Optimising Large Scale Public Transport Network Design Problems using Mixed-Mode Parallel Multi-Objective Evolutionary Algorithms

Ian M Cooper, Matthew P John, Rhydian Lewis, Christine L Mumford and Andrew Olden

Abstract— **In this paper we present a novel tool, using both OpenMP and MPI protocols, for optimising the efficiency of Urban Transportation Systems within a defined catchment, town or city. We build on a previously presented model which uses a Genetic Algorithm with novel genetic operators to optimise route sets and provide a transport network for a given problem set. This model is then implemented within a Parallel Multi-Objective Genetic Algorithm and demonstrated to be scalable to within the scope of real world, [city-wide], problems. This paper compares and contrasts three methods of parallel distribution of the Genetic Algorithm's computational workload: a job farming algorithm and two variations on an 'Islands' approach. Results are presented in the paper from both *single* and *mixed* mode strategies. The results presented are from a range of previously published academic problem sets. Additionally a real world inspired problem set is evaluated and a visualisation of the optimised output is given.**

I. Introduction

To fulfil future European public transport policies [1] and to support the transportation challenges of Smart Cities [2], more practical, efficient and economic public transport networks need to be delivered.

Current public transport networks use sets of routes (route sets), along which vehicles regularly travel. These route sets have generally been designed using local knowledge and simple guidelines [3]. Over the past 20 to 50 years, many of these route sets have not been updated even though land use patterns have changed considerably in this time [3]; the most notable of these changes being the migration of businesses away from town centres into surrounding suburban areas, and additional housing developments.

The need for automated computer based tools for the design and evaluation of public transport networks is ever increasing (see [4], [5]), especially as public transport operators are facing mounting pressure to deliver a more user satisfactory service whilst concurrently facing funding limitations. The challenge of optimising public transport networks is highly complex and the search space involved in city wide problem sets is enormous. This coupled with multiple network operator constraints creates hugely computationally expensive problems.

Ian Cooper, Matthew John, Christine Mumford and Andrew Olden are with the School of Computer Science, Cardiff University, United Kingdom. Rhydian Lewis and Matthew John are with the School of Mathematics, Cardiff University, United Kingdom. (email: {i.m.cooper, c.l.mumford, a.olden}@cs.cf.ac.uk, {JohnMP, lewisr9}@cardiff.ac.uk).

This work was partially funded by HPC WALES under grant no. HPCW 167.

The Urban Transit Network Design Problem (UTNDP) involves determining an optimal set of routes and schedules for public transportation systems such as bus, rail and tram networks. Early papers on the UTNDP considered only highly specific problem sets [6], [7]. Mandl [8], [9] concentrated on the more generic design of route sets, developing a two-stage solution, first a set of feasible routes is generated, then heuristics are applied to improve the quality of the routes in this set. Following Mandl's pioneering work, heuristic methods have been widely used to solve the UTNDP, e.g [10], [11]. With the advancement of computing technology over the last two decades, however, metaheuristic techniques have become increasingly popular for solving these problems, particularly Genetic Algorithms (GAs) [12], [13], [14], [15], tabu search and simulated annealing [16], [17].

The UTNDP problem can be separated into five main stages for bus service planning: network design, frequency setting, timetable development, bus scheduling and driver scheduling [10]. Given that each stage of the UTNDP is NP-hard [18], it is usually considered impractical to solve all the stages simultaneously.

The problem we tackle focuses on the network design element, which is tasked with determining an efficient set of routes on an already established road (or rail) network, usually with previously identified pickup and drop off locations (e.g. bus stops); this is referred to as the Urban Transport Routing Problem (UTRP).

In this paper we build upon recently published work by Mumford [19], and work by John et al. [20] that has presented a Multi-Objective GA approach to the UTRP. Their work considers the trade-offs between passenger and operator costs by producing approximate Pareto optimal sets for consideration by a human decision maker and provides specialised heuristics and operators with an associated objective function of solution quality.

The work of John et al. [20] provides an approximate Pareto set for a number of problem sets to illustrate the quality of the algorithms presented; these problem sets include the previously published, and publicly available, problem sets of Mandl [9] and Mumford [19] (of which there are 4). Generally, comparative work in the literature has been limited to Mandl's 15 vertex problem set [9], but other problem sets have been reported, e.g. Bagloee and Ceder [3] tackled real sized road networks using a combination of heuristics, a GA and an ant-system.

As the size and complexity of these problem sets increase, the running time of John et al.s serial algorithm becomes unacceptable. John et al's serial implementations of a multi-objective approach to the Mandl problem set takes approximately 3 seconds, whereas the serial implementation of the Mumford3 problem set takes 44 hours, despite the underlying graph containing only 127 nodes (e.g. bus stops). In order to provide a tool capable of solving realistically sized UTRP's of approximately 2000 nodes, the algorithms must be efficiently parallelised and run on a High Performance Computer (HPC) system. To this end we describe, in this paper, three basic models of parallelism for the UTRP using the message passing standard MPI and the shared memory standard OpenMP. We then evaluate this work on the aforementioned problem sets. Additionally we also make use of a new problem set derived from real world data, as discussed in Section III.

Parallel implementations of GAs (PGAs) have been the subject of a great deal of research. Not only does the parallel nature of execution enable such algorithms to be run on clusters of computers, thus decreasing the run time, it additionally provides a structure that has shown to improve the underlying GA's quality in many cases [22]. There have been numerous parallel models proposed, such as segregation and reunification [23], but in this paper we restrict the discussion to a master slave panmitic approach and a cellular Islands approach. The panmitic model treats the population as a whole, evolving each generation using a choice from all population members, but farming out sub-generational computation tasks to worker nodes. The Islands model splits the population into sub-populations, evolving them separately with members being passed between sub-populations to maintain a collaborative approach. Both of these models have had their proponents: Gagné [24] arguing that the master slave model performed better than Islands on a Beowulf cluster, however other authors favour the Islands approach [22], [25].

