International Journal of Production Research
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tprs20

Genetic-algorithms-based algorithm portfolio for inventory routing problem with stochastic demand

Nagesh Shukla [a b], M.K. Tiwari [c] & Darek Ceglarek [a d]

[a] International Digital Laboratory, WMG, University of Warwick, Coventry, UK

[b] SMART Infrastructure Facility, University of Wollongong, Wollongong, Australia

[c] Department of Industrial Engineering & Management, Indian Institute of Technology, Kharagpur, India

[d] Department of Industrial & Systems Engineering, University of Wisconsin-Madison, Madison, USA

Published online: 20 Feb 2012.

To cite this article: Nagesh Shukla , M.K. Tiwari & Darek Ceglarek (2013): Genetic-algorithms-based algorithm portfolio for inventory routing problem with stochastic demand, International Journal of Production Research, 51:1, 118-137

To link to this article: http://dx.doi.org/10.1080/00207543.2011.653010

International Journal of Production Research
Vol. 51, No. 1, 1 January 2013, 118–137

Taylor & Francis
Taylor & Francis Group

Genetic-algorithms-based algorithm portfolio for inventory routing problem with stochastic demand

Nagesh Shukla[ab]*, M.K. Tiwari[c] and Darek Ceglarek[ad]

[a]*International Digital Laboratory, WMG, University of Warwick, Coventry, UK;* [b]*SMART Infrastructure Facility, University of Wollongong, Wollongong, Australia;* [c]*Department of Industrial Engineering & Management, Indian Institute of Technology, Kharagpur, India;* [d]*Department of Industrial & Systems Engineering, University of Wisconsin-Madison, Madison, USA*

(*Received 4 July 2011; final version received 19 December 2011*)

This paper presents an algorithm portfolio methodology based on evolutionary algorithms to solve complex dynamic optimisation problems. These problems are known to have computationally complex objective functions, which make their solutions computationally hard to find, when problem instances of large dimensions are considered. This is due to the inability of the algorithms to provide an optimal or near-optimal solution within an allocated time interval. Therefore, this paper employs a bundle of evolutionary algorithms (EAs) tied together with several processors, known as an algorithm portfolio, to solve a complex optimisation problem such as the inventory routing problem (IRP) with stochastic demands. EAs considered for algorithm portfolios are the genetic algorithm and its four variants such as the memetic algorithm, genetic algorithm with chromosome differentiation, age-genetic algorithm, and gender-specific genetic algorithm. In order to illustrate the applicability of the proposed methodology, a generic method for algorithm portfolios design, evaluation, and analysis is discussed in detail. Experiments were performed on varying dimensions of IRP instances to validate different properties of algorithm portfolio. A case study was conducted to illustrate that the set of EAs allocated to a certain number of processors performed better than their individual counterparts.

Keywords: genetic algorithms; algorithm portfolios; inventory routing; stochastic demand

1. Introduction

There are several NP hard problems in the areas of graph, scheduling, and coding theory for which the computationally efficient solution has not been found or shown to be non-existent (Stinson 1987). This uncharacteristic behaviour is due to the stochastic nature of problem size and complexity. Furthermore, the nonlinear characteristics of the objective function of NP hard problems also contribute to increasing computational complexity. The common examples of the non-deterministic engineering problems with uncertainties include stochastic vehicle routing problems (Yang *et al*. 2000), inventory routing problems (Aghezzaf *et al*. 2006, Shen *et al*. 2007), travelling salesman problems (TSP) of varying sizes (de Berg *et al*. 2005), and lot-sizing problems with stochastic demands (Raa and Aghezzaf 2005).

To solve the aforementioned problems with stochastic complexity, the solution methodology must be capable of providing optimal or suboptimal solution to the problems within the allocated time frame. In other words, the performance of the solution methodology can be characterised by (1) the quality of the solution it provides; and (2) the computational time it takes. There are various methods used in the literature to obtain an optimal solution for smaller instances of the problem such as using simplex algorithms based on linear programming (LP) formulations and dynamic programming (DP) based optimisation approaches (Taha 2006). However, when a problem of significant size is solved by the mathematical programming approaches, the computational time becomes huge. As a result, the use of mathematical programming approaches in practical applications is restricted.

In order to reduce the computational time involved to solve complex problems with a stochastic size, researchers have often applied evolutionary algorithms (EAs), which use mechanisms inspired by biological evolution, reproduction, mutation, recombination, and selection. EAs have comprehensively reduced the computational time

*Corresponding author. Email: nshukla@uow.edu.au

ISSN 0020–7543 print/ISSN 1366–588X online

http://dx.doi.org/10.1080/00207543.2011.653010
http://www.tandfonline.com

taken to resolve NP-hard problems without much depreciation in the solution quality. In literature, different types of EAs have been extensively used for solving complex problems of different domains. However, the performance of EAs depends heavily upon random numbers generated during the course of their run, which causes large variations in the quality of the solution obtained using an EA. As a result, EAs, when used individually, may provide an optimal or near-optimal solution for some instances of a particular problem. However, the use of individual EAs cannot be considered as a best-performing approach for an entire range of instances of a particular optimisation problem. Therefore, this paper details a novel concept of algorithm portfolios, which can be used to resolve computationally hard problems with stochastic nature (varying sizes). The concept of algorithm portfolios is first proposed by Gomes and Selman (2001) for combinatorial complex problems. This paper presents a generic applicability and statistical background to validate the concept of algorithm portfolios based on EAs.

An algorithm portfolio can be defined as '*a collection of different algorithms or different copies of the same algorithm running on different processors*' (Rice 1976, Brown *et al.* 2003). The need for constructing algorithm portfolios is due to the large variations in the performance of EAs for the same type of problems with varying sizes and complexities. Users can be benefited by such variations, by combining several EAs to form algorithm portfolios, and running them in parallel or interleaving them on single processor. The main contribution of this paper is to introduce algorithm portfolios based on EAs to solve complex problems with stochastic problem sizes. The EAs are used to form an algorithm portfolio for reducing the computational time required in obtaining near optimal solutions. Moreover, this strategy is fruitful in situations where one has to find the best solutions (not necessarily optimal solutions) to a particular problem within the stipulated time frame. Various types of EAs are available in the literature for optimisation. The most widely used and easily applicable representatives of EAs are genetic algorithms (GAs) (Holland 1975, Goldberg 1989). GAs were developed based on the Darwinian principle of *survival of the fittest* and the natural process of evolution through reproduction. Based on its effective and efficient searching ability, GAs have been widely used in the literature (Ghosh *et al.* 1996, Gen and Cheng 1997, Merz and Freisleben 1997). Therefore, the present study employs GA and its four variants for constructing algorithm portfolios. These four variants are: (1) the age-genetic algorithm (AGA) (Ghosh *et al.* 1996), (2) memetic algorithm (MA) (Dawkins 1976), (3) genetic algorithm with chromosome differentiations (GACD) (Singh *et al.* 2006), and (4) gender-specific (sexual) genetic algorithms (SGA) (Goh *et al.* 2003, Wagner and Affenzeller 2005). GA and its four variants are utilised in this study to illustrate the basic mechanism and criteria for: design, evaluation, analysis, and performance of the algorithm portfolios. In order to show the applicability of the algorithm portfolios, this paper takes up the problem of inventory routing for stochastically demanded products in the supply chain. The inventory routing problem (IRP) is a challenging NP-hard problem. It deals with the issue of coordinating inventory replenishment policies and distribution plans in a cost-effective manner. In addition, the routing policy is adopted for products whereby the number of sales points is not fixed and can increase/decrease with time, which contributes to stochastic problem instance sizes in IRP. The IRP model for high-consumption products have been proposed earlier by Aghezzaf *et al.* (2006), but they have assumed the number of sales points to be fixed and demands of the products to be deterministic. Although this model may seem to be correct for deterministic scenarios (fixed demand and sales points), the market for products is stochastic, i.e. the demand rate and number of sales points vary considerably. This led us to use stochastic scenarios for solving IRP where the decision-maker has to find the best solutions to an IRP instance with stochastic sizes and demands in a minimum amount of time. Moreover, EA-based search techniques have been rarely applied on IRP. Therefore, it also becomes necessary to test EAs on IRPs.

A comprehensive and rigorous experimentation has been conducted in this paper to evaluate the performance of various algorithm portfolios. The best algorithm portfolio cases corresponding to a set of processors are determined using Analytical Hierarchy Process (Winkler 1990). Moreover, selection of the best-performing algorithm within the algorithm portfolio for a given optimisation problem is obtained using the proposed hamming distance-based algorithm selection procedure. In addition, a comparative analysis of the best algorithm portfolio cases and the best-performing constituent algorithm has been carried out over 20 IRP problem instances with varying sizes and complexity.

The remainder of the paper is organised as follows. In Section 2, the need for algorithm portfolios, the inventory routing problem, a brief description about evolutionary algorithms considered, and the statistical validation of the proposed concept are described. The design, evaluation, and analysis of algorithm portfolios over IRP instances are detailed in Section 3. Finally, Section 4 concludes the paper with a discussion on its future research directions.

2. Algorithm portfolios

2.1 *Need for algorithm portfolios*

Although some EAs can perform better than others, it is not possible to declare a best algorithm for a given type of optimisation problem. This is because different instances of an optimisation problem are generally associated with varying complexity, and in general NP-hard problems have more pronounced variations of their run times and the quality of solution obtained by algorithms (Brown *et al*. 2003). Therefore, selection of the best algorithm for a given optimisation problem involves a difficult decision-making process. Initial research in this field started in the 1970s, when Rice characterised this problem as an algorithm selection problem and provided a provisional solution (Rice 1976). Over the years, the research community has focused on developing algorithms for effective and efficient optimisation rather than concentrating on selecting algorithms for optimisation. In recent years, few research studies have considered algorithm selection problems, thereby producing some literature for researchers and practitioners (Rice 1976, Lagoudakis and Littman 2000, Gomes and Selman 2001, Brown *et al*. 2003). Huberman *et al*. (1997) have proposed a general method for combining existing algorithms to develop a new algorithm that always performs better than the individual constituent algorithm. Furthermore, an ensemble of discrete differential evolution algorithms with parallel populations have been explored to illustrate its effectiveness over individual best-performing algorithms (Song *et al*. 2009, Tasgetiren *et al*. 2010). Although these approaches highlight the need for development of interlinked algorithms, the use of multiple processors in an industrial settings has rarely been discussed. Therefore, this paper discusses the development of algorithm portfolios, which employs a competing set of EAs together with multiple processors to solve complex optimisation algorithms.

