Decentralised Distributed Multiple Objective Particle Swarm Optimisation Using Peer to Peer Networks

Ian Scriven, Andrew Lewis, David Ireland and Junwei Lu

***Abstract*—This paper describes a distributed particle swarm optimisation algorithm (PSO) based on peer-to-peer computer networks. A number of modifications are made to the more traditional synchronous PSO algorithm to allow for fully decentralised, scalable and fault-tolerent operation. The modified algorithm uses staggered propagation of objective-space knowledge between sub-swarms to eliminate the need for a centralised data store. Analytical test functions are used to examine the performance of the proposed algorithm and its variations in comparison with a basic synchronous PSO implementation. The results clearly show the feasibility of decentralised particle swarm optimisation.**

I. Introduction

Computational optimisation is becoming more and more prevalent in a number of fields where simple analytical solutions to a problem are not possible. Typically, intelligent search algorithms are used, which attempt to locate optimal solutions to a problem involving a number of different parameters through systematic evaluation of a subset of all possible solutions. For problems with many parameters and extremely large search spaces, utilising distributed computing resources becomes an attractive proposition, especially when individual solutions require significant computational time.

Typically, specialised supercomputers or computer clusters are used for these applications, however increasing demand for such resources often means limited availability and long wait times. To address this problem, a number of systems have been proposed which bring together idle computing resources (desktop PCs, laptops, servers etc.) over a peer to peer network to form temporary ad-hoc computational grids [1]. While these systems can significantly increase the computational power available, they pose a number of problems to traditional distributed optimisation algorithms. This paper discusses these issues, and proposes a decentralised, distributed multiple objective particle swarm optimisation algorithm designed to operate within a peer-to-peer grid environment.

II. Distributed Optimisation Algorithms

The majority of distributed optimisation algorithms make use of a master-slave architecture [2], [3], [4], where a single controlling processor (the master) runs the actual optimisation algorithm, while all particle evaluations are performed on secondary (slave) processors [5]. Intelligent search algorithms such as the particle swarm algorithm and the genetic algorithm are well suited to this type of distribution, as they involve a large number of individual, unrelated, computationally intensive calculations. As such, this type of system allows near-linear speedup with respect to the amount of computational power available.

The master-slave architecture is, however, not well suited to the highly dynamic and heterogeneous environment likely to be encountered in an ad-hoc peer-to-peer grid. The non-dedicated nodes are unreliable, and in some cases not even the existence of a dedicated, constantly available master processor can be relied upon. Heterogeneity in node specifications result in diverse solution evaluation times, which negatively impacts the performance of many parallel algorithms that utilise the master-slave approach [6].

Network connections in peer-to-peer networks are typically far less reliable than interconnects in dedicated high performance computing resources. The peer-to-peer grid may be divided into multiple segments for significant periods of time, and any algorithms should be able to continue to function with minimal slowdown in the event of such network interruptions. The nodes making up the ad-hoc peer-to-peer grid are usually not dedicated, and as such their constant availability is not guaranteed. Applications being run by local users will typically have higher priorities than grid jobs, and the nodes may be restarted or halted without warning. As such, any optimisation algorithm must be highly fault tolerant. The loss of a significant number of nodes must not seriously impact the operation of the overall algorithm.

Limited investigation has been made into peer-to-peer based genetic algorithms using gossip-based communication protocols and island models [7], [8], where each node runs a genetic algorithm with a unique population. Peer-to-peer based particle swarm optimisation (PSO) algorithms have little to no attention in the literature at the time of writing, which is somewhat surprising considering the growing popularity of particle swarm method. PSO algorithms are highly suitable for use in distributed peer-to-peer environments, as the fixed population size negates the population size control issue encountered in decentralised genetic algorithms [7]. P2P PSO algorithms also require less information about the entire population when generating new possible solutions. Whereas genetic algorithms 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 should reduce the communication overhead

Ian Scriven, David Ireland and Junwei Lu are with the School of Engineering, Griffith University, Brisbane, Queensland, Australia (email: {I.Scriven, D.Ireland,J.Lu}@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).

required, and allow P2P PSO algorithms to continue to function during periods of network downtime.

