# Job assignment problem and traveling salesman problem: a linked optimisation problem.

OGUNSEMI, A., MCCALL, J., KERN, M., LACROIX, B., CORSAR, D. and OWUSU, G.

2022

# Job Assignment Problem and Traveling Salesman Problem: A Linked Optimisation Problem*

Akinola Ogunsemi[1], John McCall[1], Mathias Kern[3], Benjamin Lacroix[1], David Corsar[2], and Gilbert Owusu[3]

[1] National Subsea Centre, UK
{a.ogunsemi, j.mccall, b.m.e.lacroix}@rgu.ac.uk
[2] Robert Gordon University, Aberdeen, UK
{d.corsar1}@rgu.ac.uk
[3] BT Applied Research, Ipswich, UK
{mathias.kern, gilbert.owusu}@bt.com

**Abstract.** Linked decision-making in service management systems has attracted strong adoption of optimisation algorithms. However, most of these algorithms do not incorporate the complexity associated with interacting decision-making systems. This paper, therefore, investigates the linkages between two classical problems: job assignment problem and travelling salesman problem (JAPTSP) of a service chain system where service personnel perform tasks at different locations. We formulate a novel mathematical model from a linked optimisation perspective with objectives to minimise job cost and total travel distance simultaneously. We present three algorithmic approaches to tackling the JAPTSP: Non-dominated Sorting Genetic Algorithm for Linked Problem (NSGALP), Multi-Criteria Ranking Genetic Algorithm for Linked Problem (MCR-GALP), and Sequential approach. We evaluate the performance of the three algorithmic approaches on a combination of JAP and TSP benchmark instances. Results show that selecting an appropriate algorithmic approach is highly driven by specific considerations, including multi-objective base performance metrics, computation time, problem correlation and qualitative analysis from a service chain perspective.

**Keywords:** Service Chain Optimisation · Linked Optimisation Problem · Multi-Criteria Decision Making · Multi-Objective Optimisation

## 1 Introduction

The concept of linked decision-making arises in several different research communities involving two or more optimisation problems whose solutions interact. Optimisation algorithms are increasingly adopted to support such decision problems. But, most of these algorithms do not incorporate the complexity associated with the interacting decision-making process in the joint systems [1]. A linked optimisation problem explains the joint optimisation task involving $n$ (i.e. $n \geq 2$)

* Supported by BT and The DataLab

interdependent problems where a decision made for one problem causes a ripple effect on other dependent problems. Real-world problems, like service chains, are systems characterised by such interdependency, where some features of the subcomponents of the problem are linked [2]. In such linkages, an optimal solution for individual operational components might not guarantee an optimal solution for the overall problem [3]. An example is the integration between job assignment (JAP) and travelling salesman problem (TSP) of a service chain system where service personnel perform tasks at different locations. Thus, integrating the two problems results in multiple travelling salesman problems (MTSP).

JAP and TSP are two distinct classical optimisation problems whose integration applies to hospital resource planning, field service management [4], and supply chains. The recent global shocks (COVID-19, climate change, blockage of the Suez canal) have demonstrated the importance of interdependencies and the need to create service chains that are more resilient and have significantly reduced impact on the environment. For instance, during the first COVID-19 wave in England, three of five mandated health visiting services were paused to redeploy health visitors to respond to caseloads across several communities [5]. It was recommended that a clear plan for health visiting service is required to ensure sufficient capacity and manage missed appointments backlog [5]. JAP and TSP can be applied to the health visiting problem to provide sufficient capacity and allow spare capacity redeployment to respond to COVID caseloads.

We investigate the integration of JAP and TSP to minimise the total job assignment cost and travelling cost of visiting the job locations by the agents. To study JAPTSP, we use 114 combined problem instances of existing benchmarks in JAP and TSP. We exploit three algorithmic approaches to these combined problem instances. These algorithms include; Nondominated Sorting Genetic Algorithm for Linked Problem (NSGALP), Multi-Criteria Ranking Genetic Algorithm for Linked Problem (MCRGALP), and Sequential approach.

The paper is organised as follows. Section 2 gives a brief review of related work of JAPTSP. We define the JAPTSP in Section 3, Section 4 describes our approach and Section 5 provides our experimental setup and discuss results. Lastly, Section 6 concludes and presents future works.

## 2 Problem Background

JAPTSP refers to a class of optimisation problems where service personnel/agents are assigned to perform tasks in different cities. JAPTSP is an extension of the multiple travelling salesman problems (MTSP) and workforce scheduling and routing problems studied in the literature. So far, different variations of JAPTSP have been explored in the literature [6]. Castillo-Salazar et al. undertake a survey study in workforce scheduling [6]. They refer to scenarios where personnel carry out tasks at different locations as Workforce Scheduling and Routing Problem (WSRP). In the study of WSRP, [4] describes an iterated local search ILS algorithm. The paper evaluated ILS against a mixed integer programming (MIP) model and an adaptive larger neighbourhood search (ALNS) algorithm. Similarly, [7] proposed a greedy heuristic algorithmic design for five time-dependent

constraints for WSRP. Another variation of JAPTSP is seen in [8] involving the investigation of a Travelling Maintainer Problem (TMP) based on a generalised formulation of TSP. Their proposed problem seeks to find the best route for maintainers that minimises the travel, maintenance, and expected failure cost for all cities. The authors present a genetic algorithm and particle swarm optimisation solutions for comparison in the TMP study. Similarly, [9] adopts a genetic algorithm for a team scheduling problem. Also, [10] presents a mixed integer programming for multi-depot multiple travelling salesman problems (MmTSP) where an individual salesman travels from a particular location to a set of locations to complete tasks and return to the original location.

