Evolutionary Multi-Objective Optimization: A Parallel Computing Approach

Rahul Krishna George Mathew

Department of Computer Science
North Carolina State University

December 9, 2015

Outline

Background
- → Multi-Objective Problem

Models
- → DTLZ2
- → XOMO
- → POM3

Algorithms
- → Evolutionary Algorithm
- → Differential Evolution
- → GALE: Geometric Active Learner

Parallelization Strategies
- → The Island Model
- → Master-Slave Model

Evaluation Metrics
- → Evaluation
- → Measures

Experimental Setup

Results
- → Island Model
- → Master-Slave Model

Future Work
- → Feature Models
- → Results: Feature Models
- → Other Extensions:

Outline

Multi-Objective Problem

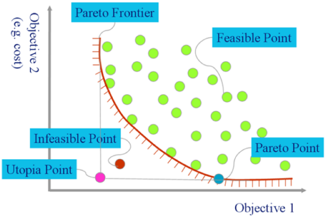


Figure: Sample Pareto Frontier

Multi-Objective Problem

- **Pareto Frontier** State of solutions which are equally good.

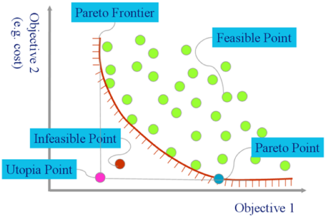


Figure: Sample Pareto Frontier

Multi-Objective Problem

- **Pareto Frontier** State of solutions which are equally good.
- **Pareto Point** A point that lies on the Pareto frontier.

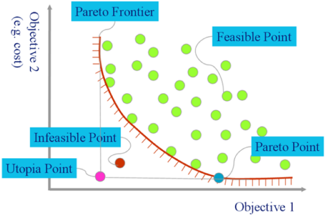


Figure: Sample Pareto Frontier

Multi-Objective Problem

- **Pareto Frontier** State of solutions which are equally good.
- **Pareto Point** A point that lies on the Pareto frontier.
- **Feasible Point** A satisfiable solution for the problem but not necessarily the optimum one.

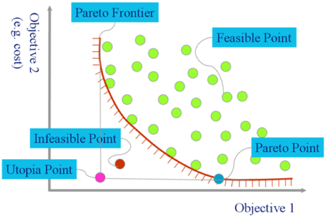


Figure: Sample Pareto Frontier

Multi-Objective Problem

- **Pareto Frontier** State of solutions which are equally good.
- **Pareto Point** A point that lies on the Pareto frontier.
- **Feasible Point** A satisfiable solution for the problem but not necessarily the optimum one.
- **Infeasible Point** A solution outside the Pareto frontier

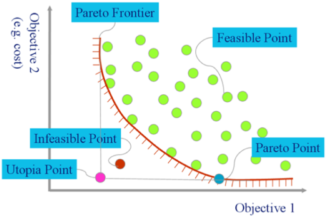


Figure: Sample Pareto Frontier

Multi-Objective Problem

- **Pareto Frontier** State of solutions which are equally good.
- **Pareto Point** A point that lies on the Pareto frontier.
- **Feasible Point** A satisfiable solution for the problem but not necessarily the optimum one.
- **Infeasible Point** A solution outside the Pareto frontier
- **Utopia Point** The ideal theoretical solution we would love to reach but practically its not possible

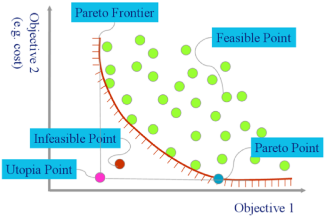


Figure: Sample Pareto Frontier

Outline

DTLZ-2

- **Decisions** DTLZ-2 has 30 decisions where each decision ranges between 0 and 1.

$$0 \leq x_i \leq 1 \quad \textit{where} \quad i = 1, 2, 3....30$$

- **Objectives** A point that lies on the Pareto frontier.

