to a neighbourhood of the current solution. Moves implying a move back to previous solutions are declared tabu for some iterations and are stored in a tabu list. The search is guided by the choice of the length and the type of the tabu list, by the definition of the neighbourhood, and by the function used for evaluating solutions. The elements used for guiding the search are often modified during the search procedure in order to create periods of intensification and diversification. Intensification forces the search to focus on solutions similar to the solution currently investigated, whereas diversification implies that the heuristic is forced away from the current solution.

The basic steps of a general tabu search heuristic are given below, where ξ denotes the solution (feasible or infeasible) at iteration k, $\xi_{\text{f–best}}$ denotes the best feasible solution found so far, and K denotes the maximum number of iterations. The state at iteration k is denoted by S^k and it defines, for instance, the solutions or moves that are currently tabu and the level of violation of some relaxed constraints during recent iterations. For a solution ξ^k, the set $N_{\text{cand}}(\xi^k, S^k)$ is the candidate set of

non-tabu neighbouring solutions. An aspiration criterion may be utilized, such that a 'tabu solution' can be selected as the next iterate if, for example, it is better than $\xi_{\text{f–best}}$. A modified objective function, $c^m(\xi^k, S^k)$, is used, which is based on the original objective function, $c(\xi^k)$, and additional penalty terms included for guiding the heuristic towards feasible solutions.

1. *Initialization*
   - Determine ξ^0 and S^0. Let $\xi_{\text{f–best}}/\xi^0$ if ξ^0 is feasible, and $k \leftarrow 0$.
2. *Selection of* ξ^{k+1}
   - Determine $N_{\text{cand}}(\xi^k, S^k)$.
   - Choose $\xi^{k+1} \in N_{\text{cand}}(\xi^k, S^k)$, that minimizes $c^m(\xi^{k+1}, S^k)$.
3. *Update and termination*
   - Update S^{k+1} and if $c(\xi^{k+1}) < c(\xi_{\text{f–best}})$ and ξ^{k+1} feasible, let $\xi_{\text{f–best}}/\xi^{k+1}$.
   - Terminate if any stopping criterion is satisfied (e.g. if $k \geq K$) else let $k \leftarrow k+1$ and goto step 2.

Let ξ^k denote a solution to our scheduling problem. In our tabu search we make sure that the solution ξ^k satisfies formulation [R-SCH]. We keep track of the best solution to [R-SCH], denoted ξ_{best}, with corresponding value of the modified objective function given by $c^m(\xi_{\text{best}}, S)$, where the state S includes a specification of how much to penalize infeasibility. Additionally, we keep track of the best 'feasible solution' $\xi_{\text{f–best}}$ which is feasible with respect to [SCH].

It is possible to use any of the variables s_{mt}, y_{mt} or $s^{\text{seq}}_{m\hat{m}t}$ to uniquely define a schedule. In our tabu search heuristic we use the start-up variables s_{mt} for defining a schedule. Then, given the values of s_{mt}, the corresponding values of the variables y_{mt} can easily be computed. Thereby, all other variables can also be computed by using the relations given in section 3.

4.1. *Neighbourhoods*

In order to define a neighbourhood of a given solution ξ^k to the relaxed scheduling problem [R-SCH], we use several sub-neighbourhoods. The complete neighbourhood $N(\xi^k)$ is the union of these sub-neighbourhoods. Then, we typically consider only a subset of this complete neighbourhood at the kth iteration, i.e. $N_{\text{cand}}(\xi^k, S^k) \subseteq N(\xi^k)$, where $N_{\text{cand}}(\xi^k, S^k)$ is the set of solutions we investigate at iteration k. The sub-neighbourhoods introduced in this section are illustrated by examples in Appendix B.

The sub-neighbourhood denoted by $N_{e/l}(\xi^k)$ (earlier/later), consists of all solutions where a start-up of a run-mode is performed either one time period earlier or one period later than the start-up times in ξ^k. Only one change is allowed, i.e. any solution where two start-ups are altered simultaneously is not included in $N_{e/l}(\xi^k)$.

The sub-neighbourhood, $N_{\text{move}}(\xi^k)$, is defined by solutions where one of the current start-ups is changed to a start-up of another run-mode but in the same time period and in the same processing unit. We do not allow a start-up of a run-mode that is already in use or the start-up of a run-mode that is the next one to be started.

The sub-neighbourhood $N_{\text{switch}}(\xi^k)$ allows the temporal order (in time) of two consecutive run-modes to change start-up positions and lengths. The changes can be characterized as two simultaneous changes associated with the sub-neighbourhood $N_{\text{move}}(\xi^k)$.

The three sub-neighbourhoods given so far do not allow for any changes in the number of start-ups. The sub-neighbourhood $N_{\text{add}}(\xi^k)$ is obtained by adding a single start-up at all possible positions without violating constraints (1) and without making double start-ups of the same run-mode. The sub-neighbourhood $N_{\text{remove}}(\xi^k)$ consists of solutions where a single start-up is removed from the solution ξ^k.

Next the issue of how to use the sub-neighbourhoods is discussed. In order to reduce the total number of solutions to be evaluated in $N_{\text{cand}}(\xi^k, S^k)$ and to direct the search, the different sub-neighbourhoods are included at some given periodicities, $p_{e/l}$, p_{move}, p_{switch}, p_{add}, and p_{remove}, respectively. The sub-neighbourhood $N_{e/l}(\xi^k)$ is used in almost all iterations and the other sub-neighbourhoods not as often. (The periodicities used are presented later in table 2 of section 5.3.) However, the sub-neighbourhood $N_{e/l}(\xi^k)$ is excluded when another sub-neighbourhood is forced to be utilized. For example, the periodicity $p_{\text{move}} = 43$ implies that at every 43rd iteration the sub-neighbourhood $N_{\text{move}}(\xi^k)$ is used. Then, the sub-neighbourhood $N_{e/l}(\xi^k)$ is excluded from the neighbourhood ($N_{\text{cand}}(\xi^k, S^k)$) investigated at iteration k. The exclusion of $N_{e/l}(\xi^k)$ is referred to as the *forced* feature. The sub-neighbourhood $N_{\text{move}}(\xi^k)$ is also used at every $\lfloor p_{\text{move}}/2 \rfloor$ iteration, but in this case, the sub-neighbourhood $N_{e/l}(\xi^k)$ is not excluded from the neighbourhood explored. The objective of this feature is to favor diversification. During the initial phase of the heuristic, the neighbourhood $N_{\text{add}}(\xi^k)$ is used more often than implied by the periodicity p_{add}. This makes sense if the starting solution ξ^0 contains very few start-ups. This initialization is described in section 5.2.

The neighbourhood used at an iteration is reduced in a stochastic manner in order to speed-up the heuristic, that is, sampling is utilized for reducing the number of evaluations. This is referred to as using a *reduced* evaluation of the neighbourhood. On average, only 50% of the solutions in the neighbourhood $N_{\text{move}}(\xi^k)$ and 25% of the solutions in the neighbourhood $N_{\text{add}}(\xi^k)$ are randomly selected. Since the number of solutions in these sub-neighbourhoods is relatively many compared to the other sub-neighbourhoods, a substantial speed-up is obtained.

A speed-up is also achieved by partially locking the solution corresponding to the run-modes of one processing unit for some iterations. Then, only such solutions that correspond to changes of the schedule for the other processing units are included in the candidate set of solutions $N_{\text{cand}}(\xi^k, S^k)$. Since parts of the solution space are locked, this implies that intensification increases, and only solutions with some similarities are investigated. This feature is referred to as *locked*. If any processing unit should be locked and which one to lock is decided upon randomly. The probability of exclusively locking one processing unit was set to $1/(|Q| + 1)$.