In most of the current EAs applications, the algorithm performing best on a given optimisation problem is regarded as the best strategy based on the principle of '*winner-takes-all*' (Brown *et al*. 2003). The drawback with the *winner-takes-all* strategy is the fact that a selected algorithm may show a better average performance for a given problem instance; however, it does not provide a better solution for other instances of the same optimisation problem with varying sizes and complexities. Therefore, within the purview of these shortcomings, the winner-takes-all strategy is not suitable for use by decision-makers, who work in different scenarios and have to face problems of varying dimensions and complexities. Thus, the selection of the best algorithm to obtain the optimised solution to a problem is crucial. This has motivated authors to develop algorithm portfolios by combining different EAs to gain a viable and insightful perspective to find an optimised solution.

The algorithm portfolios can be considered as a bundle of algorithms tied together and running in parallel on multiple processors without any interactions between them. The concept of running different algorithms on multiple processors improves the performance of the algorithms in terms of expected computational time and quality of the solution obtained, thereby minimising the overall risk associated when using a single algorithm on a single processor. The following subsection details the inventory routing problem for high-consumption products and the instances explored having a different size and complexity.

2.2 *Problem description and instances explored*

The inventory routing problem (IRP) is an NP-hard problem that integrates the inventory levels and the product distribution characteristics in the same planning problem. Owing to the presence of stochastic number of the sales points and their demand rates, the IRP problem becomes even more challenging. The stochastic nature of these IRPs is due to the high demands of particular product in the market. It deals with developing the cyclical distribution plan from a single distribution centre 'c' to a set of sales-points $\mathbf{P}$ for a particular product. The consumption or demand rate of the ith sales point is d_i (where $i \in \mathbf{P}$) units (tons) per hour and is capable of maintaining a small or local inventory. A set of homogeneous vehicles $\mathbf{V}$, each having the capacity to distribute 'q' number of products in a single load is available for distribution. The overall objective of the problem is to minimise the expected distribution and inventory costs without causing stock-out at any of the sales points. The two major decisions that have to be made while solving the inventory routing problems are related to: (1) the frequency with which each sales point should be served and (2) the determination of the routes to be used. Needless to say, these objectives are interlinked with each other and cannot be optimised individually.

Aghezzaf *et al*. (2006) have described the IRP model for highly demanded goods assuming a fixed number of sales point and their demand. However, owing to an ever-changing market demand of products IRP, stochastic conditions such as sales-points demand and a varying number of sales points is introduced in IRP. The IRP model considers the concept of vehicle multi-tours. Multi-tours of a vehicle represents that a vehicle can take multiple tours

to and from distribution centres to replenish different sales-points as opposed to single tour or route where a vehicle can make only one tour distributing products from a distribution centre to sales points. The attributes that play a crucial role in understanding the concept of multi-toured vehicles and their cycle times are illustrated as follows:

- Cycle times: The time between two consecutive iterations of the tour of a vehicle v is called *cycle time*, and it is denoted by T^v_{cycle}. The cycle time has its own lower and upper bounds, i.e., T^v_{cycle} as determined by the travel time owing to limited vehicle capacity and travelling time associated with the traveling salesman tour (TSP-tour).
- Minimal cycle time $T^{\min}_{\text{cycle}}$: The minimal cycle time is obtained when a vehicle tour is made to replenish a set of sales-points by travelling along the TSP-tour. In the case where multi-tours of a vehicle is allowed, $T^{\min}_{\text{cycle}} = \sum_{i=1}^{r} T^{\min}_{\text{cycle}_i}$, where r is the number of tours allowed to the vehicle.
- Maximal cycle time $T^{\max}_{\text{cycle}}$: There also exists the maximal cycle time that results from limited capacity of a vehicle, i.e. the maximum capacity that can be distributed in a particular tour must not be more than $\sum_{i\in\mathbf{P}} d_i$ (cumulative demand rate of the sales points in a particular tour) where P is the subset of $\mathbf{P}$ that a vehicle covers in a particular tour. Therefore, the maximal cycle time in the tour is equal to the vehicle capacity divided by the cumulative demand rate of the sales points in a particular tour, i.e. $T^{\max}_{\text{cycle}} = \frac{q}{\sum_{i\in P} d_i}$, where P is the subset of $\mathbf{P}$ that a vehicle covers. In the case of multi-tours vehicles, $T^{\max}_{\text{cycle}} = \min_{i\ldots r}\{T^{\max}_{\text{cycle}_i}\}$, where r is the number of tours allowed to the vehicle. For a tour to be feasible, it is necessary to have $T^{\max}_{\text{cycle}} \geq T^{\min}_{\text{cycle}}$, as sales-points demands in a tour will be greater than the amount that is supplied in the consecutive tours.

The notations used in the model formulation are provided next for easy illustration:

s	is the travelling speed (miles/hour) of the vehicle;
φ^v	is the fixed cost for using vehicle v;
λ^v	is the travelling cost of a vehicle v per Mile;
ψ_i	is the cost per delivery at sales point i;
k_i	is the holding cost per ton per hour at the sales point i;
t_{ij}	is the time taken to travel from sales point $i \in \mathbf{P}^+$ to sales point $j \in \mathbf{P}^+$ (where, $\mathbf{P}^+ = \mathbf{P} + \{c\}$ and $\{c\}$ represents distribution centre);
$\mathbf{X}^v$	is the binary variable that equals 1 if vehicle $v \in \mathbf{V}$ is being used and 0 otherwise;
Y^v_{ij}	is the binary variable that equals 1 if sales point $j \in \mathbf{P}^+$ is served immediately after sales point $i \in \mathbf{P}^+$ by vehicle $v \in \mathbf{V}$ and 0 otherwise;
Z^v_{ij}	represents the sum of demand rates of remaining sales points in a tour covered by vehicle $v \in \mathbf{V}$ when it travels to the sales point $j \in \mathbf{P}^+$ immediately after it has served sales point $i \in \mathbf{P}^+$.

The objective function of the model comprises four different cost components for each multi-tour, and these are described below:

- Fixed operating cost: The cost incurred when a vehicle $v \in \mathbf{V}$ is in use and is denoted by ϕ^v per hour.
- Transportation cost: The movement of vehicle from one place to another is associated with the cost that varies with the distance travelled. The transportation cost is defined as $\lambda^v \cdot s \cdot T^{\min}_{\text{cycle}}$ per iteration of the multi-tour of vehicle $v \in \mathbf{V}$. Therefore the transportation cost rate is $\lambda^v \cdot s \cdot T^{\min}_{\text{cycle}}/T_{\text{cycle}}$ per hour.
- Delivery handling cost: The delivery handling cost at the sales-points is given by $\sum_{i\in\mathbf{P}} \psi_i$ per cycle. The delivery handling rate is given by $\sum_{i\in\mathbf{P}} \psi_i / T^v_{\text{cycle}}$ per hour.
- Stock holding cost: The quantity delivered at the sales-point i is just enough to cover the demand until next delivery. Therefore, the average stock level during the T^v_{cycle} hours at the sales point i is $d_i \cdot T^v_{\text{cycle}}/2$ units. Hence, the stock holding cost per cycle at sales point i is given by $d_i \cdot (T^v_{\text{cycle}})^2/2$. The total stock holding cost for the multi-tour is thus $(T^v_{\text{cycle}})^2 \sum_{i\in\mathbf{P}} (\kappa_i \cdot d_i/2)$ per cycle, or the holding cost rate is $T^v_{\text{cycle}} \sum_{i\in\mathbf{P}} (\kappa_i \cdot d_i/2)$per hour.

In a case when the demand rates of sales-points are stochastic owing to a changing market, the IRP is therefore modelled as a short-term planning problem (i.e. in which the same optimal route is followed for a short duration).

This problem is mathematically modelled with the help of nonlinear mixed integer formulation, which is defined as the following set of equations.

$$OBJ = \sum_{v \in \mathbf{V}} \left[\phi^v \mathbf{X}^v + \frac{1}{T^v_{\text{cycle}}} \cdot \sum_{i \in \mathbf{P}^+} \sum_{j \in \mathbf{P}^+} s\lambda^v t_{ij} Y^v_{ij} + \sum_{i \in \mathbf{P}^+} \left(\psi_i \frac{1}{T^v_{\text{cycle}}} + \frac{1}{2} k_i d_i T^v_{\text{cycle}} \right) \cdot \sum_{j \in \mathbf{P}^+} Y^v_{ij} \right] \tag{1}$$

subject to:

$$\sum_{v \in \mathbf{V}} \sum_{i \in \mathbf{P}^+} Y^v_{ij} = 1 \quad \forall j \in \mathbf{P} \tag{2}$$

$$\sum_{i \in \mathbf{P}^+} Y^v_{ij} - \sum_{k \in \mathbf{P}^+} Y^v_{ik} = 0 \quad \forall v \in \mathbf{V},\ j \in \mathbf{P}^+ \tag{3}$$

$$\sum_{i \in \mathbf{P}^+} \sum_{j \in \mathbf{P}^+} t_{ij} Y^v_{ij} - T^v_{\text{cycle}} \leq 0 \quad \forall v \in \mathbf{V} \tag{4}$$

$$\sum_{v \in \mathbf{V}} \sum_{i \in \mathbf{P}^+} Z^v_{ij} - \sum_{v \in \mathbf{V}} \sum_{k \in \mathbf{P}^+} Z^v_{ik} = d_j \quad \forall j \in \mathbf{P} \tag{5}$$

$$Y^v_{cj} - X^v \leq 0 \quad \forall v \in \mathbf{V},\ j \in \mathbf{P} \tag{6}$$

$$T^v_{\text{cycle}} Z^v_{cj} \leq q^v \quad \forall v \in \mathbf{V},\ j \in \mathbf{P} \tag{7}$$

$$Z^v_{ij} - \left(\sum_{k \in \mathbf{P}} d_k \right) Y^v_{ij} \leq 0 \quad \forall v \in \mathbf{V},\ i,j \in \mathbf{P}^+ \tag{8}$$

$$X^v \in \{0,1\}, T^v_{\text{cycle}} \geq 0 \quad \forall v \in \mathbf{V},\ i,j \in \mathbf{P}^+ \tag{9}$$

$$Y^v_{ij} \in \{0,1\}, \quad Z^v_{ij} \geq 0 \quad \forall v \in \mathbf{V},\ i,j \in \mathbf{P}^+. \tag{10}$$

Equation (2) represents the service of only one vehicle for each sales-point. The usual flow conservation constraints of the vehicle are shown by Equation (3). Equation (4) indicates that the cycle time of the vehicle should be greater than the total time it has to travel. The constraint represented in Equation (5) describes the scenario where the cumulative demand rates of the remaining sales point in the tour carried by the vehicle $v \in \mathbf{V}$ serving sales point $j \in \mathbf{P}$ are reduced by demand rates d_j when the vehicle leaves this sales point. Equation (6) states that a vehicle can only leave the distribution centre c when it is being used. Equation 7 represents the capacity restrictions over vehicle. Equation (8) states that Y^v_{ij} cannot carry any cumulated demand rates unless Z^v_{ij} equals 1. X^v and Y^v_{ij} are binary decision variables, which can take only one of two values $\{0,1\}$. T^v_{cycle} and Z^v_{ij} are two quantities representing the time and sum of demand rates and, hence, cannot be negative.

