Dynamic Search Initialisation Strategies for Multi-Objective Optimisation in Peer-to-Peer Networks

Ian Scriven, Andrew Lewis and Sanaz Mostaghim

Abstract— **Peer-to-peer based distributed computing environments can be expected to be dynamic to greater of lesser degree. While node losses will not usually lead to catastrophic failure of a population-based optimisation algorithm, such as particle swarm optimisation, performance will be degraded unless the lost computational power is replaced. When resources are replaced, one must consider how to utilise newly available nodes as well as the loss of existing nodes. In order to take advantage of newly available nodes, new particles must be generated to populate them. This paper proposes two methods of generating new particles during algorithm execution and compares the performance of each approach, then investigates a hybridised approach incorporating both mechanisms.**

I. Introduction

Parallel and distributed computing has drawn a lot of attention in the last decades due to the rapidly advancing technology and the fact that the scale of computing resources is increasing. Large clusters of computers, grid technology, peer-to-peer (P2P) networks, and multi-core systems are some examples of such parallel environments. Such parallel systems can significantly reduce the computation time for highly complex modeling, simulation, and optimisation problems from science and industry. While dedicated parallel systems and clusters can deliver significant computing capacity, they are generally expensive to acquire and maintain, and have a fixed maximum capacity or scale. Attention has thus been turned increasingly to grid computing and P2P environments. In addition to the promise of affordable, large scale computing, these environments bring with them challenges inherent in their nature: they are often heterogeneous and highly dynamic. It is a fundamental design consideration that apparent failure, of computing resources or communications, is to be expected in distributed systems. Consideration of the fault tolerance of algorithms used becomes a critical issue.

In this paper, we study optimisation methods that can be run in parallel on P2P systems, with the aim of unlocking the potential of the large number of non-dedicated PCs currently found in many organisations. Parallel optimisation using traditional, dedicated high performance computing resources is not a new field [1], [2], but the existing models of parallelisation have to be adapted and investigated more when we run them on some new technologies such as P2P networks. P2P networks are typically used for connecting nodes via largely *ad hoc* connections. Such networks are useful for many purposes. Sharing content files containing audio, video, and real time data such as telephony traffic, are some examples for using P2P technology. The conventional Parallel Optimisation methods are categorised in three different paradigms such as (1) Master-Slave, (2) Island and (3) Diffusion model. However, in order to study optimisation on a P2P system, we have to design a new model, such as those studied in the work of Laredo *et al.* [3], [4].

We have considered, in particular, multi-objective optimisation algorithms, the solution of which is usually a set of solutions represented as an optimal front i.e., none of these solutions can be improved in one objective without getting worse with respect to some other objective. In other words, the optimal solutions are related to each other through a Pareto-domination relation. For solving such problems, population-based methods such as Evolutionary Algorithms (EA) and Particle Swarm Optimisation (PSO) have been shown to be good candidates, as they are inherently parallel by design [5]. Some difficulties arise when we solve multi-objective problems, as a set of solutions rather than one optimal solution have to be found [2], [6], [7], [8], [9], [10], [11].

PSO algorithms are highly suitable for use in distributed P2P environments, as the fixed population size negates the population size control issue encountered in decentralised EAs [12]. P2P PSO algorithms also require less information about the entire population when generating possible new solutions. Whereas EAs require up-do-date knowledge of a significant portion of the population in order to generate new population members and remove old ones, PSO algorithms only require a reasonable idea of where the optimal points lie in the solution space in order to guide the existing particles. This reduces the communication overhead required, and allows P2P PSO algorithms to continue to function during periods of network downtime.

When working on a P2P network, we deal with a dynamically changing environment in which nodes dynamically enter and leave the network. This is known as the **churn** effect and is one of the chief concerns addressed in this paper. We assume that the optimisation starts with a set of computing resources and then analyze the churn effect by adding some new nodes to the system. The challenge here is to make use of the approximated front obtained so far and estimate the new positions of particles in the new nodes. We investigate four different methods and compare them on a set of test functions. The results show that the way the particles are initialised has a great impact on the quality of

Ian Scriven is with the School of Engineering, Griffith University, Brisbane, Queensland, Australia (email: Ian.Scriven@griffith.edu.au).

Andrew Lewis is with the School of Information and Communication Technology, Griffith University, Brisbane, Queensland, Australia (email: A.Lewis@griffith.edu.au).

Sanaz Mostaghim is with Institute AIFB, University of Karlsruhe, Germany (email: mostaghim@aifb.uni-karlsruhe.de).

978-1-4244-2959-2/09/$25.00 ©

the solutions.

