CHAPTER 3

BI-OBJECTIVE CAPACITATED SUPPLY CHAIN NETWORK

3.1 BI-OBJECTIVE RESOURCE ALLOCATION PROBLEM WITH VARYING CAPACITY

One of the important extensions of the classical RA problems in the context of bi-objective capacitated supply chain is Mixed Capacitated Arc Routing Problems (MCARP). The Generalized Assignment Problem (GAP) is a well-known, NP-hard combinatorial optimization problem which involves finding the minimum cost assignment of $n$ jobs to $m$ agents such that each job is assigned to exactly one agent, subject to an agent's available capacity. Assignment of jobs to computers in a computer network, storage space allocation, design of communications network with node capacity constraints, allocation of customers to warehouses are examples of practical applications to GAP. It also appears as sub-problem in many real-life problems such as vehicle routing, plant location, warehouse allocation and flexible manufacturing systems. Recent extensive reviews of applications of the GAP and the existing exact and heuristic algorithms can be found in (Edwin Romeijn, H. and Dolores Romero Morales, 2000; Maria A. Osorio, and Manuel Laguna, 2003; Salim Haddadi and Hacene Ouzia, 2004) will not be repeated here. The existing exact algorithms are only effective in certain GAP instances where the constraints are loose. For the more difficult highly capacitated problems, exact algorithms can only solve problems involving up to a few hundred decision variables before the search trees grow prohibitively large. Thus larger-sized problems are often tackled by applying heuristics to obtain approximate solutions. Let $I = \{I_1, \ldots, I_m\}$ be the set of agents and $J =$
\{I,\ldots,n\}$ the set of jobs. A standard integer programming formulation for the GAP is the following:

\begin{align}
\text{Minimize} \quad & z = \sum_{i \in I} \sum_{j \in J} c_{ij}x_{ij} \\
\text{Subject to} \quad & \sum_{i \in I} x_{ij} = 1 \quad \forall \ j \in J, \quad (3.1) \\
& \sum_{j \in J} a_{ij}x_{ij} \leq b_i \quad \forall \ i \in I, \quad (3.2) \\
& x_{ij} \in \{0,1\} \quad \forall \ i \in I, \forall \ j \in J \quad (3.3)
\end{align}

Where $c_{ij}$ is the cost of assigning job $j$ to agent $i$. $a_{ij}$ is the resource required for processing job $j$ by agent $i$, and $b_i$ is the available capacity of agent $i$. Decision variables $x_{ij}$ are set to 1 if job $j$ is assigned to agent $i$, 0 otherwise. Constraint (3.2) (3.3) together with the integrality conditions on the variables, state that each job is assigned exactly to one agent. Constraint (3.4) ensures that the resource availability of agents is not exceeded. Assignment of jobs to computers in a computer network, allocation of storage space, communication network design with capacity constraints on the node are some of the practical examples of GAP. But, often it appears as sub-problem in various real-life problems such as location of facilities, vehicle routing, scheduling of resources, cellular and flexible manufacturing systems. This is addressed by various researchers and numerous solution methodologies are proposed for this variant. The variant considered in this chapter is in the context of bi-objective capacitated supply chain network and one of the classical RA problems and it is an extension of GAP with the consideration of two objectives in the problem and constraints on varying capacity.

This is a new variant which is less addressed by previous researchers in the literature and the variant is termed as Bi-Objective Resource Allocation Problem with Varying Capacity (BORAPVC). The base variant of BORAPVC is GAP. The BORAPVC can be defined as: Given $n$ source nodes,
destination nodes, the two objective \( O1_{ij} \) and \( O2_{ij} \) the influence \( I_{ij} \) corresponding to the assignment of source node \( j \) to destination node \( i \), and the total capacity \( C_i \) available for destination node \( i \), the BORAPVC is to determine how to select and assign each of the source nodes to exactly allocate to one of the destination nodes in order to maximize or minimize or compromise the two objectives, subject to the capacity constraints of destination source to produce non-dominated solutions. BORAPVC applicable to a real world practical supply chain problem of allocating a set of retailers to multiple distributors with different capacities with two specific performance objectives such as travel distance and travel time is considered. This paper considers bi-objective BORAPVC for the application of physical distribution and logistics. Development of a methodology that addresses the BORAPVC is a target, which solves it within reasonable limits of accuracy and computational time and generate a wide range of non-dominated solutions without the determination of weights. When embedded in the planner’s software, it can be a valuable tool towards providing service of high quality at low cost with high number of Pareto solutions. Applications of BORAPVC are in automotive and process industry which include warehouse allocation to customers in distribution, supplier allocation to manufacturing plant in sourcing and distributor allocation to retailer in delivery.

