The Economic Lot Scheduling Problem under Power-Of-Two Policy

MING-JONG YAO*
Department of Industrial Engineering, P.O. Box 985
Tunghai University, Taichung City, 40407 Taiwan, R.O.C.
myao@ie.thu.edu.tw

S. E. ELMAGHRABY
Department of Industrial Engineering and Graduate Program in Operations Research
P.O. Box 7906, North Carolina State University, Raleigh, NC 27695, U.S.A.

(Received and accepted May 2000)

Abstract—We present further analysis on the economic lot scheduling problem (ELSP) without capacity constraints under power-of-two (PoT) policy. We explore its optimality structure and discover that the optimal objective value is piece-wise convex. By making use of the junction points of this function, we derive an effective (polynomial-time) search algorithm to secure a global optimal solution. The conclusions of this research lay the foundation for deriving an efficient heuristic, and also creates a benchmark for evaluating the quality of the heuristics for the conventional ELSP under PoT policy. © 2001 Elsevier Science Ltd. All rights reserved.

Keywords—Inventory, Scheduling, Power-of-two policy, Global optimum, Search algorithm.

1. INTRODUCTION AND PERSPECTIVE

This study presents an analysis of the economic lot scheduling problem (ELSP) without capacity constraints under power-of-two (PoT) policy. In this section, we provide some background on the ELSP and PoT policy, and explain the motivation to its study.

1.1. Background: The ELSP

The conventional ELSP is concerned with scheduling the cyclical production of \( n \geq 2 \) products on a single facility in equal lots over the infinite planning horizon, assuming stationary and known demand for each item. The objective of the ELSP is to determine the lot size and the schedule of production of each item so as to minimize the total cost incurred per unit time. The costs considered include the (stationary) setup costs and inventory holding costs.

A production plan in the context of ELSP usually schedules the items within 'basic periods', where a basic period (b.p.), denoted by \( B \), is an interval of time that is devoted to the setup and production of a subset (or all) of the products. The solution of the ELSP is the set of multipliers \( K(B) = \{k_i \mid B\} \) and the b.p. in which each product is produced.

There is extensive literature on the solution methodologies for solving the ELSP; one may refer to [1–3] for reviews. The problem formulations for the ELSP that use b.p.s can be classified as

*Author to whom all correspondence should be addressed.
either the basic period-approach (BP) or the 'extended basic period-approach' (EBP). The BP-approach assumes that the production runs of all products shall be made in each b.p. Then the b.p. must be long enough to accommodate the production of all the products. This is a rather restrictive condition which usually results in suboptimal solutions. The EBP-approach removes this restriction and admits the possibility that in any b.p. only a subset of the products shall be produced. This obviates the waste of capacity of the production facility.

The ELSP, using the concept of b.p. as its foundation, may be formulated as a nonlinear integer program as follows.

**Problem ELSP (GI).**

Minimize \( \text{TC} (B, \{k_i\}) = \sum_{i=1}^{n} \frac{a_i}{k_i B} + \frac{h_i}{2} d_i (1 - \rho_i) k_i \), \( \text{(1a)} \)

subject to \( \sum_{i=1}^{n} [s_i + \rho_i k_i B] \leq B \), \( \text{(1b)} \)

\( k_i : \text{integer}, \ k_i \in \{1, 2, \ldots\} \), \( \text{(1c)} \)

where

- \( d_i \) = demand rate for product \( i \),
- \( a_i \) = the set-up cost for product \( i \),
- \( h_i \) = the holding cost per unit per unit time for product \( i \),
- \( p_i \) = production rate for product \( i \), and
- \( s_i \) = setup time for product \( i \),
- \( \rho_i = \frac{d_i}{p_i} \).

The GI in the identification of the problem indicates that the model is formulated under general-integer policy, as indicated by constraints (1c), which require that the \( k_i \)s be positive integers. The term \( \rho_i k_i B \) in (1b) measures the processing time of a production run of product \( i \). Using the feasibility conditions derived by Davis [4], the ELSP using the EBP-approach can also be formulated as another integer nonlinear program by replacing (1b) with a set of capacity constraints. (See [1,4-6].)

The solution methodologies for the ELSP that have been proposed so far may be divided into two major categories: analytical and heuristic. For a given value of the b.p. \( B \), the analytical approaches usually employ either dynamic programming (DP) or integer nonlinear programming (INLP) models, see [1,7,8] for DP models and [3-5] for binary integer programs. It has been proven by Hsu [9] that the ELSP is NP-hard. Since solving DP models is in fact implicit enumeration, and the branch-and-bound algorithms for solving integer programs require excessive bookkeeping loads, it takes long run times for these analytic approaches to solve relatively 'small' problems of, say, 10-products. The solution of large-scale ELSP problems seems to be out of reach for these analytical approaches. The heuristic approaches (see [5,6,10-15]) suffer from the fact that none is able to guarantee the quality of its solution, or even guarantee convergence of its search scheme, (e.g., oscillatory behavior may happen while modifying the multipliers, see [16] for an example).

### 1.2. The Power-of-Two Policy

The power-of-two (PoT) policy requires that \( k_i = 2^p, \ p \geq 0; \) integer, for all \( k_i \) in the set of multipliers \( K(B) \). Recently, PoT policy became quite popular for lot sizing problems. Roundy [17] presents a special case of the ELSP where the capacity of the production facility is defined by
the annual available setup time. Jackson, Maxwell and Muckstadt [18] focus on the joint replenishment problem, which is actually a special case of ELSP where the capacity of the production facility is unlimited (i.e., infinite capacity). Federgruen and Zheng [19] use "unrestricted (nested or otherwise) and stationary power-of-two policies" for multistage production and inventory systems.

Several reasons support the adoption of the PoT policy. It is interesting from a theoretical point of view since several algorithms and worst case bounds may be derived, an advantage not shared by other procedures. Under PoT policy, researchers were able to derive some easy and effective heuristics to solve both uncapacitated and capacitated lot sizing problems. It is also interesting from a practical point of view since the worst case bounds for PoT policy are actually reasonably tight. For example, Jackson, Maxwell and Muckstadt [18] derive a 94% bound, while Roundy [17] and Federgruen and Zheng [19] provide a 98% bound.

