Improving Genetic Algorithm Performance with Intelligent Mappings from Chromosomes to Solutions

[Extended Abstract]

Justin Collins
Computer Science Dept.
Seattle University
Seattle, WA 98122
collinsj@seattleu.edu

David Joslin
Computer Science Dept.
Seattle University
Seattle, WA 98122
joslind@seattleu.edu

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Heuristic methods, Scheduling

General Terms

Algorithms

Keywords

Genetic Algorithms, optimization, representations

1. INTRODUCTION

Genetic algorithms (GA) generally use a simple, straightforward mapping from phenotype to genotype. The mapping is often so simple that we can think of the chromosome space and solution space as being identical. In this research, we look instead at many-to-one mappings from phenotypes to genotypes, as defined by polynomial-time domain-specific functions. These mappings may transform the search space of a particular problem so that it is easier for the GA to converge on good solutions. In order to study the effect these many-to-one mappings could have on a problem, we applied increasingly "intelligent" mappings to the problem of scheduling college courses. We looked at five different mappings, ranging from a very simple and direct mapping, to one that makes use of all hard and soft constraints in a flexible manner. Each of the mappings implement a greedy approach to scheduling the courses.

2. PROBLEM DESCRIPTION

We applied a steady-state genetic algorithm to the problem of college course scheduling for a student planning on earning a BS in computer science with a minor in math from Seattle University. We had three hard constraints, which we combined to form the requirements of a valid schedule: the student must satisfy all degree requirements, a course may only be taken during the quarter when it is offered, and a maximum of 18 credits may be taken per quarter. We also had three soft constraints: finishing within four years (length), balance of "difficult" courses to less demanding courses each quarter, and penalties for extraneous courses. These constraints were treated hierarchically.

The mappings we applied to the phenotypes were increasingly complex and domain-aware. The simplest mapping (M1) schedules each course in the first non-full quarter in which the course is offered, stopping when the degree requirements were met. The second mapping (M2) improves upon this by deferring the scheduling of a course if the prerequisites for that course at not yet met. The third mapping (M3) builds upon the first two, but instead of deferring the scheduling of courses with unmet prerequisites, it will recursively schedule all prerequisites.

The fourth mapping (M4) helps avoid unnecessary courses by first checking if a course will contribute to a degree before scheduling it. Courses which do not contribute to the degree will be silently dropped. If a course does contribute, it is scheduled in the same manner as the third mapping. The fifth (M5) mapping adds the consideration of soft constraints. When scheduling a course, the mapping starts at the first quarter in which all prerequisites are met and the course is offered, then checks the soft constraints for each successive quarter. It then schedules the course in the period with the lowest penalty.

3. EXPERIMENT RESULTS

Using Mapping 1, the GA never finds a valid solution. Mapping 2 finds valid solutions but rarely finds solutions that satisfy even one soft constraint, and never solutions that satisfy more than the first soft constraint. Mapping 3 finds valid solutions quickly, and gradually finds solutions that satisfy one and sometimes two of the soft constraints, but none that satisfy all three soft constraints. Mapping 4 never does worse than finding solutions that satisfy two of the three soft constraints, even in the initial random pool. It does find solutions that satisfy all of the soft constraints fairly easily. After 1000 iterations a solution satisfying all of the soft constraints had been found in 72% of the runs. Little improvement was seen after that point. Mapping 5 found optimal solutions (valid solutions meeting all soft constraints) in the initial random pool most of the time, and it always found an optimal solution within one hundred iterations.

To better understand the way in which the various mappings transform the search space, we generated one million random chromosomes for each mapping and counted the number of solutions at each quality level (valid solutions,

GECCO'06, July 8–12, 2006, Seattle, Washington, USA.
ACM 1-59593-186-4/06/0007.

Quality	M2	M3	M4	M5
Valid	100%	100%	100%	100%
1 soft constr.	None	.0023%	76.62%	47.77%
2 soft constr.	None	None	13.81%	10.13%
Optimal	None	None	.0043%	.89%

Figure 1: One million random chromosomes

and valid solutions with one, two or all three soft constraints satisfied) that resulted from applying the mapping. The table in Figure 1 shows the results. The columnn for Mapping 1 is omitted because it never found valid solutions. It is striking that Mapping 4 and Mapping 5 were able to produce optimal solutions from random chromosomes. It also helps explain the performance of Mapping 5, since it has a good chance of finding at least one optimal solution in just the initial random population.

