Proceedings of the Symposium

Evolutionary Systems

A symposium at the AISB 2009 Convention (6-9 April 2009)
Heriot-Watt University, Edinburgh, Scotland

Symposium Chairs
Prof. David Corne
Dr Pier Frisco
Dr Alan Reynolds

Published by SSAISB:
The Society for the Study of Artificial Intelligence
and the Simulation of Behaviour
http://www.aisb.org.uk/

ISBN - 1902956761

Evolutionary Systems

A one-day symposium at AISB 2009 (6-9 April 2009).
http://www.macs.hw.ac.uk/~dwcorne/edcs/

PROGRAMME CHAIRS

Prof David Corne, Heriot-Watt University, UK
Dr Pier Frisco, Heriot-Watt University, UK
Dr Alan Reynolds, Heriot-Watt University, UK

INTRODUCTION

Evolutionary computation is now well established in both research and practice as a highly effective approach in solving a substantial range of problems in science and industry. But this success has raised more questions than answers, and evolutionary computation now covers an immense range of techniques and applications, with subfields of research opening up with each new challenge thrown up by applicatons and/or by hybridisation with other techniques. Some current frontiers of evolutionary computation research include, for example, large-scale systems, multiobjective optimisation, bioscience applications, dynamic problems, and many more.

In this one-day symposium we experience a snapshot of the current state of the art in a number of these prominent areas, with a focus on multiobjective optimisation, and on biosystems applications, and financial applications.

These foci reflect how scientists and practitioners are increasingly resorting to evolutionary computation for difficult, large-scale problems that deal with complex real-world systems.

PROGRAMME COMMITTEE

Prof David Corne, Heriot-Watt University, UK
Dr Pier Frisco, Heriot-Watt University, UK
Dr Alan Reynolds, Heriot-Watt University, UK

Table of Contents

Multi-Objective Optimization of Cancer Chemotherapy Using Swarm Intelligence

Andrei Petrovski[1], John McCall[1] and Bhavani Sudha[1]

Abstract. Cancer chemotherapy aims at achieving a number of treatment goals, not all of which are commensurate. Because of this, the problem of chemotherapy optimization necessitates the use of multi-objective optimization methods. The techniques based on swarm intelligence have certain features that make them applicable and effective in addressing multiple treatment objectives of cancer chemotherapy. This paper demonstrates the adaptive capabilities of particle swarm optimization (PSO) that enables this bio-inspired metaheuristic to carry out an efficient search for both effective and versatile chemotherapy treatments.

Keywords. Swarm intelligence, multi-objective optimization, medicine, metaheuristics.

1. INTRODUCTION

Chemotherapeutic cancer treatment necessitates making complex, and often life-critical, decisions about the best way to administer cytotoxic (i.e. destroying the cells) drugs. Typically patient information is incomplete and noisy, the range of treatment options is subject to complex constraints and the aims of treatment are multi-objective. For these reasons, medical decision support is a rich application area for evolutionary algorithms and related techniques [5].

The authors have developed a decision support system, the Oncology Workbench [4], the primary aims of which are to help oncologists to design, evaluate and optimise multi-drug chemotherapy treatments of cancer. The optimization engine of the Oncology Workbench (OWCH) is capable of determining better sequencing and dosing of the specified cytotoxic drugs, and recommends the optimised treatment(s) to the oncologists for consideration.

The current version of the optimization facility has several limitations though. Firstly, the search for better treatment schedules is conducted using genetic algorithms. Our previous studies [6] and [4] showed that alternative evolutionary techniques of computational optimization, the particle swarm optimization (PSO) in particular, can facilitate more efficient optimization of chemotherapeutic treatments. The better efficiency is achieved through finding feasible (i.e. satisfying all cancer chemotherapy constraints) treatment schedules much faster.

Secondly, the current version of the OWCH decision support system implements a single-objective optimization of cancer chemotherapy. The treatment schedules are selected on the basis of the tumour reduction criterion – that is, how effectively they reduce the overall tumour burden during the treatment period. Although tumour reduction, and ideally elimination, remains the primary objective of cancer chemotherapy, its achievement is not always possible due to various medical constraints. It is essential therefore to provide oncologists with the option to change the objective of treatment optimization from finding the best curative treatment to developing a good controlling (palliative) treatment capable of extending the patient survival time (PST) or the quality of patients' lives.

The authors have successfully explored the possibility of multi-objective optimization of cancer chemotherapy in their previous work [7], but confined their study to the domain of genetic algorithms only. Given inferior performance of this optimization technique in comparison with particle swarm optimization (PSO) and estimation of distribution algorithms (EDAs), it seems logical to revive the multi-objective approach to cancer chemotherapy optimization in the context of more efficient heuristics with the purpose to enhance the optimization facility of the developed decision support system for oncologists.

The remainder of this paper is organized as follows. In Section 2 we explain the objectives of cancer chemotherapy optimization, highlighting their conflicting nature that excludes the possibility of converting the problem from multiple to single objective optimization using weighting or any other techniques. Section 3 discusses how the multi-objectivity of chemotherapy optimization can be addressed by the PSO method. In Section 4 we present the experimental results that are followed by a short concluding Section 5, where we outline the direction of future work.

2. MILTI-OBJECTIVE NATURE OF CHEMOTHERAPY OPTIMIZATION

Chemotherapy is the use of cytotoxic drugs to control or eliminate cancer. The drugs are administered to the body by a variety of routes with the aim to create a certain concentration in the bloodstream that will act to systematically kill cells [1]. This means that both cancerous and normal cells will suffer due to the effect of these drugs.

The intention of the drug administration is to eradicate the tumour or, at least, to control the proliferation of cancerous cells. However, the treatment has toxic side-effects on the rest of the body. The success of chemotherapy therefore depends crucially on maintaining sufficient damage to the tumour while effectively managing the toxic side-effects.

[1] School of Computing, The Robert Gordon University, AB25 3UE, UK.
Email: {ap, jm, bs}@comp.rgu.ac.uk

Designing chemotherapeutic treatments is a complex optimal control problem that involves mathematical modelling of tumour growth, satisfaction of several medical constraints, and resolving conflicting objectives of treatment optimization. We will address all these aspects of cancer chemotherapy in the following subsections.

2.1 Mathematical Modelling of Tumour Growth and Reduction

Various models can be used to simulate the response of a tumour to chemotherapy. The most popular is the Gompertz growth model with a linear cell-loss effect [RASC'99]. The model takes the following form:

$$\frac{dN}{dt} = N(t)\cdot\left[\lambda \ln\left(\frac{\Theta}{N(t)}\right) - \right. \\ \left. - \sum_{j=1}^{d}\kappa_j \sum_{i=1}^{n} C_{ij}\left\{H(t-t_i) - H(t-t_{i+1})\right\}\right] \quad (1)$$

where $N(t)$ represents the number of tumour cells at time t; λ and Θ are the parameters of tumour growth, $H(t)$ is the Heaviside step function; C_{ij} denote the concentrations of anti-cancer drugs exceeding the threshold concentration level below which the drugs have no therapeutic effect; and κ_j are the quantities representing the efficacy of anti-cancer drugs.

The first term in equation (1) represents the growth of an untreated tumour, whereas the second term models the effects of cytotoxic drugs leading to tumour reduction. The pharmacodynamics (PD) of drugs is taken into account by estimating their concentration above the therapeutic threshold that is equal to the minimum drug concentration necessary to kill any cancerous cells. The pharmacokinetics (PK) of treatment is addressed by means of finding drug concentrations at the tumour site, eliminating the necessity of tracking the drug dissipation throughout the body.

Although more detailed models of a tumour response to treatment have been introduced and studied in the context of computational optimization ([10], [14] and [15]), they rely on the correct estimation of a considerable number of PD/PK parameters, which significantly complicates their use in practice. Furthermore, the substitution $u(t) = \ln(\Theta/N(t))$ yields an analytical solution to Equation (1) that can be used evaluating the quality of treatment schedules found by computational search heuristics [3].

2.2 Constraints of Cancer Chemotherapy

The adverse effects of cancer chemotherapy stem from the systemic nature of this treatment: drugs are delivered via the bloodstream and therefore affect all body tissues. Since most anti-cancer drugs are highly toxic, they inevitably cause damage to sensitive tissues elsewhere in the body. In order to limit this damage, toxicity constraints need to be placed on the amount of drug applied at any time interval, on the cumulative drug dosage over the treatment period, and on the damage caused to various sensitive tissues. In addition to toxicity constraints, the tumour size (i.e. the number of cancerous cells) must be maintained below a lethal level during the whole treatment period for obvious reasons [7].

Cytotoxic drugs are usually delivered according to a discrete dosage program in which there are n doses given at times $t_1, t_2, \ldots t_n$ [3]. In the case of multi-drug chemotherapy, each dose is a cocktail of d drugs characterised by their concentration levels $C_{ij}, i \in \overline{1,n},\ j \in \overline{1,d}$ in the bloodplasma. If we denote a set of these concentrations as a vector $\mathbf{c} = (C_{ij})$, then the constraints of chemotherapeutic treatment of cancer can be represented in the following general form [7]:

1. Maximum instantaneous dose $C_{\max}$ for each drug acting as a single agent:

$$g_1(\mathbf{c}) = \left\{ C_{\max\, j} - C_{ij} \geq 0 \,\vdots\, \forall i \in \overline{1,n}, \forall j \in \overline{1,d} \right\} \quad (2)$$

2. Maximum cumulative C_{cum} dose for drug acting as a single agent:

$$g_2(\mathbf{c}) = \left\{ C_{\text{cum}\, j} - \sum_{i=1}^{n} C_{ij} \geq 0 \,\vdots\, \forall j \in \overline{1,d} \right\} \quad (3)$$

3. Maximum permissible size $N_{\max}$ of the tumour:

$$g_3(\mathbf{c}) = \left\{ N_{\max} - N(t_i) \geq 0 \,\vdots\, \forall i \in \overline{1,n} \right\} \quad (4)$$

4. Restriction on the toxic side-effects of multi-drug chemotherapy:

$$g_4(\mathbf{c}) = \left\{ C_{\text{s-eff}\ k} - \sum_{j=1}^{d} \eta_{kj} C_{ij} \geq 0 \,\vdots\, \forall i \in \overline{1,n}, \forall k \in \overline{1,m} \right\} \quad (5)$$

The factors η_{kj} in the last constraint represent the risk of damaging the k^{th} organ or tissue (such as heart, bone marrow, lung etc.) by administering the j^{th} drug.

It is also possible to introduce constraints associated with the drug delivery schedule. Standard chemotherapies often use fixed dosages and drug combinations at equally spaced intervals throughout the treatment period. The main reasons for such a delivery mode are that it is easier to organise patients to present for treatment at fixed times and that fixed dosages and combinations are more easily communicated and administered [4].

2.3 Objectives of Chemotherapy Treatments

The main goals of chemotherapeutic treatments are:

- *Cure*: curative treatments attempt to eradicate the tumour. In our studies we define eradication to mean a reduction of the tumour from an initial size of around 10^9 cells (minimum detectable tumour size) to below 10^3 cells.
- *Control*: if cure is not possible, the goal is to control the disease, i.e. stop the cancer from growing and spreading. One commonly used performance measure of controlling treatments is the patient survival time (PST) that needs to be extended.
- *Palliation*: if the cancer is at an advanced stage, even controlling the disease, let alone curing it, might become impossible. In such cases chemotherapy drugs may be used to relieve symptoms caused by the cancer, thereby improving the quality of life, even though the drugs may not lengthen the patient's life.

For some people, chemotherapy is the only treatment mode used in an attempt to cure, control, or palliate the cancer. At the same time, chemotherapy may be given along with other treatments – it may be used as *neoadjuvant* therapy (before surgery or radiation in order to shrink the tumour, for example), or as *adjuvant* therapy (after surgery or radiation to prevent the growth of stray cancer cells remaining in the body) [ACS].

Using the notation from the previous subsections, the objective of a curative treatment can be mathematically formulated as follows [3]:

$$\underset{\mathbf{c}}{\text{maximise}} \quad f_1(\mathbf{c}) = \int_{t_1}^{t_n} \ln\left(\frac{\Theta}{N(\tau)}\right) d\tau \qquad (\mathbf{6})$$

subject to the state equation (1) and the constraints (2)-(5).

If we denote the PST as T, then the treatment objective of extending the patient survival time (PST) can be represented as:

$$\underset{\mathbf{c}}{\text{maximise}} \quad f_2(\mathbf{c}) = \int_{t_1}^{T} d\tau = T \qquad (\mathbf{7})$$

again subject to (1)-(5).

The main objective of a palliative treatment is to improve the quality of life of the patients for whom neither curative nor controlling treatments can be found. As such, the last objective is qualitative in nature and it is not a straightforward task to express it in rigorous mathematical terms. One crude approach to improving the quality of patients' life is to minimise the amount of drugs delivered simultaneously in the attempt to weaken toxic side-effects caused by chemotherapy.

As can be concluded from the discussion above, cancer chemotherapy is a complex treatment mode that might be used for pursuing a variety of treatment objectives. What makes chemotherapy an interesting domain for applying multi-objective optimization techniques is that not all of these objectives are mutually exclusive. In fact, oncologists are often faced with unpredicted reactions from the patients to well-known and widely-used chemotherapy schedules. In such situations the treatment objective will have to be dynamically adjusted, or alternative treatment strategies will have to be developed that yield satisfactory outcomes with respect to more than one treatment objective. In the next section we will look at how swarm intelligence can assist oncologists in finding such strategies.

3. MULTI-OBJECTIVE OPTIMIZATION USING SWARM INTELLIGENCE

Swarm intelligence (SI) is based on the principles underlying the behaviour of natural systems consisting of many "agents", and exploiting local communication form, highly distributed control, and emergent strategies [12]. In contrast to centralized programs, based on the global goal, the SI algorithms use the synergy of the individual efforts of a population of agents that are not aware of the global objective to be reached. This creates a framework conducive for discovering versatile strategies highly adaptable to changing environments.

The driving force of the SI algorithms is the interactions between the agents themselves as well as between the agents and the environment. The capability of an individual agent is fairly limited; however, when a large number of agents are brought together, constructive behaviour can emerge that leads to effective addressing of the objectives not known *a priori*.

Computational modelling of swarms has resulted in numerous successful applications aimed at global function optimization [12], finding optimal routes and schedules, and at locating multiple optima [10]. In the next section we will look at how the SI algorithms – the Particle Swarm Optimization (PSO) method – can be adapted for solving multi-objective optimization problems.

3.1 Multi-objective Particle Swarm Optimization (MOPSO)

In general, a multi-objective optimization problem (MOP) consists of n decision variables comprising a decision vector $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \Omega \subset \Re^n$, m constraints $g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_m(\mathbf{x})$, and k objectives expressed as (non)linear criteria or objective functions $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})$. Brought together, the multiple objectives define the evaluation function $F\big(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})\big): \Omega \to \Lambda \subset \Re^k$, which, if some of the objectives are in conflict, places a partial, rather than normal, ordering on the search space Ω. In order to mathematically define this partial ordering, a notion of Pareto dominance is introduced in the objective space Λ.

The specificity of multi-objective optimization is to find a set of non-dominated decision vectors rather than the global optimum, which might not even exist. Pareto-optimal decision vectors cannot be improved in any objective without causing deterioration of at least one other objective. Such decision vectors comprise the Pareto-optimal set, $P^* \subset \Omega$, in the search space.

The mapping of the Pareto-optimal set to the objective function space gives rise to the Pareto front PF^*. The Pareto front can be non-convex and non-connected; nonetheless, if it is known, or at least approximated reasonably well, the decision maker will be able to select a solution via a choice of acceptable objective performance and, as a result of this, the problem of multi-objective optimization can be resolved.

Solving multi-objective optimization problems therefore involves a decision maker (DM), who brings additional knowledge about the problem, on the basis of which a particular solution from the Pareto set is chosen. Depending on the stage at which the DM is consulted, multi-objective optimization is categorized into *a priori*, *a posteriori*, and interactive optimization [11].

In *a priori* optimization the DM is consulted at an early stage with the purpose to set the optimization priorities. If *a posteriori* optimization is used, the DM is presented with a set of Pareto optimal solutions, and the task is to select the most suitable one. The interactive optimization combines both approaches and provides dynamic feedback from the DM that is used to guide the optimization process.

Many studies have shown that the process of multi-objective optimization can be effectively implemented by evolutionary algorithms. A number of these algorithms specifically aimed at solving multi-objective optimization problems include the non-dominated sorting GA II (NSGA II), the niched Pareto GA (NPGA), the Pareto-archived evolutionary strategy (PAES), the Pareto-envelope based selection algorithm (PESA), the strength Pareto evolutionary algorithm 2 (SPEA2), the incrementing multi-objective optimization evolutionary algorithm (IMOEA), and the like [11].

A relatively new addition to the class of evolutionary algorithms used in the context of multi-objective optimization is the approach based on particle swarms. PSO seems particularly suitable for multi-objective optimization mainly because of the high convergence speed [2]. Furthermore, a number of techniques have been proposed to improve the efficiency of multi-objective PSO (MOPSO) [13].

A detailed description of a MOPSO algorithm is provided in [2]. We have adopted this algorithm for addressing the problem of cancer chemotherapy optimization given multiple treatment objectives. The implementation issues are discussed in the next section.

3.2 Applying MOPSO for Optimising Chemotherapeutic Treatments

In our previous work [6] we explored the local- and global-best PSO algorithms on a single-objective optimization problem of finding a curative cancer chemotherapy treatment. In our implementation, each particle represents a candidate treatment schedule $\mathbf{c} = (C_{ij}), i \in \overline{1,n},\ j \in \overline{1,d}$, encoded as a string of integers. The representation space $\mathbf{I}$ (a discretized version of Ω) can then be expressed as a Cartesian product:

$$\mathbf{I} = C_1^1 \times C_1^2 \times \ldots \times C_1^d \times C_2^1 \times C_2^2 \times \ldots \times C_2^d \times \ldots \times C_n^1 \times C_n^2 \times \ldots \times C_n^d \quad \textbf{(8)}$$

of the sets C_i^j, each of which represents an integer value of concentration units for the i^{th} dose of the j^{th} drug. The magnitude of the concentration unit is calculated as a certain portion of the maximum instantaneous dose $C_{\max j}$ for each cytotoxic drug used.

The PSO algorithm is initialised with a population of random candidate solutions, conceptualised as particles. These particles are flown through the hyperspace Ω of solutions to the chemotherapy optimization problem. The position of each particle $\bar{c}_i^{k+1}$ at iteration $k+1$ corresponds to a treatment schedule of cytotoxic drugs and is determined by the following formula:

$$\bar{c}_i^{k+1} = \bar{c}_i^{k} + \bar{v}_i^{k} \quad \textbf{(9)}$$

where is $\bar{v}_i^{k}$ a randomised velocity vector assigned to each particle in a swarm. The velocity vector drives the optimization process and reflects the 'socially exchanged' information. Each particle in the swarm is attracted towards the locations representing best chemotherapeutic treatments found by the particle itself, its neighbours, and/or the entire population. This is achieved by defining the velocity vector in (9) for each particle as:

$$\bar{v}_i^{k} = w \cdot \bar{v}_i^{k-1} + b_1 \cdot r_1 \cdot \left(\bar{c}_i^{*} - \bar{c}_i^{k-1}\right) + b_2 \cdot r_2 \cdot \left(\bar{c}_i^{**} - \bar{c}_i^{k-1}\right) \quad \textbf{(10)}$$

where:

- w is the inertia coefficient the value of which is randomly generated from the range $[0.5,1]$;
- b_1 and b_2 are empirical coefficients used to improve PSO performance – in our experiments $b_1 = b_2 = 4$;
- r_1 and r_2 are random numbers is the range $[0,1]$;
- $\bar{c}_i^{*}$ and $\bar{c}_i^{**}$ are the best locations in Ω found by the particle i and the entire population respectively;
- $\bar{v}_i^{k-1}$ is the value of particle i velocity at previous iteration of the algorithm; the values $\bar{v}_i^{0}$ are initialised at random from the range $[0,2]$, i.e. $\bar{v}_i^{0} \in [0,2],\ i \in \overline{1,50}$. Particle velocities calculated from (10) have the lower and upper bounds, $|v_i| \le 1$, that reduce the particles' oscillation.

The search heuristic of the PSO algorithms, defined by equations (9)-(10), tend to focus exploration around remembered locations in the search space – a particular advantage when optimal solutions are confined to the boundary of a feasible region (wherein all constraints are satisfied)[4]. Multi-objective PSO approximates this boundary using the archive of non-dominated solutions, which in our implementation has the maximum size of

500 chemotherapy schedules that perform well with respect to the multiple treatment objectives described in Section 2.3.

The program implementing the MOPSO algorithms is written in Java. The termination criterion was chosen to be 250,000 fitness function evaluations. With the population of 50 particles, this implies evolving 5,000 generations of particle swarms. The results of our experiments are presented in the next section we present.

4. EXPERIMENTAL RESULTS

In order to provide a quantitative assessment of the MOPSO performance, three measures are often taken into consideration – the distribution of particles, their spread across the Pareto optimal front, and the ability to attain the global tradeoff [Liu, Gou, et al].

For the problem of multi-objective chemotherapy optimization neither the global tradeoff is known, nor can the true Pareto front be determined precisely. Therefore, as far as the performance measures are concerned, we are limited to only qualitative analysis of the particles' distribution and to the speed with which MOPSO can find feasible solutions to the multi-objective problem of chemotherapy optimization. Also, we can compare the quality of the solutions found by MOPSO with those discovered by a single-objective PSO algorithm that has been previously proposed by the authors [6].

Figure 1shows the spread of non-dominated solutions stored in the MOPSO archive at the end of a typical experimental run. The x-axis represents the quality of solutions with respect to the curative objective of chemotherapy treatment (6), whereas the y-axis shows the quality measure of palliative treatment (in both cases larger values are preferable because the objectives need to be maximized).

As can be seen from the figure, the non-dominated solutions are well-spread, providing a good incentive for the particles to fully explore the solution space Ω in the vicinity of the Pareto set. This is particularly important in the context of MOPSO because for each particle the global best reference $\overline{c}_i^{**}$ is selected from the set of non-dominated solutions based on the closest Euclidian distance.