1.3. The Motivation to Study the ELSP (PoT) without Capacity Constraints

Our study on the ELSP without capacity constraints under PoT policy is motivated by a desire to develop a reasonably efficient procedure for the determination of the lot sizes and their timing (schedule). It is based on our observation of the plot of the optimal total cost (of the conventional ELSP (GI)) as a function of the size of the b.p. B (such a plot is presented in Figure 1). It seems that, to date, no study has been made on the optimality structure of the lot sizing problems under PoT policy; a gap in our understanding of the problem, which we hope this research fills.

The variation of the optimal value TC (B, {ki}) with the length of the b.p. may be observed as follows. For a given value of B, we use a binary-expansion of the nonlinear integer program presented in (1), see equations (3)–(5). Denote by \( K_{\text{BIP}} (B) \) the set of optimal multipliers secured by the BIP model at a value of \( B \). When one plots the optimal value (the 'total cost', TC) of the BIP model, i.e., the \( TC_{\text{BIP}}(K_{\text{BIP}}(B), B) \) as a function of \( B \), it shows the following. (See Figure 1. The data for the example in Figure 1 is from Example 6 in [20]. The example is also used in [21].)

1. The function is piecewise convex over intervals of \( B \), with the latter varying in width as \( B \) ranges over its feasible values.

![Figure 1. The optimal solutions of the BIP model.](image-url)
2. THE OPTIMAL FUNCTION

Problem unconstrained ELSP (PoT) may be formulated as a nonlinear integer program as follows.

Problem Unconstrained ELSP (PoT).

\[
\begin{align*}
\text{Minimize} & \quad TC_{\text{PoT}}(B, \{k_i\}) = \sum_{i=1}^{n} \frac{a_i}{k_iB} + \frac{h_i}{2} d_i (1 - \rho_i) k_i B, \\
\text{subject to} & \quad k_i = 2^{m_i}; \quad m_i \in \{0, 1, 2, \ldots \}. 
\end{align*}
\]

For a given value of \(B\), we restate the objective function of problem as a linear, binary integer programming BIP model. The key to rewriting the nonlinear objective function is the binary expansion of any integer \(k_i\) and its reciprocal:

\[
\begin{align*}
k_i &= 2^0 x_{i0} + 2^1 x_{i1} + \cdots + 2^{v_i} x_{iv_i}, \\
k_i^{-1} &= 2^0 x_{i0} + 2^{-1} x_{i1} + \cdots + 2^{-v_i} x_{iv_i},
\end{align*}
\]

and

\[
\sum_{j=0}^{v_i} x_{ij} = 1,
\]

where \(v_i\) is a nonnegative integer, \(x_{ij}\) are binary variables, and \(2^{v_i} B\) is the upper bound on the production cycle \(T_i\). (Refer to the Appendix for further details.) Upon substituting equations (3)–(5) into (2), we obtain

\[
\begin{align*}
\text{minimize} & \quad TC_2 = \sum_{i=1}^{n} \frac{a_i}{B} \left( \sum_{j=0}^{v_i} 2^{-j} x_{ij} \right) + \sum_{i=1}^{n} \frac{h_i}{2} d_i (1 - \rho_i) \left( \sum_{j=0}^{v_i} 2^j x_{ij} \right) B \\
& \quad = \sum_{i=1}^{n} \sum_{j=0}^{v_i} \left\{ \frac{a_i}{B} 2^{-j} x_{ij} + \frac{h_i}{2} d_i (1 - \rho_i) 2^j x_{ij} B \right\}, \\
\text{subject to} & \quad \sum_{j=0}^{v_i} x_{ij} = 1, \quad \text{for } i = 0, 1, \ldots, n, \\
\text{where} & \quad x_{ij} \in \{0, 1\}, \quad \text{for all } i, j.
\end{align*}
\]

The objective function in (6) is still a nonlinear function. However, for a given \(B\), the objective function is separable; actually, (6) becomes a linear BIP model which can be solved by a number of software packages, e.g., LINGO.
2.1. The Small-Step Search Procedure (sssp)

An obvious, albeit tedious, approach to secure the optimal solution to the unconstrained ELSP (PoT) is through the small-step search procedure (sssp), proceeding in decrements of magnitude \( \Delta \) from high to low values of \( B \). The sssp starts its search at \( T_{cc} \), i.e., the cycle time of the common cycle approach (see [1,22]), where

\[
T_{cc} = \max \left\{ \frac{2 \sum_{i=1}^{n} d_i}{\sum_{i=1}^{n} s_i}, \frac{\sum_{i=1}^{n} \rho_i}{1 - \sum_{i=1}^{n} \rho_i} \right\}
\]

It is well known that \( T_{cc} \) is an upper bound on the value of \( B \), and the value of the production schedule based on the \( T_{cc} \) is an upper bound on the value of the optimum. A lower bound on the value of \( B \) is given by \( \max_i \{(1 + \rho_i)s_i\} \), since it is impossible to secure a feasible solution for the ELSP (using the EBP approach) at \( B < \max_i \{(1 + \rho_i)s_i\} \). Since the sssp searches along the \( B \)-axis from the upper bound to the lower bound, it covers the feasible range of \( B \). One may obtain a solution that is 'close' to a global optimum of the unconstrained ELSP (PoT) if one selects the search step length \( \Delta \) 'small enough'.

Let \( K(B^{(r)}) \) denote the set of multipliers secured at the optimal value of the \( TC_{PoT} \) at a particular value of the basic period \( B^{(r)} \).

2.2. Some Insights into the \( TC_{PoT} \) Function

Recall that function \( TC_{PoT}(B) \) denotes as the optimal value of the unconstrained ELSP (PoT) at a given \( B \). The following remarks provide some insights into the \( TC_{PoT} \) function.

