An Improved Multi-objective Algorithm for the Urban Transit Routing Problem

Matthew P. John[1,2], Christine L. Mumford[1], and Rhyd Lewis[2]

[1] Cardiff School of Computer Science & Informatics, UK
[2] Cardiff School of Mathematics, UK
JohnMP@cardiff.ac.uk

Abstract. The determination of efficient routes and schedules in public transport systems is complex due to the vast search space and multiple constraints involved. In this paper we focus on the Urban Transit Routing Problem concerned with the physical network design of public transport systems. Historically, route planners have used their local knowledge coupled with simple guidelines to produce network designs. Several major studies have identified the need for automated tools to aid in the design and evaluation of public transport networks. We propose a new construction heuristic used to seed a multi-objective evolutionary algorithm. Several problem specific mutation operators are then combined with an NSGAII framework leading to improvements upon previously published results.

1 Introduction

The Urban Transit Network Design Problem (UTNDP) involves the determination of an efficient set of routes and schedules for public transportation systems such as bus, rail and tram networks. Ceder and Wilson [1] identified five main stages for bus service planning: network design, frequency setting, timetable development, bus scheduling and driver scheduling. Given that each stage of the UTNDP is NP-hard [2], it is usually considered impractical to solve all the stages simultaneously.

In this paper we focus 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). Building upon recently published work by Mumford [3], we present a much improved multi-objective approach based upon NSGAII [4]. Our approach considers the trade offs between passenger and operator costs by producing approximate Pareto optimal sets for consideration by a human decision maker. Furthermore, we provide a discussion of the many specialised heuristics and operators we have tested during the development of our approach.

Historically, route planners have used a combination of local knowledge and simple guidelines to produce route sets. Several major studies (see [5, 6]) have identified the need for automated computer based tools for the design and evaluation of public transport networks. Automation is, however, highly complex

C. Blum and G. Ochoa (Eds.): EvoCOP 2014, LNCS 8600, pp. 49–60, 2014.

and computationally expensive due to the large search space and multiple constraints involved in public transportation planning. The increase in congestion, pollution, greenhouse gas emissions and dwindling oil resources have placed emphasis on the use of public transport in recent years in an attempt to reduce the reliance of the private car. Achieving an increase in public transportation usage is clearly desirable but is also an extremely complex issue. However frequent and reliable cost-effective services are clearly key attributes.

Bagloee and Ceder [7] have recently pointed out that many public transit networks have not been reappraised from anywhere between 20 to 50 years. Land use patterns have changed considerably in this time period with the migration away from town centres into surrounding suburban areas; however public transport has been relatively slow to respond. It is our view that the development of automated tools to aid public transport networks is timely.

Prior to 1979, the few papers published on the UTNDP considered only highly specific problem instances [8, 9]. In 1979, Christoph Mandl [10, 11] approached the problem in a rather more generic form. He concentrated on the network design phase, and developed a two-stage solution. First a set of feasible routes is generated, second, 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. [1, 12]. 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) [13–16]. Other metaheuristic methods such as tabu search and simulated annealing can also be seen in [17, 18]. Nevertheless, comparative work has been limited to Mandl's 15 vertex instance.

One of the first approaches using a GA was proposed by Pattnaik et al. [15] utilising a route set generation procedure guided by a demand matrix, designer knowledge, and route constraints to produce a set of candidate routes. The role of the GA is simply to select a number of routes from the candidate set, where each route is given a unique identifier encoded in binary. This approach is similar to those proposed by Tom and Mohan [16] and Fan and Machemehl [19].

Chakroborty and Dwivedi [14] generated an initial set of candidate route sets each with a fixed number of routes. A GA was then applied using two crossover operators: 1) Inter-string crossover which exchanges routes from the parent route sets, and 2) Intra-string crossover which exchanges parts of a route in a parent if two routes share a common vertex. Mutation is then applied by randomly selecting a vertex and changing it to any of its acceptable adjacent vertices.

Similar to Chakroborty and Dwivedi, Fan et al. [20] generated an initial population of feasible route sets. Their GA used a mutation operator where a vertex can be either added to the end of a route or removed from the start. However, their GA lacked a crossover operator.

