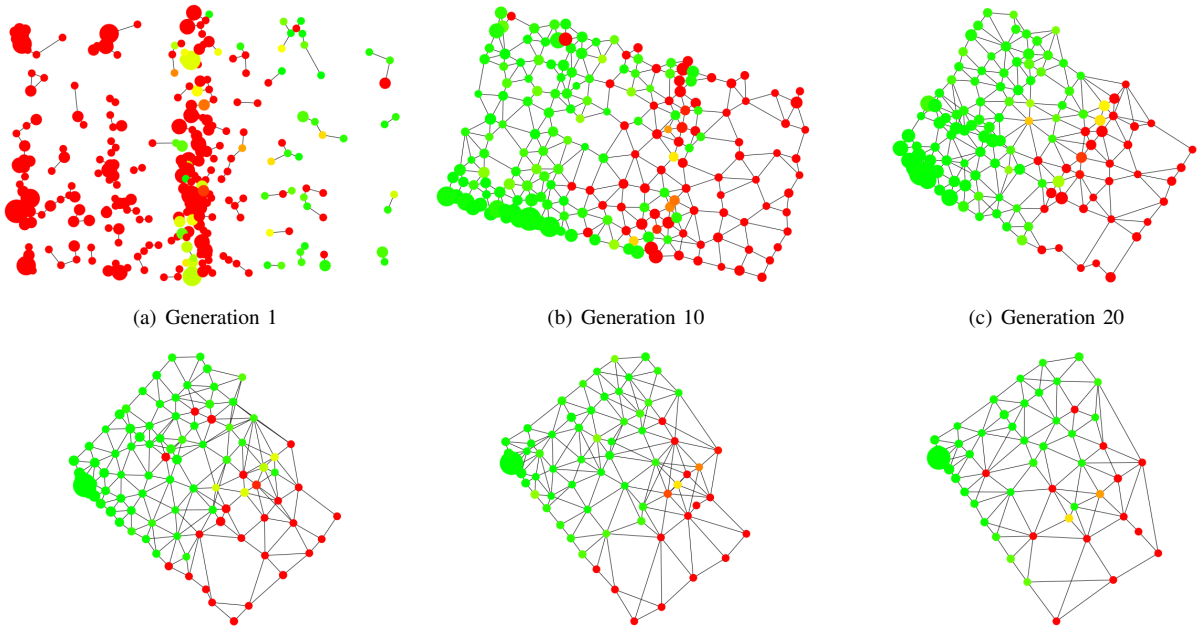
Supplementary Document for the Paper "Visualising the Evolution of Computer Programs for Genetic Programming"

Su Nguyen, *Member, IEEE* and Mengjie Zhang, *Senior Member, IEEE* and Damminda Alahakoon, and Kay Chen Tan, *Fellow, IEEE*

I. Visualizing fitness diversity and phenotypic diversity

A. Cross-generational visualisation

This section presents the visualisation of Genetic Programming (GP) evolution, in which the color of nodes in Growing Neural Gas (GNG) network represent the fitnesses of evolved programs (ranging from green, i.e. good programs, to red, i.e. bad programs). The fitness-based visualisation of GNG networks for TGP (default), TGP-M, TGP-L, TGP-H, SGP, and MGP are presented from Fig. A1 to Fig. A6 (the detailed discussions of these GP algorithms can be found in the paper). The visualisation results here can be used along with the visualisation in Figs. 4–9 (in the paper) to gain more insights about the GP evolutionary process. By comparing these figures, we can confirm that the areas where GP algorithms are indeed the areas with better fitness values. Table A1 is similar to Table IV in the paper but is enriched with the patterns from fitness-based visualisation.



(a) Generation 1 (b) Generation 10 (c) Generation 20

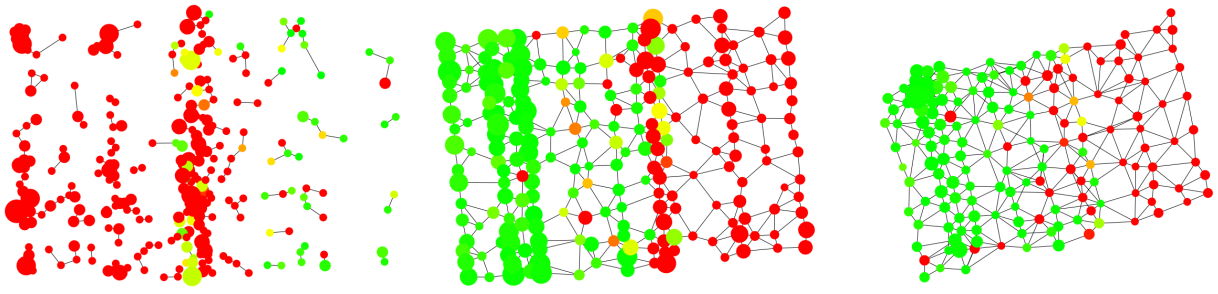
(d) Generation 30 (e) Generation 40 (f) Generation 50

Fig. A1. TGP with high crossover rate default parameters, crossover rate of 80%, mutation rate of 15%. The x-axis and the y-axis are for the first and second principal components respectively obtained from PCA. Nodes which are close to each other represent programs with similar phenotypic characteristics or behaviours. The color of a node represents the fitness values of the corresponding generated programs (red for the worst fitness and green for the best fitness).