This paper is organised as follows. In the next section, we briefly study multi-objective particle swarm optimisation and the P2P approach. In Section III, the churn effect is addressed in the paper and we proposed four different approaches. Section IV and V are dedicated to experiments and the results. Section VI concludes the paper.

II. P2P Multi-Objective Particle Swarm Optimisation

A. Particle Swarm Optimisation

The particle swarm optimisation algorithm, introduced by Kennedy and Eberhart in 1995 [13], utilises a population of potential solutions (called particles), which move around the design space in a search for optimal solutions. The movement of these particles is governed by two equations:

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t)(y_{ij}(t) - x_{ij}(t)) + c_2 r_{2j}(t)(\hat{y}_{ij}(t) - x_{ij}(t)) \quad (1)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \quad (2)$$

Here, $v_{ij}(t)$ is the velocity of particle *i* in dimension *j* at time *t* and $x_{ij}(t)$ is the position of particle *i* in dimension *j* at time *t*. The velocity of a particle depends on both the best position that particle has found to time *t*, $y_{ij}(t)$, and the best solution the entire swarm has found to time *t*, $\hat{y}_{ij}(t)$. Selecting $\hat{y}_{ij}(t)$ depends on the topology selected for the particles. We select star topology and thereby $\hat{y}_{ij}(t)$ is the best found solution in the entire population. However, when we deal with multi-objective problems, there is no one single optimum and the $\hat{y}_{ij}(t)$ has to be selected from a set of non-dominated solutions [14]. Also, the personal best $y_{ij}(t)$ solutions is updated when the new position is better than the old personal best in terms of the domination criterion.

The inertial weight, *w*, and c_1 and c_2 are constants (usually defined in the context of multi-objective particle swarm optimisation as 0.4, 1 and 1, respectively) used to control the impact of the previous, local, and global components in velocity equation (1). The vector *r* is a vector of random numbers evenly distributed between zero and one generated for each particle at each time step *t*. Once new particle velocities have been calculated, the position of each particle is updated as in (2).

A multi-objective particle swarm optimisation algorithm (MOPSO) starts from an initial population of particles. The particles get iteratively updated by using Equations (1) and (2) over several generations. In each iteration, the non-dominated solutions obtained so far are stored in an external population called the **archive**. The MOPSO is run until a stopping criterion such as maximum number of generations is met.

B. P2P-MOPSO

For running a MOPSO on a P2P network, a population of particles is assigned a to each node in the network as already proposed by Scriven *et al.* [15], [16]. These populations (typically of small size) are called sub-swarms. Individual sub-swarms share their knowledge of the solution space using epidemic or gossip based communication [17]. When a node enters the network, it receives a job description from one of the other nodes. If the job description is suitable, it runs a MOPSO, otherwise it leaves the network. Running the MOPSO starts by initialisation of the particles in the population. After that the particles are updated. At this time the node sends the new found positions to the other nodes and performs the optimisation until a stopping criterion is met. At the end, the set of non-dominated solutions found in a node are sent to the other nodes. This algorithm is represented in pseudocode as Algorithm 1 below. There is no limit imposed on the size of the sub-swarms, which can be as low as one if there are enough nodes on the grid to handle the desired swarm size in this way. Smaller sub-swarms are preferred, to help reduce the effect of mid-iteration node failures.

Algorithm 1 P2P-MOPSO

```
Receive job description
if this node is suitable then
  Initialise population
  repeat
    for all particles in subswarm do
      Calculate new particle position and velocity
      Evaluate particle
      Update archive of non-dominated solutions
    end for
    Propagate results to neighbours
  until Termination condition met
end if
```

In P2P particle swarm algorithms such as this one, node failure will not cause algorithm failure thanks to the de-centralised nature of the algorithm. However, in highly dynamic environments like those that may be encountered when dealing with P2P-based computational grids, continual node losses will cause the performance of the algorithm to degrade as less resources are available. To combat this, new nodes must be brought into the system to replace those lost. In order to utilise these nodes, however, new particles must be generated for the nodes to operated on, as the previous particle positions and trajectories will be lost with the outgoing nodes.

C. Discussion

The generation of new particle positions is a new problem which requires investigation. Particle generation during the initialisation stage of evolutionary optimisation algorithms has been investigated, with the general opinion tending towards random initialisation of particles when no information on the problem space is available [18]. Some studies, however, have shown that with some prior knowledge of the problem space, intelligent positioning of the initial population leads to faster convergence to the optimal solutions [19], [20], [21].

When generating new particles during the optimisation process, significant information regarding both the problem

space and the current swarm location will be available, and this paper investigates a number of methods of utilising this information to improve convergence speeds in unreliable distributed environments.