In JAPTSP, determining the optimal job assignment and obtaining the best multiple permutations of tours are the two decisions that must be taken simultaneously. In tackling the JAPTSP, we need to identify how the two problems (JAP and TSP) are connected. There are several ways of connecting them depending on how they interdepend. The integration of JAP & TSP causes complexity in designing appropriate algorithms for solving the problem.

## 3 Problem Formulation

### 3.1 Linked Problem Perspective

A linked optimisation problem $P$ of $n$ connected problems is;

$$P = \{p_1, p_2, \cdots, p_n, (D)\} : \quad p_\iota \in P \quad and \quad \iota = 1, \cdots, n \tag{1}$$

$$p_\iota = \left\{ x^\iota_{\{x^1_*, \cdots, x^n_*\} \setminus x^\iota_*}, f^\iota_{\{x^1_*, \cdots, x^n_*\} \setminus x^\iota_*}, c^\iota_{\{x^1_*, \cdots, x^n_*\} \setminus x^\iota_*} \right\} \tag{2}$$

In Eq. 2, $x^\iota_*$ denotes candidate solution in $x^\iota$. $\left\{x^1_*, \cdots, x^n_*\right\} \setminus x^\iota_*$ are feasible solutions for other problems in $P$ which affect $p_\iota$. $x^\iota_{\{x^1_*, \cdots, x^n_*\} \setminus x^\iota_*}$ is a search space of problem $p_\iota$. $f^\iota_{\{x^1_*, \cdots, x^n_*\} \setminus x^\iota_*} : x^\iota \to R$ denotes the objective function. $c^\iota_{\{x^1_*, \cdots, x^n_*\} \setminus x^\iota_*}$ denotes constraints set of $p_\iota$. $D = \left\{D^X, D^F, D^C\right\}$ connects the problems in Eq. 3.

$$D^X_{\iota j} / D^F_{\iota j} / D^C_{\iota j} = \begin{cases} 1, & \text{if } x^\iota_* \textit{ changes } x^j / f^j / c^j \\ 0, & \textit{Otherwise} \end{cases} \tag{3}$$

### 3.2 Job Assignment Problem JAP

From [16], let $I = \{1, 2, \cdots, m\}$ be a set of agents, and let $J = \{1, 2, \cdots, \mathbf{n}\}$ be a set of jobs, where $i \in I, j \in J$ respectively. Let $c_{ij}$ be the cost of assigning job $j$ to agent $i$, $\mathbf{r}_{ij}$ be the resource required by agent $i$ per job $j$, and $b_i$ be the capacity of agent $i$. $y_{ij}$ represents a $0-1$ variable, where 1 denotes that agent $i$ performs job $j$ and 0 otherwise. JAP seeks to find the assignment of jobs to agents $x_{_{JAP}}$ that minimises:

$$min f^1(x_{_{JAP}}) = \sum_{i=1}^{m} \sum_{j=1}^{\mathbf{n}} c_{ij} y_{ij} \tag{4}$$

Subject to:

$$\sum_{i=1}^{m} y_{ij} = 1 \quad \forall j \in J \tag{5}$$

$$\sum_{j=1}^{\mathbf{n}} \mathbf{r}_{ij} y_{ij} \leq b_i \quad \forall i \in I \tag{6}$$

$$y_{ij} \in \{0,1\} \quad \forall i \in I, \forall j \in J \tag{7}$$

Constraints 5 ensures that each job is assigned to exactly one agent, Constraints 6 ensures that the resource requirement of jobs assigned to an agent does not exceed the agent's capacity and Constraints 7 defines the decision variables.

### 3.3 Travelling Salesman Problem TSP

The travelling salesman problem (TSP) is one of the famous classical optimization problems that involves determining a tour that minimises the total distance traveled by a salesman [17]. TSP is defined by $\mathbf{n} \times \mathbf{n}$ distance matrix of $\mathbf{n}$ cities where the salesman is required to visit each city once. The distance between two cities $j$ and $k$ is defined by $d_{jk}$ and the objective function is given by:

$$min f^2(\mathbf{x}_{TSP}) = \sum_{j=2}^{\mathbf{n}} d_{j-1,j} + d_{\mathbf{n},1} \quad \mathbf{x}_{TSP} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_\mathbf{n}) \tag{8}$$

where $f^2(\mathbf{x}_{TSP})$ denotes the total traveling distance minimisation as the optimisation criterion of TSP and $\mathbf{x}_{TSP}$ represents a permutation of $\mathbf{n}$ cities.

### 3.4 JAPTSP

JAPTSP seeks to minimise the cost of job assignments and total traveling distance of visiting the assigned job locations by the agents. It is defined by $\mathbf{n}$ cities with given distance matrix $\{d_{jk}\}$, and $\mathbf{n}$ jobs to be assigned to $m$ agents/personnel with given availability/skill capacity/requirements. Here, the number of jobs for JAP corresponds to the number of locations in TSP. Each job in the JAP has a location in TSP. We define JAPTSP in Eq. 9

$$\begin{cases} min f^1(x_{JAP}) = \sum_{i=1}^{m} \sum_{j=1}^{\mathbf{n}} c_{ij} y_{ij} \\ min f^2_{x_{JAP}}(\mathbf{x}_{TSP}) = \sum_{i=1}^{m} f^2(\mathbf{x}_{TSP_i}) \end{cases} \tag{9}$$

JAPTSP is constrained by Constraints 5 - Constraints 7.

## 4 Proposed Approach

We propose Sequential approach, NSGALP and MCRGALP. Approaches are described in sections 4.1-4.4.