The Islands approach has a long and popular history. For example, Alba and Troya [22] detail and discuss the advantages of asynchronous communications between the islands from a real time perspective, and Araujo et al. [25] consider five migration factors: number of migrants, frequency of migration, choice of immigrant solutions, replacement policy, and synchronisation of replacement. In [25] Araujo et al. warn of rapid convergence whilst discussing various simple immigrant choices such as choosing both migrant and replacement based on fitness. They counter this approach by choosing a random migrant, allowing diversity to be maintained in the sub-populations.

Approaches that involve more complex decision making include adaptive migrant choice based on the receiving island's need [26]. Araujo et al.'s *multikulti* approach introduces diversity considerations into the migration policy by looking at genotypic differences. As it is not practical for a receiving island to ask for a specific solution, the sending island sends a random selection of solutions to the receiving island and the receiving island chooses one (although all sent solutions are above a threshold value). Lardeux and Goëffon [27] propose a dynamic probability of migration from one island to another. If an island sends a solution and the average quality of the receiving island improves at the next generation, then the probability of that immigration route increases. They show an improvement for knapsack and MAX-SAT problems over a fixed migration policy. A fresh approach to the diversity problem is given by De-jong et al. [28], who convert a single objective problem to a multi-objective one with diversity as an extra objective.

Recent work on parallel GAs has concentrated on their ability to run on large scale cluster systems, with a focus on massive run-time improvements. In this field hybrid solutions have made a significant impact. Wang et al. [29] run a Hybrid PGA on benchmark single objective problems on a small system (4 nodes of 2 cores) using MPI communication between islands, whilst using the OpenMP to distribute the evaluation tasks in a master slave arrangement. They state that the main thread performs the Genetic Algorithm framework and all the genetic operators while others are responsible for the evaluation of the objective function. With their 8 cores they achieve a speed-up of 3.6 - 3.7. Narayanan et al. [30] also use the Islands approach for their GA's population and further parallelise bespoke Finite Element Analysis code among cores, speeding up individual evaluations. Rocha et al. [31] provide a solution to a single objective problem using MPI between the islands, passing from 0 to 100 solutions between the islands in a round robin approach, with a single synchronous migration per generation. They provide a master slave solution using OpenMP to distribute the island's sub-population among the cores. The load balancing is achieved by statically deciding on the population sizes based on the number of cores on each island's machine.

II. Problem Definition

The network design problem that we consider, can be formally stated as follows. Given a graph $G = (V, E)$ where $V = \{v_1, \ldots, v_n\}$ is a set of vertices and $E = \{e_1, \ldots, e_m\}$ is a set of edges, we are given:

- A weight for each edge, W_{e_i}, which defines the time it takes to traverse edge e_i;
- A matrix $\mathbf{D}_{n\times n}$ where D_{v_i,v_j} gives the passenger demand between a pair of vertices v_i and v_j.

A route R_i is defined as a simple path (i.e. no loops/repeated vertices) through the graph G. Let $G_{R_i} = (V_{R_i}, E_{R_i})$ be the subgraph induced by a route R_i. A solution is defined as a set of overlapping routes $\mathcal{R} = \{R_1, \ldots, R_r\}$ where the number of routes, r, and the minimum, m_1, and maximum, m_2, number of vertices in a route are to be specified by the user. In order for $\mathcal{R}$ to be valid the following conditions must hold:

$$\bigcup_{i=1}^{|\mathcal{R}|} V_{R_i} = V \tag{1}$$

$$m_1 \leq |V_{R_i}| \leq m_2 \; \forall R_i \in \mathcal{R} \tag{2}$$

$$G_{\mathcal{R}} = (\bigcup_{i=1}^{|\mathcal{R}|} V_{R_i}, \bigcup_{i=1}^{|\mathcal{R}|} E_{R_i}) \text{ is connected} \tag{3}$$

$$|\mathcal{R}| = r \tag{4}$$

Constraint (1) ensures that all vertices in V are in at least one route in $\mathcal{R}$. Constraint (2) specifies that each route must contain a number of vertices between m_1 and m_2 (these values are based upon considerations such as driver fatigue and the difficulty of maintaining the schedule [32]). Constraint (3) specifies that all vertices can be reached from all other vertices. If Constraint (1) is satisfied then $G_{\mathcal{R}} = (V, \bigcup_{i=1}^{|\mathcal{R}|} E_{R_i})$. Finally, Constraint (4) ensures that the solution contains the correct number of routes.

For this problem formulation, the following assumptions are also made:

1) There will always be sufficient vehicles on each route $R_i \in \mathcal{R}$ to ensure that the demand between every pair of vertices is satisfied.
2) A vehicle will travel back and forth along the same route, reversing its direction each time it reaches a terminal vertex.
3) The transfer penalty (representing the inconvenience of moving from one vehicle to another) is set at a fixed constant. In this study a value of 5 minutes is used in line with previous studies (e.g. [13], [33]).
4) Passenger choice of routes is based on shortest travel time (including transfer penalties).