### 3.2 SOLUTION METHODOLOGY TO SOLVE BORAPVC

Unified solution methodologies were developed to solve \( GAP \) and \( BORAPVC \)

- Mathematical programming model for \( BORAPVC \)
- Unified heuristic named Simulated Annealing with Population Size Initialization through Neighbourhood Generation (SAPING) is proposed to solve both \( GAP \) and \( BORAPVC \).
3.2.1 Mathematical Programming Model for BORAPVC

Mathematical programming model for BORAPVC is detailed below:

Minimize \[ z_1 = \sum_{k,I} \sum_{j,J} d_{ij} x_{ij} \] (3.5)

Minimize \[ z_2 = \sum_{k,I} \sum_{j,J} t_{ij} x_{ij} \] (3.6)

Subject to

\[ \sum_{i,J} x_{ij} = 1 \quad \forall \ j \in J, \] (3.7)

\[ \sum_{j,J} v_i x_{ij} \leq q_i \quad \forall \ i \in I, \] (3.8)

\[ x_{ij} \in \{0,1\} \quad \forall \ i \in I, \forall \ j \in J \] (3.9)

Where \( d_{ij} \) is the travel distance and \( t_{ij} \) is the travel time between retailer \( j \) to distributor \( i \). At each retailer node \( j \), retailer demand is denoted as \( v_j \), and at each distributor \( j \), its capacity is denoted as \( q_j \). Objective function (3.5) minimizes total travel distance, while objective function (3.6) minimizes total travel time between distributors and retailers allocated to them. Constraint (3.7) ensures that each retailer is assigned to one of the distributors. Constraint (3.8) ensures that the total demand of retailers does not exceed the capacity of distributors serving them. Constraint (3.9) is the integer constraint. The objective function of Gengui Zhou et al. (2003) is used for the problem.

3.2.2 Simulated Annealing with Population Size Initialization through Neighbourhood Generation (SAPING) for GAP and BORAPVC

Given the successful application of Simulated Annealing with Population Size Initialization through Neighbourhood Generation (SAPING) to solve multi-objective optimization and combinatorial optimization problems, we proposed and developed a simulated annealing with Population Size Initialization through Neighbourhood Generation (SAPING) to handle
both $GAP$ and $BORAPVC$. Ever since the $SAPING$ was introduced to handle combinatorial problems, it has emerged as one of the most efficient intensive solution search procedures for solving multi-objective optimization problems.

*A step wise description of SAPING is given below:

Step 1 Generate data and find the initial solution

Step 1.1 Generate random data

Step 1.2 Find the sorted matrix. The sorted matrix is the matrix which stores the position of cost or time data in ascending order.

The matrix is such that $[a]i0$ gives the position of smallest data item in the $i^{th}$ row, $[a]i1$ gives the next smallest data item.

Step 1.3 Find the min cost solutions.

Step 1.3.1 Repeat for number of elements in a sequence

Step 1.3.2 Do

Step 1.3.3 seq[i]=[$a$]iz

Step 1.3.4 If the demand of the $i^{th}$ city or customer exceeds capacity of warehouse given by the $z^{th}$ column of the sorted matrix, then $z++$ and end Do

Step 1.3.5 End repeat

Step 1.3.6 If $z$ is equal to number of warehouses, then display that there is no such solution.

Steps 1.3.7 similarly do for the time data.

Step 1.3.8 Store the initial solutions in seq1, seq2

Initialization: Set the initial temperature $t_{max}$ and cooling rate $\alpha$. Set the iteration number $ITER = 0$.