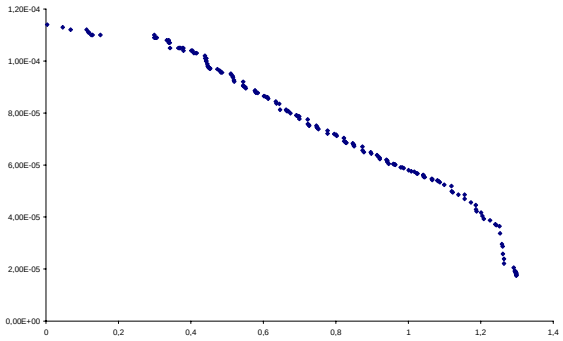

Figure 1. A set of non-dominated solutions found by the MOPSO algorithm

Secondly, the quality of the MOPSO solutions can be evaluated by comparing them with the best solutions found by a single-objective PSO algorithm in [PPSN'04]. The analysis shows that the performance index of the best MOPSO solution (Treatment **A**) with respect to the treatment objective (6) is 3.328 (indicating the log value of the ratio of the initial tumour size N_0 and its size at the end of the treatment interval), whereas the single-objective PSO (Treatment **B**) yields the value of 3.330 for the same objective.

Figure 2 shows the effects of both treatments on the same tumour and on the patient survival time. It demonstrates that the Treatment A found by MOPSO reduces the tumour almost as effectively as Treatment B, but given its reasonable performance with respect to the palliative treatment objective, causes substantially weaker toxic side-effects, improving thereby the quality of the patient life.

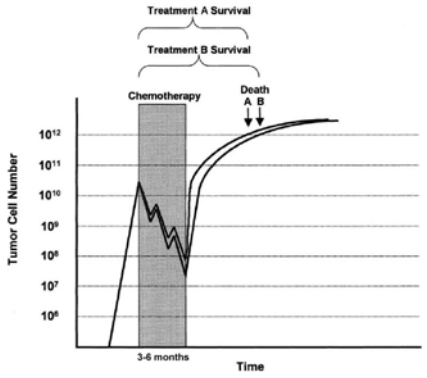

Figure 2. Multi- vs. single-objective PSO on a commensurate treatment objective

Thirdly, our experimental results have shown, that the MOPSO algorithm finds a feasible solution within 25 generations, indicating its superior search efficiency in comparison with other evolutionary algorithms – genetic algorithms in particular [6].

5. CONCLUSIONS AND FUTURE WORK

The optimization of cancer chemotherapy based on the swarm intelligence approach demonstrates high adaptive capabilities that can be effectively used for addressing multiple treatment objectives. The swarm's ability to retain information on the good solutions found by each particle and pass it round the whole population in an effective manner is a valuable asset in solving both single- and multi-objective chemotherapy optimization problems. The provision of a multi-objective optimization facility will expand the scope and enhance the quality of the decision support available to the oncologists from the Oncology Workbench system developed by the authors [4].

REFERENCES

[1] American Cancer Society:
http://www.cancer.org/docroot/home/index.asp

[2] Coello, C., Pulido, G., Lechuga, M.: Handling multiple objectives with particle swarm optimization. In *IEEE Transactions on Evolutionary Computation*, **8**, No. 3 (2004), pp. 256--79.

[3] Martin, R., Teo, K.: Optimal Control of Drug Administration in Cancer Chemotherapy. World Scientific, Singapore New Jersey London Hong Kong (1994).

[4] .McCall, J., Petrovski, A., Shakya, S.: Evolutionary Algorithms for Cancer Chemotherapy Optimization. In *Computational Intelligence in Bioinformatics* by Fogel D., Corne D., and Pan Y. (Eds.). IEEE Press (2008) pp. 265-96.

[5] Petrovski, A., McCall, J.: Smart problem solving environments for medical decision support. Proceeding of the *Workshops on Genetic and Evolutionary Computation (GECCO'05)*. ACM Press, New York (2005), pp.152-58.

[6] Petrovski, A., Sudha, B., McCall, J.: Optimising cancer chemotherapy using particle swarm optimization and genetic algorithms. Proceeding of the *8th International Conference on Parallel Problem Solving from Nature*, Lecture Notes in Computer Science, Volume **3242**. Springer, Berlin, Heidelberg (2004), pp. 633-41.

[7] Petrovski, A., McCall, J. A.: Multi-objective optimization of cancer chemotherapy using evolutionary algorithms. *Evolutionary Multi-Criterion Optimization*, Lecture Notes in Computer Science, Volume **1993**, Zitzler E., et al. (Eds.) Springer, Berlin, Heidelberg (2001), pp. 531-45.

[8] Petrovski, A., McCall, J.: Computational optimization of cancer chemotherapies using genetic algorithms. In: John, R., Birkhead, R. (eds.): *Soft Computing Techniques and Applications*. Series on Advances in Soft Computing. Physica-Verlag (1999), pp. 117-122.

[9] Liu, D., Tan, K., Goh, C., Ho, W.: A multi-objective memetic algorithm based on particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, **37**(1), (2007), pp. 42-50.

[10] Kay Chen Tan, Khor, E. F., Cai, J., Heng, C. M., Lee, T. H.: Automating the drug scheduling of cancer chemotherapy via evolutionary computation. Artificial Intelligence in Medicine **25**(2), (2002, pp. 169-185.

[11] Knowles, J., Corne, D.: Memetic algorithms for multi-objective optimization: Issues, methods and prospects. In W. Hart, N. Krasnogor and J.E. Smith (Eds.), *Recent Advances in Memetic Algorithms*. Springer: Studies in Fuzziness and Soft Computing, **166**, (2005). pp. 313—52.

[12] Olariu, S., Zomaya, A. (Eds.): *Handbook of Bioinspired Algorithms and Applications*. Chapman and Hall (2006).

[13] Reyes-Sierra, M., Coello, C.: A study of techniques to improve the efficiency of a multi-objective particle swarm optimizer. In S. Yang, Y.-S. Ong and Y. Jin (Eds.), *Evolutionary Computation in Dynamic and Uncertain Environments*. Springer (2007), pp. 269—96.

[14] Tse, S., Liang, Y., Leung, K., Lee, K., Mok T.: A memetic algorithm for multiple-drug cancer chemotherapy schedule optimization. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics* (2007), Vol. **37-1**, pp. 84-91.

[15] Villasana, M. and Ochoa, G.: Heuristic design of cancer chemotherapy. *IEEE Transactions of Evolutionary Computation* (2004), Vol. **8**, pp. 513-21.

On the Use of Sharpe's Index in Evolutionary Portfolio Optimization Under Markowitz's Model

Feijoo Colomine Duran[1] and **Carlos Cotta**[2] and **Antonio J. Fernández**[2]

Abstract. Portfolio optimization is a problem that lends itself naturally to multiobjective approaches, e.g., aimed to maximize the return of the investment, simultaneously minimizing the risk. The selection of an actual portfolio requires exercising a decision-making process on the set of efficient solutions thus obtained. In this work we consider the case in which knowledge of this selection criterion is available, and used within the optimizer. We use Sharpe's index, a measure of excess return per unit of risk, for this purpose. It is shown that a multi-start single-objective evolutionary algorithm based on this index can provide a better coverage of the relevant regions of the Pareto front than state-of-the-art multiobjective evolutionary algorithms. An extensive experimental analysis is conducted using real data from a Latin American stock exchange.

1 Introduction

Portfolio optimization is a conspicuous problem in the area of financial management. Broadly speaking, it amounts to determining an adequate distribution of investments, such that an acceptable economic return is obtained, and good risk diversification is achieved. Obviously, the extent to which a particular return is considered acceptable or a certain risk diversification is good depends on the profile of the investor. There are a number of theoretical studies regarding the risk/performance relation. Among these, Markowitz's model [11] has become an essential theoretical reference for portfolio selection.

Markowitz's model is based on the rational behavior of the investor, who tries to maximize her profit and rejects the risk. The collection of portfolios offering a combination of risk/profitability such that no higher profit can be obtained without increasing the risk as well is termed the *efficient frontier*, and once known the investor can select her optimal portfolio according to her preferences. Of course, the determination of this efficient frontier (or Pareto front) is by no means an easy task in general. Fortunately, powerful optimization techniques can be used for this purpose, such as for example multiobjective evolutionary algorithms (MOEAs) [5, 3].

MOEAs have been deployed on portfolio optimization problems in numerous occasions, e.g., see [17, 15, 7] among other works. However, in this work we are specifically concerned not just about the calculation of a quasi-optimal Pareto front, but also on the subsequent decision-making process. In a recent work [4] we have analyzed the performance of several MOEAs in light of a very precise selection criterion, namely Sharpe's index. This index measures how much excess profit per risk unit delivers a certain portfolio, and is parameterized by a value that indicates the observed (or desired) return of a risk-free portfolio. An important consequence of the use of this selection criterion is the fact that specific regions of the Pareto front turn out to be more relevant, and hence algorithmic comparisons based on absolute multiobjective performance (as measured by standard quality indicators) do not necessarily coincide with the relative performance of selected solutions. This fact lead us to consider the inclusion of this selection criterion within the optimization process. This is specifically interesting when the parameter determining the risk-free return is not known or given in advance. We will assess the performance of a multistart EA based on this selection index, and analyze the quality of the results obtained in a variety of scenarios, with special emphasis in the comparison to state-of-the-art multiobjective EAs.

[1] Universidad Nacional Experimental del Táchira (UNET), Laboratorio de Computación de Alto Rendimiento (LCAR), San Cristóbal, Venezuela, email: fcolomin@unet.edu.ve

[2] Dept. Lenguajes y Ciencias de la Computación, ETSI Informática, University of Málaga, Campus de Teatinos, 29071 - Málaga, Spain, email:{ccottap,afdez}@lcc.uma.es

2 Background

As state before, we consider an investment scenario in which an investor wishes to maximize profitability and minimize risk. This will be formalized within Markowitz's model in next subsection.

2.1 Markowitz's Model

Markowitz's model [11] assumes that the future performance that a specific investment can offer can be determined from both experience and investigation. Two main components have to be taken into account: profitability and risk to be assumed by the investor. The overall risk of the portfolio is defined as a weighted quadratic combination of the covariances of the assets included in it, i.e.,

$$\sigma^2(\vec{R}|\vec{W}) = \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \sigma_{ij}(\vec{R}) \tag{1}$$

where $\vec{W} = \{w_i\}$, $1 \leqslant i \leqslant n$, is a vector comprising the fraction of the budget allocated to each asset ($w_i \geqslant 0$), and $\sigma_{ij}(\vec{R})$ is the covariance of the performance of the i-th asset and the j-th asset, defined as:

$$\sigma_{ij}(\vec{R}) = \sum_{t=1}^{T} \frac{[R_{it} - E(R_i)]\,[R_{jt} - E(R_j)]}{T} \tag{2}$$

where $\vec{R} = \{R_{it}\}$, $1 \leqslant i \leqslant n$, $1 \leqslant t \leqslant T$, is a matrix containing the profitability of each asset at each time interval t, $E(R_i)$ is the mean profitability of the i-th asset, and T is the number of intervals in the time horizon.

Similarly to the risk, the profitability $E(\vec{R}|\vec{W})$ of a portfolio is defined as the weighted average of the assets involved, i.e.,

$$E(\vec{R}|\vec{W}) = \sum_{i=1}^{n} w_i E(R_i) \tag{3}$$

Once the profitability and risk of a portfolio is defined, there remains the issue of determining which portfolio, among all available possibilities, should be preferred. This is tackled in next subsection.

2.2 Sharpe's Index

Generally speaking, the investor looks for the curve of utility with $E(\vec{R}|\vec{W}) = \infty$ and $\sigma^2(\vec{R}|\vec{W}) = 0$, but this not a realistic option as this curve is limited by the existing assets that never have this nature. We note that for the assets without risk (i.e., those with null profit-variance), the utility is equal to the expected profitability because there is no penalization due to the risk.

To evaluate the quality of a portfolio we have to define a measure that accounts for both the profitability and the risk of the assets involved. Such a measure can also allow the comparison between different portfolios. To this end, we have considered Sharpe's index [13], that determines performance according to the ratio between excess profitability and risk. More precisely,

$$S(\vec{R}|\vec{W}) = \frac{E(\vec{R}|\vec{W}) - R_0}{\sigma(\vec{R}|\vec{W})} \tag{4}$$

where R_0 is the performance of a portfolio without risk. $E(\vec{R}|\vec{W}) - R_0$ is therefore the excess performance (that is, the extra profit obtained by taking some risks), which is divided by the risk of the portfolio (measured as the standard deviation of returns). Basically, the index indicates how much extra performance is expected with respect to the risk. The higher the value returned is, the higher the success of the fund management is.

3 Evolutionary Portfolio Selection and Sharpe's Index

The portfolio selection problem posed in previous section will be tackled with evolutionary algorithms (EAs). This can be done from a multiobjective perspective, that is, finding a quasi-optimal set of efficient solutions and selecting one of them using a specific decision-making procedure (maximizing Sharpe's index in our case). Alternatively, this selection criterion can be directly embedded within the fitness function. Both possibilities are described next.

3.1 Optimization Setting

As stated before, Markowitz's model is based on the assumption the investor abhors risk, which can be represented as the variability of returns for a certain investment. At the same time, she wants to maximize her profits. Hence, we can define a portfolio as efficient if it achieves the profit sought by the investor at the minimum risk. This consideration leads naturally to a multiobjective scenario in which Pareto-optimal portfolios are sought, i.e., portfolios whose profitability cannot be increased without increasing the risk as well (and vice versa, the risk cannot be reduced without decreasing the expected return too). This set of efficient portfolios can be calculated by solving the bi-objective problem $\left\{\min \sigma^2(\vec{R}|\vec{W}), \max E(\vec{R}|\vec{W})\right\}$ subject to $\sum_{i=1}^{n} w_i = 1$.

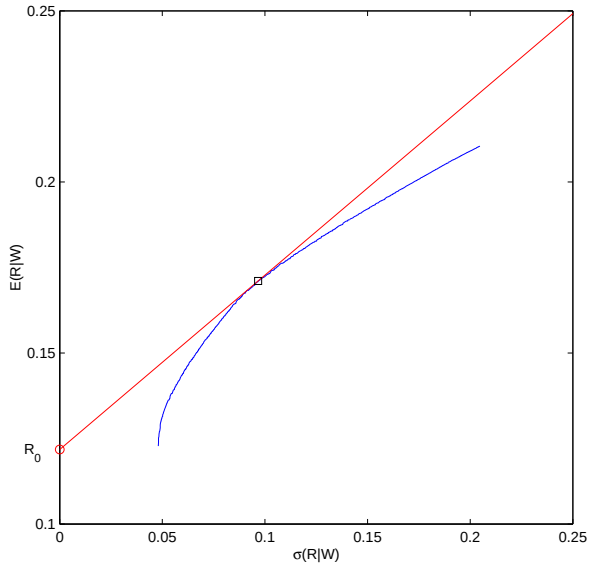

Figure 1. Selection of a solution from the Pareto front using Sharpe's index. The reference risk-free return R_0 is marked with an open circle on the Y axis, whereas the selected solution from the Pareto front (shown in blue) is marked with a square. In this example, both the Pareto front and the reference point R_0 correspond to real data used in the experimentation (mixed funds, see Sect. 4).

Subsequently, once the optimal Pareto set (or a good approximation to this set) has been calculated, the decision-making criterion can be exercised to extract a single preferred solution from this set. In our case, we consider the use of Sharpe's index for this purpose, as stated before.

It is interesting to note the geometrical interpretation of this decision-making procedure based on Sharpe's Index. Since this index is the ratio between excess profit and risk, all profit/risk pairs located along a straight line running through $(0, R_0)$ correspond to hypothetical portfolios with the same value for Sharpe's index (which is the slope of this line, actually). Furthermore, only points below the optimal Pareto front may represent attainable portfolios. Therefore, the maximum value of Sharpe's index corresponds to the line of highest slope that runs through both the point $(0, R_0)$ and an actual portfolio. This corresponds to a tangent to the Pareto front as illustrated in Fig. 1.

This geometrical interpretation leads to an additional observation: given two portfolios $\vec{W}_1$ and $\vec{W}_2$, if the former has a higher value of Sharpe's index, then it is not dominated (in the Pareto sense) by the latter. This can be easily seen by noting that in order for $\vec{W}_1$ to be dominated by $\vec{W}_2$, it must be that $E(\vec{R}|\vec{W}_1) \leqslant E(\vec{R}|\vec{W}_2)$ and $\sigma^2(\vec{R}|\vec{W}_1) \geqslant \sigma^2(\vec{R}|\vec{W}_2)$, with at least one of the inequalities being strict. However, if this is the case, then $\vec{W}_2$ has a higher value of Sharpe's index, since it has a larger numerator and a smaller denominator in Eq. (4). This fact paves the way to defining a single-objective EA that directly tries to maximize Sharpe's index: by progressing toward better values of the index, the EA will also advance toward the Pareto front, ideally converging to the optimal tangent point. Of course, this ideal behavior does not have to happen necessarily, since there may be local optima in the search landscape along the way (as dictated by the risk/profit profile of attainable portfolios). We will return to this point in Sect. 4, when analyzing the experimental results.

3.2 Algorithmic Approaches

The multiobjective portfolio optimization problem has been tackled with three state-of-the-art MOEAs, namely NSGA-II (Non-dominated Sorting Genetic Algorithm II) [6], SPEA2 (Strength Pareto Evolutionary Algorithm 2) [22] and IBEA (Indicator-Based Evolutionary Algorithm) [21]. The first two ones are the second-generation version of two previous algorithms (NSGA [14], and SPEA [23] respectively). As such, they rely on the use of elitism (an external archive of non-dominated solutions in the case of SPEA2, and a plus-replacement strategy –keeping the best solutions from the union of parents and offspring– in the case of NSGA-II). More precisely, the central theme in these algorithms is assigning fitness to individuals according to some kind of non-dominated sorting, and trying to preserve diversity among solutions in the non-dominated front. The third algorithm considered is IBEA, which is aimed to maximize some multiobjective performance indicator, and uses a replacement strategy that tries (in a greedy way) to optimize the value of this indicator for the current population. In this work, we have considered an IBEA based on the ε-indicator [19].

In addition to the multiobjective EAs mentioned before, we have also considered a single-objective EA aimed to optimizing Sharpe's index. This EA is termed SEA –after Sharpe's index-based EA– and has the advantage of being a simpler approach, since no archiving of solutions nor Pareto-based selection/replacement is necessary. More precisely, we have considered the use of binary tournament selection, and a plus replacement strategy.

In all algorithms considered, solutions –that is, a vector of rational values in the $[0, 1]$ range, indicating the fraction of the portfolio devoted to each fund– are represented as binary strings. Each fund is assigned 10 bits, yielding a raw weight $\bar{w}_i$. These weights are subsequently normalized as $w_i = \bar{w}_i / \sum_j \bar{w}_j$ to obtain the actual composition of the portfolio. Evaluation is done by computing the risk and return of the portfolio using the formulation depicted before. As to reproduction, we consider standard operators such as two-point crossover and bit-flip mutation.

4 Results

The data used in the experiments is taken from the Caracas Stock Exchange (*Bolsa de Valores de Caracas* - BVC), the only securities exchange operating in Venezuela. More precisely, we have considered data spanning from five up to eight years of stock trading. This time interval is large enough to be representative of the evolution of shares, and not too large to include irrelevant –for prediction purposes- data (the status of funds can fluctuate in the long term, commonly making old data useless for forecasting the future evolution of shares). According to this, our sample – $\sim 35,000$ daily prices of different mutual funds: fixed, variable, and mixed– comprises funds operating for at least five years and still available in the BVC [1]. To be precise, we have used weekly market data from year 1994 to year 2002, corresponding to 26 Venezuelan mutual funds: 12 fixed funds, 7 variable funds, and 7 mixed funds. Data up to year 2001 is used for training purposes, whereas data corresponding to the year 2002 will be used for testing the obtained portfolios with respect to an investment portfolio indexed in the BVC. The relative ratio of share values in successive weeks is calculated to compute the profitability of each fund. This is done for each week in the year, and subsequently averaged to yield the annual weekly mean and thus obtain the annual profit percentage. The covariance matrix of these profitability values is also computed, as a part of Markowitz's model.

Experiments have been done with the four algorithms described before, namely NSGA-II, SPEA2, IBEA, and SEA. For the first three techniques –the multiobjective EAs– we have utilized the PISA library (A Platform and Programming Language Independent Interface for Search Algorithms) [2]. In all cases, the crossover rate is $P_X = 0.8$, the mutation rate is $P_M = 1/\ell$, and the population size is 2ℓ, where ℓ is the total number of bits in a solution. The algorithms have been run for a maximum number of 100 generations. The number of runs per data set is 30.

The first part of the experimentation considers the use of the value $R_0 = 0.1218$ observed during the time window considered. Using this value, a single solution can be selected from the Pareto front obtained by each multiobjective EA. Likewise, the SEA can use this value to evolve portfolios with high values of Sharpe's index. The results are shown in Fig. 2. As it can be seen, while there is a certain amount of variability in the results provided by the MOEAs in each run, SEA does consistently provide a focused result, notably better (with statistical significance at the standard 0.05 level, using a Wilcoxon ranksum test [10]) than the remaining algorithms.

A natural question arises from these previous results, namely the extent to which SEA would be capable of beating the remaining multiobjective approaches should the value of R_0 be different. Notice that by varying this value, the tangent point to the Pareto front varies, and the convergence properties of SEA may be different. The MOEAs are not affected by this change though, since R_0 is only used in the decision-making process, after the algorithm has been run.

To investigate this issue, we have considered values of R_0 ranging from 0 to $R_0^{\max}$ (0.39, 0.20 and 0.40 for fixed, mixed and variable funds respectively; these values have been chosen by considering the highest profit attained in the grand overall Pareto front). To allow a fair comparison between SEA and the MOEAs, 30 equally spaced values from this range have been selected, and a single run of SEA has been done on each of them. The underlying question we want to address is whether it is better to devote the allotted computational effort (in this case, 30 runs of 10,000 evaluations each) to 30 multiobjective runs, or to 30 different single-objective runs.

The first aspect of the results we have analyzed is the structure of the Pareto front attained in each case. Notice that by archiving the solutions obtained by SEA in each run, we can actually build a non-dominated front, even when each of the independent runs was mono-objective. Of course, it cannot be expected in principle that the front provided this way by SEA is competitive in a general sense with that of the MOEAs, but an analysis of this front can anyway provide interesting information on the behavior of SEA. This analysis has been conducted using two well-known quality indicators: the hypervolume indicator [18] and the R_2 indicator [8] – see also [20]. The first one provides an indication of the region in fitness space that is dominated by the front (and hence the larger, the better), and the second indicator estimates the extent to which a certain front approximates another one (the true Pareto-optimal front if known, or a reference front otherwise). To be precise, we have considered the unary version of this indicator, taking the combined NSGA-II/SPEA2/IBEA/SEA Pareto front as reference set. Being a measure of distance to the reference set, the lower a R_2 value, the better. The lower right corner of the minimum box comprising this combined front has been used as reference point for hypervolume calculation.