The IRP literature does not support any commonly used benchmark dataset, so in this study we have generated a set of IRP instances of varying sizes and complexity based on the following parameters:

- number of sales-points ($\mathbf{P}$) considered;
- number of homogeneous vehicles ($\mathbf{V}$) in the fleet;
- different demand rates (d_i (where $i \in \mathbf{P}$)).

These factors are crucial in determining the size (i.e. number of sales points $|\mathbf{P}|$) and complexity (number of homogeneous vehicles $|\mathbf{V}|$, and demand rates d_i) of IRP. Based on the above-mentioned factors, 20 different instances of IRP have been generated. The instances are divided into three major groups, i.e. small, medium, and large, based on the number of sales points ($|\mathbf{P}|$), number of homogeneous vehicles ($|\mathbf{V}|$), and demand rates (d_i). The description of IRP instances within these categories is illustrated in Table 1. The size of the IRP instance is

Table 1. Parameters used for generating IRP instances of varying sizes and demands.

Parameters of IRP instances	Size of IRP instance		
	Small	Medium	Large
No. of instances	7	7	6
No. of sales points	5–35	65–100	165–200
Vehicles	1–5	8–13	20–25
Type of sales points location	Geog. + Eucl.	Geog. + Eucl.	Geog. + Eucl.
Vehicle speed (mile/h)	50	50	50
Vehicle capacity (ton)	100	100	100
Fixed operating cost (/h)	50	50	50
Travelling cost (/mile)	1	1	1
Demand rate (ton/h)	0.1–1.5	0.5–2.5	1.0–3.5
Inventory holding cost (/ton-h)	10	10	10
Delivery handling cost	30–50	30–50	30–50

Note: Eucl., Euclidean (i.e., Cartesian coordinates); Geog., geographical (i.e., latitude and longitude).

determined by the number of sales points ($|\mathbf{P}|$), and complexity is determined by the number of homogeneous vehicles ($|\mathbf{V}|$) and demand rates (d_i). These IRP instances are solved by integrating them with the selected portfolios in order to minimise the overall risk associated with the computational cost.

2.3 *Algorithm portfolio description*

The algorithm portfolios are designed in such a way that minimises the expected risk or computational time (as determined by the number of objective function evaluations) in obtaining the solution. This paper utilises the powerful searching ability of EAs to solve the IRP. In particular, EAs such as the Genetic Algorithm (GA) have been used in this paper to resolve the inventory routing problem (IRP). Along with GA, four other variants have been employed to make the portfolio. These GA variants are: age genetic algorithm (AGA), gender-specific genetic algorithm (also known as sexual GA or SGA), genetic algorithm with chromosome differentiation (GACD), and MA. These EAs are allowed to solve the IRP problem instance in each groups, i.e. small, medium, and large. Then, EAs competing with each other or having comparable performances are combined to form algorithm portfolios. There exist situations when the decision-maker wants to have good solution quickly rather than always getting the best solution. Therefore, it is imperative to design a robust solution methodology that is able to provide a good solution (not necessarily optimal) within the allotted time frame. The following text provides a short description about the GA and its four variants.

2.4 *Algorithmic suit*

This section provides the details of the implementation aspect of the EAs and a brief description about the working of these algorithms. EAs such as GA and its four variants require a similar representation scheme because they are all based on the theory of population genetics and the principle of *survival of the fittest*. In the interest of brevity and to avoid rephrasing the literature, the following discussion presents a short overview of the aforementioned five algorithms. The performance of these EAs on IRP instance is illustrated and discussed in detail in Section 3.2. The representation scheme is detailed as follows.

2.4.1 *Representation scheme*

The solution string is generated by taking into account the number of the sales-point, number of vehicles and their trips. A cyclic travelling salesman problem (TSP) string is formed by taking several dummy nodes of the distribution centre. Each dummy node signifies the route taken up by a particular vehicle trip. The formation of solution string is presented in Figure 1.

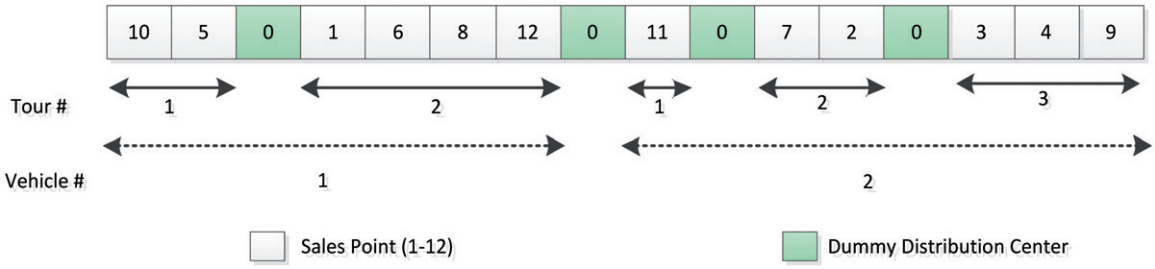


Figure 1. Representation scheme for the chromosome (solution string) for an IRP instance with 12 sales points, and two vehicles.

A total of five EAs have been considered for constructing algorithm portfolios. The following text presents a brief description about the working of GA and its four variants.

2.4.2 *GA*

GA is an adaptive method to solve search and optimisation problems, based on the principles of natural evolution (Holland 1975, Goldberg 1989). A solution to a given problem is represented in the form of a string, called chromosome, consisting of a set of elements, called genes that hold a set of values for the optimisation variables (Goldberg 1989). GA work with a random population of solutions (chromosomes). The fitness of each chromosome is determined by evaluating it against an objective function. To simulate the process of survival of the fittest, best chromosomes exchange information (through crossover and mutation) to produce the offspring chromosomes. For the inventory routing problem, partially matched crossover (PMX) has been used to retain the feasibility of the chromosome (see (Gen and Cheng 1997) for more about PMX operator). The offspring population members are then evaluated and are used to evolve the population if they provide better solutions. This process is repeated for large number of generations to obtain a best individual. The pseudo-code for the GA is provided in Figure 2.

2.4.3 *AGA*

To increase the search capabilities of GA, the concept of age of an individual (solution string) is introduced in GAs. As soon as a new individual (solution string) is generated in a population its age is assumed to be zero. After each generation, the age factor of each individual is increased by one. As in natural genetic system, young and old individuals are assumed to be less fit compared with adult individuals (Ghosh *et al*. 1996).

The effective fitness of an individual at any iteration is measured by considering not only the objective function value (Equation (1), but also including the effect of its age. The modified GA with age component is known as AGA (Ghosh *et al*. 1996). In GA, once a particular individual becomes more fit, it goes on getting chances to produce offspring until the end of the algorithm, if a proportional selection is used. This procedure increases the chance of generating similar type of offspring in GA. Therefore, more fit individuals are retained, and individuals with lesser fitness value are removed. Whereas in AGA, fitness of individuals with respect to age is assumed to increase gradually up to a pre-defined upper age limit (number of iterations), and then gradually decreases. This ensures a natural death for each individual keeping its offspring alive. Thus, in this case, a particular individual cannot dominate for a longer period of time in the algorithm. Rest of the process of evolution in AGA is same as that in GA. The main advantage of AGA is that it promotes more exploration in search space by associating age factor to each individuals and removing them after certain number of iterations. The pseudo-code for AGA is provided in Figure 2.

2.4.4 *MA*

MAs are inspired by the notions of a meme (Dawkins 1976). Here, the chromosomes are formed by the memes not genes (as in conventional GA). The unique aspects of the MA algorithm are that all chromosomes and offspring are allowed to gain some experience (i.e. promoting local search) before being involved in the process of evolution. The experience of the chromosomes is simulated by incorporating a local search operation. Merz and Freisleben (1997) proposed a method to perform a local search through a pairwise interchange heuristic. The local neighbourhood