**Remark 1.** If one plots the cost expression for product \( i \), i.e.,

\[
TC_{PoT,i}(k_i, B) = \frac{a_i}{k_i B} + \frac{h_i}{2} d_i (1 - \rho_i) k_i B
\]

as a function of \( B \) under PoT policy (i.e., \( k_i = 1, 2, 4, \ldots \)), then the minimum cost function for product \( i \), denoted by \( \overline{TC}_{PoT,i}(B) \), is a piece-wise convex curve where

\[
\overline{TC}_{PoT,i}(B) = \min_{k_i} \{TC_{PoT,i}(k_i, B)\}.
\]

The examples for Remark 1 are shown in Figure 2.

**Remark 2.** For each \( k_i \), one can secure the local minimum for item \( i \), \( TC_{PoT,i}(k_i, B) \), at

\[
B = \lambda_i (k_i) = \frac{1}{k_i} \sqrt{\frac{2a_i}{h_i d_i (1 - \rho_i)}},
\]

with the minimum cost of

\[
\overline{TC}_{PoT,i} = \min_B \{TC_{PoT,i}(B)\} = \sqrt{2a_i h_i d_i (1 - \rho_i)},
\]

which corresponds to the economic production quantity (EPQ).

**Remark 3.** The optimal value \( TC_{PoT}(B) \) is the sum of the minimum cost functions of the \( n \) products, i.e.,

\[
TC_{PoT}(B) = \sum_{i=1}^{n} TC_{PoT,i}(B).
\]
Since the sum of piece-wise convex functions is still a piece-wise convex function, the plot of the $\text{TC}_{\text{p.o.}}(B)$ function is a piece-wise convex curve.

Define $\text{TC}^*_{\text{p.o.}} \equiv \min_B \{ \text{TC}_{\text{p.o.}}(B) \}$; in other words, $\text{TC}^*_{\text{p.o.}}$ is the minimum of the $\text{TC}_{\text{p.o.}}(B)$ function.

**Remark 4.** Observe that $\text{TC}^*_{\text{p.o.}} \geq \sum_i \text{TC}_{\text{p.o.},i}$, since the $\text{TC}_{\text{p.o.}}(B)$ function is the sum of piece-wise convex functions, and it is improbable that all of the $\text{TC}_{\text{p.o.},i}$ coincide at the same value of b.p. (For instance, in Example 6 of [20], $\text{TC}^*_{\text{p.o.}}$ is secured at 21519.112, and $\sum_i \text{TC}_{\text{p.o.},i} = 21218.574$.)

The following result constitutes a cornerstone of our procedure.

**Theorem 1.** If $\text{TC}^*_{\text{p.o.}}$ is secured at $B^*_{\text{p.o.}}$ with the set of multipliers $K^*_{\text{p.o.}}(B^*_{\text{p.o.}})$, then one can secure another minimum solution at $B^*_{\text{p.o.}}/2$ with the set of multipliers $2K^*_{\text{p.o.}}(B^*_{\text{p.o.}})$.

**Proof.** Given that $\text{TC}^*_{\text{p.o.}}$ is secured at $B^*_{\text{p.o.}}$, and $K^*_{\text{p.o.}}(B^*_{\text{p.o.}}) = \{ k_i^* \}$, one has

$$\text{TC}^*_{\text{p.o.}} = \sum_{i=1}^{n} \frac{a_i}{k_i^* B_{\text{p.o.}}} + \frac{h_i}{2} d_i (1 - \rho_i) k_i^* B_{\text{p.o.}}.$$  \hspace{1cm} (11)
It can be observed that

\[ \text{TC}_{P_{oT}} = \sum_{i=1}^{n} \frac{a_i}{(2k_i^*) (B^*_{P_{oT}}/2)} + \frac{k_i^*}{2} d_i (1 - \rho_i) (2k_i^*) \left( \frac{B^*_{P_{oT}}}{2} \right). \]  

Therefore, one can secure another minimum solution at \( B^*_{P_{oT}}/2 \) with the set of multipliers \( 2K^*_{P_{oT}} \).

The example for Remarks 3 and 4 and Theorem 1 is shown in Figure 3.

An immediate corollary of Remarks 1-4 and Theorem 1 is the following.

**Proposition 1.** The \( \text{TC}_{P_{oT}}(B) \) function is a lower bound on the optimal objective function of the ELSP (EBP, PoT).

**Proof.** The \( \text{TC}_{P_{oT}}(B) \) function is secured by relaxing the feasibility requirements of the ELSP (EBP, PoT).

3. ANALYSIS OF THE JUNCTION POINTS

3.1. The Location of 'Junction Points' of the \( \text{TC}_{P_{oT}} \) Function

We now introduce the concept of 'junction points' in the \( \text{TC}_{P_{oT}} \) function. The piece-wise convex curve is a concatenation of convex curves plotted on consecutive intervals of the B-axis. Define as junction points the B values where two neighboring convex curves meet. These junction points play a key role in determining 'which product i' and 'where on the B-axis' to change its multiplier \( k_i \) to \( 2k_i \) in order to secure the optimal values for the ELSP (PoT).

Naively, one can design a search algorithm similar to the sssp (Section 2.1) to determine which product i and where on the B-axis to change \( k_i \) so as to locate the junction points. But this is neither efficient nor accurate, since the step size of the search algorithm determines its performance. One's first impulse is to utilize the derivative of the objective function to provide us with information on how much the objective function will change when we perturb the multiplier \( k_i \) infinitesimally. Actually, the results derived from such an approach are at variance
with observations, and we were led to difference instead of derivative arguments. This is due to the fact that the objective function is changed noninfinitesimally from \( k_i \) to \( 2k_i \). Consequently, we pursue the idea of 'difference changes' instead of derivatives. Given the current set of optimal multipliers \( K = \{k_i\} \), if the algorithm changes the multiplier from \( k_i \) to \( 2k_i \), the difference for product \( i \) is given by

\[
-\frac{a_i}{2k_iB} + \frac{h_i d_i}{2} (1 - \rho_i) k_i B.
\]

One then chooses the product \( i \) which value of difference first reaches zero and becomes negative from that point on, as the search algorithm progresses from \( T_{cc} \) toward smaller values of \( B \). The meaning behind this scheme is that one keeps using the current set of optimal multipliers until the value of the objective function can be improved. Actually, this concept not only provides us with the information on 'which product \( i \)' to modify, but also on 'where on the \( B \)-axis' to replace \( k_i \) by \( 2k_i \). We identify a junction point by \( \delta_i(k_i) \).