Figs. 3 and 4 show the results for the experiments realized. In each case, the upper boxplot indicates the distribution of indicator values for the fronts obtained in each execution of the MOEAs (it does not make sense in the case of SEA, since each execution is going to be focused on a very specific region of the front, and hence the indi-

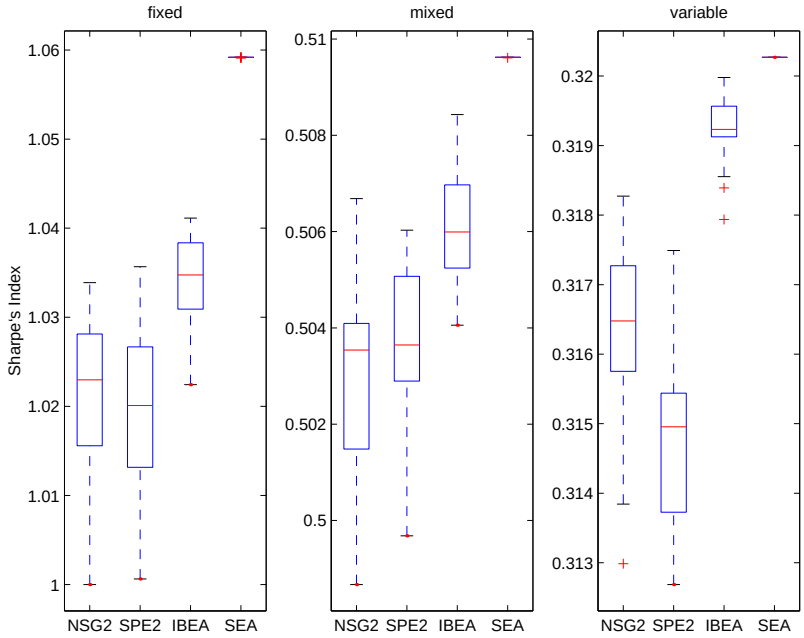

Figure 2. Boxplot of Sharpe's index values attained by NSGA-II, SPEA2, IBEA and SEA on fixed funds (right), mixed funds (middle) and variable funds (left). As usual, the boxes comprise the two middle quartiles of the distribution, the central line indicates the median of the distribution, the whiskers span 1.5 times the interquartile range, and outliers are indicated by a + symbol.

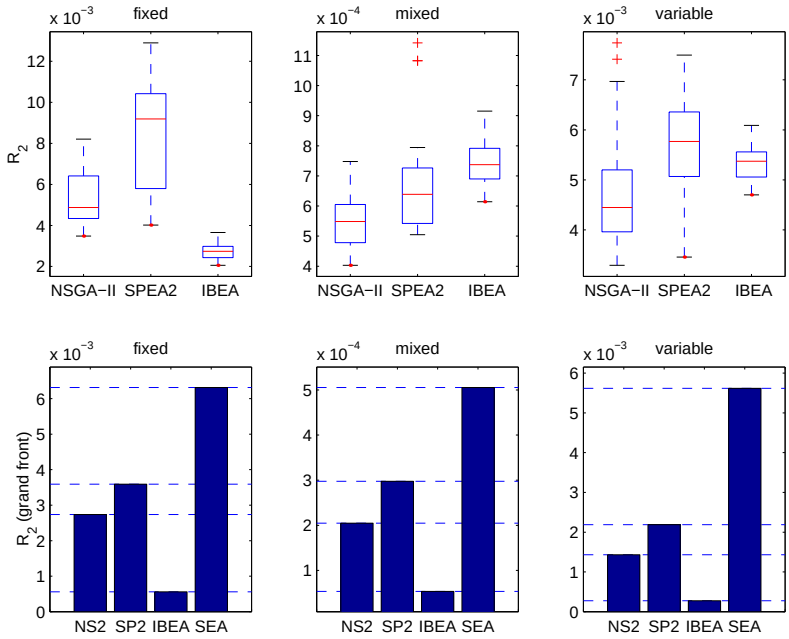

Figure 3. (Top) Boxplot of the R_2 indicator for NSGA-II, SPEA2 and IBEA. (Bottom) Indicator values for the grand fronts.

cator values are going to be poor and uninformative), and the lower bar graph indicates the indicator value for the grand front obtained aggregating the 30 runs of each algorithm.

Let us consider the R_2 indicator in first place (Fig. 3). As it can be seen, the grand front generated by IBEA is notably better than the remaining fronts; the grand front provided by SEA is on the contrary much worse than that of the other algorithms. This can be explained by the fact that SEA only focuses on the regions of the front that are tangent to a straight line originating at $(0, R_0)$ for the values of R_0 considered. The coverage of intermediate zones between these regions is weaker, and hence the grand SEA front exhibits a larger distance to the grand overall front used as reference. Notice that IBEA also exhibits an uneven behavior in individual runs, but manages to provide a good coverage of the reference front when its

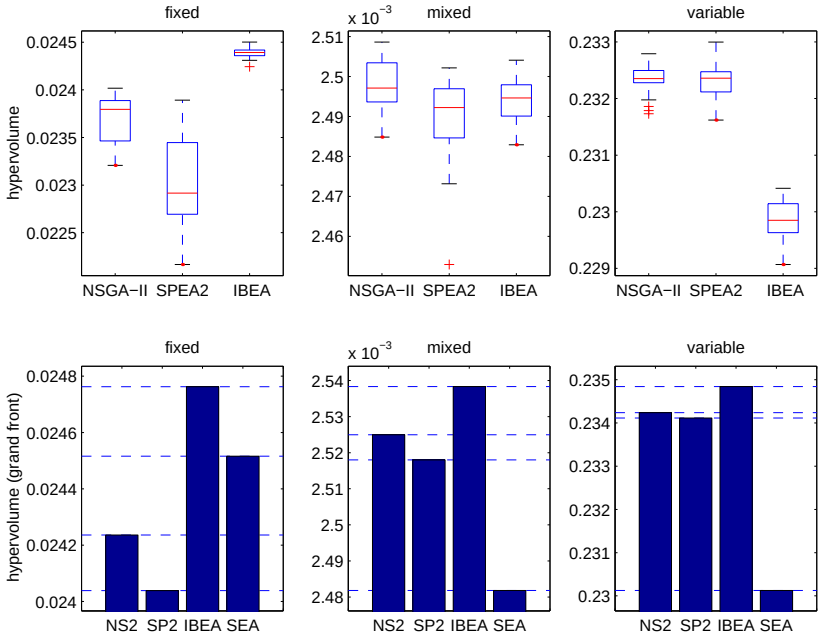

Figure 4. (Top) Boxplot of the hypervolume indicator for NSGA-II, SPEA2 and IBEA. (Bottom) Indicator values for the grand fronts (notice the range of values in the vertical axis).

30 runs are aggregated. As to the hypervolume indicator, it indicates that IBEA provides the best front too, but this time SEA does not appear to be much worse (actually, the difference is very small, as it can be seen by inspecting the range of values in the Y axis). This has an explanation: although SEA only achieves a good coverage of specific parts of the front, it is capable of advancing much deeper there, thus increasing the dominated hypervolume.

To see how the qualitatively different behavior of SEA with respect to the MOEAs affects its performance on different investment scenarios, the next step of the analysis focuses on the relative performance of selected solutions under different values of R_0. To be precise, for each algorithm we have considered its grand front, and selected the best solution (according to Sharpe's index) for 100 different values of R_0 (between 0 and the corresponding value of $R_0^{\max}$). Notice that this sample of values of R_0 is larger than that used in SEA runs, and includes many intermediate values not considered in the execution of this latter algorithm. Fig. 5 shows the results. Values of Sharpe's index have been normalized, dividing by the corresponding values returned by SEA. Therefore, values above (resp. below) 1.0 indicate better (resp. worse) values of Sharpe's index that those provided by SEA.

In general, there is an intermediate range of R_0 values for which all algorithms perform similarly. This range is larger in the case of mixed funds than for fixed or variable funds. In any case, NSGA-II and SPEA2 perform below SEA, and start to diverge quickly for large values of R_0. IBEA is capable of performing similarly to SEA on a broader range of R_0 values, but again falls below for larger values. In all cases this indicates that SEA has a better coverage of the upper-right region of the front (high profit, high risk), containing the solutions selected for increasing values of R_0. It is also interesting to note the behavior of SEA for low values of R_0 on fixed funds. There is a small interval of R_0 values where the MOEAs perform better than SEA, which converges toward a suboptimal region of the front (cf. Sect. 3.1). Subsequent multiple runs of SEA on this particular interval of values indicate that this suboptimal region has a strong basin of attraction when Sharpe's index is used as fitness function. Only around 5% of SEA runs are capable of converging toward the best region identified by the MOEAs for this particular case. However, this behavior does not take place outside this small interval of R_0 values, nor in the remaining data sets.

5 Conclusions

Portfolio optimization is a natural arena for multiobjective optimizers, which are both powerful and flexible enough to deal with this kind of problems. This is specifically true if the optimization process is done in absence of knowledge on the particular decision-making process that will take place afterwards, in order to select a solution from the Pareto front. However, if this knowledge is available, the case for a full-fledged multiobjective approach is not so strong, at least to the extent that this multiobjective approach treat all points in the Pareto front on the same basis. In this sense, we have shown that a multi-start single-objective EA based on Sharpe's index can outperform state-of-the-art MOEAs on several portfolio optimization instances, and on a different range of investment scenarios, providing a selective coverage of interesting regions of the Pareto front.

This does not imply by any means that MOEAs cannot be useful for this optimization task. For example, we have obtained evidence that the SEA can in certain circumstances face difficulties to locate the optimal region (according to the decision-making procedure chosen) of the Pareto front for a given profit objective. Although multiple additional runs may solve this problem, further analysis is clearly required here. We plan to carry on a deeper study of the fitness landscape in this scenario to characterize this situation. Future work will be also directed to analyze other variants of the problem where additional constraints are introduced, e.g., cardinality constraints, lot sizes, etc. This line of research is underway. Another line of future research concerns the measure of risk. We have focused on variance here, but there are other options. We may for example consider value at risk, i.e., the maximum loss that can take place at a certain confi-

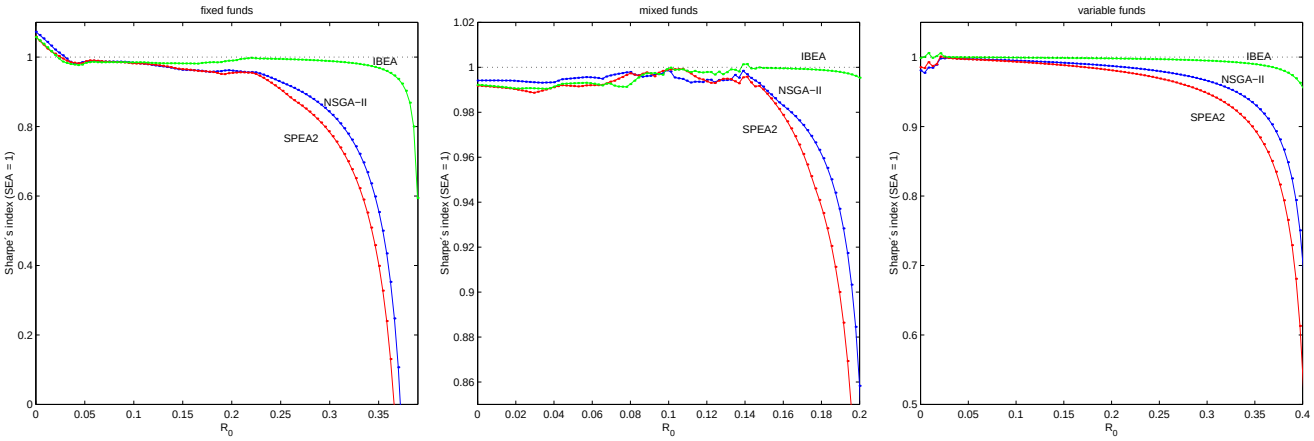

Figure 5. Normalized values of Sharpe's index provided by NSGA-II, SPEA2, and IBEA for different values of R_0.

dence level. A related measure is the conditional value at risk, namely the expected shortfall in the worst $q\%$ of cases, where q is a parameter. Other possible measures are Jensen index [9], Treynor index [16], or models emanating from capital asset pricing theory (CAPM) [12], among others.

ACKNOWLEDGEMENTS

This work is partially supported by projects TIN2008-05941 (of Spanish Ministry of Innovation and Science) and P06-TIC2250 (from Andalusian Regional Government).

REFERENCES

[1] AVAF. Annual report of the venezuelan fund management association, 2003. `http://www.avaf.org/`.

[2] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler, 'PISA—A Platform and Programming Language Independent Interface for Search Algorithms', in *Evolutionary Multi-Criterion Optimization – EMO 2003*, eds., C. M. Fonseca et al., volume 2632 of *Lecture Notes in Computer Science*, pp. 494–508, Berlin Heidelberg, (2003). Springer-Verlag.

[3] Carlos Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, volume 5 of *Genetic Algorithms and Evolutionary Computation*, Kluwer Academic Publishers, 2002.

[4] F. Colomine Duran, C. Cotta, and A.J. Fernández, 'Evolutionary optimization for multiobjective portfolio selection under Markowitz's model with application to the Caracas Stock Exchange', in *Nature-Inspired Algorithms for Optimisation*, ed., R. Chiong, volume 193 of *Studies in Computational Intelligence*, 489–509, Springer-Verlag, (2008).

[5] Kalyanmoy Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, Chichester, UK, 2001.

[6] Kalyanmoy Deb, Samir Agrawal, Amrit Pratab, and T. Meyarivan, 'A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II', in *Parallel Problem Solving from Nature VI*, eds., Marc Schoenauer et al., volume 1917 of *Lecture Notes in Computer Science*, pp. 849–858, Paris, France, (2000). Springer-Verlag.

[7] J.E. Fieldsend, J. Matatko, and M. Peng, 'Cardinality constrained portfolio optimisation', in *Intelligent Data Engineering and Automated Learning – IDEAL 2004*, eds., Z.R. Yang, R. Everson, and H. Yin, volume 3177 of *Lecture Notes in Computer Science*, pp. 788–793, Berlin Heidelberg, (August 2004). Springer-Verlag.

[8] M.P. Hansen and A. Jaszkiewicz, 'Evaluating the quality of approximations to the nondominated set', Technical Report IMM-REP-1998-7, Institute of Mathematical Modelling Technical University of Denmark, (1998).

[9] M. C. Jensen, 'The performance of mutual funds in the period 1945 - 1964', *Journal of Finance*, **23**, 383–417, (May 1968).

[10] E.L. Lehmann and H.J.M. D'Abrera, *Nonparametrics: Statistical Methods Based on Ranks*, Prentice-Hall, Englewood Cliffs, NJ, 1998.

[11] Harry M. Markowitz, 'Portfolio selection', *Journal of Finance*, **7**, 77–91, (1952).

[12] W.F. Sharpe, 'Capital assets prices: A theory of market equilibrium under conditions of risk', *Journal of Finance*, **19**, 425–442, (1964).

[13] W.F. Sharpe, 'Mutual fund performance', *Journal of Business*, **39**, 119–138, (1966).

[14] N. Srinivas and Kalyanmoy Deb, 'Multiobjective optimization using nondominated sorting in genetic algorithms.', *Evolutionary Computation*, **2**, 221–248, (1994).

[15] F. Streichert, H. Ulmer, and A. Zell, 'Evolutionary algorithms and the cardinality constrained portfolio selection problem', in *Operations Research Proceedings 2003*, eds., D. Ahr et al., pp. 3–5, Heidelberg, (2003). Springer-Verlag.

[16] J.L Treynor, 'How to rate management of investment funds', *Harvard Business Review*, **43**, 63–75, (1965).

[17] Ganesh Vedarajan, Louis Chi Chan, and David Goldberg, 'Investment portfolio optimization using genetic algorithms', in *Late Breaking Papers at the 1997 Genetic Programming Conference*, pp. 255–263, (1997).

[18] E. Zitzler and L. Thiele, 'Multiobjective optimization using evolutionary algorithms - A comparative case study', in *Parallel Problem Solving from Nature V*, eds., A. E. Eiben et al., volume 1498 of *Lecture Notes in Computer Science*, pp. 292–301, Berlin Heidelberg, (September 1998). Springer-Verlag.

[19] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca, 'Performance Assessment of Multiobjective Optimizers: An Analysis and Review', *IEEE Transactions on Evolutionary Computation*, **7**(2), 117–132, (2003).

[20] Eckart Zitzler, Joshua D. Knowles, and Lothar Thiele, 'Quality assessment of pareto set approximations', in *Multiobjective Optimization, Interactive and Evolutionary Approaches*, eds., Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Roman Slowinski, volume 5252 of *Lecture Notes in Computer Science*, pp. 373–404, Berlin Heidelberg, (2008). Springer-Verlag.

[21] Eckart Zitzler and Simon Künzli, 'Indicator-based selection in multiobjective search', in *Parallel Problem Solving from Nature VIII*, eds., X. Yao et al., volume 3242 of *Lecture Notes in Computer Science*, pp. 832–842, Berlin Heidelberg, (2004). Springer-Verlag.

[22] Eckart Zitzler, Marco Laumanns, and Lothar Thiele, 'SPEA2: Improving the strength pareto evolutionary algorithm', in *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, eds., K. Giannakoglou et al., pp. 95–100, Athens, Greece, (2002).

[23] Eckart Zitzler and Lothar Thiele, 'Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach', *IEEE Transactions on Evolutionary Computation*, **3**(4), 257–271, (1999).

Gene Expression Classification using Multi-Objective Ensembles

Ed Keedwell[1] and Ajit Narayanan[2]

Abstract. The classification of microarray data is an important application of computational intelligence techniques. It is important for the techniques chosen to construct models that are accurate, parsimonious and explicable. This paper investigates the use of multi-objective genetic algorithms to create a number of rules for classifying gene expression data and the combination of these rules into an ensemble classifier. The results demonstrate that the method of creating the ensemble is an intrinsic component of the classification process and that the use of ensembles can increase the accuracy of the classification.

1 INTRODUCTION

The advent of microarray technology has allowed biologists an unprecedented view of the genetic and biomolecular mechanisms underlying the behaviour of organisms. A typical microarray is capable of capturing the expression level of up to 30,000 genes and their transcripts at any particular time. The data generated by microarrays, however, has proved somewhat of a challenge for traditional analytical techniques, due largely to the size of the arrays themselves and the inherent combinatorial problems that arise from analysing a dataset with tens of thousands of variables. This paper describes a novel computational technique using multi-objective genetic algorithms for discovering genes that classify samples in large gene expression datasets.

Gene Expression Data Analysis

Generally speaking, microarray experiments fall into three categories: temporal, static and classificatory. Temporal microarray data is created by measuring gene expression values over a number of timesteps, often whilst subjecting the organism to some stimulus, so that the behaviour of genes can be measured to determine possible cause-effect relationships. The goal of analysing such data is to determine the pattern of excitations and inhibitions between genes, gene products and proteins that make up a gene network for that organism through observing the interactions between genes between one timestep and the next . Genetic algorithms [1] and multi-objective genetic algorithms [2] have previously been successfully used for this purpose.

Static microarray data measures just once the gene expression values of a population of individuals. The aim is then to apply various techniques to separate the individuals into mutually exclusive subsets based on dividing the data into gene groupings that share patterns of co-expression. Typically, cluster analysis (hierarchical, k-means) sort samples into groups so that the degree of association between two samples is maximal if they belong to the same group and minimal otherwise. Cluster analysis is widely used to discover structures in data that may then require subsequent analysis and explanation.

Classification microarray data is created by taking an individual sample of a number of individual organisms or tissue samples which differ in some known respect from each other. Classification studies are most often seen in cancer research, where individuals are pre-sampled and separated into classes (typically case and control) according to an independent diagnosis, with membership of each class determined by the pathology of the individuals involved. The task for the analytical technique here is typically to differentiate between those individuals diagnosed with cancer and those without and also to identify a reduced set of genes for doing so, using the gene expression values of each of those individuals and the class to which they belong. Such 'gene reduction' methods are a form of feature selection due to their use of 'original' measurement values. In contrast, typical feature selection methods, such as support vector machines and recursive feature selection methods generally require an explicit search procedure, an evaluation criterion and a stopping criterion for terminating the search procedure [3]. One of the challenges for such techniques is how to identify optimal combinations (subsets) of features that together lead to successful feature selection, given the strategy of examining each feature in turn.

A common problem encountered with these studies is that the measurements gained from these studies often span thousands of genes (fields, attributes) and only a records (time-steps, samples), which makes it relatively unusual in structure and not as amenable to traditional analysis as static biological databases. The difficulty for all methods is 'under-determinism', i.e., there are very many variables in comparison to the number of records available. Another common problem is that, often, combinations of gene are required to correctly determine the class of the sample or the activation of the regulated gene. That is, one gene is unlikely by itself to classify all samples. In such circumstances, the search space grows from thousands of possibilities to a much

[1] School of Engineering, Computing and Mathematics, University of Exeter, Harrison Building, North Park Road, Exeter, EX4 4QF Email:E.C.Keedwell@ex.ac.uk.

[2] School of Computing and Mathematical Sciences, Auckland University of Technology, Auckland 1142, New Zealand. Email: Ajit.Narayanan@aut.ac.nz.

larger potential search space of billions of combinations. It is this 'curse of dimensionality' that causes problems for traditional statistical and algorithmic approaches.

Multi-Objective Evolutionary Computing in Gene Expression Analysis

Recently, 'intelligent' approaches based on machine learning techniques in artificial intelligence have been applied to the problems of microarray analysis (e.g. [4],[5]). The task of such techniques is to adopt traditional machine learning approaches, such as artificial neural networks and genetic algorithms (e.g. [6], [7]), to produce results that are primarily interpretable by experts in the domain rather than significant according to some statistical method. Multi-objective genetic algorithms are well-known for providing a number of solutions to select from and provide a more transparent result for interpretation by the domain expert therefore they represent a good choice of algorithm for use in this domain. Multi-objective genetic algorithms utilise similar genetic operators to their single-objective counterparts, namely a stochastic population generation process, random mutation and recombination, but they differ in the way that solutions are evaluated. Each solution in a MOEA will have multiple fitnesses and the superiority or otherwise of one solution over another is established via the dominance criterion. The algorithm produces a set of solutions which represent the optimal discovered trade-off between two or more objectives and is known as a pareto-front. For this to work effectively, objectives normally need to be antagonistic (e.g. a decrease in one objective incurs an increase in the other) and this is what is represented in the approaches below.

