COMPLEXITY ANALYSIS OF AN ASSIGNMENT PROBLEM
WITH CONTROLLABLE ASSIGNMENT COSTS AND ITS
APPLICATIONS IN SCHEDULING*

Liron Yedidsion†
Faculty of Industrial Engineering and Management
Technion - Israel Institute of Technology, Haifa, Israel

Dvir Shabtay and Moshe Kaspi‡
Department of Industrial Engineering and Management
Ben-Gurion University of the Negev Beer-Sheva, Israel

Abstract
We extend the classical linear assignment problem to the case where the cost of assigning agent \(j\) to
task \(i\) is a multiplication of task \(i\)'s cost parameter by a cost function of agent \(j\). The cost function of
agent \(j\) is a linear function of the amount of resource allocated to the agent. A solution for our assignment
problem is defined by the assignment of agents to tasks and by a resource allocation to each agent. The
quality of a solution is measured by two criteria. The first criterion is the total assignment cost and the
second is the total weighted resource consumption. We address these criteria via four different problem
variations. We prove that our assignment problem is \(\mathcal{NP}\)-hard for three of the four variations even if
all the resource consumption weights are equal. However, and somewhat surprisingly, we find that the
fourth variation is solvable in polynomial time. In addition, we find that our assignment problem is
equivalent to a large set of important scheduling problems whose complexity has heretofore been an open
question for three of the four variations.

Keywords: Assignment problem, scheduling, controllable processing times, complexity, resource alloca-
tion, bicriteria optimization.

1 Introduction
This paper presents and analyzes a new extension of the classical linear assignment problem (LAP) which
has some very important applications in deterministic scheduling theory. Assignment problems deal with
the question of how to assign a set of \(n\) agents to a set of \(n\) tasks such that each task is performed only once
and each agent is assigned to a single task such that a specific predefined objective will be minimized. An

*This research was supported by THE ISRAEL SCIENCE FOUNDATION (grant No. 633/08). Partial support by the Paul
Ivanier Center for Robotics and Production Management, Ben-Gurion University of the Negev is also gratefully acknowledged.
†e-mails: lirony@ie.technion.ac.il
‡e-mails: dvirs@bgu.ac.il and moshe@bgu.ac.il
assignment is simply a permutation $\phi$ which maps each element $i$ of $\{1, 2, \ldots, n\}$ onto a unique element $\phi(i)$ of $\{1, 2, \ldots, n\}$. It can be represented by either a permutation vector

$$\phi = (\phi(1), \phi(2), \ldots, \phi(m)),$$

where $j = \phi(i)$ means that agent $j$ is assigned to task $i$ in permutation $\phi$, or by a permutation matrix $X = (x_{ij})$ with $x_{ij} = 1$ if $j = \phi(i)$ and $x_{ij} = 0$ if $j \neq \phi(i)$ [5]. The set of all $n!$ possible assignments, $\Phi$, are given by the following set of assignment constraints:

$$\sum_{i=1}^{n} x_{ij} = 1, \text{ for all } j = 1, \ldots, n \quad (1)$$

$$\sum_{j=1}^{n} x_{ij} = 1, \text{ for all } i = 1, \ldots, n \quad (2)$$

$$x_{ij} \in (0, 1) \text{ for all } i, j = 1, \ldots, n \quad (3)$$

The first set of constraints assures that each agent will be assigned only to a single task and the second that each task will be assigned only once.

In the classical LAP, the cost of assigning agent $i$ to task $j$ is given by $c_{ij}$ for any $i, j = 1, \ldots, n$. The objective is to find a permutation, $\phi^*$, which minimizes the total assignment cost that is given by

$$c(\phi) = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} = \sum_{i=1}^{n} c_{i\phi(i)}.$$  \hspace{1cm} (4)

over the set of all $n!$ possible permutations given by the assignment constraints (1)-(3). It is well known that the LAP problem can be solved in $O(n^3)$ time (see, e.g., Papadimitriou and Steiglitz [41]).

As stated above, in the classical LAP the cost of assigning agent $j$ to task $i$ is a fixed parameter, $c_{ij}$. However, in our extension of this problem, we assume that the cost of assigning agent $j$ to task $i$ is given by

$$c_{ij} = \omega_i \times p_j(u_j),$$  \hspace{1cm} (5)

where $\omega_i$ is task $i$'s assignment cost parameter and $p_j(u_j)$ is the assignment cost function of agent $j$. The assignment cost function is given by the following linear model for $j = 1, \ldots, n$:

$$p_j(u_j) = \overline{p}_j - b_j u_j, \quad 0 \leq u_j \leq \overline{u}_j \leq \overline{p}_j / b_j,$$  \hspace{1cm} (6)

where $\overline{p}_j$ is the non-compressed assignment cost for agent $j$; $u_j$ is a decision variable that represents the amount of nonrenewable resource allocated to agent $j$; $\overline{u}_j$ is the upper bound on the amount of resource that can be allocated to agent $j$; and $b_j$ is the positive cost compression rate of agent $j$. We study two cases, one in which the resource is continuous and the other in which the resource is used in discrete quantities. In such a framework, a solution is defined by a permutation $\phi$ and by a resource allocation vector $u = (u_1, u_2, \ldots, u_n)$.

The quality of a solution is measured by two different criteria. The first is the total assignment cost
which is given by
\[ c(A) = \sum_{i=1}^{n} \sum_{j=1}^{n} \omega_i \times (p_j - b_j u_j) \times x_{ij} = \sum_{i=1}^{n} \omega_i \times (p_{\phi(i)} - b_{\phi(i)} u_{\phi(i)}) = \sum_{i=1}^{n} c_{i, \phi(i)}(u_{\phi(i)}), \] (7)

where \( A = (\phi, u) \). The second criterion is the total resource consumption cost, given by
\[ U(A) = \sum_{j=1}^{n} v_j u_j, \] (8)

where \( v_j \) is the cost of assigning one unit of resource to agent \( j \). Both criteria have to be minimized. We refer to our assignment problem as resource dependent assignment problem or RDAP in short. Since RDAP is essentially a problem with two criteria which are given by eqs. (7) and (8), the following four different problem variations can arise:

- The first, which we denote by RDAP1, is to minimize the total integrated cost, \( c(A) + U(A) \), as defined by:
\[ \sum_{i=1}^{n} \sum_{j=1}^{n} \omega_i \times (p_j - b_j u_j) \times x_{ij} + \sum_{j=1}^{n} v_j u_j = \sum_{i=1}^{n} \omega_i \times (p_{\phi(i)} - b_{\phi(i)} u_{\phi(i)}) + \sum_{i=1}^{n} v_{\phi(i)} u_{\phi(i)} \] (9)

subject to the assignment constraints (1)-(3) and
\[ 0 \leq u_j \leq u^*_j \text{ for } j = 1, \ldots, n. \] (10)

- The second, which we denote by RDAP2, is to minimize \( c(A) \) given by eq. (7) subject to the assignment constraints (1)-(3), eq. (10) and
\[ U(A) = \sum_{j=1}^{n} v_j u_j \leq U_v, \] (11)

where \( U_v \) is an upper bound on the total resource consumption cost.

- The third, which we denote by RDAP3, is to minimize eq. (8) subject to the assignment constraints (1)-(3), eq. (10) and
\[ c(A) \leq K, \] (12)

where \( K \) is a given upper bound on the total assignment cost.

- The last, which we denote by RDAP4, is to identify the set of Pareto-optimal solutions for \( (c(A), U(A)) \), where an assignment \( A \) is called Pareto-optimal if there does not exist another assignment \( A' \) such that \( c(A') \leq c(A) \) and \( U(A') \leq U(A) \), with at least one of these inequalities being strict. It should be noted that solving RDAP4 requires either the solution of RDAP2 for any possible \( U_v \) value or the solution of RDAP3 for any possible \( K \) value.

The RDAP has many real life applications. For example, substantiating a project of any kind involves many decisions that will eventually effect its profitability. Two of the most important decisions are assigning managerial personnel to the different departments\(, \) positions and allocating a budget to the various departments. Those very important decisions are usually made separately and independently of each other producing a sub-optimal solution. Modeling the contribution of each person when assigned to a given department as a function of the budget allocated to this department and solving the resulting RDAP might be
Another direct application for our model is in solving assignment problems where we can crash the
duration of each task in a similar way to what is being done in the context of project and job scheduling
problems (see [31] and [49] for surveys). Each agent is actually a team of people that has a standard (basic)
processing time of $p_j$ to perform each task. However, there is a possibility to allocate additional resources
(people) to each team and crash their basic duration of processing time, such that, given that additional
$u_j$ people are allocated to team $j$, the standard processing time of team $j$ is given by the linear model (6).
Moreover, each task $i$ has its own relative difficulty factor, $\omega_i$ for $i = 1,...,n$, which means that the actual
time to perform task $i$ by agent $j$ is given by eq. (5) and the total processing time to perform all tasks (as
a function of the assignment and resource allocation decisions) is given by eq. (7). The marginal resource
allocation cost to team $j$ is $v_j$ and thus the total resource allocation cost is given by eq. (8).