III. Addressing Churn in MOPSO

As we discussed in the Introduction, P2P networks are dynamically changing environments. Nodes may fail working, leave the network, or get overloaded. Also, some nodes can enter the system and have the potential to contribute to the optimisation task. In much of the work considering distributed or decentralised optimisation algorithms, the main focus has been on how the algorithms respond to the decentralisation and the issues of failure or churn have been ignored (see, for example, Janson *et al.* [22]. Where the possibility of node failure is admitted, it is often the case that "no special provisions are taken to deal with failure" [23], the assumption being the decentralised structure of the algorithm itself would provide robustness. There has been little or no investigation of how to most effectively utilise newly-arrived computing resources, the subject of the work presented here, with most researchers opting for simple, random initialisation. In the following, we assume that during the optimisation all the nodes have access to the current approximation of the Pareto-front. The methods to be discussed are aimed at industrial problems where the fitness evaluation is very time-consuming. Therefore, the added computational overheads of new particle initialisation can be considered to be negligible.

A. Random method

A very trivial approach for assigning new tasks to the newcomer nodes, is to randomly find the new positions in the search space. This approach can be improved if the random positions are selected through a reasonable exploration method such as a Binary Search method [10].

B. Filling Gaps in the approximated front

By using the information obtained from the approximated front obtained so far, new particle positions can be generated with aim to improve the quality by attempting to fill the gaps in the front.

The gaps in the front can be found easily by computing the distances between the objective values of neighboring solutions. In a two dimensional objective space, the Euclidean distances between the neighboring solutions has to be computed. The two neighbors with the largest distance can be selected as the solutions making the gap (Figure 1). In general, in an m, ($m \geq 2$), dimensional space, the gaps can be found in the same way by finding the nearest neighbor for each point in the set of non-dominated solutions by means of calculating the Euclidian distance between them in objective space, and selecting the pair for which this distance is a maximum.

The governing equations of the particle swarm algorithm, neglecting the inertia term, cause particles to search between two points in the search space - the local and global guide solutions. Using a similar approach when generating new particles is therefore an obvious choice. Instead of using local and global guides (the former being unavailable as we are generating a *new* particle), the two locations to search between can be taken as the two bounding solutions in the largest gap in the current approximated front e.g., solutions $\vec{x}'$ and $\vec{x}''$ in Figure 1.

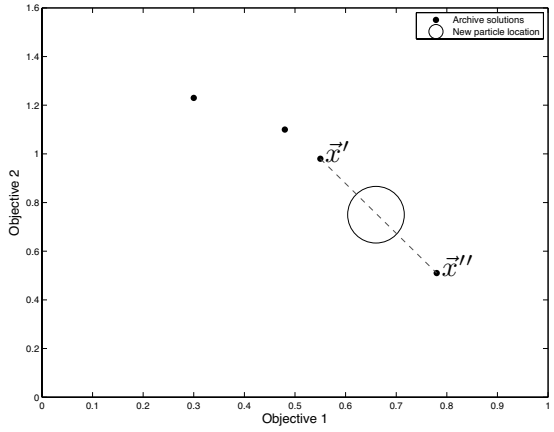

Fig. 1. Generating a new particle by filling the largest gap in the front

Once the bounding solutions are selected, each parameter x_k is calculated for the new particle using equation 3.

$$x_k = \frac{r_1 x'_k + r_2 x''_k}{r_1 + r_2} \tag{3}$$

The symbols r_1 and r_2 in (3) represent uniformly distributed random numbers between zero and one. Given the assumption of correlation between parameter and objective space (which is inherent in the particle swarm method), the newly generated particle should fall roughly between the two bounding solutions in the objective space.

C. Extending the Edges of the Front

The gap-filling approach may cause the swarm to cluster towards the centre of the approximated front, as it has a low chance to generate new particles outside the bounds of the current solutions on the edge of the front. Like the above approach, new particle positions can be generated with aim to improve the front obtained by extending its edges. The edge points are defined by sorting on an objective and choosing the extremes. New particle positions are generated by projection through the edge points, as illustrated in Figure 2. As the dimensionality of objective space increases, so do the number of edges to be considered and it is not clear that the function of the algorithm will be best served by choosing only extremal points. The effects of higher dimensionality on algorithm performance is yet to be fully investigated.