### 4.1 Genetic Components

The sequential algorithmic approach uses two genetic algorithms for each subproblem. The sub-problems have two different solution representations that uniquely differentiate the two algorithms in terms of encoding and genetic operators used by each algorithm. In NSGALP and MCRGALP, we embed the different encodings and the genetic operators in a single algorithmic process.

**Encoding** The encoding of JAPTSP uses two mechanisms; integer-based encoding for JAP and permutation-based encoding for TSP. The Integer-based solution representation addresses the assignment of jobs to agents. The permutation-based mechanism addresses a sequence of travel by agents.

**Initialisation** An initialisation is done by randomly generating a population of size $N$. In the sequential approach, each algorithm in Algorithm 1 generates its population separately and applies it to the genetic search process. It is quite different in NSGALP and MCRGALP. In Algorithms 2 and 3, each randomly generated solution of JAP instantiates the TSP and then generates a random solution for the modified TSP and pairs with the solution of the JAP.

**Non-Dominated Sort** Fast sort algorithm [18] is adopted in NSGALP to sort initial population. In the sorting process, we create a set $S$ that contains all the dominated solution pairs. Individuals left out form the first front. Next, the ones that dominate others in set $S$ are placed in the next. The process continues until we find the subset of $S$ where no individual dominates each other.

**Crowding Distance** Crowding distance is assigned front-wise and allows comparison within each front [19]. Calculation of crowding distance is defined in Eq. 10. Each front is considered individually and then sorted in non-decreasing order so that the first and last solution pairs are assigned infinite values.

$$\text{dist}^{\alpha,\beta}_{(JAP,TSP)} = \sum_{\iota=1}^{n} \frac{f^{\iota}_{\alpha,\beta+1} - f^{\iota}_{\alpha,\beta-1}}{f^{\iota}_{\alpha,max} - f^{\iota}_{\alpha,min}} \quad \forall \alpha \tag{10}$$

**TOPSIS** We adapted TOPSIS method as a selection operator in MCRGALP. TOPSIS is one of multiple criteria decision making methods that was first introduced by Yoon and Hwang [20]. TOPSIS decision-making technique is classified into five main steps [22]. Step 1 normalises decision matrix in Eq. 11.

$$r_{i\iota} = \frac{f^{\iota}_{i}}{\sum_{i=1}^{|pop^t|} (f^{\iota}_{i})^2} \tag{11}$$

Step 2 determines normalized weighted value $v_{i\iota}$ with weight $w_{\iota} = (w_1, \cdots, w_n)$ in Eq. 12.

$$v_{i\iota} = r_{i\iota} * w_{\iota} \tag{12}$$

Step 3 identifies the ideal best solutions $v^{+}_{\iota}$ and ideal worst solutions $v^{-}_{\iota}$ in Eq. 13 and Eq. 14.

$$v^{+}_{\iota} = \left\{ (\max v_{i\iota} | \iota \in I), (\min v_{i\iota} | \iota \in I'), i = 1, \cdots, |pop^t| \right\} = \left\{ v^{+}_{1}, \cdots, v^{+}_{n} \right\} \tag{13}$$

$$v^{-}_{\iota} = \left\{ (\min v_{i\iota} | \iota \in I), (\max v_{i\iota} | \iota \in I'), i = 1, \cdots, |pop^t| \right\} = \left\{ v^{-}_{1}, \cdots, v^{-}_{n} \right\} \tag{14}$$

Step 4 calculates Euclidean distance from $v^{+}_{\iota}$ and $v^{-}_{\iota}$, where $i = 1, 2, \cdots, |pop^t|$.

$$S^{+}_{i} = \sqrt{\sum_{\iota=1}^{n} (v_{i\iota} - v^{+}_{\iota})^2} \tag{15}$$

$$S^{-}_{i} = \sqrt{\sum_{\iota=1}^{n} (v_{i\iota} - v^{-}_{\iota})^2} \tag{16}$$

Step 5 calculates performance score and ranks the solution pairs.

$$\mathcal{P}_i = \frac{S_i^-}{S_i^+ + S_i^-} where \quad 0 \leq \mathcal{P}_i \leq 1 \tag{17}$$

**Genetic Operators** A genetic method uses crossover and mutation operators to update solutions during a search process[23]. Here, we use the same pair of crossover and mutation in the individual approach for the respective solution types. We use Integer SBX crossover operator for the JAP solutions and partially mapped crossover PMX for TSP solutions. Regarding mutation operators, we use Integer Polynomial mutation for updating the JAP solutions and permutation swap mutation for the TSP solutions, respectively. The genetic methodological framework uses the same crossover and mutation operators for all three approaches. In Algorithm 1, we adopt an integer-coded genetic algorithm $A_{JAP}$ which uses integer SBX crossover and integer polynomial mutation operators to generate offspring for the JAP. In terms of the TSP, we use a permutation-coded genetic algorithm $A_{TSP}$ in the sequential approach. $A_{TSP}$ uses PMX and permutation swap mutation to update solutions. We employed tournament selection in algorithms $A_{JAP}$ and $A_{TSP}$. The procedure for offspring generation is the same for Algorithms 2 and 3. The procedure is outlined as follows; Generate **n** offspring of JAP solutions from mating pool $\mathcal{R}^t_{JAP}$ using integer SBX crossover and integer polynomial mutation operators. Then, use each offspring to instantiate problem TSP and randomly generate $N$ TSP solutions. Next, perform crossover and mutation operations on $N$ solutions of TSP and generate **n** offspring. Then, evaluate the offspring and sort them in descending order. Last, select best offspring from **n** offspring of TSP and pair with each offspring of JAP.

