Island Model Optimization Using Factored Evolutionary Algorithms with Dynamic and Static Migratory Rates for Solving Flexible Job Shop Scheduling Problems

Paul Reesman PAUL.REESMAN@SMARTSHEET.COM
Smartsheet, Inc.
10500 NE 8th St.
Suite 1300
Bellevue, WA 98004, USA

Editor: Hannah Bartlett

Abstract

Optimization problems are a largely studied area of computation where many similar algorithms have been proposed and offshoots from each have sparked a wide array of tools for solving these problems. Dividing these algorithms, while focusing on evolutionary algorithms, into two sets, we have those in the combative set and those in the cooperative set. Many popular algorithms like the traditional genetic algorithm or the island model take the combative approach. In the other set we have cooperative coevolutionary algorithms which includes factored evolutionary algorithms. There have been attempts in the past to merge some of these algorithms to find a balance between the combative nature of one algorithm and the cooperative nature of the other. We see in this paper that a merging of island models and factored evolutionary algorithms gives way to an effective algorithm to tackle optimization problems such as the job shop scheduling problem and its flexible extension. This optimization of the island model allows islands to optimize over a proper subset of the feature set yet formulate a feasible solution that quickly converges to a good solution.

Keywords: Machine Learning, Genetic Algorithms, Factored Evolutionary Algorithms, FEA, Island Model, Job Shop, Flexible Job Shop, FJSSP

1. Introduction

Within the topic of combinatorial optimization problems there is a widely studied instance known as the Job Shop Scheduling Problem (JSSP), which stems from various practical uses observed at facilities such as lumber mills and warehouses. Being NP-Complete, the class of job scheduling problems such as JSSP, including extensions like the Flexible Job Shop Scheduling Problem (FJSSP) which is studied in depth in the paper, are among the hardest optimization problems (Garey et al., 1976). Evolutionary Algorithms (EA), including Genetic Algorithms (GA), tend to be used for finding a "good enough" solution for optimization problems in general. One main reason behind this trend is because these algorithms do exceedingly well when trying to escape locally optimal points in order to explore as much of the search space as possible to find a better solution. In this paper we will use the global minima as the optimal solution since JSSP is a minimization optimization problem.

This paper proposes taking a popular style of genetic algorithms known as Island Models (IM) and inserting a cooperative co-evolutionary algorithm (CCEA) known as factored evolutionary algorithms (FEA), to allow data sharing while optimizing certain features. A background of GAs as a whole is reviewed along with IMs, followed by a review of FEA. Particle swarm optimization is reviewed in relevance to this problem as it is another popular algorithm used for this problem. This paper continues on with an explanation of how this IM-FEA algorithm works, along with its key configuration attributes. Afterwards we compare evaluations of the IM-FEA to those of IMs and a traditional GA in terms of fitness score and convergence rate.

With a proposed IM-FEA algorithm, we hypothesize that the introduction of FEA to IM will provide a faster convergence rate with a statistically insignificant difference in fitness values. This evaluation will be done using FJSSP as the testing scenario and compared to static and dynamic migration patterns. The sharing and competition techniques (later discussed) by the FEA provide solid ground for high global exploration towards more optimal solutions.

2. Background

2.1 Job Shop Scheduling Problem

The core concept of JSSP is finding an optimal schedule for processing an arbitrary amount of items on an arbitrary amount of machines. Each item has a single machine that it can be processed on along with the time it takes to complete the processing of the job – this time is referred to the cost. The set of jobs are organized into several sets of jobs known as tasks. Sometimes the meaning of terms "jobs" and "tasks" are flipped, however for the sake of this paper, a job will represent the atomic level unit being processed. Each job within the task must be completed in order, regardless of which machines they need to be processed on. Each machine can be used in parallel, as long as the next task scheduled on a machine does not have to wait for earlier jobs in the task to complete. If the next job scheduled on a machine has to wait before it can begin processing, then the entire machine stalls until the job is ready. To effectively schedule the jobs, there needs to be as little waiting as possible.