4.2. *The modified objective function*

Given a solution ξ^k, which is feasible in [R-SCH], a modified objective function value $c^m(\xi^k, S^k)$ can be computed, where S^k denotes the state of the tabu search at iteration k. The modified objective function value $c^m(\xi^k, S^k)$ is computed according to

$$c^m(\xi^k, S^k) = c(\xi^k) + \text{TP}^I(\xi^k, S^k) + \text{TP}^E(\xi^k, S^k), \tag{10}$$

where $c(\xi^k)$ is the original objective function value defined by (8). $\text{TP}^I(\xi^k, S^k)$ and $\text{TP}^E(\xi^k, S^k)$ are penalty costs with respect to inventory level constraints (6) and to resource constraints (7), respectively.

The penalty for solution ξ^k with respect to the inventory level constraints (6) can be computed as

$$\mathrm{TP}^I(\xi^k, S^k) = \sum_{p\in P}\sum_{t\in T}\left[P^{\bar{I}}(S^k)(I_{pt} - \bar{I}_{pt})^+ + P^{\underline{I}}(S^k)(\underline{I}_{pt} - I_{pt})^+\right], \tag{11}$$

where $P^{\bar{I}}(S^k)$ and $P^{\underline{I}}(S^k)$ are parameters expressing how much to penalize the violation of the lower and upper inventory level constraints, respectively. The non-negativity constraints on I_{pt} can be ignored since solutions which do not satisfy these constraints are penalized in (11). In our implementation $P^{\bar{I}}(S^k)$ and $P^{\underline{I}}(S^k)$ are equal and represented by the parameter $P(S^k)$, i.e. $P(S^k) = P^{\bar{I}}(S^k) = P^{\underline{I}}(S^k)$. This parameter can be allowed to vary depending on the state of the tabu search. We adjusted the value of the parameter $P(S^k)$ depending on the smoothed average value of $\mathrm{TP}^I(\xi^k, S^k)$ in previous iterations. This feature, called *dynamic penalty*, is achieved by using the formula $P(S^k) = \alpha_1 \cdot P(S^{k-1}) + (1-\alpha_1)\cdot\alpha_2 \cdot \mathrm{TP}^I(\xi^{k-1})/(P(S^{k-1}))$. In our implementation, $\alpha_1 = 0.99$, implying that $P(S^k)$ is close to $P(S^{k-1})$, and $\alpha_2 = 1$ or $\alpha_2 = 0.5$. The rationale for this formula is that if there has been a fairly high level of feasibility violation for some iterations, it appears reasonable to increase $P(S^k)$ in order to discredit solutions which are infeasible in [SCH]. If, on the other hand, there has been a low level of infeasibility, it appears reasonable to decrease the penalty, allowing the 'solution path' to traverse infeasible regions and hopefully find better solutions.

Solutions which do not satisfy the restrictions on the usage of the common resources (7) are penalized by computing

$$\mathrm{TP}^E(\xi^k, S^k) = \sum_{r\in R}\sum_{t\in T}\left(P_r^E(S^k)\frac{\left(\sum_{m\in M} g_{mr}y_{mt} - \bar{E}_r\right)^+}{\bar{E}_r}\right), \tag{12}$$

where $P_r^E(S^k)$ specifies how much to penalize each fraction over-utilization of the resource r. Here we penalize over-utilization of the resources by a constant factor (independent of the state).

In order for the tabu search heuristic to be efficient, a fast evaluation of $\mathrm{c}^m(\xi^k, S^k)$ is crucial. Given a schedule represented by the start-up variables s_{mt}, the variables y_{mt} and $s^{\mathrm{seq}}_{m\hat{m}t}$ can be computed efficiently. Additionally, in order to compute the modified objective function value $\mathrm{c}^m(\xi^k, S^k)$, we need to compute the inventory levels of I_{pt}.

In order to speed up the computation of I_{pt} we utilize the fact that solutions in $N_{\mathrm{cand}}(\xi^k, S^k)$ often differ only with respect to a single start-up compared to the start-ups in ξ^k. This implies that it is necessary to recalculate I_{pt} only for those products p affected by the run-modes altered. Furthermore, only the time periods after the actual change in ξ^k need to be considered, since a change in start-ups at time period $\hat{t}$ only affects inventory levels in periods $t \geq \hat{t}$.

4.3. *Tabu lists*

The number of solutions to investigate in $N_{\mathrm{cand}}(\xi^k, S^k)$ is reduced by using two types of tabu lists associated with different (move) attributes. The first type of tabu list ensures that an element of ξ^k that has switched value from one to zero, cannot switch back again during the next L^k number of iterations and applies to all sub-neighbourhoods except $N_{\mathrm{remove}}(\xi^k)$. The tabu list length, L^k, is implemented to vary

randomly and is obtained by generating a uniformly distributed number between $0.5l^k$ and $2l^k$ in each iteration, where l^k is a parameter used for deciding the tabu list length. Two approaches for deciding the length l^k were tested. In the first approach the length l^k is assigned the constant value l^{st} which in turn is decided by the size of the problem. In the second approach, referred to as using the *varying length* feature, the parameter l^k varies between $0.25\ l^{st}$ and $1.5l^{st}$. After the initial phase of the tabu search is completed, l^k is assigned the value $0.25l^{st}$. Then, it is increased by one unit every 20th iteration until $l^k = 1.5l^{st}$, when it is reset to the value $0.25l^{st}$. With this approach, the heuristic oscillates between periods whose emphasis is on intensification and periods whose emphasis is on diversification.

The second type of tabu lists implemented is associated with a specific sub-neighbourhood. The tabu list associated with $N_{\text{move}}(\xi^k)$ does not allow the reverse move. For moves associated with $N_{\text{switch}}(\xi^k)$, a run-mode which was put before another run-mode in one iteration using $N_{\text{switch}}(\xi^k)$ cannot be put later during some subsequent iterations when using $N_{\text{switch}}(\xi^k)$. The neighbourhood $N_{\text{add}}(\xi^k)$ is treated in the same way by making sure that the same run-mode is not added twice in a row. Based on computational experience, the lengths of the second type of tabu lists are set to four times the periodicities of using the associated sub-neighbourhoods.

In our implementation no aspiration criterion is used to override the tabu moves, not even if a solution is better than the best solution found so far. This strategy reduces the number of necessary solution evaluations.

4.4. *Summary: Main features of the tabu search*

The tabu search presented includes several features, which are used for directing the search. These features are:

- *forced*, which limits the use of the neighbourhood $N_{e/l}$ at iterations when another (more complex) sub-neighbourhood is utilized;
- *reduced*, which implies that the neighbourhood is sampled;
- *locked*, which locks a part of the solution;
- *dynamic penalty*, which allows the penalty level ($P(S^k)$) to vary;
- *varying length*, which makes the tabu list length (l^k) vary.

5. Computational experiments

In this section we report on the results obtained when using the tabu search heuristic on problem instances based on real-life scheduling scenarios from Nynas. The purpose is to evaluate the performance of our proposed tabu search heuristic and to illustrate what features make a difference to its performance. When measuring the performance of the heuristic, we will primarily use the modified objective function ($c^m(\xi_{\text{best}}, S)$) for the best solution found (ξ_{best}). Recall that infeasible solutions with respect to [SCH] are of interest for the planners at the refinery, since we are penalizing with respect to constraints which are somewhat soft.