Bagloee and Ceder [7] tackled real sized road networks using a combination of heuristics, a GA and an ant-system. They determined the location of stops based upon the distance to high concentrations of travel demand and then used a system inspired by Newtonian gravity theory to produce a set of candidate

routes. A GA was then used to search through the candidate routes to find a good solution – the frequencies of the routes was computed simultaneously.

Recently Mumford [3] presented an approach using a constructive heuristic to generate an initial population. This was then evolved using a multi-objective evolutionary algorithm (MOEA). Mumford's crossover operator alternates the selection of routes between two parents, favoring routes containing vertices that are not currently present in the partial solution, while ensuring route connectivity. Two mutation operators are also proposed that add or delete a bounded random number of vertices from a route set. Four benchmark instances based on real-world bus route networks were also created and made publicly available [3].

2 Problem Description

The network design problem 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 should contain between m_1 and m_2 vertices (these values are based on considerations such as driver fatigue and the difficulty of maintaining the schedule [6]). Constraint (3) specifies that a path exists between all pairs of vertices in $G_{\mathcal{R}}$. 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. [14, 21]).
4. Passenger choice of routes is based on shortest travel time (which includes 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 Fig. 1 with the original network expanded to include transfer vertices and transfer links. The shortest path evaluation is thus completed on the transit network Fig. 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 as given by Mumford [3]:

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

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, as given by Mumford [3]:

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

3 Methodology

In this paper we propose an approach that seeds a MOEA with a high quality initial population formed using a powerful heuristic construction procedure. NSGAII [4] is then used with the crossover and repair operators proposed by Mumford [3] along with several new mutation operators.

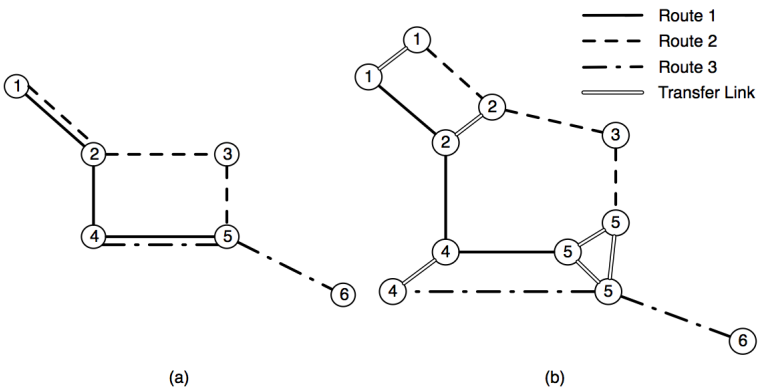

Fig. 1. (a) Route network – road network with routes overlayed (b) Transit network – network used for evaluation

3.1 NSGAII

NSGAII is an elitist non-dominated sorting MOEA widely used to solve multi-objective optimization problems. It has been shown to find a better spread of solutions and convergence nearer to the true Pareto-optimal front compared with other Pareto based methods [4] .

The basic form of an NSGAII generation proceeds by creating an offspring population of size N. This is then combined with the parent population of size N to produce a population, $P = \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_{2N}\}$. Let us now define two attributes of a route set $\mathcal{R}_i$: 1) $\mathcal{R}_{i_{\text{rank}}}$ the non-dominated front that $\mathcal{R}_i$ belongs to, and 2) $\mathcal{R}_{i_{\text{dist}}}$ the crowding distance associated with $\mathcal{R}_i$ as defined by Deb et al. [4]. 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 $\mathcal{R}_{i_{\text{dist}}} \geq \mathcal{R}_{j_{\text{dist}}}$ for $i < j$. The successor population is then formed by taking the first N solutions in P.

Similar to [4], in our case a new population is generated using binary tournament selection with a crossover probability of 0.9. The probability of mutating each route in an offspring is set as $\frac{1}{r}$.

3.2 Heuristic Construction

As mentioned earlier, heuristic construction was widely used for the network design stage before the use of metaheuristic algorithms became more widespread in the literature. Our construction heuristic creates solutions by incorporating knowledge that exploits the underlying structure of the problem.