III. Peer-to-Peer Multiple Objective Particle Swarm Optimisation Algorithm

The particle swarm optimisation algorithm utilises a population of potential solutions, or particles, which move around the design space with every iteration in a search for 'good' 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, $\hat{y}_{ij}(t)$. The inertial component is scaled by constant w, and c_1 and c_2 are constants, usually defined between 0.5 and 3.0, used to control the impact of the 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).

As stated in the previous section, the particle swarm optimisation algorithm is inherently well suited to the dynamic, heterogeneous environments encountered in peer-to-peer ad hoc grids. Individual particles require no information on the current state of the other particles in the swarm, they simply need to have a general idea of where in the problem space 'good' solutions can be found. The peer-to-peer multiple objective particle swarm optimisation (P2P-MOPSO) algorithm proposed in this paper makes use of small sub-swarms of particles located on each node in the peer-to-peer grid. There is no limit imposed on the size of these 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.

The P2P-MOPSO algorithm has been implemented on top of the FreePastry peer-to-peer framework developed at Rice University, which provides all underlying network functionality [9]. Support exists for running multiple optimisation jobs simultaneously, and these jobs are initiated a job description to all available nodes in the peer-to-peer network using an efficient multicast algorithm [10].

When forming ad-hoc grids in heterogeneous environments, it is important to consider that some nodes may be unsuitable for certain jobs for a number of reasons, including software availability or system architecture. For this reason the job description can contain requirements that a node must meet to be used for the job. When a node receives such a job description, it will access its own suitability for that job [11]. If deemed suitable, a node will immediately initiate and begin operating its own unique sub-swarm of particles, following the procedure outlined in Fig. 1. The size of the sub-swarm is currently specified in the job description, however a mechanism is in current development to allow the sub-swarm size to be adjusted on a node-by-node basis to answer the question of node heterogeneity. For all intents and purposes, each node can be considered to be operating its own distinct realisation of a traditional particle swarm algorithm, albeit with an unusually small swarm size.