5.1. *Problem scenarios*

Three different scenarios are used for the computational tests. Scenarios 1 and 2 are based on scheduling situations at Nynas covering a planning horizon of one month, while Scenario 3 covers two months. Scenario 3 represents what might be of interest in the future if the planning horizon is extended and the accuracy in the forecasts of the demand is improved. Scenario 1 includes fewer products compared

to the other two scenarios, and represents the planning situation where the planner has reduced the problem by ignoring some products and run-modes. This reduction has been discussed with planners at the refinery. Scenarios 2 and 3 include negative production costs (c^{y}_{mt}), sequence dependent changeover costs ($c^{\text{seq}}_{m\hat{m}t}$), and a set of limited common resources (R), whereas Scenario 1 does not include these features. The penalties for violating inventory level constraints (5) and resource constraints (7) are set to 0.5 and 100, respectively, which is equivalent to letting $P(S) = P^{I}(S) = P^{\underline{L}}(S)$ be equal to 0.5 in formula (11) and $P^{E}_{r}(S)$ be equal to 100 in formula (12).

Basic characteristics of the three scenarios are given in table 1. In the number of *Products* we include crude oil, input products (components) and end-products. *Processing units* denotes the number of units considered in the different scenarios. The number of *Run-modes* includes both production modes and stop-modes. We also give details of the number of run-modes in each processing unit (*Run-modes per unit*). Further, table 1 shows the number of *Resources*, *Time periods*, and *Start-up variables* (s_{mt}) in the mathematical formulation. For Scenarios 1 and 2, we generated 19 and 10 problem instances, respectively, where the demand patterns were randomly generated to conform to the real situation at the refinery. The randomly generated problem instances were in some cases slightly modified in order to obtain problems with at least one feasible solution with respect to [SCH]. The original demand patterns were also included in the set of problem instances and referred to as the *original instance*. In total we have 20, 11 and 1 problem instances for Scenarios 1, 2 and 3, respectively. No additional problem instances were generated for Scenario 3, since testing the heuristic on this scenario is time consuming and no strong lower bound on the objective function can be obtained for this rather large scenario. The start-up costs for the 20 problem instances of Scenario 1 were set at two different levels, one high and one low (including the original instance), where the lower level was equal to 25% of the higher. Additional details of the scenarios and generated problem instances can be found in Persson (2002).

In the last row of table 1, we present the average of the best-known lower bound (LBD^{SCH}) of the total cost for the given scenarios for formulation [SCH]. For all instances of Scenarios 1 and 2, the lower bound is less than 4% from a known feasible solution to [SCH]; and in 16 problem instances of Scenario 1, the optimal solutions to [SCH] are known. For Scenario 3, the gap between the best-known solution and the lower bound is 10%. These bounds were obtained by using a branch-and-bound procedure combined with valid inequalities and extensive CPU times, of up to four days (for details of this approach see Persson 2002).

Characteristics	Scenario 1	Scenario 2	Scenario 3
Products ($\|P\|$)	13	23	24
Processing units ($\|Q\|$)	3	3	4
Run-modes ($\|M\|$)	14	23	23
Run-modes per unit ($\|M_q\|$)	7/3/4	12/4/7	7/6/7/3
Resources ($\|R\|$)	0	2	2
Time periods ($\|T\|$)	31	31	61
Start-up variables	434	713	1401
Average Lower bound (LBD^{SCH})	1531	1949	3956

Table 1. Characteristics of the test scenarios.

In that method of computing the lower bounds, the sequence-dependent changeover costs ($c^{\text{seq}}_{m\hat{m}t}$) and resource constraints had to be ignored.

In the tests, it is assumed that no specific starting solution is available. We assume that the initial run-modes are known, i.e. it is known which run-modes are used at $t = 0$. We construct the start solution ξ^0, by letting the initial run-modes be used throughout the whole planning horizon.

5.2. *Calibration of the parameters*

In this section, we present experimental results obtained when using different parameter settings. The aim is not to find the optimal parameter setting, but rather to identify reasonably good parameter values. Further, we want to find out whether the performance is highly sensitive to a particular parameter setting or not.

In table table 2, the values of the different parameters are presented. The sub-neighbourhood periodicities are based on empirical observations. The penalty parameter α_2 is equal to 1 for Scenario 1 and to 0.5 for Scenarios 2 and 3. We use different values of α_2 in order to compensate for the different start-up costs, which are slightly lower on average for the problem instances of Scenario 1. The initial penalty factors $P(S^0)$ and $P^E_r(S^0)$ are set to 0.5 and 100, respectively.

The parameter l^{st} is used for determine the tabu list length at each iteration. Limited computational experiments indicated that the values 16, 20, and 24 are reasonable values of l^{st} for the three different scenarios. For the three scenarios, this corresponds to computing l^{st} according to $0.6(|P||M||T|)^{1/3} + 5$ where $|P|$, $|M|$, and $|T|$ represent the total number of products, run-modes, and time periods, respectively.

The neighbourhood $N_{\text{add}}(\xi^k)$ is used in every third iteration during the initial phase. This is reasonable, since in our tests the heuristic is started with a solution including no start-ups at all. We let the initial phase last for $3l^{st}$ iterations.

In the computational tests the heuristic is terminated after a specified number of solution evaluations (denoted K^e). By using this termination criterion (and not the iteration number K), we ensure that each test run on the same problem instance will require almost identical CPU times given that K^e is identical. The reason for this is that a major part of the CPU time is required for evaluating solutions. For Scenario 1, the tabu search heuristic will be stopped after 75 000 solution evaluations, i.e. $K^e = 75\,000$. For Scenarios 2 and 3, we allow 150 000 and 300 000 solution evaluations, respectively. At these numbers of K^e, the heuristic is performing

Parameters	Value
$p_{e/l}$	1
p_{move}	43
p_{switch}	41
p_{add}	49
p_{remove}	45
α_1	0.99
α_2	0.5/1
$P(S^0)$	0.5
$P^E_\tau(S^0), \tau \in R$	100
R^f	0.4

Table 2. Values of the parameters used in the heuristic.

reasonably well, but can still be improved. In all the computational tests, the CPU time is given for a Sun Ultra Sparc, 300 MHz. The 75 000 solution evaluations for Scenario 1 correspond approximately to 4400 iterations and 2 min of CPU time. Correspondingly, 7 and 15 CPU min are utilized for each test run for Scenarios 2 and 3.

When half of the solution evaluations have been completed, the search is restarted with the best solution found so far, i.e. $\xi^k \leftarrow \xi_{\text{best}}$. This is also done when there are 5000 solution evaluations remaining. This feature of restarting is included since the surrounding of solution ξ_{best} might contain even better solutions, which might be found at another restart when the tabu lists are different compared to those the previous time ξ_{best} was found. Further, when using the feature *reduced*, good solutions might have been missed the previous time the solution was found.

First we study the effect on the performance when using different sub-neighbourhood periodicities. It was noted that the performance was relatively insensitive to rather large changes in the different sub-neighbourhood periodicities given in table 2. The performance appears almost insensitive to the individual changes of the periodicity associated with the neighbourhood $N_{\text{move}}(\xi^k)$ and $N_{\text{switch}}(\xi^k)$. A possible explanation is that these two types of neighbourhoods can to a large extent be replaced by each other and by the neighbourhoods $N_{\text{add}}(\xi^k)$ and $N_{\text{remove}}(\xi^k)$. However, if none of the neighbourhoods $N_{\text{move}}(\xi^k)$ and $N_{\text{switch}}(\xi^k)$ are used, i.e. the numbers of p_{move} and p_{switch} are very large, the performance deteriorates.