The majority of previous approaches for tackling network design (see [18, 19, 15]) generate a candidate pool of routes using Dijkstra's shortest path algorithm and Yen's [22] k-shortest path algorithm to enumerate all possible routes. An optimization algorithm is then applied to find the best combination of routes from the candidate pool, although there is still no guarantee that the optimal

configuration of routes will be selected. As such we have chosen to combine our approach with that used by Shih and Mahmassani [21] with modifications to produce route sets that balance the cost to the operator and passengers whilst ensuring the constraints of the problem are adhered to. Shih and Mahmassani [21] generate an initial route set using a heuristic procedure that continually adds routes until user-defined levels on directness and coverage are reached. This contrasts with our approach where the number of routes is fixed. Furthermore, our method ensures that all vertices are present within a route set, allowing a passenger to reach any vertex in the network using transfers if necessary.

Our approach generates a set of weighted graphs using a weighted sum of normalized travel time and (1 − normalized demand) with weights specified in advance by the user. A route set, $\mathcal{R}$, is then generated from a given weighted graph. A spanning subgraph is first created using an iterative procedure with the objective of minimizing the sum of the weighted edge costs. In the first iteration of this procedure the pair of vertices (seed pair) with the lowest weighted edge cost are selected. In subsequent iterations a seed pair contains one vertex already contained in the subgraph, ensuring that routes remain connected. A seed pair is then expanded to form a route by adding adjacent vertices.

During this expansion process vertices that are adjacent to the first or last vertex in the route are sorted against their weighted edge cost. The minimum cost unused vertex is then added to the route. If there are no unused vertices the minimum cost adjacent vertex is instead added to the route providing of course that the vertex is not present in the current route under construction. If multiple vertices are in the set of potential vertices with equal weight a random vertex from this set is selected. Vertices are continually inserted until a vertex can no longer be inserted that would not cause a constraint to be violated.

Providing that $|\mathcal{R}| < r$ the next stage of the heuristic procedure is applied using an approach utilized by Shih and Mahmassani [21]. The vertex pairs (v_i, v_j) that are yet to be satisfied directly (i.e. it is not possible to travel between the vertices without having to make a transfer) are extracted from the network and sorted in non-ascending order based upon the demand D_{v_i,v_j}. The unsatisfied vertex pairs are taken in order and Yen's k-shortest path algorithm is applied to determine if a valid route, originating at v_i and terminating at v_j, can be constructed that obeys all of the constraints. In our case a maximum of ten 'shortest' paths are explored for each vertex pair. An alternative to satisfying the vertex pairs that are not satisfied directly is to minimize the travel time for passengers on high demand vertex pairs. In this instance all the vertex pairs are extracted and sorted based upon non-ascending demand. If a valid route is found, the cost of the route is calculated and compared with the $\alpha_{v_i,v_j}(\mathcal{R})$. If the cost is less than $\alpha_{v_i,v_j}(\mathcal{R})$ the route is inserted into $\mathcal{R}$. This process is applied iteratively until $|\mathcal{R}| = r$.

3.3 Genetic Operators

Crossover: We use the crossover operator proposed by Mumford [3], which ensures that the problem constraints are obeyed. Given two parents, the crossover

operator constructs an offspring, $\mathcal{R}'$, by alternatively selecting a route from each parent that maximises the proportion of unseen vertices, until $|\mathcal{R}'| = r$. The set of unseen vertices is defined as $V_{\text{unseen}} = V - V_{R'}$. Consider a route $R_i = \langle 1, 7, 8, 9, 12, 14 \rangle$ that is contained in one of the parents and is being considered for insertion into $\mathcal{R}'$. If $\mathcal{R}' = \{R_1\}$ where $R_1 = \langle 12, 15, 0, 5, 3 \rangle$ then $V_{\text{unseen}} = R_i - R_1 = \{1, 7, 8, 9, 14\}$. Therefore the proportion of unseen vertices is $\frac{|V_{\text{unseen}}|}{|R_i|} = \frac{5}{6}$ in this case.

After crossover has been applied it is possible that the offspring will not contain all the vertices in V. In these cases the repair procedure used by Mumford is applied that attempts to add the missing vertices to either the back or front of the routes.

Mutation: In our approach eight mutation operators are used. Some of these apply heuristics to mutate the route set in a way that encourages an improvement in quality. Mutation must be carefully controlled to prevent violation of the problem constraints. The names of these mutation operators are *add-nodes*, *del-nodes*, *exchange*, *merge*, *replace*, *remove-overlapping*, *two-opt* and *invert-exchange*.