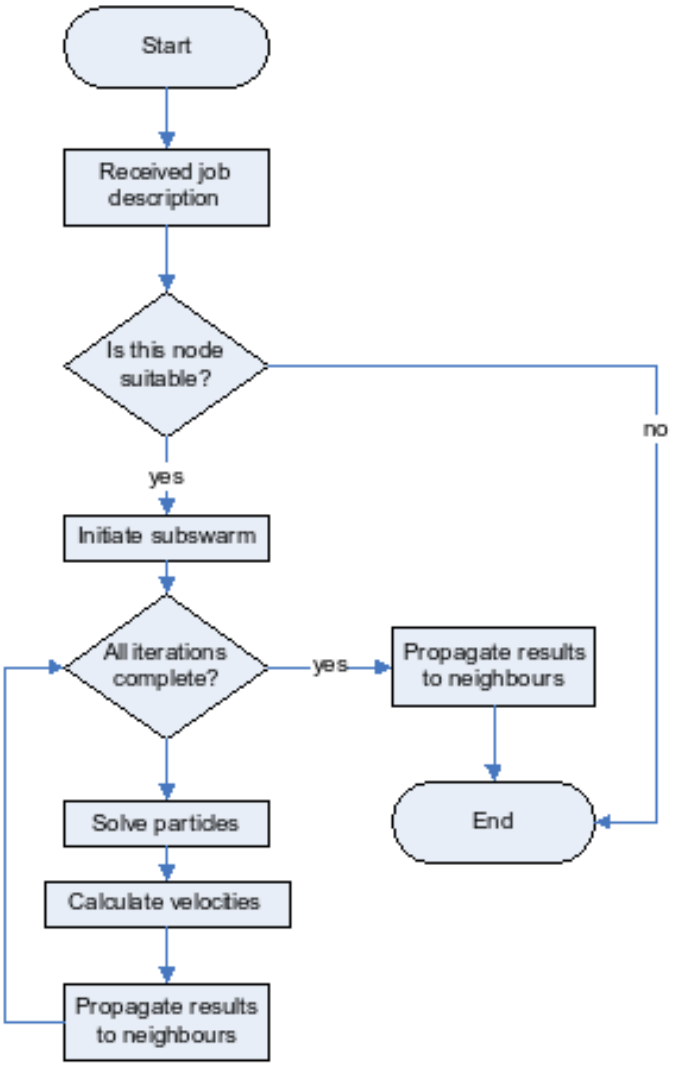

Fig. 1. Flowchart for main P2P particle swarm algorithm loop. This procedure is executed independently on all nodes which receive the job description from the initiating node.

Particle swarm algorithms utilising very small swarm sizes generally perform quite poorly, especially in complex problem spaces [12], however this problem is overcome here through the sharing of results between each individual sub-swarm in the peer-to-peer grid. This propagation is achieved through the use of gossip or epidemic based communication [13]. Instead of simultaneously sharing all information with all nodes, gossip based communication means that the

global swarm information is gradually propagated around the peer-to-peer network. By exchanging solution information in this way, the communication overhead required by the P2P-MOPSO algorithm can be greatly reduced, provided the delayed propagation does not impact too severely on algorithm convergence. Delaying solution information in this manner has the side effect of making the algorithm somewhat asynchronous. This will result in some slowdown in algorithm convergence as a function of iterations completed, however it will lead to improved parallel efficiency. Ignoring communication overheads, the P2P-MOSPO algorithm should theoretically achieve one hundred percent parallel efficiency, as there will be no delay while waiting for other nodes to complete calculations.

IV. Testing and Verification

The P2P-MOPSO system was implemented on a test-bed of twenty nodes, with each node running a sub-swarm of five particles for two hundred and fifty iterations, giving a total of 25,000 evaluations. The performance of the P2P-MOPSO system was evaluated using an analytical test function designed to give a fair representation of the types of real-world problems the system is likely to encounter. The selected test function, consisting of the three functions f_1, g and h, is of the form [14]:

$$\begin{aligned} &\text{Minimise } \mathbf{t}(\mathbf{x}) = (f_1(x_1), f_2(\mathbf{x})) \\ &\text{subject to } f_2(\mathbf{x}) = g(x_2, ..., x_n) \centerdot h(f_1(x_1), g(x_2, ..., x_n)) \\ &\text{where } \mathbf{x} = (x_1, ...x_n) \end{aligned} \tag{3}$$

This test function has a convex Pareto-optimal front given by (4)

$$\begin{aligned} f_1(x_1) &= x_1 \\ g(x_2, ..., x_n) &= 1 + 9 \centerdot (\sum_{i=2}^{n} x_i)/(n-1) \\ h(f_1, g) &= 1 - \sqrt{f_1/g} \end{aligned} \tag{4}$$

where $n = 30$ and $x_i \in [0, 1]$. The Pareto-optimal front is formed with $g = 1$ [14].

For comparison, a generic synchronous PSO was implemented with one hundred particles, in order to have an equivalent number of total evaluations and evaluations per iteration as the P2P-MOPSO system. Both the synchronous PSO and the P2P-MOPSO algorithm were run multiple times to ensure accurate results. To investigate the effectiveness of the gossip-based node communication system, propagation constants of 25%, 50% and 100% were used, leading to nodes sending solution information to five, ten and twenty nodes per iteration, respectively.

A convergence factor metric was used which measured area of the potential solution space dominated by an approximation to the Pareto-optimal front as a ratio of the solution space dominated by the actual Pareto-front. An approximation to the Pareto-optimal front that is equal to the actual Pareto-front has a convergence factor of one, and a solution set that has not converged will have a convergence factor less than one (but greater than zero). This convergence factor was monitored as function of iterations completed for both the synchronous PSO and the P2P-MOPSO algorithm. Fig. 2 shows this convergence factor data for all algorithm variations, while Fig. 3 shows the final Pareto-front approximations obtained by each algorithm.