### 4.2 Sequential Approach

The sequential algorithmic approach solves JAPTSP in sequence and is commonly known for solving problems in a hierarchical structure, usually between two decision-makers [25] [26]. Algorithm 1 shows a sequential approach for solving JAPTSP. First, algorithm $A_{JAP}$ solves problem $p_{JAP}$, selects the best solution $x^*_{JAP}$ then, uses best solutions $x^*_{JAP}$ to instantiate TSP $p_{TSP}$ based on their linkage structure. Next, the instantiated $p_{TSP}$ is solved using algorithm $A_{TSP}$ and then select best solution $\mathbf{x}^*_{TSP}$.

**Algorithm 1:** *SEQUENTIAL*

$x^*_{JAP} \leftarrow A_{JAP}|p_{JAP}$;
$\mathbf{x}^*_{TSP} \leftarrow A_{TSP}|(p_{TSP}, x^*_{JAP})$ ;
**Result:** $(x^*_{JAP}, \mathbf{x}^*_{TSP})$

### 4.3 NSGALP Approach

Here, we consider the solutions of JAP and TSP as a joint solution and adapt the fast nondominated sorting procedure, a fast crowded distance estimation procedure, and a simple crowded comparison operator based on [18] framework. Details about the framework is in [18]. Algorithm 2 shows a multi-objective framework we adopted in tackling JAPTSP.

**Algorithm 2:** *NSGALP*

$pop^0_{(JAP,TSP)} \leftarrow$ initialise ;
Evaluate $pop^0_{(JAP,TSP)}$ ;
Assign non-dominated sort to $pop^0_{(JAP,TSP)}$;
Apply crowding-distance to $pop^0_{(JAP,TSP)}$;
$t \leftarrow 0$ ;
**while** *Stopping criterion not met* **do**
  $\mathcal{R}^t_{(JAP,TSP)} \leftarrow$ Select from $pop^t_{(JAP,TSP)}$;
  $Q^t_{(JAP,TSP)} \leftarrow$ Generate offspring from $\mathcal{R}^t_{(JAP,TSP)}$;
  Evaluate $Q^t_{(JAP,TSP)}$;
  $\mathbf{pop}^t_{(JAP,TSP)} \leftarrow pop^t_{(JAP,TSP)} \cup Q^t_{(JAP,TSP)}$;
  Assign fast non-dominated sort to $\mathbf{pop}^t_{(JAP,TSP)}$ ;
  Apply crowding-distance assignment to $\mathbf{pop}^t_{(JAP,TSP)}$;
  $pop^{t+1}_{(JAP,TSP)} \leftarrow$ Select survivor from $\mathbf{pop}^t_{(JAP,TSP)}$;
  $t \leftarrow t+1$ ;
**end**
**Result:** $\mathcal{F}^1_{(JAP,TSP)}$

**Algorithm 3:** *MCRGALP*

$pop^0_{(FLP,PFSP)} \leftarrow$ Randomly initialise population ;
Fitness evaluation on $pop^0_{(JAP,TSP)}$ ;
$t \leftarrow 0$ ;
**while** *Stopping criterion not met* **do**
  Assign score to each solution pair in $pop^t_{(JAP,TSP)}$ ;
  $pop^*_{(JAP,TSP)} \leftarrow$ Get best $n$ pairs of $pop^t_{(JAP,TSP)}$;
  $Q^t_{(JAP,TSP)} \leftarrow$ Generate offspring from $pop^*_{(JAP,TSP)}$;
  Fitness evaluation on $Q_{(JAP,TSP)}$ ;
  Assign score to each solution pair in $Q_{(JAP,TSP)}$ ;
  $pop^t_{(JAP,TSP)} \leftarrow pop^t_{(JAP,TSP)} \cup Q^t_{(JAP,TSP)}$;
  $pop^{t+1}_{(JAP,TSP)} \leftarrow$ Get top $N$ solution pairs with best score from $pop^t_{(JAP,TSP)}$;
  $t \leftarrow t+1$ ;
**end**
**Result:** $(x^*_{JAP}, \mathbf{x}^*_{TSP})$

### 4.4 MCRGALP Approach

MCRGALP uses a similar approach in Section 4.3 but with differences in the output returned and the comparison operators used in the tournament selection. Unlike NSGALP, MCRGALP uses a multi-criteria performance metric known as Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS), which assigns a performance score to each joint solution. TOPSIS is widely used in multi-objective evolutionary algorithms. For example, [29] used TOPSIS as an evaluation approach to prioritise candidate solutions in their algorithmic framework. MCRGALP uses TOPSIS as a comparison operator in the tournament selection process to guide the selection of $n$ joint solutions at different phases of the algorithm process. See performance score computation in Section 4.1.

## 5 Experiments

We performed series of computational experiments to evaluate the proposed algorithmic approaches. The experiments are conducted on the same computer

environment with Intel Core i9, 2.4GHz, 32GB RAM, and Windows 10 Enterprise OS. The three algorithmic approaches are implemented in Java.

### 5.1 Benchmark Problems

We evaluate the proposed algorithmic approaches on two sets of instances for both problems in JAPTSP. The first set is JAP instances using Beasley [30] benchmark. The second set contains TSP instances of Gerhard's [31] benchmark. We combined each instance in JAP benchmark with each TSP instance based on their problem size. We assume that a job in JAP corresponds to a location/city in TSP. So, we obtained 114 combined instances in total. See Table 1.

**Table 1.** Linked Problem Instances

| Problem Size | No. of Agents | No. of Instances |
|---|---|---|
| 100 | 5 | **30** |
| 100 | 10 | **30** |
| 100 | 20 | **24** |
| 200 | 5 | **10** |
| 200 | 10 | **10** |
| 200 | 20 | **10** |
| **Total** | | **114** |

### 5.2 Exploratory Analysis of Problem Linkages

In the data exploration analysis, we generate randomly and evaluate 10000 solutions for JAP ($p_1$). For each solution generated on the JAP, we instantiate TSP ($p_2$) based on our linked optimisation framework. We then, generate randomly and evaluate 1000 solutions for the instantiated TSP and compute its mean value. Next, for each JAPTSP instance, we determine the relationship between the sub-problems using Spearman's correlation coefficient.