Add-nodes and *del-nodes* were both proposed in [3]. At the start of *add-nodes* or *del-nodes* an integer I is generated uniform randomly in the range $[1, r \times \frac{m_2}{2}]$, giving the number of vertices to be added or removed from $\mathcal{R}$. A route $R_i \in \mathcal{R}$ is then selected at random and, in the case of *add-nodes*, vertices are added to the end of the route until the addition of a vertex would cause Constraint (2) to be violated or, result in R_i no longer being a simple path. Following a similar approach, vertices are then added to the front of the route if possible. This process is repeated for each $R_i \in \mathcal{R}$ until I vertices have been added to $\mathcal{R}$ or all routes have been exhausted. The case is the same for *del-nodes* with I vertices being removed from $\mathcal{R}$ whilst ensuring feasibility.

The *Exchange* operator, as proposed by Mandl [10], selects a route at random. The route set is then searched to determine if there exists a route with a common vertex to the selected route. The problem constraints are also checked to determine if $\mathcal{R}$ will be valid after the mutation has been applied. If valid, the two routes are split at the first common vertex, creating four route segments. The two original routes are then replaced by exchanging the segments to create two new routes. The exchange of route parts attempts to reduce the number of transfers passengers must make.

Similar to *exchange*, the *merge* operator, selects a random route and searches the remaining routes to find a route that shares a common terminal vertex. The two routes are then merged creating one continuous route, disregarding one of the common terminal vertices – providing that Constraint (2) is not violated and the merged route is a simple path. If successful, a route generation procedure (*route-gen*) is then used to generate a new route for insertion. The route generation procedure is as follows: vertex pairs that are not yet satisfied directly are extracted from the route set and sorted via non-ascending demand (i.e. given every pair of vertices in the network we are only interested in those that cannot be reached without transfers, given the current configuration of $\mathcal{R}$).

Yen's k-shortest path algorithm is then used to generate a bounded number of paths, in our case ten, between the two vertices and a random path selected for insertion. If the number of vertex pairs yet to be satisfied directly is zero, a path is generated between the vertices with the highest demand in the network.

The *Replace* operator removes a route $R_i \in \mathcal{R}$ that satisfies the least demand directly compared with all other routes in $\mathcal{R}$. A replacement route is then generated using the *route-gen* procedure described above. The purpose of the *replace* mutation is to sacrifice routes that serve a relatively low demand in place of high demand routes. *Replace* can cause a route set to become invalid if the removed route acted as a transfer hub for routes, i.e. the route set was only connected when the removed route was present. If this situation occurs the repair procedure used during crossover is applied and, if successful, the mutated solution is returned. Otherwise the mutation is abandoned.

The *Remove-overlapping* operator replaces a route that is a subset of another route. If an overlapping route is discovered, it is removed and the route generation procedure described above is used to produce a replacement route. Replacing the route provides the operator with the ability to remove duplicate services and use these resources to serve other passenger demand.

Two-opt, proposed in 1958 by Croes [23] for use with the traveling salesman problem selects two vertices at random in a route and inverts the vertices between them. Its original purpose was to remove crossover points in a route, however this is not allowed to occur in this context. In our case it reorders the vertices in a route attempting to reduce the travel time between vertex pairs, abandoning infeasible attempts.

Invert-exchange selects two routes at random and generates two random index locations. The vertices between the two random index locations are then inverted and exchanged between the two routes. For example, given two routes $R_1 = \langle 3, 5, 8, 10, 12, 15\rangle$ and $R_2 = \langle 1, 6, 9, 8, 11, 7\rangle$ with the selected indices of *3* and *5*. We invert everything in R_1 between the indices giving $\langle 12, 10, 8\rangle$ then replace the vertices in R_2 between the indices with the inverted section from R_1. In this case the resultant two routes would be $R_3 = \langle 3, 5, 11, 8, 9, 15\rangle$ and $R_4 = \langle 1, 6, 12, 10, 8, 7\rangle$. *Invert-exchange* attempts to decrease the travel time between vertices and prevent passengers having to make a transfer. Similar to *two-opt* there is a high possibility that the majority of routes created using this approach will be infeasible. As such, two routes are continually chosen at random until the routes have been exhausted or a feasible solution has been found.