Additional and very important set of applications of the RDAP
arises form a large set of equivalent
scheduling problems where the sequencing problem reduces to an assignment problem and the processing
time of each job is a linear function of the resource allocated to its performance.

The paper proceeds as follows. In Section 2 we prove that RDAP1 is solvable in polynomial time while
RDAP2-RDAP4 are $\mathcal{NP}$-hard for any set of $\omega_i$ parameters satisfying $\omega_i \neq \omega_j$ for any $i \neq j$. A special case
of RDAP1-RDAP4 that requires a reduced computational effort is presented in Section 3. In Section 4 we
show that there are some important scheduling problems, which are equivalent to RDAP with a specific set
of $\omega_i$ parameters associated with each one of them. Section 5 includes a summary of the obtained results
and discussion and future research section concludes the paper.

2 Complexity Analysis of RDAP1-RDAP4

Hereafter, without loss of generality, we assume that $\omega_1 \geq \omega_2 \geq ... \geq \omega_n \geq 0$. The following corollary is
straightforward from the fact that the decision versions of RDAP2 and RDAP3 are identical (they both ask
if there is a feasible assignment with $\sum_{i=1}^n v_{\phi(i)}u_{\phi(i)} \leq U_v$ and $\sum_{i=1}^n \omega_i \times (p_{\phi(i)} - b_{\phi(i)}u_{\phi(i)}) \leq K$).

**Corollary 1** If the RDAP2 problem is solvable in polynomial time, then the RDAP3 is solvable in polynomial
time as well and vice versa. In addition, if RDAP2 is $\mathcal{NP}$-hard, then RDAP3 (and also RDAP4) is $\mathcal{NP}$-hard
as well and vice versa.

The following two lemmas are applicable for both continuous and discrete types of resources.

**Lemma 1** For any given $u = (u_1, u_2, ..., u_n)$ vector which fixes the agent penalty function, the optimal assignment, $\phi^*$ for all problem types can be obtained in $O(n \log n)$ time by ordering the $p = (p_1(u_1), p_2(u_2), ..., p_n(u_n))$
vector in a non-decreasing order. The optimal assignment is then attained by matching the agent in the $j^{th}$
position in this vector to task $j$.

**Proof.** A given $u = (u_1, u_2, ..., u_n)$ vector fixes the resource allocation cost value $U = \sum_{j=1}^n v_ju_j$ and
the assignment cost function, $p_j(u_j) = p_j - b_ju_j$ for $j = 1, ..., n$. Thus, all that remains to be done is to
assign agents to tasks so as to minimize eq. (7) with a fixed set of processing times. The lemma then follows
from a well-known result in linear algebra regarding the minimization of a scalar product of two vectors (see
Hardy *et al.* [17]).
Lemma 2  For a given permutation, $\phi$, the problem of finding the optimal resource allocation for the RDAP2 problem reduces to a knapsack problem.

Proof.  For a given permutation, $\phi$, the RDAP2 problem is reduced to minimize $\sum_{i=1}^{n} \omega_i \left( p_{\phi(i)} - b_{\phi(i)} u_{\phi(i)} \right)$, or equivalently to maximize $\sum_{i=1}^{n} \omega_i b_{\phi(i)} u_{\phi(i)} = \sum_{j=1}^{n} \theta_j u_j$ where $j = \phi(i)$ and $\theta_j = \omega_j b_j$ subject to $\sum_{j=1}^{n} v_j u_j \leq U_v$. This problem is known as the knapsack problem. \hfill \blacksquare

The following two Corollaries are straightforward from well-known results about the continuous and the discrete knapsack problems, respectively.

Corollary 2  For a given permutation, $\phi$, the problem of finding the optimal resource allocation for the RDAP2 problem with a continuous type of resource is reduced to a continuous knapsack problem and thus can be solved in $O(n \log n)$ time by ordering the agents in a non-increasing $\theta_j/v_j$ order and packing them greedily in this order until $\sum_{j=1}^{n} v_j u_j = U_v$ is reached (see, e.g., Kellerer et al. [30]). This implies that the knapsack problem has at most $n$ different solution sets over varying $U_v$ values, and therefore, for a given $\phi$, we can easily obtain all the Pareto points in $O(n \log n)$ time as well. In other words, for a given permutation, the RDAP2-RDAP4 problems are all solvable in $O(n \log n)$ time for a continuous type of resource.

Corollary 3  The RDAP2-RDAP4 problems with a discrete type of resource are all $NP$-hard in the ordinary sense even for a fixed permutation, since the discrete knapsack problem which is known to be $NP$-hard in the ordinary sense (see, e.g., Kellerer et al. [30]) reduces to RDAP2.

The analysis in the remainder of the section is applicable for both continuous and discrete types of resources.

2.1 A polynomial time solution to the RDAP1 problem type

In this subsection we show that the optimal resource allocation decision for each agent is a sole function of the task he is assigned to, independent of the other assignment decisions. This will enable us to reduce RDAP1 to LAP and thus solve it in $O(n^3)$ time for both continuous and discrete types of resources.

Lemma 3  When expressed as a function of the assignment decision, the optimal resource allocation, $u^*_\phi(i)$ is:

$$u^*_\phi(i) = \begin{cases} 
0 & \text{if } \omega_i b_{\phi(i)} < v_{\phi(i)} \\
 u_{\phi(i)} \in [0, \pi_{\phi(i)}] & \text{if } \omega_i b_{\phi(i)} = v_{\phi(i)} \text{ for } i = 1, \ldots, n. \\
\pi_{\phi(i)} & \text{if } \omega_i b_{\phi(i)} > v_{\phi(i)}
\end{cases}$$

For a given permutation, $\phi$, the RDAP2 problem is reduced to minimize $\sum_{i=1}^{n} \omega_i \left( p_{\phi(i)} - b_{\phi(i)} u_{\phi(i)} \right)$, or equivalently to maximize $\sum_{i=1}^{n} \omega_i b_{\phi(i)} u_{\phi(i)} = \sum_{j=1}^{n} \theta_j u_j$ where $j = \phi(i)$ and $\theta_j = \omega_j b_j$ subject to $\sum_{j=1}^{n} v_j u_j \leq U_v$. This problem is known as the knapsack problem.

Proof.  The derivative of (9) with respect to $u_{\phi(i)}$ equals $v_{\phi(i)} - \omega_i b_{\phi(i)}$ for $i = 1, \ldots, n$. Therefore, if $v_{\phi(i)} > \omega_i b_{\phi(i)}$, we should not allocate any resource to agent $j = \phi(i)$, if $v_{\phi(i)} < \omega_i b_{\phi(i)}$, we allocate the maximal feasible amount of resource to agent $j = \phi(i)$ and if $v_{\phi(i)} = \omega_i b_{\phi(i)}$, any feasible resource allocation can be optimal. \hfill \blacksquare

As an outcome of Lemma 3, we can conclude that if agent $j$ has been assigned to task $i$, the optimal resource allocation for him is

$$u^*_i = \begin{cases} 
0 & \text{if } \omega_i b_j < v_j \\
u_j \in [0, \pi_j] & \text{if } \omega_i b_j = v_j \text{ for } i, j = 1, \ldots, n. \\
\pi_j & \text{if } \omega_i b_j > v_j
\end{cases}$$
Therefore, if we define the value $c_{ij}$ by

$$ c_{ij} = \omega_i (p_j - b_j u^*_i) + v_j u^*_i = \omega_i p_j + u^*_i (v_j - \omega_i b_j) = \begin{cases} 
\omega_i p_j & \text{if } \omega_i b_j \leq v_j \\
\omega_i p_j + u^*_i (v_j - \omega_i b_j) & \text{if } \omega_i b_j > v_j 
\end{cases} \quad (15) $$

it represents the minimal possible cost resulting from assigning agent $j$ to task $i$. Since each agent should be assigned to a single task and each task should be performed only once, RDAP1 is reduced to LAP. To summarize our analysis we present the following optimization algorithm to solve RDAP1.

**Algorithm 1** An optimization algorithm for RDAP1 for both continuous and discrete types of resources.

1. Calculate the $c_{ij}$ values by using eq. (15).
2. Solve LAP to determine the optimal assignment, $\phi^*$, of agents to tasks.
3. Allocate the resources according to eq. (13) with $\phi = \phi^*$.

**Theorem 1** Algorithm 1 solves RDAP1 in $O(n^3)$ time for both continuous and discrete types of resources.

**Proof.** The correctness of the algorithm follows from the analysis that appears in this section. Step 1 requires $O(n^2)$ time. Step 2 requires the solution of LAP which takes $O(n^3)$ time and Step 3 can be performed in linear time. Thus, the overall computational complexity of the algorithm is indeed $O(n^3)$. ■

### 2.2 The $\mathcal{NP}$-hardness of RDAP2-RDAP4

In this subsection, we prove the $\mathcal{NP}$-hardness of the RDAP2 and RDAP3 problem types for any given set of task cost parameters with $\omega_i \neq \omega_j$ for any $i \neq j$ by showing that their decision version, denoted by $DVP$ and defined below, is $\mathcal{NP}$-complete. This will clearly be an $\mathcal{NP}$-hardness proof also for the RDAP4 problem (note that the decision versions of RDAP2 and RDAP3 are identical). The proof is applicable for both continuous and discrete types of resources.