$$f_1(x) = (1 + g(x_M)) \cos(x_1\pi/2).... \cos(x_{M-1}\pi/2)$$

$$f_2(x) = (1 + g(x_M)) \cos(x_1\pi/2).... \cos(x_{M-1}\pi/2)$$

$$f_3(x) = (1 + g(x_M)) \sin(x_1\pi/2)$$

$$\textit{where} \quad g(x_M) = \sum_{x \in x_M} (x_i - 0.5)^2$$

- **Optimal Solution:** Ideal Decisions are $x_i = 0.5$ where $i = 1, 2, 3...30$
  Ideal objectives should satisfy the equation $\sum_{m=1}^{3} f_m^2 = 1$

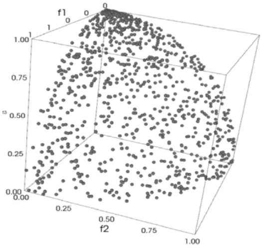


Figure: Pareto Frontier

Outline

XOMO

- Monte Carlo Simulator modelling NASA's space program software.

XOMO

- Monte Carlo Simulator modelling NASA's space program software.
- 23 decisions - lines of code, storage, cyclometric complexity etc.

XOMO

- Monte Carlo Simulator modelling NASA's space program software.
- 23 decisions - lines of code, storage, cyclometric complexity etc.
- 4 objectives all minimized
  - Total Developer **Effort**.
  - **Months** to complete project.
  - Total **Defects** in project.
  - **Risk** involved in project.

Outline

POM3

- Implements Boehm & Turner model of agile programming where

POM3

- Implements Boehm & Turner model of agile programming where
- Teams select tasks as they appear in the scrum backlog.

POM3

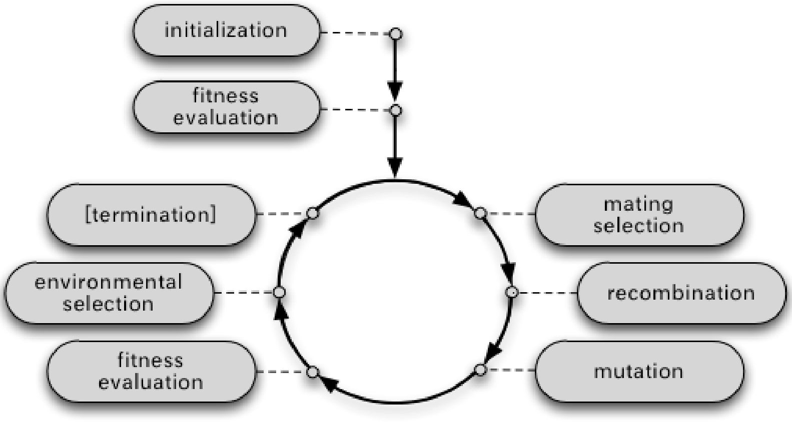
- Implements Boehm & Turner model of agile programming where
- Teams select tasks as they appear in the scrum backlog.
- 9 decisions like size of project, project plan, team size etc.

POM3

- Implements Boehm & Turner model of agile programming where
- Teams select tasks as they appear in the scrum backlog.
- 9 decisions like size of project, project plan, team size etc.
- The model contains 4 objectives
  - Minimize **Cost**.
  - Maximize **Utility**.
  - Maximize **Completion Percentage**.
  - Minimize **Idle Time** for Developers.

Outline

Evolutionary Algorithm



Outline

Differential Evolution(DE)

- Stochastic evolutionary optimization technique.

Differential Evolution(DE)

- Stochastic evolutionary optimization technique.
- Iteratively approximates the shape of the Pareto Frontier

Differential Evolution(DE)

- Stochastic evolutionary optimization technique.
- Iteratively approximates the shape of the Pareto Frontier
- Advantages:
  - Simple & Computationally Inexpensive.
  - High dimensional problems can be handled easily.
  - Solutions are very stable.