Step 2 Create initial populations

Problem Representation: Represent the solution string as follows
<table>
<thead>
<tr>
<th>Customer</th>
<th>6</th>
<th>3</th>
<th>7</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warehouse</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Neighbourhood Generation: From the structured initial solution $s_j$, generate the initial set of neighbourhood solutions $N(s_j)$ randomly. Generate the equivalent new feasible neighbourhood set $N(s_j)$ randomly for $N(s_j)$.

Step 3 Insert seq1 and seq2 into the initial population

Step 4 Start Simulated Annealing

Step 4.1 While $t_{\text{max}} < t_{\text{min}}$ or number of iterations exceed $ITER$ (A maximum iteration value defined by user)

Step 4.2 Find total cost and time for each sequence

Step 4.3 Find fitness

Step 4.3.1 Find min of each objective function (which is the value of objective function for the initial solutions)

Step 4.3.2 Find scaled objective function: $f(\text{minimum}) / (\text{actual})$

Evaluation: Calculate the objective function with respect to time and cost $g(s_j)$, $
\forall s_j \in \{1, ..., N\} \text{ Minimize } g(s_j)$ where $g(s_j) =$ Total allocation cost

Step 4.3.3 Find penalty for each sequence.

Step 4.3.3.1 Repeat till end of sequence

Step 4.3.3.2 If capacity of warehouse signified by $i^{\text{th}}$ value of sequence is less than the demand of corresponding customer or city then put a penalty

Step 4.3.3.3 End repeat

Step 4.3.3.4 return penalty value

Step 4.3.4 Find the overall objective function: Penalty / ($f1 * f2$)
Step 4.4 Start initializing the next generation.

Step 4.4.1 Repeat for each sequence in the population.

Step 4.4.2 If the sequence is a non-dominated solution with respect to the initial solutions provided, then accept it into next generation; else generate a neighbour of the sequence and depending on the entropy, put the corresponding sequence into next generation.

Step 4.4.3 End repeat.

Step 4.5 Calculate $t_{\text{max}} = t_{\text{max}} \times \alpha$.

Temperature Assignment: Calculate the maximum temperature $t_{\text{max}}$ for each neighbourhood $s_j$ as $t_{\text{max}}(S_j) = (t_{\text{max}}(S_j))^\alpha \alpha$ where $\alpha [\alpha \in [0,1]]$ is the cooling rate and $N_j$ is the parent set.

Step 4.6 Do the selection process.

Selection: Compare each objective function of $g(s_j)$ with corresponding $g(s_j)$,

If $g(s_j) \geq g(s_j)$,

Then, set $s_j' = s_j$.

i.e., $s_j$ goes to the parent set $N_j$.

Else, check for the following condition:

Is $\text{Random}(0,1) \leq e^{\frac{-\Delta s_j s_j'}{t_{\text{max}}(s_j')}}$,

[where $\Delta s_j s_j' = g(s_j) - g(s_j')$]

If so, set $s_j' = s_j$, i.e., place $s_j'$ in the parent set $N_j$.

Else, retain $s_j$ in the parent set $N_j$.

$t_{\text{max}}(s_j) = \alpha \times t_{\text{max}}(s_j)$;

After every iteration, increment $I$ by 1.

Step 4.6 End while. Termination: Stop, if the solution converges or if the maximum number of iterations has been reached. Else, go to Step 1.
Step 5 Get the non dominated set from the final population. SA used utilizes the preservation of parento front for the two criteria i.e. it preserves the solutions that generates the non dominating set with respect to the initial solution. The initial solutions, which are best solutions for either criterion, can be used to delineate the extent of search. While generating the neighbourhood for any sequence, the point mutation is used where the sequence at any random position is perturbed to form the new neighbourhood. The neighbourhood size is 5.