**Definition 1** $DVP$: Given an assignment problem with the assignment cost given by eq. (5) with (6), is there a feasible assignment with $\sum_{i=1}^{n} v_{\phi(i)} u_{\phi(i)} \leq U_v$ and $\sum_{i=1}^{n} \omega_i (p_{\phi(i)} - b_{\phi(i)} u_{\phi(i)}) \leq K$?

**Theorem 2** $DVP$ is $\mathcal{NP}$-complete for any set of $\omega_i$ parameters satisfying $\omega_i \neq \omega_j$ for any $i \neq j$, even for $v_j = 1$ for $j = 1, ..., n$ for both cases of continuous and discrete types of resources.

**Proof.** The $\mathcal{NP}$-completeness of $DVP$ will be proven by showing that the $\mathcal{NP}$-complete partition problem is polynomially reducible to $DVP$. The $\mathcal{NP}$-complete partition problem is defined as follows:

**Definition 2** Given a finite set $A = \{a_1, a_2, ..., a_h\}$ of positive integers where $\sum_{j=1}^{h} a_j = B$, can set $A$ be partitioned into two disjoint subsets, $A_1$ and $A_2$, where $\sum_{j \in A_i} a_j = B/2$ for $i = 1, 2$?

We construct the following instance of $DVP$ from an instance of the partition problem: There are $n = 2h$ agents and tasks with an assignment cost as given by eq. (5) with (6). Without loss of generality we assume that the task assignment cost parameters are renumbered such that $\omega_1 > \omega_2 > ... > \omega_n$. The assignment cost functions for agents $j$ and $j + h$ are given by

$$ p_j (u_j) = \frac{j B}{\omega_{2h}} + \frac{a_j - 2u_j}{\omega_{2j-1} + \omega_{2j}} \quad \text{for } 0 \leq u_j \leq a_j, \quad (16) $$

$$ u_j^* = \frac{B}{\omega_{2h}} \quad \text{for } 0 \leq u_j \leq a_j, \quad (17) $$

$$ v_j^* = B \quad \text{for } 0 \leq v_j \leq B, \quad (18) $$

Hence, $RDA\bar{P}1$ is polynomially reducible to $DVP$. The above construction is also $\mathcal{NP}$-hard.
\[ p_{j+h} = \bar{p}_{j+h} = \frac{jB}{\omega_{2h}} \]  \\
for \( j = 1, \ldots, h \), respectively. The one-unit resource consumption cost satisfies \( v_j = 1 \) for \( j = 1, \ldots, n \), and the limitations are 

\[ K = \sum_{j=1}^{h} (\omega_{2j} \bar{p}_j + \omega_{2j-1} \bar{p}_{j+h}) - \frac{B}{2}, \]  \\
and

\[ U_v = B/2. \]

Note that for the above instance for any \( 1 \leq j < k \leq h \) we have

\[ \max(p_j(u_j), p_{j+h}) \leq p_j(0) = \frac{jB}{\omega_{2h}} + \frac{a_j}{\omega_{2j-1} + \omega_{2j}} < \frac{jB + 0.5a_j}{\omega_{2h}} \leq \frac{(j + 0.5)B}{\omega_{2h}} \leq \frac{(k - 0.5)B}{\omega_{2h}} \]

\[ \leq \frac{kB - 0.5a_k}{\omega_{2h}} < \frac{kB}{\omega_{2h}} - \frac{a_k}{\omega_{2k-1} + \omega_{2k}}. \]

Since

\[ \frac{kB}{\omega_{2h}} - \frac{a_k}{\omega_{2k-1} + \omega_{2k}} = p_k(a_k) \leq p_k(u_k) \]

for any \( 0 \leq u_k \leq a_k \), and

\[ \frac{kB}{\omega_{2h}} - \frac{a_k}{\omega_{2k-1} + \omega_{2k}} < \frac{kB}{\omega_{2h}} = p_{h+k} \]

we can conclude that

\[ \max(p_j(u_j), p_{j+h}) \leq \min(p_k(u_k), p_{k+h}) \]  \\
for any \( j < k \leq h \),

which implies that

\[ p_k(u_k) \geq p_j(u_j); p_k(u_k) \geq p_{j+h}; p_{k+h} \geq p_j(u_j) \]  \\
and \( p_{k+h} \geq p_{j+h} \) for any \( j < k \leq h \),

for any \( u = (u_1, u_2, \ldots, u_h) \). Thus, according to Lemma 1, the optimal assignment is given by matching agents \( j \) and \( j+h \) to tasks \( 2j-1 \) and \( 2j \) for \( j = 1, \ldots, h \), while the question of which of the two agents will be assigned to which of the two tasks is a function of the resource allocation. Since \( p_j(u_j) \) is a decreasing function of \( u_j \) with \( p_j(a_j/2) = p_{h+j} \), we get that \( j = \phi(2j) \) and \( h+j = \phi(2j-1) \) under an optimal assignment for \( 0 \leq u_j < a_j/2 \), and vice versa for \( a_j/2 \leq u_j \leq a_j \).

Let us now show that if there is an instance which yields a positive answer to the partition problem, then there exists an assignment for the corresponding instance of DVP with \( U \leq U_v \) and \( c \leq K \). If there is such an instance, we construct the following solution for the assignment problem (denoted by assignment \( A_P \)): If \( j \in A_1 \) then \( j = \phi(2j-1) \) and \( h+j = \phi(2j) \) while if \( j \notin A_1 \) then \( j = \phi(2j) \) and \( h+j = \phi(2j-1) \). In addition, we set the resource allocation in \( A_P \) to be

\[ u_j = \begin{cases} 
  a_j & \text{for } j \in A_1 \\
  0 & \text{otherwise.}
\end{cases} \]  \\
(20)
Thus, for any agent under an optimal assignment, we have

\[ c \left( A_{NR} \right) = \sum_{j=1}^{2h} c_{j,\phi(j)}(A_{NR}) = \sum_{j=1}^{h} \left( \omega_{2j} \frac{jB}{\omega_{2h}} + \frac{a_j}{\omega_{2j-1} + \omega_{2j}} \right) = \sum_{j=1}^{h} \left( \omega_{2j} \frac{jB}{\omega_{2h}} + \omega_{2j-1} \frac{a_j}{\omega_{2j-1} + \omega_{2j}} + \omega_{2j-1} \frac{a_j}{\omega_{2j-1} + \omega_{2j}} \right) \]

Let us denote by \( A_{NR} \) the assignment where no resources are allocated. According to the above analysis, under an optimal assignment, \( j = \phi(2j) \) and \( h+j = \phi(2j-1) \) for \( j = 1, ..., h \), and thus the minimal cost for \( A_{NR} \) is given by

\[ c(\alpha_{NR}) = \frac{2h}{A_{j,\phi(j)}}(A_{NR}) = \sum_{j=1}^{h} \left( \omega_{2j} \frac{jB}{\omega_{2h}} + \frac{a_j}{\omega_{2j-1} + \omega_{2j}} \right) = \sum_{j=1}^{h} \left( \omega_{2j} \frac{jB}{\omega_{2h}} + \omega_{2j-1} \frac{a_j}{\omega_{2j-1} + \omega_{2j}} \right) \]

We can now calculate the difference between \( c(A_P) \) and \( c(A_{NR}) \). We define \( \Delta c_j = c_{j,\phi(2j)}(A_{NR}) + c_{2j-1,\phi(2j-1)}(A_{NR}) - c_{j,\phi(2j)}(A_P) - c_{2j-1,\phi(2j-1)}(A_P) \) for \( j = 1, ..., h \). It is clear from the definition of assignment \( A_P \) that for any job \( j \in A_2 \) we have that \( c_{2j,\phi(2j)}(A_{NR}) = c_{2j,\phi(2j)}(A_P) \) and \( c_{2j-1,\phi(2j-1)}(A_{NR}) = c_{2j-1,\phi(2j-1)}(A_P) \) and thus \( \Delta c_j = 0 \). However, for any \( j \in A_1 \) we have

\[ \Delta c_j = \omega_{2j} \left( \frac{jB}{\omega_{2h}} + \frac{a_j}{\omega_{2j-1} + \omega_{2j}} \right) + \omega_{2j-1} \frac{a_j}{\omega_{2j-1} + \omega_{2j}} \left( \frac{jB}{\omega_{2h}} - \frac{a_j}{\omega_{2j-1} + \omega_{2j}} \right) \]

Hence,

\[ c(A_{NR}) - c(A_P) = \sum_{j=1}^{h} \Delta c_j = \sum_{j \in A_1} a_j = B/2 \]

i.e.,

\[ c(A_P) = c(A_{NR}) - B/2 = \sum_{j=1}^{h} \left( \omega_{2j} \frac{jB}{\omega_{2h}} + \omega_{2j-1} \frac{a_j}{\omega_{2j-1} + \omega_{2j}} \right) \]

Conversely, we show that if the answer to the partition problem is negative, then there isn’t an assignment for the corresponding instance of DVP with \( U \leq U_v \) and \( c \leq K \). By contradiction, let us assume that there exists an assignment, \( A_P \), for the corresponding instance of DVP with \( U = \sum_{j=1}^{n} u_j \leq U_v = B/2 \) and \( c \leq K \). Then without loss of generality we may assume that \( \sum_{j=1}^{n} u_j = B/2 \). Let \( A_{NR}^2 \) be the set of agents with \( u_j = 0 \); \( A_{P}^2 \) be the set of agents with \( 0 < u_j < a_j \); and \( A_{P}^3 \) be the set of agents with \( u_j = a_j \) in \( A_P \). Since the answer to the partition problem is negative, set \( A_{P}^2 \) must include at least a single agent.