In this problem we consider both the *passenger cost* and *operator cost*. In general, passengers would like to travel to their destination in the shortest possible time, but avoiding the inconvenience of making too many transfers. We define a shortest path between two vertices using the route set $\mathcal{R}$ as $\alpha_{v_i,v_j}(\mathcal{R})$. A path may include both transport links and transfer links (a transfer link facilitates the changing from one vehicle to another with the associated time penalty). This is shown in Figure 1 with the original network expanded to include transfer vertices and transfer links. The shortest path evaluation is thus completed on the transit network Figure 1 (b). The minimum journey time, $\alpha_{v_i,v_j}(\mathcal{R})$, from any given pair of vertices is thus made up of two components: in vehicle travel time and transfer penalty. We define the *passenger cost* for a route set $\mathcal{R}$ to be the mean journey time over all passengers:

$$F_1(\mathcal{R}) = \frac{\sum_{i,j=1}^{n} D_{v_i,v_j} \alpha_{v_i,v_j}(\mathcal{R})}{\sum_{i,j=1}^{n} D_{v_i,v_j}} \tag{5}$$

Operator costs depend on many factors, such as the number of vehicles needed to maintain the required level of service, the daily distance travelled by the vehicles and the costs of employing sufficient drivers. We use a simple proxy for operator costs: the sum of the costs (in time) for traversing all the routes in one direction, defined as:

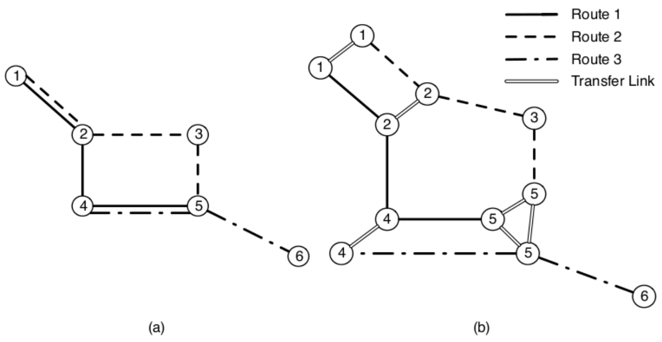

Fig. 1. (a) Route network – road network with routes overlayed (b) Transit network – network used for evaluation.

$$F_2(\mathcal{R}) = \sum_{\forall R_i \in \mathcal{R}} \sum_{\forall e_j \in R_i} W_{e_j} \tag{6}$$

III. Generation of Problem Sets

In addition to previously published problem sets, a more realistic problem set has been derived for our tests. United Kingdom road network data has been acquired from the Ordnance Survey [34]. Non-navigable or limited access roads such as alleyways and pedestrianised streets are removed from the network during a pre-processing stage. In addition, location information of bus stops was obtained for the city of Cardiff, Wales from the National Public Transport Access Node data set. Each bus stop in the required test area is mapped to the road network. Least cost paths are calculated between each pair of bus stops based upon travel time across graph edges, where the travel time is considered a function of the road type (motorway, main road, local road etc) and the distance travelled. A graph is produced $G = (V, E)$ where $V = \{v_1, \ldots, v_n\}$, the set of vertices represent the bus stops, and $E = \{e_1, \ldots, e_q\}$ the set of edges, represent the least cost paths. A simplification process is then performed to reduce the graph from a complete graph, to a more sparse graph in which edges between pairs of vertices (u, v) are removed if they are seen to pass through one or more other vertices. Vertices are identified on a shortest path and sub-paths between are extracted and stored in an external list. Selection and replacement is used to ensure the members of the external list are the shortest paths between any two vertices. Upon completion of the process the graph is seen to consist of only direct paths, that is to say paths without any intermediate vertices.

Demand metrics are simulated in the data sets using Census information and bus stop / network timetable information. In this paper we are seeking the demand for additional services. Population weighted centroids and attribute data for the 2001 UK Census population Output Areas (OA) were obtained from the UK Data Service. Attribute data as well as timetable information relating to bus operators serving Cardiff was obtained from the Association of Transport

Coordinating Officers in order to create a simulated database of bus service frequencies in Cardiff. The demand metric generated is a variation of the accessibility metric identified in [35] and given in Equation (7), where b is the number of times a bus stops at stop s, ρ is the population of all census OAs intersecting a 400m geometric buffer area around a bus stop s, and W is a weighting factor [36] based upon the distance between the centroid of a census OA and the bus stop s.

$$D = \frac{1}{b^s}((\rho^s W^s)) \tag{7}$$

The weighting value is used to implement the hypothesis that, as the distance to a bus stop increases, the appeal of that stop decreases. The distance of 400m can be seen in the literature as a representative threshold for the maximum walking distance to a bus stop [37], [38] and is a value widely used by urban planners [39], [40]. The demand value for any pair of locations is the total of each D value for a pair of nodes. Following the acquisition of travel times and demand values, the information is stored as two separate data files, each representing separate criteria. In addition, a further lookup table is created allowing the reverse identification for each stop in the matrix, allowing visualisation of the algorithm results.

IV. Serial implementation of Algorithm

The serial implementation of John et al.s algorithm [20] is based on the NSAGII Multi Objective framework [21], used with specialised crossover and repair operators proposed by Mumford [19].

The basic form of an NSGAII generation proceeds by creating an offspring population of size p, which is then combined with the parent population of size p to produce a combined population $P_c = \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_{2p}\}$.

Defining: $\mathcal{R}_{i_{\text{rank}}}$ – the non-dominated front to which $\mathcal{R}_i$ belongs [21], and $\mathcal{R}_{i_{\text{dist}}}$ – the crowding distance associated with $\mathcal{R}_i$ [21], P is then sorted such that:

$$\forall \mathcal{R}_i, \mathcal{R}_j \in P \; \mathcal{R}_{i_{\text{rank}}} \leq \mathcal{R}_{j_{\text{rank}}}$$

and

$$\mathcal{R}_{i_{\text{dist}}} \geq \mathcal{R}_{j_{\text{dist}}} \text{ for } i < j$$

The successor population P is then formed by taking the first p solutions from P_c.