2.1.1 Flexible Job Shop Scheduling Problem

This paper focuses on a modern take of JSSP, by relaxing the constraint where each job can only run on a single machine. This makes the problem a little more flexible thus the problem studied here is the extension of JSSP known as FJSSP. Each job now has a set of machines attributed to it, each with their own cost where the goal is still to minimize overall cost (Wei and Qiaoyun, 2009). The jobs still need to be processed in order and once a process starts, it cannot be stopped until completion. This relaxation adds an amount of complexity to the problem as the search space increases for each machine added to a job. The goal is for a GA to be able to explore this widened search space and be able to still find a reasonable solution that minimizes the total cost of the problem. Many GAs have been used to solve FJSSP before and research has been conducted to integrate GAs with CCEAs (Xing et al., 2011) which gives reason to explore FEAs contribution to a common GA.

2.2 Evolutionary Computation

The term "evolutionary computation" encompasses a wide variety of stochastic search algorithms that are biologically inspired. They tend to focus on generational adaptation in order to optimize the genes of individuals in the population. The functions used by these algorithms to achieve this optimization are generally a combination of reproduction, mutation, recombination, and selection (Vikhar, 2016). One of the earliest evolutionary-inspired algorithms and one of the most popular is the genetic algorithm (GA). This uses all aforementioned functions to achieve its goal. There have been many offshoots of the traditional GA to help achieve better performance, whether it be a better selection process, faster convergence, higher exploration, etc. One of these algorithms is the Island Model (IM) which will be discussed in detail later in this paper. Another popular algorithm used falls into the category of swarm intelligence (SI) called the particle swarm optimization (PSO). The PSO is technically not an evolutionary-type algorithm (since there is not reproduction), but it is still widely considered a close relative and is often used for comparisons. It instead uses individual spatial positioning, along with global swarm positioning, to explore the search space rather than reproducing.

2.2.1 Genetic Algorithm

The GA is an iterative process in which the problem features are encoded into fixed length chromosomes (Jones) and injected into a static sized population. By use of genetic operators: reproduction, mutation, and crossover, the genetic pool of the population explores a wide range of the search space and uses the genetic operators to find an optima simultaneously.

GAs calculate a fitness score for each individual within the population as the selection criteria. An arbitrary amount of low performers are then selected to die off and be replaced, while the top performers are selected for reproduction. The reproduction process contains the steps: crossover and mutation.

The crossover operator takes members of the top performing set and randomly recombines their chromosomes to create a child that is a mixture of each parent. This allows the population to attempt to find optimal values for each attribute of the feature set. If the child takes on a poor crossover from the parents, the calculated fitness score will be low and might put the child in danger of being replaced in the next generation. This is highly possible with a "bumpy" search space in which there are many local minima and maxima. However the goal is that we create a child that performs better than either parent and becomes eligible for reproduction in the next generation. A well-formed crossover operator can increase the efficiency of the search as a whole (Sudholt, 2012). There are many different types of crossover operators, however popular ones include breaking up the chromosome into n different segments where n is the number of parents and assigning each parent to a segment to copy its genes over to. This operator tends to force convergence of the population.

In order to slow convergence and maintain a diverse population, mutation plays a crucial role. Each gene within the newly created child's chromosome has a random chance to mutate to any other value in that gene's value set. This allows the possibility of moving the search out of a local optima. This mutation chance is usually rather small, thus when a mutation

does occur, most of the other genes are untouched and might still represent a strong solution even if the spatial change was quite large.