4 Results

In this section we show how the algorithm of Mumford [3] (Algorithm A), based on the SEAMO2 framework, can be improved, by seeding the MOEA with our heuristically generated solutions (Algorithm B). We then look at the effects of adding our mutation operators (Algorithm C) and finally look at the effects of using the NSGAII framework (Algorithm D) as opposed to SEAMO2. All experiments use an initial population of size 200 and are run for 200 generations, the

Table 1. Problem instances used for comparison with the lower bound (LB) for each objective

Instance	Vertices	Edges	r	m_1	m_2	Vertices in Typical Transit Net.	LB_{F_1}	LB_{F_2}
Mandl	15	20	6	2	8	$6 \times \frac{(2+8)}{2} = 30$	10.0058	63
Mumford0	30	90	12	2	15	102	13.0121	94
Mumford1	70	210	15	10	30	300	19.2695	294
Mumford2	110	385	56	10	22	896	22.1689	749
Mumford3	127	425	60	12	25	1110	24.7453	928

same as in [3] so that valid comparisons can be made. Running times range from a couple of seconds for Mandl's instance up to two days for Mumford3. Twenty replicate runs are used and the results have been combined into approximate Pareto sets for comparison. Problem instances generated by Mumford [3] along with Mandl's [10] benchmark are used. Table 1 summarises the details of each instance along with the parameters used and lower bounds as given in [3].

Using our heuristic construction procedure, a subset of unique solutions are randomly selected for insertion into an initial population. Randomly generated solutions are then used to top-up the initial population if there are too few heuristic solutions. These random solutions are created using the same approach as Mumford [3] to seed her MOEA. In summary this approach constructs a route set one route at a time. A route length is randomly generated between m_1 and m_2 and a random vertex, $v \in V$, is selected as the seed. A randomly selected adjacent vertex is then added to the back of the route, and this process is repeated. Once the vertices that can be added to the back are exhausted the process is repeated from the front of the route until the desired length is achieved.

We firstly augment Mumford's SEAMO2 algorithm with our heuristic method for generating the initial population (Algorithm B). Table 2 presents the best

Table 2. Best objective values extracted from twenty replicate runs using heuristic seeding for the initial population (Algorithm B). Mumford's [3] results are given in brackets.

		Mandl	Mumford0	Mumford1	Mumford2	Mumford3
Best for passenger	F_1	**10.25**(10.33)	**15.40**(16.05)	**23.91**(24.79)	**27.02**(28.65)	**29.50**(31.44)
	F_2	212(224)	745(759)	1861(2038)	5461(5632)	6320(6665)
Best for operator	F_1	13.48(15.13)	32.78(32.40)	39.98(34.69)	32.33(36.54)	36.12(36.92)
	F_2	**63**(63)	**95**(111)	**462**(568)	**1875**(2244)	**2301**(2830)

Table 3. S-metric comparison over the five benchmark instances for our proposed modifications

Instance	Alg. A	Alg. B	Alg. C	Alg. D
Mandl	2620.19	2620.21	2626.94	2631.16
Mumford0	14951.24	15031.81	15304.90	15451.50
Mumford1	111947.82	114614.99	114972.74	117866.22
Mumford2	306261.94	322753.72	322618.47	337987.85
Mumford3	507983.61	538371.55	539296.98	562793.74

solutions from the passenger and operator perspective compared to the findings of [3]. We see that our heuristic is clearly beneficial, producing an improvement over all the instances. Where an improvement is made in the objective value, either from the passenger or operator perspective, we can see that an improvement is also made to the other objective on the majority of instances. An improvement from the operator perspective on Mandl's instance is not possible as 63 is the lower bound for this instance [3]. However, a decrease is observed in the