Once the extremal solutions are selected, each parameter x_k is calculated for the new particle using equation 4.

$$x_k = \frac{r_1 x'_k + r_2 (x'_k + (x'_k - x''_k))}{r_1 + r_2} \tag{4}$$

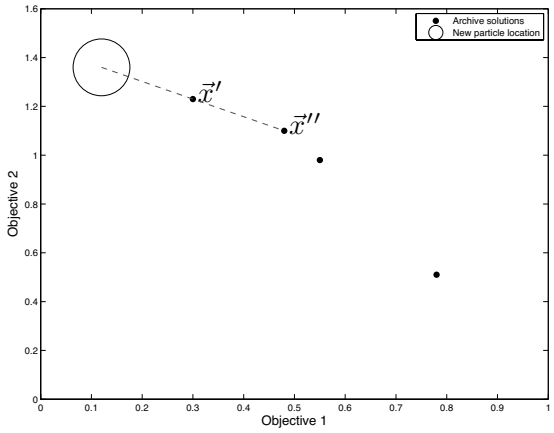

Fig. 2. Generating a new particle by extending the edges of the front

The symbols r_1 and r_2 in (4) again represent uniformly distributed random numbers between zero and one. The newly generated particle is this time be positioned past the current edges of the front in the solution space, again assuming correlation between the parameter and objective spaces.

D. Hybrid Method

Both of the discussed gap filling and edge extending approaches have advantages and disadvantages. The gap filling method promotes good coverage of the global front, but only between the currently located extremal solutions. The edge extending approach, on the other hand, promotes exploration outside the current approximation to the approximated front, but at the expense of convergence in the centre. It makes sense, then, to use a combination of the two approaches in order to gain the benefits of both while limiting the disadvantages. front, but only between the currently located extremal solutions. The edge extending approach, on the other hand, promotes exploration outside the current approximation to the approximated front, but at the expense of convergence in the centre. It makes sense, then, to use a combination of the two approaches in order to gain the benefits of both while limiting the disadvantages.

IV. Experiments

In order to examine the performance of the proposed approaches, a testing environment was created utilising a thirty particle PSO algorithm, with one particle being removed and replaced using each of the various approaches at set intervals. Four popular analytical test functions were used, as described in the following section. Test functions ZDT1, ZDT2 and DTZL were run for 200 iterations, with a particle replacement occurring every five iterations. We regard this level of volatility as being reasonably realistic. The FF test function was run for 70 iterations (this is a problem simpler and easier to solve than the others), with particle replacement every second iteration.

Each optimisation process is executed 200 times, with convergence (Hypervolume metric [24]) and coverage metric data being averaged to give reliable, accurate results. The coverage metric used is described later in this section. As well as the three approaches previously described, a baseline test was included where no particles were added or removed.

A. Test Functions

1) Test Function ZDT1: The test functions used are well-known in the literature [24]. The first, consisting of the three functions f_1, g and h, is of the form :

$$\begin{aligned} &\text{Minimise } \mathbf{t}(\mathbf{x}) = (f_1(x_1), f_2(\mathbf{x})) \\ &\text{subject to } f_2(\mathbf{x}) = g(x_2, ..., x_n) \cdot h(f_1(x_1), g(x_2, ..., x_n)) \\ &\text{where } \mathbf{x} = (x_1, ...x_n) \end{aligned} \tag{5}$$

This test function has a convex Pareto-optimal front given by (6)

$$\begin{aligned} f_1(x_1) &= x_1 \\ g(x_2, ..., x_n) &= 1 + 9 \cdot (\sum_{i=2}^{n} x_i)/(n-1) \\ h(f_1, g) &= 1 - \sqrt{f_1/g} \end{aligned} \tag{6}$$

where $n = 30$ and $x_i \in [0, 1]$. The Pareto-optimal front is formed with $g = 1$.

2) Test Function ZDT2: This test function has a non-convex Pareto-optimal front given by (7), and is of the same form as the above ZDT1 function.

$$\begin{aligned} f_1(x_1) &= x_1 \\ g(x_2, ..., x_n) &= 1 + 9 \cdot (\sum_{i=2}^{n} x_i)/(n-1) \\ h(f_1, g) &= 1 - (f_1/g)^2 \end{aligned} \tag{7}$$

where $n = 30$ and $x_i \in [0, 1]$. The Pareto-optimal front is formed with $g = 1$.

3) Test Function FF: This test function has a non-convex Pareto-optimal front given by (8)

$$\begin{aligned} f_1(\mathbf{x}) &= 1 - exp(-\sum_{i=0}^{n-1}(x_i - \frac{1}{\sqrt{n}})^2) \\ f_2(\mathbf{x}) &= 1 - exp(-\sum_{i=0}^{n-1}(x_i + \frac{1}{\sqrt{n}})^2) \end{aligned} \tag{8}$$

where $n = 10$ and $x_i \in [-4, 4]$.