2.2.2 Island Models

Island Models were not the first attempt to subdivide populations in optimization problems, but they have become a great tool in solving them. The ease of being able to parallelize the problem creates an efficient solution and generally finds better solutions faster than typical serial genetic algorithms (Whitley et al., 1998). If the problem is solved via a parallel architecture or distributed system, significant speed performance can be seen (Grosso, 1985). The concept is that each island represents an independent population for which evolution will occur. The selection process then compares the individuals of an island only to each other, wherein the most fit individuals will reproduce and the least fit will die off. Since the populations of the islands are less than that of the whole, assuming there is more than one island, the pressure on each individual to perform is increased. This increased pressure keeps each island as diverse as possible. Skolicki analyzed island models in evolutionary computation and compared strong and weak selection processes, "The choice of selection pressure has important implications on other parameters. Generally, it seems that the selection should not be too weak or too strong. Selection is needed to identify good offspring" (Skolicki, 2007). If we remove the assumption of having more than one island, systems with only one island can emerge, which then acts entirely like a traditional genetic algorithm - as there is no increased selection pressure on individuals compared to the entire population as a whole. In order to prevent stagnation of individual islands and to promote diversity within, individual members may from time to time migrate to other islands. This migration model helps islands share information as described by Whitley, "One reason for the improved search quality is that the various "islands" maintain some degree of independence and thus explore different regions of the search space while at the same time sharing information by means of migration. This can also be seen as a means of sustaining genetic diversity" (Whitley et al., 1998).

2.2.3 Dynamic Migration Island Models

Static migration policies for IMs tend to be circular in topology and do not reflect the overall "goodness" of the solution attempts of islands. Incorporating a dynamic migration policy should change the topology of the island graph, while also using island based attraction to encourage migration to good islands. This attraction attribute is the dynamic weight that is applied to the connections between islands, where each connection is uni-directional. That is, a connection from island A has an attraction weight of X to island B, where island B has an attraction weight of Y to island A. Duarte et al., take into account the population of an island along with any immigrant individuals that have recently joined the island, "the attractiveness of an island to another is measured by the ability of the neighboring island to improve its set of candidate solutions before the migration process. This information is defined in terms of its "native" population and its immigrants from the interested island" (Duarte et al., 2017). An improved version of this uses the island's ability to create better solutions as an input to the attractiveness score rather than just population size (Duarte et al., 2018), this allows islands to become less popular if the solutions that have recently

migrated to a popular island have drastically decreased the overall island's solution score (even if the island's population remains large). A known issue that can emerge from dynamic IMs is that a fully connected topology can again create an instance that behaves similarly to single population methods. Ishimizu and Tagawa took this notion of a fully connected topology and compared it to topologies with reduced edges, creating a ring or hypercube. Their findings showed that these other topologies can provide better results than a fully connected graph (Ishimizu and Tagawa, 2010).

2.2.4 Cooperative CoEvolutionary Algorithms

CCEAs, like other multi-population evolutionary algorithm methods, divide their population into disjoint sets and have each set cooperate with one another to establish a good solution. CCEAs are one of the earliest algorithms (in an EA sense) to subdivide the populations; Potter and Jong, in their research, created an algorithm called Cooperative CoEvolutionary Genetic Algorithm (CCGA) which seemed to outperform traditional genetic algorithms (Potter and De Jong, 1994). The primary concept is that each subdivision is treated as its own species and is only a partial solution to the whole. The method assigns variables from the variable set disjointly to the set of sub-populations, where each sub-population optimizes over the variables it was assigned. One issue with this approach is that it treats each variable naively as an independent variable. With this issue, CCGAs tend to perform poorly when there is significant dependence among the variables. One step to approach this hurdle is by introducing the concept of dynamic sub-populations where sub-populations can be created and removed depending on stagnation of the overall population (Potter and De Jong, 2000), this leads to the possibility of overlapping of the sub-populations thus removing the dependence on disjoint sets.