Multi-objective genetic algorithms have been used for deriving gene regulatory networks [2] whereby the connectivity of the network is one objective, and the RMSE of the derived network is the other. The resulting optimisations therefore create a pareto-front that consists of multiple networks that represent a trade-off between parsimony and accuracy. This trade-off is an important idea in science and embodies the principle of Occams' razor, which in this case is the notion that given two regulatory networks with equal accuracy, the network with fewer connections should be selected. The work here uses a similar principle as applied to gene expression classification.

So far, only two systems have been proposed that use MOEAs for the classification of gene expression data. A system known as Memetic NSGA [8] was tested on a small 50 gene leukaemia dataset and was used to demonstrate the efficacy of the algorithm rather than to generate biological information. This is in contrast to the goal of this work which is designed to operate on gene expression datasets of any size. The most widely-cited system is that of Deb and Reddy [9] who proposed a gene expression classification system based on NSGA-II (the same algorithm that is used here) that classifies full gene expression datasets. This algorithm was able to discover small numbers of highly accurate genes. The objectives optimised by the MOEA were f1, the number of genes in the rule, f2, the number of errors in the training data and f3, the number of errors in the test data. These objectives gave rise to a number of rules which were able to classify three biological datasets, namely the leukaemia (ALL-AML) (considered here), colon and lymphoma cancer datasets. The system was shown to be capable of discovering multiple rulesets with 100% accuracy on these datasets and often had less than five genes in the rule. However, the use of the test set in the optimisation procedure, firstly combined into one objective and latterly separated into two objectives removes the independence of that test set. The approach detailed here is significantly different, firstly from an algorithm perspective as this work uses a neural-genetic approach, secondly because the rules are then combined into ensembles for testing and finally because only the training dataset is considered in the computation of the fitness function, ensuring that the test set remains independent from the process of optimisation, a key factor in classification experiments.

Ensemble Classification

Ensemble classification is an established method of combining several independently developed classifiers into one classification mechanism. There is no established method for combining the classifiers into one classification approach but we investigate three popular choices here, using a winner-takes-all approach, a majority-vote approach and an averaged approach. A prerequisite for ensemble classification is that multiple classifiers must be created to be subsequently combined together.

A number of methods have been proposed for the creation of multiple classifiers, the most established of which are bagging and boosting. Bagging [10] involves randomly sampling from the dataset and then creating multiple classifiers from the sampled datasets. Boosting [11] is a sequential technique that creates multiple classifiers based on the errors of those classifiers that precede the current one. More recently, research has been conducted into the process of creating multiple classifiers with the view to them performing optimally as an ensemble. Significant work on the creation of neural network ensembles has been conducted [12] whereby diversity in the classifier is deliberately encouraged to increase performance when the networks are then combined into an ensemble. The same group has also considered the power of using a multi-objective algorithm to develop neural network ensembles [13]. The objectives used are the accuracy and diversity of the networks. It is this is concept that is adopted in this paper, where a number of different objectives are considered.

Furthermore, a good number of approaches to ensemble creation have been applied to microarray classification. A new technique and a review of the current approaches can be seen in [19]. The approaches shown in the papers seen in [19] involve the ensemble of feature selection and classification techniques, often arranged in pairs. This differs significantly from the approach proposed here which simply uses the multi-objective algorithm to generate the rules required for ensemble creation. The use of feature selection methods has been criticised in the past as by necessity, they restrict the space available to the classifier and therefore may remove genes which look unpromising in isolation, but are influential when combined with other genes. Additionally, the

complexity of these techniques both in terms of implementation and computational complexity are likely to exceed that of a multi-objective GA run given that they comprise two separate stages and search the space of combined of classifier and feature-selection algorithms. In the case of [19] there are 6 classifiers and 7 feature-selection algorithms which must be searched combinatorially using a GA and its attendant computational cost before the feature-selection and classifier algorithms themselves are run.

2 METHOD

In 2003, a neural-network and genetic algorithm hybrid for the classification of microarray data was proposed [14]. In this paper, this technique is expanded to utilise a multi-objective algorithm from which a set of rules are created and then combined together to form an ensemble. The neural-genetic technique uses the evolutionary algorithm to select up to *k* genes from the set of possible genes in the data. The training data from these genes is then passed to the neural network along with the classification information. The neural network then adjusts its weights to reduce the error of the classification as much as possible. In this paper, a simple single layer is used to determine the weights, but more complex multi-layer arrangements are possible. The combined gene names and weights can then be printed in the form of rules which can then be easily interpreted by non-technical personnel. For a more in-depth description of the algorithm, readers are directed to [14].

The algorithm used here is significantly different in that it uses a multi-objective genetic algorithm. For this purpose the genetic algorithm has been replaced with NSGA-II [15], one of the most popular and successful elitist multi-objective genetic algorithms in current use. NSGA-II is popular because there are few additional parameters over the standard genetic algorithm and so it can replace the single objective GA without too much further parameter tuning. However, more objectives need to be identified for the problem and these are determined as follows:

2-Objective Example

1. Minimise error – over all samples in the training data
   Where m = number of misclassifications and n is the size of the training set
2. Minimise number of genes in each rule

$$E = \frac{m}{n} \cdot 100$$

3-Objective Example

1. Minimise error
2. Minimise number of genes
3. Maximise diversity – *diversity is determined as the number of genes present in the current chromosome that are not present in the other individuals in the population*

4-Objective Example

1. Maximise sensitivity – TP/(TP+FN)
2. Maximise specificity – FP/(FP+TN)
   Where TP = Number of true positives, FN = Number of false negatives, FP = Number of false positives, TN = Number of true negatives.
3. Maximise diversity
4. Minimise number of genes

The two objective example simply tries to maximise the conflicting requirements of accuracy and parsimony. The three objective example adds to this the dimension of diversity which is a key component of the work in [12] and [13] and allows a comparison to be drawn between the simple approach and one which incorporates a diversity component. There is evidence from [12] and [13] that the introduction of a diversity component improves the performance of the subsequent ensemble. The four objective example augments the three-objective with two established and conflicting accuracy measurements in sensitivity and specificity. This essentially means that the algorithm is maximising the area under the ROC (Receiving Operator Characteristics) curve and the diversity whilst simultaneously minimising the number of genes in each of the rules.

In each example, the algorithm is run for a set number of generations and the resulting pareto set of rules is used to construct the ensemble classifier. An additional further factor in the performance of an ensemble is how the classifiers are combined. Here, three methods are investigated:

Majority Vote – Each rule is given a single vote and the class with the most votes is then returned as the final decision. If there are equal votes for a class then the sum of fitnesses (accuracies) for that class is used to determine the classification.

Average Vote – Each rule is fired and the results averaged. If the average exceeds 0.5 then the classification corresponding to 1 is returned, otherwise the 0 classification is returned.

Weighted Vote – Similar to the average vote, but where the average is weighted according to the accuracy of each rule.

In the following experiments, the genetic algorithm is run 20 times for each set of objectives. For each run, the three methods of combining the rules to make an ensemble are tested against each other for accuracy on the training dataset and a completely separate testing dataset in terms of accuracy.

3 DATA

The two datasets used here are well studied in the gene expression literature. The first of these is known as the ALL-AML dataset and is designed to differentiate between two types of leukaemia, acute lymphoblastic leukaemia (ALL) and acute myeloid leukaemia (AML) and was collected and studied in [16]. The data consists of 7129 genes, and 72 cases, of which 38 are of type B cell ALL, and 34 of type AML. This dataset is designed to discern between these cancers as they present very similar symptoms, but respond

very differently to treatments. This dataset is randomly separated (although preserving the class balance) into a training set of 52 and test set of 20 samples The second dataset is taken from [17] and consists of 70 gene expression values for each of 104 individuals, of whom 73 suffer from myeloma and the remaining 31 are diagnosed as normal. The task is for the classification algorithm to correctly separate the individuals into the appropriate groups using minimal gene sets. A random test set of 40 individuals was used and the remaining 64 used for training.

4 RESULTS

The following experiments are designed to determine whether an ensemble approach can yield better performance on gene expression data. Furthermore, the goal is to determine the feasibility of an ensemble-based neural-genetic method for the classification of data in general. The experimentation is also expected to yield information as to which set of objectives yields rules that perform best as an ensemble, along with which combination method yields the best results.

ALL-AML Dataset

Table 1 – Comparison of the performance of each objective set on the ALL-AML dataset

	Mean Rule Number	Mean Train Error	Mean Test Error
2-Object.	5.38	8.14	4.98
3-Object.	6	8.41	5.08
4-Object.	11.57	8.98	4.84

Table 1 shows the performance of each set of objectives on the ALL-AML problem. As expected, the number of rules in the ensemble increases with an increase in the number of objectives. However this phenomenon is much more obvious with the move from 3 to 4 objectives than from 2 to 3, indicating that the diversity measure does not dramatically

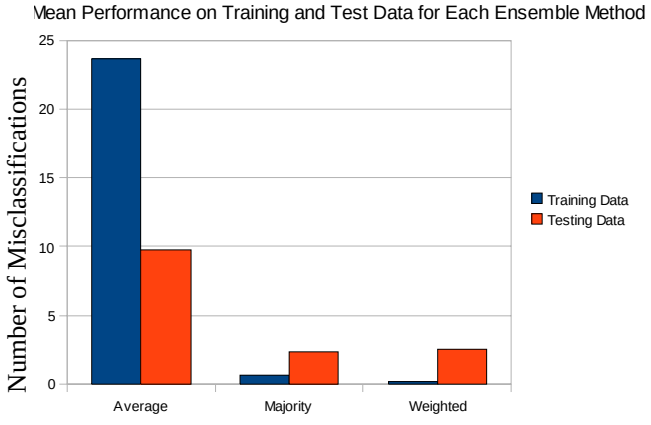

Figure 1 – Comparison of performance for each ensemble technique on the ALL-AML data

increase the number of rules in the ruleset, although it should increase the diversity of the rules. The training and test errors appear to be rather poor, but this is due to the performance of one of the ensemble creation techniques, as we can see in Figure 1.

As can be seen in Figure 1, the average method of combining the pareto-set of rules yields very poor performance. This indicates that the method of averaging the response of each of the rules is by far the least effective here and that the number of rules that fire and their accuracy are important factors in correct classification. Table 2 below shows the errors with the average method removed.

Table 2 – Revised Table with the average ensemble method removed

	Mean Rule Number	Mean Train Error	Mean Test Error
2-Object.	5.38	0.38	2.26
3-Object.	6	0.43	2.61
4-Object.	11.57	0.39	2.45

Table 2 shows the performance of just the weighted and majority vote ensembles on the same data and this indicates that performance particularly on the training data is much improved with less than half an error per ensemble on average. The results shown here therefore result in average testing classification accuracies of 87% (3 objective) to 89% (2 objective). However, individual rulesets achieved 100% accuracy on testing data (see Further Analysis).

Myeloma Dataset

The average ensemble method performed similarly poorly on the myeloma dataset and therefore the following results have had this method removed.

Table 3 - Comparison of the performance of each objective set on the Myeloma dataset with the Average ensemble method removed

	Mean Rule Number	Mean Train Error	Mean Test Error
2-Object.	6	0.86	3.59
3-Object.	6	0.95	3.29
4-Object.	7.6	0.95	3.68

Table 3 shows the relative performance of each set of objectives on the myeloma problem. An interesting property of this particular set of runs is that the 3-objective run did not yield larger rulesets than the 2-objective run in this case. This suggests that again, the diversity of the rules does not influence the generated pareto-front particularly. In fact, the number of rules in the ensemble does not change in the same way for any of the objectives as it did for the ALL-AML

dataset. This is likely to reflect the smaller number of genes in this dataset, which has been preprocessed and therefore only includes a small number of genes that are related to the classification. With relatively few genes to choose from, it is unlikely that rules with many genes will be present in the data and therefore this provides a natural limit to the size of the ruleset. The results shown here therefore result in average testing classification accuracies of 91% (2 objective) to 92% (3 objective). However, individual rulesets achieved 95% accuracy on testing data (see Further Analysis).

Further Analysis

The performance of the ensemble tends to track that of the best performing rule in most instances. There are some occasions where the ensemble improves on the performance of the top performing rule which appear more often in the myeloma dataset than the ALL-AML. In general the ensemble performance on the myeloma dataset is at least as good as the best performing rule and often better than it.

The diversity preservation mechanism did not appear to have the desired effect of creating a number of different individuals with similar classification performance. The aim was to minimise the number of similar genes with other individuals in the whole population but perhaps this measure should be limited to those individuals in the pareto-set to further reduce the number of overlapping genes.

The best rule discovered by the approach classifies both the training and testing data with zero errors on the ALL-AML dataset. This rule states that AML is characterised by the absence of L05148, M89957 and X69111. LO1548 codes for protein-tyrosine kinase and is involved with T-cell regulation [18]. M89957 codes for the cell surface glycoprotein, part of the B-cell receptor complex and therefore has involvement in the immune system. Finally, X69111 codes for the helix-loop-helix protein ID3 which is expressed in blood lymphocytes.

The best rule discovered on the myeloma dataset classifies the training data with zero errors and makes two errors on the testing set. This rule states that myeloma is characterised by the absence of three genes, M63928 a T-cell activation antigen and tumour necrosis factor receptor, X16832 codes for human cathepsin H. A number of cathepsins (G,L and K) have previously been associated with myeloma, but H does not appear to have such a strong biological association. L36033 codes for a pre-B cell stimulating factor which would again appear to have immune system influences.

5 CONCLUSIONS

The neural-genetic method of discovering classification models from gene expression data has been extended to include a multi-objective element. The use of a multi-objective GA allows the algorithm more flexibility in the rules that it creates in that it can balance the requirements for parsimony and accuracy. The pareto set of solutions can then be interrogated by experts in the field and an extra level of confidence can be attributed to a set of similar rules with differing numbers of genes.

The creation of a set of results as opposed to a single rule begs the question of how best to combine the multiple results into a single classifier. This has been investigated here through the creation of an ensemble from the rules generated by the multi-objective GA. Three different methods of combining the rules were investigated and the conclusion can be drawn that the weighted and majority vote approaches performed very similarly in the trials on the data here and both are preferred to the averaging approach which yielded very poor results. An interesting point to note here is that the performance of the ensemble was limited to the performance of the best performing rule for the ALL-AML dataset, but this was not the case for the myeloma dataset where the ensemble performance was often better than the best performing rule. The tentative conclusion can be drawn then that the use of an ensemble is recommended to ensure robust classification over a number of datasets providing that the weighted or majority vote ensembles are used. If this is the case then the evidence suggests that the ensemble will deliver results at least as good as those of the best performing rule and occasionally better depending on the dataset.

Three different sets of objectives were considered to create the ensembles, and the genuinely surprising result is that the number and type of objective had little effect on the resulting accuracy of the ensemble. It was expected that the introduction of a diversity measure would increase the generality of the ensemble and reduce the potential for it to overfit, but there appeared to be little advantage in these areas for the three and four objective examples where the diversity measure was present. The increase in objectives did yield an increase in the number of rules in most cases, but it is not at all evident that the increased number of rules results in more robust classification or any increased protection against overfitting. This is a surprising result and perhaps suggests that certain of the rules enjoy a dominance over the others in the ruleset. This is certainly possible in the weighted voting strategy as rules with good fitnesses will prevail, but the majority vote strategy provides an unbiased view providing that tie-breaks are not routinely required.

Overall, the ensemble system has not yielded dramatic improvements in accuracy over the original multi-objective system. However, the rules generated by the system have been shown to be both accurate (zero errors for ALL-AML and two errors for myeloma datasets) and to have some biological plausibility according to the current known function of these genes.

REFERENCES

[1] Keedwell, E., Narayanan, A., (2005) "Discovering Gene Regulatory Networks with a Neural-Genetic Hybrid" IEEE/ACM Transactions on Computational Biology and Bioinformatics, July-September 2005, Vol 2., No.3, pp 231-243, IEEE Computer Society

[2] Spieth C, Streichert F., Speer N., and Zell, A., (2005) “Multi-Objective Model Optimization for Inferring Gene Regulatory

Networks" in the proceedings of Conference on Evolutionary Multi-Criterion Optimization (EMO) 2005 Guanajuato, Mexico, 2005. LNCS 3410, pp. 607-620, Springer-Verlag

[3] Dash, M and Liu, H (1997) Feature Selection for Classification Intelligent Data AnalysisVol 1., pp131-156

[4] Keedwell, E.C. and Narayanan, A. (2005). *Intelligent Bioinformatics: The Application of Artificial Intelligence Techniques to Bioinformatics Problems.* Wiley.

[5] Fogel, G.B., Corne, D.W. and Pan, Y. (2007) *Computational Intelligence in Bioinformatics*. Wiley-IEEE.

[6] Ando, S., Iba H.,. (2001a) "Inference of Gene Regulatory Model by Genetic Algorithms", Proceedings of Conference on Evolutionary Computation 2001 pp712-719

[7] Ando, S., Iba H., (2001b) "The Matrix Modeling of Gene Regulatory Networks -Reverse Engineering by Genetic Algorithms-", Proceedings of Atlantic Symposium on Computational Biology, and Genome Information Systems & Technology 2001.

[8] Praveen K., Sharath S. Rio D'Souza G, K. Chandra Sekaran (2007) "Memetic NSGA A Multi-Objective Genetic Algorithm for Classification of Microarray Data" in the Proceedings of the 15th International Conference on Advanced Computing and Communications, 2007, pp75-80

[9] Deb, K. and Reddy, A.R., (2003) "Classification of two-class cancer data reliably using evolutionary algorithms", BioSystems **72** (2003), p. 111.

[10] Breiman, L.: Bagging predictors. Machine Learning, 24, 123-140 (1996)

[11] Freund, Y. (1995): Boosting a weak learning algorithm by majority. Information and Computation 121, 256-285 (1995).

[12] Brown, G. Xin Yao Wyatt, J. Wersing, H. Sendhoff, B. (2002) Exploiting ensemble diversity for automatic feature extraction Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP '02. Vol. 4, pp 1786- 1790 vol.4

[13] Chandra, A and Xin Yao, "Multi-objective Ensemble Construction, Learning and Evolution," Proc. of the PPSN Workshop on Multi-objective Problem Solving from Nature (part of the 9th International Conference on Parallel Problem Solving from Nature: PPSN-IX), 9 - 13 September 2006, Reykjavik, Iceland.

[14] Keedwell, E.C. and Narayanan, A. (2003) "Genetic algorithms for gene expression analysis" in Applications of Evolutionary Computing LNCS 2611 Gunther Raidl et al (Eds.), proceedings of EvoBIO2003 1st European Workshop on Evolutionary Bioinformatics pp 76-86

[15] Deb. K, Amrit Pratap, Sameer Agarwal and T. Meyarivan (2000). A Fast and Elitist Multi-Objective Genetic Algorithm-NSGA-II. KanGAL Report Number 2000001

[16] Golub T.R., Slonim D. K., Tamayo P., Huard C., Gaasenbeek M., Mesirov J.P., Coller H., Loh M. L. Downing J. R., Caligiuri M. A., Bloomfield C. D., Lander E. S.. (1999) "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring" Science Vol 286 pp 531-536

[17] Page, D., Zhan, F., Cussens, J., Waddell, W., Hardin,J., Barlogie,B., Shaughnessy, J. "Comparative Data Mining for Microarrays: A Case Study Based on Multiple Myeloma." Poster presentation at International Conference on Intelligent Systems for Molecular Biology August 3-7, Edmonton, Canada. Technical report available from mwaddell@biostat.wisc.edu

[18] Chan AC, Iwashima M, Turck CW, Weiss ª ZAP-70: a 70 kd protein-tyrosine kinase that associates with the TCR zeta chain. Cell. 1992 Nov 13;71(4):649-62

[19] Kim, K-J and Cho, S-B (2008) "An Evolutionary Algorithm Approach to Optimal Ensemble Classifiers for DNA Microarray Data Analysis" in IEEE Transactions on Evolutionary Computation Vol. 12 No. 3, June 2008.

Adaptive Gene Expression Programming Using a Simple Feedback Heuristic

Jonathan Mwaura [1] **and Ed Keedwell** [1]

Abstract. Gene expression programming has been shown to be an important algorithm in the optimisation of complex systems. However, it has many more operators and parameters than standard genetic programming, each of which needs to be set when applying the algorithm to new problems. In this paper, an adaptive approach for setting probabilities for genetic operators in gene expression programming (GEP) using parameter control is investigated. The adaptive approach implements simple functions that regulate the probabilities of using a genetic operator e.g. mutation, in a given generation based on the comparison between the fitness of the parent organism and child organism in the previous generations. Using this method, it is shown that by using an adaptive approach an increase in the success rate of a run and decrease in the number of generations required in order to achieve success can be achieved.

1 INTRODUCTION

Gene expression programming [1] is a relatively novel evolutionary algorithm which aims to be more biologically plausible than its predecessors genetic programming (GP) and genetic algorithms (GA). It has been shown to be capable of complex applications similar to genetic programming but has been shown to be more efficient due to its genome/phenome distinction [1] which is not present in GP or GA.

It is generally accepted that for these algorithms to work, variation has to occur to bring about diversity in the population. In earlier evolutionary algorithms (e.g. GAs [7] and GP [8]), mutation and cross-over have been used to deliver this variation whereas in Gene Expressing Programming (GEP, [1]) there are a total of seven genetic operators that are used. All these operators depend on probabilities that they will occur in a given run, and the problem is that it is difficult to determine which probabilities will work for every single problem. The evolution of the system therefore, not only depends on what genetic operators are used but also on the probabilities used to determine the occurrence of the genetic variation. Whilst there are occasionally some 'rules of thumb' for parameter settings in standard EAs, there is no clear agreement on what probabilities to set for GEP for use even in the same problem domains.

In this study we propose an adaptive method that fine tunes the genetic operator's probabilities using a feedback heuristic that checks the effect of a probability in a previous generation and adapts it by either increasing or decreasing the probability in the next generation.

[1] School of Engineering, Computer Science and Mathematics, University of Exeter, EX4 4QF, UK. Email:{jm329, E.C.Keedwell}@ex.ac.uk

The remainder of the paper is arranged as follows: GEP concept, related works, methodology and experiments, results, conclusion and future work.