It is clear that \( \Delta c_j = u_j = 0 \) for any agent \( j \in A_{P}^2 \). Moreover, according to eq. (21) we have \( \Delta c_j = u_j = a_j \) for any agent \( j \in A_{P}^3 \). However, as we prove below, \( \Delta c_j < u_j \) for any agent \( j \in A_{P}^2 \). We divide the proof into two different cases for agent \( j \in A_{P}^2 \). The first case is where \( 0 < u_j < a_j/2 \) and the second is where \( a_j/2 \leq u_j < a_j \).

Case 1: If \( 0 < u_j \leq a_j/2 \) for agent \( j \in A_{P}^2 \), then \( j = \phi(2j) \) and \( h+j = \phi(2j-1) \) in an optimal assignment. Thus,

\[ \Delta c_j = \omega_{2j} \left( \frac{jB}{\omega_{2h}} + \frac{a_j}{\omega_{2j-1} + \omega_{2j}} \right) - \omega_{2j} \left( \frac{jB}{\omega_{2h}} + \frac{a_j - 2u_j}{\omega_{2j-1} + \omega_{2j}} \right) = \left( \frac{2\omega_{2j}}{\omega_{2j-1} + \omega_{2j}} \right) \times u_j < u_j \]
Case 2: If \( a_j/2 \leq u_j < a_j \) for agent \( j \in A^2_p \), then \( j = \phi(2j - 1) \) and \( h + j = \phi(2j) \) in an optimal assignment. Thus,

\[
\Delta c_j = (\omega_{2j} - \omega_{2j-1}) \left( \frac{jB}{\omega_{2h}} + \frac{a_j}{\omega_{2j-1} + \omega_{2j}} - p_{j+h} \right) + \omega_{2j-1} \left( \frac{2u_j}{\omega_{2j-1} + \omega_{2j}} \right)
\]

\[
= (\omega_{2j} - \omega_{2j-1}) \times \left( \frac{a_j}{\omega_{2j-1} + \omega_{2j}} \right) + \omega_{2j-1} \times \left( \frac{2u_j}{\omega_{2j-1} + \omega_{2j}} \right)
\]

\[
= a_j - \left( \frac{2\omega_{2j-1}}{\omega_{2j-1} + \omega_{2j}} \right) \times (a_j - u_j) < u_j.
\]

Then we have that

\[
c(A_{NR}) - c(A_P) = \sum_{j=1}^{h} (\Delta c_j) + \sum_{j=1}^{h} (\Delta c_j) + \sum_{j=1}^{h} (\Delta c_j) < \sum_{j=1}^{h} u_j = B/2,
\]

and thus

\[
c(A_P) > c(A_{NR}) - B/2 = \sum_{j=1}^{h} (\omega_{2j}P_j + \omega_{2j-1}p_{j+h}) - B/2 = K
\]

which contradicts the assumption that there exists an assignment with \( \sum_{j=1}^{n} u_j = B/2 \) and \( c \leq K \) and completes the proof. □

3 A Special Case of RDAP1-RDAP4 with a Reduced Computational Time

The following lemma is applicable for both cases of continuous and discrete types of resources and all problem types (RDAP1-RDAP4).

**Lemma 4** Consider that agents \( l \) and \( m \) satisfy the following conditions: \( b_l \geq b_m \), \( v_l \leq v_m \), \( \pi_l \geq \pi_m \) and \( \tilde{\pi}_l \leq \tilde{\pi}_m \). Then for any \( U \) value, there is an efficient assignment such that if agent \( l \) is assigned to task \( r \) and agent \( m \) is assigned to task \( s \) then \( r < s \), i.e., \( \omega_r > \omega_s \).

**Proof.** Consider an efficient assignment \( A \), in which agents \( l \) and \( m \) satisfy all of the conditions of the lemma. Assume that agent \( m \) is assigned to task \( r \) and agent \( l \) is assigned to task \( s \) with \( r < s \) in \( A \). In addition, let \( u_l(A) \) and \( u_m(A) \) be the amount of resource allocated to agents \( l \) and \( m \) in assignment \( A \), respectively. Since \( A \) is an efficient assignment, according to Lemma 1 we have that \( p_m(u_m(A)) \leq p_l(u_l(A)) \). Thus, since \( \pi_l \geq \pi_m \) and \( b_l \geq b_m \), it implies that \( u_m(A) \geq u_l(A) \). Interchange the assignment of agents \( l \) and \( m \) and their resource allocation to get assignment \( \tilde{A} \), i.e., in assignment \( \tilde{A} \), agent \( l \) is assigned to task \( r \), agent \( m \) is assigned to task \( s \) and \( u_m(\tilde{A}) = u_l(A) \) and \( u_l(\tilde{A}) = u_m(A) \) (this is feasible since \( \pi_l \geq \pi_m \)). Since \( v_l \leq v_m \), we have

\[
U = \sum_{j=1}^{n} v_j u_j(A) - \sum_{j=1}^{n} v_j u_j(\tilde{A}) = v_l(u_l(A) - u_m(A)) + v_m(u_m(A) - u_l(A)) = (v_m - v_l)(u_m(A) - u_l(A)). \quad (22)
\]
The term in eq. (22) is nonnegative since \( v_m \geq v_l \) and \( u_m(A) \geq u_l(A) \). In addition, we have

\[
\begin{align*}
c(A) - c(\bar{A}) &= \omega_r \times (\overline{p}_m - b_m u_m(A)) + \omega_s \times (\overline{p}_l - b_l u_l(A)) - \omega_r \times (\overline{p}_l - b_l u_m(A)) - \\
&\quad \omega_s \times (\overline{p}_m - b_m u_l(A)) = (\omega_r - \omega_s)(\overline{p}_m - \overline{p}_l) + (b_l - b_m)(u_m(A)\omega_r - u_l(A)\omega_s).
\end{align*}
\]

This last term is nonnegative as well since each of the terms in brackets that appears in the last equality in (23) is nonnegative. Thus, \( \bar{A} \) is also an efficient assignment.

Lemma 4 implies that we can restrict our search for efficient assignments in which every pair of agents satisfying the lemma’s conditions also satisfies the precedence constraint \( l \preceq m \), i.e., \( l \) is sequenced before \( m \) in the optimal permutation. Since there are four ordering conditions in the lemma, the partial order \( \preceq \) can be considered the intersection of these four linear orders, and thus it is a four-dimensional partial order. In any special case when these four orders are agreeable, i.e., they order all agents in the same sequence, the partial order \( \preceq \) becomes a linear (complete) order, which provides us with the optimal permutation, \( \phi^* \).

Then, we have the following corollary:

**Corollary 4** If the following four orders, \( b_l \geq b_m, v_l \leq v_m, \overline{u}_l \geq \overline{u}_m \) and \( \overline{p}_l \leq \overline{p}_m \), are agreeable, i.e., all agents are ordered in the same sequence, then the optimal permutation \( \phi^* \) can be obtained in \( O(n \log n) \) time.

The following three corollaries are now straightforward from Corollaries 2-4

**Corollary 5** If the following four orders, \( b_l \geq b_m, v_l \leq v_m, \overline{u}_l \geq \overline{u}_m \) and \( \overline{p}_l \leq \overline{p}_m \), are agreeable, then RDAP2-RDAP4 are all solvable in \( O(n \log n) \) time for a continuous type of resource.

**Corollary 6** The RDAP2-RDAP4 problems are all \( NP \)-hard in the ordinary sense even if the following four orders, \( b_l \geq b_m, v_l \leq v_m, \overline{u}_l \geq \overline{u}_m \) and \( \overline{p}_l \leq \overline{p}_m \), are agreeable for a discrete type of resource.

**Corollary 7** If the following four orders, \( b_l \geq b_m, v_l \leq v_m, \overline{u}_l \geq \overline{u}_m \) and \( \overline{p}_l \leq \overline{p}_m \), are agreeable, then due to Corollary 4, a simple sorting algorithm can replace Steps 1-2 in Algorithm 1 and thus the RDAP1 problem type is solvable in \( O(n \log n) \) time for both cases of continuous and discrete types of resources.