2.2.5 Factored Evolutionary Algorithms

One issue with IMs is that each island in the model represents a full solution to the problem and an issue known as hitchhiking can occur (Potter, 1997). Hitchhiking is the idea that subpar solutions that are associated with good schema will become established and hitchhike their way into much later iterations, thus lowering the overall fitness as described by Ochoa. "... the phenomenon of hitchhiking where unfavorable alleles may become established in the population following an early association with an instance of a highly fit schema" (Ochoa et al., 2008). Instead of having each sub-population compete with each other, another idea is to have them cooperate to achieve a more optimal solution. While the migration policy for IMs allow sharing of information between populations, they are not mutually shared. Instead, one island gives up a solution to another island. FEAs encourage overlapping, shared information between subsets of solutions. FEAs have three main functions to provide good solutions to problems: solving, competition, and sharing (Strasser et al., 2017). Working backwards, the sharing model allows overlapping populations to inject shared knowledge into one another, which gives each subpopulation a much more diverse set of solutions. The competition stage is the fitness evaluation of each subpopulation and that is what provides good solutions to share in the previous step. Lastly within the optimize step, FEAs optimize over their solutions and try to optimize over a set of variables (Strasser et al., 2017).

2.2.6 Particle Swarm Optimization

While crossover and mutation of top-performing individuals of a population might generate good solutions, PSO takes a different route. Instead of looking at phenotypical traits, PSO uses a more social approach to model the problem. (Eberhart and Kennedy, 1995). Each individual maintains a velocity attribute and attempts to move towards previously known best points, globally and locally. As the velocity determines the step size of each individual, PSO takes time to converge to a granular optimal solution (Jones). Although the global and local best points are conceptually similar to the crossover operation of the GAs (Eberhart and Kennedy, 1995), the velocity vs. mutation operations are what set this apart.

2.2.7 Overlapping Swarm Intelligence

FEAs have been used to create better performing versions of tradition algorithms in the past. Overlapping Swarm Intelligence (OSI) is an interesting method that combines FEAs and PSOs. OSI uses multiple swarms instead of the traditional single swarm, where each swarm represents a partial solution that optimizes an overlapping set of variables. "In traditional PSO algorithms, the position of each particle in a swarm encodes a solution to the associated optimization problem... the swarms overlap to account for the fact [that] local decisions are being made at each node, thus requiring each node to solve its own optimization problem. Thus, the position of each particle encodes a solution to a specific optimization problem from the perspective of the node at the center of the swarm" (Haberman and Sheppard, 2012). Here, each swarm has a particle at the center of the swarm and a particle for all of the swarm's immediate neighbors, which allows the center particle to make decisions based off of the overlap of its neighbors from the other particles in its own swarm. It is also noted that this approach has been extended to help learn both Bayesian and neural networks (Fortier et al., 2015) (Russell and Norvig, 2009).

3. IM-FEA

The base flow of the IM-FEA is similar to that of a traditional IM. A random initialization is done with a predetermined amount of islands, population per island, and starting migration rate. From there, the FEA takes over and begins optimizing over the features of the data set. Once complete, the islands then compete to make up a feasible global solution, similar to how traditional IMs take the best performing island as the global solution. Afterwards, the lowest performing individuals of each island share the global solution. From there the islands then migrate individuals. Once the end condition is met, the global solution is used for the end result.

3.1 Initialization

The world is given a predetermined amount of islands and each island is given a starting population amount. For each individual in each island, a random permutation, yet feasible, of the search space is created as the chromosome for the individual. With respect to FJSSP, the amount of machines that the problem instance has is the amount of chromosomes each individual has, while each gene is a job from the set of tasks. The islands are also given a

proper subset of machines to optimize over. As per the FEA, each island is encouraged to overlap some of its optimization domain with other islands.

3.2 Optimization

Every island runs their population through the GA fitness function. One additional parameter to this function is the set of machines that are being optimized over. For each machine being optimized, the total time of the jobs on the machine are used as its fitness score, thus each machine has its own score. Since not all machines are being optimized at once per individual, the summation has to be naive. Given that each machine independently is already feasible given the guarantee during the initialization phase, the job dependencies are ignored during this summation since the dependency may lie on a machine that is not being optimized over. This gives us an optimistic look at how well each machine is set. Each island then uses the best performing individual as its champion to compete for the global solution.