DE - Algorithm

Pseudo-code 3.2 Differential Evolution

```
Begin
   Generate randomly an initial population of solutions.
   Calculate the fitness of the initial population.
   Repeat
      For each parent, select three solutions at random.
      Create one offspring using the DE operators.
      Do this a number of times equal to the population size.
      For each member of the next generation
         If offspring(x) is more fit than parent(x)
            Parent(x) is replaced.
   Until a stop condition is satisfied.
End.
```

Outline

Geometric Active LEarner (GALE)

- Near linear time Multi-Objective Evolutionary Algorithm.

Geometric Active LEarner (GALE)

- Near linear time Multi-Objective Evolutionary Algorithm.
- Builds piecewise approximation to the best solutions of the pareto frontier.

Geometric Active LEarner (GALE)

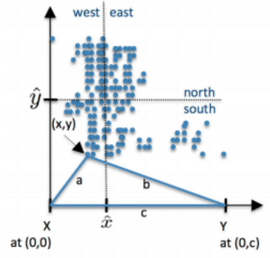
- Near linear time Multi-Objective Evolutionary Algorithm.
- Builds piecewise approximation to the best solutions of the pareto frontier.
- Based on WHERE which is a recursive clustering technique based on Dimensionality Reduction.

Geometric Active LEarner (GALE)

- Near linear time Multi-Objective Evolutionary Algorithm.
- Builds piecewise approximation to the best solutions of the pareto frontier.
- Based on WHERE which is a recursive clustering technique based on Dimensionality Reduction.
- Advantages:
  - Less Number of Computations.
  - Adept at handling problems that are **non-differentiable**, **non-liner**, **multi-dimensional** or **multi-constraint**.
  - Concise representation of problem space.

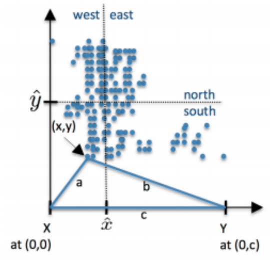
GALE - Algorithm

- Cluster data based on WHERE



GALE - Algorithm

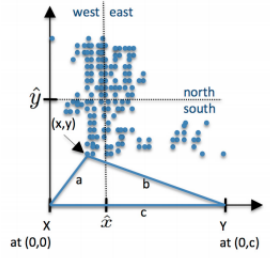
- Cluster data based on WHERE



- Pick point **X** from the cluster. Then pick point **East** furthest from **X** and point **West** furthest from **East**. Let **c** be the distance between **East** and **West**.

GALE - Algorithm

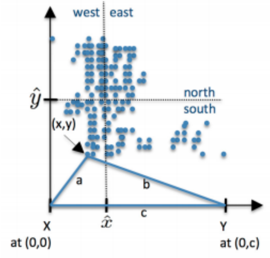
- Cluster data based on WHERE



- Pick point **X** from the cluster. Then pick point **East** furthest from **X** and point **West** furthest from **East**. Let **c** be the distance between **East** and **West**.
- For every other point in the cluster, compute **a** and **b** which represents distance of the point from **East** and **West** respectively.

GALE - Algorithm

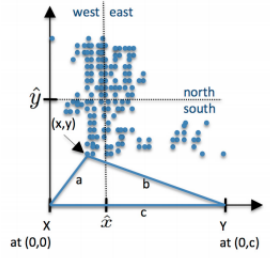
- Cluster data based on WHERE



- Pick point **X** from the cluster. Then pick point **East** furthest from **X** and point **West** furthest from **East**. Let **c** be the distance between **East** and **West**.
- For every other point in the cluster, compute **a** and **b** which represents distance of the point from **East** and **West** respectively.
- Compute the projection **x** as $\boldsymbol{x} = (\boldsymbol{a}^2 + \boldsymbol{c}^2 - \boldsymbol{b}^2)/2\boldsymbol{c}$

GALE - Algorithm

- Cluster data based on WHERE

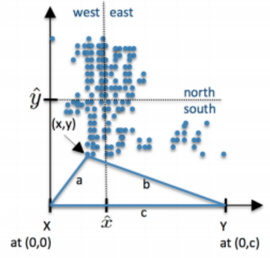


- Pick point **X** from the cluster. Then pick point **East** furthest from **X** and point **West** furthest from **East**. Let **c** be the distance between **East** and **West**.
- For every other point in the cluster, compute **a** and **b** which represents distance of the point from **East** and **West** respectively.
- Compute the projection **x** as $\boldsymbol{x} = (\boldsymbol{a}^2 + \boldsymbol{c}^2 - \boldsymbol{b}^2)/2\boldsymbol{c}$

- Select the best point from the non-dominated cluster and mutate towards it and store the best points.