## 4 Some Important Scheduling Problems which are Equivalent to RDAP1-RDAP4

In this section we show that a large set of scheduling problems with controllable processing times is equivalent to RDAP and thus any result that was obtained concerning RDAP1-RDAP4 will hold for those scheduling problems as well. Next we briefly review the field of scheduling with controllable processing times and then, in each sub-section, present a different set of scheduling problems and show their equivalence to RDAP. We note that the set of scheduling problems presented here provides only a subset of a larger set of problems which are equivalent to RDAP.

In classical deterministic scheduling, job processing times are considered constant parameters. In various real-life systems, however, processing times may be controllable by allocating resources, such as additional money, overtime, energy, fuel, catalysts, subcontracting, or additional manpower, to the job operations (see, e.g., Janiak [22], Trick [53], Kayan and Akturk [29] and Shakhlevich and Strusevich [51]). Due to the large variety of applications, there is extensive literature on the subject of scheduling with controllable processing times (see, e.g., Vickson [54], Janiak [20], [21], [23] and [24], Alidaee and Ahmadian [2], Cheng et al. [9],...
Wan et al. [57], Hoogeveen and Weoinger [18], Ng et al. [37], Shakhlevich and Strusevich [50], Shabtay and Kaspi [46], Gurel and Akturk [14] and [16], Shabtay and Steiner [48], Yedidsion et al. [59] and Leyvand et al. [33]. A survey of results up to 1990 is provided by Nowicki and Zdrzalka [38], and more recent surveys are presented by Chudzik et al. [10], Janiak et al. [27] and Shabtay and Steiner [49].

A formal definition of scheduling problems with controllable processing times on a single machine may be stated as follows: $n$ independent jobs, $J = \{1, 2, \ldots, n\}$, are to be processed on a single machine. The processing time of job $j$, $p_j$, is a bounded linear function of the amount of resource, $u_j$, allocated to the processing operation as given by eq. (6). A solution is specified by a resource allocation vector $u = (u_1, u_2, \ldots, u_n)$ and by a job permutation $\phi \in \Phi$ where $\Phi$ is the set of all $n!$ possible permutations of the $n$ jobs (it is clear that any job permutation satisfies the assignment constraints (1)-(3)). The quality of a solution is measured by two criteria: The first, $f$, is a scheduling criterion and is dependent on the job completion times, and the second, $U$, is the resource consumption criterion. Among the $f$ criteria we consider here are $f \in \{\sum_{j=1}^n C_j; \sum_{j=1}^n W_j; \sum_{j=1}^n E_j; \sum_{j=1}^n T_j; C_{\max}\}$, where $C_j$ is the completion time of job $j$; $W_j = C_j - p_j$ is the waiting time of job $j$; $d_j$ is the due date of job $j$; $L_j = C_j - d_j$ is the lateness of job $j$; $T_j = \max(0, L_j)$ is the tardiness of job $j$; $E_j = \max(0, -L_j)$ is the earliness of job $j$ and $C_{\max} = \max_{j=1,\ldots,n} C_j$ is the maximal completion time (makespan). The $U$ criterion we consider is given by eq. (8) where here $v_j$ is the cost of assigning one unit of resource to the operation of job $j$.

To describe each scheduling problem in short we will use the standard three-field notation $x|y|z$ introduced by Graham et al. [13] and the extensions by T’kindt and Billaut [52] for formulating multicriteria scheduling problems. The $x$ field describes the machine environment. Since we deal with single machine problems, we set $x = 1$. The $y$ field exhibits the processing characteristics and constraints. We extend the $y$ field by including the information needed about the processing time function used. For example, if $dscr$ appears in this field, it means that we are dealing with a discrete type of resource and if $\text{lin}$ appears, it means that the linear function given by eq. (6) is assumed. We also put the upper bound constraints into the $y$ field for problem types P2 and P3. The $z$ field contains the optimization criteria.

Similar to our assignment problem, a scheduling with controllable processing times is essentially a problem with two criteria. Thus, four different variations of the scheduling problem can arise (see Hoogeveen [19] for a general review on multicriteria scheduling):

- The first one, which we denote by P1, is to minimize the total integrated cost, i.e., $f + U$. Using the scheduling notation introduced in [52], this problem can also be referred to as $1|\text{lin}|f + U$.
- The second, which we denote by P2, is to minimize $f$ subject to $U \leq U_\omega$. Following the notation in [52], we refer to this problem as $1|\text{lin}|e(f/U)$;
- The third, which we denote by P3, is to minimize $U$ subject to $f \leq K$, where $K$ is a given upper bound on the scheduling criterion. We refer to this problem by $1|\text{lin}|e(U/f)$ (based on [52]).
- The last, which we denote by P4 (and referred to by $1|\text{lin}|\#(f,U)$), is to identify the set of Pareto-optimal schedules for $(f,U)$.

In the following, we show that there is a large set of scheduling problems whose scheduling criterion is given by eq. (7) by presenting a subset of these set of problems. In those problems $\omega_i$ represents a fixed positional penalty of the $i$th position in the job permutation, $\phi$, and $p_j(u_j)$ represents the processing time of job $j$. Since according to $\phi$ each job can be assigned to a single position and each position can be
assigned only once, this set of scheduling problems is equivalent to RDAP, with a given set of assignment cost parameters, \( \omega_1, \omega_2, \ldots, \omega_n \), respectively. This will imply that the P2-P4 scheduling problems are all \( \mathcal{NP} \)-hard even for \( v_j = 1 \) for \( j = 1, \ldots, n \) and that the P1 scheduling problem can be solved in \( O(n^3) \) time. In addition, for the special case presented in Section 3 the P1-P4 scheduling problems can be solved in \( O(n \log n) \) time.

Since in all the scheduling problems we present below, the optimal schedule does not include idle times, we have that

\[
C_{\phi(i)} = \sum_{j=1}^{i} p_{\phi(j)}(u_{\phi(j)}), \tag{24}
\]

where \( j = \phi(i) \) represents the assignment of job (agent) \( j \) to position (task) \( i \) in job permutation \( \phi \).

### 4.1 The problem of minimizing the sum of completion times

The sum of completion time is one of the most important scheduling criterion. According to Pinedo [42], "the sum of completion time criterion is usually used as a surrogate criterion for minimizing the Work-In-Process (WIP) inventory. WIP ties up capital, and large amount of it can clog up operation. WIP increases handling cost, and older WIP can easily be damaged or become obsolete. Products are often not inspected until after they have completed their path through the production process. If a defect that is detected during final inspection is caused by a production step at the very beginning of the process, then all the WIP may be affected." It should be noted that minimizing the total completion time results in minimizing the mean waiting time and mean lateness in addition to minimize the WIP. According to eq. (24) the sum of completion times is given by

\[
f = \sum_{i=1}^{n} C_{\phi(i)} = \sum_{i=1}^{n} \sum_{j=1}^{i} p_{\phi(j)}(u_{\phi(j)}) = \sum_{i=1}^{n} (n - i + 1) \times p_{\phi(i)}(u_{\phi(i)}). \tag{25}
\]

Obviously eq. (25) is equivalent to eq. (7) with \( \omega_i = n - i + 1 \) for \( i = 1, \ldots, n \) and thus the related P1-P4 scheduling problems with \( f = \sum_{i=1}^{n} C_{\phi(i)} \) are equivalent to the assignment problems RDAP1-RDAP4 with \( \omega_i = n - i + 1 \) for \( i = 1, \ldots, n \), respectively. We note that this result can be extended to the more general case of \( m \) identical parallel machines in which the sum of completion times is given by (see, e.g., Gurel and Akturk [15])

\[
f = \sum_{i=1}^{n} C_{\phi(i)} = \sum_{i=1}^{n} \left[ \frac{n - i + 1}{m} \right] p_{\phi(i)}(u_{\phi(i)}). \tag{26}
\]

For this extended case, eq. (26) is equivalent to eq. (7) with \( \omega_i = \left[ \frac{n - i + 1}{m} \right] \) for \( i = 1, \ldots, n \).

