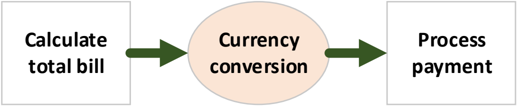
A GP Approach to QoS-Aware Web Service Composition including Conditional Constraints

Alexandre Sawczuk da Silva, Hui Ma, Mengjie Zhang

IEEE Congress on Evolutionary Computation, 25-28 May 2015

Introduction

Service-Oriented Architecture (SOA): Organise processes and data in reusable modules for integration into new applications.

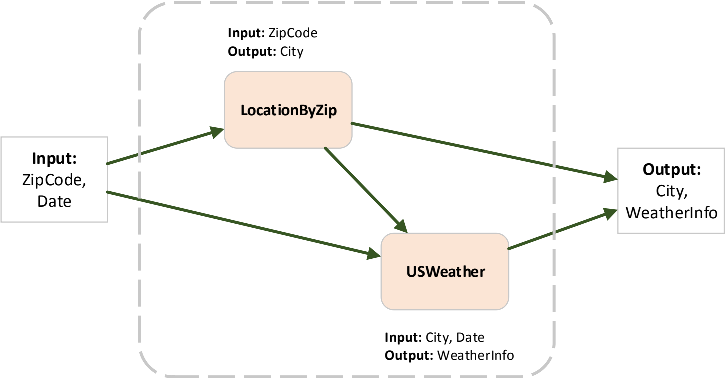


Web service

A functionality module that provides operations accessible over the network via a standard communication protocol.

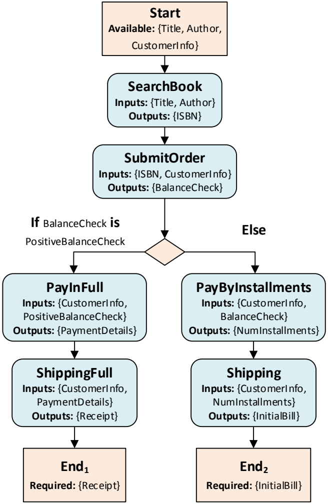
Web Service Composition

The combination of Web services to achieve a more complex task.
Fully automated scenario:



New weather by zip code service

A Composition Example with Branching



Certain compositions require alternative paths according to runtime values.

Example: Depending on balance, pay in full or pay in installments.

Composition Dimensions

1. **Solution feasibility:** Service inputs and outputs must be properly linked (e.g. *City* $\rightarrow$ *Location*, but not *PhoneNumber* $\rightarrow$ *Location*).

2. **Conditional constraints:** Condition leading to multiple possible execution paths (e.g. if *City* is a *NewZealandCity*, produce *WindForecast* instead of *GeneralForecast*).

3. **Quality of Service (QoS):** The overall quality of the composition (e.g. lowest execution time, lowest cost).

Existing Approaches

AI Planning

Build a solution service by service.
Dimensions: *Solution feasibility, conditional constraints.*

Evolutionary Computation (EC)

Improve population of solutions over multiple generations.
Dimensions: *Solution feasibility, QoS.*

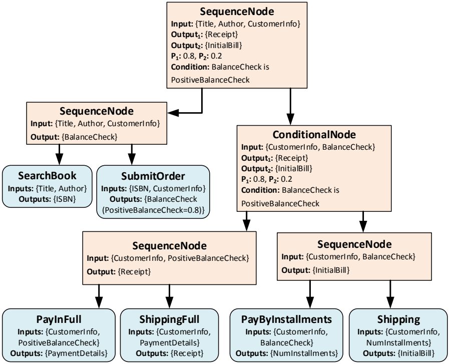
Hybrid Approaches

Combine AI planning and EC ideas.
Dimensions: *Solution feasibility, QoS.*

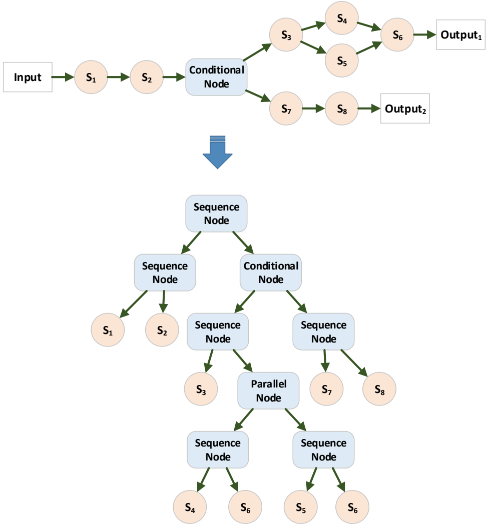
Goal

To propose a Genetic Programming (GP) composition approach that simultaneously considers all dimensions.

1. Trees preserve solution feasibility.
2. Conditions encoded in trees.
3. Optimisation performed on QoS.



Candidate Representation

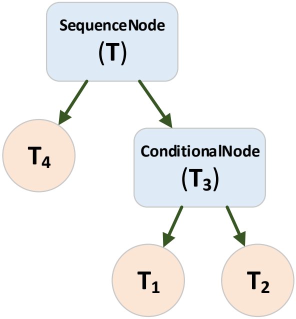


- Tree equivalent to graph composition.
- Parallel, sequential, and conditional represented as non-terminal nodes.
- Candidate services as terminal nodes.