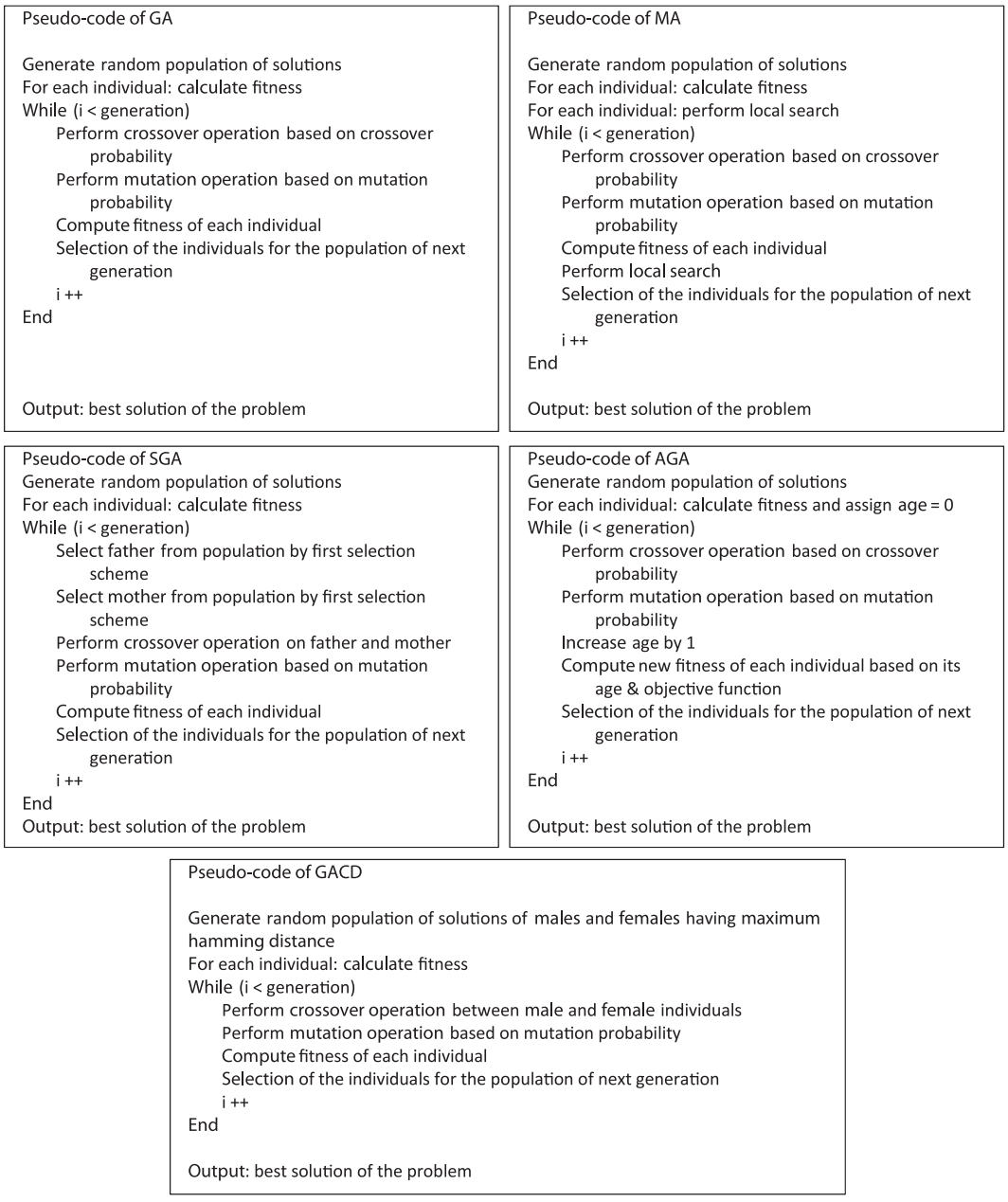


Figure 2. Pseudo-code of all the five EAs.

search is defined as a set of all solutions that can be reached from the current solution by swapping two elements in the chromosome. For a chromosome of length n, the neighbourhood size for the local search is $N = \frac{1}{2}n(n-1)$. This helps MA to search locally for optimal solutions in the solution search space. A pseudo-code for an MA procedure is given in Figure 2.

2.4.5 *SGA*

The selection of parent chromosomes for reproduction, in the case of GA, is done using only one selection strategy such as a roulette wheel and tournament selection. When considering the model of gender-specific selection in the area of population genetics, it becomes obvious that the process of choosing mating partners in natural populations is different for male and female individuals. Inspired by the idea of male vigour and female choice, Wagner and

Afenzeller (2005) have proposed SGA that utilises two different selection strategies for the selection of two parents required for the crossover. The first type of selection scheme utilises random selection, and another selection strategy uses roulette-wheel selection for the selection of two parents. This helps SGA to diversely explore the solution search space. The rest of the process is similar to that of GA. The pseudo-code for SGA is provided in the Figure 2.

2.4.6 *GACD*

In GACD, the population is divided into male and female populations on the basis of sexual differentiation (Singh *et al*. 2006). In addition, these populations are made artificially dissimilar, and both populations are generated in a way that maximises the hamming distance between the two classes. The crossover is only allowed between individuals belonging to two distinct populations, thus introducing a greater degree of diversity and simultaneously leading to greater exploration in the search space. Selection is applied over the entire population, which serves to exploit the information gained so far. Thus, GACD accomplishes a greater equilibrium between exploration and exploitation, which is one of the main features for any adaptive system. For more details about implementation aspects, pseudo-code for the GACD has been provided in Figure 2. The chromosomes in the case of GACD are different, as it contains an additional gene that helps in determining the sex of the individuals in the current population.

The aforementioned EAs (GA and GA variants) are employed in this paper for building algorithm portfolios for effective optimisation of IRP. The following section details the performance measures of algorithm portfolios based on their computational efficiencies.

2.5 *Portfolio evaluation and related performance measures*

An algorithm portfolio is evaluated based on the computational time (risk) associated with it. The main advantage to construct an algorithm portfolio is to gain a computational advantage while solving the problem at hand. The number of objective function evaluations taken by an algorithm portfolio to attain the best solution to a problem is the computational time. The variation in the number of function evaluation is the measure of computational cost while using the portfolio of algorithms. Thus, conceptually the standard deviation of number of function evaluation is a measure of risk associated with the portfolio. In the present case, let $\mathbf{R} = \{R_1, R_2, \ldots, R_n\}$ be the set of random variables associated with n algorithms under consideration. Each outcome R_i is the number of objective function evaluations taken by the ith algorithm to attain the pre-specified performance level.

Let there be $\mathbf{A} = \{A_1, A_2, \ldots, A_{|\mathbf{A}|}\}$ algorithms embedded in a portfolio with ℓ processors such that any algorithm $A_i \in \mathbf{A}$ runs over A_i^{ℓ} processors. Hence, $\ell = A_1^{\ell} + A_2^{\ell} + \cdots + A_{|\mathbf{A}|}^{\ell}$. As defined in the previous paragraph, a random variable (R_i) can be associated with the algorithm portfolio. Let R be the combined probability event distribution for all $A_i \in \mathbf{A}$. Further, for an algorithm A_i, it can be argued that $P[R_i] = r$ is defined as the probability that it requires the rth outcome to attain the prescribed objective performance (Trivedi 2003). Since, EA search technique is utilised in this paper, where it is well established that at a particular stage, generating any solution has equal probability unless some pre-specified bias is not introduced (Gen and Cheng 1997). Here, the term pre-specified bias relates to search strategies based on greedy/gradient-based search, which, however, is not the case in EAs. Hence, each generation in EAs can be seen as a sequence of Bernoulli trials, whereby, instead of counting the number of successes in a fixed number of trials, the number of trials is counted until the first success (obtaining best solution to a problem) is obtained, i.e. attaining a specific performance level.

Let a failure of a single trial be denoted by 0 and a success by 1; then the sample space S of the generations consists of a set of all binary strings with an arbitrary number of 0s followed by a single 1.

$$S = \{0^{r-1}1 | r = 1, 2, 3, \ldots\}. \tag{11}$$

Again, R_i is the random variable such that the value assigned to the sample point $0^{r-1}1$ is r. In order to obtain the probability mass function (*pmf*) of R_i, it is evident that the event $P[R_i] = r$ is true *iff* there is a sequence of $r-1$ failures followed by one success. If probability of each success is p, then the *pmf* of random variable R_i is given by

$$p_{R_i}(r) = p(1-p)^{r-1}, \quad r = 1, 2, 3, \ldots, \ell, \tag{12}$$

where ℓ is the maximum number of object function evaluations allowed for EAs. The probability distribution function (*pdf*) of R_i, can thus be given by:

$$F_{R_i}(t) = \sum_{r=1}^{t} p(1-p)^{r-1} = \frac{p[(1-p)^t]}{1-(1-p)} = 1-(1-p)^{\ell} \quad \forall t \geq 0. \tag{13}$$

Let two algorithms A_i and A_j, associated with random variables R_i and R_j, running on different processors to be be analysed. Let there be another random variable, $\mathcal{R}$, which defines the event that at least one algorithm reach the prescribed success level. Thus,

$$\mathcal{R} = \min(R_i, R_j). \tag{14}$$

Lemma 2.1: *The random variable $\mathcal{R}$ associated with the portfolio of ℓ processors is geometrically distributed with parameter $1-(1-p)\ell$.*

Proof: To begin with, two processors are taken into account, and the probability $P(\mathcal{R} > t)$ can be calculated provided that all the algorithms are independent (Trivedi 2003); the independence of the algorithms (with random variables R_i and R_j) is axiomatic, as the processors considered in the paper have no interactions between them. Hence,

$$P(\mathcal{R} > t) = P(R_i > t \text{ and } R_j > t) \tag{15}$$

$$= P(R_i > t) \cdot P(R_j > t). \tag{16}$$

In terms of *pdf* and using Equation (16),

$$\begin{aligned} P(\mathcal{R} > t) = 1 - F_{\mathcal{R}}(t) &= \left[1 - F_{R_i}(t)\right] \cdot \left[1 - F_{R_j}(t)\right] \\ \Rightarrow F_{\mathcal{R}}(t) &= F_{R_i}(t) + F_{R_j}(t) - F_{R_i}(t) \cdot F_{R_j}(t). \end{aligned} \tag{17}$$

From Equation (13), $F_{R_i}(t) = 1-(1-p)^t \& F_{R_j}(t) = 1-(1-p)^t$. Clearly, as per Equation (17):

$$\begin{aligned} F_{\mathcal{R}}(t) &= 2[1-(1-p)^t] - [1-2(1-p)^t + (1-p)^{2t}] = 1-(1-p)^{2t} \\ &= 1-[(1-p)^2]^t. \end{aligned} \tag{18}$$

Hence, it can be concluded that $\mathcal{R}$ is geometrically distributed with the parameter $1-(1-p)^2$.

On similar grounds, let the already defined case pertaining to overall portfolio be investigated with ℓ processors and $|\mathbf{A}|$ algorithms. The algorithm running over any processor i is associated with a random variable $R_i | \forall_i \in \{1, 2, \ldots, \ell\}$, which is geometrically distributed and is mutually independent. Generalising the lemma from the case of two processors to ℓ processors,

$$F_{\mathcal{R}}(t) = 1 - \prod_{i=1}^{\ell} \left[1 - F_{R_i}(t)\right] - 1 - \prod_{i=1}^{\ell} \left[1 - (1-(1-p)^t)\right] = 1 - \prod_{i=1}^{\ell} \left[(1-p)^t\right]. \tag{19}$$

Thus, the random number $\mathcal{R}$ is geometrically distributed with parameter $1-(1-p)^{\ell}$. □

Next, the following lemma states that using multiple processors in a portfolio is a better option than running one EA on a single processor.