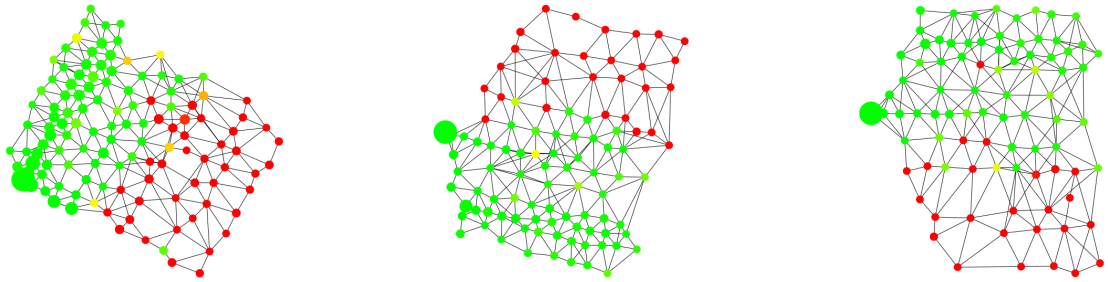
B. Generation snapshot

This section shows the snapshots at Generation 25 from multiple independent runs of the four GP algorithms presented in the paper. These snapshots are similar to ones presented in Fig. 11 in the paper. From these snapshots we can see the common patterns for each algorithm.

The snapshots of the tree-based GP (TGP) are shown in Figs A7–A8. The most common pattern here is that most programs are located in the area with high fitnesses and programs get worse as they move away from this area. This further supports

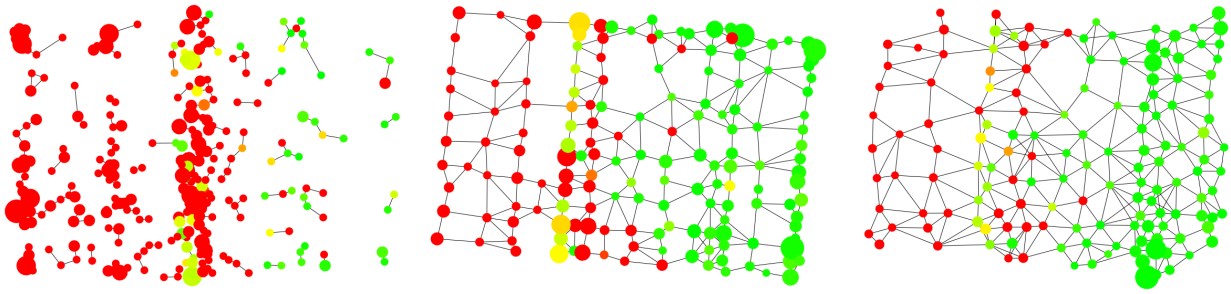


(a) Generation 1 (b) Generation 10 (c) Generation 20

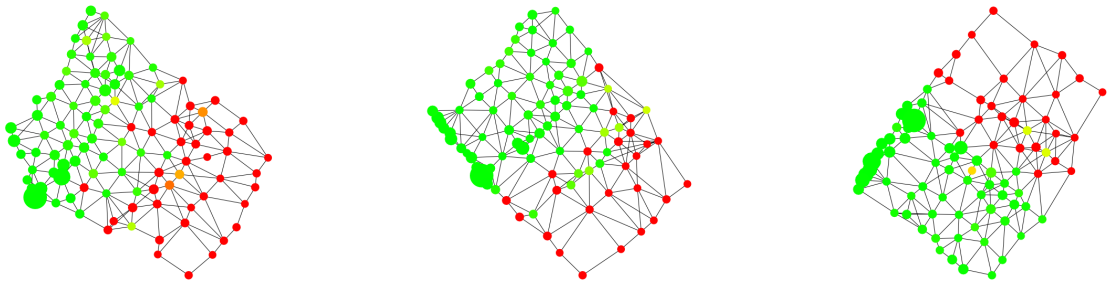


(d) Generation 30 (e) Generation 40 (f) Generation 50

Fig. A2. TGP with high mutation rate (TGP-M).