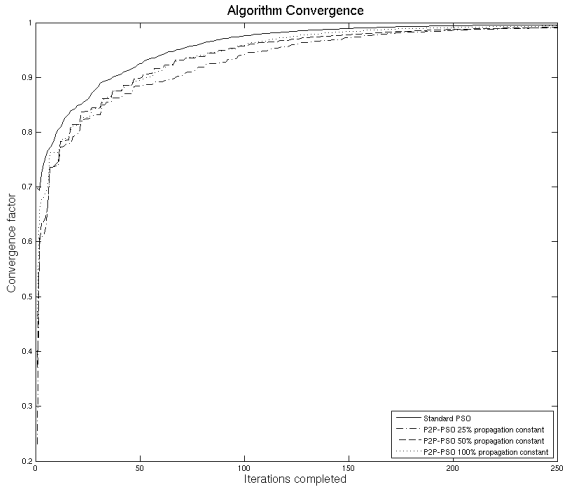

Fig. 2. Algorithm convergence for propagation constants of 25%, 50% and 100%, with traditional synchronous PSO performance provided for comparison.

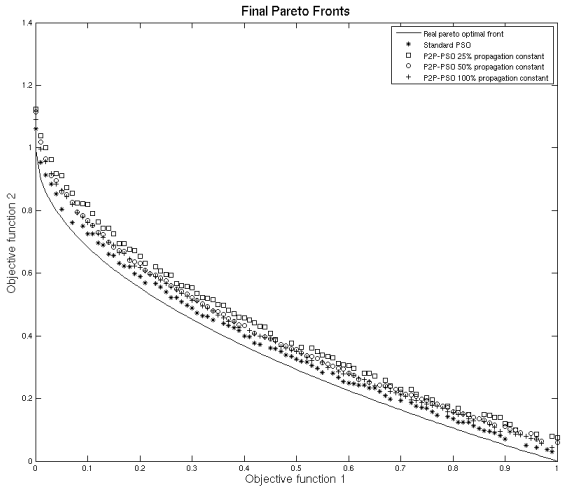

Fig. 3. Final Pareto-front approximations provided by the various PSO implementations.

These results show that the performance of the P2P-PSO is quite comparable to that of the regular synchronous PSO implementation. While the synchronous PSO converges somewhat faster than the P2P-PSO in terms of iterations completed, as shown in Fig. 2, the inherent asynchronous nature of the P2P-PSO will decrease the total execution time of the algorithm when compared to a traditional parallel

PSO implementation using the master-slave model. As such, the final approximations to the Pareto-front shown in Fig. 3 will be reached faster by the P2P-MOPSO algorithm. If executed for the same fixed amount of time, the P2P-MOPSO algorithm should obtain results similar to those reached by the synchronous particle swarm.

Varying the propagation constant used by the P2P-MOPSO algorithm demonstrated the effect this type of communication has on overall algorithm performance. By passing solution information to more neighbours after each iteration, the performance of the algorithm can be seen to improve, both in terms of convergence (Fig. 2) and the final solution reached (Fig. 3). However, the results obtained seem to suggest that the performance increase gained diminishes as the propagation constant is raised higher and higher. The above figures show that the performance of the P2P-MOSPO is quite similar for propagation constants of 50% and 100%, and given the significant communication overhead difference between the two, a propagation constant of 50% would be preferred.

V. Conclusions

In this paper a distributed peer-to-per based multiple objective particle swarm optimisation algorithm was presented. A number of changes have been made to the traditional synchronous particle swarm algorithm to allow scalable, fault tolerant operation in a dynamic, heterogeneous environment. By using gossip protocol based communication, the need for a distributed data store is removed, which would otherwise be a possible point of critical failure.