V. Algorithm Parallelism

To allow real world instances of urban transport problems, such as the UTRP, to be solved, algorithms such as John et al.'s [20] need to efficiently recruit the computational power of high performance computing clusters. Within this paper we detail three models for the parallelism of such algorithms: a job farming algorithm (OMP_XED) and two variations on an 'Islands' approach. These models use OpenMP style shared memory, and a mixed mode hybrid of MPI style message passing and OpenMP.

Algorithm 1: OMP_XED NSGAII

```
p = $|P|$
—Distribute Loop Iterations Among Cores
for $i \in \{1 \ldots |P|\}$ do
    if rnd > crossover threshold then
        $P1$ =binaryTournement($P, p$)
        $P2$ =binaryTournement($P, p$)
        offspring = crossover(P1,P2)
    else
        offspring = P1
    if rnd > mutation threshold then
        mutate(offspring)
    evaluate(offspring)
    $P = P \bigcup$ offspring      (Critical Section)
—End Distribution
sort(P)
truncate(P, p)
```

Shared Memory

In this paper, OpenMP crossover and evaluation distribution (OMP_XED) is presented as a new, parallel, implementation of John et al.s [20] UTRP algorithm. OMP_XED uses OpenMP to distribute NSGAII's population evolution across the cores of a single host machine in the same style as Wang et al. [29]. Algorithm 1 is run for each generation of the Genetic Algorithm such that a single loop iteration – or job – is passed to a core at a time and as each core finishes its current job, it is assigned the next available job in the loop iteration. In this dynamic manner, the work is load balanced so if one core were to have a series of faster jobs (if the crossover was probabilistically not recruited), it would be available to take extra loop iterations from other cores. An implementation detail worth noting is that the $P = P \bigcup$offspring must be contained within a critical section of code, and cannot therefore be executed in parallel. Due to the heuristic nature of the initial population creation and its evaluation, this process is also distributed in the same style.

An alternative to distributing the population crossover and evaluation amongst the nodes cores, is to spilt the evaluation algorithm up [30]. This has the added benefit of reducing the time taken for a single operation within an optimisation; reducing single operation time allows for more fine grain check pointing within the significantly larger problem sizes.

Dijkstra's shortest path algorithm was implemented in parallel [41] but found to take longer than the serial Floyd-Warhsal [41] implementation. This is due to the parallel Dijkstra's shortest path having a complexity $O(|E|+|V|\log|V|)$ and the number of edges in a road network $|E|$ being too large.

MPI Islands

The Islands approach to the parallelism of GAs has long been talked about. It is akin to Sewall Wright's 1932

Algorithm 2: Islands

```
input $p_i$ =Population size at island
L = initial Population
————————
L = NSGAII(L,$p_i$)
Broadcast(top t $\in L$)
for $i \in islands$ do
    if $i \neq this\ island$ then
        $S =$ ReceiveIfAvailable($i$)
        $L = L \bigcup S$
sort(L)
```

description of a species being subdivided into many smaller subspecies, each largely breeding within themselves yet occasionally cross breeding with each other [42]. This idea is highly relevant to the distribution of a population within a Parallel Genetic Algorithm. The relatively infrequent occurrence of interbreeding between sub-populations significantly reduces the amount of inter process communication that has to be achieved.

In this paper, two implementations of the Islands model are presented for the UTRP, the first of which is outlined in Algorithm 2. This algorithm is executed for each generation of the GA. Each MPI Rank (or node) $n \in N$ initially creates and evaluates a local population L where $|L| = \frac{p}{|N|}$. Static population size load balancing [31] is not required because, in this case, the algorithm will be executed on a homogeneous cluster architecture. At each Rank, one generation of NSGAII is run on the local population L, which is then sorted (as detailed in IV), truncated to the original size of $\frac{p}{|N|}$, and each Rank sends the top t solutions to all other $n \in N$ via a non-blocking send operation. After the send operation, each Rank receives all currently available solutions from the other Ranks and adds them to the local population L (as suggested by Alba and Troya [22]). The NSGAII algorithm is then run on the local population L to produce $\frac{p}{|N|}$ offspring but using a choice of all available solutions, where currently $|L| = \frac{p}{|N|} + t \times x$ where x is the number of received messages in this generation.

The second Islands implementation of the UTRP uses a random selection of solutions to migrate between islands as in Araujo et al. [25].

As an extension to the above three methods and in the style of Wang et al. [29], the two modes of parallelism – OpenMP and MPI– can be combined. The MPI communication can be used to distribute the computation between the distinct cluster nodes, whilst the OpenMP style OMP_XED can also be used to distribute the local population across the nodes' cores. A hybrid solution has been developed, an Islands / OMP_XED cross, with 8 nodes this implementation can utilise 96 cores using islands of 25 solutions on 8 × 12 core nodes.

VI. Results

In this section, the results of optimisations performed with the methods described in Section V are presented.

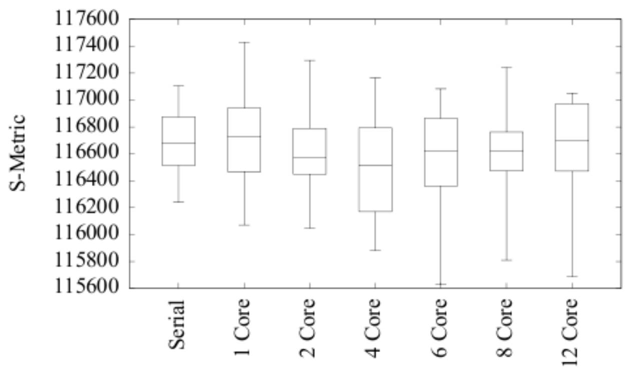

Fig. 2. Range of SMetric values from 20 runs of the OMP_XED UTRP parallelism using different numbers of cores. Mumford1 problem set.