4) Test Function DTZL: DTZL is a minimisation problem given by (9)

$$\begin{aligned} f_1(\mathbf{x}) &= (1 + g(\mathbf{x}_M)) \cos(x_i \pi/2) ... \cos(x_{M-1} \pi/2) \\ f_2(\mathbf{x}) &= (1 + g(\mathbf{x}_M)) \cos(x_i \pi/2) ... \sin(x_{M-1} \pi/2) \\ \text{where } g(x_M) &= \sum_{i=0}^{n-1}(x_i - 0.5)^2 \end{aligned} \tag{9}$$

where $n = 30$ and $x_i \in [0, 1]$. The Pareto-optimal front is formed with $x_i = 0.5$. Pareto-optimal solutions also satisfy

$$S = f_1(\mathbf{x})^2 + f_2(\mathbf{x})^2 = 1.$$

B. Algorithm Performance Metrics

1) Convergence Metric: The metric used to measure algorithm convergence (the closeness of a non-dominated set to the real Pareto-optimal front) in this study does so by measuring the volume/area of the objective space which is not dominated by a given approximation to the Pareto-optimal front. This is somewhat similar to the popular hypervolume metric, which evaluates the area/volume of the objective space which is dominated by a given set of non-dominated solutions [24]. This inverted hypervolume metric was used as the lower bounds for all objective in each of the test functions used are known to be zero, providing a convenient boundary for a hypervolume-like metric. In the case of minimisation problems, such as the test functions used for this paper, a smaller value for this metric denotes a closer approximation to the global Pareto-optimal front.

2) Coverage Metric: The existing diversity metrics [25] measure the statistical spread or evenness of the Pareto-front, rather than quantifying how well a given non-dominated set covers the objective space. An approximated front may be very evenly spread (that is, the distances between adjacent particles are relatively similar), however it may contain very few solutions and/or not adequately cover the entire range of possible objective values. In such cases existing diversity metrics would give a misleading evaluation of the quality of the approximate Pareto-optimal front.

The coverage metric proposed (similar to that in Mostaghim and Teich [26]) divides the objective space into sectors originating from the utopia point. The value of the diversity metric is then given as the ratio of these sectors which contain at least one member of the non-dominated set to the total number of sectors, or, mathematically:

$$\Psi = \frac{1}{N}\sum_{n=1}^{N}\psi_n \tag{10}$$

$$\text{where } \psi_n = \begin{cases} 1, \text{ if } \exists \mathcal{S} \in \mathcal{PF}, \alpha_{n-1} \leq \tan\frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} \leq \alpha_n \\ 0, \text{ otherwise} \end{cases}$$

A simple example of this metric can be seen in Figure 3. In this example, the solution space is divided into ten equal segments, with seven of these segments containing solutions in the current approximation to the Pareto-optimal front. This approximate Pareto front would therefore be classified as having a 0.7 or 70% coverage factor. Practically, more segments are used to give a higher resolution result - in this study, the same number of segments were used as there were particles in the swarm. Care should be taken extending the use of the metric to objective spaces of higher dimensionality. The metric operates essentially by taking an $N-1$ dimensional "slice" through the objective space, progressively hiding more information about the distribution of solutions in a projection onto a single dimension.

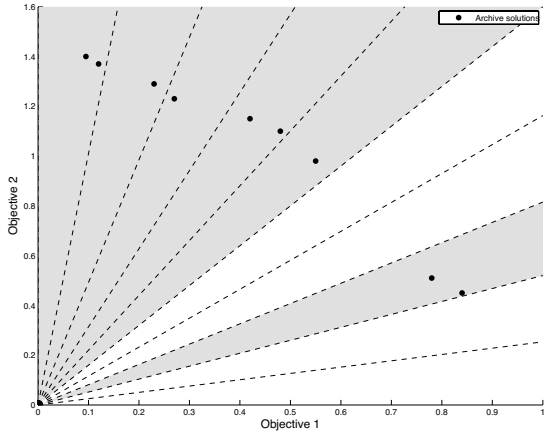

Fig. 3. Application of the proposed coverage metric to an example pareto front

V. Experimental Results

Box plots of the convergence and coverage metrics for each test function and algorithmic approach can be seen in Figures 4-11. The baseline algorithm is a multi-objective particle swarm optimisation algorithm using Sigma update [14]. Each of the methods proposed for addressing churn are applied as an extension to this baseline algorithm.