In figure 2, the result of using different periodicities p_{add} and p_{remove}, when using the heuristic on the original instance of Scenario 1, is plotted. In the figure, the x-axis is given with a logarithmic scale and shows the periodicity for p_{remove}. The periodicity for p_{add} is set to 49/45 times the periodicity of p_{remove} (representing the relation between the original values). The plotted line, $c^m(\xi_{\text{best}}, S^0)$, represents the average modified objective function value obtained for 10 test runs at each setting for the original instance of Scenario 1. The upper plotted line shows the average of the sum of the modified objective function value and the standard deviation $c^m(\xi_{\text{best}}, S^0) + \sigma$.

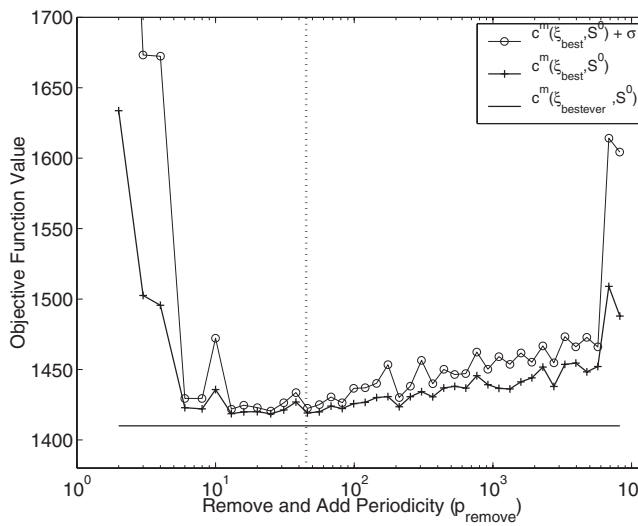

Figure 2. Performance for different add and remove periodicities, plotted using 10 test tuns for each setting on the *original instance* of Scenario 1.

These two values are compared to the best-known solution value to [R-SCH], that is $c^m(\xi_{\text{best–ever}}, S^0)$. The dotted vertical line gives the periodicity suggested in this implementation ($p_{\text{remove}} = 45$).

The computational tests indicate that the performance is relatively insensitive to the choice of these periodicities, at least when using periodicities greater than 10 and smaller than 100. Outside this range the performance sometimes deteriorates at the same time as the spread in the results increases. Additionally it has been observed that deterioration of the performance occurs if one or two of the sub-neighbourhoods are not used at all (i.e. the periodicities are set to a large number).

Additional tests of using different values of periodicities have been performed. Two tests of using a periodicity of 44 for p_{move}, p_{switch}, p_{add} and p_{remove} add and p_{remove} have been carried out. In the first, all the sub-neighbourhoods were used simultaneously every 44th iteration. In the second test, the usage of the sub-neighbourhoods were phase shifted such that one of these neighbourhoods were used every 11th iteration. Additionally tests of including sub-neighbourhoods randomly with a probability corresponding to the suggested periodicity of usage were carried out. None of these tests showed on an improvement of the performance of the heuristic. Instead its performance deteriorated significantly for the case were the sub-neighbourhoods were used simultaneously every 44th iteration.

Further, the performance is relatively insensitive to changes in the parameter l^{st}, which is used for deciding the tabu list length. However, extreme values of l^{st} affect the performance. As an example, we present graphically in figure 3 some results of the performance obtained when using different values of l^{st} on the original instance of Scenario 1. The dotted vertical line corresponds to the value $l^{st} = 16$, which was used for this scenario.

We also carried out additional experiments with different values of the parameter α_2 (used for deciding the penalty factor $P(S^k)$) and with different constant values of $P(S^k)$. The test indicated that the setting of the parameter α_2 has some effect on

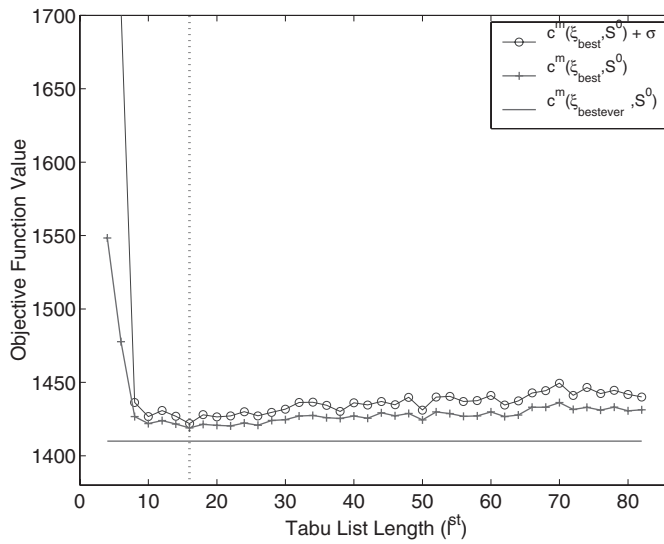

Figure 3. Performance for different lengths of tabu list plotted using 10 test runs for each setting on the original instance of Scenario 1.

the performance of the heuristic. For Scenario 2 and 3, we observed that it is favorable to set the value of α_2 to 0.5.

Overall, our experiments confirm that the parameters can be allowed to vary relatively widely before the performance significantly deteriorates. Hence, the results show the *robustness* of the heuristic with regard to the choice of the parameters used for directing the search.

5.3. *Effect of the different features on the performance of the heuristic*

Next we study the impact of removing different combinations of the features presented in Section 4. We perform these tests in order to assess which features are important for the performance of the heuristic.

First we study the behaviour of the heuristic when all the features are excluded, i.e. the neighbourhood $N_{e/l}(\xi^k)$ is used in every iteration, no sampling is utilized, no part of the solution space is locked, and both the penalty level and the length of the tabu list are kept constant. The results for this case are presented in the first row of table 3. Then we include exclusively one feature at a time and study the change in performance and present the results in the following rows of table 3. For each version, we give the average value of the modified objective function for the best solution found, $c^m(\xi_{\text{best}}, S^0)$, based on 20 runs. In table 3, these values are in turn based on the average results for the 20 constructed problem instances of Scenario 1, therefore the notation 'Scenario 1 (1–20)'. Hence, each row of the table represents the testing of the heuristic 20 times on 20 different problem instances. In the column 'σ ', we display the average standard deviation of $c^m(\xi_{\text{best}}, S^0)$ for each setting, for the 20 test runs. The average level of infeasibility for the best solution found is also included in the table, i.e. the value of $\text{TP}(\xi_{\text{best}}, S^0) = \text{TP}^I(\xi_{\text{best}}, S^0) + \text{TP}^E(\xi_{\text{best}}, S^0)$ is presented. In the last column, we show the fraction in percentage of the test runs, %feas, which produced a feasible solution with respect to [SCH]. In the first row of table 4, we study the objective function value when all the features are included, and in subsequent rows we exclude one feature at a time.

The analysis of including or removing each of the features is also performed for Scenarios 2 and 3. The results are presented in tables 5–8.

The results indicate that by including all the features we always obtain better performance than by not including any of the features. We also note that by including one feature the performance improves or approximately remains the same; see tables 3, 5 and 7. When we remove one feature, the performance of the heuristic worsens or approximately remains the same; see tables 4, 6 and 8. We also note that if we exclude only one of the features *reduced* or *locked*, the performance significantly

	Scenario 1 (1–20)			
Feature included	$c^m(\xi_{\text{best}}, S^0)$	σ	TP (ξ_{best}, S^0)	%feas
None	1827	360	259	66
Only Forced	1756	174	192	70
Only Reduced	1695	199	133	75
Only Locked	1594	62	31	90
Only Dynamic penalty	1758	173	192	68
Only Varying length	1812	191	245	65

Table 3. Tests of including no or just one feature for Scenario 1 (1–20).