3.3 Competition

The output of the optimizations are pushed onto priority queues where each queue represents a single machine. From here we have an ordering of individuals based off of their fitness score for the given machine. Individuals do not compete for machines that they are not being optimized for. We then iterate through the set of priority queues and take the next job from the best individual in that queue whose job has not already been added to the global solution set and the job's dependency job, if any, has already been added. This allows us to create a global solution that remains feasible, while also prioritizing individuals that have a better fitness score. Once all jobs have been added to the global solution, the competition ends.

3.4 Share

The actual reproduction operator of the traditional IMs is changed slightly. The worst individual of each island copies the global solution. Since this solution was a combined effort throughout all of the islands, new information should be introduced at a high rate given enough islands to help diversify the islands that actually contributed to the global solution. The second worst performing individual is then re-initialized. This allows each island to both attempt to converge to the global solution and continue to explore the search space.

3.5 Migrate

We kept an open topology for the island migration paths in order to better generalize between static and dynamic migration instances. In each case, the islands are given an incoming migration rate n such that $n \in (0, 1)$ during initialization. We keep the rate from ever being 0 or 1 in order to always give an individual the chance of migrating or not migrating to the island.

3.5.1 Static Migration

Instances that were deemed to have a static migration were given an immutable incoming migration rate. This is the traditional migration model for IMs. This model produces a continuously even chance to migrate to any island independent on how well the given island is performing. This allows under-performing islands to regain a competitive edge.

3.5.2 Dynamic Migration

Dynamic migration IMs are a good performing alternative to the traditional migration model. Although this can increase island performance as a whole, it does put n amount of restrictions on the topology of the islands in order to perform well. In order for dynamic IMs to remain competitive against the static migration sibling, the island topology must be fully connected (Duarte et al., 2017). Here, if the island's average fitness score is above the average global fitness score for all of the islands, then the island's migration rate improves. If the average fitness is worse than the average global score, the migration rate lessens; it remains the same if it is equal to the average global fitness. This model allows better performing islands to have a more diverse population as individuals from other islands are drawn to it. Conversely, poorly performing islands have their population migrate to more prosperous islands, however this puts an increased pressure on these poorly performing islands to create individuals with better fitness scores.

4. Evaluation Using FJSSP

This problem as stated before is quite computationally expensive. We decided to implement IM-FEA using an efficient programming language to decrease the memory footprint of each algorithm and any superfluous computation added by language interpreters.Thus the programming language Rust was used. Multi-threading the implementation added yet another performance increase; one thread was run per trial instance which then spawned one thread per algorithm (5 were used, GA, static IM, dynamic IM, static IM-FEA, and dynamic IM-FEA). From here each algorithm spawned one thread per island during the fitness/optimization stage. This parallelization was made possible by running the tests on a single c5.9xlarge EC2 instance on AWS which provides 36 virtual CPUs and 72 GiB of memory.

Parameter	GA	Static IM	Dynamic IM	Static IM-FEA	Dynamic IM-FEA
# of Islands	1	1...10	1...10	1...10	1...10
Pop/Island	*	1...20	1...20	1...20	1...20
Migration Rate	N/A	0.05...0.25	0.05...0.25	0.05...0.25	0.05...0.25

Table 1: Initial configurations for each algorithm during each run.

The number of islands increment by 1 while population per island and migration rate step by 5 and 0.05 respectfully. The GA doesn't have a population per island since there technically aren't any islands, thus the GA population is equivalent to the rest of the algorithms.