(a) Generation 1 (b) Generation 10 (c) Generation 20



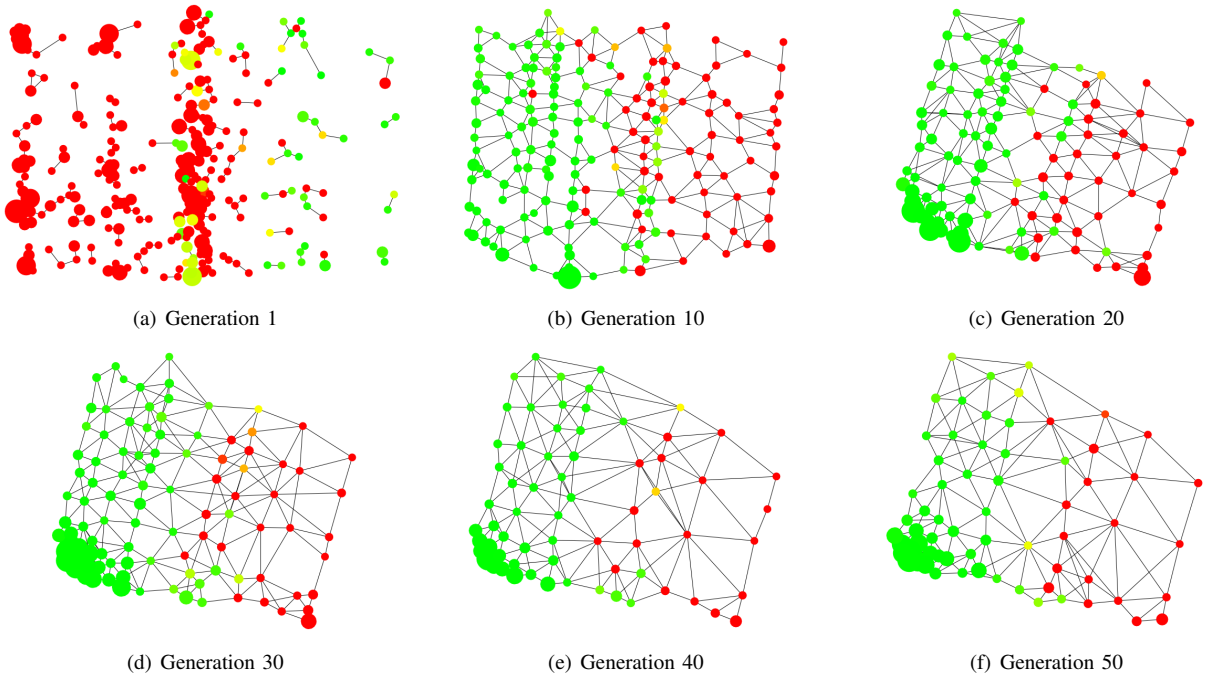
(d) Generation 30 (e) Generation 40 (f) Generation 50

Fig. A3. TGP with low selection pressure (TGP-L).

our claim in Section V.D in the paper. The pattern in Run 1 is different because TGP has not yet converged in this particular run.

The snapshots of the TGP with low selection pressure (TGP-L) are shown in Figs A9–A10. In all snapshots, we can see that TGP-L explore the search space much more aggressively as compared to the default TGP above. Because the network obtained with TGP-L covers a larger area, it is more difficult to discriminate bad programs from good programs although there are some correlations between phenotypic characteristics and program fitnesses. This indicates that the search space of evolved programs are very complex with multiple local optima.

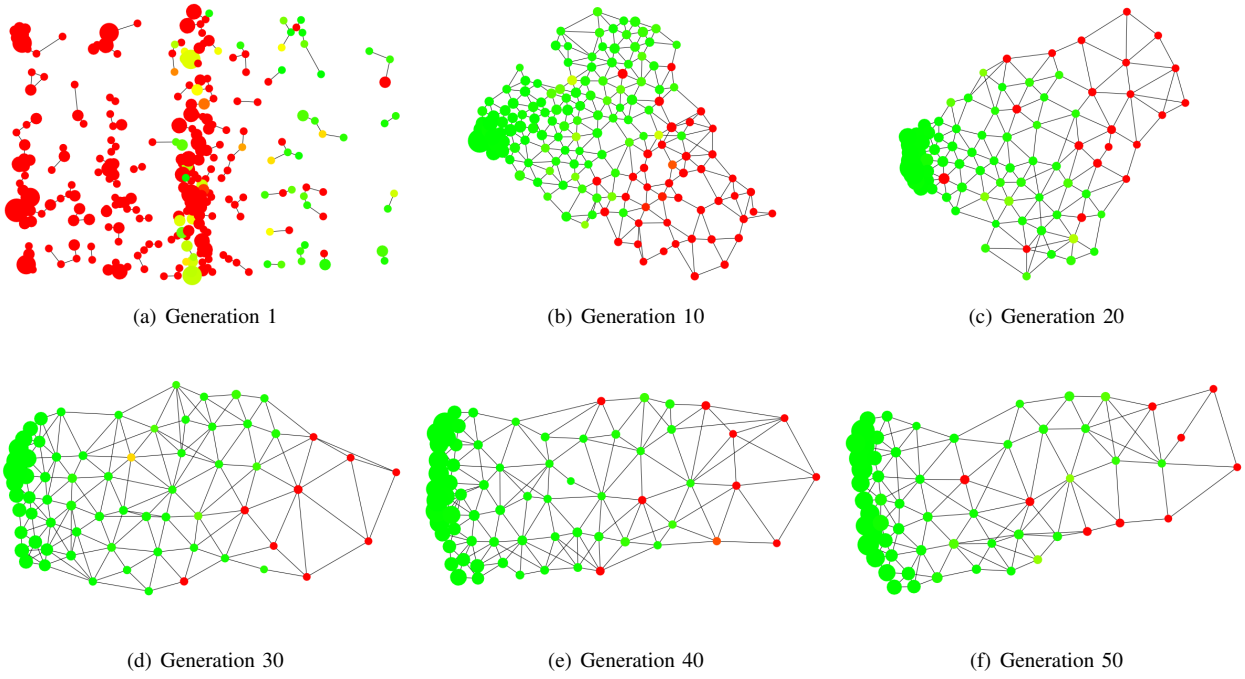
The snapshots of surrogate-assisted GP (SGP) are shown in Figs A11–A12. Due to the pre-selection scheme based on the surrogate model, it is more likely for SGP to generate good programs as compared to TGP or TGP-L. As compared to TGP-L,



(a) Generation 1 (b) Generation 10 (c) Generation 20

(d) Generation 30 (e) Generation 40 (f) Generation 50

Fig. A4. TGP with high selection pressure (TGP-H).