### 5.3 Performance Metric

We use four performance metrics. This includes; Hypervolume (HV) [32], Relative Hypervolume (RHV), Inverted Generational Distance (IGD)[33] and Multiplicative Epsilon [34]. For each problem instance, we obtained a reference point $\mathbf{r}$ and a reference front $\mathbf{Z}$ as input parameters for metric computations.

**Relative Hypervolume RHV** The relative hypervolume measures the proportion of hypervolume achieved by individual approach. This is computed by dividing the hypervolume of approximations by individual approach by the hypervolume of the true Pareto front. A higher RHV indicates that approximations are closer to the true Pareto front.

$$RHV(\mathbf{Z}, A) = \frac{HV(A, \mathbf{r})}{HV(\mathbf{Z}, \mathbf{r})} \tag{18}$$

where $0 \leq RHV(\mathbf{Z}, A) \leq 1$

**Hypervolume HV** HV considers the volume of the objective space dominated by an approximation set [35] bounded by a given reference point $\mathbf{r} \in R^2$. Higher HV values indicate a better performance of the corresponding approaches.

**Inverted Generational Distance IGD** IGD assesses the quality of approximations achieved by multi-objective algorithm to the Pareto front[33]. The metric measures how the approximations convergence towards the true Pareto front. The smaller the IGD value, the closer the calculated front to the true Pareto front[36]. IGD is calculated as follows:

$$IGD(A, \mathbf{Z}) = \big(\frac{1}{|\mathbf{Z}|}\sum_{i=1}^{|\mathbf{Z}|}\min_{a\in A}\mathbf{d}(\mathbf{z}, a)^2\big)^{\frac{1}{2}} \tag{19}$$

where $\mathbf{d}(\mathbf{z}, a) = \sqrt{\sum_{\iota}^{n}(\mathbf{z}_\iota, a_\iota)}$ with $a_\iota$ being the $\iota th$ fitness value of point $a$ from the approximations $A$ and $\mathbf{z}_\iota$ being an $\iota th$ fitness value of point $\mathbf{z}$ from the true Pareto front $\mathbf{Z}$.

**Multiplicative Epsilon $\epsilon$** The Epsilon indicator gives a factor by which an approximation set is worse than another with respect to all objectives [34]. A lower Epsilon value corresponds to a better approximation set, regardless of the type of problem (minimization, maximization or mixed). We compute as follows;

$$epsilon(A, \mathbf{Z}) = \max_{\mathbf{z}\in\mathbf{Z}}\min_{a\in A}\max_{1\leq\iota\leq n} epsilon(a_\iota, \mathbf{z}_\iota) \tag{20}$$

where $epsilon(a_\iota, \mathbf{z}_\iota) = a_\iota/\mathbf{z}_\iota$

### 5.4 Parameter Settings

Table 2 shows the parameters used by the individual approach. To measure the behavior of our approaches for solving the JAPTSP, we maintained the same parameter settings for the different genetic algorithm in all the approaches. An additional set of parameters are used by the TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution) method adopted in MCRGSLP approach. We set the weight for JAP fitness to 0.35 and its constraint, in case of violation, to 0.30. The weight for TSP is set to 0.35

**Table 2.** Parameter Settings

| **Parameters** | **NSGALP** | **MCRGALP** | **SEQUENTIAL** | |
|---|---|---|---|---|
| No. of Algorithms | **1** | **1** | | **2** |
| Experimental Runs | 100 | 100 | 100 | 100 |
| Population Size | 100 | 100 | 100 | 100 |
| Max Evaluations | 10000 | 10000 | 10000 | 10000 |
| Mating Pool Size | 100 | 100 | - | - |
| Offspring Size | 100 | 100 | 20 | 20 |
| IntegerSBXCrossover | 0.9 | 0.9 | 0.9 | - |
| PMXCrossover | 0.1 | 0.1 | - | 0.1 |
| Integer Polynomial Mutation | - | - | - | - |
| Permutation Swap Mutation | 0.5 | 0.5 | - | 0.5 |

### 5.5 Experimental Results and Analysis

**Performance Metrics** Table 3 shows the mean results of the four metrics. The best values are highlighted in bold font. NSGALP shows the best performance across all four metrics, although the mean results appear close to each other, especially in the hypervolume metric. MCRGALP slightly outperforms NSGALP

**Table 3.** Mean values of relative hypervolume, hypervolume, inverted generational distance and epsilon metrics of MCRGALP, NSGALP and SEQUENTIAL

| Size | m | RHV | | | HV | | | IGD | | | EPSILON | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MCRGALP | NSGALP | SEQ | MCRGALP | NSGALP | SEQ | MCRGALP | NSGALP | SEQ | MCRGALP | NSGALP | SEQ |
| 100 | 5 | 0.749 | **0.775** | 0.703 | 0.174 | **0.180** | 0.163 | 7048.24 | **6544.26** | 10598.16 | **1.186** | 1.187 | 1.226 |
| 100 | 10 | 0.711 | **0.778** | 0.670 | 0.186 | **0.204** | 0.172 | 8716.75 | **7049.15** | 12260.79 | 1.237 | **1.195** | 1.280 |
| 100 | 20 | 0.670 | **0.763** | 0.610 | 0.195 | **0.224** | 0.171 | 12505.42 | **8987.10** | 17753.56 | 1.301 | **1.188** | 1.448 |
| 200 | 5 | 0.794 | **0.834** | 0.794 | 0.154 | **0.163** | 0.154 | 11026.66 | **7454.52** | 13790.09 | 1.134 | **1.101** | 1.125 |
| 200 | 10 | 0.734 | **0.815** | 0.758 | 0.157 | **0.175** | 0.161 | 14559.31 | **9678.85** | 13654.81 | 1.195 | **1.133** | 1.177 |
| 200 | 20 | 0.697 | **0.808** | 0.721 | 0.159 | **0.186** | 0.163 | 18550.02 | **12371.31** | 17784.36 | 1.236 | **1.140** | 1.217 |

