Improving Genetic Algorithms by Search Space Reductions (with Applications to Flow Shop Scheduling)

Stephen Chen
The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
chens@ri.cmu.edu
http://www.cs.cmu.edu/~chens

Stephen F. Smith
The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
sfs@ri.cmu.edu
http://www.cs.cmu.edu/~sfs

Abstract

Crossover operators that preserve common components can also preserve representation level constraints. Consequently, these constraints can be used to beneficially reduce the search space. For example, in flow shop scheduling problems with order-based objectives (e.g. tardiness costs and earliness costs), search space reductions have been implemented with precedence constraints. Experiments show that these (heuristically added) constraints can significantly improve the performance of Precedence Preserving Crossover--an operator which preserves common (order-based) schemata. Conversely, the performance of Uniform Order-Based Crossover (the best traditional sequencing operator) improves less--it is based on combination. Overall, the results suggest that conditions exist where Precedence Preserving Crossover should be the best performing genetic sequencing operator.

1 INTRODUCTION

Due to their lower development cost, it is appealing to use domain independent search techniques (e.g. genetic algorithms) rather than knowledge intensive approaches. For certain factory scheduling domains, it has been shown that genetic algorithms (GAs) can also deliver the best performance [RHW96][WHR98]. In these studies, schedules (sequences) were evaluated by simulation.

When evaluation is by simulation, only global payoff information may be available (e.g. total cost). Despite not being able to directly assign credit, it can still be inferred that a good solution has good sub-components (schemata). Fitness-based selection over a population of solutions increases the proportion of fit schemata. Crossover is then used to recombine these schemata into new solutions [Hol75][Gol89].

For standard representations, the standard crossover operators always preserve common schemata. However, many early crossover operators designed for sequence representations did not (e.g. Order Crossover [Dav85] and Uniform Order-Based Crossover [Sys91]). The commonality hypothesis suggests that schemata common to above-average solutions are above average [CS98][CS99], so they should be preserved. Several design models which require common schemata to be preserved during crossover have been proposed [Rad91][EMS96][CS98].

For operators that preserve common schemata, not only are the (hypothesized) above-average schemata of the parents preserved, but a second beneficial effect can occur--constraints may also be preserved. For example, if a precedence constraint requires job `i` to be processed before job `j`, then two feasible parents will both process job `i` before job `j`. A crossover operator that preserves common (order-based) schemata will also preserve this precedence constraint.

The objectives of most scheduling problems are based on two relationships: job adjacency (if job `i` *immediately* precedes job `j`, then job `i` and job `j` are adjacent) and job order. For example, to minimize makespan, it is necessary to minimize (sequence-dependent) set-up times. Typically, a job's set-up time is only dependent on it's immediate predecessor. Thus, makespan is an adjacency-based objective. Conversely, tardiness and earliness are order-based objectives. The cost of a job depends more on its relative order in the sequence, than on its immediate neighbors.

It is important that the solution representation matches a problem's constraints and objectives. For example, the Sequential Ordering Problem (SOP) is a Hamiltonian path problem with additional precedence constraints [Esc88]. These (order-based) constraints are difficult to observe in adjacency-based representations. Thus, adjacency-based methods (e.g. branch and cut [Asc96]) can require complex modifications.

Order-based methods have an advantage on the SOP. These methods can transparently preserve precedence constraints. In fact, these methods appear to perform better with the addition of precedence constraints (which cause order-based search spaces to be reduced [CS98]). Inspired by this observation, we attempt to improve the performance of genetic sequencing operators by developing beneficial (precedence) constraints for unconstrained flow shop problems with order-based objectives.

A reduced search space may not contain the optimal solution. However, it is a standard textbook proposition that sampling non-delay schedules will be more productive than sampling active schedules, even though the optimal schedule may be active, but not non-delay [CMM67][Fre82]. Similarly, for the allotted time, it may be possible to find a better solution in a reduced search space, even if the optimal solution is outside the search space.

To examine if heuristically generated precedence constraints can create beneficial search space reductions, a series of random flow shop problems has been generated. These problems have sequence-dependent set-up times and order-based cost objectives. Specifically, tardiness and earliness penalties are imposed. For these objectives, (adjacency-based) methods designed to minimize makespan are not relevant. Overall, these general features may better characterize typical manufacturing environments. They also describe a problem domain where order-based genetic operators are likely to provide the best optimization technique.

