Benchmarking of modcell-hpc for large scale problems

Research report

Sergio Garcia

October 11, 2019

Brief description of implementation

- The Matlab implementation of ModCell2 is slow and cannot handle large-scale problems due to both its inability to effectively parallelize beyond a few cores and its inefficient use of CPU and memory resources.
- modcell-hpc was written from the bottom up in C and uses MPI to implement island parallelization. In this approach multiple instances of the algorithm run in individual processes known as islands. These islands exchange individuals with certain periodicity in a process called migration.

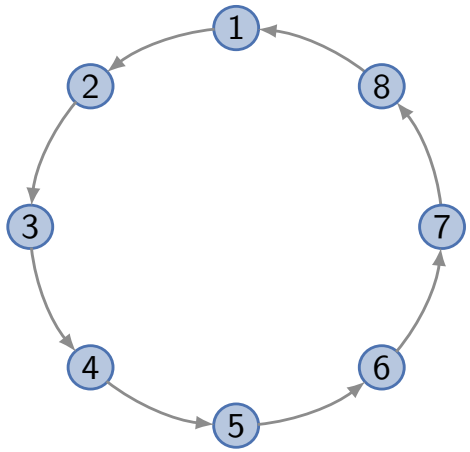


Figure: Islands with 8 processors in ring topology.

- Migration occurs asynchronously so islands may continue to evolve while messages are being passed, greatly enhancing performance.
- This approach can potentially scale to hundredths or thousands of cores since the program does not contain any sequential operations that would become a bottleneck (See Amdahl's Law).

Reproduce previous results

- Garcia and Trinh Metabolic Engineering 2019: *E. coli* model 20 products, wGCP-4-0. In a single run, 32 non-dominated solutions were found, including the target solution compatible with 17 products. In several runs, a total of 701 solutions were found, with 5 of them having a compatibility (threshold 0.6) of 17:

Solution index	Deletion ID
66	ACALD, PTAr, TKT2, LDH_D
202	ACALD, PTAr, GND, LDH_D
187	ACALD, TKT1, PTAr, LDH_D
125	ACALD, PGL, PTAr, LDH_D

Parameters and metrics

Parameters under consideration for stage 1 (20 products):

- Population size: Number of individuals per island.
- Migration type: 1) ReplaceBottom, top individuals are sent and bottom individuals are replaced. 2) Random, random individuals are sent and replaced.
- Migration interval: Number of generations between migration events.

Parameters under consideration for stage 2 ($>$ 100 products):