4. RELATED WORK

Although it is most common in Genetic Algorithm implementations to have chromosomes that map to solutions in a very simple and straightforward fashion, examples of more complex mappings can be found. A commercially successful example is described in a patent by Gil Syswerda [6], concerning a system for task scheduling for manufacturing applications. In that system a deterministic "schedule builder" takes a sequence of tasks and attempts to construct a valid schedule, respecting the hard constraints of the problem. A Genetic Algorithm was used to search the space of chromosomes. Our research follows essentially this same structure with the goal of better understanding the tradeoffs that result from increasing the domain-specific knowledge of such "builder" functions.

Another commercially successful application, also patented, is Squeaky-Wheel Optimization (SWO) [4]. Like Syswerda's approach, SWO uses a greedy algorithm that maps a sequence of tasks or other problem elements to a candidate solution. Rather than using a GA to search the space of task sequences, however, SWO uses only a "directed mutation" or "genetic engineering" approach. SWO has been successfully applied to a wide variety of domains, including factory scheduling and graph coloring [4], satellite downlink scheduling [1], satellite observation scheduling [3], project scheduling [5], and scheduling of airborne astronomical observations [2].

5. CONCLUSIONS

Of the mappings used, Mapping 1 does the least amount of work. All it does is divide the courses into quarters, stopping when the degree requirements have been met. Thus, the bulk of the responsibility for finding a good solution is on the GA. However, Mapping 1 never even found a valid schedule in our experiments. Starting with Mapping 2, the structure of the rest of the mappings guaranteed valid schedules, which was already an large improvement over a simple mapping. However, our results showed even the guarantee of valid schedules did not necessarily lead to the evolution of desirable schedules, since the soft constraints still needed to be satisfied.

Each mapping improves on the previous mappings, so they can be thought of as increasing in both complexity and domain awareness. The mappings with the most domain-specific heuristics performed the best in terms of finding a good solution quickly. Mapping 5 was able find nearly perfect solutions out of the initial random population, then perfect solutions within a few iterations.

The experiments using random input to the mappings suggest that, as we anticipated, the complex mappings make it easier for the GA to find good solutions in this particular problem by providing larger "targets." Numerous chromosomes are funneled to the same point in the solution space, and points in solution space representing invalid and poor solutions are in some cases not reachable at all from any point in the space of chromosomes.

A polynomial-time mapping that always produces perfect solutions from any input sequence would make the GA entirely unnecessary, but of course for many interesting and realistic problems we do not know how to define such a mapping. As mappings are given more and more control over the decisions made in the generation of solutions, we might imagine that at some point these non-optimal mappings could become "too clever" in a way that nullifies the GAs attempts to find improvements. In our experiments we did not find this to be the case, but it is clearly a potential pitfall for this approach. A mapping could even make it impossible for a GA to find the best solutions if it happens that none of the chromosomes are mapped to those solutions. This is a potential downside of using a many-to-one mapping as opposed to a one-to-one mapping.

These results are admittedly preliminary because the experiments are limited in scope. The results are still very encouraging. Defining effective mappings was not difficult and did not require a great deal of trial-and-error. The successes of other mapping-based algorithms, such as SWO [4] and the algorithm patented by Syswerda ([6]), are encouraging in this regard. We look forward to extending this work to other domains, toward a better understanding of how to make the most effective use of complex, domain-aware mappings in GAs.

6. REFERENCES

[1] L. Barbulescu, D. Whitley, and A. Howe. Leap before you look: An effective strategy in an oversubscribed scheduling problem. In *Proc. of the 19th National Conference on Artificial Intelligence*, 2004.

[2] J. Frank and E. Kürklü. Mixed discrete and continuous algorithms for scheduling airborne astronomy observations. In *Proc. of the 2nd Intl. Conference on Constraint Programming, Artificial Intelligence and Operations Research*, 2005.

[3] A. Globus, J. Crawford, J. Lohn, and A. Pryor. A comparison of techniques for scheduling earth observing satellites. In *Proc. of the 16th Conference on the Innovative Applications of Artificial Intelligence*, 2004.

[4] D. E. Joslin and D. P. Clements. Squeaky wheel optimization. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98), Madison, WI*, pages 340–346, 1998.

[5] T. Smith and J. Pyle. An effective algorithm for project scheduling with arbitrary temporal constraints. In *Proc. of the 19th National Conference on Artificial Intelligence*, 2004.

[6] G. P. Syswerda. Generation of schedules using a genetic procedure, 1994. U.S. Patent number 5,319,781.