Testing results using analytical test functions were presented to show that the P2P-PSO algorithm is only slightly out performed by a traditional synchronous particle swarm implementation, however the inherent asynchronous nature of the P2P-PSO will lead to increased parallel efficiency, thereby reducing total algorithm execution time, which will work to negate the slightly slower convergence rate of the algorithm.

The effect of using gossip-based protocols for communication between nodes was shown to have only a limited negative effect of overall algorithm performance, while significantly reducing the required communication overhead. However, in the implementation discussed in this paper, propagation only occurs at the end of each sub-swarm iteration. Considering many engineering problems rely on solvers that may take many minutes or even hours to complete, it may be preferable to perform the information propagation in a concurrent fashion at set time intervals. Using gossip protocols, sub-swarm knowledge could be passed to all other nodes in the network during solver execution in a very efficient manner, eliminating the slight negative effect delayed knowledge propagation appears to have on the P2P-MOPSO algorithm.

Additional future work is planned to extend the system to allow additional nodes to be added during the optimisation procedure, which will involve investigation methods for generating new particle positions from existing solutions. Elements of genetic algorithms may prove useful in this regard, possibly turning the P2P-MOPSO into a hybrid algorithm.

References

[1] M. Smith, T. Friese and B. Freislebel, "Towards a Service-Oriented Ad Hoc Grid," *Proceedings of the 3rd International Symposium on Parallel and Distributed Computing*, 2004, pp. 201-208.

[2] B. Koh, A. D. George, R. T. Haftka and B. J. Fregly, "Parallel asynchronous particle swarm optimization," *International Journal for Numerical Methods in Engineering*, vol. 67, no. 4, pp. 578-595, 2006.

[3] G. Venter and J. Sobieszczanski, "Multidisciplinary optimization of a transport aircraft wing using particle swarm optimization," *Structural and Multidisciplinary Optimization*, vol. 26, no. 1-2, pp. 121-131, 2004.

[4] J. F. Schutte, J. A. Reinbolt, B. J. Fregly, R. T. Haftka and A. D. George, "Parallel global optimization with particle swarm algorithm," *International Journal for Numerical Methods in Engineering*, vol. 61, pp. 2296-2315, 2004.

[5] S. Mostaghim, J. Branke and H. Schmeck, "Multi-Objective Particle Swarm Optimization on Computer Grids," *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 2007, pp. 869-874.

[6] I. Scriven, D. Ireland, A. Lewis, S. Mostighim and Jürgen Branke, "Asynchronous Multiple Objective Particle Swarm Optimisation in Unreliable Distributed Environments," *submitted to IEEE Congress on Evolutionary Computation (CEC)*, 2008.

[7] W. R. M. U. K. Wickramasinghe, M. van Steen and A. E. Eiben, "Peer-to-peer Evolutionary Algorithms with Adaptive Autonomous Selection," *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 2007, pp. 1460-1467.

[8] J. C. C. Litrán, X. Défago and K. Satou, "Asynchronous Peer-to-Peer Communication for Failure Resilient Distributed Genetic Algorithms," *Proceedings of the 15th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS)*, 2003, pp. 769-773.

[9] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 2001, pp. 329-350.

[10] M. Castro, P. Druschel, A-M. Kermarrec and A. Rowstron, "SCRIBE: A large-scale and decentralised application-level multicast infrastructure," *IEEE Journal on Selected Areas in Communication (JSAC)*, vol. 20, no. 8, pp. 100-110, 2002.

[11] I. Scriven, A. Lewis, M. Smith and T. Friese, "Resource Evaluation and Node Monitoring in Service Oriented Ad-hoc Grids," *Sixth Australasian Symposium on Grid Computing and e-Research (AusGrid2008)*, 2008.

[12] A. Carlisle, G. Dozier, "An Off-The-Shelf PSO", *Proceedings of the 2001 Workshop on Particle Swarm Optimization*, 2001, pp. 1-6.

[13] 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.

[14] E. Zitzler, "Evolutionary algorithms for multiobjective optimization: methods and applications," Thesis (doctoral), University of Zurich, Zurich, Switzerland, 1999.