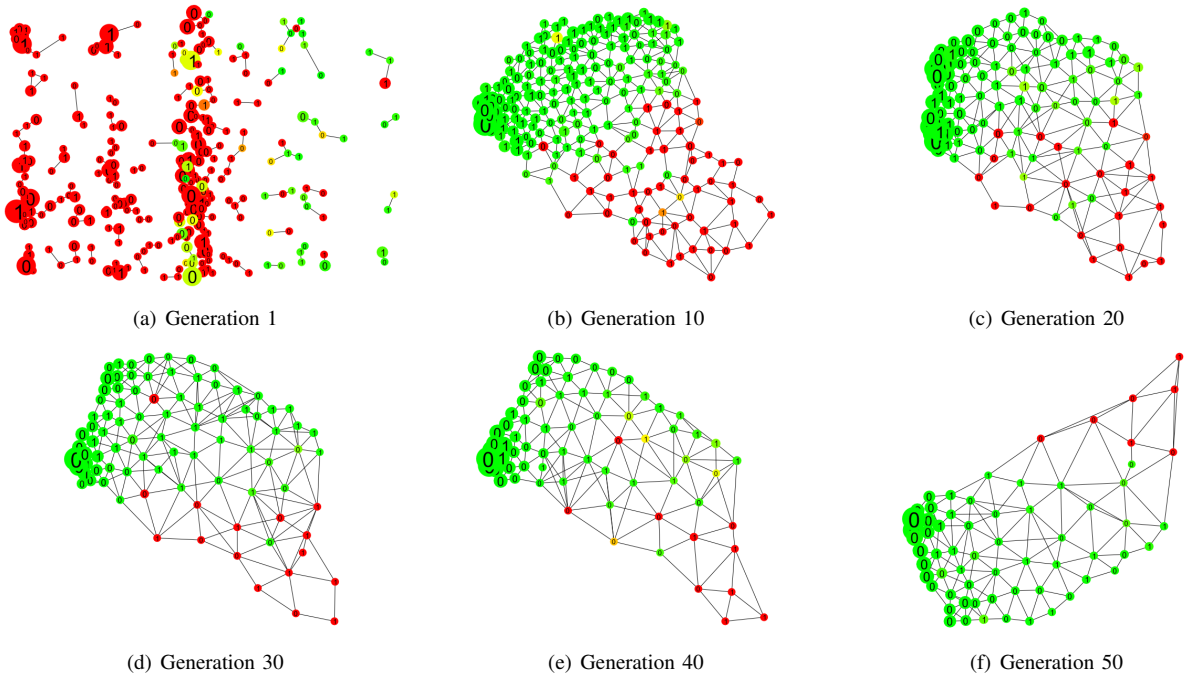
(a) Generation 1 (b) Generation 10 (c) Generation 20

(d) Generation 30 (e) Generation 40 (f) Generation 50

Fig. A5. Surrogate-assisted GP.

SGP still maintains a good diversity in the population but focuses the search on a more restricted area.

The snapshots of multipopulation GP (MGP) are shown in Figs A13–A14. MGP is similar to SGP in term of maintaining a large number of good programs in the population but tends to form two main clusters (labeled with 0 and 1) for the two subpopulations. This pattern suggests that the genetic materials or building blocks from each subpopulation do restrict behaviours of programs evolved in that subpopulation. Certain building blocks will be more useful to explore programs with certain characteristics.



(a) Generation 1 (b) Generation 10 (c) Generation 20

(d) Generation 30 (e) Generation 40 (f) Generation 50

Fig. A6. Multipopulation GP.