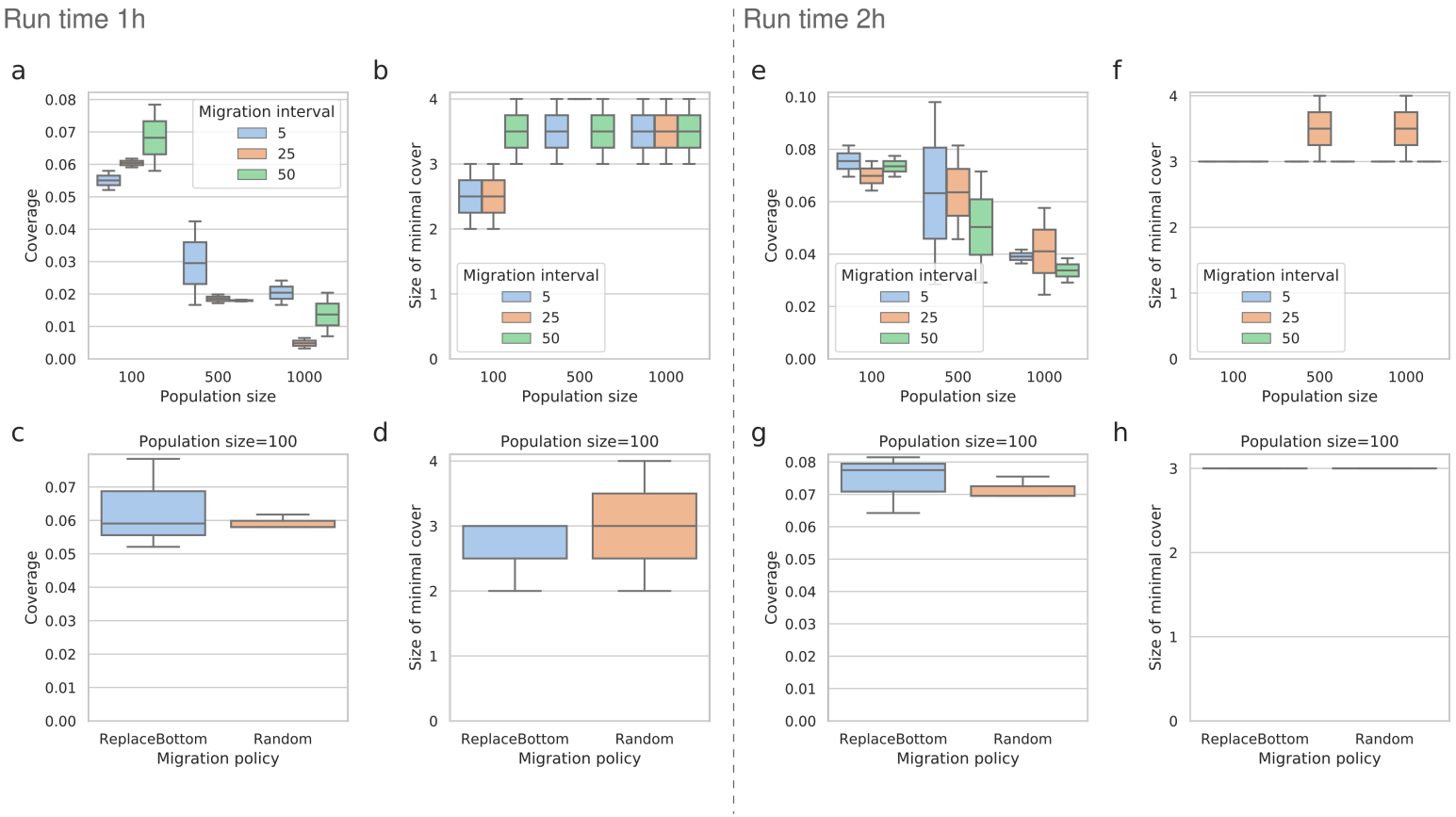
- Run time: Fixed run time for which the problem runs.
- Number of cores: Number of cores that corresponds to the number of islands

Metrics:

- Coverage: Fraction of points from all Pareto optimal solutions found captured by a given parameter configuration. Only unique Pareto points are taken and thus alternative solutions are not taken into account.
- Smallest number of designs that cover all compatible products. Note that if only a few products are compatible this can lead to smaller cover sizes, so the same number of products should be compatible to compare this value.

E. coli and 20 products: Results

These simulations were conducted on ACF with 16 processes each and design parameters $\alpha = 6, \beta = 1$.