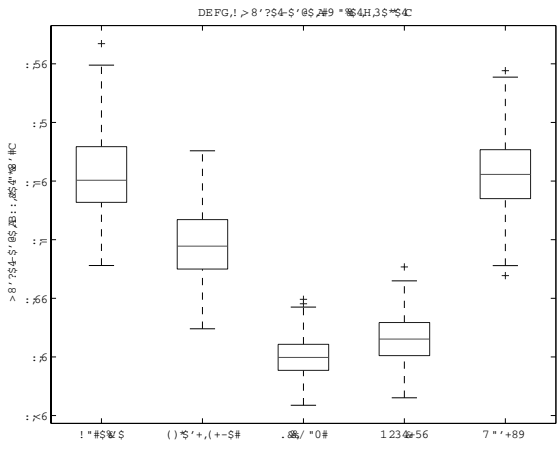

Fig. 4. Convergence of PSO algorithm on ZDT1 problem

It can be seen that the gap filling method significantly out-performs the edge extending approach in both the ZDT1 and ZDT2 test functions, in terms of both convergence and coverage of the Pareto-optimal front. Both approaches appear to result in better optimisation results than the baseline algorithm and the random generation method.

The two approaches faired slightly differently when applied to the FF test function. The gap filling method still resulted in increased algorithm convergence when compared to extending the edges. However, the difference between the