and sequential approaches in problem instance ($Size = 100$ and $m = 5$) in terms of epsilon metric. We check the significance of the difference between the statistical results using the Wilcoxon signed-rank test at 0.05 significance level. Table 4 summarizes the corresponding p values among the compared algorithms on instances grouped by problem size and size of agents. We highlight with bold font in Table 4 the comparisons that indicate no statistical difference in performance between the algorithms. Table 4 suggests that, despite the exceptional performance of NSGALP, MCRGALP and sequential approaches can also effectively tackle some instances of the linked problem, but that depends on how the two problems are linked. Figure 1 gives the overall perspective of the performance of the algorithmic approaches, and the selection of the best approach points toward NSGALP. However, there are several explanations for why NSGALP outperforms the other two. Metrics are multi-objective based and are influenced mainly by the number of non-dominated points produced by an algorithm.

**Table 4.** The p values of all metrics among the three algorithmic approaches on different problem combinations

| Size | m | RHV | | | HV | | | IGD | | | EPSILON | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | NSGALP vs SEQ | NSGALP vs MCRGALP | MCRGALP vs SEQ | NSGALP vs SEQ | NSGALP vs MCRGALP | MCRGALP vs SEQ | NSGALP vs SEQ | NSGALP vs MCRGALP | MCRGALP vs SEQ | NSGALP vs SEQ | NSGALP vs MCRGALP | MCRGALP vs SEQ |
| 100 | 5 | 1.49E-04 | **1.30E-01** | 9.88E-03 | 3.03E-03 | **3.40E-01** | 1.11E-04 | 3.50E-03 | **2.90E-01** | 1.22E-02 | 2.01E-04 | **9.00E-01** | 6.36E-05 |
| 100 | 10 | 4.80E-07 | 3.82E-10 | **8.24E-02** | 2.92E-02 | 6.38E-03 | 2.39E-04 | 5.08E-03 | **5.01E-02** | **1.67E-01** | 2.84E-04 | **3.55E-01** | 1.39E-06 |
| 100 | 20 | 1.20E-06 | 1.34E-08 | 3.28E-02 | **5.39E-02** | 3.53E-03 | 7.21E-05 | 1.01E-03 | 6.29E-03 | **1.40E-01** | **8.45E-01** | 4.88E-03 | 3.75E-04 |
| 200 | 5 | **8.90E-02** | **7.57E-02** | **9.70E-01** | 4.52E-02 | **1.86E-01** | 1.01E-03 | 2.57E-02 | **1.40E-01** | **8.90E-02** | 2.57E-02 | **2.73E-01** | 4.40E-04 |
| 200 | 10 | 4.52E-02 | 3.30E-04 | **5.71E-01** | **6.40E-02** | **1.40E-01** | 1.71E-03 | **1.40E-01** | 2.11E-02 | **2.41E-01** | 5.80E-03 | **6.78E-01** | 1.71E-03 |
| 200 | 20 | 3.76E-02 | 5.83E-04 | **5.71E-01** | **1.21E-01** | 2.57E-02 | 7.28E-03 | **7.57E-02** | 7.28E-03 | **2.41E-01** | **6.40E-02** | **3.45E-01** | 2.83E-03 |

**Computational Time Complexity** We also consider the performance of the algorithmic approaches based on computational time. Figure 2 shows four plots of mean computational time against the selected performance metrics. Figure 2a shows the mean computing time against the relative hypervolume metric, Figure 2b shows the mean computing time against the hypervolume metric, Figure 2c IGD and Figure 2d shows the mean computing time against the epsilon metric. There is no doubt that the sequential approach required less computational time than the other approaches in all combinations of problem instances.

**Correlation Analysis** We further consider algorithm performance in terms of the correlation score obtained by randomly generated solutions, as discussed in Section 5.2. We compare the performance of the competing algorithms on instances that obtained the lowest (-0.0264), median (-0.0016) and highest (0.0198)

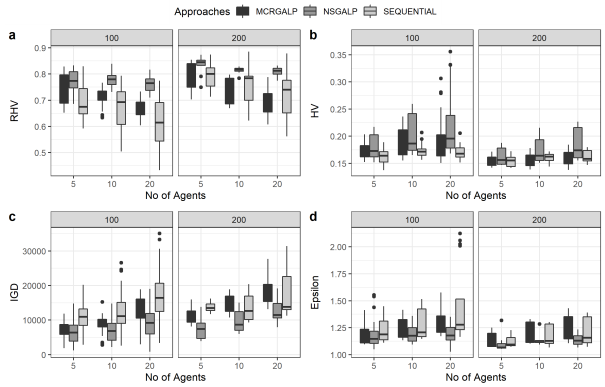


**Fig. 1.** Overall performance based on RHV, HV, IGD and Epsilon metrics

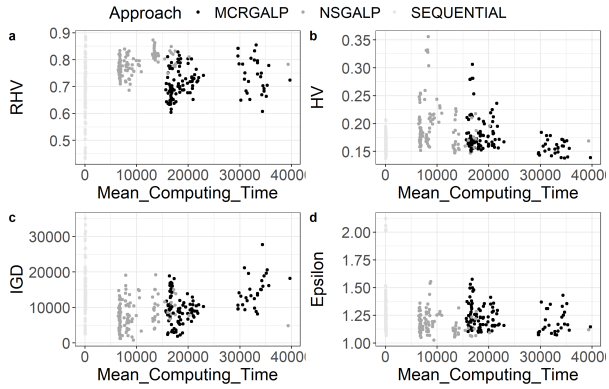


**Fig. 2.** Mean Computation Time against Performance Metrics

correlation coefficients. Figure 3 shows the empirical attainment function obtained by competing algorithmic approaches on the instances with respective minimum, median and maximum correlation scores. Overall, the NSGALP approach produces the best attainment coverage.