Lemma 2.2: *The number of function evaluations for a search to end up with desired quality solution always has a higher probability when used with ℓ processors than with a single processor,* i.e.

$$F_{R_i}(t) < F_{\mathcal{R}}(t). \tag{20}$$

Proof: Let, first the following relation be considered for analysis

$$f = F_{R_i}(t) - F_{\mathcal{R}}(t) = (1-(1-p)^t) - \left(1 - \prod_{i=1}^{\ell} [(1-p)^t]\right) \tag{21}$$

$$= \prod_{i=1}^{\ell} [(1-p)^t] - (1-p)^t. \tag{22}$$

It can be argued that since p is the probability of success of a single trial, $p < 1$. Also, t and ℓ are greater than zero. Since $(1-p)^t < 1$, $\prod_{i=1}^{\ell}[(1-p)^t]$ is therefore bound to be less than 1, thus making the first term of Equation (21) smaller. In an algorithm portfolio, more than one algorithm or processor is considered, so it can be concluded that in all cases an algorithm portfolio is expected to work faster (i.e. require fewer objective function evaluations) than cases with a single algorithm, hence establishing Equation (20). □

The expectation and standard deviation of a portfolio provide the measure of efficiency. Mathematically, for the case of a single algorithm with a geometrically distributed random variable R_i and *pmf* $p_{R_i}(r) = p(1-p)^{r-1}$, the expectation is given as

$$\begin{aligned} E[R_i] &= \sum_{r=1}^{\infty} rp(1-p)^{r-1} \\ &= p\sum_{r=1}^{\infty} \frac{d}{dq(q^r)} \quad \text{where } q = (1-p) \\ &= \frac{p}{(1-q)^2} = \frac{1}{p}. \end{aligned} \tag{23}$$

Similarly, for the case of ℓ processors each having the same or different copies of algorithms running,

$$E[\mathcal{R}] = \frac{1}{1-(1-p)^{\ell}}. \tag{24}$$

Evaluating in a similar manner for variance 'Var' (σ^2) also, it can be established that

$$\text{Var}[R_i] = \frac{(1-p)}{p^2}. \tag{25}$$

and

$$\text{Var}[\mathcal{R}] = \frac{1-[1-(1-p)^{\ell}]}{[1-(1-p)^{\ell}]^2}. \tag{26}$$

From the expression obtained in Equation (26), it is suggestive that the probability of success and failure of portfolio at any stage is constant, which can be logical in case of random stochastic algorithms (such as the EAs considered in this paper). In the same view, the next subsection details the experimental design for the algorithm portfolio assessment.

3. Portfolio computation: design, analysis, evaluation

This section presents a formal procedure for developing algorithm portfolios for resolving IRPs. A brief overview of the overall procedure that is described in this section is illustrated in Figure 3.

3.1 *Tuning of EAs*

In order to initiate the experiments, all five EAs are tuned within their parameter limits. This is done because any EA is a general algorithmic template whose parameters need to be properly tuned so that it can perform at its best. The tuning of EAs is all about assigning a specific value to the free parameters at which EA performs the best. The aim of this work is to define an automatic hands-off procedure for finding the best parameter settings through statistically guided experimental evaluations. Owing to a large number of constraints, researchers often perform limited empirical studies where candidate algorithms with hand-tuned parameters are tested. The shortcoming of this methodology has been pointed out in several previous studies (Whitley *et al.* 1996, DeJong *et al.* 1997, Eiben and Smit 2011). Therefore, the tuning of algorithms has been illustrated with the aid of statistical box plots.

The box plot shows the median and variance of number of function evaluations in which the algorithm finds the best objective function value (in particular parameter settings in 100 runs). The box plots are constructed by varying the mutation and crossover probabilities in the range [0.01, 0.2] and [0.1, 0.9]. As a result, EAs are allowed to run on an IRP instance for different combinations of mutation and crossover probabilities. Figure 4 illustrates the

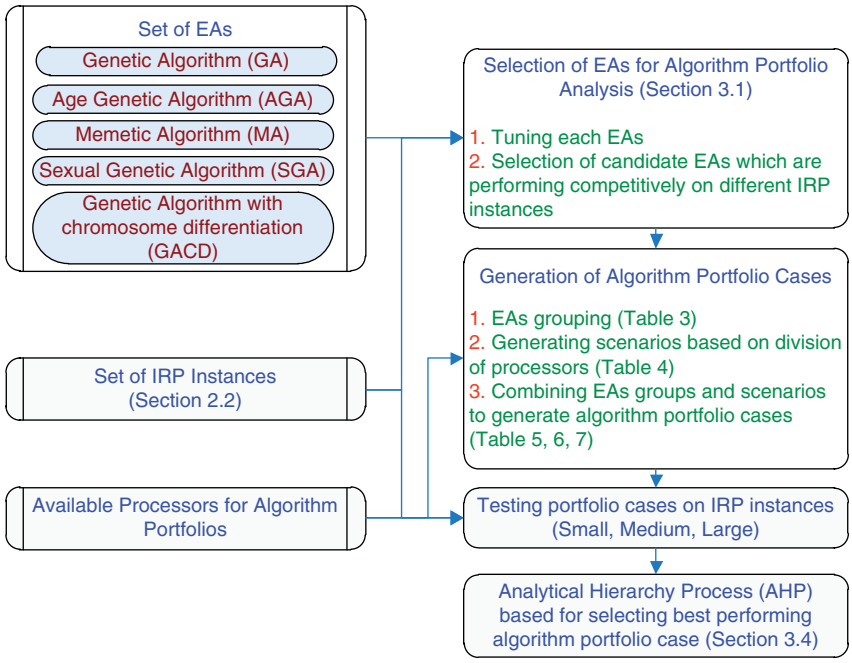


Figure 3. Overall procedure for algorithm portfolio computation.

performance (i.e. the number of objective function evaluations) of each of the EAs with different combinations of mutation and crossover probabilities. The performance is measured by the number of objective function evaluations required by each EA to reach 0.05% of the best-known objective function value. That is, all of the EAs are stopped when either they reach 0.05% of the best known objective function value or the number of function evaluations becomes higher than the pre-defined maximum limit. The crossover and mutation probabilities for each EA are selected based on the median of number of function evaluations required for achieving 0.05% of the best-known objective function value (Figure 4). That is, the lower the median of the number of function evaluations, the better EA performs. Figure 4 shows the tuning of the EAs and the parameters at which the algorithms performs best is shown in Table 2. The number of box plots in the case of SGA is much lower, because the crossover probability has been fixed to 0.8, and accordingly the mutation rate is varied.

3.2 *Selection of EAs for constructing algorithm portfolios*

With the tuned parameters, all five EAs are allowed to be tested on the three groups of IRP instances of varying complexities (Table 1). These algorithms are made to run on a single processor for 100 trials. The results are reported in Figure 5. The stopping criterion for EAs is assumed to be 0.05% of the best-known objective function value. If an EA is not able to find a solution that is in the range of 0.05% of the best solution, then the algorithm stops when the number of function evaluations exceeds its pre-defined maximum limit. The best-known objective function value for IRP is obtained by running all five EAs on IRP instances and recording the number of function evaluations (representative of the time in which the best solution is found by EA). Cumulative frequency graphs (the cumulative frequency versus the number of function evaluations) are plotted on the basis of the number of times (in 100 trials) an EA crossed 0.05% range of the best found objective function value (obtained by previously running all five EA on IRP instance). Based on Figure 5, it can be seen that the performance trends of all EAs are overlapping, i.e. none of the EAs crossed 0.05% range in less number of function evaluations all the times. Therefore, it is evident that none of the EAs are always performing best for all of the problem instances. However, the EAs can be regarded as a set of competing algorithms with comparable performances on IRP instances of varying sizes. Figure 5 shows that AGA performs well for small IRP problems, but it is unable to keep up its performance for a medium class of IRP problems. Moreover, there is no best-suited strategy among the five EAs to efficiently resolve the IRP problem instances of varying size. Therefore, all five EAs have been considered for constructing the algorithm portfolios.