The remainder of this paper is presented as follows. First, genetic sequencing operators are reviewed in section 2. In section 3, the flow shop problems are presented. In section 4, the results for unconstrained search are presented. In section 5, the precedence constraints used to reduce the search space are developed, and the results with search space reductions are presented in section 6. The results are discussed in section 7, and final conclusions are summarized in section 8.

2 SEQUENCING OPERATORS

Many genetic sequencing operators have been developed. Some operators (e.g. Order Crossover [Dav85]) are for Hamiltonian cycle problems like the Traveling Salesman Problem (TSP). These operators "wrap-around", so they are not relevant to flow shop problems. Other operators (e.g. Edge Recombination [SMM91]) are adjacency-based, so they are ill-suited for order-based objectives. Overall, it has been suggested that Uniform Order-Based Crossover (UX) [Sys91] is the best operator for scheduling problems [SWM92][RHW96][WHR98].

Uniform Order-Based Crossover combines random (order-based) schemata taken from the parents, but it does not guarantee that schemata common to both parents are preserved. Thus, UX may undo precedence constraints (and any benefits they may provide). Conversely, there exists an order-based operator--Precedence Preservative Crossover (PPX) [BMK96]--that maintains all common precedence relationships. For unconstrained search, PPX is less effective than UX. However, PPX will preserve all precedence constraints, and thus (compared to UX) may better explore a reduced search space. The UX and PPX operators are hereby reviewed to highlight their effects on (common) order/precedence schemata.

2.1 UNIFORM ORDER-BASED CROSSOVER

The processing order for jobs in a flow shop can be represented as a permutation (sequence) solution. On this sequence, Uniform Order-Based Crossover uses a uniform crossover mask to select jobs. It takes the jobs of Parent 1 at the sites where the mask has a `1` and places them in the offspring at the same sites. The remaining jobs are filled

```
Parent 1:   b a c e f d h j k i g l
Parent 2:   a b c d e f g h i j k l
mask:       0 1 1 0 1 0 1 0 1 0 0 1

taken:        a c   f   h   k     l
remaining:  b     e   d   j   i g
order:        b   d e   g   i j

Offspring:  b a c d f e h g k i j l
```

Figure 1: Example of UX. Common order for jobs `e` and `f` is not transferred by a single parent. Thus, their order may be reversed in the offspring.

Parent 1:	b	a	c	**e**	**f**	d
Parent 2:	a	b	c	d	**e**	**f**
mask:	0	1	1	0	1	0
From 1:	b					
From 2:	a					
From 2:	~~a~~	~~b~~	c			
From 1:	~~b~~	~~a~~	~~c~~	**e**		
From 2:	~~a~~	~~b~~	~~c~~	d		
From 1:	~~b~~	~~a~~	~~c~~	~~e~~	**f**	
Offspring:	b	a	c	**e**	d	**f**

Figure 2: Example of PPX. Job `e` will always be drawn before job `f`.

into the empty sites in the order they appear in Parent 2. (See Figure 1.) Overall, UX combines order (and position) information from Parent 1 with order information from Parent 2. However, if a (common) order relationship is not taken from a single parent, it is possible for it to be reversed during the combination process.

2.2 PRECEDENCE PRESERVATIVE CROSSOVER

On a sequence representation, Precedence Preservative Crossover uses a uniform crossover mask to select the parent from which the next job is drawn. The selected parent is scanned for the first job that has not yet been drawn. This job is appended to the offspring. (See Figure 2.) This process of "drawing" jobs from the parents guarantees that all common precedence relations are preserved, and that all precedence relations in the offspring come from one of the parents. Subsequently, PPX "transparently" enforces precedence constraints.

3 FLOW SHOP PROBLEMS

A series of random two-machine flow shop problems has been generated. For these problems, 500 independent jobs must be processed in the same sequence on both machines. The jobs have a processing requirement on each machine, a sequence-dependent set-up time (based on their immediate predecessor), a transfer time between machines, a due date, a tardiness cost weight, and an earliness cost weight.

The job parameters have been generated randomly from uniform distributions. The ranges are 25-100 (time units) for processing times, 5-70 for set-up times, and 5-50 for transfer times. The tardiness weights have a range of 2-20, and the earliness weights have a range of 1-5. The due dates are uniformly distributed from time 0 to the expected average makespan (50,000).

Five problem instances have been generated. Each instance consists of five 500-element vectors and a 500x500 full matrix for set-up times. To evaluate a sequence, the processing of a non-delay schedule through the two-machine factory is simulated. For this simulation, it is assumed that there are no set-up times required for the first job. Tardiness and earliness costs are calculated based on each job's completion time on machine 2.