**Service Chain Perspective** Suppose we consider the two problems from the perspective of two service companies or business units involved in a service chain. Our results can offer guidance on the benefits and costs they will likely experience based on the approach used to solve the overall problem. For example, the NSGALP is quite interesting due to the large extent of variability in fitness values of the JAP criterion. In Figure 4a, the NSGALP solves the first problem in a view to trade-off the other problem. However, the trade-off of the second problem is not that much, as there is a peak at the lower TSP fitness value. However, the MCRGALP tends to hit a sweet spot for both problems in 99% of the problem instances, but this does not quantify the performance metrics

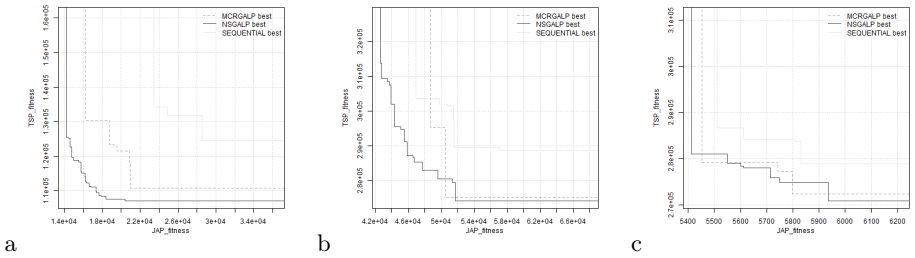


**Fig. 3.** Empirical Attainment Function of algorithmic approaches on problem instances with minimum, median and maximum correlation coefficients Approaches

used. Therefore, in deciding how to solve the problem with the sequential and NSGALP, it is more apparent that both companies must consider the impact that optimising one problem will have on the other and decide if the resulting costs/benefits are acceptable. In contrast, the MCRGALP maintains a balanced compromise on both problems.

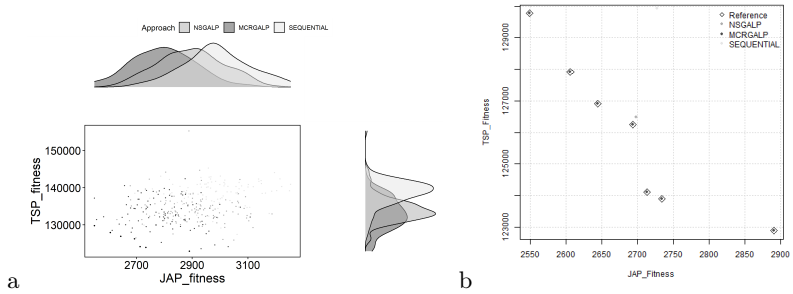


**Fig. 4.** Distribution of solutions found by all approaches on problem size 100 & m=5.

## 6 Conclusion and Future Work

This paper presented a linked optimisation problem (JAPTSP) of two minimisation problems: JAP and TSP. The JAP assigns agents to jobs that minimise the job cost, while the TSP determines a subset of tours that minimise total travelling distance. We employed three algorithmic approaches; NSGALP, MCRGALP, and SEQUENTIAL to tackle the JAPTSP. We compare the performance of the three approaches on 114 combinations of problem instances. Empirical results indicate that NSGALP outperforms the other two methods. We also consider other factors in selecting an appropriate algorithmic method. These factors include: mean computation time, degree of correlation between the combined problem instance, and qualitative analysis from a service chain perspective. MCRGALP seems to maintain balanced multiple decision-making

without sacrificing one for the other. In terms of mean computational time, the sequential method outperforms the other two methods. Concerning future research, we need to consider the use of appropriate performance metrics that measure how algorithms perform towards obtaining results that converge to an equilibrium point (i.e. a balanced joint solution) which is unbiased towards a method. Further exploration of some properties of the algorithms can be undertaken as the good performance of the NSGALP and MCRGALP results in sacrificing computational time.