Feature included	Scenario 1 (1–20)			
	$c^m(\xi_{\text{best}}, S^0)$	σ	TP (ξ_{best}, S^0)	%feas
All	1569	45	13	93
All but Forced	1577	46	17	89
All but Reduced	1599	68	35	85
All but Locked	1707	203	145	72
All but Dynamic penalty	1580	53	21	91
All but Varying length	1569	38	13	92

Table 4. Tests of including all features or all but one feature for Scenario 1 (1–20).

Feature included	Scenario 2 (1–11)			
	$c^m(\xi_{\text{best}}, S^0)$	σ	TP (ξ_{best}, S^0)	%feas
None	2178	193	145	71
Only Forced	2165	143	127	75
Only Reduced	2073	136	78	86
Only Locked	2082	88	48	85
Only Dynamic penalty	2162	142	142	63
Only Varying length	2196	176	161	66

Table 5. Tests of including no or just one feature for Scenario 2 (1–11).

Feature included	Scenario 2 (1–11)			
	$c^m(\xi_{\text{best}}, S^0)$	σ	TP (ξ_{best}, S^0)	%feas
All	2012	28	24	89
All but Forced	2007	27	25	91
All but Reduced	2067	93	50	83
All but Locked	2040	114	58	82
All but Dynamic penalty	2013	30	18	96
All but Varying length	2028	37	23	94

Table 6. Tests of including all features or all but one feature for Scenario 2 (1–11).

Feature included	Scenario 3			
	$c^m(\xi_{\text{best}}, S^0)$	σ	TP (ξ_{best}, S^0)	%feas
None	8125	2790	3683	0
Only Forced	8879	3456	4445	0
Only Reduced	5473	748	1053	15
Only Locked	6064	996	1621	0
Only Dynamic penalty	8350	2260	3922	0
Only Varying length	7860	2843	3410	0

Table 7. Tests of including no or just one feature for Scenario 3.

Feature included	Scenario 3			
	$c^m(\xi_{best}, S^0)$	σ	TP (ξ_{best}, S^0)	%feas
All	5187	909	763	5
All but Forced	5660	1040	1246	0
All but Reduced	6006	932	1555	0
All but Locked	5437	861	1008	5
All but Dynamic penalty	5393	738	988	5
All but Varying length	5191	733	800	10

Table 8. Tests of including all features or all but one feature for Scenario 3.

worsens. These two features also appear to significantly improve the performance if exclusively included in the heuristic.

5.4. *Performance of the heuristic*

In this section, we study the impact on the performance of the heuristic when we let it run for different CPU times by using different values of the parameter K^e (total number of evaluations). In figures 4–6, we study the performance of the heuristic when it performs 20 test runs with different values of K^e for each problem instance. Based on these runs, we present the average modified objective function, $c^m(\xi_{best}, S^0)$, for each scenario. As a comparison, we present the average of the modified objective function for the best-ever found solution (best-known solution), denoted $c^m(\xi_{best-ever}, S^0)$. We also present the objective function value of the average best found feasible solution of each run with respect to [SCH] ($c(\xi_{f-best})$). In the lower parts of the figures, we display the average fraction of the runs which produced a feasible solution. We also plot the average lower bounds (LBD^{SCH}) of the [SCH] formulation for the corresponding scenario.

In figures 4–6 a trend can be observed which indicates that larger numbers of K^e give better performance. Hence, the suggested heuristic is not a simple local search procedure, but the tabu lists and other features allow the heuristic to escape local optimal solutions. The tabu search heuristic finds solutions which on average have objective function values which are relatively close to the lower bound of [SCH], at least for Scenarios 1 and 2. Also the average best solution found ($c^m(\xi_{best}, S^0)$) in each run is relatively close to the average best-ever found solution ($c^m(\xi_{best-ever}, S^0)$. The fraction of runs finding a feasible solution for Scenario 3 is rather low; one explanation is that the long time horizon makes it difficult for the heuristic to find a solution, which in all aspects is feasible with respect to [SCH]. Further, whenever a feasible solution is found it has a rather good objective function value compared to $c^m(\xi_{best}, S^0)$. The reason for this is probably that the penalty cost parameters are rather high and give a large impact on the average objective function ($c^m(\xi_{best}, S^0)$) in those runs when no feasible solution was found.

5.5. *Comparison with branch-and-bound*

In the following we compare the performance of the tabu search heuristic with the performance obtained when using the branch-and-bound (B&B) method presented in Persson (2002). In that thesis, valid inequalities are added to problem [SCH] in order to improve the lower bound of the linear relaxation, and thereby decrease the number of solutions which are enumerated by the B&B procedure.

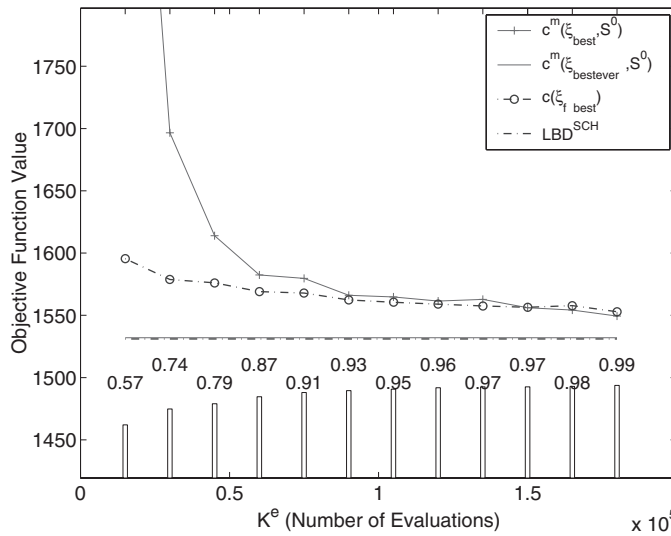

Figure 4. The average performance for different K^e for 20 instances of Scenario 1.

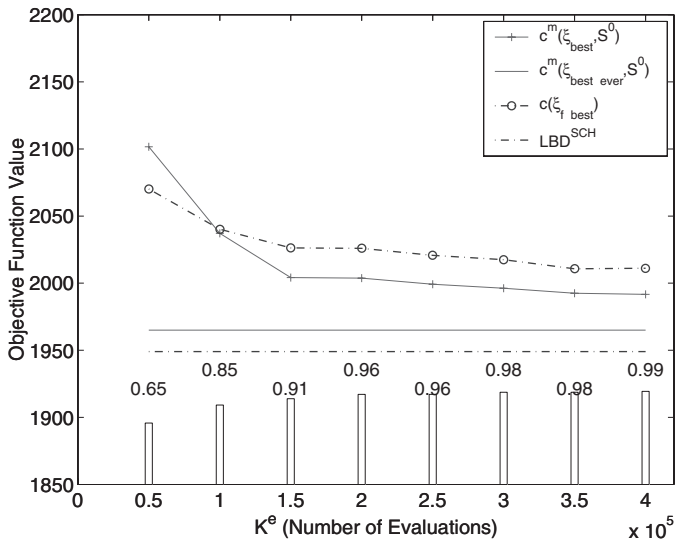

Figure 5. The average performance for different K^e for 11 instances of Scenario 2.