4 INITIAL RESULTS

The UX and PPX operators have been implemented in GENITOR [WS90]. The parameters were set to 1000 for the population size, 200,000 for the number of trials, and 2.00 for the selection bias (2-tournament selection). Ten runs were conducted for each of the five problems. The results are normalized against the cost of the Earliest Due Date (EDD) dispatch sequence[1]. (See Table 1.)

For the allotted time, neither operator provides an advantage over EDD. Similar to results in [RHW96], randomly initialized genetic search does not provide a significant advantage over dispatch techniques. However, the results with PPX are about 10 times worse than the results with UX. The commonality hypothesis suggests that schemata common to above-average solutions should be above average [CS98]. However, the common components of random solutions in the initial population are likely to be equally random (i.e. not significantly above average). It is likely that PPX over-exploits these early building blocks and converges prematurely in a poor region of the search space.

Table 1: Average performance of sequencing operators relative to EDD on randomly-generated 2-machine flow shop problems.

Instance	UX	PPX
1	0.745	7.306
2	1.409	16.550
3	1.106	14.088
4	1.490	22.071
5	0.567	5.538
average	1.063	13.111

[1] EDD is used because it is a factory independent dispatch rule. Most advanced rules use job processing times (which include factory dependent set-up times).

5 SEARCH SPACE REDUCTIONS

The goal of this paper is to demonstrate that constraints which improve the performance of genetic algorithms can be developed. These constraints will reduce the search space and cause search to start/focus in a promising region. For the TSP, a similar heuristic is to build tours using only edges connected to each element's ten nearest neighbors. This is a reasonable reduction because most of the edges in the optimal tour are present in this reduced search space [Rei94].

For flow shop problems with order-based objectives, precedence constraints can be used to reduce the search space. For example, if job `i` is due `x` time units after job `j` is due, set a precedence constraint that requires job `i` to be processed before job `j`. If precedence constraints are set for all pairs of jobs using the above condition, then the allowed time difference `x` defines a neighborhood size.

For the previously presented flow shop problems, precedence constraints have been set using 1, 100, 300, 500, 1000, 3000, and 5000 time units for the value of `x`. The constraints generated by this means define search neighborhoods with an average of 0, 2, 6, 20, 60, and 100 non-EDD neighbors for each job. Essentially, due dates are an average of 100 time units apart, so one additional neighbor (on each side) can be expected for each 100 time unit increase in `x`.

6 RESULTS WITH SEARCH SPACE REDUCTIONS

The search space reductions are only enforced for (the randomly generated solutions of) the initial population. After this initialization, any offspring solution is allowed in the population. The experimental set-up is the same as that used in section 4 (i.e. GENITOR, 200,000 solutions, etc). Over ten runs on each of the five flow shop instances, the average performance for each operator relative to EDD was measured for all values of `x`. (See Figure 3.)

When `x` is 300 time units or more, the results with PPX are about 40% better than UX. Overall, the cost objective can be reduced by over 80% relative to the EDD sequence and the performance of unconstrained genetic search[1]. The added precedence constraints define a reduced search space that contains only the most promising solutions. PPX

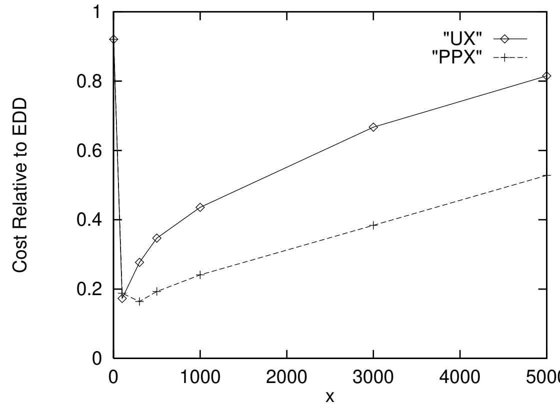

Figure 3: Average performance of UX and PPX for different values of `x`.

aggressively samples this space and finds better solutions in the allotted time than UX. Conversely, UX wanders out of the reduced search space and wastes time sampling less promising regions of the (overall) search space. This unconstrained wandering is only advantageous when the added precedence constraints are too restrictive (e.g. when `x` is 100).