## References


1. M. Ibrahimov, A. Mohais, S. Schellenberg, and Z. Michalewicz, "Evolutionary approaches for supply chain optimisation: part i: single and two-component supply chains," *IJICC*, vol. 5, no. 4, pp. 444–472, 2012a.
2. M. R. Bonyadi, Z. Michalewicz, and L. Barone, "The travelling thief problem: The first step in the transition from theoretical problems to realistic problems," in *2013 IEEE CEC*. IEEE, 2013, pp. 1037–1044.
3. D. K. Vieira, G. L. Soares, J. A. Vasconcelos, and M. H. Mendes, "A genetic algorithm for multi-component optimization problems: the case of the travelling thief problem," in *EvoCOP*. Springer, 2017, pp. 18–29.
4. F. Xie, C. N. Potts, and T. Bektaş, "Iterated local search for workforce scheduling and routing problems," *JH*, vol. 23, no. 6, pp. 471–500, 2017.
5. G. Conti and A. Dow, "The impacts of covid-19 on health visiting services in england: Foi evidence for the first wave," 2020.
6. A. Castillo-Salazar, D. Landa-Silva, and R. Qu, "A survey of workforce scheduling and routing," 2012.
7. J. A. Castillo-Salazar, D. Landa-Silva, and R. Qu, "A greedy heuristic for workforce scheduling and routing with time-dependent activities constraints," 2015.
8. F. Camci, "The travelling maintainer problem: integration of condition-based maintenance with the travelling salesman problem," *JORS*, vol. 65, no. 9, pp. 1423–1436, 2014.
9. T. Zhang, W. Gruver, and M. H. Smith, "Team scheduling by genetic search," in *IPMM'99 (Cat. No. 99EX296)*, vol. 2. IEEE, 1999, pp. 839–844.
10. M. Assaf and M. Ndiaye, "Multi travelling salesman problem formulation," in *2017 4th ICIEA*. IEEE, 2017, pp. 292–295.
11. Y. Shuai, S. Yunfeng, and Z. Kai, "An effective method for solving multiple travelling salesman problem based on nsga-ii," *SSCE*, vol. 7, no. 2, pp. 108–116, 2019.
12. J. Stolk, I. Mann, A. Mohais, and Z. Michalewicz, "Combining vehicle routing and packing for optimal delivery schedules of water tanks," *OR Insight*, vol. 26, no. 3, pp. 167–190, 2013.
13. L. Chen, A. Langevin, and Z. Lu, "Integrated scheduling of crane handling and truck transportation in a maritime container terminal," *EJOR*, vol. 225, no. 1, pp. 142–152, 2013.
14. T.-L. Chen, C.-Y. Cheng, Y.-Y. Chen, and L.-K. Chan, "An efficient hybrid algorithm for integrated order batching, sequencing and routing problem," *IJPE*, vol. 159, pp. 158–167, 2015.
15. S. Moons, K. Ramaekers, A. Caris, and Y. Arda, "Integrating production scheduling and vehicle routing decisions at the operational decision level: a review and discussion," *CIE*, vol. 104, pp. 224–245, 2017.

16. P. C. Chu and J. E. Beasley, "A genetic algorithm for the generalised assignment problem," *COR*, vol. 24, no. 1, pp. 17–23, 1997.
17. R. Gerhard, "The traveling salesman: computational solutions for tsp applications," *Lecture Notes in Computer Science*, vol. 840, pp. 1–223, 1994.
18. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE TEC*, vol. 6, no. 2, pp. 182–197, 2002.
19. P. K. Muhuri, Z. Ashraf, and Q. D. Lohani, "Multiobjective reliability redundancy allocation problem with interval type-2 fuzzy uncertainty," *IEEE TFS*, vol. 26, no. 3, pp. 1339–1355, 2017.
20. C.-L. Hwang and K. Yoon, "Methods for multiple attribute decision making," in *MADM*. Springer, 1981, pp. 58–191.
21. R. Rahim, S. Supiyandi, A. Siahaan, T. Listyorini, A. P. Utomo, W. A. Triyanto, Y. Irawan, S. Aisyah, M. Khairani, S. Sundari *et al.*, "Topsis method application for decision support system in internal control for selecting best employees," in *JP: Conference Series*, vol. 1028, no. 1. IOP Publishing, 2018, p. 012052.
22. E. Triantaphyllou, B. Shu, S. N. Sanchez, and T. Ray, "Multi-criteria decision making: an operations research approach," *EEEE*, vol. 15, no. 1998, pp. 175–186, 1998.
23. G. Luo, X. Wen, H. Li, W. Ming, and G. Xie, "An effective multi-objective genetic algorithm based on immune principle and external archive for multi-objective integrated process planning and scheduling," *IJAMT*, vol. 91, no. 9, pp. 3145–3158, 2017.
24. T. Geetha and K. Muthukumaran, "An observational analysis of genetic operators," *IJCA*, vol. 63, no. 18, pp. 24–34, 2013.
25. F. Legillon, A. Liefooghe, and E.-G. Talbi, "Cobra: A cooperative coevolutionary algorithm for bi-level optimization," in *2012 IEEE CEC*. IEEE, 2012, pp. 1–8.
26. M. Ibrahimov, "Evolutionary algorithms for supply chain optimisation." Ph.D. dissertation, 2012.
27. C. K. Y. Lin, "Solving a location, allocation, and capacity planning problem with dynamic demand and response time service level," *MPE*, vol. 2014, 2014.
28. C. A. Ullrich, "Integrated machine scheduling and vehicle routing with time windows," *EJOR*, vol. 227, no. 1, pp. 152–165, 2013.
29. A. Nourmohammadi and M. Zandieh, "Assembly line balancing by a new multi-objective differential evolution algorithm based on topsis," *IJPR*, vol. 49, no. 10, pp. 2833–2855, 2011.
30. J. E. Beasley, "Or-library: distributing test problems by electronic mail," *JORS*, vol. 41, no. 11, pp. 1069–1072, 1990.
31. G. Reinelt, "Tsplib95," *IWR, Heidelberg*, vol. 338, pp. 1–16, 1995.
32. E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE TEC*, vol. 3, no. 4, pp. 257–271, 1999.
33. L. C. Bezerra, M. López-Ibánez, and T. Stützle, "An empirical assessment of the properties of inverted generational distance on multi-and many-objective optimization," in *EMO*. Springer, 2017, pp. 31–45.
34. E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE TEC*, vol. 7, no. 2, pp. 117–132, 2003.
35. E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms—a comparative case study," in *PPSN*. Springer, 1998, pp. 292–301.
36. J. A. Manson, T. W. Chamberlain, and R. A. Bourne, "Mvmoo: Mixed variable multi-objective optimisation," *JGO*, vol. 80, no. 4, pp. 865–886, 2021.