Given \( K_{POT}(T_{cc}) \), i.e., the set of optimal multipliers \( \{k_i\} \) secured by the ELSP (POT) model at the value of the b.p. equal to the \( T_{cc} \), the following search procedure is devised to implement this idea.

### 3.2. The 'Incremental Difference' (ID) Search Procedure

1. Let \( B^{(1)} = T_{cc} \) and \( r = 1 \).
2. Put \( r \leftarrow r + 1 \). For each \( k_i \in K(B^{(r-1)}) \), compute \( \delta_i(k_i) \) such that

\[
-\frac{a_i}{2k_iB} + \frac{h_i d_i}{2} (1 - \rho_i) k_i B = 0,
\]

i.e.,

\[
\delta_i(k_i) = \frac{1}{k_i} \sqrt{\frac{a_i}{h_i d_i (1 - \rho_i)}}.
\]

3. Secure the information on 'which product \( i \)' by \( k_i = \arg \max_i \{\delta_i(k_i)\} \), and the information on 'where on the \( B \)-axis' to move by \( \delta = \delta_i(k_i) \).
4. Let \( B^{(r)} = \delta \) and \( K(B^{(r)}) = \{k_i | k_i \neq k_i\} \cup \{2k_i\} \). If \( B^{(r)} > \max_i \{(1 + \rho_i) s_i\} \), then go to Step 2; otherwise, stop.

Theorem 1 motivates our thrust for searching toward smaller values of b.p. and replacing \( k_i \) by \( 2k_i \) in Step 4 of the incremental difference search procedure.

**Remark 5.** For a product \( i \), the equation

\[
\lambda_i(k_i) = \sqrt{2} \delta_i(k_i)
\]

prescribes the relation between a local minimum \( \lambda_i(k_i) \) (defined in equation (8)) and the next junction point \( \delta_i(k_i) \) (defined in equation (13)) below (i.e., smaller than) \( \lambda_i(k_i) \).

We continue to employ the same example to show the implementation of this procedure. Table 1 not only improves the accuracy of the location of the junction points of \( T_{POT} \), but also shows how to change \( k_i \). In this example, the set of optimal multipliers for the problem ELSP (POT) at \( T_{cc} = 25.0284 \) is \( \{1, 2, 2, 8, 4, 2, 1, 1, 1, 1\} \).

The sets of multipliers \( \{k_i\} \) shown in Table 1 are exactly the same as those obtained by the sspp. It is easy to see that the search procedure above is actually looking for the optimal values over the \( B \)-axis. The key improvement of the ID search procedure is that instead of using a small step-size search, the location and the candidate multiplier for change, \( k_i \), can be secured by a closed form calculation.

Next, we discuss some interesting properties of the junction points of the \( T_{POT} \) function, and present a more efficient procedure than the ID search procedure. This improved search procedure not only locates all the junction points, but also provides an easy way to secure the set of optimal multipliers for a given \( B \).
3.3. Some Properties for the Junction Points of the $T_CPO_T$ Function

**Lemma 1.** Suppose that $k_i^{(L)}$ and $k_i^{(R)}$, respectively, are the optimal multipliers of the left-side and right-side convex curves with regard to a junction point in the plot of the $T_CPO_T(B)$ function defined in Remark 1. Then $k_i^{(L)} = 2k_i^{(R)}$.

**Proof.** By equation (13),

$$\delta_i(2^u) < \cdots < \delta_i(2^{m+1}) < \delta_i(2^m) < \cdots < \delta_i(1),$$

where $2^u$ is an upper bound on $k_i$ (derived in the Appendix).

Denote as $k_i^*(B)$ the optimal multiplier for $T_CPO_T(B)$ at a given $B$. Because of inequality (15) and the convexity of $T_CPO_T(B)$, one may assert that

$$k_i^*(B) = \begin{cases} 1, & \text{if } B \in [\delta_i(1), \infty), \\ 2^{m+1}, & \text{if } B \in [\delta_i(2^{m+1}), \delta_i(2^m)), \text{ for } m = 0, 1, \ldots, u_i. \end{cases}$$

Equation (16) states exactly that $k_i^{(L)} = 2k_i^{(R)}$.

**Proposition 2.** All the junction points for the piece-wise convex curve of the minimum cost expression of product $i$, i.e., $T_CPO_T(B)$ in equation (7), will be inherited by the piece-wise convex curve of $T_CPO_T(B)$. In other words, if a junction point $w$ shows on one piece-wise convex curve $T_CPO_T(B)$, $w$ must also show on the piece-wise convex curve of the $T_CPO_T(B)$ function as a junction point.

**Proof.** Recall that function $T_CPO_T$ is a separable function where $T_CPO_T(B) = \sum_{j=1}^{m} T_CPO_T_j(B)$. Assume that $w$ is not a junction point for the minimum cost curves of all other $(n - 1)$ products.
Then, there must exist \( \varepsilon > 0 \) such that

1. the curve for \( \sum_{j \neq i} TC_{p_{\alpha},j}(B) \) is convex in the interval of \([w - \varepsilon, w + \varepsilon]\) since each one of \( TC_{p_{\alpha},j}(B) \) where \( j \neq i \) is convex in \([w - \varepsilon, w + \varepsilon]\), and

2. \( TC_{p_{\alpha},i}(B) \) is convex in the intervals of \([w - \varepsilon, w] \) and \([w, w + \varepsilon]\).