In Persson (2002) it was noted that using B&B on a model without valid inequalities was not very successful on this type of problem. The valid inequalities developed can only be used together with formulation [SCH] and not [R-SCH]. Thus we can only test the B&B approach together with formulation [SCH] when using the B&B approach, sequence-dependent costs and resource constraints are ignored, otherwise the model becomes too large for using a B&B solution procedure. The purpose of this comparison is to check if a B&B approach can compete with the tabu search heuristic in finding good schedules in reasonable time for this problem.

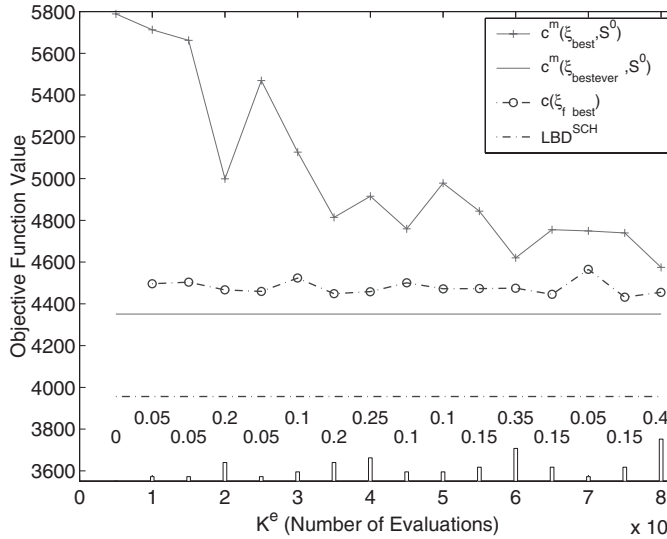

Figure 6. The average performance for different K^e for 1 instance of Scenario 3.

In table 9, we compare the solution approaches for the problem instances of the three different scenarios. Under the heading 'Tabu search', we present the CPU time in min for each test run, the average modified objective function value ($c^m(\xi_{\text{best}}, S^0)$, and the average standard deviations of these values (σ), based on 20 test runs for each problem instance. We also present the average value of the penalty term ($P(\xi_{\text{best}}, S^0)$) and the average objective function value of the best solution found ($c(\xi_{\text{f–best}})$) with respect to formulation [SCH]. Within parentheses we also present for $c^m(\xi_{\text{best}}, S^0)$ and $P(\xi_{\text{best}}, S^0)$ the best result obtained from any of the 20 test runs. The average value of $c(\xi_{\text{f–best}})$ is based on those test runs when a feasible solution was found and we give the percentage of such runs in the table (%feas). The results are given for 150 000, 300 000, and 600 000 solution evaluations, respectively, for the three scenarios. Other parameters are set as in previous tests (see table 2). In the table, we also include average values of all problem instances for Scenarios 1 and 2, which is presented in the rows '1,(1–20)' and '2,(1–11)'.

Under the heading 'Branch-and-bound', the results are given when using the method suggested in Persson (2002). We used CPLEX 6.5 with its default setting and terminated when the specified CPU time or memory allowance was exceeded. The LBD^{SCH} presented is the average lower bound obtained from the test runs and it is a lower bound to formulation [SCH]. (After these tests were completed, we were able to improve the lower bounds additionally by using a faster computer and longer computation times; these results were presented in table 1). $\mathcal{V}$ represents the best feasible solution found during the branch-and-bound process, but with the sequence-dependent costs and resource constraints ignored, and hence it is not a true upper bound for Scenarios 2 and 3.

The tabu search heuristic finds on average solutions with lower costs in less CPU time. One important reason for this is that the branch-and-bound procedure was unable to find a feasible solution at all during the specified CPU time. Note that the branch-and-bound solution procedure was terminated due to time or memory

Scenario instance	Tabu search						Branch-and-bound		
	CPU	$c^m(\xi_{best}, S^0)$	σ	$P(\xi_{best}, S^0)$	$c(\xi_{f-best})$	%feas	CPU	LBD^{SCH}	ν
1, 1	4	1576 (1568)	2	0	1576 (1568)	100	4	1568	1568
1, 2	4	1722 (1708)	13	0	1722 (1708)	100	20	1708	1708
1, 3	4	1801 (1752)	66	30	1779 (1752)	85	18	1750	1750
1, 4	4	1781 (1774)	9	0	1781 (1774)	100	12	1770	1770
1, 5	4	1648 (1642)	12	8	1650 (1642)	100	180	1613	1940
1, 6	4	1825 (1801)	28	1	1830 (1801)	100	15	1801	1801
1, 7	4	1773 (1754)	41	17	1764 (1754)	95	10	1754	1754
1, 8	4	1749 (1732)	9	11	1756 (1752)	100	35	1752	1752
1, 9	4	1816 (1752)	57	43	1882 (1756)	95	16	1751	1751
1, 10	4	1752 (1726)	13	10	1756 (1726)	100	10	1724	1724
1, 11	4	1337 (1335)	4	0	1337 (1335)	100	30	1335	1335
1, 12	4	1346 (1334)	8	0	1346 (1334)	100	180	1334	1334
1, 13	4	1314 (1288)	12	0	1314 (1288)	100	180	1248	1282
1, 14	4	1299 (1274)	16	0	1299 (1274)	100	14	1274	1274
1, 15	4	1409 (1393)	7	0	1409 (1393)	100	97	1392	1392
1, 16	4	1343 (1337)	8	0	1343 (1337)	100	180	1321	1460
1, 17	4	1438 (1420)	8	14	1455 (1446)	80	152	1394	1584
1, 18	4	1338 (1312)	20	0	1338 (1312)	100	1	1307	1307
1, 19	4	1356 (1336)	16	1	1358 (1336)	100	70	1327	1327
1, 20	4	1465 (1449)	14	9	1463 (1449)	90	153	1438	1438
1, (1–20)	4	1554 (1534)	18	7	1558 (1537)	97	69	1528	1563
2, 1	14	2074 (2063)	6	46	2104 (2065)	100	360	1961	2053
2, 2	14	2051 (2019)	37	52	2068 (2019)	100	360	1975	2045
2, 3	14	1966 (1942)	23	3	1967 (1942)	100	278	1941	1941
2, 4	14	1959 (1923)	21	9	1975 (1923)	90	360	1902	1921
2, 5	14	2153 (2137)	14	9	2174 (2137)	100	360	2078	2147
2, 6	14	1803 (1783)	12	90	1878 (1790)	90	54	1765	1765
2, 7	14	1855 (1808)	32	1	1865 (1816)	100	360	1783	2291
2, 8	14	2042 (2021)	21	6	2046 (2021)	100	244	1991	1991
2, 9	14	2178 (2140)	16	33	2202 (2164)	100	360	2064	2346
2, 10	14	1990 (1966)	22	2	1992 (1966)	100	19	1946	1946
2, 11	14	1887 (1878)	18	7	1922 (1910)	100	56	1898	1898
2, (1–11)	14	1996 (1971)	20	24	2017 (1978)	98	256	1937	2031
3	30	4620 (4353)	189	207	4475 (4423)	35	2600	3956	–

Table 9. Results obtained by the tabu search and by a branch and bound with valid inequalities.

requirements, hence, relatively large gaps between $\mathcal{V}$ and LBD remained at termination.