There are some earlier results regarding problem types P1-P3 with \( f = \sum_{i=1}^{n} C_{\phi(i)} \). Vickson [54] studied the P1 problem type 1 \([\text{lin}, b_j = 1] \sum_{j=1}^{n} C_j + \sum_{j=1}^{n} v_j u_j\) and showed that it reduces to a linear assignment problem, which can be solved in \( O(n^3) \) time. Vickson’s [54] result can easily be extended to arbitrary \( b_j \) values and thus the 1 \([\text{lin}] \sum_{j=1}^{n} C_j + \sum_{j=1}^{n} v_j u_j\) problem is solvable in \( O(n^3) \) time as well. Janiak et al. [26] showed that the time complexity of the algorithm for the 1 \([\text{lin}] \sum_{j=1}^{n} C_j + \sum_{j=1}^{n} v_j u_j\) problem can be reduced to \( O(n^2) \) if \( b_j = 1 \) and \( \overline{u}_j = \overline{p}_j \) for \( j = 1, \ldots, n \). Lee [32] performed a sensitivity analysis on the optimal solution of the 1 \([\text{lin}, b_j = 1] \sum_{j=1}^{n} C_j + \sum_{j=1}^{n} v_j u_j\) problem by identifying the ranges of job processing times in which the optimal job sequence remains unchanged. Chen et al. [7] showed that Vickson’s method can also be used to solve the 1 \([\text{dscr}] \sum_{j=1}^{n} C_j + \sum_{j=1}^{n} v_j u_j\) problem in \( O(n^3) \) time. The special case of the P2 and the P3 problem types where \( b_j = b \) and \( \overline{p}_j = p \) for \( j = 1, \ldots, n \) was studied by Cheng et al. [9]. They proved
that the $1|\text{lin, dscr}, b_j = b, \overline{p}_j = p|\epsilon \left(\sum_{j=1}^{n} C_j / \sum_{j=1}^{n} u_j\right)$ problem is solvable in $O(n \log n)$ time and that the $1|\text{lin, dscr}, b_j = b, \overline{p}_j = p|\epsilon \left(\sum_{j=1}^{n} u_j / \sum_{j=1}^{n} C_j\right)$ problem is solvable in $O(n \log n \log(\sum_{j=1}^{n} \tau_j))$ time, where $\tau_j$ is the number of different possible processing times for job $j$. However, Cheng et al. did not determine the complexity of the more general case of arbitrary $b_j$ and $\overline{p}_j$ values, which heretofore remained an open question.

### 4.2 Due date assignment problems with an earliness/tardiness scheduling criterion

Meeting due-dates is one of the most important scheduling objectives. While traditional scheduling models considered due-dates as constant parameters, it is well-known that in a more flexible and integrated system, they are decision variables that are determined by taking into account the system’s ability to meet them. For this reason, recent studies have begun to view the due-date assignment as part of the scheduling process. Many different due-date assignment methods have been suggested in the literature (see Gordon et al. [11, 12] for extensive surveys on this subject). Four of the more commonly used methods are presented below:

- **The common** due date assignment method (referred to as $CON$), in which all jobs are assigned the same due date, that is $d_j = d$ for $j = 1, \ldots, n$, where $d_j$ denotes the due date of job $j$ and $d \geq 0$ is a decision variable (see Panwalkar et al. [39]).

- **The common due window** assignment method (referred to as $CONW$), in which the scheduler can assign a desired time window $[d, \overline{d} = d + D]$ for the completion time of each job. In this model, it is assumed that the earliness of a job is calculated with respect to $d$, while the tardiness is calculated with respect to $\overline{d}$ (see Liman et al. [35]). The scheduling criterion, $f$, includes a linear penalty for both $d$ and $D$. It is easy to see that the $CON$ method is a special case of the $CONW$ method in which the penalty for the window length ($D$) is large enough.

- **The slack** due date assignment method (referred to as $SLK$), in which all jobs are given a flow allowance that reflects equal waiting time (equal slacks), that is, $d_j = p_j + slk$ for $j = 1, \ldots, n$, where $p_j$ is the processing time of job $j$ and $slk \geq 0$ is a decision variable (see Adamopoulos and Pappis [1]).

- **The unrestricted** due date assignment method (referred to as $DIF$), in which each job can be assigned a different due date with no restrictions (see Seidmann et al. [43]).

The most common objective in scheduling with due date assignment and controllable processing times, includes penalties due to earliness, tardiness, due date assignment and makespan, as given by the following equation:

$$f(\phi, d) = \alpha \sum_{j=1}^{n} E_j + \beta \sum_{j=1}^{n} T_j + \gamma \sum_{j=1}^{n} d_j + \delta C_{\text{max}},$$

where $\alpha$, $\beta$, $\gamma$ and $\delta$ are nonnegative parameters representing the cost of one unit of earliness, tardiness, due date, and operation time, respectively, and $d = (d_1, d_2, \ldots, d_n)$ is the due date assignment decision vector. The objective in eq. (27) is usually converted into the following one when dealing with the $CONW$ method:

$$f(\phi, d) = \alpha \sum_{j=1}^{n} E_j + \beta \sum_{j=1}^{n} T_j + \gamma_1 nd + \gamma_2 nD + \delta C_{\text{max}},$$

(28)
where here \( \mathbf{d} = (\tilde{d}, D) \), and \( \gamma_1 \) and \( \gamma_2 \) are nonnegative parameters representing the cost of one unit of due date and window length, respectively.

In the following we show that the scheduling problems of minimizing eq. (28) for the CONW method and of minimizing eq. (27) for the CON, SLK and DIF are equivalent to RDAP. We close this subsection with an overview of some relevant results from the literature, all of which relate to the P1 problem type.

4.2.1 The CON due date assignment method

Panwalkar et al. [39] presented the following lemma which defines the optimal due date assignment strategy for the CON due date assignment method.

**Lemma 5** For the CON due date assignment method, there exists an optimal due date equal to \( C(\ell) \), where

\[
\ell = \max \left( \left\lceil \frac{n \times (\beta - \gamma)}{\alpha + \beta} \right\rceil, 0 \right),
\]

and \( C(0) = 0 \) by definition.

As a result of lemma 5, the following holds for any fixed \( \mathbf{u} \) and \( \phi \):

\[
d_j^* = d^* = C(\ell^*) = \sum_{i=1}^{\ell^*} p(\phi(i)) \quad \text{for} \quad j = 1, \ldots, n;
\]

\[
E(\phi(i)) = \begin{cases} 
\sum_{j=i+1}^{\ell^*} P(\phi(j)) & \text{for} \quad i < \ell^* \\
0 & \text{for} \quad i \geq \ell^*
\end{cases};
\]

\[
T(\phi(i)) = \begin{cases} 
\sum_{j=\ell^*+1}^{n} P(\phi(j)) & \text{for} \quad i > \ell^* \\
0 & \text{for} \quad i \leq \ell^*
\end{cases}.
\]

By substituting eqs. (30)-(32) into eq. (27) and considering the case of controllable processing times, we get a new expression for our scheduling criterion under an optimal due date assignment strategy for the CON due date assignment method:

\[
f(\phi, \mathbf{u}, \mathbf{d}^*(\phi, \mathbf{u})) = \sum_{i=1}^{\ell^*} (\alpha(i - 1) + \gamma n + \delta) \times p(\phi(i)) (u(\phi(i))) + \sum_{i=\ell^*+1}^{n} (\beta(n - i + 1) + \delta) \times p(\phi(i)) (u(\phi(i))),
\]

where \( \mathbf{d}^*(\phi, \mathbf{u}) \) denotes the optimal common due date assignment strategy.

Obviously eq. (33) is equivalent to eq. (7) with

\[
\omega_i = \begin{cases} 
\alpha(i - 1) + \gamma n + \delta & \text{for} \quad i \leq \ell^* \\
\beta(n - i + 1) + \delta & \text{for} \quad i > \ell^*
\end{cases}.
\]

Thus, the related P1-P4 scheduling problems are equivalent to the assignment problems RDAP1-RDAP4 with \( \omega_i \) as given by eq. (34) for \( i = 1, \ldots, n \).
4.2.2 The CONW due date assignment method

Liman et al. [35] presented the following lemma which defines the optimal due date assignment strategy for the CONW due date assignment method.

**Lemma 6** [35] Calculate

\[ l_1^* = \min \left( \max \left( \left\lfloor \frac{n \times (\gamma_2 - \gamma_1)}{\alpha} + 1 \right\rfloor, 0 \right), n \right) \]  
and

\[ l_2^* = \max \left( \left\lfloor \frac{n \times (\beta - \gamma_2)}{\beta} + 1 \right\rfloor, 0 \right). \]

If \( l_1^* < l_2^* \), then there exists an optimal \( \mathbf{d}^* \) equal to \( C_\phi(l_1^*) \) and an optimal \( \mathbf{\bar{d}}^* \) equal to \( C_\phi(l_2^*) \). However, if \( l_1^* \geq l_2^* \), then there exists an optimal window of zero length. Therefore, for this case the CONW method reduces to the CON method with \( \mathbf{d}^* = \mathbf{\bar{d}}^* \) equal to \( C_\phi(l^*) \), where \( l^* \) is calculated by eq. (29) using \( \gamma = \gamma_1 \) and \( C_\phi(0) = 0 \) by definition.