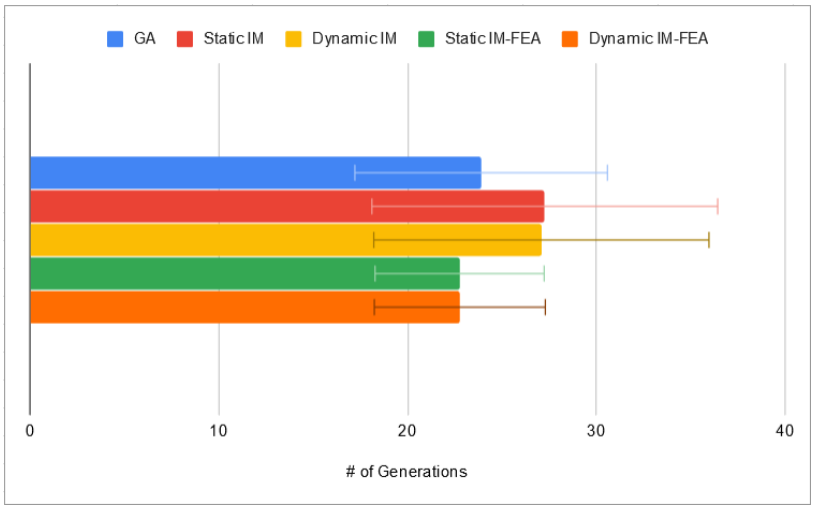

Figure 1: Rate of convergence of each algorithm

Each algorithm was run against 196,000 test instances of FJSSP. The stopping criteria for each instance was a moving average of delta changes from generation to generation less than 10 with a window of 15 generations. The window was prepolulated with values at 100. This assured, since the cost returned by the fitness function could only be positive, that a minimum of 15 generations would have to pass before convergence was allowed. We started off with a minimum generation hard limit along with the moving average but quickly found that all test instances of each algorithm were converging at the minimum generation amount. Scaling backwards from 2,000 generations we found that they were converging at a much smaller generation amount. Removing the hard limit for the generations and only following the moving window average, instances were converging above the minimum set time rather than at the time itself.

The above figure shows the average number of generations it took for each algorithm to converge. As we can see, the average case for each algorithm was under 30 generations. Both instances of IMs converged at about the same time with dynamic IM fairing a bit better. Similarly, the IM-FEAs converged at similar rates on average. Running a Welch's two tailed t-test between static IM and static IM-FEA and then again for dynamic IM and dynamic IM-FEA, we gather a p-value in both cases such that $p < 0.001$ therefore we reject the null hypothesis and show that indeed, IM-FEA converges quicker than traditional IM algorithms. Again, we test the results against the traditional GA and we can show with the same p-value that IM-FEA converges at a statistically significant lower rate.

While quick convergence was the goal in mind, a constraint was that the fitness scores were not significantly affected by this quicker convergence. This comparison was much tougher since the optimal values from each test instance varied widely.

After normalizing the fitness scores to an appropriate scale we were then able to compare the algorithms. Surprisingly, IM-FEA did better than traditional IMs in both static and dynamic trials as well. With a $p - value < 0.001$ there is significant evidence to reject the null hypothesis. The IM-FEA seemed to have done better than traditional IMs on average when it came to convergence and fitness scores.