Since \( TC_{p_{\alpha}}(B) = TC_{p_{\alpha},i}(B) + \sum_{j \neq i} TC_{p_{\alpha},j}(B) \), one may observe that \( TC_{p_{\alpha}}(B) \) is still convex in the intervals \([w - \varepsilon, w] \) and \([w, w + \varepsilon]\). Therefore, \( w \) becomes a junction point in the curve of \( TC_{p_{\alpha}}(B) \). (This proof also verifies the assertion in Remark 3, viz., that the \( TC_{p_{\alpha}} \) function is a piece-wise convex function since it is a sum of \( n \) piece-wise convex functions.)

The following theorem is an immediate result of Lemma 1 and Proposition 2.

**Theorem 2.** Suppose that \( K^{'(L)} \) and \( K^{'(R)} \), respectively, are the set of optimal multipliers for the left-side and right-side convex curves with regard to a junction point in the plot of the \( TC_{p_{\alpha}}(B) \) function. Then there is one and only one product \( i \) such that \( k_{i}^{(L)} = 2k_{i}^{(R)} \).

### 3.4. An Improved Procedure to Locate All the Junction Points

The above results permit us to present a more efficient procedure than the ID search procedure to locate all the junction points of \( TC_{p_{\alpha}}(B) \). Recall that, in the ID search procedure, the search is conducted in a fashion of ‘hopping’ from a larger basic period to a smaller one. It requires \( O(n) \) comparisons to secure the information on ‘where on the \( B \)-axis’, and ‘which product \( i \)’ should change its \( k_{i} \).

Proposition 2 leads to an improved procedure to locate all the junction points of the \( TC_{p_{\alpha}}(B) \) function without searching along the \( B \)-axis. In fact, one can find all the junction points of the \( TC_{p_{\alpha}}(B) \) function by plugging \( k_{i} \) into equation (13) for \( k_{i} = 1, 2, \ldots, 2^{w} \) (by equation (21) in the Appendix). An improved procedure, namely, the junction point (JP) locating procedure is presented as follows.

**The JP Locating Procedure.**

for \( i = 1, \ldots, n \)

Compute upper bound \( v_{i} \) by equation (21).

Set \( found = 0 \) and \( m = 0 \).

while \( found = 0 \)

\( k_{i} = 2^{m} \).

Compute \( \delta_{i}(k_{i}) = \frac{1}{k_{i}} \sqrt{\frac{\alpha_{i}}{\kappa_{i}d_{i}(1 - \rho_{i})}} \) (refer to equation (13) for details).

\( m \leftarrow m + 1 \).

if \( m > v_{i} \), then \( found = 1 \).

endwhile

end for

Let \( \nu_{\text{max}} \Delta \sum_{i} \{ v_{i} + 1 \} \). We note that the total number of junction points found in the JP locating procedure is less than \( \nu_{\text{max}} \), i.e., \( \sum_{i} \{ v_{i} + 1 \} \leq \nu_{\text{max}} \). Therefore, the complexity of the JP locating procedure is bounded by \( O(\nu_{\text{max}}) \).

**Example 1.** We continue with the same example to show how to efficiently establish a search plan. By the JP locating procedure, one can calculate the junction points by substituting \( k_{i} = 1, 2, \ldots, 32 \) as summarized in Table 2. If one sorts all the junction points of the \( n \) products, and starts the search from \( \text{Tcc} \), one should change \( k_{8} = 1 \) to \( k_{8} = 2 \) at \( B = 23.3285 \), which is the next junction point in the sorted sequence. The search continues and changes \( k_{3} = 2 \) to \( k_{3} = 4 \) at \( B = 21.1516 \), and so on. One may refer to the sorted sequence of all the junction points in Table 1. Note that constructing this sorted sequence actually establishes an efficient search plan using all the junction points. This sorted sequence will be used in the proposed global optimum search algorithm.
Table 2. The junction points for the TC_{PoT} function in the example.

<table>
<thead>
<tr>
<th>Product</th>
<th>32</th>
<th>16</th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.6310</td>
<td>1.2619</td>
<td>2.5238</td>
<td>5.0476</td>
<td>10.0953</td>
<td><strong>20.1905</strong></td>
</tr>
<tr>
<td>2</td>
<td>1.0497</td>
<td>2.0994</td>
<td>4.1988</td>
<td>8.3976</td>
<td><strong>16.7951</strong></td>
<td>33.5902</td>
</tr>
<tr>
<td>3</td>
<td>1.3220</td>
<td>2.6439</td>
<td>5.2879</td>
<td>10.5758</td>
<td><strong>21.1516</strong></td>
<td>42.3032</td>
</tr>
<tr>
<td>4</td>
<td>3.3487</td>
<td>6.6975</td>
<td><strong>13.3950</strong></td>
<td>26.7900</td>
<td>53.5800</td>
<td>107.1599</td>
</tr>
<tr>
<td>5</td>
<td>1.7325</td>
<td>3.4650</td>
<td>6.9300</td>
<td><strong>13.8600</strong></td>
<td>27.7199</td>
<td>55.4398</td>
</tr>
<tr>
<td>6</td>
<td>1.2650</td>
<td>2.5299</td>
<td>5.0598</td>
<td>10.1196</td>
<td><strong>20.2392</strong></td>
<td>40.4784</td>
</tr>
<tr>
<td>7</td>
<td>0.3948</td>
<td>0.7897</td>
<td>1.5794</td>
<td>3.1587</td>
<td><strong>6.3175</strong></td>
<td><strong>12.6349</strong></td>
</tr>
<tr>
<td>8</td>
<td>0.7290</td>
<td>1.4580</td>
<td>2.9161</td>
<td>5.8321</td>
<td>11.6642</td>
<td><strong>23.3285</strong></td>
</tr>
<tr>
<td>9</td>
<td>0.5046</td>
<td>1.0902</td>
<td>2.0184</td>
<td>4.0369</td>
<td>8.0737</td>
<td><strong>16.1475</strong></td>
</tr>
<tr>
<td>10</td>
<td>0.5617</td>
<td>1.1235</td>
<td>2.2469</td>
<td>4.4939</td>
<td>8.9878</td>
<td><strong>17.9755</strong></td>
</tr>
</tbody>
</table>