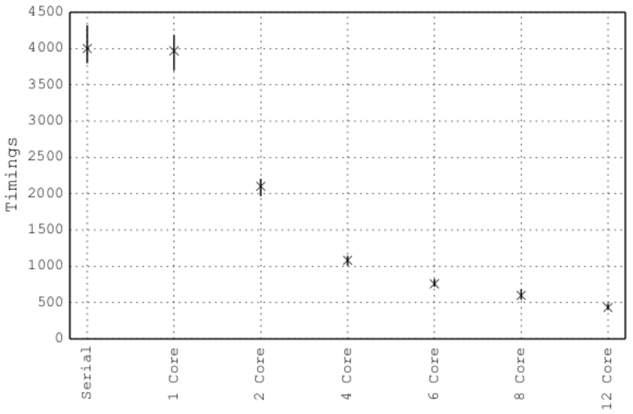

Fig. 3. Range of timings from 20 runs of the OMP_XED UTRP parallelism using different numbers of cores. Mumford1 problem set.

Initially the OpenMP panmitic solution OMP_XED is discussed which is then built on using the Islands approach. The results presented are from the problem sets Mumford1 and Mumford3 [19]. Additionally at the end of this section, results are presented from a real-world inspired problem set described in Section III.

For all results presented, a box plot is given, derived from 20 runs of the same model each with a different initial random seed. Timings given do not include the initial algorithm set-up from the configuration files, and the performance results are presented as an S-Metric [43] of the final population's Pareto set. The S-Metric reference coordinates for the Mumford1 and Mumford3 problem sets are (70, 3000) and (100, 10000) respectively.

Each of the model runs have been executed on either the HPCWales Cardiff Westmere cluster, or the HPCWales medium level HPC system at Treforest, Wales. The Cardiff cluster provides a maximum potential of 162 BX922 dual-processor nodes, each having two six-core Intel Westmere Xeon X5650 2.67 GHz CPUs and 36GB of memory. This provides a total of 1994 Intel Xeon cores (with 3 GB of memory/core) [44]. The HPCWales medium level HPC system at Treforest comprises 54 nodes which provide 648 Westmere X5650 2.67GHz cores [44]. Each set of tests where direct comparisons are made have been run on the same system.

OMP_XED

In order to evaluate the panmitic population distribution by the OpenMP model OMP_XED, results are presented from the algorithm running in serial with no OpenMP and constrained to one core, the algorithm running with OpenMP but constrained to one core, and the OMP_XED algorithm running using multiple cores. In each case where the node's full complement of cores is not utilised, exclusive access to the node is given to the job, i.e. no other jobs can be scheduled on that node in the cluster.

The results presented in Figure 2 show the difference in solution quality between each of the methods. It can be seen from the plot that there is no significant difference (one-way anova, $F_{6,133} = 0.86, p = 0.527$), which is to be expected. The timings in Figure 3 show that there is no significant difference in the serial and single core OpenMP implementations (one-way anova, $F_{1,38} = 0.63, p = 0.432$). Figure 3 also shows the speed up when using multiple cores. The speed up based on average timings is 1.9 for 2 cores, 3.69 for 4 cores, 5.28 for 6 cores, 6.69 for 8 cores, and 9.26 for 12 cores. This shows an efficiency of between 95% for 2 cores and 77% for 12 cores.

Comparisons have also been obtained for the Mumford3 problem set running serially and on 12 cores. The average time for a serial run is 157365s and the average time using OMP_XED with 12 cores is 14637, giving a speed up over 12 cores of 10.7, an efficiency of 89.9%.

Island / OMP_XED

Results are now presented from the Islands model in conjunction with the previously reported 12 core OMP_XED population distribution. This mixed mode implementation can massively increase the number of cores that can be effectively used, and reduces the evaluation time to a practical level.

The results presented in Figure 4 show the quality of the solutions for the Mumford1 problem set when run on a selection of population distributions, 200x1 indicates a population of 200 on a single island, where as 50x4 indicates a population of 50 on each of 4 islands. Figure 4 also shows a comparison between the number and type of solutions passed between the islands with _S4 indicating the top 4 solutions (see ordering policy described in Section IV) and RND2 indicating two random solutions.

When using the top t solution migration policy, the quality of the solutions can be seen to drop for the Mumford1 problem set as the size of the population in each island decreases. This could be due to the restriction on genetic diversity imposed by the smaller populations coupled with the aggressive migration policy. This is not noticeable in the Mumford3 problem set (Figure 6): in fact the quality of the solutions is generally rising (one-way anova, $F_{3,75} = 6.78, p = 0.000334$). The authors attribute this to there being a greater genetic diversity available in the larger problem set.

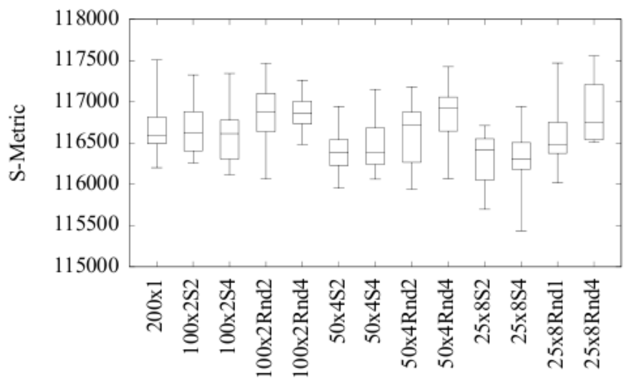

Fig. 4. Islands solution quality for Mumford1 problem set.