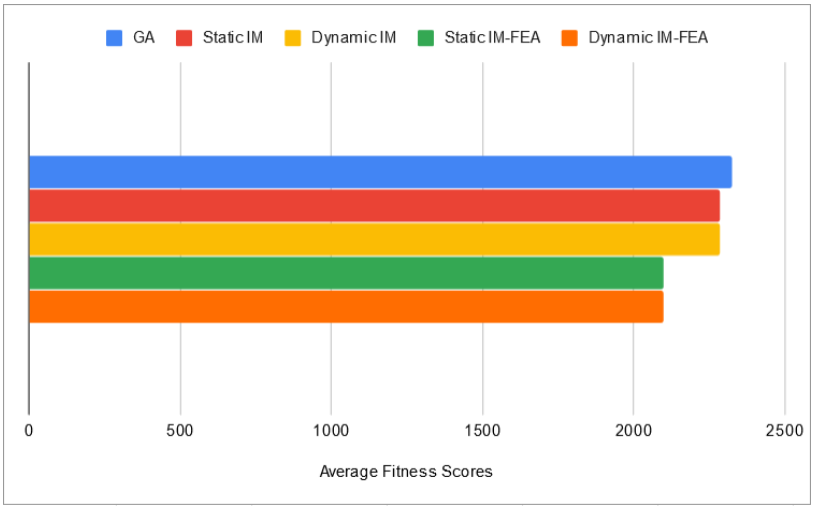

Figure 2: Average normalized fitness scores

Looking at the raw test results, these results seem consistent throughout the test instances. There is a trend that shows for both IMs and IM-FEAs that the fitness and convergence do better between the values of 5 and 7. With the population of each island configured to test between 5 and 20, instances with larger starting populations tended to create better fitness results while lower populations tended to converge quicker. Migration rate seemed to have a large impact on convergence speed. Lower values converged quicker while larger values had trouble stabilizing.

5. Discussion

The key configurations of IM-FEA had profound impact on the results when comparing to the other algorithms. Optimal parameters tended to be optimal across the board. The change in migration rate during instances for the dynamic algorithms was constant. For every generation an island's average fitness score was above the global average then the migration rate for that island would increment by a percent. If the fitness score was less than the global average, it would decrement by a percent. Admittedly, this was a naive approach but the focus was less on the optimality of the dynamic migration and more on the comparison between two algorithms using a dynamic migration model.

The ability to guarantee a feasible solution at the global level after island-machine optimization gave the sharing mechanism a jump start. The lower performing individuals were able to quickly start producing good fitness scores after the FEA sharing.

The FEA competition phase was probably the most important phase for overall global health. Using a priority list for each machine based on individual fitness while maintaining a feasible solution allowed partially global optimal solutions to merge into a semi-optimal global solution. After several generations, this became the highest fitness individual of any island.

There are several outstanding issues that should be addressed for future work. First and foremost the ability for the global solution to function with an infeasible solution

should be looked into. This would require the IM-FEA to put a constraint on convergence that in addition to the moving window average, the global solution must also be feasible. Furthermore, calculating a reasonable fitness score for an infeasible solution should be found. This paper assumed a fully connected topology to meet the requirement of the dynamic IM, looking into how IM-FEA fares with different topologies including the popular circular topology. Lastly, expanding the IM-FEA to test against different domains outside the realm of FJSSP would be beneficial to the overall understanding of the "fitness" of IM-FEA itself.

6. Conclusion

Within the large world of evolutionary computation, one of the most popular methods is in the form of genetic algorithms. These simple yet effective algorithms have been used time and time again and have been built upon to provide better results in optimization problems. Certain popular variations of this method take the form of subdividing the whole population into subsets and using this as a way to combat early convergence and encourage high diversity. These subdivisions can be separated into either combative or cooperative subpopulations. IMs take the combative from as each island vies to have the single highest fitness score. CCEAs on the other hand take to cooperative approach and together with individual sharing mechanisms, they can reach high fitness values. Each have their merits. In this paper we proposed merging one algorithm from each set and testing it on FJSSP. We saw that the combative nature followed by the mandatory sharing of the global solution was beneficial for the overall health of the world.

References

Dennis Behnke and Martin Josef Geiger. Test instances for the flexible job shop scheduling problem with work centers.

Paolo Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations research*, 41(3):157–183, 1993.

Stéphane Dauzère-Pérès and Jan Paulli. An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Annals of Operations Research*, 70:281–306, 1997.

G. Duarte, A. Lemonge, and L. Goliatt. A dynamic migration policy to the island model. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 1135–1142, June 2017. doi: 10.1109/CEC.2017.7969434.

G. Duarte, A. Lemonge, and L. Goliatt. A new strategy to evaluate the attractiveness in a dynamic island model. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, July 2018. doi: 10.1109/CEC.2018.8477706.

Russell Eberhart and James Kennedy. Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks*, volume 4, pages 1942–1948. Citeseer, 1995.