3.5. An Alternative Way to Secure the Set of Optimal Multipliers

Another by-product of Proposition 2 is an easier way to secure $K_{PoT}(B)$ at any $B$. In general, for any given $B$, one can secure each $k_i \in K_{PoT}(B)$ by

$$
 k_i(B) = \begin{cases} 
 1, & B > \frac{a_i}{h_i d_i (1 - \rho_i)} \\
 2^m, & \frac{1}{2^{m-1}} \sqrt{\frac{a_i}{h_i d_i (1 - \rho_i)}} \geq B > \frac{1}{2^m} \sqrt{\frac{a_i}{h_i d_i (1 - \rho_i)}} 
\end{cases}
$$

(17)

Therefore, one can secure $K_{PoT}(B)$ by the K-PoT search procedure as follows.

THE K-PoT SEARCH PROCEDURE.
for $i = 1, \ldots , n$
Set $found = 0$ and $m = 0$.
if $B > \sqrt{(a_i/h_i d_i (1 - \rho_i))}$, then $found = 1$.
while $found = 0$
  $m \leftarrow m + 1$.
  if $B > (1/2^m) \sqrt{(a_i/h_i d_i (1 - \rho_i))}$, then $found = 1$.
endwhile
Set $k_i(B) = 2^m$.
endfor

One can apply the K-PoT search procedure to Example 1 and secure $K_{PoT}(T_{cc}) = \{1, 2, 2, 8, 4, 2, 1, 1, 1, 1\}$, where $k_i \in K_{PoT}(T_{cc})$ are indicated by the bold numbers in Table 2.

4. A GLOBAL OPTIMUM SEARCH ALGORITHM

The search algorithm to secure the global optimal solution depends on being able to locate the local minima of the TC_{PoT} function, which, in turn, depends on the junction points of the TC_{PoT} function. Recall that each junction point $\delta_i(k_i)$ provides the information that one should change the optimal multiplier of item $i$ from $k_i$ to $2k_i$ at $\delta_i(k_i)$ to secure the optimal value for the TC_{PoT}(i)(B) function. Therefore, given all the junction points $\{\delta_i(k_i) \mid i = 1, \ldots , n\}$ of the TC_{PoT} function (secured by the JP locating procedure), we generate an array of (sorted) ordered pairs in which the first element is the location (i.e., value of $B$) of the junction point and the second element is the identity of the product $i$. The list is sorted on the location (in descending order), which are now denoted by $\{w_j\}$, where $w_{j+1} < w_j$, $j = 1, 2, \ldots$. Another sequence of product indices, denoted by $\{\iota_j(w_j)\}$, is generated accordingly to correspond to the $w_i$. We now have in hand an array of (sorted) ordered pairs $\{(w_j, \iota_j(w_j))\}$. We refer to this procedure as the JP sorting procedure.

Since the JP sorting procedure sorts the junction points, of which there are at most $nv_{max}$, it is clear that the complexity of the JP sorting procedure is bounded by $O(nv_{max} \log n v_{max})$. 

4.1. Check of Local Optimality and the Termination Condition

First, we address the issue of local optimality. Suppose that \( w_{j+1}, w_j \), with \( w_{j+1} < w_j \), are two neighboring junction points of \( TC_{PoT} \), and that \( \{k_i\} \) is the set of optimal multipliers for the interval of \( (w_{j+1}, w_j) \). Denote by \( \hat{B}_j \) the local minimum in this interval. By the convexity of the \( TC_{PoT} \) function, the local minimum is either inside the interval or at \( w_{j+1} \). To achieve this determination, one secures the derivative of the \( TC_{PoT} \) function \( w.r.t. B \) (for the given \( \{k_i\} \)) and equates it to zero,

\[
\frac{dTC_{PoT}}{dB} = \frac{d}{dB} \left[ \sum_{i=1}^{n} \left( \frac{a_i}{k_i B} + \frac{h_i d_i}{2} (1 - \rho_i) k_i B \right) \right] = \sum_{i=1}^{n} \left[ -\frac{a_i}{k_i B^2} + \frac{h_i d_i}{2} (1 - \rho_i) k_i \right] = 0
\]

(18)

If \( \hat{B}_j \in (w_{j+1}, w_j) \), then \( \hat{B}_j \) is a local minimum of the \( TC_{PoT} \) function. Else, \( \hat{B}_j \) must be at the extreme point of the interval, \( w_{j+1}, \hat{B}_j = w_{j+1} \).

We now address the issue of the termination condition of the global optimal search. The following theorem asserts an interesting result which is used to determine the termination condition.

**Theorem 3.** Suppose that the sequences \( \{\hat{B}_j\} \) are the local minima of the \( TC_{PoT} \) with \( \hat{B}_{j+1} < \hat{B}_j \) for all \( j \). The function of the optimal solution for the unconstrained ELSP (PoT) (i.e., the \( TC_{PoT} \) function) in the interval of \( \hat{B}_1/2, \hat{B}_1 \) repeats in the intervals \( [\hat{B}_{1/2}^{p+1}, \hat{B}_{1/2}^{p}] \) for all \( p = 1, 2, \ldots \).

**Proof.** The proof is similar to that for Theorem 1. 

Theorem 3 indicates that when searching from higher to lower values of \( B \), one may eliminate the search below \( \hat{B}_1/2 \), since it is impossible to secure a better solution due to the optimality of \( \hat{B}_1/2 \). This reduces the total search effort.