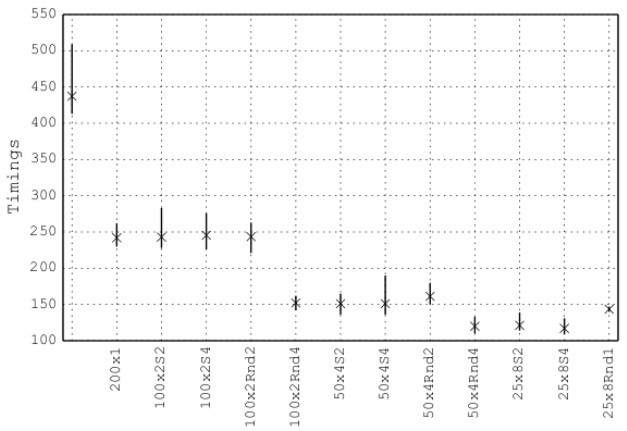

Fig. 5. Islands timing for Mumford1 problem set.

The use of the random selection migration policy on the Mumford1 problem set shows an improvement compared with the top t migration policy and in some cases, generally the higher migration rates, an improvement on the serial implementation. This enhanced solution quality, unfortunately, comes at the price of a longer run time (as seen in Figure 5) which is due to a higher crossover success rate [19], made possible by a higher genetic diversity.

The speed up for the average Mumford1 implementations using the top t solutions is 1.8 for 2 nodes giving a 90% efficiency, 2.8 for 4 nodes giving an approximate 72% efficiency, and 3.6 for 8 nodes giving a 46% efficiency. Overall, including the 12 core OMP_XED speed up, this shows a 16.5 times speed up for 2 nodes giving a 68.9% efficiency over 24 cores, 26.3 for 4 nodes giving a 54% efficiency over 48 cores, and 33.6 for 8 nodes giving a 35% efficiency over 96 cores.

The speed up for the average Mumford3 implementations using the top t solutions is 1.66 for 2 nodes giving a 83% efficiency, 2.8 for 4 nodes giving an approximate 72% efficiency, and 3.44 for 8 nodes giving a 43% efficiency. Over all, including the 12 core OMP_XED speed up, this shows a 17.8 times speed up for 2 nodes giving a 74.4% efficiency over 24 cores, 30.1 for 4 nodes giving a 62.8% efficiency over 48 cores, and 37 for 8 nodes giving a 38.5% efficiency

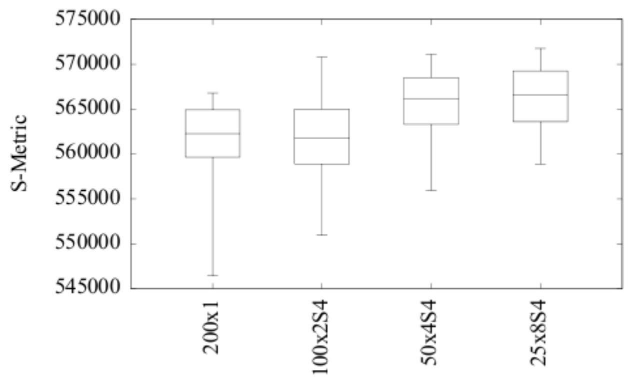

Fig. 6. Islands solution quality for Mumford3 problem set.

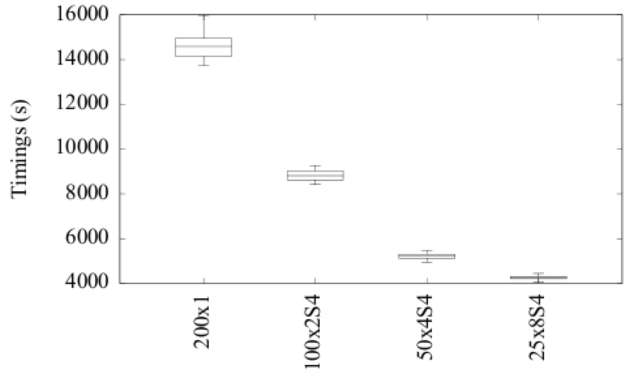

Fig. 7. Islands timing for Mumford3 problem set.

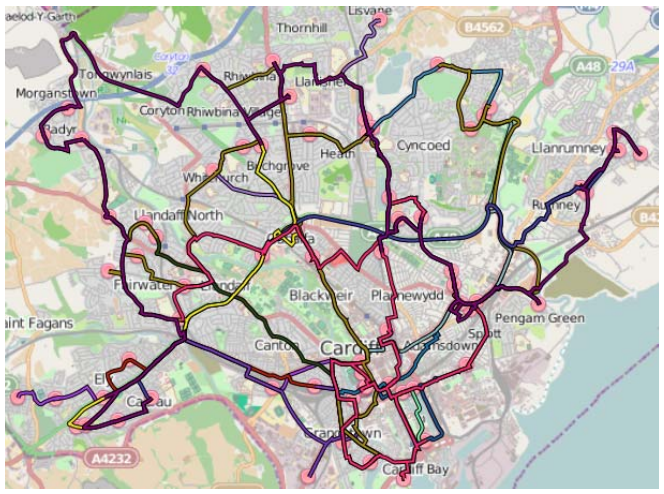

Fig. 8. Visualisation of optimised route set giving a passenger ATT= 39.4, and operator cost = 253 for the Cardiff70 problem set

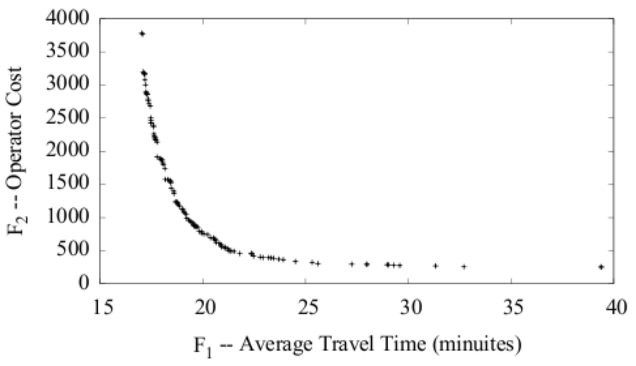

Fig. 9. Pareto set of solutions for Cardiff70 problem set.

over 96 cores.