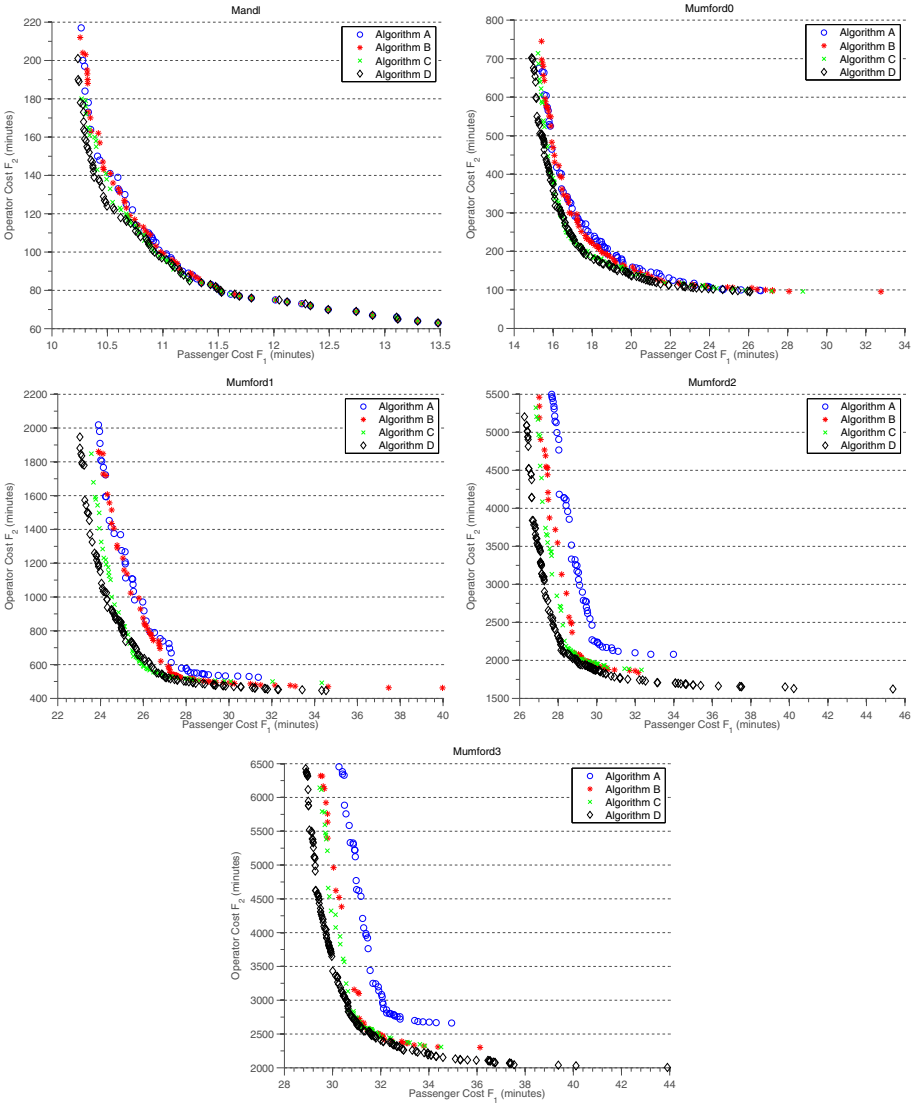

Fig. 2. Combined Pareto fronts extracted from twenty runs for each of the five benchmark instances using the four algorithms

passenger objective associated with the operator objective of 63 compared to that found by Mumford.

Taking the best algorithm (Algorithm B) from the previous experiment we now examine the effect of augmenting the algorithm with our proposed mutation operators. Comparing S-metric values for Algorithms B and C (Table 3) we can see that an improvement is achieved with all instances apart from Mumford2. If the Pareto set is plotted for Mumford2 (Fig. 2) it can be seen that there is an improvement in the passenger objective for the majority of solutions. However, we struggle to make improvements in the extremes of the operator objective.

Given the popularity of NSGAII and its stated ability to produce a Pareto set closer to the true Pareto-optimal front compared with other Pareto based methods [4], it was used instead of SEAMO2 in our third set of experiments. Here, we used our mutation operators and heuristic seeding, together with Mumford's crossover operator. As mentioned earlier a probability of crossover and mutation of 0.9 and $\frac{1}{r}$ respectively are used. A comparison of S-metric values, Table 3, shows that NSGAII gives an improvement over all the problem instances. This is displayed graphically in Fig. 2. These improvements can be attributed to the following: 1) A higher selection pressure compared with SEAMO2, and 2) Increased rate of mutation leading to a greater exploration of the search space.