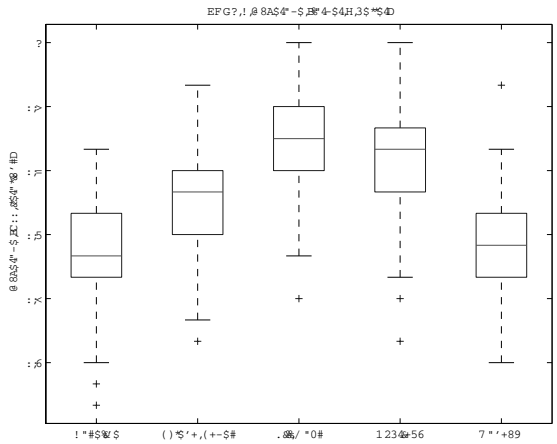

Fig. 5. Coverage of PSO algorithm on ZDT1 problem

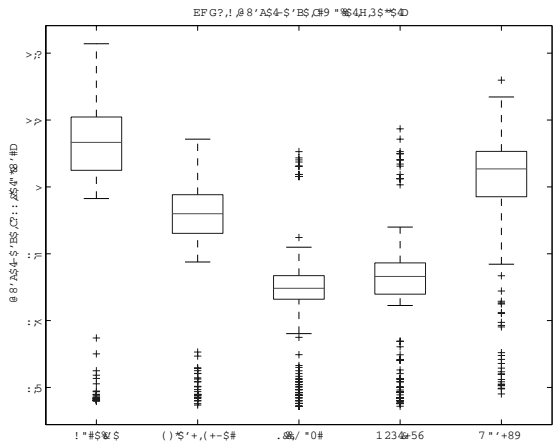

Fig. 6. Convergence of PSO algorithm on ZDT2 problem

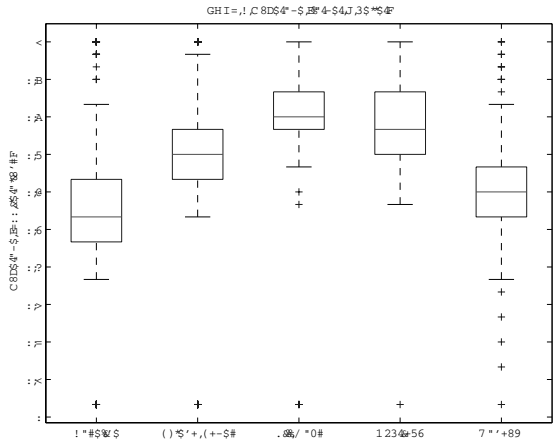

Fig. 7. Coverage of PSO algorithm on ZDT2 problem

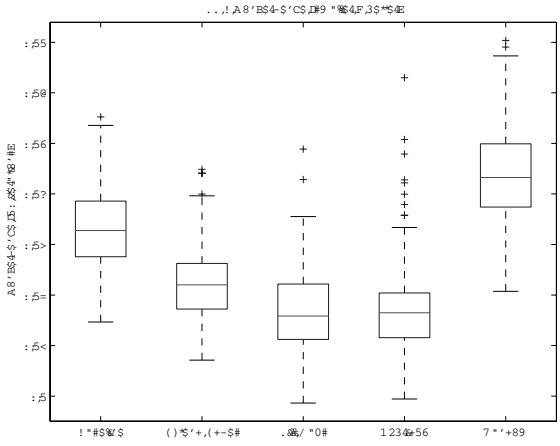

Fig. 8. Convergence of PSO algorithm on FF problem

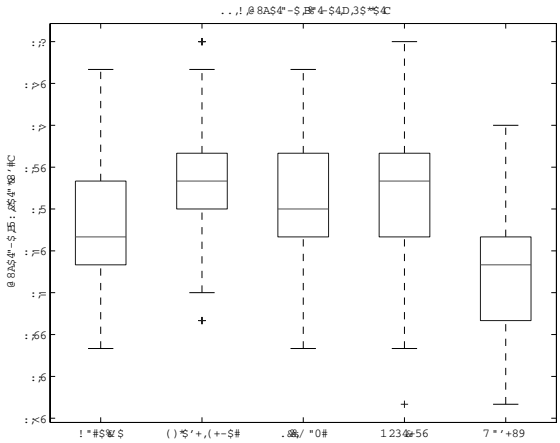

Fig. 9. Coverage of PSO algorithm on FF problem

two approaches here was significantly less than for the ZDT functions. Unlike the results for two ZDT functions, here the edge extending approach achieves, on average, better coverage of the Pareto front than the gap filling method for generating new particles.

The results for the DTZL test function are comparable to those obtained using the ZDT1 and ZDT2 functions, with the gap filling approach being the top performer.

The hybrid approach, which generated new particles using the gap filling approach 75% of the time and used the edge extending approach the other 25%, can be seen to give the most consistent results across all four test functions. In all cases, the results produced by the hybrid approach compare quite favorably with those of the top performing method.

VI. Conclusions and Future Work

In order to fully utilise available resources in decentralised, fault-prone distributed environments, particle swarm optimisation algorithms must be able to efficiently generate

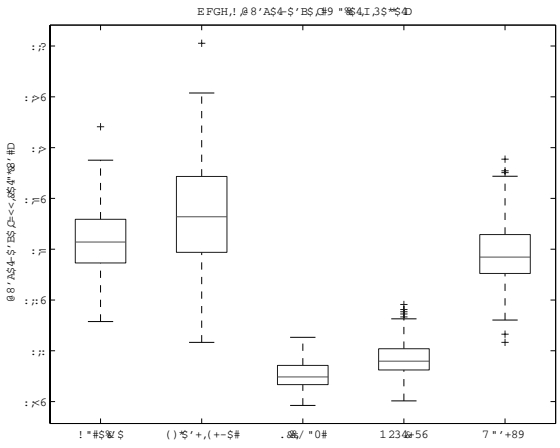

Fig. 10. Convergence of PSO algorithm on DTZL problem

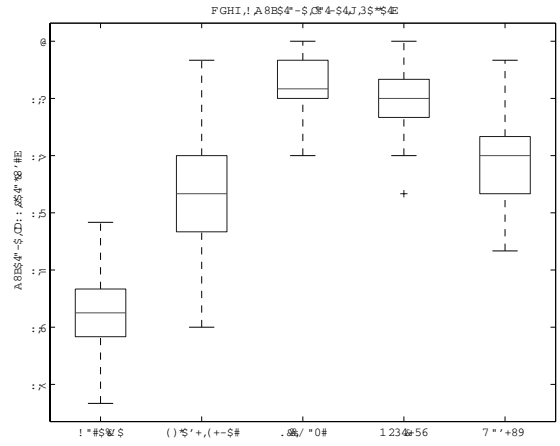

Fig. 11. Coverage of PSO algorithm on DTZL problem

particles to populate new nodes when they become available. This paper investigated two mechanisms, Pareto-front gap filling and edge extending, and further proposed a hybrid approach, in order to achieve this.

The proposed approaches were tested using a number of popular analytical optimisation test functions, with convergence and coverage metrics being averaged over a number of test runs to evaluate the efficacy of each approach. Generating new particles by attempting to fill gaps in the Pareto-optimal front generally performed better than extending the front's edges. However, it was shown that extending the edges of the Pareto front can be more advantageous in certain scenarios. From this, a hybridised method was formulated to utilise both approaches while favoring gap filling, and this hybrid method was shown to perform consistently well across all problems used for testing.

It was noted that in almost every case the algorithms that removed and replaced particles using either of the proposed approaches out-performed the general PSO baseline algorithm. Whether hybridising the generic PSO algorithm with the gap filling and edge extending mechanisms would lead to overall improvements in performance on a wider range of problems is a matter of ongoing investigation. In the experiments reported in this paper, a realistically moderate degree of volatility was simulated. It may be interesting to assess the performance of the modified algorithms in environments where volatility is extreme.

REFERENCES

[1] Y. Censor and S. A. Zenios, *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, New York/Oxford, 1997.

[2] E.-G. Talbi, S. Mostaghim, T. Okabe, H. Ichibushi, G. Rudolph, and C. C. Coello, *Parallel Approaches for Multiobjective Optimization*, pp. 349–372. Springer Verlag, 2008.

[3] J. L. J. Laredo, P. A. Castillo, A. M. Mora, and J. J. Merelo, "Evolvable agents, a fine grained approach for distributed evolutionary computing: walking towards the peer-to-peer computing frontiers," *Soft Computing*, vol. 12, no. 12, pp. 1145–1156, 2008.