2 GEP CONCEPT

Representation

Gene Expression Programming is a relatively recent genotype/phenotype evolutionary algorithm developed in 2001 [1]. It follows in the footsteps of genetic algorithms and genetic programming in mimicking the Darwinian Theory of evolution to solve complex problems in mathematics, computer science and related fields. GEP starts with forming linear character chromosomes representing the solutions, akin to a standard GA representation, and is referred as the genotype. The linear representation is formed in two domains, i.e. head and tail; the head contains both terminals and functions while the tail contains only terminals. The length of the head is normally provided as a user defined variable during a run while the length of the tail is a function of the head as shown in Equation 1:

$$t = h\,(n-1) + 1 \qquad (1)$$

Where t is the length of the tail, h is the length of the head and n is the maximum number of arguments of any function in the function set. After forming the linear representation the chromosome undergoes translation where the chromosome is decoded to form a coding region, (i.e. the part of the chromosome representing the region that will be used to solve the problem), and a non coding part. The coding region is further translated into a tree-like structure similar in nature to that in GP, this is known as the phenotype and is expressed as a structure known as the Expression Tree (ET). The phenotype can also be expressed as a linear structure known as an Open Reading Frame (ORF) which is similar to natural biology where the ORF covers only the coding region of the genome with non-coding region upstream and downstream. Unlike natural genes though, the ORF in GEP only has non-coding alleles downstream. It is the concept of these coding and non-coding regions that provides the capability of GEP to form correct ETs every time the genotype undergoes a genetic operation. An additional difference between traditional EA representations and GEP chromosomes is that they can be made up of more than one gene, i.e. multigenic chromosomes are possible. The different genes are linked together using a linking function which is chosen prior to a run. This multigenic nature of GEP helps to solve more complex problems by providing new materials every time a run is carried, readers are directed to [1] for more information on GEP.

Operators
Selection is carried out using standard tournament or roulette wheel selection, as is common in GAs and GP. As with GAs and GP, mutation and one and two point recombination are used, however, in addition, GEP also incorporates gene recombination, insertion sequence transposition, root transposition and gene transposition (for a description of how these operators work, please see [1]). These extended techniques ensure that new material is provided in the organisms as the evolution continues and provides a mechanism for sharing information between multiple genes. Regarding selection, roulette wheel, tournament selection and rank based selection can all be used depending on the problem. For the work presented, roulette wheel selection with elitism has been used.

The difficulty with GEP is that whereas there are a number of 'rules of thumb' that exist for the setting of mutation and crossover probabilities in GAs and GP, with seven operators there are no such rules for GEP. With so many operations, many of them potentially able to swap large sections of code and therefore, be somewhat destructive, it will be difficult to find optimal or even reasonable settings for a number of problems. This problem is overcome in this paper with the use of an adaptive technique which is shown to improve the results of the approach and also provides interesting information on the adapted settings throughout a run.

3 RELATED WORK

In GEP, the standard EA method is used whereby a probability is used to determine when each operation in a run will be conducted. In [1] and [2] a probability rate equivalent to two one point mutations is used and a 0.7 probability for one point crossover. In other work, presented in [1], the probabilities implemented are 0.051 for mutation, 0.2 for one point recombination, 0.5 for two point recombination, and 0.1 for all of the other operations. How these probabilities have been pre-selected is not stated, however most of the work carried out in GEP as seen in [2] shows that these probabilities are being used in the majority of applications. It is assumed that some sort of parameter fine tuning has been conducted to arrive at these probabilities as their success in the majority of the work is marked. However, it cannot be assumed that there is always a canonical probability to use for every particular problem at hand; also there is a huge cost of fine tuning parameters particularly when a complex task is being optimised. Adaptive parameter control means that the algorithm can be started with certain probabilities as a seed and during the course of the run the system can be left to regulate them according to fitness success. Using the method shown here; we demonstrate that the choice of the initial probability seed is not a major concern as the system determines finally what works best for a particular problem.
The work in [2] proposes the automatic fine tuning of parameters using GEP as the evolutionary methodology and another meta-heuristic for parameter control, in this case, a GA. This works through reporting success when fine tuning gene head and gene length, but employs a parallel EA run which is computationally costly, and is demonstrated by the requirement of a server and various clients to run the optimization. Note that this was used for fine tuning gene head and length and similar probabilities in [1] have been used with regard to genetic operators.
In [3] cloud control has been used to control and tune only the crossover and mutation probabilities, the study proposes and eliminates what is known as non-valid individuals, i.e. These are child and parent organisms who have same fitness after the genetic variation was carried out, e.g. if Ind1 and Ind2 are parent organisms, after they undergo cross over we have Ind3 and Ind4, If fitness of Ind1=Ind3 and Ind4=Ind4 or Ind1=Ind4, Ind2=Ind3, then the set {Ind1,Ind2,Ind3,Ind4} is termed as a set of non-valid individuals. Though the exclusion of these non-valid individuals results in a faster search and improved search performance, it nevertheless leads to a loss of good building blocks that are likely to be beneficial to the optimisation. Work in [5] Seeks to optimize real parameters using GEP while [4] uses a momentum-based and standard mutation to form a hybrid system that adapts the mutation rate in a GA system.
Therefore despite this existing research, there remains a need for a simple system that can optimise the multiple parameter settings of GEP throughout an optimisation run.

4 METHODOLOGY

Here it is proposed that by automatically fine tuning probabilities as input parameters, better results can be obtained for GEP. A probability of operation determines how likely every operation will be carried out and thus has a direct impact on the success rate of the run.

4.1 Implementing GEP methodology

The basic GEP algorithm and genetic operators underlying our system are implemented following Ferreira's approach [1].
In our investigations, we have used the same symbolic regression and sequence induction equations as used in [1] (see equations 3 and 4 in 4.3 below). Fitness, f_i of an individual i was determined by the absolute error [see [1] 4.1a] as shown in equation two below:

$$F_i = \sum_{j=1}^{Ct} (M - |C_{i,j} - T_j|) \quad (2)$$

Where M is the range of selection, C_t is the number of Fitness cases, $C_{i,j}$ is the fitness of the individual i for fitness case j, and T_j is the target value for fitness case j. In this case the perfect organism will have $C_{i,j} = T_j$ *and* $f_i = f_{max} = C_t \times M$. The error margin is 0.01 and effectively means that if the distance between C_j and T_j is less than this figure, M is returned as the fitness.
The roulette and tournament selection were both implemented, although to permit comparison with results in [1] all experiments were conducted using roulette wheel with elitism, and cloning the best individual in the previous generation.
The runs were started using different seeds for the operator probabilities, but a set of runs was also conducted using the seed probabilities suggested in [1]. Regarding the stopping criteria, the number of generations (please refer to table 1 for the number of generations used) was used to stop the algorithm or when an organism with maximum fitness was achieved (in symbolic regression problem maximum fitness is 1000 while in the sequence induction, maximum fitness is 200).

To represent chromosomes an array of strings was used, when using multigenic chromosomes the linking function was specified beforehand. In this case the addition function was used for linking the genes.

4.2 Feedback heuristics

A generational approach was used in running the GEP algorithm. In this case there are (n-1) new organisms produced and taken to the next generation, where n is the total population.
A counter was implemented for each of the genetic operators used in GEP. After each operational run, i.e. a run in one generation (for a population of size n, there are n/2 operational runs due to crossover providing 2 children), the counters are updated depending on whether fitness for the child organism has been improved or not, (i.e. one is added to the counter if fitness is improved and one is subtracted if fitness is lowered).
Note that, the fitness of the new organism is calculated after all the genetic operations have been carried out as determined by the probability. Since a probability is used to determine whether a certain genetic operation will be carried out, it means some may not be carried out at all. In the case where the operation did not occur for an individual, there is no effect on the particular counter for the genetic operator in question.
These counters run for the entire generation, after which, the probabilities for genetic operations are updated depending on whether the counter is positive or negative (i.e. accumulated negatives would mean that the operation had a negative effect on multiple organisms). Operations that generally lead to improved fitness are encouraged by increasing their probability and operations that generally provide worse solutions are discouraged by reducing their probability in the next generation. The new probabilities for the genetic operators are used in the subsequent generation. This is repeated until the stop criterion is met. We update the probabilities in small steps as shown by the pseudocode below;
Consider a probability for Mutation; we have kept a counter, mutationCounter

```
gepAdaptMut(mutationCounter)
{
       if(mutationCounter >0)
              proMutation=(proMutation      +
(proMutation*0.2));
       if(mutationCounter <0)
              proMutation=(proMutation      -
(proMutation*0.2));
       if(probability_Mutation>maximum_Prob
ability)
              proMutation=(proMutation      -
(proMutation*0.2));
       if(probability_Mutation<=minimum_Pro
bability)
              proMutation=(proMutation      +
(proMutation*0.2));
}
```

Where maximum probability is set as 1 and minimum probability is set as 0.The counter is updated during the course of operational run.

4.3Experiments

Two functions are optimised as follows:

i) A fourth order symbolic regression,[1].
i.e.
$F(a) = a^4 + a^3 + a^2 + a^1$ (3)
Ten values of (a) are given and the corresponding values of f (a), please see [1] for details.
In the first experiment we tested and compared an adaptive GEP using all the genetic operators, with the parameters set as shown in Table 1,Exp 1. This was compared to a standard run. This is a case for a unigenic organism.
In the second experiment a standard run is conducted with all genetic operators and an adaptive run where all genetic operators are adapted. This is a case for a multigenic organism
The Functions and Terminals are set up as in [1].

ii) A Sequence Induction problem given in [1]
i.e. $N=5a^4 + 4a^3 + 3a^2 + 2a + 1$ (4)
Ferreira in [1] uses this problem to show how GEP creates constants . In our case since our aim is to show the effect of adaptation, we use the same problem and the constants were created from scratch by the algorithm, i.e. we do not use ephemeral random constants. This an important capability of GEP as in most problems either electrical or mathematical, the constants needed are not known *a priori*.
In the first experiment an adaptive GEP with Mutation (Pm= 0.022) and one point crossover (P=0.7) and a unigenic gene with h=6 and no of genes =7 was compared to a standard run.
This same experiment was then conducted using all genetic operators, adapting all of them and then compared with a standard run.
The addition function was used as the linking function for all experiments with more than one gene.

Table 1. Parameters used in the experiment involving Symbolic Regression and Sequence Induction

	Exp 1	Exp 2	Exp 3	Exp 4
Population	30	30	50	50
Generations	50	50	100	100
Function Set	/,*,-,+	/,*,-,+	/,*,-,+	/,*,-,+
Terminal set	a	a	a	a
Head Length	24	6	6	6
Number of genes	1	3	7	3
Mutation Rate	0.051	0.051	0.022	0.3
One Point Cross-over rate	0.2	0.2	0.7	0.7
Two Point Cross-over rate	0.5	0.5	0.1	0.2
Gene Cross-over rate	0.1	0.1	0.1	0.1
IS Transposition rate	0.1	0.1	0.1	0.1
Ris Transposition rate	0.1	0.1	0.1	0.1
Gene Transposition Rate	0.1	0.1	0.1	0.1

Table 1 shows all parameters used for the runs for both experiments. Please note that for all standard runs the actual parameters shown on the table are used whereas in the adapting case, this parameters act as a seed.

5 RESULTS

In each of the following experiments, the system was run 100 times with two features recorded; the success rate (expressed as a percentage) which is calculated as the number of runs which resulted in the correct result, and the average number of generations required to obtain a given result. This is indicative of the performance of the system as those runs which have found the optimal result will be terminated, whereas those which do not are restricted to the generation limit.

Fig1 compares a standard run using all genetic operators as determined by the probabilities in Table 1, Exp 1 and compares it with same probabilities used as initial probabilities for adaptation. This is a unigenic case.
Fig 2 compares a standard run using all the seven genetic operators probabilities a seen in Table 1, Exp 2 and compares this to a run with the genetic operators probabilities adapting with generations. This is a multigenic case.
Fig 3 compares a standard run using all the seven genetic operators probabilities as seen in Table 1 and compares this to a run with all the genetic operators probabilities adapting with generations, this is done in the case of sequence induction.
Fig 4 shows the effect of probability rate as they adapt during the course of one run. Note the oscillatory dynamics we get as the run progresses. This is a single multigenic run using a P_m =0.051, P1r =0.2, P_{2r}=0.5, P_{gr}, P_{is}, P_{ris}, P_{gtrn} =0.1 (table 1, Exp 2)
Fig 5 shows the effect of probability rate as they adapt during the course of one run. In this case we repeat use the same set up as used in Fig 4 but with different initial probabilities, see table 1,Exp 4.

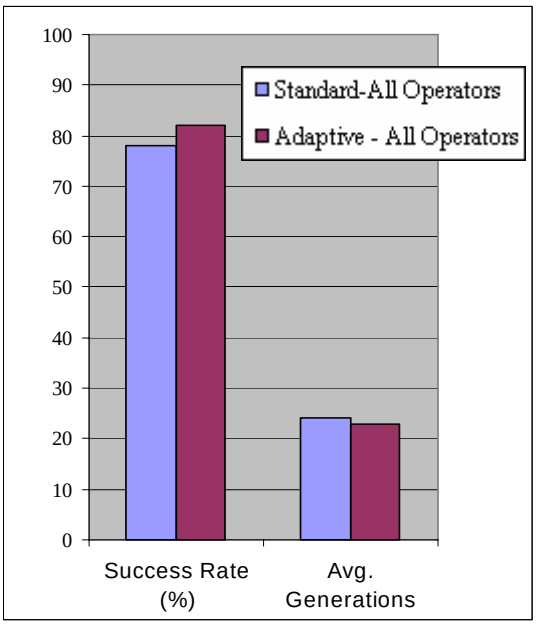

Fig.1. Comparison of success rate and average number of fitness of the population using parameters undergoing adaptation versus standard/fixed ones in unigenic organisms on the symbolic Regression problem

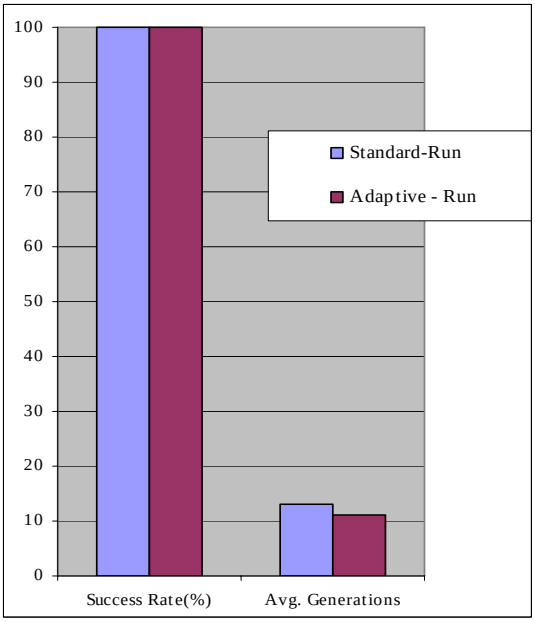

Fig.2. Comparison of success rate and average number of fitness of the population using parameters undergoing adaptation versus standard/fixed ones in multigenic organisms on the symbolic regression problem.

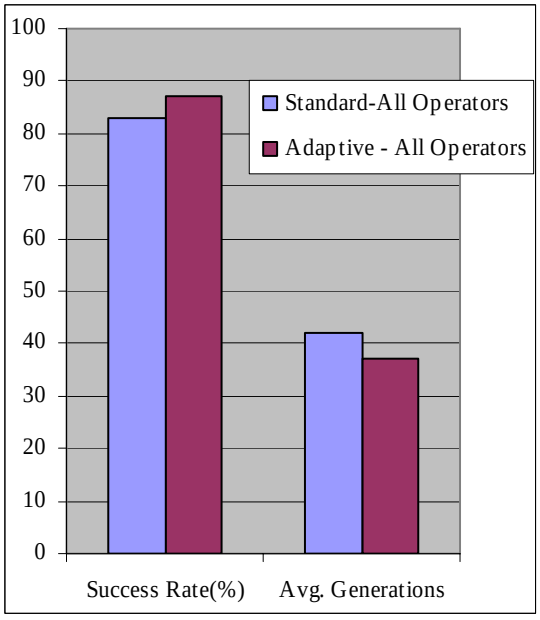

Fig.3. Comparison of success rate and average number of fitness of the population using parameters undergoing adaptation versus standard/fixed ones in multigenic organisms on the sequence induction problem

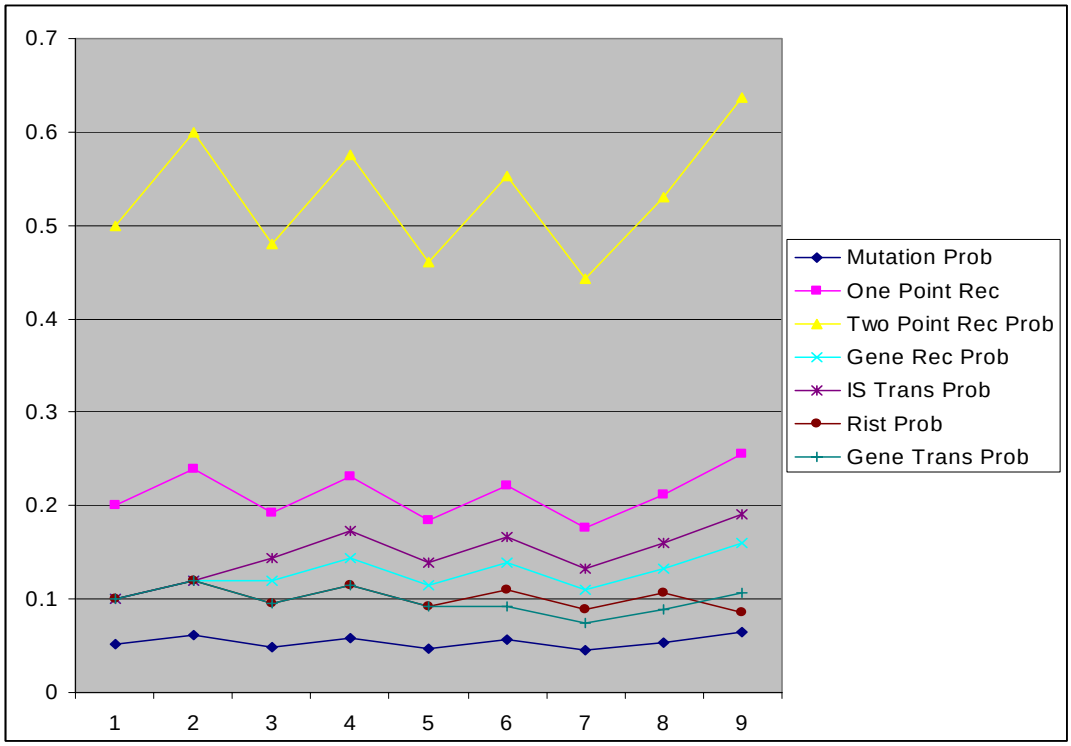

Fig.4. Progression of adaptation of probabilities of genetic operators. A typical case in Experiment 2.

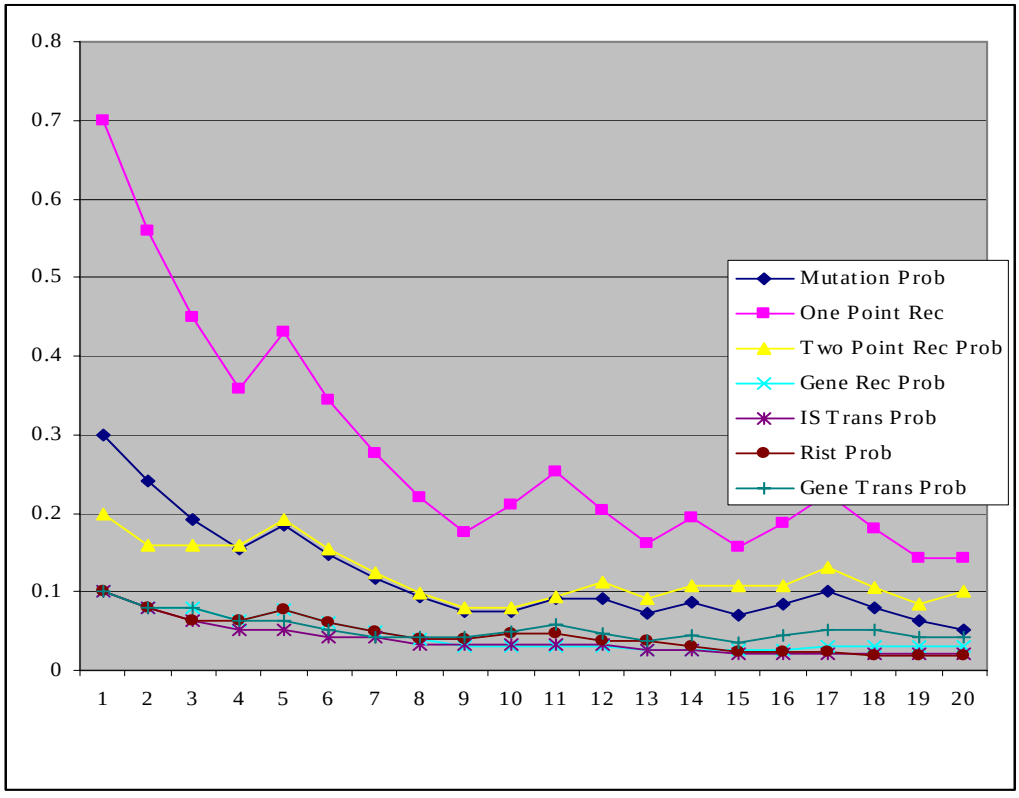

Fig.5. Progression of adaptation of probabilities of genetic operators. A typical case in Experiment 3.Please note this is a similar set up with Fig 4 above but with changed initial genetic operators' probabilities.

6 DISCUSSION

The study investigates a method of tuning input parameters during the course of the run to try to counter the problem of starting with parameters that do not suit the problem at hand; rarely is there a case where the optimal parameters to use are known *a priori* before solving the problem. As shown from the results obtained using this method, the system is able to vary the parameter settings in such a way that performance in improved over the standard run by 5% in terms of success rate, also comparing with the average generations required to get a successful organism; the method decreases this by more than 5%.

With adaptive parameters we can also infer that the algorithm avoids the problem of descending into local optima, as the increasing or decrease of the probability rate affects the search space. The results show that although convergence speed is increased, the quality of results does not suffer, indicating that the systems does not descend into local minima. This might be due to the idea that an increase or decrease in probability will tend to improve diversity in the population thereby giving better models every time a good organism is obtained.

Figures 4 and 5 shows how the adaptive algorithm works with different initial probabilities. Starting with small probabilities, the probabilities increase with small steps as the organisms formed are better than in previous generations, but as the algorithm progresses through the generations, organisms become better and we have constant probabilities. It would appear that the majority of the operators have a level at which they are most effective. This is best seen with the mutation which, despite starting at 0.3 (or 30%) in Figure 5 diminishes to a more plausible value of ~0.05. This trend is also seen with the one point crossover which has a tendency to descend and hover around the 0.2 mark despite the seed value provided. The main counter-example to this is two-point recombination which appears to remain constant in this single experiment, an observation which is not repeated when the averages are plotted (not shown).