GALE - Algorithm

- Cluster data based on WHERE



- Pick point **X** from the cluster. Then pick point **East** furthest from **X** and point **West** furthest from **East**. Let **c** be the distance between **East** and **West**.
- For every other point in the cluster, compute **a** and **b** which represents distance of the point from **East** and **West** respectively.
- Compute the projection **x** as $\boldsymbol{x} = (\boldsymbol{a}^2 + \boldsymbol{c}^2 - \boldsymbol{b}^2)/2\boldsymbol{c}$

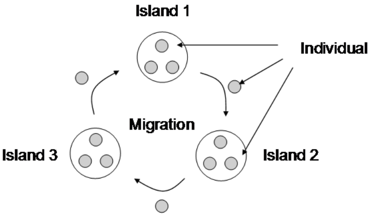
- Select the best point from the non-dominated cluster and mutate towards it and store the best points.
- Repeat for **n** generations.

Outline

Island Model

- Divide the initial population(**N**) into sub-populations of equal size(**n**) among **k** processors.

$$n = \frac{N}{k}$$

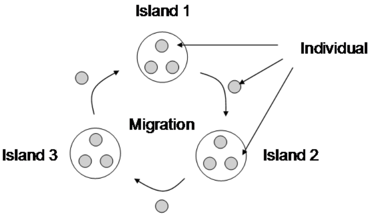


Island Model

- Divide the initial population(**N**) into sub-populations of equal size(**n**) among **k** processors.

$$n = \frac{N}{k}$$

- Evolve each sub-population independently.

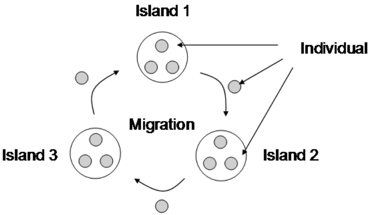


Island Model

- Divide the initial population(**N**) into sub-populations of equal size(**n**) among **k** processors.

$$n = \frac{N}{k}$$

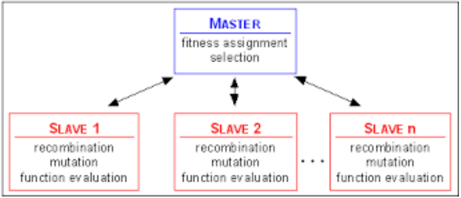
- Evolve each sub-population independently.
- Aggregate the final population of each processor after the total number of generations.



Outline

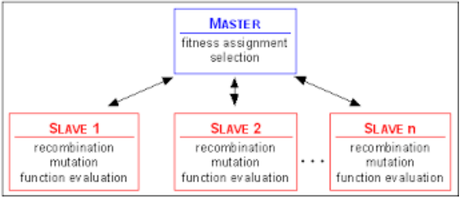
Master-Slave Model

- Master selects a population for each free slave in each generation.



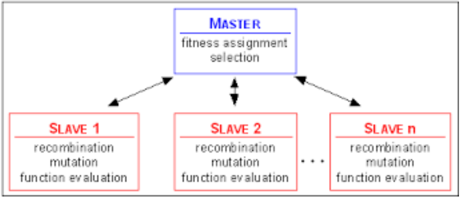
Master-Slave Model

- Master selects a population for each free slave in each generation.
- Slave evaluates the fitness and computes the best solution(s) for each population set.



Master-Slave Model

- Master selects a population for each free slave in each generation.
- Slave evaluates the fitness and computes the best solution(s) for each population set.
- Slave performs mutation on each population subset and sends it to master for next generation.