Population Initialisation

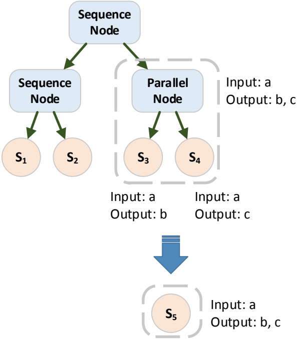
An algorithm is used to create a candidate in graph format, and then translate it into a tree representation.

```
Input  : I, O1, O2, C, P
Output: candidate tree T
if O2 ≠ ∅ then
    G1 ← createGraph(I ∪ C.if, O1);
    G2 ← createGraph(I∪C.else, O2);
    T1 ← toTree(G1.input);
    T2 ← toTree(G2.input);
    T3 ← new ConditionalNode(C);
    T3.leftChild ← T1;
    T3.rightChild ← T2;
    if C ⊑ I then
        T3.prob ← P;
        return T3;
    else
        G4 ← createGraph(I, C.else);
        T4 ← toTree(G4.input);
        T3.prob ← T4.final.P;
        T ← new SequenceNode();
        T.leftChild ← T4;
        T.rightChild ← T3;
        return T;
    end
else
    G ← createGraph(I, O1);
    T ← toTree(G.input);
    return T;
end
```

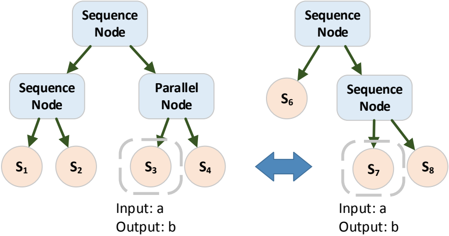


Mutation and Crossover

Mutation: Selects random node and replaces it with equivalent subtree.



Crossover: Swaps any two equivalent terminal nodes.

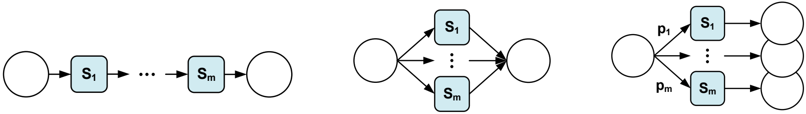


Fitness Function

Measures the overall quality of a composition candidate (minimising).

$$fitness_i = w_1(1 - A_i) + w_2(1 - R_i) + w_3 T_i + w_4 C_i$$

$$\text{where } \sum_{i=1}^{4} w_i = 1$$



Experiments

- Lack of datasets supporting composition with branching.
- Lack of comparable approaches that produce solutions with multiple output possibilities.

Decision: Create datasets, execute for conditional compositions and also for each branch separately.

Parameters:

Independent runs	50	Elitism candidates	1
Population size	20	Tournament size	7
Crossover probability	0.9	Fitness weights	0.25 (all)
Mutation probability	0.1		

Creation of Datasets

Modified from WSC2008.

- Can be extended to contain QoS.
- Provides ontology of input and output values.

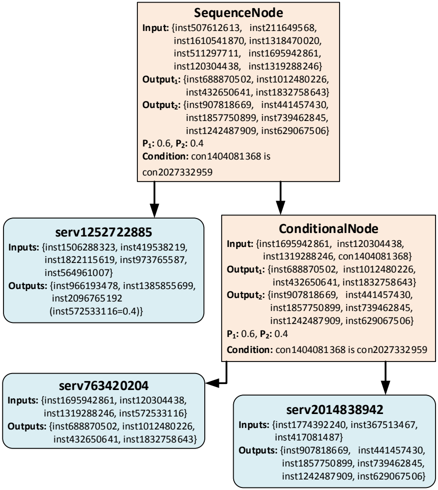
Tasks requiring branching were created.

Results

	Conditional	
Set (size)	**Avg. fitness**	**Avg. time (s)**
1 (158)	0.60 ± 0.01	1.29 ± 0.10
2 (558)	0.71 ± 0.01	2.83 ± 0.25
3 (604)	0.63 ± 0.01	13.29 ± 1.23
4 (1041)	0.72 ± 0.05	6.15 ± 0.57
5 (1090)	0.70 ± 0.01	11.76 ± 0.95
6 (2198)	0.66 ± 0.02	92.39 ± 11.35
7 (4113)	0.58 ± 0.01	97.34 ± 13.71
8 (8119)	0.66 ± 0.01	326.39 ± 37.66

	Non-conditional			
	If branch		Else branch	
Set (size)	**Avg. fitness**	**Avg. time (s)**	**Avg. fitness**	**Avg. time (s)**
1 (158)	0.51 ± 0.00	0.56 ± 0.14	0.59 ± 0.04	0.72 ± 0.08
2 (558)	0.59 ± 0.08	1.49 ± 0.53	0.69 ± 0.02	1.53 ± 0.19
3 (604)	0.37 ± 0.00	4.39 ± 0.77	0.79 ± 0.00	7.10 ± 0.91
4 (1041)	0.69 ± 0.06	4.51 ± 1.18	0.74 ± 0.43	3.57 ± 0.43
5 (1090)	0.45 ± 0.00	5.73 ± 0.76	0.69 ± 0.01	6.49 ± 0.74
6 (2198)	0.41 ± 0.06	58.30 ± 12.77	0.65 ± 0.02	52.31 ± 5.79
7 (4113)	0.36 ± 0.00	44.84 ± 5.93	0.69 ± 0.03	51.73 ± 4.26
8 (8119)	0.47 ± 0.00	106.12 ± 7.15	0.77 ± 0.00	186.90 ± 20.01

Solution Example



Conclusions

Novel approach addresses three composition dimensions simultaneously (fully feasible, contain branches, quality-optimised).

- Solutions found with similar performance as non-branching technique.

Future work: More than two branches, more complex branching conditions, analysis of convergence behaviour.

Thank you!

Questions?