TABLE A1
COMMON OBSERVED PATTERNS FROM FITNESS-BASED VISUALISATION

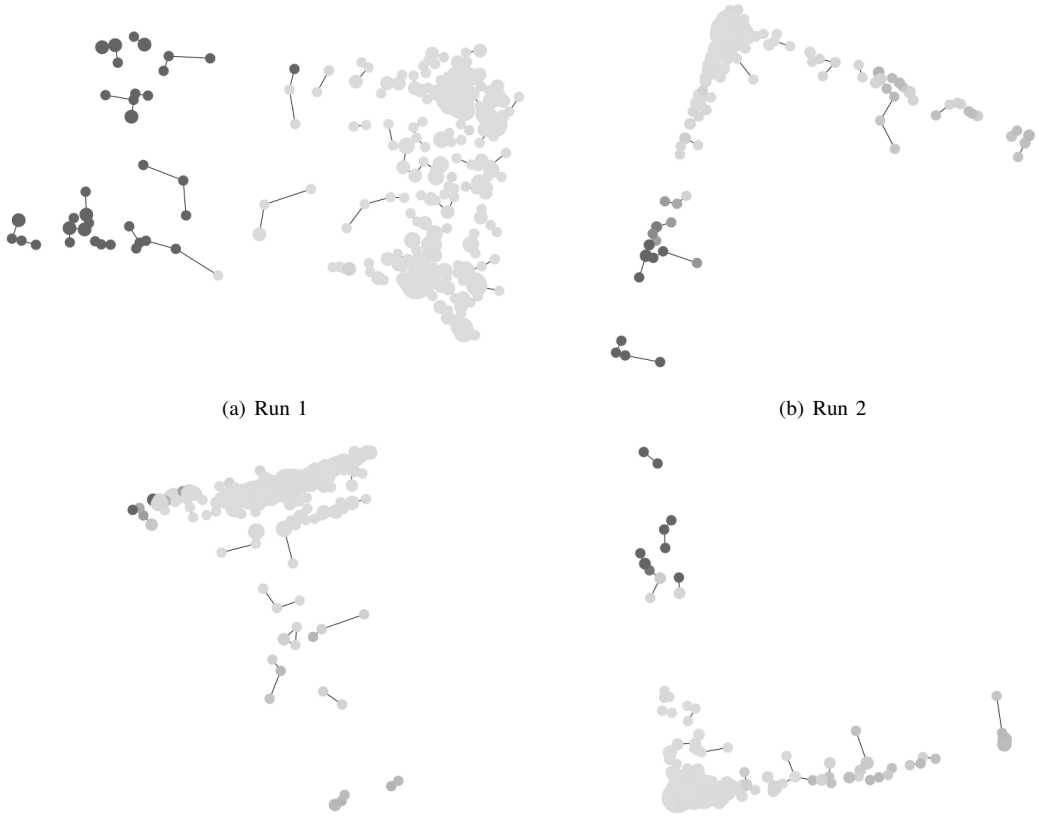
Pattern	Behaviour	Algorithm					
		TGP-C	TGP-M	TGP-L	TGP-H	SGP	MGP
Large number of components and nodes with different colours	initialisation, random search	○	○	○	○	○	○
Large nodes with different colours scattered randomly over the network	uninformed or ill-informed exploration	○◐	○◐⊙	○◐⊙	○	○	○
Network rotation	genetic drift	○◐	○◐⊙◑	○◐⊙◑	○◐	○◐⊙	○◐⊙◑
Large nodes in a narrow "greener" area of the network and only a few "red" nodes	exploitation, convergence	⊙◑	⊙◑	◑	◐⊙◑	◐⊙◑	◐⊙◑
Large "green" nodes in a wide area of the network	exploitation, exploration	⊙			◐	◐⊙◑	◐⊙◑
Large nodes forming multiple "green" clusters in the network	exploitation, exploration						◐⊙

○: first generations; ◐: early generations; ⊙: mid-generations; ◑: later generations

II. PSEUDO CODES FOR GP ALGORITHMS

Fig. A15 shows the pseudo code for the TGP algorithm examined in the paper. Each program Δ_i contains two trees for dispatching rules and routing rules. To apply crossover and mutation, a random program is selected based on tournament selection. From the selected program, a random tree is selected and the genetic operations are applied to generate new programs.

Fig. A16 shows the pseudo code for the surrogate-assisted GP (SGP) algorithm. This is similar to TGP except that an intermediate population P' is produced by genetic operations and nearest neighbor is used to quickly estimate fitnesses of newly generated programs before moving the most promising programs to P for the next generation. To improve the diversity, the size of P' will be much bigger than P (in our experiments $|P'| = 5|P|$ and duplicated programs (based on their phenotypic characteristics) are allowed in P. The multipopulation GP (MGP) is similar to SGP but with two subpopulations. During the evolution, the shared archive $\mathcal{A}$ is used for estimating the fitnesses of programs in their corresponding intermediate population.



(a) Run 1

(b) Run 2

(c) Run 3

(d) Run 4

Fig. A7. Visualizing fitness diversity and phenotypic diversity of TGP – Generation 25.

(a) Run 5 (b) Run 6

(c) Run 7 (d) Run 8

(e) Run 9 (f) Run 10

Fig. A8. Visualizing fitness diversity and phenotypic diversity of TGP – Generation 25 (cont.).