Starting with higher initial probabilities, organisms formed could be worse than those existing, this is generally because of the deleterious effects of per-gene mutation in particular, the probabilities as seen in Fig 5 will therefore tend to decrease sharply and as generations progresses, they become more constant.

These two experiments also show how the effect of all the genetic operators working together. An increase or decrease in any of the probabilities means the other operators get affected, i.e. if the effect of mutation is negative, the algorithm may opt to use insertion sequence transposition or root transposition or any other operator which appear to guide the evolution to better solutions. This in turn creates the oscillations as shown in Figs 4 and 5 above.

7 Conclusions and future work

All the genetic operators in the standard GEP affect the convergence and success rate of the evolution. In this study an adaptive method is proposed to adapt the set of probabilities rates used in any given problem with an aim of avoiding local optima as well as increasing performance and reducing computing time. We have shown that this works by providing a simple feedback heuristic in the standard algorithm.

In our future work we seek to extend this study by combining this with an adaptation of the number of genes as well as the gene length.

References:

[1]. Ferreira, C. (2001) *Gene Expression programming: A new Adaptive Algorithm for Solving Problems*. Complex Systems, 13(2): 87-129

[2]. Park, Ho-Hyun et el (2008). *Parallel hybrid evolutionary Computation: Automatic tuning of parameters for parallel gene expression programming*. Science Direct. Applied Mathematics and Computation 201: 108-120.

[3]. Jiang, Yue et el (2008). *Adaptive Gene Expression Programming Algorithm Based on Cloud Model*. International Conference on BioMedical Engineering and Informatics. IEE computer society (2008) doi 10.1109/BMEI.2008.42

[4]. Temby,L. Vamplew,P. Berry, A (2005). *Accelerating Real-Valued Genetic Algorithms Using Mutation-With-Momentum*. The 18th Australian Joint Conference on Artificial Intelligence, Sydney, Australia, 5-9 Dec 2005

[5]. K, Xu et. el. (2008). *A novel method for real parameter optimization based on Gene Expression Programming*. Applied soft computing journal 2008),doi:10.1016/j.asoc.2008.09.007

[6]. Ferreira, C (2002). *Analyzing the Founder effect in simulated evolutionary processes using gene expression programming*. In A. Abraham, J. Ruiz-de-Solar, and M. Koppen (eds), Soft Computing Systems: Design, Management and Applications, pp. 153-162, IOS Press, Netherlands.

[7]. Goldberg D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley Longman Inc.

[8]. Koza, R.J (1992). *Genetic Programming: on the Programming of Computers by Means of natural Selection*, The MIT Press, Cambridge, MA.

A Multi-Objective Evolutionary Algorithm for Portfolio Optimisation

Noël-Ann Bradshaw and **Chris Walshaw** and **Constantinos Ierotheou** and **A. Kevin Parrott** [1]

Abstract. The use of heuristic evolutionary algorithms to address the problem of portfolio optimisation has been well documented. In order to decide which assets to invest in and how much to invest, one needs to assess the potential risk and return of different portfolios. This problem is ideal for solving using a Multi-Objective Evolutionary Algorithm (MOEA) that maximises return and minimises risk. We are working on a new MOEA loosely based on Zitzler's Strength Pareto Evolutionary Algorithm (SPEA2) [20] using Value at Risk (VaR) as the risk constraint. This algorithm currently uses a dynamic population in order to overcome the problem of loosing solutions. We are also investigating a dynamic diversity and density operator.

1 Introduction

Allocating one's assets optimally in a portfolio is the goal of every investor.

Much of the theory of portfolio optimisation is detailed in Markowitz [17]. Investors need to balance the objective of maximising the return on their investment with the constraint of minimising the risk involved. It is generally accepted that the greater the expected return, the greater the risk.

Volatility is the standard measure of risk used by Markowitz and others but the drawback to this approach is that volatility is the measure of share price fluctuation without reference to direction. Thus we propose to use VaR as the measure of risk.

Portfolio optimisation problems can be shown to be NP-Hard [10] and thus heuristic algorithms are a typical way to address them. Gilli and Këllezi [14] used Threshold Accepting and Maringer [16] classifies a number of other single-objective heuristic algorithms. Typically these approaches aim to maximise the return with the constraint that the risk should not be above a given level.

This paper, alongside work by Armañanzas and Lozano [4] and Chiam et al. [8], recasts the problem with a multi-objective approach where the maximisation of return and minimisation of risk are treated as two separate objectives. These are then combined into a single fitness function with each objective given equal weighting.

The use of a multi-objective algorithm allows several solutions to be produced for the same problem but with different attributes [11]. Thus at a glance it is possible to see how much return one gains in comparison to the level of risk incurred. With large complex portfolios this seems to be a promising approach to investigate and we propose using a MOEA similar to SPEA2 to address it.

The rest of this paper is organized as follows: section 2 recounts the background behind portfolio management and the various measures of risk, section 3 looks at optimisation methods such as multi-objective and genetic algorithms, section 4 describes our algorithm, section 5 presents our results and finally, section 6 provides the conclusion.

[1] University of Greenwich, UK, email: n.bradshaw@gre.ac.uk

2 Portfolio Management

2.1 Background

Portfolio management is concerned with choosing shares that will yield the best return over a given period of time. Markowitz [17] shows that a wide diversification of shares will tend to give the maximum return with the least amount of risk because shares from similar companies might behave in the same way, yielding similar risks and returns at similar times. If there is diversification in the portfolio then it is less likely to have periods of high return followed by periods of high risk.

It is important to note that the risk of the overall portfolio (the combined assets) is less than the average risk of the individual assets. Markowitz's model of portfolio optimisation provides a detailed algorithm that maximizes the return and minimises the risk. However, this model is impractical when looking at large numbers of assets. Thus alternative approaches need to be sought.

2.2 Risk Measures

Risk can be used to represent what is unknown about a portfolio. In this context it can mean the chance of the value of the shares decreasing or increasing. Traditionally the measurement of risk used is volatility. Volatility is the measure of how the value of the asset changes and can thus be positive or negative depending on price movement.

This paper uses risk purely as a negative measure. In other words what could go wrong and how much can be lost or 'how bad can things get?' [15] There are a number of different ways of determining risk; we consider the Value at Risk approach.

2.2.1 Value at Risk

VaR was recommended as the appropriate measure of financial risk by the Basel Accord Amendment [1] in 1996 and has been heavily promoted by J.P Morgan in RiskMetrics™Technical Document [2]. Since then it has become widely used by banks, securities firms, commodity merchants and other trading organisations. According to RiskMetrics™[2], 'VaR is a measure of the maximum potential change in value of a portfolio of financial instruments with a given probability over a pre-set horizon'.

Its popularity is due to it being easy to understand: it reduces the level of risk to a single number which can be easily compared. Hull

[15] shows this clearly; for example it might be stated that a particular portfolio has a 1-day VaR of $1 million at the 99% confidence level. This means that there is a 1% chance that the value of the portfolio will decrease by $1 million or more during one day.

However it has drawbacks; the major one being that it assumes the share price returns follow a normal distribution which is not always accurate and may result in the risks being under-estimated [16].

For this reason it is probably advisable to also look at Conditional Value at Risk (CVaR), also known as Expected Tail Loss (ETL) and Expected Shortfall (ES), which is the expected risk in the worst $x\%$ of cases (see below).

VaR describes the expected loss of a portfolio with a given probability. There are three main ways to calculate it, this paper uses the simplest and most usual method: the variance covariance approach.

2.2.2 Variance Covariance Method

As the name suggests this method involves constructing a large covariance matrix of all the assets using data from the variances. It is the approach used in RiskMetrics™[2].

The variances can be computed using a variety of methods including historic volatility, EWMA and GARCH (1,1) which are detailed by Hull [15].

The main assumption that has to be made in order to compute this method is that the risk factor returns are normally distributed. This is equivalent to saying that over a period of h days we have profit / loss of

$$\Delta R_t = R_{t+h} - R_t \tag{1}$$

where R_t indicates the return at time t and ΔR_t the total change, i.e. profit or loss over this period then,

$$\Delta_h R_t \sim N(\mu, \sigma_t^2) \tag{2}$$

In other words over h days the returns follow a normal distribution with mean μ and standard deviation σ.

To compute VaR for a portfolio we take the covariance matrix C of all the assets and the vector of weights v. Using the following formula we find the variance σ_t^2.

$$\sigma_t = \sqrt{(v'Cv)} \tag{3}$$

This in turn allows us to compute VaR using the Z values from the tables for the normal distribution:

$$\text{VaR} = Z_\alpha \sigma_t \tag{4}$$

Values for Z will change depending on the chosen value for α. It is usual to take $\alpha = 5\%$ but this can vary depending on which level of confidence one uses.

2.2.3 Conditional Value at Risk

CVaR can be considered to be an extension of VaR. The problem with the VaR model described above is that the value in the tail might be higher than anticipated.

CVaR was introduced by Artzner et al. [5]. Whereas VaR finds the minimum level of loss that can be expected, CVaR finds the expected or average loss, given that a loss has occurred. Thus the value of CVaR will be at least as large as VaR and often larger, informing investors the extent of the risk [3].

Although CVaR is not used in this paper, its usefulness as a measure of risk will be investigated as part of the further work.

2.3 Summary

This section has provided some of the background information on portfolio management in connection with calculating risk.

3 Optimization Methods

Heuristic optimisation methods are necessary where problems are NP-hard and would take an unreasonable amount of computational time to solve exactly [16].

3.1 Evolutionary Algorithms

The Genetic Algorithm (GA) was first developed by John Holland in the 1960's [6]. It uses the process of Natural Selection to find the optimal solution through a series of populations that combine and mutate solutions until the optimal is reached. Different optimisation problems require different methods of mutating and combining solutions.

GAs come under one branch of 'nature inspired' heuristic algorithms that are known generally as Evolutionary Algorithms (EAs). EAs allow for more flexibility than following exactly gene-based rules and can, for example, accommodate more than one objective function [11].

One of the key aspects of the EA is identifying the fitness function. The level of fitness of a solution determines whether that solution remains in the population or not. Thus the fitness function must accurately describe the objective.

3.2 Multi-Objective Evolutionary Algorithms

A major difference between single-objective optimisation and multi-objective optimisation is that in the former we obtain a single solution and in the later we have a number of non-dominated solutions [11].

If both objectives are of equal importance it is not possible to say which solution is best as there is always a trade-off between a solution being good for one objective and less good for another. However if one is of greater concern than another it might be more obvious which solutions are best.

The use of MOEAs began with David Schaffer's thesis in 1984 [18] which suggested a way to use a GA to capture multiple trade-off solutions. These have been adapted and since the mid-1990's have been used more widely including areas such as scheduling and routing problems, and finance and economics [11]. General applications of MOEAs have been surveyed by Coello Coello and Lamont [9] and more specifically within the area of finance by Castillo Tapia and Coello Coello [7].

MOEAs present the user with a range of solutions that they can then choose from. For example with portfolio optimisation there will be some solutions with a high rate of return and a high risk and others with a low risk and a low return. Thus an investor will have to decide what sort of trade-off between risk and return they are prepared to accept.

The solutions produced by a MOEA are usually bounded by a curve, known as the Pareto-optimal front. The solutions lying on this curve are known as Pareto-optimal solutions [11].

There are several well documented and tested MOEAs [12] [20]. Some of these have also been used to solve the Portfolio Optimisation problem. Skolpadungket et al [19] found that SPEA2 out-performed other algorithms. Our proposed algorithm is loosely based on SPEA2 but incorporates a dynamic population in order to include all non-dominated solutions as well as the highest ranked solutions.

3.3 Summary

This section has outlined the importance and basics of evolutionary heuristic optimization methods.

4 Method

4.1 A MOEA for Portfolio Optimization

We are working on an algorithm which uses a dynamic diversity operator. Currently however, the code is in prototype form and to date only a dynamic population mechanism has been incorporated.

Portfolio optimisation seeks to maximise the portfolio return and minimise the risk. In order to implement this we need to either maximise the return and negative risk or conversely minimise the risk and negative return. This is known as the Duality Principle and is discussed in Fryer [13].

The algorithm selects combinations of different weights of different assets and allocates them to a portfolio. It computes the total return of this portfolio and the portfolio VaR and then sorts solutions with regard to their fitness. Return is calculated by:

$$R_i = \sum_{j=1}^{n} w_j r_j \tag{5}$$

where w_j is the weight of each asset and r_j is the return associated with each asset.

Risk is calculated using VaR as in equation 4 with $\alpha = 5\%$.

Currently the solutions chosen to remain in the population are computed by sorting the population to a single valued fitness function. This fitness function consists of taking the normalised value of the portfolio return less the normalised value for the portfolio VaR. They are then sorted and the fittest ones saved into the population for the next generation.

$$\text{Max}||R_i|| - ||V_i|| \tag{6}$$

where $||R_i||$ is the total normalised return and $||V_i||$ is the total normalised risk.

To avoid losing Pareto-optimal solutions which (under this criteria) might not be the fittest solutions, the solutions are also sorted into dominated and non-dominated arrays based on their values both for total return and also for VaR. If necessary, the population is then increased each generation to include all non-dominated solutions. The number of mutations and combinations is also increased as a percentage of the entire population.

Because this algorithm is multi-objective there is no one solution but a range of possible solutions. The 'best' one will depend on whether an investor wants to obtain maximum return which will have a higher risk or wants to decrease risk and take a lower return.

4.2 Crossover Operator

The problem of portfolio management necessitates a different process for combining and mutating solutions. The weights of the shares in the portfolio are the variables that are being subjected to change in these processes. It is necessary to ensure that the total weight is kept the same. This is not easy to do with a conventional combination method so a new combination algorithm that allows for a small mutation has been designed. This means that two different weight solutions combine but then a random weight is mutated to ensure that the total weights add up to the same amount. The example in Figure 1 helps to illustrate this. Here we are taking the total weight as 50 and only using five assets.

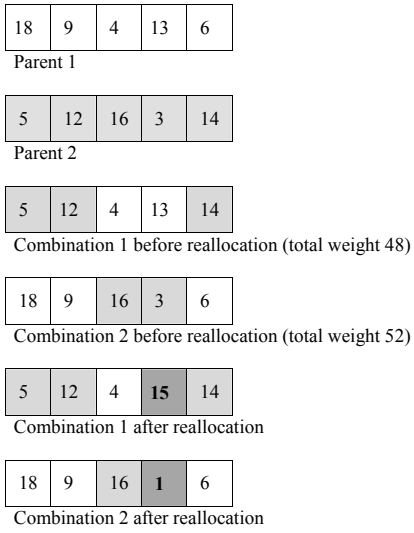

Figure 1. Example of the combination algorithm

This shows that combination 1 and 2 are initially formed of weights taken randomly from parent 1 and 2. This will mean that the total weight does not add up to 50. So another of the weights is chosen at random and an amount is either removed or added as appropriate. The code will also choose multiple weights at random in the case where the first weight chosen is not sufficient to re-balance the total weight of the combination.

This is in contrast to the crossover operation described in Chiam et al. [8]. Here only one solution is used in the operation and only the order of the weights is changed. This would seem to limit the diversity of the population.

4.3 Mutation Operator

A separate mutation operator is also needed to ensure that the population is as diverse as possible and thus a large part of the solution space is examined. This operator takes the weights and identifies one to mutate at random. A random percentage change is subtracted from this random weight. Another random weight is selected and this change is added to it thus ensuring that the total weight is kept the same. In Figure 2 we have an example of the algorithm selecting a vector of weights at random. It then chooses a non-zero cell at random, the third cell in the above example. It calculates a percentage of this and returns a whole number, which in this case is 7. This amount '7' is taken off the first cell and then added to another random cell which in this example is the last one.

This again differs from Chiam et al. [8] as their algorithm swaps two cells rather than adding and subtracting a random amount. We believe that our approach increases diversity, something that is necessary in these algorithms.

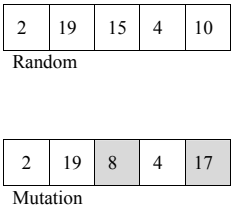

Figure 2. Example of the mutation algorithm

4.4 Summary

This section details the changes required between a standard MOEA and the one used in relation to the specific problem of portfolio optimisation.

5 Preliminary Results

5.1 Test Bed

In order to test the MOEA we have set up a case study with a limited number of shares. Ultimately we would hope to use all the shares in the S&P100 index and create an algorithm that picked those with the best investment potential. This case study uses the assets shown in table 1. The final work will use further test data from several exchanges.

Table 1. Table to show assets used by name and S&P abbreviation

Name	Abbreviation
Citigroup Inc	C
Coca Cola Co (The)	KO
Bank of America Cp	BAC
Ford Motor Co	F
General Electric	GE
Apple Inc	AAPL
International Business Machines	IBM
Oracle Corp	ORCL
McDonald's Cp	MAC
Xerox Cp	XRX
Wal-Mart Stores Inc	WMT
Regions Financial Cp	RF
Procter and Gamble Co	PG
Pfizer Inc	PFE
Microsoft	MSFT
3M Company	MMM
Hewlet Pacard Co	HPQ
JP Mporgan	JPM
Dell Inc	DELL
Walt Disney-Disney Cp	DIS

The MOEA was run with data from these assets between January 2000 and December 2006. This period was chosen as it was also used for our previous work on EAs for finding trading rules and research into volatility in the emerging market using assets from the stock exchange in Mauritius.

The algorithm was run several times using varying values for the initial population size (50,100 and 500) and number of generations (from 100 to 1000). The initial number of mutations and combinations were set as 50% of the initial population and then grew as the population increased. The confidence interval for VaR was kept at 5% but can be altered, as can the total weight of the portfolio.

The greater the initial population and the more generations that were run the longer the computational time. However, although numbers of solutions increased, the general span of the non-dominated solutions and the Pareto-optimal curve remained approximately the same.

5.2 Results

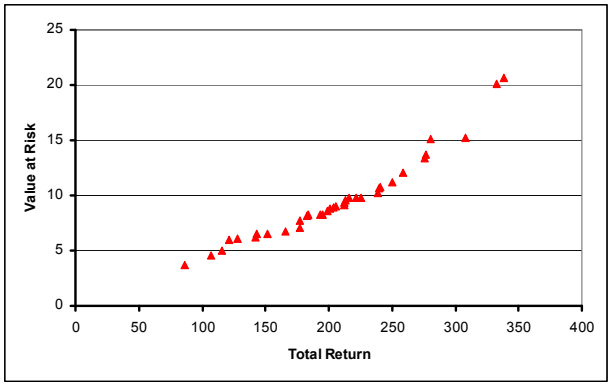

Figure 3. Results using an initial population of 50 with 200 generations

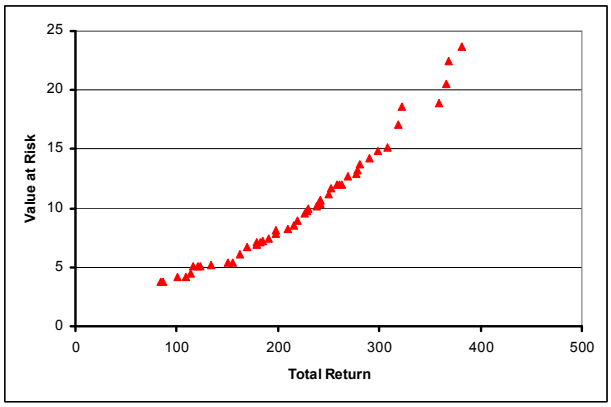

Figure 4. Results using an initial population of 50 with 500 generations

We can see in Figure 3 that we have a reasonably diverse population. The lowest VaR value of 3.7 corresponds to a return of 86, whereas to obtain a higher return of 338 a considerably higher risk of 20.6 is incurred. Although there are a greater number of solutions in Figure 4 there is little change except that there are now an increased

number of solutions lying on the Pareto-front. There are also more solutions with a high return and high risk measurement. Maximum return is now 381 and maximum risk 23.6.

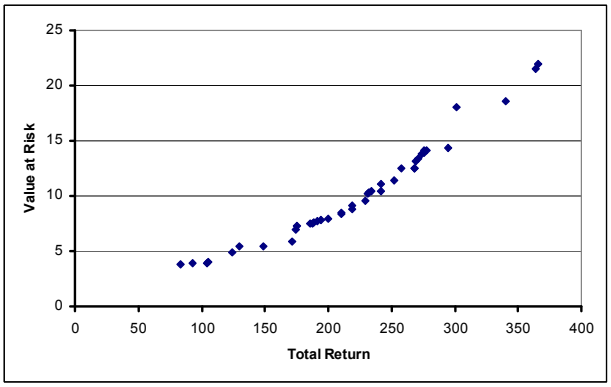

Figure 5. Results using an initial population of 100 with 200 generations

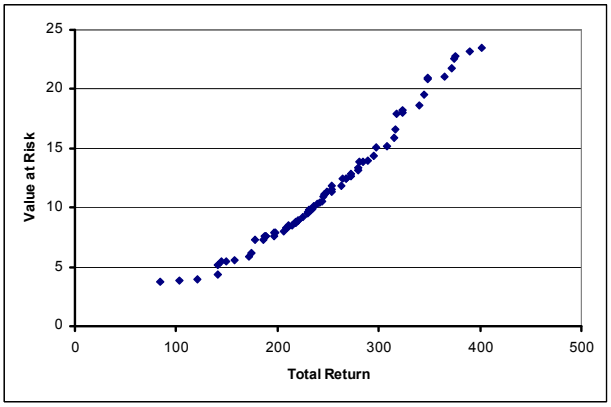

Figure 6. Results using an initial population of 100 with 500 generations

Increasing the number in the initial population makes little difference to the results except to effect the amount of time the algorithm takes to run. It can be seen in Figure 8 that there is not much difference in computational time between initial populations of 50 and 100 (merely a matter of seconds) but this lengthens considerably when increased to 500. Final population sizes are also vastly different as can be seen in table 2.

Table 2. Table to show final population size when running the algorithm for 200 generations for different initial population sizes.

Initial Population Size	Final Population Size
50	4366
100	10084
500	36786

When the algorithm is run using an initial population of 500 we obtain a similar solution to Figure 6 but now with only 200 generations. We have a maximum return of 393 corresponding to a maximum return of 21.6 and a minimum return of 102 corresponding to VaR of 3.6.