We are now ready to enunciate the global optimum search algorithm. It uses the array of the (sorted) ordered pairs \( \{(w_j, \mu_j(w_j))\} \) as the backbone of its search scheme. By definition, \( w_1 = T_{cc} \) is the largest junction point of the \( TC_{PoT} \) function. The algorithm searches from \( w_1 \), in descending order, toward lower values of \( w \) in the sequence \( \{w_j\} \). Recall that \( w_j \) is the \( j \)th largest junction point of the \( TC_{PoT} \) function where one should replace \( k_{i_j}(w_j) \) with \( 2k_{i_j}(w_j) \). The search scheme starts with \( K_{PoT}(w_1 + \varepsilon) = \{1, \ldots, 1\} \), where \( \varepsilon \) is a small positive real number and \( \{1, \ldots, 1\} \) is a set of \( n \) elements of 1s. By equation (13) and Lemma 1, \( K_{PoT}(w_1 + \varepsilon) = \{1, \ldots, 1\} \) is the set of optimal multipliers for all \( B \in (w_1, \infty) \). Since \( K_{PoT}(w_1 + \varepsilon) = \{1, \ldots, 1\} \), we note that \( w(K_{PoT}(w_1 + \varepsilon)) = T_{cc} \), by the common cycle approach. Let \( K(w_1) = (K_{PoT}(w_1 + \varepsilon) - \{k_{i_j}(w_1)\}) \cup \{2k_{i_j}(w_1)\} \), in which one replaces \( k_{i_j}(w_1) \) with \( 2k_{i_j}(w_1) \) at \( w_1 \) to secure the optimal value for the \( TC_{PoT} \) function for \( B \leq w_1 \). Denote by \( K(w_j) \) the set of optimal multipliers in the interval \( (w_{j+1}, w_j) \). By the same token, one can secure \( K(w_j) \) by

\[
K(w_j) \triangleq (K(w_{j-1}) - \{k_{i_j}(w_j)\}) \cup \{2k_{i_j}(w_j)\}, \quad j = 2, \ldots
\]

Also, one should employ condition (18) to check if the local minimum for \( K(w_j) \) exists in the interval \( (w_{j+1}, w_j) \). One proceeds with this search scheme until the termination conditions in Theorem 3 are satisfied. A global optimal solution can then be secured by choosing the minimum among all the local minima.
We label as \( l \) the index for the local optima of the \( \text{TCPoT} \). Hence, \( \hat{B}_l \) is the \( l \)-th local optimal solution secured in the search process of the global optimum search algorithm. The step-by-step procedure is presented as follows.

1. Secure all the junction points of \( \text{TCPoT} \) function by the JP locating procedure.
2. Generate the array of the (sorted) ordered pairs, i.e., \( \{ (w_j, t_j(w_j)) \} \), by the JP sorting procedure.
3. Set \( \text{TCPoT}(w_1 + \varepsilon) = \{ 1, \ldots, 1 \} \). Employ the local optimality checking condition (18) to check if
   \[ \hat{B}(\text{TCPoT}(w_1 + \varepsilon)) = T_{cc} \in (w_1, \infty). \]
   If it does, let \( l = 1, \hat{B}_1 = T_{cc}, \) and compute \( \text{TCPoT}(\text{TCPoT}(w_1 + \varepsilon), T_{cc}) \); otherwise, let \( l = 0, j = 1, \) and \( K(w_1) = \{ \text{TCPoT}(w_1 + \varepsilon) - \{ k_{t_1(w_1)} \} \} \cup \{ 2k_{t_1(w_1)} \} \); go to Step 4.
4. Employ the local optimality checking condition to check if \( \hat{B}(\hat{K}(w_j)) \in (w_{j+1}, w_j] \). If it does, let \( l = l + 1, \hat{B}_l = \hat{B}(\hat{K}(w_j)), \) and compute \( \text{TCPoT}(\hat{K}(w_j), \hat{B}_l) \); otherwise, go to Step 5.
5. Let \( j = j + 1 \). If \( w_j < \hat{B}_j/2, \) go to Step 6; otherwise, secure \( \hat{K}(w_j) \) by \( \hat{K}(w_j) = \{ \hat{K}(w_{j-1}) - \{ k_{t_1(w_{j-1})} \} \} \cup \{ 2k_{v_1(w_{j-1})} \} \) and go to Step 4.
6. Secure \( (K^*, B^*), \) i.e., the global optimal solution by
   \[ (K^*_{\text{TCPoT}}, B^*_{\text{TCPoT}}) = \arg \min \{ \text{TCPoT}(K(w_j), w_j^*) \}, \]
   and stop.

Recall that the complexity of the JP locating procedure and the JP sorting procedure is bounded by \( O(nv_{max}) \) and \( O(nv_{max} \log nv_{max}) \), respectively. Also, the number of iterations in the loop of Steps 4 and 5 is less than \( \sum_i (v_i + 1) \), and is surely less than \( nv_{max} \). Therefore, the complexity of the global optimum search algorithm is bounded by \( O(nv_{max} \log nv_{max}) \).

### 5. A NUMERICAL EXAMPLE

In this section, we employ Example 6 in [20] to demonstrate the proposed global optimum search algorithm.

1. Use the JP locating procedure to locate all the junction points of \( \text{TCPoT} \) and their corresponding product indices (for replacing \( k_i \) by \( 2k_i \)) as shown in Table 2.
2. Generate the array of the (sorted) ordered pairs, i.e., \( \{ (w_j, t_j(w_j)) \} \), by the JP sorting procedure. We note that \( w_1 = 107.1599 \) and \( t_1(w_1) = 4 \).
3. Set \( w_1 + \varepsilon = 108 \) and \( \text{TCPoT}(w_1 + \varepsilon) = \{ 1, \ldots, 1 \} \) where \( \varepsilon = 0.8401. \)
   (a) Employ the Local Optimality Checking condition to check if \( \hat{B}(\text{TCPoT}(w_1 + \varepsilon)) \in (w_1, \infty) \) : secure \( T_{cc} = \hat{B}(\text{TCPoT}(w_1 + \varepsilon)) = 25.0284, \) and therefore, \( \hat{B}(\text{TCPoT}(w_1 + \varepsilon)) \notin (w_1, \infty). \)
   (b) Set \( K(w_1) = \{ 1, \ldots, 1 - \{ k_{t_1(w_1)} = k_4 = 1 \} \} \cup \{ 2k_{t_1(w_1)} = 2k_4 = 2 \}, \) i.e., \( K(w_1) = \{ 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1 \}. \)
4. Move to \( w_2 = 55.4398 \) where \( t_2(w_2) = 5 \). Employ the local optimality checking condition (18) to check if \( \hat{B}(\hat{K}(w_1)) \) exists in the interval \( (w_2, w_1) \) : secure \( \hat{B}(\hat{K}(w_1)) = 24.8609 \notin (w_2, w_1) = (55.4398, 107.1599). \)
5. Set \( K(w_2) = \{ \hat{K}(w_1) - \{ k_5 \} \} \cup \{ 2k_5 \}, \) i.e., \( K(w_2) = \{ 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1 \}. \)