(a) Run 1

(b) Run 2

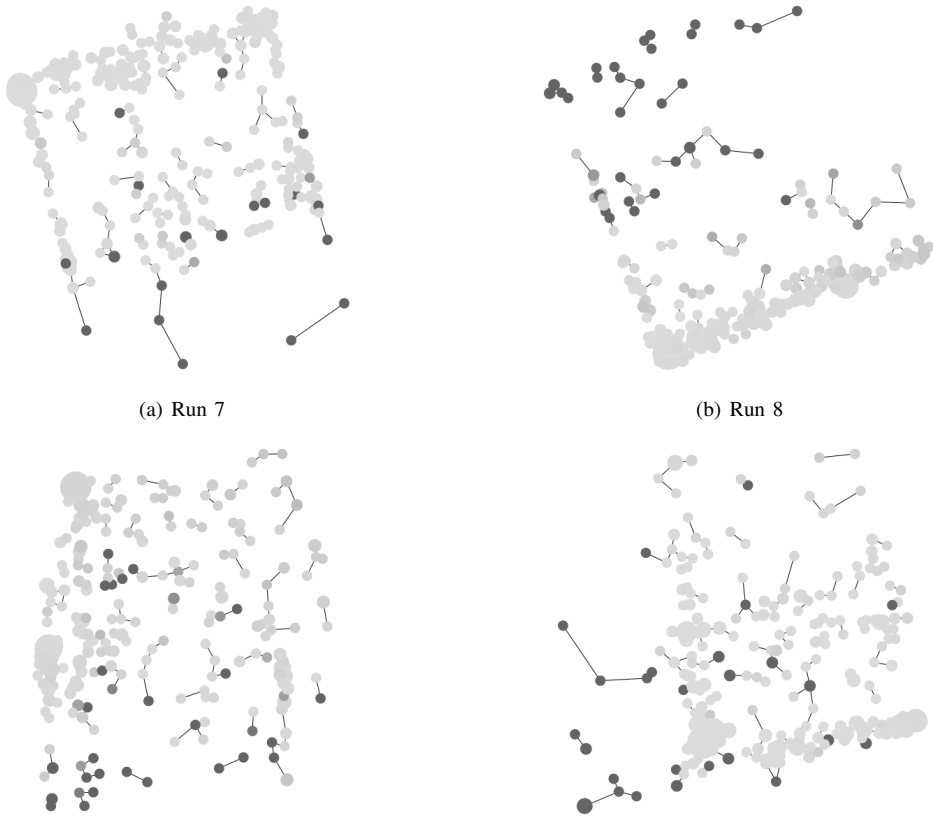
(c) Run 3

(d) Run 4

(e) Run 5

(f) Run 6

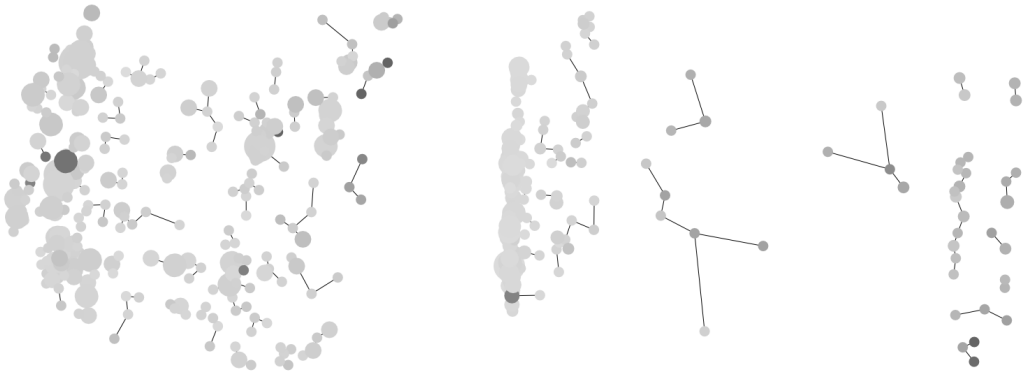
Fig. A9. Visualizing fitness diversity and phenotypic diversity of TGP-L – Generation 25.



(a) Run 7 (b) Run 8

(c) Run 9 (d) Run 10

Fig. A10. Visualizing fitness diversity and phenotypic diversity of TGP-L – Generation 25 (cont.).

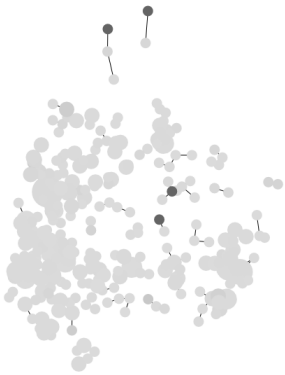


(a) Run 1 (b) Run 2

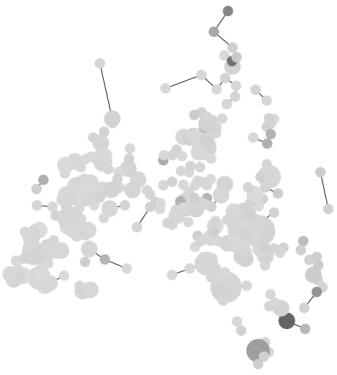
Fig. A11. Visualizing fitness diversity and phenotypic diversity of SGP – Generation 25.