E. coli and 20 products: Conclusions

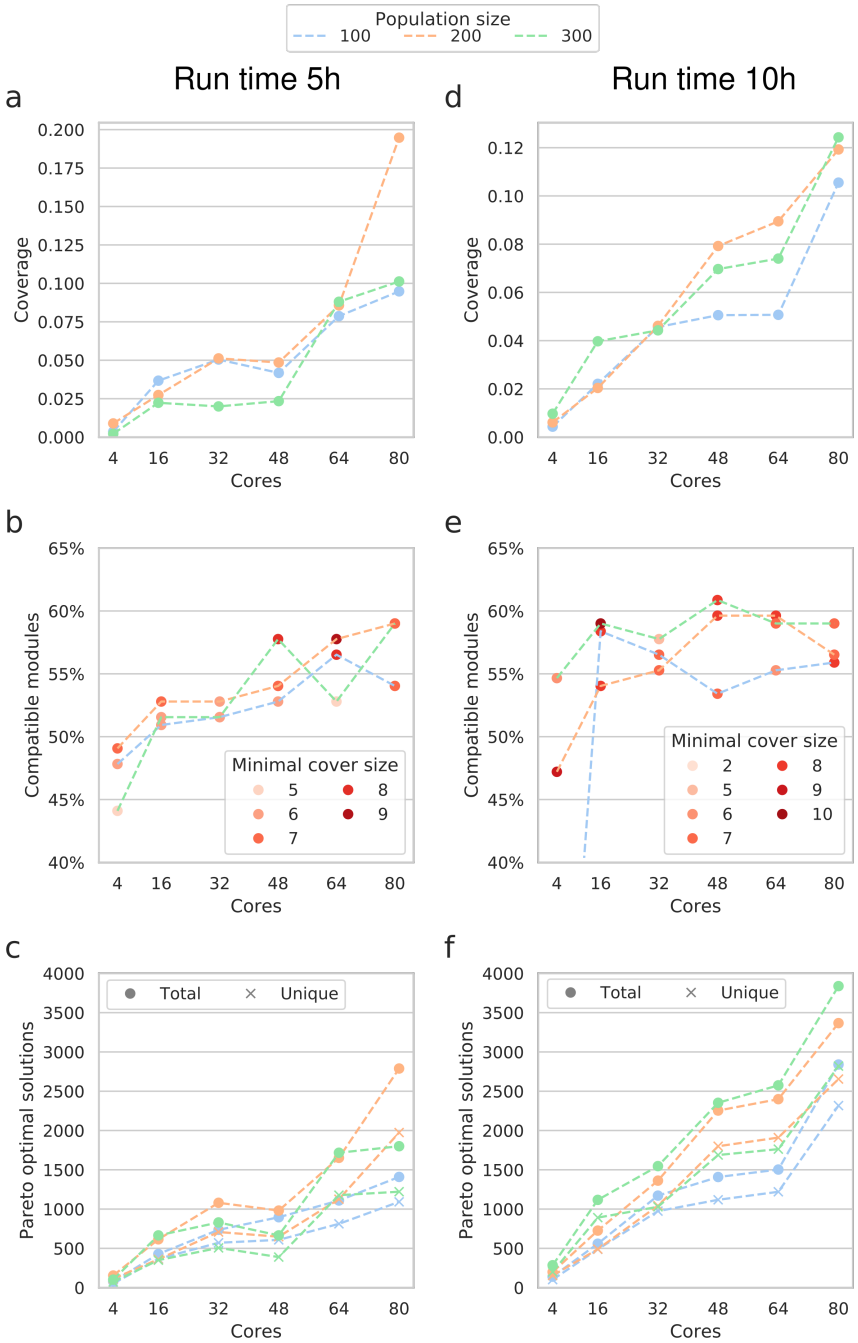
Findings:

- For limited run time, smaller populations go through more generations and achieve better results. Thus when run time is fixed, instead of generations as in the MOEA comparison paper, an optimal population size can be found.
- The effect of the migration frequency is tied to the number of generations, which is affected by the population size.
- The ReplaceBottom migration policy is better than random.

Conclusion:

- The ReplaceBottom migration policy will be selected.
- A migration frequency of 25 will be selected.
- Different population sizes will be tested for the larger problem to determine a good value.

E. coli and 161 products: Results



E. coli and 161 products: Conclusions

Findings:

- At 5h a population size of 200 is better in all metrics. While at 10h 200 and 300 are more similar. This indicates that the population size of 100 is too small, and that after a given number of generations populations sizes of 200 and 300 are comparable. So we will use 200 as the smallest valid population size per island.
- Increasing the number of cores leads to more solutions, but this does not mean those solutions are necessary better in terms of minimal covers. There is a trade-off in communication as the number of cores increases, which could affect overall convergence of the meta-population. This could be addressed with different topologies (e.g., hypercube) but that is currently beyond the scope of this study.
- In both cases around 48 cores reach a plateau in terms of compatible modules and cover size.

Conclusions:

- A population size of 200 is the minimum required for proper convergence and should be used under limited run times.
- 48 cores will be used for future studies.
- An increase in run time provides better solutions. Often the run time will be constrained by computational resources and project timeline. Problems with more objctives should use higher runtime since generations will take longer to complete.