This "wandering" nature of UX is also observed as a slower convergence rate. Even though UX performs better than PPX when `x` is 100, UX tends to lag PPX. (See Figure 4.) For larger values of `x`, the lag is presumed to be greater, so UX is likely unable to catch PPX during the time allowed. However, it appears that given enough time, UX can find better solutions than PPX for all problem instances.

7 DISCUSSION

Handling constraints and incorporating problem specific knowledge (heuristics) are two recurring challenges faced

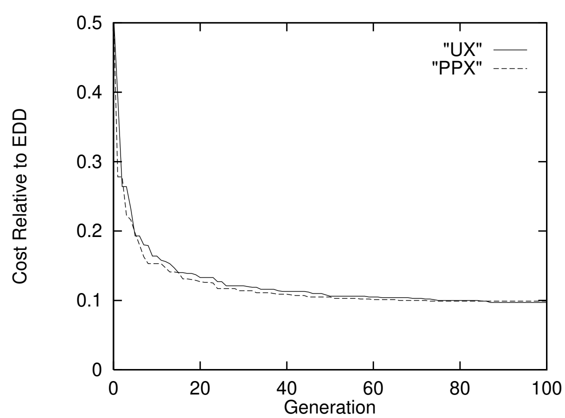

Figure 4: Sample run of UX and PPX on flow shop instance 1 with windowing condition of 100.

[1] Without bottlenecks, these problems can have very low cost solutions. For problems where the optimal solution still has a high cost, percentage cost reductions should be measured with respect to the surplus from (the unknown) optimum.

when working on real-world problems. Genetic operators that preserve common schemata may provide a means to do both. For constraints that can be observed in the representation alone[1], operators can be designed to "transparently" preserve the (satisfied) constraints of two feasible parents.

Further, heuristics which can be expressed as (representation-level) constraints may now be used to improve the performance of genetic algorithms. Traditionally, heuristics have been incorporated by redesigning crossover [GGR85][SG87][NK97]. By expressing heuristics as constraints, the performance of genetic algorithms is improved because the search space has been beneficially reduced. These reductions have an advantage in that they can be individually tailored to each problem instance and/or dynamically updated at run-time.

In this paper, precedence constraints (based on due dates) have been added to flow shop scheduling problems. With the resulting search space reductions, Precedence Preservative Crossover can perform better than Uniform Order-Based Crossover--previously, the best genetic sequencing operator. Further, PPX converges much faster than UX. For problem domains with stricter time limits on computational effort, the advantage of PPX over UX should be magnified.

Other methods to reduce the search space for scheduling problems include rolling time horizons [MP93] and problem decompositions [HNN94]. Unlike these other methods, the search space reductions caused by precedence constraints appear to be continuous and symmetric. Regardless, one benefit they can provide is to help determine the ideal rolling time horizon or ideal (neighborhood) size for problem decompositions.

For example, it appears that for the problems considered in this paper, the best value for x is 300. The resulting reduced search space defines a neighborhood structure where each job has about 8 possible neighbors. Thus, in designing a local search operator, the ideal neighborhood structure might include only the 8 nearest (due date) neighbors for each job. However, when local search (in the form of random swaps) was compared against genetic algorithms in [RHW96], the "local" neighborhood included all jobs. In this study, the effectiveness of local search may have been diluted by an excessively large neighborhood.

[1] Certain constraints require the solution representation to be processed. For example, a constraint that requires a job to be completed within a given time window cannot be observed in the representation alone.

8 CONCLUSIONS

Precedence constraints have been used to reduce the search space for flow shop problems with order-based objectives. A crossover operator that preserves these constraints benefits more from the advantages that they provide. Specifically, Precedence Preservative Crossover benefits more than Uniform Order-Based Crossover from the search space reductions studied in this paper. For search spaces reduced by precedence constraints, PPX can perform better than UX--previously, the best performing genetic sequencing operator.

Acknowledgments

The work described in this paper was sponsored in part by the Advanced Research Projects Agency and Rome Laboratory, Air Force Material Command, USAF, under grant numbers F30602-95-1-0018 and F30602-97-C-0227, and the CMU Robotics Institute. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Advanced Research Projects Agency and Rome Laboratory or the U.S. Government.

References

[Asc96] N. Ascheuer. (1996) *Hamiltonian Path Problems in the On-line Optimization of Flexible Manufacturing Systems.* ZIB Technical Report TR 96-3.