### 3.2.2.1 Parameter settings for SAPING

From the pilot study, the values of the parameters used in SAPING are selected as follows:

- Initial population = 200
- Initial (maximum) temperature \( (t_{max}) = 4500 \)
- Cooling rate / temperature reduction coefficient \( (\alpha) = 0.98 \)
- Maximum number of iterations = 500

### 3.3 COMPUTATIONAL EXPERIMENTS AND RESULTS

Extensive computational experiments are carried out to evaluate the performance of the SAPING method. Real data from (Melachrinoudis and Min 2000; Min and Melachrinoudis 1999) are used to test the proposed SAPING for the BORAPVC. Readers can refer (Zhou et al 2003) to know more details about the data sets. Although there are many measures in supply chain, the most critical measures such as cost and time are considered as two objectives. In practice, warehouse capacity may vary from one to another. Reflecting this reality, we tested the proposed SAPING for BORAPVC two scenarios: (1) Multiple warehouses with equal capacity (size); (2) Multiple warehouses with varying capacity. Under scenario (1), we set the warehouse capacity equal between the two (nearly one-third of total demand). Under scenario (2), we set
the warehouse capacity in such a way that warehouse 2 is approximately 10% larger than warehouse 1 and warehouse 3 is 10% larger than warehouse 2. However, notice that the total warehouse capacity for both scenarios is equal for comparative purposes. The proposed SAPING based on the two strategies generated various Pareto solutions and compared with the number of Pareto solutions generated by Genetic Algorithm (GA) proposed by (Zhou et al 2003). The minimum cost and time solution for SAPING, the number of non-dominated solutions generated by SAPING and GA is summarized in Table 3.1. 11 random data sets are generated and the minimum cost and time solution for SAPING, the number of non-dominated solutions generated by SAPING and GA is summarized in Table 3.2. From the results of Table 3.1 It is clear that SAPING outperforms GA in creation of number of non-dominated solutions which is very useful for decision-makers for making operational and tactical decisions.
Table 3.1 Comparison of *SAPING* and *GA* for *BORAPVC* for benchmark data sets