Outline

Evaluation

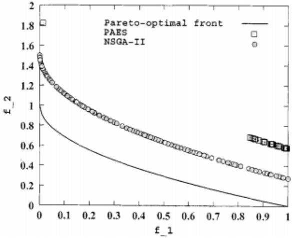
- **Runtime:** Time taken to run the algorithm. This can be measured using a profiler.

Evaluation

- **Runtime:** Time taken to run the algorithm. This can be measured using a profiler.
- **Speed-Up:** Serialized Runtime version of the algorithm over the Parallelized version of it.

Evaluation

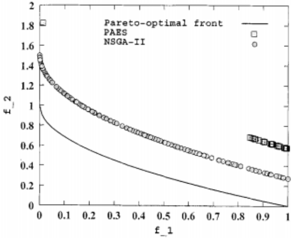
- **Runtime:** Time taken to run the algorithm. This can be measured using a profiler.
- **Speed-Up:** Serialized Runtime version of the algorithm over the Parallelized version of it.
- **Solution Quality:**



Evaluation

- **Runtime:** Time taken to run the algorithm. This can be measured using a profiler.
- **Speed-Up:** Serialized Runtime version of the algorithm over the Parallelized version of it.
- **Solution Quality:**

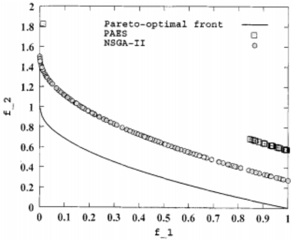
- **Convergence:**
  - Accuracy of the obtained solutions.
  - Represents the HyperVolume between the obtained solutions and Pareto Frontier



Evaluation

- **Runtime:** Time taken to run the algorithm. This can be measured using a profiler.
- **Speed-Up:** Serialized Runtime version of the algorithm over the Parallelized version of it.
- **Solution Quality:**

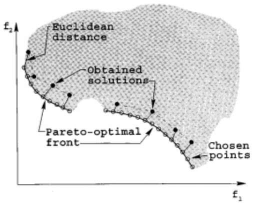
- **Convergence:**
  - Accuracy of the obtained solutions.
  - Represents the HyperVolume between the obtained solutions and Pareto Frontier
- **Diversity:**
  - Spread of the proposed solutions.
  - Ideally the solutions should be well distributed across the Pareto Frontier



Outline

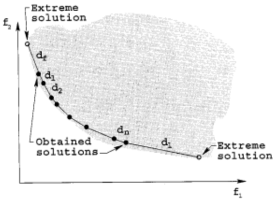
Measures

Convergence:



- Find a set of H optimal solutions.
- For each solution, compute the minimum eucledian distance from each of the solutions to a point on the Pareto Frontier.
- The average of these distances represent convergence.

Diversity:



- d_i is the distance between consecutive solutions.
- $\bar{d}$ is the mean of d_i
- d_f & d_l are distance between extreme and boundary solutions.

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_f + d_l + (N-1)\bar{d}}$$

Experimental Setup

- **Python**
  - Support for scientific computation: numpy, scipy, etc.
  - Quick prototyping and benchmarking.

Experimental Setup

- **Python**
  - Support for scientific computation: numpy, scipy, etc.
  - Quick prototyping and benchmarking.
- **Open-MPI**
  - Open Source Message Passing Interface with Python wrapper.
  - The Open MPI Project is actively developed and maintained by a consortium of academic, research, and industry partners.

Experimental Setup

- **Python**
  - Support for scientific computation: numpy, scipy, etc.
  - Quick prototyping and benchmarking.
- **Open-MPI**
  - Open Source Message Passing Interface with Python wrapper.
  - The Open MPI Project is actively developed and maintained by a consortium of academic, research, and industry partners.
- **Multi-Processing**
  - Offers local and remote concurrency.
  - Overrides python Global Interpreter Lock.