(a) Run 3

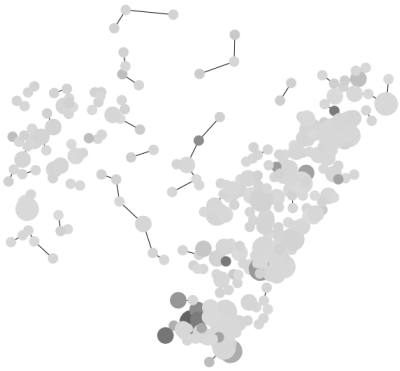
(b) Run 4



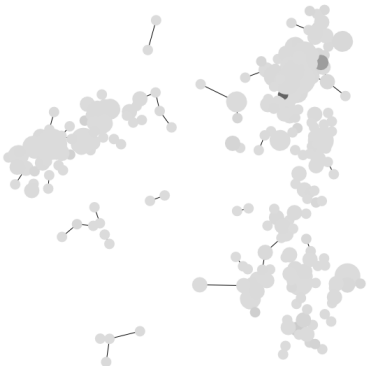
(c) Run 5



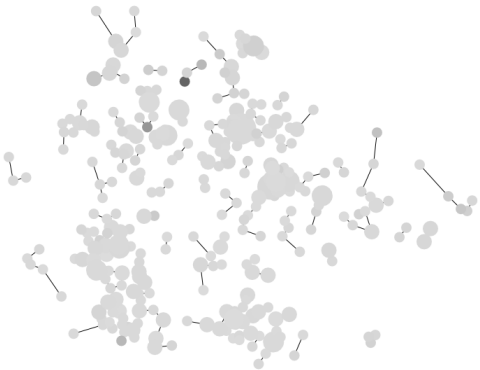
(d) Run 6



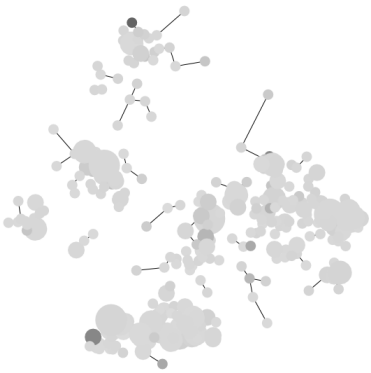
(e) Run 7



(f) Run 8

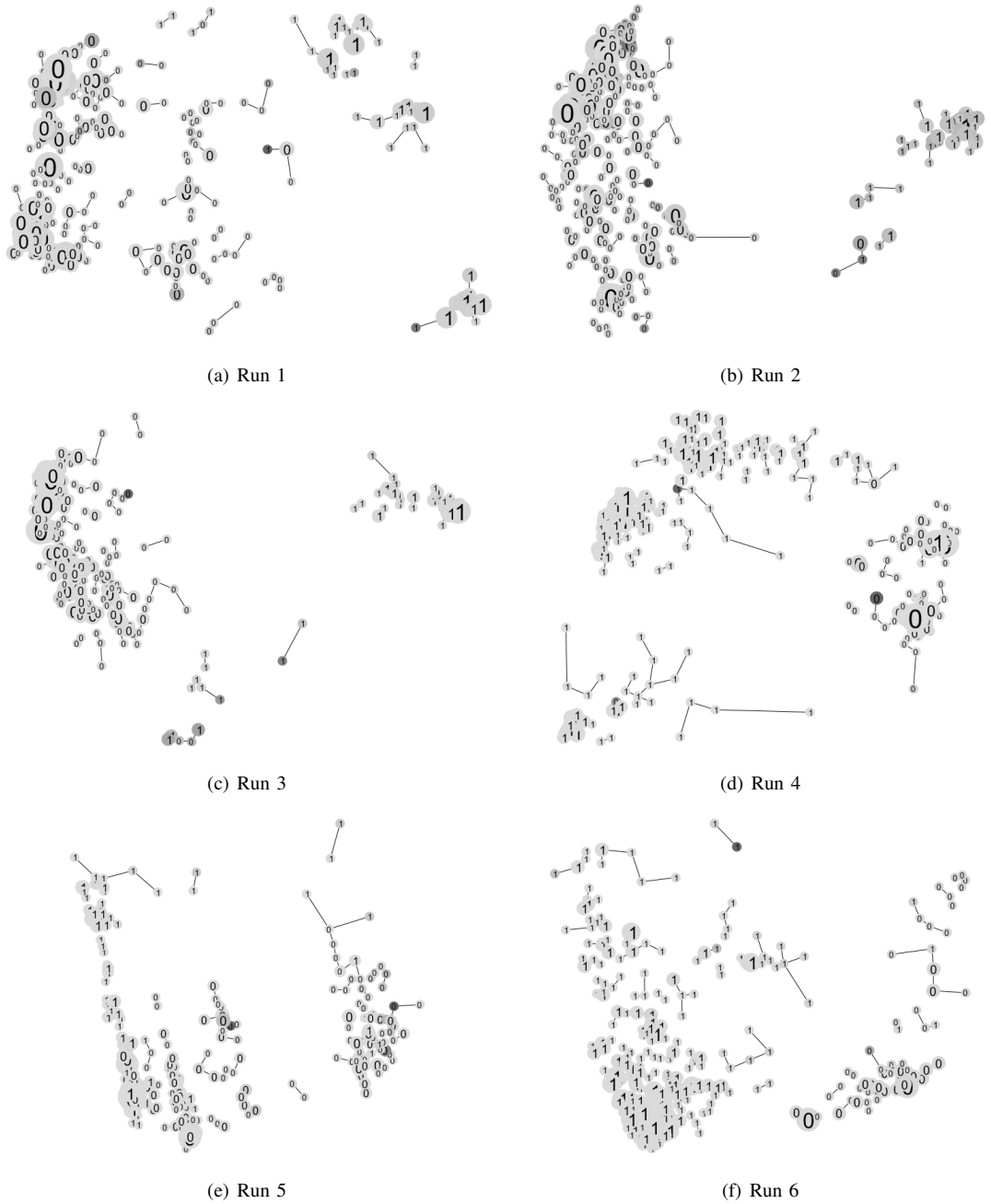


(g) Run 9



(h) Run 10

Fig. A12. Visualizing fitness diversity and phenotypic diversity of SGP – Generation 25 (cont.).