Real World Inspired Problem Sets

A real world inspired problem set was produced based on data available from Cardiff as detailed in Section III. This initial set, Cardiff70, includes 70 bus stops (vertices) and the output can be easily visualised. This problem set took 40 seconds to run using the mixed mode Islands / OMP_XED implementation with 8 x 12 core nodes. Figure 8 shows an optimised output from the UTRP tool, it shows a set of bus routes that give an average passenger travel time and operator cost roughly in the middle of the spectrum.

The tool provides an approximate Pareto set of route sets, as shown in Figure 9, each of which can be mapped.

A Cardiff300 problem set was also optimised, taking approximately 72 hours to execute 12 generations. This extrapolates to approximately 50 days for 200 generations. Executing this algorithm in series would take approximately 5 years. On a practical note, the use of check pointing allows evolved populations, at the end of the standard HPC 72hour job execution window, to be re-entered to the GA and re-submitted to the HPC system for further generations.

VII. Conclusions

The results presented in this paper show that the efficiency of the mixed mode Islands / OMP_XED using a random migration policy improves for more complex problem sets and reaches 38.5% when using 96 cores on a Mumford3 problem set. It additionally improves the performance of the serial GA implementation for certain problem sets. These results concur with Araujo et al. [25] showing that a random migration policy between islands is better than a top t policy. Whilst the quality of the solutions is of the utmost importance, the run-times need to be considered from a practical point of view.

The paper demonstrates that a tool to optimise the UTRP can be implemented using GAs and HPC and will run, town and city sized, real world inspired problem sets in a realistic time frame.

References

[1] "European Union," 2013. [Online]. Available: "http://ec.europa.eu/transport/themes/sustainable/index_en.htm "

[2] "Smart Cities: Outcomes of the 2nd high level group meeting, 14 October 2013, Brussels."

[3] S. A. Bagloee and A. A. Ceder, "Transit-network design methodology for actual-size road networks," *Transportation Research Part B*, vol. 45, no. 10, pp. 1787–1804, Dec. 2011.

[4] G. Nielsen, "Hitrans best practice guide 2: Public transport planning the networks," European Union Interreg IIIB (North Sea Region), Tech. Rep., 2005.

[5] F. Zhao and A. Gan, "Optimization of transit network to minimize transfers," Lehman Center for Transportation Research, Florida International University, 605 Suwannee Street, MS 30, Tallahassee FL 32399-0450, Tech. Rep. BD-015-02, December 2003.

[6] W. Lampkin and P. D. Saalmans, "The design of routes, service frequencies, and schedules for a municipal bus undertaking: A case study," *Journal of the Operational Research Society*, vol. 18, pp. 375–397, Dec. 1967.

[7] L. Silman, Z. Barzily, and U. Passy, "Planning the route system for urban buses," *Computers & Operations Research*, vol. 1, no. 2, pp. 201 – 211, 1974.

[8] C. E. Mandl, *Applied network optimization*. Academic Press, 1979.

[9] C. E. Mandl, "Evaluation and optimization of urban public transportation networks," *European Journal of Operational Research*, vol. 5, no. 6, pp. 396 – 404, 1980.

[10] A. Ceder and N. H. M. Wilson, "Bus network design," *Transportation Research Part B: Methodological*, vol. 20, no. 4, pp. 331–344, August 1986.

[11] M. H. Baaj and H. S. Mahmassani, "Hybrid route generation heuristic algorithm for the design of transit networks," *Transportation Research Part C: Emerging Technologies*, vol. 3, no. 1, pp. 31 – 50, 1995.

[12] J. Agrawal and T. V. Mathew, "Transit route network design using parallel genetic algorithm," *Journal of Computing in Civil Engineering*, vol. 18, no. 3, pp. 248–256, 2004.

[13] P. Chakroborty and T. Wivedi, "Optimal route network design for transit systems using genetic algorithms," *Engineering Optimization*, vol. 34, no. 1, pp. 83–100, 2002.

[14] S. B. Pattnaik, S. Mohan, and V. M. Tom, "Urban bus transit route network design using genetic algorithm," *Journal of Transportation Engineering*, vol. 124, pp. 368–375, July/August 1998.

[15] V. M. Tom and S. Mohan, "Transit route network design using frequency coded genetic algorithm," *Journal of Transportation Engineering*, vol. 129, pp. 186–195, March/April 2003.

[16] W. Fan and R. B. Machemehl, "A tabu search based heuristic method for the transit route network design problem," *Computer-aided Systems in Public Transport*, pp. 387–408, 2008.

[17] W. Fan and R. B. Machemehl, "Using a simulated annealing algorithm to solve the transit route network design problem," *Journal of Transportation Engineering*, vol. 132, no. 2, pp. 122–132, 2006.