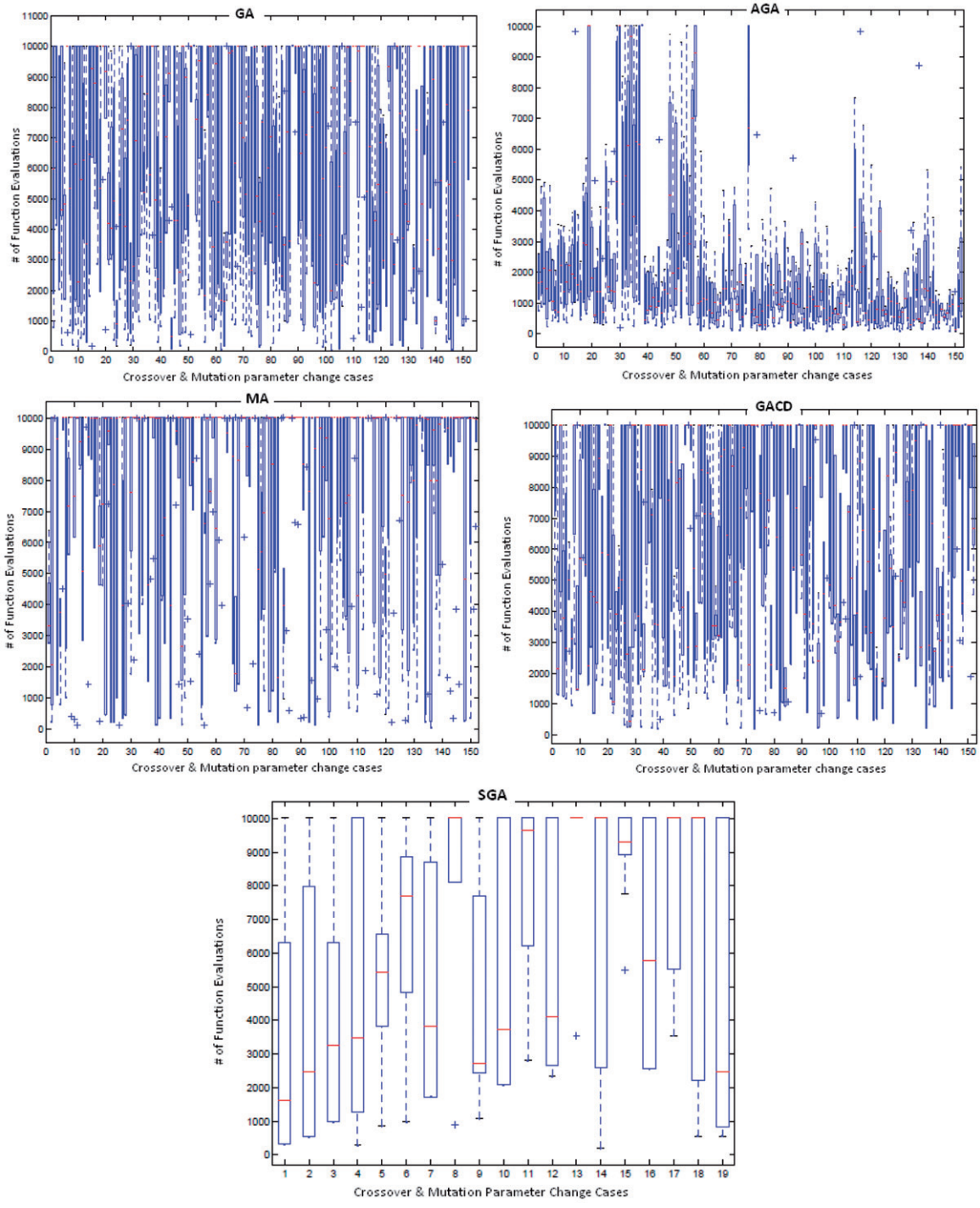


Figure 4. Tuning of GA, AGA, MA, GACD, and SGA based on the number of objective function evaluations (on the y-axis) (see Equation (1)) for different combinations of crossover and mutation probabilities (on the x-axis).

3.3 *Cases investigated and experimental runs of algorithm portfolios*

On the basis of above discussion, the EAs are grouped together with different number of processors to form different cases of algorithm portfolio. Experiments have been performed by using two and four processors with the designed set of two and four EAs. The EAs are grouped into sets of two and four EAs for constructing algorithm portfolios, and this grouping has been presented in Table 3. The basis for selecting algorithms in different groups is

Table 2. Parameters of the best-performing algorithms.

Algorithm	Crossover probability	Mutation probability
GA	0.8	0.02
AGA	0.7	0.01
GACD	0.7	0.03
MA	0.6	0.03
SGA	0.8	0.01

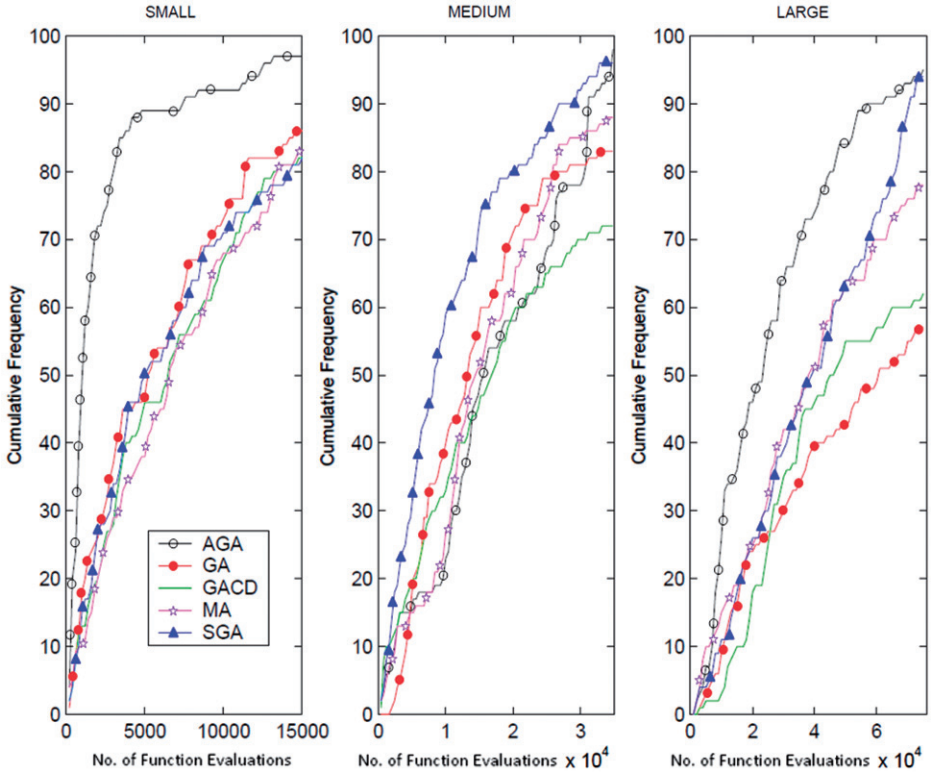


Figure 5. Cumulative frequency plots of EAs for small, medium, and large instances of IRP as described in Table 1. The cumulative frequency illustrated in graphs is the cumulative number of times an EA reached a 0.05% range of best-known objective function value.

Table 3. EA groupings for algorithm portfolio design.

	No. of processors			
No. of EAs	2		4	
2	2P–2A	SGA–GACD GA–MA AGA–SGA GA–AGA	4P–2A	AGA–GA SGA–MA AGA–GACD SGA–MA
4	NA[a]	– –	4P–4A	GA–SGA–GACD–AGA AGA–GACD–MA–SGA

Note: [a]NA, not applicable, as there were only two processors to run four EAs simultaneously.

Table 4. Cases investigated with processor assigned to EA groupings.

S. no.	2P–2A cases (two processors –two EAs)	4P–2A cases (four processors –two EAs)	4P–4A cases (four processors –four EAs)
1	2/0	4/0	4/0/0/0
2	1/1	3/1	0/4/0/0
3	0/2	2/2	0/0/4/0
4	–	1/3	0/0/0/4
5	–	0/4	3/1/0/0
6	–	–	3/0/1/0
7	–	–	3/0/0/1
8	–	–	2/1/1/0
9	–	–	2/0/1/1
10	–	–	1/1/1/1

Table 5. Algorithm portfolios design based on EAs grouping and processor allocation for two processors: two EAs.

Processor allocation No. of processors	EA groupings		
	Cases	EA_1–EA_2	EA_3–EA_4
2	2/0	$EA_1 \leftarrow 2$ processors	$EA_3 \leftarrow 2$ processors
		$EA_2 \leftarrow 0$ processors	$EA_4 \leftarrow 0$ processors
	1/1	$EA_1 \leftarrow 1$ processors	$EA_3 \leftarrow 1$ processors
		$EA_2 \leftarrow 1$ processors	$EA_4 \leftarrow 1$ processors
	0/2	$EA_1 \leftarrow 0$ processors	$EA_3 \leftarrow 0$ processors
		$EA_2 \leftarrow 2$ processors	$EA_4 \leftarrow 2$ processors

Table 6. Experimental runs for 2P–2A (two-processor and two-EA scenario) case.

	IRP instance								
	Small			Medium			Large		
EA groupings	2/0	1/1	0/2	2/0	1/1	0/2	2/0	1/1	0/2
SGA–GACD	1122	3136	3496	9414	10,254	5154	31,710	27,946	44,716
GA– MA	5652	4544	4612	9368	6330	14,710	38,046	26,353	30,026
AGA–SGA	1834	1156	3522	7899	7012	5564	19,644	28,550	20,042
GA–AGA	6346	1675	1172	6020	11,808	7676	39,286	22,514	16,292

to give each EA an equal opportunity to prove their effectiveness, although selecting two EAs from a set of five EAs (GA, GACD, SGA, MA, AGA) results in $C_2^5 = 10$ EA combinations. Similarly, selecting four EAs from a set of five EAs results in $C_4^5 = 5$ EA combinations. In order to restrict the large number of portfolio experimentations, only a few combinations are selected under the two- and four-EA set (Table 3).

With the EA groupings mentioned in Table 3, different cases of algorithm portfolio have been constructed based on the number of processors assigned to each EA in a grouping. The different cases of algorithm portfolios are illustrated in Table 4. To describe the cases specifically, the symbol 2/0 represents the case when two processors are allocated to first EA of the two-EA set, i.e. the first EA of the two-EA set is allowed to run on two processors, and the second EA is run over none. A generic algorithm portfolio design for two processors and two EA groupings is illustrated in Table 5, where $EA_1 \leftarrow 2$ processors, $EA_2 \leftarrow 0$ processors means that EA_1 is allowed to run on two processors, and EA_2 is allowed to run on none.

After formulating a comprehensive experimental scheme (presented in Tables 3 and 4), the performance results are obtained for all the cases of algorithm portfolios under consideration. Each of the algorithm portfolio settings illustrated in Table 4 has been tested for 100 independent runs, and the results have been analysed on the basis of average performances. The experimental results have been presented as the average number of function evaluations for the algorithm portfolio to reach the desired quality level, i.e. 0.05% of the best-known objective function value. The experimentation has been divided into two parts on the basis of the number of processors used.

- *Two-processor scenario*: For this scenario, four groups of EAs are taken into account (Table 3). Each group contains two EAs, and each is analysed for three 2P–2A cases, i.e. [2/0], [1/1], and [0/2] (Table 4). All cases of the algorithm portfolios have been tested on three IRP instances of different complexity groups (Table 1), and the results obtained are presented in Table 6.
- *Four-processor scenario*: In this scenario, six groups of EAs, four groups of two EAs and two groups of four EAs are taken into account (Table 3). For the groups containing two EAs, five 4P–2A cases are possible,

Table 7. Experimental runs for 4P–2A (four-processor and two-EA scenario).