5 Conclusion

This paper has presented a new construction heuristic for creating initial solutions to the transit network design problem. For this problem it has been shown that the use of heuristic solutions in a MOEA's initial population can produce an improvement in the resultant Pareto set – compared with the use of solely randomly generated solutions. Several mutation operators have also been proposed and combined with our construction heuristic to produce an improvement over previously published results. Finally, we have also shown that further improvements can be found if NSGAII is used in place of SEAMO2. We are currently investigating the introduction of frequency setting to the MOEA framework using more realistic evaluation, allowing our model to more accurately reflect passenger choice. In addition we are collecting real-world data to enable the production of more benchmark instances.

References

1. Ceder, A., Wilson, N.H.M.: Bus network design. Transportation Research Part B 20(4), 331–344 (1986)
2. Magnanti, T.L., Wong, R.T.: Network design and transportation planning: Models and algorithms. Transportation Science 18(1), 1–55 (1984)
3. Mumford, C.L.: New heuristic and evolutionary operators for the multi-objective urban transit routing problem. In: 2013 IEEE Congress on Evolutionary Computation (CEC), pp. 939–946 (2013)

4. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
5. Nielsen, G., Nelson, J.D., Mulley, C., Tegner, G., Lind, G., Lange, T.: Public transport–planning the networks. HiTrans Best Practice Guide (2005)
6. Zhao, F., Gan, A.: Optimization of transit network to minimize transfers (2003)
7. Bagloee, S.A., Ceder, A.A.: Transit-network design methodology for actual-size road networks. Transportation Research Part B 45(10), 1787–1804 (2011)
8. Lampkin, W., Saalmans, P.D.: The design of routes, service frequencies, and schedules for a municipal bus undertaking: A case study. In: OR, pp. 375–397 (1967)
9. Silman, L.A., Barzily, Z., Passy, U.: Planning the route system for urban buses. Computers & Operations Research 1(2), 201–211 (1974)
10. Mandl, C.E.: Applied network optimization. Academic Pr. (1979)
11. Mandl, C.E.: Evaluation and optimization of urban public transportation networks. European Journal of Operational Research 5(6), 396–404 (1980)
12. Baaj, M.H., Mahmassani, H.S.: Hybrid route generation heuristic algorithm for the design of transit networks. Transportation Research Part C 3(1), 31–50 (1995)
13. Agrawal, J., Mathew, T.V.: Transit route network design using parallel genetic algorithm. Journal of Computing in Civil Engineering 18(3), 248–256 (2004)
14. Chakroborty, P., Dwivedi, T.: Optimal route network design for transit systems using genetic algorithms. Engineering Optimization 34(1), 83–100 (2002)
15. Pattnaik, S.B., Mohan, S., Tom, V.M.: Urban bus transit route network design using genetic algorithm. Journal of Transportation Engineering 124(4), 368–375 (1998)
16. Tom, V.M., Mohan, S.: Transit route network design using frequency coded genetic algorithm. Journal of Transportation Engineering 129(2), 186–195 (2003)
17. Fan, W., Machemehl, R.B.: A tabu search based heuristic method for the transit route network design problem. Computer-aided Systems in Public Transport, 387–408 (2008)
18. Fan, W., Machemehl, R.B.: Using a simulated annealing algorithm to solve the transit route network design problem. Journal of Transportation Engineering 132(2), 122–132 (2006)
19. Fan, W., Machemehl, R.B.: Optimal transit route network design problem with variable transit demand: genetic algorithm approach. Journal of Transportation Engineering 132(1), 40–51 (2006)
20. Fan, L., Mumford, C.L., Evans, D.: A simple multi-objective optimization algorithm for the urban transit routing problem. In: IEEE Congress on Evolutionary Computation, CEC 2009, pp. 1–7 (2009)
21. Shih, M.C., Mahmassani, H.S.: A design methodology for bus transit networks with coordinated operations. Technical Report SWUTC/94/60016-1 (1994)
22. Yen, J.Y.: Finding the k shortest loopless paths in a network. Management Science 17(11), 712–716 (1971)
23. Croes, G.A.: A method for solving traveling-salesman problems. Operations Research 6(6), 791–812 (1958)