Nathan Fortier, John Sheppard, and Shane Strasser. Abductive inference in bayesian networks using distributed overlapping swarm intelligence. *Soft Computing*, 19(4):981–1001, Apr 2015. ISSN 1433-7479. doi: 10.1007/s00500-014-1310-0. URL `https://doi.org/10.1007/s00500-014-1310-0`.

M. R. Garey, D. S. Johnson, and Ravi Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2):117–129, 1976. ISSN 0364765X, 15265471. URL `http://www.jstor.org/stable/3689278`.

Paul Bryant Grosso. *Computer Simulations of Genetic Adaptation: Parallel Subcomponent Interaction in a Multilocus Model*. PhD thesis, Ann Arbor, MI, USA, 1985. AAI8520908.

Brian K. Haberman and John W. Sheppard. Overlapping particle swarms for energy-efficient routing in sensor networks. *Wireless Networks*, 18(4):351–363, May 2012. ISSN 1572-8196. doi: 10.1007/s11276-011-0404-1. URL `https://doi.org/10.1007/s11276-011-0404-1`.

Johann Hurink, Bernd Jurisch, and Monika Thole. Tabu search for the job-shop scheduling problem with multi-purpose machines. *Operations-Research-Spektrum*, 15(4):205–215, 1994.

T. Ishimizu and Kiyoharu Tagawa. A structured differential evolution for various network topologies. *Int. J. Comput. Commun.*, 4:2–8, 01 2010.

Karl O Jones. Comparison of genetic algorithm and particle swarm optimization.

Gabriela Ochoa, Evelyne Lutton, and Edmund Burke. The cooperative royal road: Avoiding hitchhiking. In Nicolas Monmarché, El-Ghazali Talbi, Pierre Collet, Marc Schoenauer, and Evelyne Lutton, editors, *Artificial Evolution*, pages 184–195, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-79305-2.

Mitchell A. Potter. *The Design and Analysis of a Computational Model of Cooperative Coevolution*. PhD thesis, Fairfax, VA, USA, 1997. UMI Order No. GAX97-28573.

Mitchell A. Potter and Kenneth A. De Jong. A cooperative coevolutionary approach to function optimization. In Yuval Davidor, Hans-Paul Schwefel, and Reinhard Männer, editors, *Parallel Problem Solving from Nature — PPSN III*, pages 249–257, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg. ISBN 978-3-540-49001-2.

Mitchell A. Potter and Kenneth A. De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evol. Comput.*, 8(1):1–29, March 2000. ISSN 1063-6560. doi: 10.1162/106365600568086. URL `http://dx.doi.org/10.1162/106365600568086`.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009. ISBN 0136042597, 9780136042594.

Zbigniew Skolicki. *An analysis of island models in evolutionary computation*. PhD thesis, 2007.

S. Strasser, J. Sheppard, N. Fortier, and R. Goodman. Factored evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 21(2):281–293, April 2017. doi: 10.1109/TEVC.2016.2601922.

Dirk Sudholt. Crossover speeds up building-block assembly. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 689–702. ACM, 2012.

P. A. Vikhar. Evolutionary algorithms: A critical review and its future prospects. In *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, pages 261–265, Dec 2016. doi: 10.1109/ICGTSPICC.2016.7955308.

Q. Wei and L. Qiaoyun. Solving the flexible job shop scheduling problems based on the adaptive genetic algorithm. In *2009 International Forum on Computer Science-Technology and Applications*, volume 1, pages 97–100, Dec 2009. doi: 10.1109/IFCSTA.2009.30.

Darrell Whitley, Soraya Rana, and Robert B Heckendorn. The island model genetic algorithm: On separability, population size and convergence. 1998.

Li-Ning Xing, Ying-Wu Chen, and Ke-Wei Yang. Multi-population interactive coevolutionary algorithm for flexible job shop scheduling problems. *Computational Optimization and Applications*, 48(1):139–155, Jan 2011. ISSN 1573-2894. doi: 10.1007/s10589-009-9244-7. URL `https://doi.org/10.1007/s10589-009-9244-7`.