[18] T. L. Magnanti and R. T. Wong, "Network design and transportation planning: Models and algorithms," *Transportation Science*, vol. 18, no. 1, pp. 1–55, 1984.

[19] C. Mumford, "New heuristic and evolutionary operators for the multi-objective urban transit routing problem," *IEEE Evolutionary Computation 2013(CEC)*, pp. 939–946, 2013. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6557668

[20] C. Mumford, M. P. John and R. Lewis, "An improved multi-objective algorithm for the urban transit routing problem," in *Evolutionary Computation in Combinatorial Optimization*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2014.

[21] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[22] E. Alba and J. M. Troya, "Analyzing synchronous and asynchronous parallel distributed genetic algorithms," *Future Generation Computer Systems*, vol. 17, no. 4, pp. 451–465, Jan. 2001. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0167739X99001296

[23] M. Affenzeller and S. Wagner, "SASEGASA : A New Generic Parallel Evolutionary," *Journal of Heuristics*, vol. 10(3), no. 1975, pp. 243–267, 2004.

[24] C. Gagné, M. Parizeau, and M. Dubreuil, "The master-slave architecture for evolutionary computations revisited," *Genetic and Evolutionary Computation ...*, pp. 1578–1579, 2003. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-45110-2_33

[25] L. Araujo, J. J. Merelo, A. Mora, and C. Cotta, "Genotypic differences and migration policies in an island model," *Proceedings of the 11th Annual conference on Genetic and evolutionary computation - GECCO '09*, p. 1331, 2009. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1569901.1570080

[26] E. Noda and A. Coelho, "Devising adaptive migration policies for cooperative distributed genetic algorithms," *Systems, Man and ...*, no. ii, 2002. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1175628

[27] F. Lardeux and A. Goëffon, "A dynamic island-based genetic algorithms framework," *Simulated Evolution and Learning*, 2010. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-642-17298-4_16

[28] E. de Jong, R. Watson, and J. Pollack, "Reducing Bloat and Promoting Diversity using Multi-Objective Methods," in *Genetic and Evolutionary Computation Conference GECCO '01*, pp. 11 – 18, 2001. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.28.4549

[29] Z.-r. Wang, T. Ju, D.-w. Cui, and X.-h. Hei, "A study of hybrid parallel genetic algorithm model," *2011 Seventh International Conference on Natural Computation*, pp. 1038–1042, Jul. 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6022186

[30] K. Narayanan, A. Mora, N. Allsopp, and T. E. Sayed, "A hybrid, massively parallel implementation of a genetic algorithm for optimization of the impact performance of a metal/polymer composite plate," *International Journal of High Performance Computing Applications*, vol. 27, no. 2, pp. 217–227, Jul. 2012. [Online]. Available: http://hpc.sagepub.com/cgi/doi/10.1177/1094342012451474

[31] I. Rocha, E. Parente, and A.M.C. Melo, "A hybrid shared/distributed memory parallel genetic algorithm for optimization of laminate composites," *Composite Structures*, vol. 107, pp. 288–297, Jan. 2014. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0263822313003875

[32] F. Zhao and A. Gan, "Optimization of transit network to minimize transfers," Florida Department of Transportation, Tech Rep., 2003.

[33] M. C. Shih and H. S. Mahmassani, "A design methodology for bus transit networks with coordinated operations," Tech. Rep. SWUTC/94/60016-1, 1994.

[34] Ordanance Survey, 2014, [Online]. Available:"http://www.ordnancesurvey.co.uk/business-and-government/products/itn-layer.html"

[35] M. Langford, G. Higgs, and R. Fry, "Using floating catchment analysis (fca) techniques to examine intra-urban variations in accessibility to public transport opportunities: the example of cardiff, wales," *Journal of Transport Geography: Special Section on Accessibility and Socio-Economic Activities: Methodological and Empirical Aspects*, vol. 25, no. 0, pp. 1 – 14, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0966692312001652

[36] S. Butterworth, "On the theory of filter amplifiers," *Experimental Wireless and the Wireless Engineer*, vol. 7, pp. 536–541, 1930.

[37] G. Currie, "Gap analysis of public transport needs: measuring spatial distribution of public transport needs and identifying gaps in the quality of public transport provision," *Transportation Research Record*, vol. 1895, pp. 137–146, 2004. [Online]. Available: http://arrow.monash.edu.au/hdl/1959.1/111108

[38] A. T. Murray, R. Davis, R. J. Stimson, and L. Ferreira, "Public transportation access," *Transportation Research Part D: Transport and Environment*, vol. 3, no. 5, pp. 319 – 328, 1998. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1361920998000108

[39] D. B. Hess, "Walking to the bus: perceived versus actual walking distance to bus stops for older adults," *Transportation*, vol. 39, no. 2, pp. 247–266, 2012. [Online]. Available: http://dx.doi.org/10.1007/s11116-011-9341-1

[40] M. Southworth and E. Ben-Joseph, *Streets and the Shaping of Towns and Cities*. Island Press, 2003.

[41] "Argonne national laboratory," 2014. [Online]. Available: "http://www.mcs.anl.gov/ itf/dbpp/text/node35.html"

[42] S. Wright, "The roles of mutation, inbreeding, crossbreeding and selection in evolution," *Proceedings of the sixth international congress on ...*, vol. 1, pp. 356–366, 1932. [Online]. Available: http://www.esp.org/books/6th-congress/facsimile/contents/6th-cong-p356-wright.pdf

[43] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithmsa comparative case study," *Parallel problem solving from naturePPSN V*, 1998. [Online]. Available: http://link.springer.com/chapter/10.1007/BFb0056872

[44] "HPC Wales," 2014. [Online]. Available: www.hpcwales.co.uk