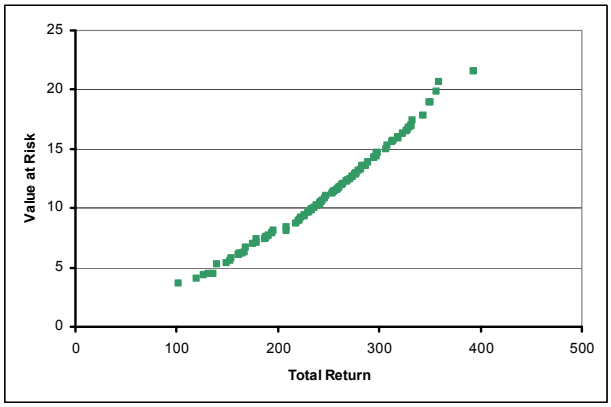

Figure 7. Results using an initial population of 500 with 200 generations

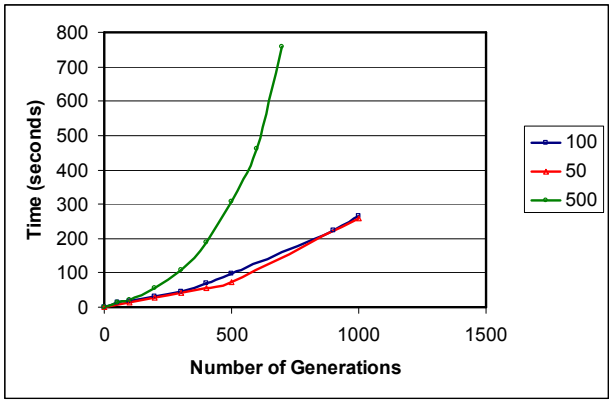

Figure 8. Graph to show how the computational time increases relative to the number of generations used with initial population sizes of 50, 100 and 500.

This test case indicates that using a dynamic population means that a small initial population size has little effect on both the results produced or the run-time. The run-time increases considerably with a large initial population and the results are not significantly improved. Obviously the number of generations also has an effect on the run-time but with a small initial population this is fairly small. The results show that the algorithm manages to produce very good results with only 200 generations after only a short amount of time; additional generations improve the result slightly, but perhaps not significantly.

In order to examine this further a new test bed was put together using data from the same companies but this time from 1990-1999. This provided an increased amount of data. The results were very similar although took longer with the increased quantity of data. The

Pareto-optimal front took slightly longer to reach but once reached (500 generations with initial population of 100) it was left unchanged with successive generations. Interestingly when making comparisons between run-time, number of generations and initial population, a similar graph (Figure 9) is obtained to the previous one. This suggests that using a dynamic population means that one only has to use a small initial population in order to achieve the Pareto-optimal front rather than use a larger initial population size where convergence takes longer.

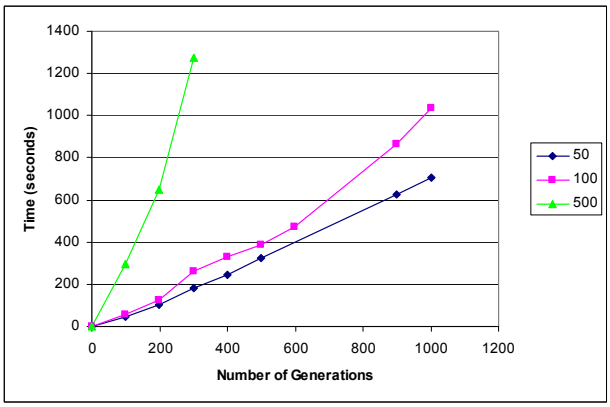

Figure 9. Graph to show how the computational time increases relative to the number of generations used with initial population sizes of 50, 100 and 500 and an increased quantity of data.

The final version of the paper will seek to establish the veracity of this conclusion with additional test cases and results.

5.3 Summary

This section has given examples of some of the preliminary results that have been obtained running this algorithm with different parameters.

6 Conclusions

This paper has shown that using a MOEA to allocate the optimal number and quantity of shares into a portfolio is worthy of consideration and could provide useful information for investors.

Preliminary results show a diverse set of solutions ranging from high return and low risk to the opposite. We are currently investigating a dynamic diversity and density operator but the work is not sufficiently developed to report on.

6.1 Future Work

Our main investigation is concerning how to preserve diversity. In terms of the result, future work will be carried out on testing the algorithm against other known algorithms and making comparisons using test data. Investigations into how to preserve diversity using density measures will also be looked at. Future work concerning portfolio optimisation will include looking at different ways of calculating volatility for VaR. Other risk measures will be examined and may be incorporated into the algorithm as a third objective function. Further investigations will also be made with other data sets and different time periods.

Acknowledgment

This work was supported in part by a bursary from the University of Greenwich, London, UK.

REFERENCES

[1] Amendment to the capital accord to incorporate market risks. Basle Committee on Banking Supervision, Jan 1996.
[2] Riskmetrics - technical document. JPMorgan/Reuters, Dec 1996.
[3] C. Alexander, *Market Models*, John Wiley and Sons LTD, Chichister, UK, 2001.
[4] R. Armananzas and J. A. Lozano, 'A multiobjective approach to the portfolio optimization problem', Technical report, ISG - Department of Computer Science and Artifical Intelligence, (2005).
[5] P. Artzner, F. Delbaen, J.-M. Ebner, and D. Heath, 'Coherent measures of risk', *Mathematical Finance*, **9**, 203–208, (1999).
[6] R. J. Bauer, *Genetic Algorithms and Investment Strategies*, John Wiley and Sons LTD, New York, USA, 1994.
[7] M. G. Castillo Tapia and C. A. Coello Coello, 'Applications of multi-objective evolutionary algorithms in economics and finance: A survey', in *IEEE Congress on Evolutionary Computation*, pp. 532–539, (Sept 2007).
[8] S. C. Chiam, K. C. Tan, and A. Al Mamum, 'Evolutionary multi-objective portfolio optimization in practical context', *International Journal of Automation and Computing*, **05**(1), 67–80, (January 2008).
[9] C. A. Coello Coello and G. B. Lamont, *Applications of Multi-Objective Evolutionary Algorithms*, World Scientific, Singapore, 2004.
[10] J. Danielsson, B. N. Jorgessen, C. G. Vries, and X. Yang, *Optimal Portfolio Allocation under a Probabalistic Risk Constraint and the Incentives for Financial Innovation*, Ph.D. dissertation, London School of Economics, June 2001.
[11] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*, John Wiley and Sons LTD, Chichister: UK, 2001.
[12] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, 'A fast and elitist multi-objective genetic algorithm: NSGA-II', Technical report, Indian Institute of Technology Kanpur, (2000).
[13] M. J. Fryer and J. V. Greenman, *Optimisation Theory - Applications in O.R. and Economics*, Edward Arnold, 1987.
[14] M. Gilli and E. Këllezi, *Computational Methods in Decision-Making, Economics and Finance*, chapter A Global Optimization Heuristic for Portfolio Choice with VaR and Expected Shortfall, 165–181, Kluwer, 2001.
[15] J. C. Hull, *Options, Futures and Derivatives*, Prentice Hall International, New Jersey, USA:, 6th edn., 2006.
[16] D. Maringer, *Portfolio Management with Heuristic Optimization*, Springer, 2005.
[17] H. M. Markowitz, 'Portfolio selection', *The Journal of Finance*, **1**(7), 77–91, (March 1952).
[18] J. D. Schaffer, *Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms*, Ph.D. dissertation, Vanderbilt University, Nashville, TN, USA, 1984.
[19] P. Skolpadungket, K. Dahal, and N. Harnpornchai, 'Portfolio optimization using multi-objective genetic algorithms', in *IEEE Congress of Evolutionary Computation*, (2007).
[20] E. Zitzler, M. Laumanns, and L. Thiele, 'SPEA2: Improving the strength pareto evolutionary algorithm', Technical report, Swiss Federal Institute of Technology (ETH) Zurich, (2001).

The Effects of Initial Solution Generators under Genetic Algorithm Framework for the Heterogeneous Probabilistic TSP

Yu-Hsin Liu[1]

Abstract. The probabilistic travelling salesman problem (PTSP) is a topic of theoretical and practical importance in the study of stochastic network problems. It provides researchers with a modelling framework for exploring the stochastic effects in routing problems. This paper focuses on investigating the performance of three initial solution generators under genetic algorithm framework for solving the heterogeneous PTSP. A set of numerical experiments based on heterogeneous PTSP instances were conducted to test the validity of three generators and their combinations. Though initial statistical testing by the permutation tests did not yield satisfactory results, it, however, directed the researcher to phenomenon of research potential. Specifically, the performance of the three generators seem to be instance-dependent for the same combination of *n* and *p*. Based on the findings, it is suggested that algorithms that embed the notion of adaptively selecting the initial solution generator based on some experiences in searching ability, or that can take into account of the patterns of individual instance could be promising strategies for solving the PTSP. Finally, to obtain more valid results, researchers are advised to include at least a certain amount of test instances with the same combination of *n* and *p* while conducting PTSP numerical experiments.

1 INTRODUCTION

The probabilistic travelling salesman problem (PTSP) is an extension of the well-known travelling salesman problem (TSP), which has been extensively studied in the field of combinatorial optimization. The goal of the TSP is to find the minimum length of a tour to all customers, given the distances between all pairs of customers whereas the objective of the PTSP is to minimize the expected length of the a priori tour where each customer requires a visit only with a given probability [1, 2, 3]. The main difference between the PTSP and the TSP is that in the PTSP the probability of each node being visited is between 0.0 and 1.0 while in TSP the probability of each node being visited is 1.0. Due to the fact that the element of uncertainty not only exists, but also significantly affects the system performance in many real-world transportation and logistics applications, the results from the PTSP can provide insights into research in other probabilistic combinatorial optimization problems. Moreover, the PTSP can also be used to model many real-world applications in logistical and transportation planning, such as daily pickup-delivery services with stochastic demand, job sequencing involving changeover cost, design of retrieval sequences in a warehouse or in a cargo terminal operations, meals on wheels in senior citizen services, trip-chaining activities, vehicle routing problem with stochastic demand, and home delivery service under e-commerce [4, 5, 6, 7, 8].

Early PTSP computational studies, dating from 1985, adopted heuristic approaches that were modified from the TSP (e.g., nearest neighbour, savings approach, spacefilling curve, radial sorting, 1-shift, and 2-opt exchanges) [1, 3, 9, 10, 11, 12]. With its less than satisfactory performance in yielding solution quality, researchers in the recent years switch to metaheuristic methods, such as ant colony optimization [13, 14], evolutionary algorithm [15], genetic algorithm [16], simulated annealing [17], threshold accepting [8] and scatter search [18, 19, 20]. Since the genetic algorithm (GA), a conceptual framework of the population-based metaheuristic method, has been shown to yield promising outcomes for solving the PTSP [16] and various complicated optimization problems in the past three decades [21, 22, 23, 24, 25], this study will further investigate the effects of different methods based on random generation and nearest neighbour algorithms for generating initial solutions under GA framework for solving the PTSP. To validate the effectiveness and efficiency of the proposed algorithmic procedure, a set of heterogeneous (90 instances) PTSP test instances as used in the previous studies [8, 16, 18, 19, 20] will be used as the base for comparison purpose.

The remainder of this paper is organized as follows. In the next section, expressions for exactly and approximately evaluating the a priori tour for the PTSP are introduced. The details of the proposed algorithmic procedure for the PTSP are then described. The results of the numerical experiments are presented and discussed in the next section, followed by concluding comments.

2 DEFINITION AND EVALUATION OF THE PTSP

The PTSP is defined on a directed graph $G := (V, E)$, where $V := \{0, v_1, v_2, ..., v_n\}$ is the set of nodes or vertices, $E \subseteq V \times V$ is the set of directed edges. Node *0* represents the depot with the presence probability of 1.0. Each non-depot node v_i is associated with a presence probability p_i that represents the possibility that node v_i will be present in a given realization. Given a directed graph *G*, the PTSP is to find an *a priori* Hamiltonian tour with minimal expected length in *G*.

Solving the PTSP mainly relies on computing the expected length of an *a priori* tour. The computation of the expected length of a specific *a priori* PTSP tour τ, denoted as $E[\tau]$,

[1] Dept. of Civil Engineering, National Chi Nan Univeristy, Puli, Nantou Hsien 54561, Taiwan. Email: **yuhsin@ncnu.edu.tw**

depends on the relative location of nodes on that tour and the presence probability of each node in a given instance. Jaillet and Odoni [26] proposed an approach to exactly calculate $E[\tau]$ in the complexity of $O(n^3)$ for the PTSP.

$$E[\tau] = \sum_{i=0}^{n} \sum_{j=i+1}^{n+1} \{d_{\tau(i)\tau(j)} p_{\tau(i)} p_{\tau(j)} \prod_{k=i+1}^{j-1} (1 - p_{\tau(k)})\} \quad (1)$$

d_{ij} represents the distance between nodes i and j; $\tau(i)$ denotes the node that has been assigned the i^{th} stop in tour τ and $p_{\tau(i)}$ is the probability of node $\tau(i)$. $\tau(0)$ and $\tau(n+1)$ represent node 0, which is the depot.

Even though (1) yields a polynomial evaluation time for the PTSP, the resulting $O(n^3)$ time for calculating $E[\tau]$ is still very long, especially for metaheuristic methods which need to repeatedly evaluate the objective function value $E[\tau]$. In this study, the proposed algorithm needs to repeatedly compare two solutions (i.e., the new solution before and after local search procedure, which is described in the next section) based on their values of $E[\tau]$. Therefore, the depth approximate evaluation [14, 18, 20] was used to increase the computation efficiency of the proposed algorithm.

$$E_{\lambda}^{AP}[\tau] = \sum_{i=1}^{n} \sum_{j=i+1}^{\min\{n+1, i+\lambda\}} \{d_{\tau(i)\tau(j)} p_{\tau(i)} p_{\tau(j)} \prod_{k=i+1}^{j-1} (1 - p_{\tau(k)})\} \quad (2)$$

The only difference between (1) and (2) is the choice of truncation position λ in (2). Equation (2) will have the computational complexity of $O(n\lambda^2)$, instead of $O(n^3)$ in (1). It is easy to see that (2) becomes more accurate when λ increases. A larger value of λ, however, requires more computation efforts for the computation of (2). Equation (2) can perform a very good approximation of $E[\tau]$ with a smaller value of λ when the value of $p_{\tau(k)}$ gets larger, because $\prod_{k=i+1}^{j-1} (1 - p_{\tau(k)})$ will yield a very small value and can be omitted. Nevertheless, Equation (2) will need a larger value of λ to perform a good approximation when the value of of $p_{\tau(k)}$ is small. The detailed procedure of using approximate evaluation shown in (2) to accelerate the algorithm is described in the next section.

3 SOLUTION ALGORITHMS

The proposed genetic algorithm consists of four components as shown in figure 1. They are the initialization (including three initial solution generators – random generator (RAN), nearest neighbour algorithm – type 1 (NN1), nearest neighbour algorithm – type 2 (NN2) and their combinations, local search with stochastically selecting from two different methods (i.e., 1-shift and 2-opt exchanges), selection, and edge recombination (ER) crossover. When starting to solve the PTSP (Generation 0, g = 0), initial solutions are generated based on three different nearest neighbour algorithms, which are then improved by the local search. Then, "selection" is called into place to further select solutions to be mated based on their solution quality (objective function value). Pairs of solutions are used to generate the new solutions via ER crossover. The newly generated solutions are then improved using the local search. The solutions are allowed to evolve through successive generations until a termination criterion is met. The detailed description of the embedded components is illustrated in the following sections.

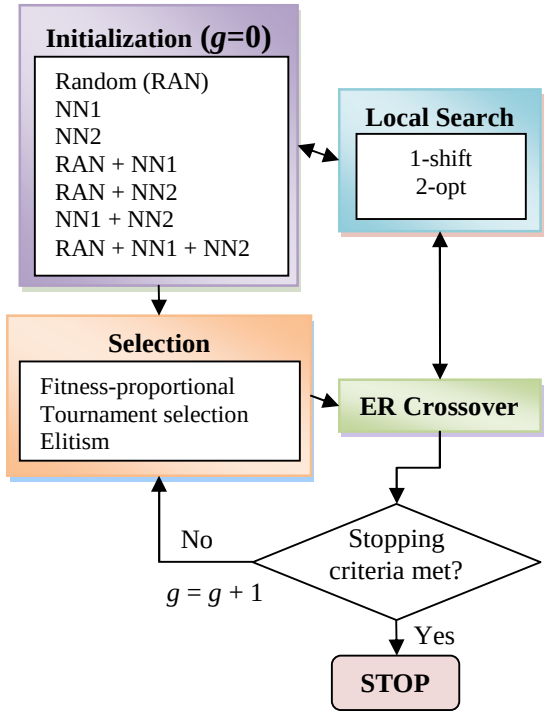

Figure 1. The solution procedure for the PTSP

3.1 Initialization

This procedure is designed to generate m initial solutions (m = 10 in this study). Three initial solution generation methods are proposed. The detailed procedures of these three methods are described as follows.

3.1.1 Random generator (RAN)

The random generator is used to randomly create a set of initial solutions. Considering a PTSP with n nodes (excluding the depot, node *0*), the node, a_{r1}, is randomly chosen from the set of unselected node (S_u) and inserted into the second location. Then, the set of unselected node (S_u) should be updated by erasing node a_{r1}. The third location can be filled by randomly choosing a node from S_u. Following the above procedure, the initial solution (tour) can be built.

3.1.2 Nearest Neighbour Algorithm – Type 1 (NN1)

Considering a PTSP with n nodes (excluding the depot, node *0*), the two nearest nodes, a_0 and b_0, from node *0* is selected first and randomly inserted into the second and the last locations, respectively. Basically, the nearest and second nearest nodes from the current node are selected by the probability of 0.9 and

0.1, respectively. This mechanism is used to generate different initial solutions. The procedure of building the initial solution is as follows. After selecting nodes a_0 and b_0, one of the top two nearest nodes from a_0 is randomly selected (a_1) based on the specific probability (i.e., 0.9 for the nearest node and 0.1 for the second nearest node) and inserted behind a_0. Similarly, the node (b_1) is selected and inserted in front of b_0. Then, among the remaining nodes, one of the top two nearest nodes from a_1 is randomly selected (a_2) based on the specific probability and inserted behind a_1, while node (b_2) is randomly selected based on the rule described previously and inserted in front of b_1. The initial solution (tour) is thus built by following the above rule and expressed as follows.

$0 \quad a_0 \quad a_1 \quad a_2 \quad \ldots \quad b_2 \quad b_1 \quad b_0 \quad 0$

3.1.3 Nearest Neighbour Algorithm – Type 2 (NN2)

Considering a PTSP with n nodes (excluding the depot, node 0), the farthest node, a_0, from node 0 is selected first and randomly inserted into a location between ($\lfloor (n+1)/2 \rfloor - 4$) and ($\lfloor (n+1)/2 \rfloor + 4$). The reason why $\lfloor (n+1)/2 \rfloor - 4$ and $\lfloor (n+1)/2 \rfloor + 4$ are used rather than the middle point of the tour ($\lfloor (n+1)/2 \rfloor$) is to generate a set of different initial solutions. The nearest neighbour algorithm, which find the unvisited node with the shortest distance, is used to build up the sequence of the tour. Basically, the nearest and second nearest nodes from the current node are selected by the probability of 0.9 and 0.1, respectively. This mechanism is used to generate different initial solutions. The procedure of building the initial solution is as follows. After selecting node a_0, one of the top two nearest nodes from a_0 is randomly selected (a_1) based on the specific probability and inserted in front of a_0. Similarly, the node (a_2) is selected and inserted behind a_0. Then, among the remaining nodes, one of the top two nearest nodes from a_1 is randomly selected (a_3) based on the specific probability and inserted in front of a_1, while node (a_4) is randomly selected based on the rule described previously and inserted behind a_2. The initial solution (tour) is thus built by following the above rule and expressed as follows.

$0 \quad \ldots \quad a_3 \quad a_1 \quad a_0 \quad a_2 \quad a_4 \quad \ldots \quad 0$

3.2 Local search

This component is used in an attempt to further enhance the solution generated via a local search procedure. As the previous studies [16, 20] showed that the diversified local search by randomly choosing two different local search methods (i.e., 1-shift and 2-opt) to improve the solution generated yielded the best results in both heterogeneous and homogeneous cases, the diversified local search strategy is used in this procedure. That is, one of these two local search methods is selected to improve the candidate solution based on the specific probability of 0.5 for these two local searches.

The procedures of 1-shift and 2-opt exchanges are briefly summarized as follows. Given an *a priori* tour τ, its 1-shift neighbourhood is the set of tours obtained by moving a node at position i to position j with the intervening nodes being accordingly shifted backwards one space. The 2-opt exchange is the set of tours obtained by reversing a section of τ.

The approximate evaluation of expected length of the *a priori* tour shown in (2) is then used to increase the computational efficiency. For a specific tour τ, $E_\lambda^{AP}[\tau]$ is always less than the value of $E[\tau]$ because of the truncation in calculating $E_\lambda^{AP}[\tau]$. Let τ_b and τ_a denote the *a priori* tour before and after a specific local search method, respectively. It means that no improvement has been found after the local search if $E_\lambda^{AP}[\tau_a] \geqq E[\tau_b]$. Equation (1) is used to exactly evaluate the solution after the local search if $E_\lambda^{AP}[\tau_a] < E[\tau_b]$. If the local search yields a better $E[\tau]$ value than the one from the original solution (i.e., $E[\tau_a] < E[\tau_b]$), the new solution (τ_a) will replace the original solution (τ_b). If no improvement has been found after the local search, no replacement will be made. The above procedure is repeated N_{LS} times for each solution ($N_{LS} = 30$ in this study).

3.3 Selection

Three selection methods are used in this study: fitness-proportionate, elitism and tournament selection. Under the fitness-proportionate selection method, the probability of selecting a particular solution for reproduction is proportional to its own fitness (i.e., E[τ]) relative to the average fitness of the entire current generation. With this selection method, the best solution tends to produce the largest amount of offspring and hence survive to future generations. This procedure can be regarded as a "biased" roulette wheel where each string in the current population occupies a roulette wheel slot sized in proportion to its fitness [23]. Selection can be done by simply spinning the weighted roulette wheel, and fitter strings will have higher chances of being selected.

Tournament selection, inspired by the competition in nature among individuals for the right to mate, picks two solutions using the fitness-proportionate selection from the population and the fittest one is selected for reproduction [22, 23]. Each solution can participate in an unlimited number of tournaments. The two winning solutions in the tournament are then subjected to the crossover operators, as described in the next section.

Under the elitism selection strategy, the top m_1 strings (m_1 is determined by the analyst) of the current generation in terms of fitness value are kept and propagated to the next generation [22]. The remaining solutions in the next generation are then generated based on the tournament selection method and the crossover operators. This procedure guarantees that the best solution in the next generation is not worse than the one in the current generation.