[4] J. L. J. Laredo, A. E. Eiben, M. van Steen, and J. J. Merelo, "On the run-time dynamics of a peer-to-peer evolutionary algorithm," in *Parallel Problem Solving from Nature X PPSN*, pp. 236–245, 2008.

[5] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, Massachusetts: Addison-Wesley Publishing Company, Inc., 1989.

[6] S. Mostaghim, J. Branke, and H. Schmeck, "Multi-objective particle swarm optimization on computer grids," in *The Genetic and Evolutionary Computation Conference*, vol. 1, pp. 869–875, 2007.

[7] J. Branke, H. Schmeck, K. Deb, and M. Reddy, "Parallelizing Multi-Objective Evolutionary Algorithms: Cone Separation," in *IEEE Congress on Evolutionary Computation*, pp. 1952–1957, 2004.

[8] K. Deb, P. Zope, and A. Jain, "Distributed computing of pareto-optimal solutions with evolutionary algorithms," in *International Conference on Evolutionary Multi-Criterion Optimization*, pp. 534–549, 2003.

[9] D. A. V. Veldhuizen, J. Zydallis, and G. B. Lamont, "Considerations in engineering parallel multiobjective evolutionary algorithms," in *IEEE Transactions on Evolutionary Computation, Vol. 7, No. 2*, pp. 144–173, April 2003.

[10] S. Mostaghim, J. Branke, A. Lewis, and H. Schmeck, "Parallel multi-objective optimization using a master-slave model on heterogeneous resources," in *IEEE*, (Congress on Evolutionary Computation (CEC)), JUN 2008.

[11] I. Scriven, D. Ireland, A. Lewis, and S. M. ad J. Branke, "Asynchronous multiple objective particle swarm optimisation in unreliable distributed environments," in *IEEE Congress on Evolutionary Computation (CEC)*, 2008.

[12] W. R. M. U. K. Wickramasinghe, M. van Steen, and A. E. Eiben, "Peer-to-peer evolutionary algorithms with adaptive autonomous selection," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 1460–1467, 2007.

[13] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, 1995.

[14] S. Mostaghim and J. Teich, "Strategies for finding good local guides in multi-objective particle swarm optimization," in *IEEE Swarm Intelligence Symposium*, pp. 26–33, 2003.

[15] I. Scriven, A. Lewis, D. Ireland, and J. Lu, "Distributed multiple objective particle swarm optimisation using peer to peer networks," in *IEEE Congress on Evolutionary Computation (CEC)*, 2008.

[16] I. Scriven, J. Lu, and A. Lewis, "An efficient peer-to-peer particle swarm optimiser for EMC enclosure design," in *The 13th Biennial IEEE Conference on Electromagnetic Field Computation (CEFC)*, 2008.

[17] I. Gupta, A. J. Ganesh, and A.-M. Kermarrec, "Efficient and adaptive epidemic-style protocols for reliable and scalable multicast," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 7, pp. 593–605, 2006.

[18] X.-F. Xie, W.-J. Zhang, and Z.-L. Yang, "Adaptive particle swarm optimization on individual level," in *Signal Processing, 2002 6th International Conference on*, vol. 2, pp. 1215–1218, 2002.

[19] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "A novel population initialization method for accelerating evolutionary algorithms," *Comput. Math. Appl.*, vol. 53, no. 10, pp. 1605–1614, 2007.

[20] P. J. Angeline, "Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences," in *Evolutionary Programming VII*, vol. 1447 of *Lecture Notes in Computer Science*, pp. 601–610, Springer Verlag, 1998.

[21] C. H. Chou and J.-N. Chen, "Genetic algorithms: initialization schemes and genes extraction," in *Fuzzy Systems, 2000. FUZZ IEEE 2000. The Ninth IEEE International Conference on*, vol. 2, pp. 965–968, 2000.

[22] S. Janson, D. Merkle, and M. Middendorf, "A decentralization approach for swarm intelligence algorithms in networks applied to multi swarm pso," *International Journal of Intelligent Computing and Cybernetics*, vol. 1, no. 1, pp. 25–45, 2008.

[23] M. Biazzini, M. Brunato, and A. Montresor, "Towards a decentralized architecture for optimization," in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, 2008.

[24] E. Zitzler, *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. Shaker, 1999.

[25] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Parallel Problem Solving from Nature VI (PPSN-VI)*, pp. 849–858, 2000.

[26] S. Mostaghim and J. Teich, "A new approach on many objective diversity measurement," in *Dagstuhl Proceedings number 04461*, (Dagstuhl, Germany), November 2004.