Experimental Setup

- **Python**
  - Support for scientific computation: numpy, scipy, etc.
  - Quick prototyping and benchmarking.
- **Open-MPI**
  - Open Source Message Passing Interface with Python wrapper.
  - The Open MPI Project is actively developed and maintained by a consortium of academic, research, and industry partners.
- **Multi-Processing**
  - Offers local and remote concurrency.
  - Overrides python Global Interpreter Lock.
- **HPC**
  - The henry2 shared memory linux cluster at NCSU.
  - Up to 16 shared memory processor cores and up to 128GB of memory accessible through a dedicated queue.

Outline

Computer Science NC STATE UNIVERSITY

DTLZ-2(Island Model)

Rank	Optimizer	Median	IQR	Quartile Chart
1	DE(Parallel)	2.30×10^{-5}	2.74×10^{-6}	
1	DE(Serial)	2.36×10^{-5}	3.04×10^{-6}	
2	GALE(Serial)	5.49×10^{-4}	8.32×10^{-6}	
2	GALE(Parallel)	5.54×10^{-4}	2.21×10^{-5}	

Figure: Convergence of serial & parallel DE & GALE

DTLZ-2(Island Model)

Rank	Optimizer	Median	IQR	Quartile Chart
1	DE(Parallel)	2.30×10^{-5}	2.74×10^{-6}	
1	DE(Serial)	2.36×10^{-5}	3.04×10^{-6}	
2	GALE(Serial)	5.49×10^{-4}	8.32×10^{-6}	
2	GALE(Parallel)	5.54×10^{-4}	2.21×10^{-5}	

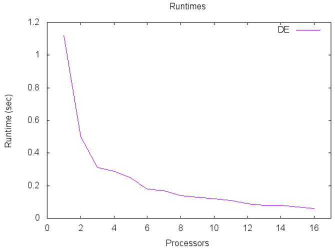
Figure: Convergence of serial & parallel DE & GALE

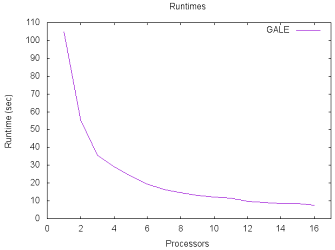
Rank	Optimizer	Median	IQR	Quartile Chart
1	GALE(Parallel)	0.416	0.070	
1	DE(Parallel)	0.417	0.049	
1	DE(Serial)	0.431	0.056	
1	GALE(Serial)	0.432	0.047	

Figure: Diversity of serial & parallel DE & GALE

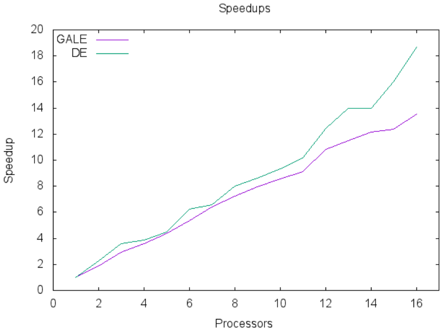
DTLZ-2(Island Model)

Runtimes:



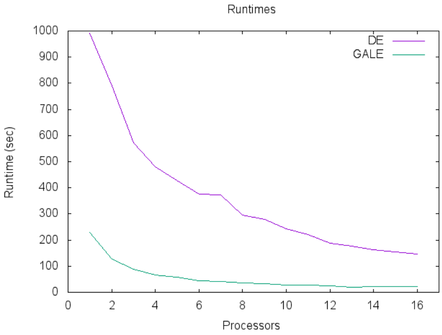


Speed Ups:

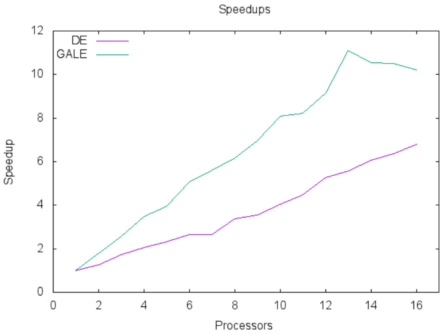


POM3(Island Model)

Runtimes:

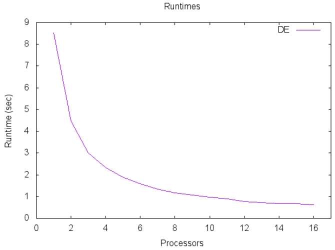


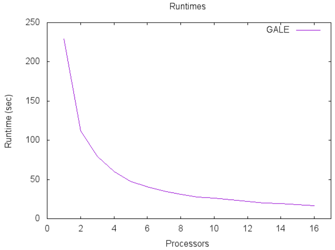
Speed Ups:



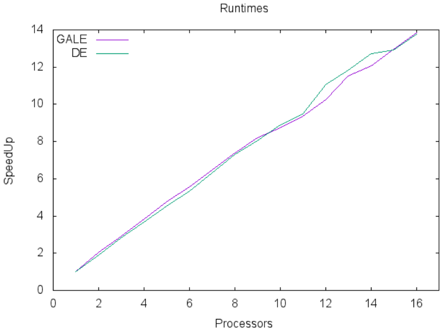
XOMO(Island Model)

Runtimes:





Speed Ups:

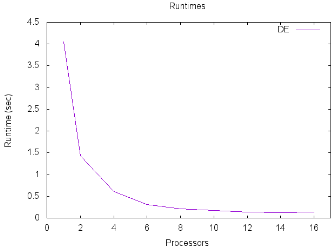


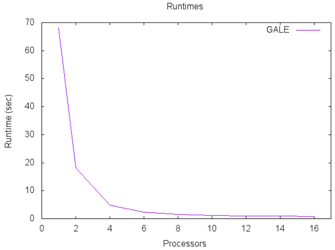
Outline

DTLZ-2(Master-Slave Model)

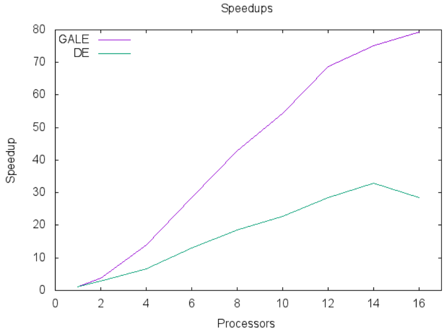
Computer Science NC STATE UNIVERSITY

Runtimes:





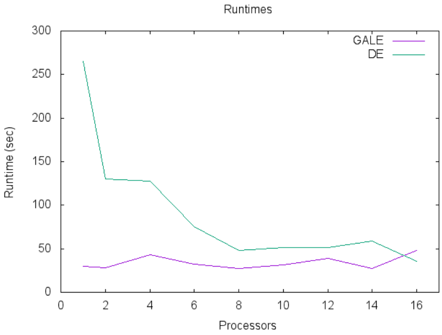
Speed Ups:



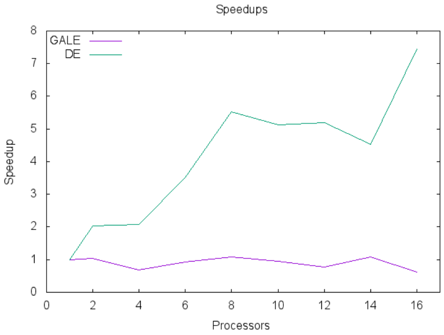
POM3(Master-Slave Model)

Computer Science
NC STATE UNIVERSITY

Runtimes:

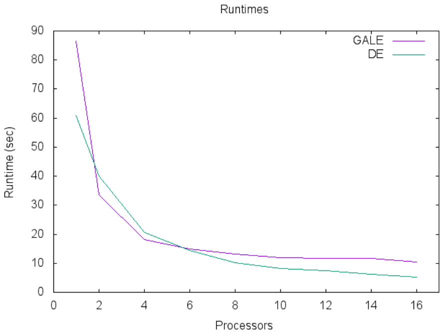


Speed Ups:

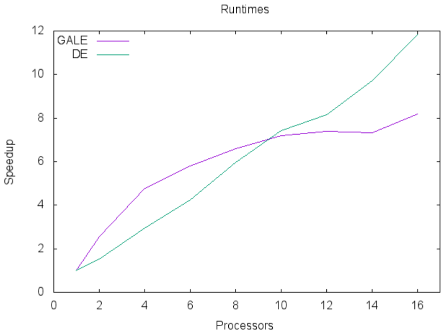


XOMO(Master-Slave Model)

Runtimes:



Speed Ups:



Outline

Feature Model

- A **feature model** is a compact representation of all the products of the Software Product Line in terms of "features". Feature models are visually represented by means of feature diagrams.

Feature Model

- A **feature model** is a compact representation of all the products of the Software Product Line in terms of "features". Feature models are visually represented by means of feature diagrams.
- Parent and Child features are categorized as
  - **Mandatory** child feature is required.
  - **Optional** child feature is optional.
  - **Or** at least one of the sub-features must be selected.
  - **Alternative** one of the sub-features must be selected.

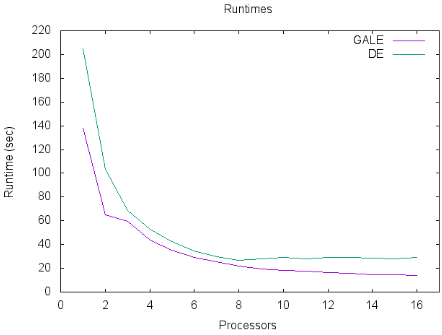
Feature Model

- A **feature model** is a compact representation of all the products of the Software Product Line in terms of "features". Feature models are visually represented by means of feature diagrams.
- Parent and Child features are categorized as
  - **Mandatory** child feature is required.
  - **Optional** child feature is optional.
  - **Or** at least one of the sub-features must be selected.
  - **Alternative** one of the sub-features must be selected.
- For our experiment we use the Emergency Response(ERS) feature model, which has **35 decisions** and **3 objectives**.

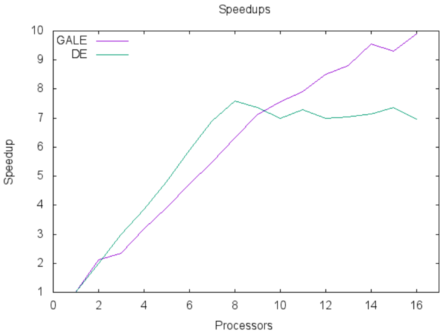
Outline

Emergency Response(Island Model)

Runtimes:



Speed Ups:



Outline

Background
- → Multi-Objective Problem

Models
- → DTLZ2
- → XOMO
- → POM3

Algorithms
- → Evolutionary Algorithm
- → Differential Evolution
- → GALE: Geometric Active Learner

Parallelization Strategies
- → The Island Model
- → Master-Slave Model

Evaluation Metrics
- → Evaluation
- → Measures

Experimental Setup

Results
- → Island Model
- → Master-Slave Model

Future Work
- → Feature Models
- → Results: Feature Models
- → Other Extensions:

Other Extensions:

- Mutation strategies for real world problems which are heavily constrained.

Other Extensions:

- Mutation strategies for real world problems which are heavily constrained.
- Extending these parallelization strategies for other Evolutionary algorithms like NSGA2, SPEA, IBEA etc

Other Extensions:

- Mutation strategies for real world problems which are heavily constrained.
- Extending these parallelization strategies for other Evolutionary algorithms like NSGA2, SPEA, IBEA etc
- Strategies for efficiently dividing the feature space for more efficient parallelization.

In Conclusion..

- Parallelization is effective for Multi Objective Evolutionary Algorithms.

In Conclusion..

- Parallelization is effective for Multi Objective Evolutionary Algorithms.
- Differential Evolution is preferred for Mathematical Problems.

In Conclusion..

- Parallelization is effective for Multi Objective Evolutionary Algorithms.
- Differential Evolution is preferred for Mathematical Problems.
- GALE is preferred for Real-World Problems where model evaluation is expensive.