Hybrid evolutionary algorithm-based real-world flexible job shop scheduling problem: application service provide approach `http://www.sciencedirect.com/science/article/pii/S1568494604000602`

Problem: The problem presented in the paper involves the development of a system that can produce optimized schedules for a set of manufacturing factories. Specifically, how a set of customer orders should be mapped to a set of factories of plastic injection machines (FPIM) in an optimal fashion. This means that a minimum amount of time should be spent on producing the ordered goods while also minimizing the amount of resources spent (e.g. how many times the mold is changed) to complete the set of orders. Furthermore, the schedule must not violate any hard constraints, e.g. not assigning the same mold to multiple machines at the same time. This is an instance of the classical NP-Hard problem (Wikipedia) of optimizing a Job Shop Schedule (JSSP) in that how should a set of orders/jobs be assigned to a limited resource at a particular time while still minimizing the total time (aka the makespan) taken to complete the required number of operations, i.e. the entire schedule.

Representation: To solve this problem the paper compares two approaches of indirect-representation, both however represents a set of schedule building instructions, meaning that each chromosome gene is an integer allele that is interpreted by the schedule builder to build a schedule.

In the first approach, each of these numbers is used to indicate which of the available Priority Dispatching Rules (PDR)s should be used to decide which order of the current unscheduled orders should be assigned to a specific machine when it becomes available. This means each allele of each gene can lie in the range 0 to the number of available PDRs which in this case was 9, this means that the search space is $PRD's^orders$ for the PDR chromosome. The genetic algorithm (GA) then seeks to search for the chromosome that features the best combination of PDRs which maps into the best/optimum schedule/phenotype. The length of the chromosome is defined by the number of orders submitted to the factory. Because the aim is to find the best combination PDRs it means that the search space is limited to the number of PDRs raised to the power of the number of orders.

In the second approach the chromosome is a list made up where each gene position represents that orders position in the order queue. The scheduler then uses these to determine which order should be scheduled next on the soon to be ready machine.

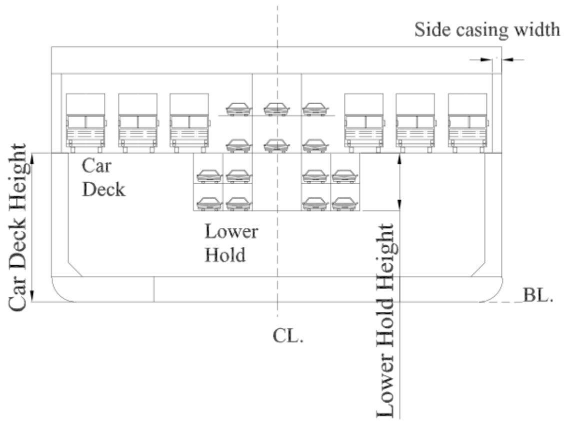
Fitness: In terms of evaluation the same function is used for both types of representation. It is responsible for two things, this includes translating the indirect chromosome into an actual schedule, secondly evaluating the fitness of this schedule with the fitness function. The fitness of a schedule is defined by several attributes. This includes the longest and shortest flows inside the schedule. The amount of times the mold was required to change in the proposed schedule. The time it takes to setup and complete every order in the schedule and the total execution time of every scheduled orders. In order to penalize any schedule that violates a defined constraint an extra penalty value Q is added to the calculated fitness. The objective of GA is to minimize the value returned by the fitness function. This combined with the penalty added for a constraint violation.

Summary: The paper investigated whether a proposed scheduling system which utilizes a combination of PDRs and a GA to schedule orders to a set of real world plastic injection machines would within a set of factories. Futhremore it compared the convergence of the individuals and the best fitness achived of the two types of representation. Using simple graph comparison they find that the GA with PDR is better in that it finds desirable schedules in a lower amount of generations. It should be noted though that this experiment was conducted in 2004 on a 450 MHz CPU, meaning that would this be run today on modern machine it would be assumed that the time required would be substaintially less. Also it might be worthwhile to look into a group based indirect representation scheme instead of a meere interger based string to compare against the PDR version.

A Reinforcement Learning Based Hybrid Evolutionary Algorithm for Ship Stability Design
`http://link.springer.com/chapter/10.1007/978-3-642-23424-8_9`

Problem: Designing a ship is a compromise between the ships safety/stability and the economic benefits it will return in the form of transporting goods. The more it can carry the more economic benefits will be returned. The is a multi-objective problem in that coming up with a design that both maximizes the ships safety/stability and the amount of space available for transporting goods/items. The paper investigates whether a framework using a hybrid EA for producing a good internal hull design can be developed. The task is to find the optimal configuration of 16 design variables (3 of which can be seen in Figure 1) and 3 major objectives. As an example they produce a design for a ROPAX ship where they compare the designs made by 2 variations of the NSGA-II algorithm and the original design.

Figure 1: Car Deck height, Lower Hold height and Side Casing width



Both approaches seek to find a good compromise solution in this multi objective problem. They do this by looking for an optimal configuration of the objectives and decision variables. However, these are often being in conflict with each other, meaning that minimizing one variable would be often result in another variable increasing in value. What this means is that becomes the task of the algorithm to find a balanced solution along the Pareto/multi-objective optimal solutions for the 16 design variables.

Representation: The exact chromosome encoding is not described in detail, however, the 3 objectives are defined as (Index A, limitingKG and the Cargo Capacity) which are all continuous values which are not calculated by the EA. The values that the EA operates on are the 16 design variables which includes the location of 13 transverse bulkheads and the 3 (continuous) variables mentioned in Figure 1. For each of these attributes there is a upper and lower bound value due to real world constraints. A possible chromosome representation for a solution could be to use real value strings to support both the continuous and discrete values.

Fitness: However what they both have in common is that they both uses a piece of software called NAPA to simulate and evaluate the overall ship design. Furthermore the software also calculates the value for the 3 objective values, this means that after the EA has tried to optimize the 16 design variables it passed to the simulation and awaitsa response. To optimize the 16 design variables the proposed hybrid version uses Q-learning as a form of reinforcement learning/evaluation where the algorithm can compare the Q values of a set of possible move actions. In practice it means that they use Q-learning to select which child should be selected for next generation of the population. This means that a child is evaluated to be the best if it has the biggest/maximum Q value(pushes the current Pareto front the furthest), meaning that the Q value describes the value gained from taking a particular step in the search space from a parent to a child position. Which in this case means a set of design variables that produces a better design.

Summary: The paper presents how they used a hybrid NSGA-II that combined machine learning and the NSGA-II algorithm to solve the real-world task of designing the internal hold of a ROPAX vessel. The proposed algorithm delivered an improved design in terms of balancing the multi objective variables of maximizing ship stability, damage stability while also maximizing the amount of cargo space for the items the ship is going to transport. e.g. it meant that the ship could increase the number of car lanes from the original 8 to 14 due to an increase in the height of the lower hold. This design improvement This was achieved while cutting around 50% of the execution time of the NSGA-II algorithm by utilizing Q-learning as a reinforcement technique to explore the Pareto front of the 16 design variables which included the 3 displayed in Figure 1.