IRP instance	EA groupings	4P–2A cases				
		4/0	3/1	2/2	1/3	0/4
Small	AGA–GA	1420	1747	922	1650	2222
	SGA–MA	2020	2012	2207	1454	1545
	AGA–GACD	622	664	1174	1504	1116
	SGA–MA	2124	2282	1619	1078	1466
Medium	AGA–GA	7884	7408	8304	7002	10,203
	SGA–MA	3562	4653	4957	5937	7561
	AGA–GACD	5462	2458	1560	1128	950
	SGA–MA	1750	2904	3369	4683	5525
Large	AGA–GA	11,862	13,565	19,986	20,363	23,586
	SGA–MA	11,444	7750	8985	9761	12,905
	AGA–GACD	8590	12,380	12,525	13,990	18,787
	SGA–MA	13,828	11,410	10,675	17,585	17,437

Table 8. Experimental runs for 4P–4A (four-processor and four-EA scenario).

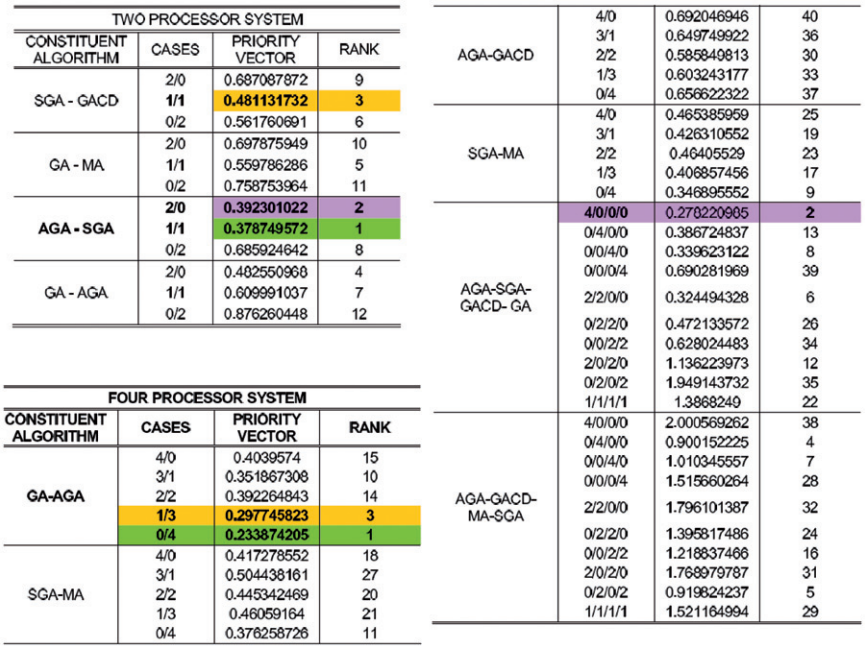
IRP instance	EA groupings	4P–4A cases									
		4/0/0/0	0/4/0/0	0/0/4/0	0/0/0/4	3/1/0/0	3/0/1/0	3/0/0/1	2/1/1/0	2/0/1/1	1/1/1/1
Small	GA–SGA–GACD–AGA	3350	2264	2030	664	2014	1864	864	1306	748	1007
	AGA–GACD–MA–SGA	692	2560	3125	1298	832	2454	1853	1020	3014	1040
Medium	GA–SGA–GACD–AGA	6156	4242	4565	4940	3506	1324	2393	3565	1356	2235
	AGA–GACD–MA–SGA	3630	4356	4035	3432	3538	1470	3046	2356	2969	2432
Large	GA–SGA–GACD–AGA	15,667	11,716	15,366	8229	19,686	21,222	10,102	19,696	18,585	22,523
	AGA–GACD–MA–SGA	9226	17,650	13,458	10,202	9936	15,622	13,566	10,252	19,696	14,555

i.e. [4/0], [3/1], [2/2], [1/3], and [0/4]. However, for groups having four EAs, a large number of cases can be possible. In order to be concise in exhaustive computational experimentation, we have selected 10 4P–4A cases (Table 4) for the study. All the cases of algorithm portfolio have been tested for the three IRPs of different size, and the results obtained are presented in Tables 7 and 8.

3.4 *Evaluating best algorithm portfolio cases*

After the experimental results, the task remains to choose the best-performing algorithm portfolio. Generally, the number of available processors for parallel runs is limited in an organisation; hence, the algorithm portfolio selection strategy has to be executed separately for each of the two scenarios (two and four processors). The varying performances of all five EAs pose a challenge to select the best algorithm portfolio among the cases explored in order to obtain a near-optimal solution to the problem with a minimum number of function evaluations (representative of computational time). The best portfolio is selected based on the analytical hierarchy process (AHP). AHP has been employed here as a tool for decision-makers for selecting the best algorithm portfolio case. AHP uses hierarchical pairwise comparisons to induce weights for different cases of algorithm portfolio (Winkler 1990, Pomerol and Barba-Romero 2000). Each algorithm portfolio case is recognised as an alternative, and a number of function evaluations for over three different IRP sizes are considered as multiple attributes. A matrix $\boldsymbol{O}$ is constructed for each processor scenario, which is mathematically defined as follows:

$$\boldsymbol{O} = \begin{pmatrix} o_{11} & o_{12} & \cdots \\ o_{21} & o_{22} & \vdots \\ \vdots & \vdots & \ddots \end{pmatrix},$$



TWO PROCESSOR SYSTEM

CONSTITUENT ALGORITHM	CASES	PRIORITY VECTOR	RANK
SGA - GACD	2/0	0.687087872	9
	1/1	**0.481131732**	**3**
	0/2	0.561760691	6
GA - MA	2/0	0.697875949	10
	1/1	0.559786286	5
	0/2	0.758753964	11
AGA - SGA	**2/0**	**0.392301022**	**2**
	1/1	**0.378749572**	**1**
	0/2	0.685924642	8
GA - AGA	2/0	0.482550968	4
	1/1	0.609991037	7
	0/2	0.876260448	12

FOUR PROCESSOR SYSTEM

CONSTITUENT ALGORITHM	CASES	PRIORITY VECTOR	RANK
GA-AGA	4/0	0.4039574	15
	3/1	0.351867308	10
	2/2	0.392264843	14
	1/3	**0.297745823**	**3**
	0/4	**0.233874205**	**1**
SGA-MA	4/0	0.417278552	18
	3/1	0.504438161	27
	2/2	0.445342469	20
	1/3	0.46059164	21
	0/4	0.376258726	11
AGA-GACD	4/0	0.692046946	40
	3/1	0.649749922	36
	2/2	0.585849813	30
	1/3	0.603243177	33
	0/4	0.656622322	37
SGA-MA	4/0	0.465385959	25
	3/1	0.426310552	19
	2/2	0.46405529	23
	1/3	0.406857456	17
	0/4	0.346895552	9
AGA-SGA-GACD- GA	**4/0/0/0**	**0.270220905**	**2**
	0/4/0/0	0.386724837	13
	0/0/4/0	0.339623122	8
	0/0/0/4	0.690281969	39
	2/2/0/0	0.324494328	6
	0/2/2/0	0.472133572	26
	0/0/2/2	0.628024483	34
	2/0/2/0	1.136223973	12
	0/2/0/2	1.949143732	35
	1/1/1/1	1.3868249	22
AGA-GACD-MA-SGA	4/0/0/0	2.000569262	38
	0/4/0/0	0.900152225	4
	0/0/4/0	1.010345557	7
	0/0/0/4	1.515660264	28
	2/2/0/0	1.796101387	32
	0/2/2/0	1.395817486	24
	0/0/2/2	1.218837466	16
	2/0/2/0	1.768979787	31
	0/2/0/2	0.919824237	5
	1/1/1/1	1.521164994	29

Figure 6. Algorithm portfolio performance evaluation using AHP.

where the rows represent the alternatives explored (algorithm portfolio cases, ϑ), and the columns represent three attributes (IRP instances) (IRP: small, medium, large); and o_{ij} represents the number of function evaluations for the ith algorithm portfolio case on the jth problem instance. Then, a priority matrix $\Omega^j_{\vartheta\times\vartheta}$ is calculated for each attribute j, where

$$\Omega^j_{ik} = \frac{o_{ij}}{o_{kj}} \quad i,k \in \vartheta. \tag{27}$$

The weight vector $\mathbf{W}^j$ associated with ϑ alternatives is then calculated for each attribute j by taking the geometric mean for the rows corresponding to matrix $\mathbf{\Omega}^j$. The ith element of $\mathbf{W}^j$ can be mathematically represented as

$$w^j_i = \sqrt[\vartheta]{\prod_{k=1}^{\vartheta} \mathbf{\Omega}^j_{ik}}. \tag{28}$$

Thus, the normalised priority vector (PV) is obtained based on the three weight vectors ($\mathbf{W}^1$, $\mathbf{W}^2$, and $\mathbf{W}^3$), which characterise the effectiveness of alternatives. The priority of the ith alternative can be defined as

$$PV_i = \frac{\sum_{j=1}^{3} w^j_i}{3}. \tag{29}$$

The priority vector is used to rank the alternatives. For the two processor scenario, a total of 12 alternatives (4 EAs groups $\times$ 3 cases of processor assignment) have been evaluated. Similarly, for the four-processor scenario, 40 alternatives have been explored. Figure 6 portrays the final priority vector and ranks for the different alternatives discussed above. More insights to the working of AHP can be found in Winkler (1990).

It is apparent from Figure 6 that AHP can provide a set of best alternatives, if they exist. For the two-processor scenario, algorithm portfolio cases for EA group *AGA–SGA* characterised by [1/1], [2/0] are found to be best (Figure 6). Similarly for the four-processor scenario, case [0/4] corresponding to EA group *GA–AGA* and case [4/0/0/0] of EA group *AGA–SGA–GACD–GA* are the best-performing algorithm portfolios. In particular, *AGA*-based algorithm portfolios can be treated as computationally viable options compared to others because it is a constituent EA of all the best-performing algorithm portfolio cases obtained. It can also be observed that often, considering the parallel runs of the same algorithm is a better option than choosing an incoherent algorithm mix over various

Table 9. Final hamming distance (H) values for EAs in the algorithm portfolio analysis.

	AGA	GA	MA	GACD	SGA
H_p	15	6	7	10	14
H_r	21	12	16	15	20
H	0.7142	0.5	0.4375	0.667	0.7

processors. The alternatives chosen by the ranking procedure can be recommended as the best-suited strategies that are recognised as the algorithm portfolio cases with minimum computational risk associated. Hence, a planner can choose a best-performing algorithm portfolio case to maintain flexibility in decision-making with a limited time frame. The following subsection details the selection of an individual EA that performs best.