It can be noticed that the average penalty cost, $P(\xi_{best}, S^0)$, is relatively small for all the solutions obtained with the tabu search heuristic. For example, for Scenario 1, an average of 7 was obtained, which corresponds to that at most $7/0.5 = 14$ tonnes of products are not within specified limits (since $P(S^0) = 0.5$). This should be compared to the 120 000 tonnes of products delivered on average in problem instances of Scenario 1; and 140 000 and 260 000 tonnes for Scenarios 2 and 3, respectively. If the positive values of $P(\xi_{best}, S^0)$ are assumed to be caused by resource constraint violations, the infeasibility can also be regarded as rather low. Hence, the solutions ξ_{best} are typically 'almost feasible' with respect to [SCH], which is a desirable feature.

Table 9 indicates that the standard deviation of the objective function value is relatively high for some problem instances (in particular this is the case for Scenario 3). The high value mainly stems from one or two fairly 'bad runs' out of the 20 test runs.

6. Conclusions and future research

We have studied how a tabu search heuristic can be utilized for finding good solutions to the process scheduling problem at an oil refinery. It has been shown that the decision problem can be formulated as a mixed integer optimization model. For a solution (a schedule) we define a number of sub-neighbourhoods. A strategy for utilizing these sub-neighbourhoods is suggested, which makes the definition of the neighbourhood vary.

We have introduced a number of features, which all affect the intensification and diversification of the search. Most of these features significantly improve the performance of the heuristic. In particular, it has been shown that sampling of the neighbourhood is important for the performance of the heuristic. Intensifying the search around the current solution by partially locking the solution appears also to be important for the performance.

For the CPU times used in the computational tests, the tabu search heuristic performed better than another tested solution approach based on (incomplete) branch-and-bound combined with the generation of valid inequalities. Further, the lower bounds obtained by using the branch-and-bound solution approach allowed us to verify that the tabu search heuristic performed relatively well on the tested scenarios.

The tabu search heuristic can be enhanced by implementing dynamic tuning of the parameters. Another potential way of improving the heuristic is to explore alternative strategies for utilizing the sub-neighbourhoods. In our experiments we considered strategies of diversification by starting from the current iterate solution and then by using the tabu list during the diversification phase. Another approach would be to diversify with the help of constructive heuristics, where information about previous good solutions is utilized. Such ideas have been explored recently in Gendron *et al* (2002) for a network design problem.

Appendix A: The optimization model

Let T denote the set of time periods, and let Q denote the set of processing units at the refinery. The set of possible run-modes at processing unit q, $q \in Q$, is denoted by M_q, and the union of all possible run-modes at all processing units is denoted by M. The set of resources, consumed when using different run-modes, is denoted R. Finally, let P denote the set of products considered in the model. Below we present input data to the model, where $p \in P, t \in T, r \in R,$ and $m, \hat{m} \in M$.

c^I_{pt} = inventory cost for product p at time period t.
$c^{\text{seq}}_{m\hat{m}t}$ = changeover cost for changing from run-mode m to $\hat{m}$ in time period t.
c^s_{mt} = start-up cost for run-mode m in time period t.
c^v_{mt} = production cost for using run-mode m in time period t.
a_{pm} = production yield of product p when operating run-mode m.
b_{pm} = consumption of product p when operating run-mode m.
d_{pt} = demand for product p in time period t.

$\underline{I}_{pt}$ = lower limit on the inventory level for product p at time period t.
$\bar{I}_{pt}$ = upper limit on the inventory level of product p at time period t.
g_{mr} = the use of resource r when using run-mode m.
$\bar{E}_r$ = the available amount of resource r in any time period.

We also define the following variables, where $p \in P, t \in T,$ and $m, \hat{m} \in M$.

x_{pt} = production of product p during time period t.
z_{pt} = consumption of product p during time period t.
I_{pt} = inventory level of product p at the end of time period t.
$y_{mt} = \begin{cases} 1 & \text{if run-mode } m \text{ is used in time period } t, \\ 0 & \text{otherwise.} \end{cases}$
$s^{seq}_{mmt} = \begin{cases} 1 & \text{if run-mode } m \text{ is changed to } \hat{m} \text{ between time period } t-1 \text{ and } t, \\ 0 & \text{otherwise.} \end{cases}$
$s_{mt} = \begin{cases} 1 & \text{if run-mode } m \text{ is started in time period } t, \\ 0 & \text{otherwise.} \end{cases}$

The following mixed integer linear programming model for the process scheduling problem can then be formulated.

$$[SCH] \min \sum_{t\in T}\sum_{p\in P} c^I_{pt} I_{pt} + \sum_{t\in T}\sum_{m\in M}\sum_{\hat{m}\in M:\hat{m}\neq m} c^{seq}_{m\hat{m}t}\, s^{seq}_{m\hat{m}t} + \sum_{t\in T}\sum_{m\in M} c^s_{mt} s_{mt} + \sum_{t\in T}\sum_{m\in M} c^y_{mt} y_{mt}, \tag{13a}$$

$$x_{pt} - \sum_{m\in M} a_{pm} y_m t = 0, \qquad p \in P, t \in T, \tag{13b}$$

$$z_{pt} - \sum_{m\in M} b_{pm} y_m t = 0, \qquad p \in P, t \in T, \tag{13c}$$

$$I_{p,t-1} + x_{pt} - z_{pt} - d_{pt} = I_{pt}, \qquad p \in P, t \in T, \tag{13d}$$

$$\underline{I}_{pt} \le I_{pt} \le \bar{I}_{pt}, \qquad p \in P, t \in T, \tag{13e}$$

$$\sum_{m\in M_q} y_{mt} = 1, \qquad q \in Q, t \in T, \tag{13f}$$

$$s^{\text{seq}}_{m\hat{m}t} + 1 \ge y_{m,t-1} + y_{\hat{m}t}, \qquad q \in Q, m \neq \hat{m} \in M_q, t \in T, \tag{13g}$$

$$y_{m,t-1} + s_{mt} \ge y_{mt}, \qquad m \in M, t \in T, \tag{13h}$$

$$\sum_{m\in M} g_{mr} y_{mt} \le \bar{E}_r, \qquad r \in R, t \in T, \tag{13i}$$

$$I_{pt}, x_{pt}, z_{pt} \ge 0, \qquad p \in P, t \in T, \tag{13j}$$

$$s^{\text{seq}}_{m\hat{m}t} \in \{0,1\}, \qquad m, \hat{m} \in M, t \in T, \tag{13k}$$

$$y_{mt}, s_{mt} \in \{0,1\}, \qquad m \in M, t \in T. \tag{13l}$$

Appendix B: Examples of the sub-neighbourhoods

In this section we first introduce an example of a processing schedule for a small problem. Then we use that example for illustrating the different neighbourhoods used in the tabu search heuristic.

Example 1. *Consider the scheduling of a single processing unit during six time periods in which three run-modes* $M0$, $M1$, and $M2$ *can be used. A schedule (or a solution) represented by the variables* y_{mt} *is given in table 10, where* $y_{m0} = \{0, 0, 1\}$ *are*

given and represent the initial state of the processing unit (illustrated in the first row for $t = 0$). This particular solution says that run-mode M2 is used in period 1, then run-mode M1 is started in time period 2 and used until run-mode M0 is started in time period 5.

The schedule of the example can also be expressed in variables s_{mt}, which is given in the left part of table 11. We now illustrate the sub-neighbourhood $N_{e/l}(\xi^k)$. Let the solution of the example above be our current iterate solution ξ^k. The solutions in the set $N_{e/l}(\xi^k)$ are illustrated in the right part of table 11, where the solutions are represented by the variables s_{mt}. A solution of $N_{e/l}(\xi^k)$ is obtained by changing one of the four '0/1' positions from zero to one and the nearby '1/0' from one to zero.