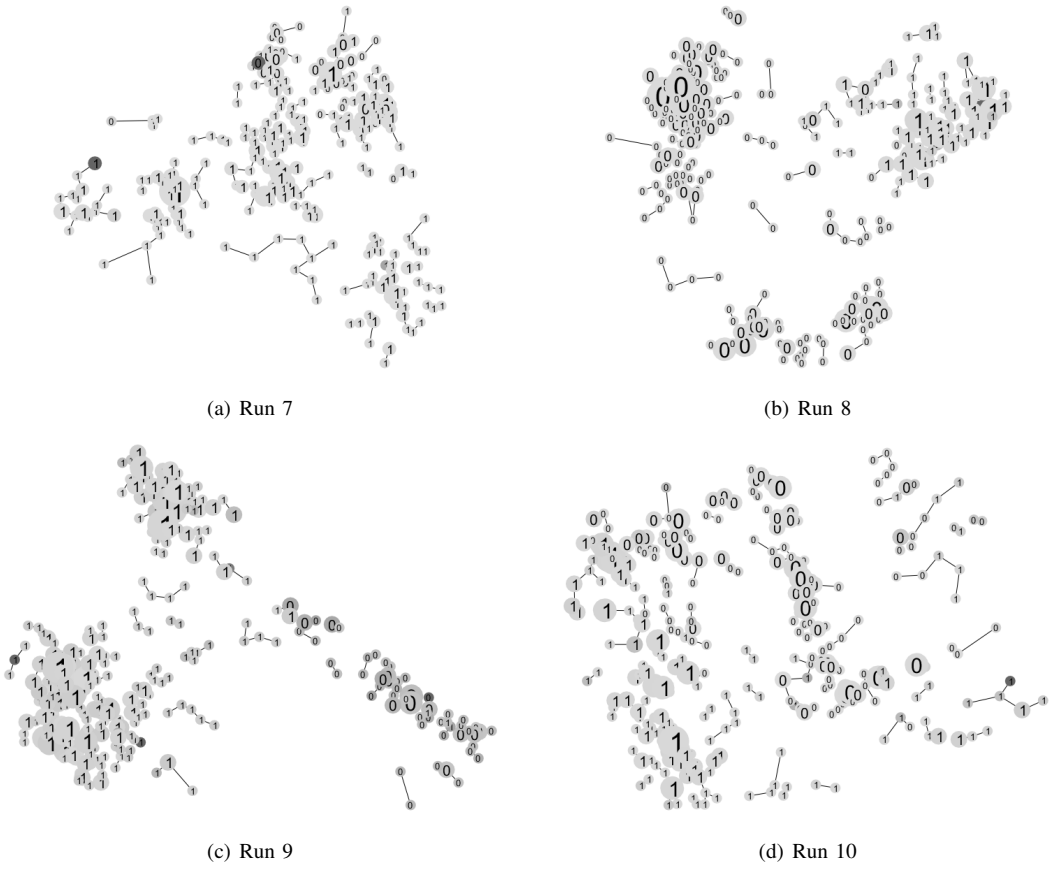


(a) Run 1 (b) Run 2

(c) Run 3 (d) Run 4

(e) Run 5 (f) Run 6

Fig. A13. Visualizing fitness diversity and phenotypic diversity of MGP – Generation 25.



(a) Run 7 (b) Run 8

(c) Run 9 (d) Run 10

Fig. A14. Visualizing fitness diversity and phenotypic diversity of MGP – Generation 25 (cont.).

Inputs: simulation scenario I
Output: the best evolved program Δ^*

randomly initialise the population $P \leftarrow \{\Delta_1, \ldots, \Delta_S\}$
set $\Delta^* \leftarrow null$ and $fitness(\Delta^*) = +\infty$
$generation \leftarrow 0$
while $generation \leq maxGenerations$ **do**
    for all $\Delta_i \in P$ **do**
        apply rule Δ_i to the scenario I
        evaluate $fitness(\Delta_i) \leftarrow$ total weighted tardiness
        if $fitness(\Delta_i) < fitness(\Delta^*)$ **then**
            $\Delta^* \leftarrow \Delta_i$
            $fitness(\Delta^*) \leftarrow fitness(\Delta_i)$
        end if
    end for
    $P \leftarrow$ apply reproduction, crossover, mutation to P
    $generation \leftarrow generation + 1$
end while
return Δ^*

Fig. A15. TGP algorithm to evolve dispatching rules and routing rules.

Inputs: simulation scenario I
Output: the best evolved program Δ^*

randomly initialise the population $P \leftarrow \{\Delta_1, \ldots, \Delta_S\}$
set $\Delta^* \leftarrow null$ and $fitness(\Delta^*) = +\infty$
$generation \leftarrow 0$
while $generation \leq maxGenerations$ **do**
    $\mathcal{A} \leftarrow \emptyset$
    for all $\Delta_i \in P$ **do**
        apply rule Δ_i to the scenario I
        evaluate $fitness(\Delta_i) \leftarrow$ total weighted tardiness
        if $fitness(\Delta_i) < fitness(\Delta^*)$ **then**
            $\Delta^* \leftarrow \Delta_i$
            $fitness(\Delta^*) \leftarrow fitness(\Delta_i)$
        end if
        $\mathcal{A} \leftarrow \mathcal{A} \cup (\Delta_i, fitness(\Delta_i))$
    end for
    $P' \leftarrow$ apply reproduction, crossover, mutation to P ($|P'| > |P|$)
    estimate fitness of $\Delta_i \in P'$ with the nearest neighbor technique using the archive $\mathcal{A}$
    $P \leftarrow$ select programs with top estimate fitnesses in P'
    $generation \leftarrow generation + 1$
end while
return Δ^*

Fig. A16. SGP algorithm to evolve dispatching rules and routing rules.