According to Lemma 6, if \( l_1^* \geq l_2^* \), then the optimal assignment of the due-window is identical to the optimal one for the CON method and therefore eq. (34) holds for the CONW method with \( \gamma = \gamma_1 \). Otherwise, if \( l_1^* < l_2^* \), then for any \( \mathbf{u} \) and \( \phi \), the following holds:

\[ \mathbf{d}^* = C_\phi(l_1^*) = \sum_{i=1}^{l_1^*} p_{\phi(i)}; \]

\[ \mathbf{\bar{d}}^* = C_\phi(l_2^*) = \sum_{i=1}^{l_2^*} p_{\phi(i)}; \]

\[ E_\phi(i) = \begin{cases} \sum_{j=i+1}^{l_1^*} p_{\phi(j)} & \text{for } i < l_1^* \\ 0 & \text{for } i \geq l_1^* \end{cases}; \]

\[ T_\phi(i) = \begin{cases} \sum_{j=l_2^*+1}^{i} p_{\phi(j)} & \text{for } i > l_2^* \\ 0 & \text{for } i \leq l_2^* \end{cases}. \]

By substituting eqs. (37)-(40) into eq. (28), and considering the case of controllable processing times, we get the following new expression for our objective under an optimal due date assignment strategy:

\[ f(\phi, \mathbf{u}, \mathbf{d}^*(\phi, \mathbf{u})) = \sum_{i=1}^{l_1^*} (\alpha(i-1) + n\gamma_1 + \delta) \times p_{\phi(i)}(u_{\phi(i)}) + (n\gamma_2 + \delta) \times \sum_{i=l_1^*+1}^{l_2^*} p_{\phi(i)}(u_{\phi(i)}) + \sum_{i=l_1^*+1}^{n} (\beta(n-i+1) + \delta) \times p_{\phi(i)}(u_{\phi(i)}), \]

where \( \mathbf{d}^*(\phi, \mathbf{u}) \) denotes the optimal due window assignment strategy as a function of \( \phi \) and \( \mathbf{u} \). Obviously eq. (41) is equivalent to eq. (7) with
\[
\omega_i = \begin{cases} 
\alpha(i - 1) + n\gamma_1 + \delta & \text{for } i \leq l_1^* \\
\nu\gamma_2 + \delta & \text{for } l_1^* < i \leq l_2^* \\
\beta(n - i + 1) + \delta & \text{for } i > l_2^* 
\end{cases}
\] (42)

Thus, the related P1-P4 scheduling problems are equivalent to the assignment problems RDAP1-RDAP4 with \(\omega_i\) as given by eq. (42) for \(i = 1, ..., n\).

4.2.3 The SLK due date assignment method

Adamopoulos and Pappis [1] showed that the CON and the SLK methods have similar properties and presented the following result for the SLK due date assignment method.

Lemma 7 For the SLK due date assignment method, there exists an optimal slack allowance, \(slk^*\), equal to \(C_{\phi(l^* - 1)}\), where \(l^*\) is given by eq. (29).

As a result of Lemma 7, the following holds for any \(u\) and \(\phi\):

\[
slk^* = C_{\phi(l^* - 1)} = \sum_{i=1}^{l^* - 1} p_i;
\] (43)

\[
d^*_\phi(j) = p_{\phi(j)} + slk^* = p_{\phi(j)} + \sum_{i=1}^{l^* - 1} p_{\phi(i)} \text{ for } j = 1, ..., n;
\] (44)

\[
E_{\phi(i)} = \begin{cases} 
\sum_{j=i}^{l^* - 1} p_{\phi(j)} & \text{for } i < l^* \\
0 & \text{for } i \geq l^* 
\end{cases}
\] (45)

\[
T_{\phi(i)} = \begin{cases} 
\sum_{j=i}^{l^* - 1} p_{\phi(j)} & \text{for } i > l^* \\
0 & \text{for } i \leq l^* 
\end{cases}
\] (46)

By substituting eqs. (44)-(46) into eq. (27) we get the following expression for our objective function under an optimal due date assignment strategy for the SLK due date assignment method:

\[
f(\phi, u, d^*(\phi, u)) = \sum_{i=1}^{l^* - 1} (\alpha i + \gamma(n + 1) + \delta) \times p_{\phi(i)}(u_{\phi(i)}) + \sum_{i=l^*}^{n} (\beta(n - i) + \gamma + \delta) \times p_{\phi(i)}(u_{\phi(i)}).
\] (47)

Obviously eq. (47) is equivalent to eq. (7) with

\[
\omega_i = \begin{cases} 
\alpha i + \gamma(n + 1) + \delta & \text{for } i \leq l^* - 1 \\
\beta(n - i) + \gamma + \delta & \text{for } i \geq l^* 
\end{cases}
\] (48)

Thus, the related P1-P4 scheduling problems are equivalent to the assignment problems RDAP1-RDAP4 with \(\omega_i\), as given by eq. (48) for \(i = 1, ..., n\).

4.2.4 The DIF due date assignment method

Seidmann et al. [43] presented the following lemma which defines the optimal due date assignment strategy for a given \(\phi\) and non-variable processing times.
Lemma 8 The optimal due date assignment strategy is defined as follows: if \( \gamma \geq \beta \) then set \( d_j = 0 \), otherwise set \( d_j = C_j \) for \( j = 1, \ldots, n \).

As an outcome of Lemma 8, our objective function with an optimal due date assignment strategy and as a function of \( u \), becomes

\[
f(\phi, u, d^*(\phi, u)) = \sum_{i=1}^{n} (\epsilon(n - i + 1) + \delta) \times p_{\phi(i)}(u_{\phi(i)}),
\]

where \( \epsilon = \min(\beta, \gamma) \). Obviously eq. (49) is equivalent to eq. (7) with

\[
\omega_i = \epsilon(n - i + 1) + \delta.
\]

Thus the related P1-P4 scheduling problems are equivalent to the assignment problems RDAP1-RDAP4 with \( \omega_i \) as given by eq. (50) for \( i = 1, \ldots, n \).

4.2.5 Overview of earlier results from the literature

Panwalkar and Rajagopalan [40] studied a special case of the P1 problem type for the CON method where \( b_j = 1 \) for \( j = 1, \ldots, n \), and \( \gamma = \delta = 0 \). They proved that the resulting P1 problem type 1 \([lin, CON, b_j = 1] \alpha \sum_{j=1}^{n} E_j + \beta \sum_{j=1}^{n} T_j + \sum_{j=1}^{n} v_j u_j \) is solvable in \( O(n^3) \) time by reducing it to a linear assignment problem. Cheng et al. [8] extended Panwalkar and Rajagopalan’s research by adding the due date cost to the objective and by also solving the problem with the SLK method in \( O(n^3) \) time. Liman et al. [34] showed that the complexity of the problem does not increase if a common due window is assigned. Cheng et al. [8] showed that if \( v_j = v \) for \( j = 1, \ldots, n \), the complexity reduces to \( O(n^2) \) for the CON, SLK and the CONW methods. For the CON method, Biskup and Jahnke [4] studied the special case where the job processing times are jointly reducible by the same proportional amount, i.e., the case where \( b_j = p_j \) and \( u_j = u \) for \( j = 1, \ldots, n \). They presented several \( O(n \log n) \) time optimization algorithms to minimize a cost function containing earliness, tardiness, resource consumption and due date assignment costs. Ng et al. [37] extended Biskup and Jahnke’s results to the case where the job processing times are jointly reducible by the same amount of the resource, i.e., where \( u_j = u \) for \( j = 1, \ldots, n \), and presented an \( O(n^2 \log n) \) time optimization algorithm for the same objective. Shabtay and Steiner [47] provided a unified optimization algorithm to minimize eq. (27) for the CON, SLK and DIF due date assignment methods in \( O(n^3) \) time. It is straightforward from the above literature review that no salient results have been achieved heretofore regarding problem types P2-P4 for any of the four different due date assignment methods.

4.3 The problem of minimizing the completion and waiting time deviation

Wang and Xia [58] remarked that "One of the most commonly occurring regular measures is the minimization of mean (or sum) completion time. Its attractiveness is perhaps due to its equivalence to mean waiting time, mean lateness and average in-process inventory. Yet in certain situations one is more interested in reducing variability in the completion time. For instance, in a service-oriented environment, one might be interested in providing as much uniform quality of service possible based on the customers’ waiting time in the system." Kanet [28] and Bagchi [3] suggested using the total absolute deviation of the jobs’ completion times and
waiting times as measures for completion time and waiting time variability, respectively. They showed that

\[ \sum_{i=1}^{n} \sum_{j=1}^{n} |C_i - C_j| = \sum_{i=1}^{n} (i-1)(n-i+1) \times p_{\phi(i)}(u_{\phi(i)}) \]  \hspace{1cm} (51)

and

\[ \sum_{i=1}^{n} \sum_{j=1}^{n} |W_i - W_j| = \sum_{i=1}^{n} i(n-i) \times p_{\phi(i)}(u_{\phi(i)}) \], \hspace{1cm} (52)

respectively. Obviously eq. (51) is equivalent to eq. (7) with

\[ \omega_i = (i-1)(n-i+1), \hspace{1cm} (53) \]

and eq. (52) is equivalent to eq. (7) with

\[ \omega_i = i(n-i). \hspace{1cm} (54) \]

Thus the related P1-P4 scheduling problems of minimizing either eq. (51) or eq. (52) are equivalent to the assignment problems RDAP1-RDAP4 with \( \omega_i \), as given by either eq. (53) or eq. (54), respectively for \( i = 1, \ldots, n \).