3.4 Edge recombination (ER) crossover

The main purpose of this component is to create new solutions using a given pair of solutions chosen by "selection." Based on the results from previous studies [15, 27], the edge recombination (ER) crossover from genetic algorithms performed best when compared to other crossover strategies for

both in TSP and PTSP. Therefore, ER crossover was adopted in this study.

ER crossover was proposed by Whitley et al. [28] to solve the traditional TSP. A 5-node PTSP is used as an example to describe the procedure of ER crossover. Assuming that two solutions (tours) are chosen from the "selection"--(0, 4, 3, 1, 2, 0) and (0, 1, 2, 3, 4, 0), the edges connected to each node are as follows. For node 0, the first solution indicates that node 0 connects to nodes 2 and 4 and the second solution shows that node 0 connects to nodes 1 and 4. Therefore, node 0 connects to nodes 1, 2, and 4 by considering these two solutions. Similarly, node 1 connects to nodes 0, 2, 3; node 2 connects to nodes 0, 1, 3; node 3 connects to nodes 1, 2, 4; node 4 connects to nodes 0, 3. These are the initial edge lists for each node.

The operation of the ER crossover is described as follows. Assuming that node 0 is selected as the starting node for the new solution, all edges incident to node 0 must be deleted from the initial edge list. As described, from node 0 we can go to nodes 1, 2, or 4, while nodes 1 and 2 have two active edges and node 4 has only one active edge by deleting node 0 from the initial edge list. The node with the fewest active edge, node 4, is picked as the node next to node 0 in the new solution. Then, the edge list for the remaining nodes (nodes 1, 2, and 3) is further updated by deleting node 4. The updated edge list is node 1 (2, 3), node 2 (1, 3), and node 3 (1, 2). From node 4, we can only go to node 3 (as node 0 is already deleted from the list). Therefore, node 3 is chosen to be the node next to node 4 in the new solution. The new solution generated is further improved by the local search.

3.5 The procedure after the first generation

The newly generated solutions from the ER crossover and local search are used to update the population in terms of the objective function value. The above procedure is repeated until a termination criterion is met. However, if there are no solutions to be updated for the population in the current generation, the initialization is used to generate (m - m_1) new solutions in the next generation, but keeping m_1 high quality solutions (m_1 = 2, in this study). In addition, if the previous three generations converge to the same best solution, the local search is used to improve that "converged" solution by repeating N_{LS2} times to exhaustively search the neighbourhood of that "converged" solution (N_{LS2} = 300, in this study).

4 NUMERICAL RESULTS AND DISCUSSIONS

Ninety instances generated by Tang and Miller-Hooks [4] with size n = 50, 75, and 100 were used as numerical experiments in this study to examine the effects of three different initial solution generators under genetic algorithm framework for solving the heterogeneous PTSP. Three groups of problem sets categorized by different intervals of customer presence probabilities were created for each problem size (n = 50, 75, and 100). Presence probabilities of customer nodes were randomly generated from a uniform distribution on intervals (0.0, 0.2], (0.0, 0.5], (0.0, 1.0], one for each problem size. The presence probability of the depot (node *0*) was assigned as 1.0. Ten different problem instances were randomly generated for each presence probability of customer nodes. For each instance, the coordinates of one depot and n customer nodes (x_i, y_i) were generated based on a uniform distribution from $[0,100]^2$. The Euclidean distance for each pair of nodes was calculated by using $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

All implementations in this study were coded in FORTRAN and run on an Intel Pentium IV 2.8 GHz CPU personal computer with 512 MB memory, running Windows XP operating system, using Compaq Visual Fortran 6.5 compiler. To compare the effectiveness among three initial solution generators, the preset maximum number of generations (g_{max}) was used as the termination criterion (g_{max} is set to be two times the number of nodes, i.e., g_{max} = 2n, in this study). In order to enhance the robustness of the results and conduct appropriate statistical analysis, the proposed methods were used to solve each problem instance 30 times. The obtained $E[\tau]$ values of ten instances were averaged (as shown in shade in Tables 1, 2, and 3 for 50-, 75-, and 100-node, respectively) and tested for the comparative solution quality of three generators by the permutation test [29]. Definitions of terms used in the column headings for Tables 1, 2, and 3 are given as follows. In "PTSP instances", to take "50-0.2-01" for example, it represents the instance with problem size 50, customer presence probability interval (0.0, 0.2), and instance number 01, while "50-0.2" represents the average of all 10 instances with n = 50 and p = 0.2.

Table 1. Detailed results for problem size n = 50

	RAN	NN1	NN2	RAN+ NN1	RAN+ NN2	NN1+ NN2	RAN+ NN1+ NN2
p = 0.2							
01	**217.83**	**217.83**	**217.83**	217.83	217.83	217.83	217.83
02	205.96	205.96	**205.96**	205.96	205.96	205.96	205.96
03	212.36	**212.29**	**212.29**	212.29	212.29	212.29	212.29
04	215.70	**215.70**	**215.70**	215.70	215.70	215.70	215.70
05	249.59	**249.59**	**249.59**	249.59	249.59	249.59	249.59
06	**218.80**	**218.80**	**218.80**	218.80	218.80	218.80	218.80
07	**245.80**	**245.80**	**245.80**	245.80	245.80	245.80	245.80
08	**239.16**	**239.16**	**239.16**	239.16	239.16	239.16	239.16
09	**237.74**	237.74	237.74	237.74	237.74	237.74	237.74
10	205.45	205.45	**205.45**	205.45	205.44	205.44	205.45
Avg.	224.84	224.83	224.83	224.83	224.83	224.83	224.83
p = 0.5							
01	313.61	313.73	**313.40**	313.61	313.44	313.52	313.48
02	349.84	349.88	**349.81**	349.90	350.05	349.91	349.91
03	335.28	**334.98**	335.87	335.43	334.98	335.28	334.69
04	348.24	348.16	**348.14**	348.27	348.20	348.18	348.21
05	**327.06**	**327.06**	**327.06**	327.06	327.06	327.06	327.06
06	**373.58**	373.98	375.98	374.48	374.95	376.16	374.82
07	324.33	324.34	**324.32**	324.34	324.33	324.34	324.32
08	**337.58**	340.36	339.43	339.66	338.98	339.53	339.43
09	352.72	**352.59**	354.03	352.75	353.49	353.57	352.89
10	348.72	348.79	**348.70**	348.67	348.75	348.74	348.83
Avg.	313.61	313.73	**313.40**	313.61	313.44	313.52	313.48
p = 1.0							
01	462.57	**459.14**	466.64	461.87	463.61	463.64	465.19
02	405.35	406.11	**405.04**	405.22	405.17	406.03	407.66
03	468.83	468.68	**466.13**	469.95	467.91	467.19	467.85
04	459.96	**457.54**	460.87	458.99	460.87	459.94	459.69
05	425.03	**424.55**	428.34	424.34	425.78	426.25	425.88
06	506.82	**502.63**	503.56	505.04	506.50	505.18	506.47
07	**482.68**	485.09	482.75	484.91	485.61	483.35	485.23
08	439.99	**439.57**	441.53	439.33	440.09	440.88	439.93
09	439.55	**437.32**	440.13	437.38	439.34	438.67	439.01
10	**417.59**	417.83	417.60	417.70	417.49	417.67	417.70
Avg.	450.84	449.85	451.26	450.47	451.24	450.88	451.46

Contrary to the researcher's expectation, none of the permutation test done on the average $E[\tau]$ values obtained by the three generators are statistically significantly different, $p > 0.05$. As studies have shown the effectiveness of incorporating the concept of diversification into metaheuristic methods [20, 30]

four more methods were generated by combining any two or three of the proposed generators (i.e., RAN+NN1, RAN+NN2, NN1+NN2, RAN+NN1+NN2). Based on the permutation test done of the seven methods (i.e., RAN, NN1, NN2, RAN+NN1, RAN+NN2, NN1+NN2, RAN+NN1+NN2), again, no significant results were found.

Table 2. Detailed results for problem size n = 75

	RAN	NN1	NN2	RAN+ NN1	RAN+ NN2	NN1+ NN2	RAN+ NN1+ NN2
p = 0.2							
01	267.24	267.35	**267.22**	267.37	267.32	267.30	267.27
02	278.13	**278.13**	278.13	278.13	278.13	278.13	278.13
03	**268.25**	**268.25**	**268.25**	268.25	268.25	268.25	268.25
04	271.55	**271.52**	271.53	271.52	271.52	271.52	271.51
05	**264.67**	264.81	264.82	264.63	264.76	264.75	264.65
06	236.27	**236.05**	236.44	236.19	235.96	236.11	236.20
07	**273.18**	273.19	273.19	273.18	273.18	273.21	273.19
08	**251.45**	251.45	251.45	251.45	251.45	251.45	251.45
09	280.46	280.45	**280.42**	280.41	280.50	280.44	280.47
10	**268.31**	268.31	268.31	268.31	268.31	268.31	268.31
Avg.	265.95	265.95	265.98	265.94	265.94	265.95	265.94
p = 0.5							
01	**398.78**	402.60	401.82	401.92	401.95	403.29	402.60
02	**424.03**	431.67	430.47	428.09	428.59	427.78	428.27
03	**427.51**	427.76	428.23	427.33	427.83	428.03	428.06
04	410.10	410.28	**405.69**	410.80	407.13	406.62	408.29
05	400.87	403.56	**399.84**	401.97	401.89	401.05	400.25
06	**378.40**	379.60	379.13	379.09	378.70	379.00	379.36
07	428.46	431.76	**425.49**	428.26	426.52	428.48	427.00
08	397.67	**395.72**	399.70	395.82	396.47	397.00	396.14
09	**367.88**	367.91	367.95	367.94	367.88	367.93	367.81
10	**401.75**	405.60	402.88	406.27	402.80	406.78	404.18
Avg.	403.54	405.65	404.12	404.75	403.98	404.60	404.20
p = 1.0							
01	521.93	525.03	**516.10**	521.94	517.51	521.04	523.19
02	547.61	**532.37**	543.87	539.70	544.46	540.22	539.93
03	489.01	502.45	**486.27**	498.76	490.17	491.54	493.80
04	579.02	**574.62**	579.25	574.03	577.60	580.78	576.15
05	531.78	**530.34**	532.84	532.76	531.53	529.19	531.93
06	492.21	**489.56**	492.75	490.52	491.52	490.50	489.36
07	**517.22**	524.34	524.06	521.29	521.11	519.85	524.77
08	560.81	**551.07**	554.83	546.44	550.88	550.35	550.66
09	535.26	534.31	**534.29**	536.19	537.27	534.82	535.43
10	**514.01**	517.00	516.86	511.28	513.38	512.66	517.66
Avg.	528.89	528.11	528.11	527.29	527.54	527.10	528.29

These unexpected outcomes alerted the author to look at the results more deeply. Specifically, rather than looking at the averaged $E[\tau]$ values of all ten instances, each of the ten instances was analyzed separately to acknowledge its maybe distinct characteristics. Moreover, as the combination of different generators may obscure the effects of the three generators, *t*-tests are conducted only on the three generators for each problem instance, respectively. In Tables 1, 2, and 3, the underlined value denotes the average value yield by the specific initial solution generator is significantly different from the best average yielded value among these three generators.

As shown in Tables 1, 2, and 3, even though none of these three initial solution generators (i.e., RAN, NN1, and NN2) consistently performed best in terms of the obtained E[τ] value across all instances, it seems to be somewhat instance-dependent for the same combination of n and p. For example, for n = 75 and $p \in (0.0, 0.5]$, the value obtained by RAN (398.7818) are significantly better than the values obtained by NN1 (402.6016) and NN2 (401.8246) for instance number 01; the value obtained by NN2 (405.6888) are significantly better than the values obtained by RAN (410.0959) and NN1 (410.2795) for instance number 04; the value obtained by NN1 (395.7191) are significantly better than the values obtained by RAN (397.6664) and NN2 (399.6999) for instance number 08. Similar outcomes are found in other combinations of n and p. The fact that some instances under the same combination of n and p have smaller E[τ] values obtained by the specific generator where others have higher E[τ] values yielded by the same generators indicates that the possibility of finding the optimal solution based on the same generator may fluctuate across instances. The phenomenon among instances suggests that the effectiveness of finding the optimal solution might be mediated by some factors, such as the characteristics of the instance (e.g., the relative location and presence probability of each node) in solving the PTSP. As such, algorithms which can dynamically select the initial solution generator according to individual instance pattern might be one direction for effectively solving the PTSP. Another promising strategy would be algorithms that could adaptively evolve by learning from some search experience so as to select the best generator, rather than fixing on one solution generator at the commencement of the test, as done in most of existing PTSP studies.

Table 3. Detailed results for problem size n = 100

	RAN	NN1	NN2	RAN+ NN1	RAN+ NN2	NN1+ NN2	RAN+ NN1+ NN2
p = 0.2							
01	277.07	**277.07**	277.07	277.07	277.07	277.07	277.07
02	298.59	**298.59**	298.59	298.59	298.59	298.59	298.59
03	**309.57**	309.57	309.58	309.57	309.58	309.57	309.57
04	298.76	**298.76**	298.76	298.76	298.76	298.76	298.76
05	319.73	319.82	**319.70**	319.78	319.69	319.74	319.68
06	301.13	301.14	**301.12**	301.14	301.13	301.13	301.13
07	301.50	**301.49**	301.58	301.47	301.54	301.48	301.58
08	294.61	**293.88**	293.98	293.92	293.62	293.87	294.34
09	304.43	**304.36**	304.47	304.39	304.42	304.38	304.39
10	303.83	303.82	**303.82**	303.82	303.82	303.82	303.82
Avg.	300.92	300.85	300.87	300.85	300.82	300.84	300.89
p = 0.5							
01	**463.73**	465.00	464.96	463.86	464.03	468.90	464.74
02	464.89	**460.58**	468.93	462.08	465.52	460.11	464.67
03	**463.59**	469.47	464.37	470.16	464.33	469.35	467.78
04	445.44	**444.40**	451.64	444.66	447.70	445.12	448.98
05	**473.26**	475.68	474.57	473.02	473.45	480.85	476.05
06	**486.30**	486.39	489.04	487.77	488.30	488.03	488.92
07	**459.90**	465.83	462.18	465.18	460.32	470.36	464.58
08	454.77	454.64	**451.45**	454.68	453.88	453.61	452.96
09	**461.00**	462.88	465.46	463.33	462.68	459.34	462.96
10	**437.79**	440.19	441.30	439.36	438.94	441.06	440.28
Avg.	461.07	462.51	463.39	462.41	461.92	463.67	463.19
p = 1.0							
01	619.64	**612.96**	619.43	616.14	621.66	619.02	616.53
02	638.51	637.55	**632.21**	635.54	635.15	642.47	634.86
03	633.07	**625.04**	634.56	630.48	633.94	626.35	629.48
04	**639.12**	640.96	639.25	636.12	636.49	644.43	638.07
05	**586.23**	590.03	588.71	591.46	593.25	596.98	588.47
06	**661.64**	662.68	662.15	662.97	660.17	663.31	662.65
07	581.35	**578.14**	583.21	574.94	580.72	588.54	579.69
08	645.27	637.55	**632.14**	636.66	637.20	636.00	636.46
09	641.42	**636.91**	644.47	634.08	634.68	641.90	640.12
10	614.06	**613.86**	618.24	616.75	620.19	618.89	618.72
Avg.	626.03	623.57	625.44	623.51	625.35	627.79	624.51

Another related suggestion from this study is with regard to the inadequacy of past studies that relied only on a single-instance numerical experiment. Explicitly, one PTSP test instance for a specific combination of n and p, as used in previous studies [14], assumed that the same combination of n and p will yield similar results by the same method. As could be

clearly seen from the results of this study, the assumption is not upheld. The numerical experiment based on one PTSP test instance may be subject to biased sampling of the PTSP instances, which may inadvertently result in over- or under-estimation of the performance of the proposed algorithms for PTSP. As such, to obtain more valid results, researchers are advised to include at least a certain amount of test instances with the same combination of n and p while conducting PTSP numerical experiments.

5 CONCLUSIONS & FUTURE WORK

In this paper, three different initial solution generators under GA framework are first proposed and developed to solve the PTSP. Though the numerical results did not support the predominate superiority of one specific generator, nor did incorporating the notion of diversification enhances any of its performance, it, however, enlightened the researcher to be aware of the possibility of context-dependent phenomenon. Particularly, further data analysis and testing showed that the $E[\tau]$ values obtained by the three generators seem to be instance-dependent for the same combination of n and p. Based on the findings, it is suggested that algorithms that embed the notion of adaptively selecting the initial solution generator based on some experiences in searching ability, or that can take into account of patterns of individual instance could be promising strategies to solve the PTSP. Finally, to obtain more valid results, researchers are advised to include at least a certain amount of test instances with the same combination of n and p while conducting PTSP numerical experiments.

ACKNOWLEDGEMENTS

This work was supported primarily by the National Science Council of Taiwan under Grant and NSC 97-2410-H-260-012.

REFERENCES

[1] D. Bertsimas, *Probabilistic Combinatorial Optimization Problems*, Ph.D. Dissertation, Massachusetts Institute of Technology, MA, U.S.A. (1988).

[2] D. Bertsimas, P. Jaillet, and A. R. Odoni. A Priori Optimization. *Operations Research*, 38:1019-1033, (1990).

[3] P. Jaillet, Probabilistic Traveling Salesman Problems, Ph.D. dissertation, Massachusetts Institute of Technology, MA, U.S.A. (1985).

[4] J. J. Bartholdi, L. K. Platzman, R. L. Collins, and W. H. Warden. A Minimal Technology Routing System for Meals on Wheels. *Interfaces*, 13:1-8, (1983).

[5] D. Bertsimas, P. Chervi, and M. Peterson. Computational Approaches to Stochastic Vehicle Routing Problems. *Transportation Science*, 29:342-352, (1995).

[6] A. M. Campbell. Aggregation for the Probabilistic Traveling Salesman Problem. *Computers & Operations Research*, 33:2703-2724, (2006).

[7] P. Jaillet. A priori Solution of a Traveling Salesman Problem in which a Random Subset of the Customers are Visited. *Operations Research*, 36:929-936, (1988).

[8] H. Tang and E. Miller-Hooks. Approximate Procedures for the Probabilistic Traveling Salesperson Problem. *Transportation Research Record*, 1882:27-36, (2004).

[9] J. J. Bartholdi and L. K. Platzman. Heuristics Based on Spacefilling Curves for Combinatorial Problems in Euclidean Space. *Management Science*, 34:291-305, (1988).

[10] D. Bertsimas and L. Howell. Further Results on the Probabilistic Traveling Salesman Problem. *European Journal of Operational Research*, 65:68-95, (1993).

[11] P. Jaillet. Stochastic Routing Problems. In *Advanced School on Stochastics in Combinatorial Optimization*. G. Andreatta, F. Mason, P. Serafini (Eds.). World Scientific Publisher, pp. 192-213 (1987).

[12] F. Rossi and I. Gavioli. Aspects of Heuristic Method in the Probabilistic Traveling Salesman Problem. In *Advanced School on Stochastics in Combinatorial Optimization*. G. Andreatta, F. Mason, P. Serafini (Eds.). World Scientific Publisher, pp. 214-227 (1987).

[13] L. Bianchi, Ant Colony Optimization and Local Search for the Probabilistic Traveling Salesman Problem: A Case Study in Stochastic Combinatorial Optimization, Ph.D. Dissertation, Universite Libre de Bruxelles, Brussles, Belgium. (2006).

[14] J. Branke and M. Guntsch. Solving the Probabilistic TSP with Ant Colony Optimization. *Journal of Mathematical Modelling and Algorithms*, 3:403-425, (2004).

[15] Y.-H. Liu, R.-C. Jou, C.-C. Wang, and C.-S. Chiu. An Evolutionary Algorithm with Diversified Crossover Operator for the Heterogeneous Probabilistic TSP. Lecture Notes in Artificial Intelligence, 4617:351-360, (2007).

[16] Y.-H. Liu. A Memetic Algorithm for the Probabilistic Traveling Salesman Problem. In: *Procs. 2008 IEEE Congress on Evolutionary Computation (CEC 2008)*, Hong Kong. (2008).

[17] N. E. Bowler, T. M. A. Fink, and R. C. Ball. Characterization of the Probabilistic Traveling Salesman Problem. *Physical Review E*, 68:036703, (2003).

[18] Y.-H. Liu. A Scatter Search Based Approach with Approximation Evaluation for the Heterogeneous Probabilistic Traveling Salesman Problem. In *Procs. 2006 IEEE Congress on Evolutionary Computation (CEC 2006)*, Vancouver, Canada. (2006).

[19] Y.-H. Liu. A Hybrid Scatter Search for the Probabilistic Traveling Salesman Problem. *Computers & Operations Research*, 34:2949-2963, (2007).

[20] Y.-H. Liu. Diversified Local Search Strategy under Scatter Search Framework for the Probabilistic Traveling Salesman Problem. *European Journal of Operational Research*, 191:332-346, (2008).

[21] T. Bäck, D. B. Fogel and Z. Michalewicz, *Handbook of Evolutionary Computation*, Oxford University Press, U.K., (1997).

[22] L. Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, U.S.A., (1991).

[23] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*, Addison Wesley, Reading, PA, U.S.A., (1989).

[24] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, Boston, MA, U.S.A., (1992).

[25] Y.-H. Liu and H. S. Mahmassani. Global Maximum Likelihood Estimation Procedure for Multinomial Probit Model Parameters. *Transportation Research B*, 34B:419-449, (2000).

[26] P. Jaillet and A. R. Odoni. The Probabilistic Vehicle Routing Problem. In *Vehicle Routing: Methods and Studies*. B. L. Golden, A. A. Assad (Eds.). North-Holland, pp. 293-318, (1988).

[27] J.-Y. Potvin. Genetic Algorithms for the Traveling Salesman Problem. *Annals of Operations Research*, 63:339-370, (1996).

[28] D. Whitley, T. Starkweather, and D. Fuquay. Scheduling Problems and Traveling Salesmen: The Genetic Edge Recombination Operator. In: *Procs. 3rd Int'l Conf. on Genetic Algorithm* (ICGA '89), Fairfax, Virginia, USA, Morgan Kaufmann. (1989).

[29] D. Basso, M. Chiarandini, and L. Salmaso. Synchronized Permutation Tests in Replicated I × J Designs. *Journal of Statistical Planning and Inference*, 137:2564-2578, (2007).

[30] B. Gendron, J. Y. Potvin, and P. Soriano. Diversification Strategies in Local Search for a Nonbifurcated Network Loading Problem. *European Journal of Operational Research*, 142:231-241, (2002).