The algorithm continues its search until it reaches \( w_{10} = 21.1516 \) (with \( k_3 = 2 \) replaced by \( k_3 = 4 \)), and secures the first local minimum. The algorithm secures \( \hat{K}(w_9) = \{ 1, 2, 2, 3, 2, 1 \} \) at \( w_9 = 23.3285, \) and the local optimum for \( \hat{K}(w_9) \) is secured at \( \hat{B}_1 = 21.9490. \) We note that \( \hat{B}_1 = 21.9490 \in (w_{10}, w_9] = (21.1516, 23.3285], \) and one secures \( \text{TC} (\hat{K}(w_9), \hat{B}_1) = 21622.85. \) The algorithm proceeds with \( \hat{K}(w_9) \) and the search process is summarized in Table 3. When the algorithm reaches \( w_{20} = 10.5758 \), the termination condition in Theorem 3 is satisfied since \( w_{20} (= 10.5758) < \frac{\hat{B}_1}{2} (= 10.9745). \)
Table 3. The search process of the global optimum search algorithm.

<table>
<thead>
<tr>
<th>$w_j$</th>
<th>$k_1$</th>
<th>$k_2$</th>
<th>$k_3$</th>
<th>$k_4$</th>
<th>$k_5$</th>
<th>$k_6$</th>
<th>$k_7$</th>
<th>$k_8$</th>
<th>$k_9$</th>
<th>$k_{10}$</th>
<th>$\omega_j^*$</th>
<th>$TC_{PoT}(\omega_j^*)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_1$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>$w_2$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>$w_3$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>$w_4$</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>$w_5$</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>$w_6$</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>$w_7$</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>$w_8$</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>$w_9$</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>$w_{10}$</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>$w_{11}$</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>$w_{12}$</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>$w_{13}$</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>$w_{14}$</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>$w_{15}$</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>$w_{16}$</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>$w_{17}$</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>$w_{18}$</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>$w_{19}$</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>$w_{20}$</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
</tbody>
</table>

Then, we may secure a global optimal solution by $(K_{PoT}^*, B_{PoT}^*) = \arg\min \{TC_{PoT}(K(w_j), B_i)\}$.

In this example, a global optimal solution is secured at $B_{PoT}^* = 14.9170$, and the set of optimal multipliers is $K_{PoT}^* = \{1, 2, 2, 4, 2, 1, 2, 1, 1\}$. The optimal value for the $TC_{PoT}$ function is $TC_{PoT}(K_{PoT}^*, B_{PoT}^*) = 21519.11$.

6. CONCLUDING REMARKS

This study presents an analysis on the economic lot scheduling problem (ELSP) without capacity constraints under PoT policy. Theorem 2 asserts that the optimality structure of the unconstrained ELSP (PoT) (i.e., the $TC_{PoT}$) is a piece-wise convex function of $B$, and the set of optimal multipliers keeps the same between two consecutive junction points. Also, by making use of the properties of the junction points and the termination condition, we propose an efficient search algorithm for securing a global optimum for the $TC_{PoT}$ with its complexity bounded by $O(nv_{max})$.

The theoretical results in this paper provide some insights into the optimality structure of the conventional ELSP under PoT policy. The global optimum for the ELSP (PoT), secured by the proposed global optimum search algorithm, may serve as an improved bound to verify the quality of other search heuristics for the ELSP (EBP,PoT).

APPENDIX

AN UPPER BOUND ON $k_i$

A simple upper bound on $k_i$ can be derived from an upper bound on the objective function of problem ELSP (EBP,PoT) and the independent solution (which is denoted by $IS$, and it is expression (9)). The optimal solution of the common cycle approach $TC^{cc}$ is a well-known upper bound on the objective function of the problem ELSP (EBP,PoT) where

$$
TC^{cc} = \sum_{j=1}^{n} \frac{a_j}{T} + \frac{h_j}{2} d_j (1 - \rho_j) T
$$

(19)
and

\[ T_{cc} = \max \left\{ \sqrt{\frac{2 \sum_{j=1}^{n} a_j}{\sum_{j=1}^{n} h_j d_j (1 - \rho_j)}}, \frac{\sum_{j=1}^{n} s_j}{1 - \sum_{j=1}^{n} \rho_j} \right\}. \]  
(20)

Let

\[ IS(n - \{i\}) = \sum_{j=1, j \neq i}^{n} c_j^i = \sum_{j=1, j \neq i}^{n} \sqrt{2a_i h_j d_j (1 - \rho_j)}. \]

Then, an upper bound on the average cost of product \( i \) is obtained by \( TC_{cc} - IS(n - \{i\}) \), and we have \( c_i = \frac{a_i}{T_i} + \frac{(h_i/2) d_i (1 - \rho_i)}{T_i} \leq TC_{cc} - IS(n - \{i\}) \) where \( T_i \) is the cycle length between production runs of product \( i \). Thus, for a given \( B \), an upper bound on \( k_i \) can be expressed by \( 2^{v_i} \), where

\[ v_i = \left\lfloor \log_2 \left( \frac{(TC_{cc} - IS(n - \{i\})) + \sqrt{(TC_{cc} - IS(n - \{i\}))^2 - 2a_i h_i d_i (1 - \rho_i)}}{h_i d_i (1 - \rho_i) B} \right) \right\rfloor. \]
(21)

REFERENCES