<table>
<thead>
<tr>
<th>Type of Problem</th>
<th>Minimum Cost Solution by <em>SAPING</em></th>
<th>Minimum Time Solution by <em>SAPING</em></th>
<th>Number of Solutions by <em>SAPING</em></th>
<th>Number of Solutions by <em>GA</em></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Weight 1</td>
<td>Weight 2</td>
<td>Weight 1</td>
<td>Weight 2</td>
</tr>
<tr>
<td>Calculated weights and equal capacities</td>
<td>7040020</td>
<td>122</td>
<td>7117630</td>
<td>110</td>
</tr>
<tr>
<td>Calculated weights and Unequal capacities</td>
<td>7078750</td>
<td>122</td>
<td>7121580</td>
<td>109</td>
</tr>
<tr>
<td>Random weights and equal capacities</td>
<td>7013370</td>
<td>116</td>
<td>7323060</td>
<td>110</td>
</tr>
<tr>
<td>Random weights and Unequal capacities</td>
<td>6917000</td>
<td>114</td>
<td>7390450</td>
<td>109</td>
</tr>
<tr>
<td>Size of Problem</td>
<td>Minimum Cost Solution by SAPING</td>
<td>Minimum Time Solution by SAPING</td>
<td>Number of Solutions by SAPING</td>
<td>CPU by SAPING</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------------------------</td>
<td>----------------------------------</td>
<td>-------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td></td>
<td>Weight 1</td>
<td>Weight 2</td>
<td>Weight 1</td>
<td>Weight 2</td>
</tr>
<tr>
<td>120x30</td>
<td>4.73E+07</td>
<td>1135.71</td>
<td>7.83E+07</td>
<td>456.562</td>
</tr>
<tr>
<td>120x10</td>
<td>4.99E+07</td>
<td>1145.86</td>
<td>8.95E+07</td>
<td>464.139</td>
</tr>
<tr>
<td>100x50</td>
<td>4.81E+07</td>
<td>1041.12</td>
<td>8.20E+07</td>
<td>463.299</td>
</tr>
<tr>
<td>100x30</td>
<td>4.12E+07</td>
<td>989.035</td>
<td>7.17E+07</td>
<td>420.122</td>
</tr>
<tr>
<td>100x10</td>
<td>4.17E+07</td>
<td>1008.71</td>
<td>7.41E+07</td>
<td>413.366</td>
</tr>
<tr>
<td>80x50</td>
<td>3.95E+07</td>
<td>982.414</td>
<td>7.08E+07</td>
<td>428.105</td>
</tr>
<tr>
<td>80x30</td>
<td>3.78E+07</td>
<td>896.582</td>
<td>6.48E+07</td>
<td>377.527</td>
</tr>
<tr>
<td>80x10</td>
<td>3.93E+07</td>
<td>884.097</td>
<td>6.99E+07</td>
<td>370.129</td>
</tr>
<tr>
<td>60x50</td>
<td>3.89E+07</td>
<td>919.244</td>
<td>6.94E+07</td>
<td>372.688</td>
</tr>
<tr>
<td>60x30</td>
<td>3.38E+07</td>
<td>838.248</td>
<td>5.65E+07</td>
<td>318.973</td>
</tr>
<tr>
<td>60x10</td>
<td>3.28E+07</td>
<td>766.877</td>
<td>5.50E+07</td>
<td>326.062</td>
</tr>
<tr>
<td>120x30</td>
<td>3.26E+07</td>
<td>732.742</td>
<td>5.17E+07</td>
<td>328.443</td>
</tr>
<tr>
<td>120x10</td>
<td>2.88E+07</td>
<td>621.375</td>
<td>4.79E+07</td>
<td>287.623</td>
</tr>
<tr>
<td>100x50</td>
<td>2.69E+07</td>
<td>665.631</td>
<td>4.61E+07</td>
<td>290.693</td>
</tr>
<tr>
<td>100x30</td>
<td>3.38E+07</td>
<td>629.876</td>
<td>4.79E+07</td>
<td>335.611</td>
</tr>
<tr>
<td>100x10</td>
<td>2.47E+07</td>
<td>534.967</td>
<td>4.22E+07</td>
<td>238.618</td>
</tr>
<tr>
<td>80x50</td>
<td>2.50E+07</td>
<td>590.224</td>
<td>4.59E+07</td>
<td>247.453</td>
</tr>
<tr>
<td>80x30</td>
<td>3.07E+07</td>
<td>553.172</td>
<td>4.79E+07</td>
<td>286.13</td>
</tr>
<tr>
<td>80x10</td>
<td>2.19E+07</td>
<td>431.235</td>
<td>3.75E+07</td>
<td>191.335</td>
</tr>
<tr>
<td>60x50</td>
<td>2.01E+07</td>
<td>436.159</td>
<td>3.25E+07</td>
<td>194.87</td>
</tr>
<tr>
<td>60x30</td>
<td>2.39E+07</td>
<td>470.044</td>
<td>3.59E+07</td>
<td>235.698</td>
</tr>
<tr>
<td>60x10</td>
<td>1.46E+07</td>
<td>320.927</td>
<td>2.32E+07</td>
<td>145.39</td>
</tr>
<tr>
<td>60x30</td>
<td>1.50E+07</td>
<td>278.131</td>
<td>2.57E+07</td>
<td>147.792</td>
</tr>
<tr>
<td>60x10</td>
<td>1.99E+07</td>
<td>345.126</td>
<td>3.08E+07</td>
<td>174.965</td>
</tr>
</tbody>
</table>
3.4 CONCLUSION

This chapter has addressed two variants GAP and BORAPVC. For this Non-deterministic Polynomial (NP)-hard problem, we have developed mathematical programming model and unified heuristic SAPING as solution methodology to solve BORAPVC where the assumption that customers are served by warehouses with equal capacity was relaxed. Two conflicting objectives are taken into account (transit time versus cost) which involves the warehouse allocation problem. To deal with two objectives simultaneously, we employed two distinctive weight generation strategies that enabled the decision maker to evaluate a greater number of potential alternative solutions using SAPING. Our experiments showed the tradeoff between total transit time and total cost. The number of non-dominated solutions generated has been compared with GA. The SAPING heuristic is tested for standard benchmark datasets of GAP and randomly generated datasets of BORAPVC. The sizes of the randomly generated datasets are ranging from 60x10 to 200x70. When compared the results SAPING with the outcome of GA for BORAPVC data sets, SAPING has generated more non-dominated solutions with less computational time. The heuristic SAPING was tested using publicly available sets of benchmark problem for GAP and performs better than the existing results. The future scope also includes development of analyst’s toolkit for finding quick and effective solutions and can be embedded into Decision Support Systems (DSS). The incorporation of “What If” rules in a DSS along with the software for the heuristic is a potential way forward.