3.5 *Best EA performing under algorithm portfolio*

The best constituent EA in the algorithm portfolio means that the algorithm is able to reach the 0.05% range of the best-known objective function value a maximum number of times. Therefore, the best algorithm in various portfolio instances can be obtained with the help of a hamming-distance-based procedure. In this procedure, ratio of the number of times an EA crosses 0.05% range to the number of times that particular EA has been allowed to run in all the algorithm portfolio cases. For each algorithm portfolio case, if an algorithm is made to run, then the value of hamming distance for running particular algorithm (H_r) becomes 1, otherwise 0. In addition, if that particular algorithm is able to reach a 0.05% limit then a value of 1 is credited to the hamming distance for performance (H_p), otherwise 0 is assigned to it. Now, the hamming distance (H), which is the measure of the performance of an algorithm in the set of portfolios, is evaluated as

$$H = \frac{\Sigma H_p}{\Sigma H_r}. \tag{30}$$

The H_p, H_r values are summed for all the algorithm portfolio cases in Equation (31) to obtain the H value. Hence, the greater the H value, the better an EA performs. The performances of AGA, GA, MA, GACD, and SGA are judged by the above procedure, and it is found that AGA is able to reach the 0.05% of the best value range a maximum number of times (see H values in Table 9).

3.6 *Performance comparison of best EA and algorithm portfolio cases over 20 IRP instances*

The best algorithm portfolio case for a two-processor scenario, i.e. [1/1] *AGA–SGA* case, and four-processor scenario, i.e. [0/4] *GA–AGA* case illustrated in Figure 6, is made to run on 20 problem instances generated in Table 1. The results corresponding to each processor system, i.e. the maximum and minimum percentage deviation from the best objective function value for 100 runs, is shown in Table 10. It is evident from Table 10 that all portfolios are able to find the solution within 0.05% of the best value found. Moreover, the performance of the best algorithm portfolio cases is improved as the number of processors is increased, which further strengthens the proposed logic of using multiple processors for computation. In order to show the efficiency of algorithm portfolio cases to resolve the problem, comparison has been made with the best-performing algorithm, i.e. *AGA* in Table 10. It can be illustrated from Table 10 that *AGA* performs competitively for small instances, whereas its performance deteriorates as the problem dimension increases. Thus, the algorithm portfolio can be recommended as the best-suited strategy that competitively performs well on the varying sizes of problem instances.

4. Conclusion and future research

This paper details the application of algorithm portfolios based on EAs to solve a complex NP-hard problems of varying size and complexity, and finding quality solutions to the IRP at minimal computational cost. The paper shows the supremacy of combined algorithms running in parallel on different processors without interlinks for solving complex problems such as IRP. Several decisions in the logistics management areas, nowadays, have to be

Table 10. Comparison between best portfolios found and best-performing EA (AGA).

	Percentage deviation from best objective function value					
	Two-processor		Four-processor		AGA	
IRP instances	Min	Max	Min	Max	Min	Max
1	0	0	0	0	0	0
2	0	0.0111	0	0	0	0.0264
3	0	0.0202	0	0	0	0.0266
4	0	0.0214	0	0	0.0036	0.0425
5	0	0.0209	0	0	0.0096	0.0969
6	0	0.0236	0	0	0.0112	0.2969
7	0.0013	0.0054	0	0	0.0235	0.6386
9	0.0021	0.0074	0	0.0002	0.0273	0.9869
9	0.0023	0.0097	0	0.0009	0.0334	1.3308
10	0.0087	0.0101	0	0.0016	0.0396	1.6759
11	0.0095	0.0108	0	0.0023	0.0457	2.0209
12	0.0119	0.0150	0	0.003	0.0518	2.3659
13	0.0135	0.0156	0	0.0037	0.0580	2.7109
14	0.0133	0.0165	0	0.0044	0.0641	3.0560
15	0.0133	0.0171	0	0.0051	0.0702	3.4010
16	0.0137	0.0172	0	0.0058	0.0764	3.7460
17	0.0141	0.0213	0.0005	0.0065	0.0826	4.0910
18	0.0145	0.0245	0.0010	0.0072	0.0886	4.4361
19	0.0149	0.0267	0.0014	0.0079	0.0948	4.7811
20	0.0154	0.0268	0.0018	0.0086	0.1009	5.1261

taken within the minimum specific time period; therefore it is necessary to utilise the coherent power of EAs. The algorithm portfolios are designed on the basis of a parallel run of various algorithms without any interlink; the absence of communication helps algorithms to take the advantage of inherent search capabilities, and the parallel implementation improves any disadvantage of randomness in the search. The theoretical analysis also supports the logic of using a portfolio of algorithms rather than a single algorithm for complex optimisation problems. The paper also proposes a methodology for finding the best algorithm portfolio cases by utilising the concept of AHP. Comparisons among multiple processor scenarios and best algorithms have been made over 20 IRP problem instances to justify the concept of algorithm portfolios. Moreover, the characteristics of algorithm portfolios such as fast and robust optimisation make them perfect tools for decision-makers to resolve various complex problems.

Future research of the algorithm portfolios would be to resolve the various complex and dynamic problems of different domains. Also, the communication among several processors may be done for further improving the searching capabilities of the EAs running on them.

References

Aghezzaf, E.H., Raa, B., and Landeghem, H.V., 2006. Modeling inventory routing problems in supply chains of high consumption products. *European Journal of Operational Research*, 169 (3), 1048–1063.

Brown, K.L. *et al*., 2003. A portfolio approach to algorithm selection. *In*: *Eighteenth International Joint Conference on Artificial Intelligence* (IJCAI-03), 9–15 August, Acapulco, Mexico. Available from: ai.stanford.edu/~shoham/www%20papers/portfolio-IJCAI03.pdf [Accessed 25 January 2012].

Dawkins, R., 1976. *The selfish gene*. Oxford: Oxford University Press.

de Berg, M., *et al*., 2005. TSP with neighborhoods of varying size. *Journal of Algorithms*, 57, 22–36.

DeJong, K A., Potter, M A., and Spears, W M., 1997. Using problem generators to explore the effects of Epistasis. *In*: T. Back, ed. *Seventh International Conference on Genetic Algorithms, 19–23 July 1997*. East Lansing, MI. Waltham, MA: Morgan Kaufmann, 338–345.

Eiben, A.E. and Smit, S.K., 2011. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1 (1), 19–31.

Gen, M. and Cheng, R., 1997. *Genetic algorithm and engineering design*. New York: Wiley.

Ghosh, A., Tsutsui, S., and Tanaka, H. 1996. Individual aging in genetic algorithms. *In*: *Australian and New Zealand Conference on Intelligent Information Systems*, 18–20 November, Adelaide, Australia. Piscataway, NJ: IEEE.

Goh, K.S., Lim, A., and Rodrigues, B., 2003. Sexual selection for genetic algorithms. *Artificial Intelligence Review*, 19 (2), 123–152.

Goldberg, D.E., 1989. *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley.

Gomes, C. and Selman, B., 2001. Algorithm portfolios. *Artificial Intelligence*, 126 (1–2), 4362.

Holland, J.H., 1975. *Adaption in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

Huberman, B.A., Lukose, R.M., and Hogg, T., 1997. An economics approach to hard computational problems. *Science*, 275 (5296), 51–54.

Lagoudakis, M. and Littman, M., 2000. Algorithm selection using reinforcement learning. *In*: *Proceedings of the Seventeenth International Conference on Machine Learning*, 29 June–2 July. San Diego, CA. Waltham, MA: Morgan Kaufmann, 511–518.

Merz, P. and Freisleben, B., 1997. A genetic local search approach to the quadratic assignment problem. *In*: C.T. Back, ed. *Proceedings of the 7th International Conference on Genetic Algorithms*. San Diego, CA. Waltham, MA: Morgan Kaufmann, 465–472.

Pomerol, J.C. and Barba-Romero, S., 2000. *Multicriterion decision in management: principles and practice*. Boston: Kluwer Academic Publishers.

Raa, B. and Aghezzaf, E.H., 2005. A robust dynamic planning strategy for lot-sizing problems with stochastic demands. *Journal of Intelligent Manufacturing*, 16 (2), 207–213.

Rice, J.R., 1976. The algorithm selection problem. *In*: M. Rubinoff and M.C. Yovits, eds. *Advances in Computers*. New York: Academic Press, 65–118.

Shen, Z., Dessouky, M., and Ordonez, F. 2007. Stochastic vehicle routing problem for large-scale emergencies. Working paper.

Singh, G. *et al*., 2006. An evolutionary approach for multi-pass turning operation. *Proceedings of the I MECH E Part B Journal of Engineering Manufacture*, 220, 145–162.

Song, L.Q. *et al*., 2009. Ensemble for Solving Quadratic Assignment Problems. *In*: *International Conference of Soft Computing and Pattern Recognition*, *SOCPAR '09*, 4–7 December, Malacca, Malaysia. Piscataway, NJ: IEEE, 190–195.

Stinson, D.R., 1987. *An introduction to the design and analysis of algorithms*. 2nd ed. Winnipeg, Manitoba: The Charles Babbage Research Centre.

Taha, H.A., 2006. *Operations research: an introduction*. Englewood Cliffs, NJ: Prentice-Hall.

Tasgetiren, M.F., Suganthan, P.N., and Pan, Q.K., 2010. An ensemble of discrete differential evolution algorithms for solving the generalized traveling salesman problem. *Applied Mathematics and Computation*, 215 (9), 3356–3368.

Trivedi, K.S., 2003. *Probability and statistics with reliability, queuing, and computer science applications*. New York: Wiley.

Wagner, S. and Affenzeller, M., 2005. SexualGA: gender-specific selection for genetic algorithms. *In*: *Proceedings of the 9th World Multiconference on Systemics, Cybernetics and Informatics*, 10–13 July, Orlando, FL [CD-ROM].

Whitley, D., *et al*., 1996. Evaluation evolutionary algorithms. *Artificial Intelligence*, 85 (1–2), 245–276.

Winkler, R.L., 1990. Decision modeling and rational choice: AHP and utility theory. *Management Science*, 36 (3), 247–248.

Yang, W., Mathur, K., and Ballou, R.H., 2000. Stochastic vehicle routing problem with restocking. *Transportation Science*, 34 (1), 99–112.