Wang and Xia [58] showed that the P1 type problems, 1 |lin, \( b_j = 1 \)|1| \( \delta_1 \sum_{i=1}^{n} \sum_{j=1}^{n} |C_i - C_j| + \delta_2 \sum_{j=1}^{n} C_j + \delta_3 \sum_{j=1}^{n} v_j u_j \) and 1 |lin, \( b_j = 1 \)|1| \( \delta_1 \sum_{i=1}^{n} \sum_{j=1}^{n} |W_i - W_j| + \delta_2 \sum_{j=1}^{n} W_j + \delta_3 \sum_{j=1}^{n} v_j u_j \) are solvable in \( O(n^3) \) time by adopting a similar approach to that used by Vickson [54]. Wang and Xia also showed that if \( v_j = u \) and \( u_j = u \) for \( j = 1, \ldots, n \), then the time complexity can be reduced to \( O(n \log n) \) for both problems. Again, also with respect to these two criteria, no one has presented heretofore any result regarding the P2-P4 problem types.

5 Summary of Results

This paper studied the complexity of an extension of the classical linear assignment problem, which has many practical and important applications in deterministic scheduling. In our extension, the cost of assigning agent \( j \) to task \( i \) is a multiplication of task \( i \)'s cost parameter by a cost function of agent \( j \) where the cost function of agent \( j \) is a linear function of the amount of resource allocated to the agent where the resource may be used either in continuous or discrete quantities. The quality of a solution is measured by two different criteria. The first criterion is the total assignment cost and the second is the total weighted resource consumption. We consider four different problem variations (RDAP1-RDAP4) for treating the two criteria.

Table 1 summarizes the complexity results obtained in this paper for all four different problem variations for both continuous and discrete types of resources. For each combination of resource type (continuous or discrete) and problem variation (RDAP1-RDAP4) the table also summarizes the set of all equivalent scheduling problems with controllable processing times where the \( f \) function can be any function that belongs to the set \( \{ \sum_{j=1}^{n} C_j, \sum_{j=1}^{n} E_j + \beta \sum_{j=1}^{n} T_j + \delta C_{\max}, \alpha \sum_{j=1}^{n} E_j + \beta \sum_{j=1}^{n} T_j + \gamma_1 nD + \gamma_2 nD + \delta C_{\max}, \sum_{i=1}^{n} \sum_{j=1}^{n} |C_i - C_j|, \sum_{i=1}^{n} \sum_{j=1}^{n} |W_i - W_j| \} \). The results for \( f = \alpha \sum_{j=1}^{n} E_j + \beta \sum_{j=1}^{n} T_j + \gamma_1 nD + \gamma_2 nD + \delta C_{\max} \) are applicable to the DIF, SLK and CON methods and the results for \( f = \alpha \sum_{j=1}^{n} E_j + \beta \sum_{j=1}^{n} T_j + \gamma_1 nD + \gamma_2 nD + \delta C_{\max} \) are applicable to the CONW method.
<table>
<thead>
<tr>
<th>Problem Variation</th>
<th>Resource Type</th>
<th>Equivalent Scheduling Problems</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDAP1</td>
<td>continuous and discrete</td>
<td>$1 \left\lfloor \text{lin} \right\rfloor f + U$</td>
<td>$O(n^3)$</td>
</tr>
<tr>
<td>RDAP1*</td>
<td>continuous and discrete</td>
<td>$1 \left\lfloor \text{lin} \right\rfloor f + U$</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>RDAP2-RDAP4</td>
<td>continuous and discrete</td>
<td>$1 \left\lfloor \text{lin} \right\rfloor v_j = 1 # (f, U)$; $1 \left\lfloor \text{lin} \right\rfloor v_j = 1 \left\lfloor \text{lin} \right\rfloor f$; $1 \left\lfloor \text{lin} \right\rfloor v_j = 1 # (f, U)$</td>
<td>$\mathcal{NP}$-hard</td>
</tr>
<tr>
<td>RDAP2-RDAP4*</td>
<td>continuous</td>
<td>$1 \left\lfloor \text{lin} \right\rfloor v_j = 1 # (f, U)$; $1 \left\lfloor \text{lin} \right\rfloor v_j = 1 \left\lfloor \text{lin} \right\rfloor f$; $1 \left\lfloor \text{lin} \right\rfloor v_j = 1 # (f, U)$</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>RDAP2-RDAP4*</td>
<td>discrete</td>
<td>$1 \left\lfloor \text{lin} \right\rfloor v_j = 1 # (f, U)$; $1 \left\lfloor \text{lin} \right\rfloor v_j = 1 \left\lfloor \text{lin} \right\rfloor f$; $1 \left\lfloor \text{lin} \right\rfloor v_j = 1 # (f, U)$</td>
<td>$\mathcal{NP}$-hard</td>
</tr>
</tbody>
</table>

* For the case where the four orders, $b_i \geq b_m$, $v_i \leq v_m$, $u_i \geq u_m$ and $p_i \leq p_m$, are agreeable

Table 1: Summary of complexity results obtained in this paper

It should be noted that the results presented in the first line of Table 1 were previously obtained separately for each scheduling criterion $f$ by Vickson [54], Wang and Xia [58], Panwalkar and Rajagopalan [40], Cheng et al. [8], Liman et al. [34] and Shabtay and Steiner [47]. However, this paper shows that this large set of results can be gathered into a single unified result. All other results obtained in this paper are new and applicable to a very large scale of scheduling problems whose complexity was previously an open question.

6 Discussion and Future Research

There are two major streams in scheduling with controllable processing times. The first (and more extensive one) is where the resource consumption function follows the linear model in (6) while the second uses a convex model. Both models capture a large set of real life applications. For example, Shakhlevich and Strusevich [51] present the following two important applications of the linear model:

- In operations management, in particular in the context of supply chain logistics, organizations are often faced with make-or-buy decisions, as well as coordinating internal production and outsourcing (see Chase et al. [6] and Waller [56]). In such cases, it may be profitable for a contractor to process only a part of the order internally for $p_j$ time units instead of its full processing requirement $\overline{p}_j$ using its own facilities and to hire a subcontractor to perform the remaining part of the order for $u_j = \overline{p}_j - p_j$ time units.

- In computing systems that support imprecise computations, a task with processing requirement $\overline{p}_j$ can be decomposed into a mandatory part which takes $p_j$ time, and an optional part that may take up to $\overline{p}_j - p_j$. If instead of an ideal computation time $\overline{p}_j$, a task is executed for $p_j = \overline{p}_j - u_j$ time, then computation is imprecise and $u_j$ corresponds to the error of computation. In this application, the total compression cost $v_j u_j$ corresponds to the total weighted error.

In contrast to these two applications, in many production processes the linear resource consumption function in eq. (6) may not be realistic as it fails to reflect the law of diminishing marginal returns. This law states that productivity increases at a decreasing rate with the amount of resource employed. In order to model this, other studies on scheduling with resource allocation assumed that the job processing time is a convex decreasing function of the amount of resource allocated to the processing of the job (see, e.g.,
Monma et al. [36], Shabtay [44], Shabtay and Kaspi [45] and Gurel and Akturk [14]). For a convex resource consumption function, researchers usually used the following function:

\[ p_j(u_j) = \left( \frac{\theta_j}{u_j} \right)^k, \]  

where \( \theta_j \) is a positive parameter, which represents the workload of job \( j \) and \( k \) is a positive constant. Monma et al. [36] pointed out that \( k = 1 \) corresponds to many actual government and industrial operations and the \( k = 0.5 \) case arises from VLSI (very large scale integration) circuit designs, where the product of the silicon area (resource) and the square of time spent equals a constant value (the workload) for an individual job. Other applications which support the convex model assumption appear in Janiak [22] and [25] and Trick [53].

It is well known that the analysis of the same problem for a different type of resource consumption function may lead to a completely different form of analysis and different complexity results. For example, in this paper we show that the P2-P4 problem variations of minimizing the sum of completion times on a single machine using the linear model in (6) are all \( \mathcal{NP} \)-hard. However, Shabtay and Kaspi [45] showed that the same problem variations are solvable in \( O(n \log n) \) time for the convex model in (55). Thus, one important issue for future research would be to analyze the four different variants of the RDAP problem for the convex model in (55) rather than for the linear model in (6).

We prove that the RDAP2-RDAP4 problems are all \( \mathcal{NP} \)-hard by reduction from the partition problem (see Theorem 1) leaving open the question of whether these problems are ordinarily or strongly \( \mathcal{NP} \)-hard. Our subsequent research has shown that the RDAP2-RDAP4 problems are \( \mathcal{NP} \)-hard in the ordinary sense by providing pseudo-polynomial algorithms to solve them. However, due to space limitations, we decided to describe these algorithms in a separate paper. We mention here that these algorithms are based on exploiting the well-known all-or-none property for scheduling problems with a linear model of the job processing time (see, e.g., Vickson [55] and Shabtay and Steiner [49]). This property states that there exists an optimal solution in which the processing time (the assignment cost) of each job (agent) \( j \in J \) (except at most a single one) is either fully reduced, i.e., \( p_j = \overline{p}_j - b_j \overline{\pi}_j \) or not reduced at all, i.e., \( p_j = \overline{p}_j \). Approximability issues could be a subject for future research as well.
References