[BMK96] C. Bierwirth, D.C. Mattfeld, and H. Kopfer. (1996) "On Permutation Representations for Scheduling Problems." In *Parallel Problem Solving In Nature IV*, H.-M. Voight et al, eds. Springer-Verlag.

[CMM67] R. Conway, W. Maxwell, and L. Miller. (1967) *Theory of Scheduling*. Addison-Wesley.

[CS98] S. Chen and S.F. Smith. (1998) "Experiments on Commonality in Sequencing Operators." In *Genetic Programming 1998: Proceedings of the Third Annual Conference*.

[CS99] S. Chen and S.F. Smith. (1999) "Putting the "Genetics" back into Genetic Algorithms (Reconsidering the Role of Crossover in Hybrid Operators)." To appear in *Foundations of Genetic Algorithms 5*, W. Banzhaf and C. Reeves, eds. Morgan Kaufmann.

[Dav85] L. Davis. (1985) "Applying Adaptive Algorithms to Epistatic Domains." In *Proc. Ninth International Joint Conference on Artificial Intelligence*.

[EMS96] L.J. Eshelman, K.E. Mathias, and J.D. Schaffer. (1996) "Convergence Controlled Variation." In *Foundations of Genetic Algorithms 4*, R. Belew and M. Vose, eds. Morgan Kaufmann.

[Esc88] L.F. Escudero. (1988) "An Inexact Algorithm for the Sequential Ordering Problem." In *European Journal of Operations Research*, 37:236-253, 1988.

[Fre82] S. French. (1982) *Sequencing and Scheduling--An Introduction to the Mathematics of Job Shops*. Ellis Horwood Limited.

[GGR85] J. Grefenstette, R. Gopal, B. Rosmaita, and D. Van Gucht. (1985) "Genetic Algorithms for the Traveling Salesman Problem." In *Proc. of an International Conference on Genetic Algorithms and their Applications*.

[Gol89] D. Goldberg. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

[HNN94] N. Hirabayashi, H. Nagasawa, and N. Nishiyama. (1993) "A Decomposition Scheduling Method for Operating Flexible Manufacturing Systems." In *International Journal of Production Research*, 32:161-178.

[Hol75] J. Holland. (1975) *Adaptation in Natural and Artificial Systems*. The University of Michigan Press.

[MP93] T.E. Morton and D.W. Pentico. (1993) *Heuristic Scheduling Systems: With Applications to Production Systems and Project Management*. John Wiley & Sons.

[NK97] Y. Nagata and S. Kobayashi. (1997) "Edge Assembly Crossover: A High-power Genetic Algorithm for the Traveling Salesman Problem." In *Proc. Seventh International Conference on Genetic Algorithms*.

[Rad91] N.J. Radcliffe. (1991) "Forma Analysis and Random respectful Recombination." In *Proc. Fourth International Conference on Genetic Algorithms*.

[Rei94] G. Reinelt. (1994) *The Traveling Salesman: Computational Solutions for TSP Applications*. Springer-Verlag.

[RHW96] S. Rana, A.E. Howe, L.D. Whitley, and K. Mathias. (1996) "Comparing Heuristic, Evolutionary and Local Search Approaches to Scheduling." In *AIPS-96*.

[SG87] J.Y. Suh and D. Van Gucht. (1987) "Incorporating Heuristic Information into Genetic Search." In *Proc. Second International Conference on Genetic Algorithms and their Applications*.

[SMM91] T. Starkweather, S. McDaniel, K. Mathias, C. Whitley, and D. Whitley. (1991) "A Comparison of Genetic Sequencing Operators." In *Proc. Fourth International Conference on Genetic Algorithms*.

[SWM92] T. Starkweather, L.D. Whitley, K. Mathias, and S. McDaniel. (1992) "Sequence Scheduling with Genetic Algorithms." In *New Directions in Operations Research*.

[Sys91] G. Syswerda. (1991) "Schedule Optimization using Genetic Algorithms." In *Handbook of Genetic Algorithms*, L. Davis, ed. Van Nostrand Reinhold.

[WHR98] L.D. Whitley, A.E. Howe, S. Rana, J.-P. Watson, and L. Barbulescu. (1998) "Comparing Heuristic Search Methods and Genetic Algorithms for Warehouse Scheduling." In *Systems, Man and Cybernetics*, 1998.

[WS90] L.D. Whitley and T. Starkweather. (1990) "GENITOR II: A distributed Genetic Algorithm." In *Journal of Experimental and Theoretical Artificial Intelligence*, 2:189-214, 1990.