Let the candidate move underlined in table 11 be selected. The next iterate solution, ξ^{k+1}, is illustrated in table 12, in which bold face highlights the changes compared to the previous solution, ξ^k.

The sub-neighbourhood $N_{\text{move}}(\xi^k)$ is exemplified in table 13 for the solution of the example, where a '0/1' may change from zero to one. Since we do not allow the start-up of a run-mode that is already in use or the start-up of a run-mode that is started next, there is only a single solution in $N_{\text{move}}(\xi^k)$ as illustrated in table 13.

The sub-neighbourhood $N_{\text{switch}}(\xi^k)$ is exemplified in table 14, which is based on the schedule of the example above.

The sub-neighbourhood $N_{\text{add}}(\xi^k)$ is illustrated in table 15, based on ξ^k of the example. Positions in the schedule where start-ups can be performed are illustrated with the notation '0/1' in table 15.

	y_{mt}		
t	M0	M1	M2
0	0	0	1
1	0	0	1
2	0	1	0
3	0	1	0
4	0	1	0
5	1	0	0
6	1	0	0

Table 10. An example of a solution expressed in variables y_{mt}.

	ξ^k, (s_{mt})			$N_{e/l}(\xi^k)$		
t	M0	M1	M2	M0	M1	M2
1	0	0	0	0	0/1	0
2	0	1	0	0	1/<u>0</u>	0
3	0	0	0	0	0/<u>1</u>	0
4	0	0	0	0/1	0	0
5	1	0	0	1/0	0	0
6	0	0	0	0/1	0	0

Table 11. Sub-neighborhood $N_{e/l}(\xi^k)$ illustrated for the solution $\xi^k$of the example.

ξ^{k+1}, (s_{mt})			
t	M0	M1	M2
1	0	0	0
2	0	**0**	0
3	0	**1**	0
4	0	0	0
5	1	0	0
6	0	0	0

Table 12. The solution ξ^{k+1} expressed in the start-up variables s_{mt}.

ξ^k (s_{mt})			N_{move} (ξ^k)		
t	M0	M1	M2	M0	M1	M2
1	0	0	0	0	0	0
2	0	1	0	0	1	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	1	0	0	1/0	0	0/1
6	0	0	0	0	0	0

Table 13. Sub-neighborhood $N_{\text{move}}(\xi^k)$ illustrated for the solution ξ^k of the example.

ξ^k (s_{mt})			N_{switch} (ξ^k)		
t	M0	M1	M2	M0	M1	M2
1	0	0	0	0	0	0
2	0	1	0	0/1	1/0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	1	0	0	1/0	0/1	0
6	0	0	0	0	0	0

Table 14. Sub-neighbourhood N_{switch} (ξ^k) illustrated for the solution ξ^k of the example.

ξ^k, (s_{mt})			N_{add} (ξ^k)		
t	M0	M1	M2	M0	M1	M2
1	0	0	0	0/1	0	0
2	0	1	0	0	1	0
3	0	0	0	0	0	0/1
4	0	0	0	0	0	0/1
5	1	0	0	1	0	0
6	0	0	0	0	0/1	0/1

Table 15. Sub-neighbourhood N_{add} (ξ^k) illustrated for the solution ξ^k of the example.

References

AMOS, F., RÖNNQVIST, M. and GILL, G., 1997, Modelling the pooling problem at the New Zealand refining company. *Journal of the Operational Research Society*, **48**, 767–778.

BALLINTIJN, K., 1993, Optimization in refinery scheduling: Modeling and solution. In T. A. Ciriani and R. C. Leachman, (eds), *Optimization in Industry* (Chichester: John Wiley), chapter 10, pp. 191–199.

CONSTANTINO, M., 1995, *A Polyhedral Approach to Production Planning Models: Start-Up Costs and Times, Upper and Lower Bounds on Production*. Ph.D. thesis, Faculté des Science, Université Catholique de Louvain, Belgium.

CONSTANTINO, M., 1996, A cutting plane approach to capacitated lot-sizing with start-up costs. *Mathematical Programming*, **75**, 353–376.

COXHEAD, R. E., 1994, Integrated planning and scheduling systems for the refining industry. In T. A. Ciriani and R. C. Leachman, (eds), *Optimization in Industry 2* (Chichester: John Wiley), chapter 14, pp. 185–199.

DEWITT, C. W., LASDON, L. S., WAREN, A. D., BRENNER, D. A. and MELHEM, S. A., 1989, OMEGA: An improved gasoline blending system for Texaco. *Interfaces*, **19**, 85–101.

DREXL, A. and KIMMS, A., 1997, Lot sizing and scheduling – survey and extensions. *European Journal of Operational Research*, **99**, 221–235.

GENDRON, B., POTVIN, J.-Y., and SORIANO, P., 2002, Diversification strategies in local search for a nonbifurcated network loading problem. *European Journal of Operational Research*, **142**, 231–241.

GLOVER, F., 1989, Tabu search – part I. *ORSA Journal on Computing*, **1**, 190–206.

GLOVER, F., 1990, Tabu search – part II. *ORSA Journal on Computing*, **2**, 4–32.

GLOVER, F. and LAGUNA, M., 1993, Tabu search. In C. R. Reeves, (ed), *Modern Heuristic Techniques for Combinatorial Problems* (Oxford: Blackwell Science), chapter 3, pp. 70–142.

GÖTHE-LUNDGREN, M., LUNDGREN, J. T. and PERSSON, J. A., 2002, An optimization model for refinery production scheduling. *International Journal of Production Economics*, **78**, 255–270.

KARMARKAR, U. S. and SCHRAGE, L., 1985, The deterministic dynamic product cycling problem. *Operations Research*, **33**, 326–345.

KIMMS, A., 1996, Competitive methods for multi-level lot sizing and scheduling: Tabu search and randomized regrets. *International Journal of Production Research*, **34**, 2279–2298.

KIMMS, A., 1999, A genetic algorithm for multi-level, multi-machine lot sizing and scheduling. *Computers and Operations Research*, **26**, 829–848.

KIMMS, A. and DREXL, A., 1998, Proportional lot sizing and scheduling: some extensions. *NETWORKS*, **32**, 85–101.

LEE, H., PINTO, J. M., GROSSMANN, I. E. and PARK, S., 1996, Mixed-integer linear programming model for refinery short-term scheduling of crude oil unloading with inventory management. *Industrial & Engineering Chemistry Research*, **35**, 1630–1641.

PERSSON, J. A., 2002, *Production Scheduling and Shipment Planning at Oil Refineries: Optimization Based Methods*. Ph.D. thesis, Department of Mathematics, Linköping University, Sweden.

POCHET, Y. and WOLSEY, L. A., 1995, Algorithms and reformulations for lot sizing problems. In W. Cook, L. Lovász, P. Seymour, (eds), *Combinatorial Optimization*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol 20, pp. 245–293.

REKLAITIS, R. V., 1996, Overview of scheduling and planning of batch process operations. In G. V. Reklaitis, A. K. Sunol, D. W. T. Rippin, Ö. Hortacsu, (eds), *Batch Processing Systems Engineering*, Springer, NATO ASI Series F, pp. 660–705.

RIGBY, B., LASDON, L. S. and WAREN, A. D., 1995, The evolution of Texaco's blending systems: From OMEGA to Star Blend. *Interfaces*, **25**, 64–83.

SHAH, N., 1996, Mathematical programming techniques for crude oil scheduling. *Computers and Chemical Engineering*, **20**,1227–1232.