Very Large-Scale Neighborhood Search Techniques in Timetabling Problems

Carol Meyers[1] and James B. Orlin[2]

[1] Operations Research Center, Massachusetts Institute of Technology
carol@mit.edu *
[2] Sloan School of Management, Massachusetts Institute of Technology
jorlin@mit.edu *

Abstract. We describe the use of very large-scale neighborhood search (VLSN) techniques in examination timetabling problems. We detail three applications of VLSN algorithms that illustrate the versatility and potential of such algorithms in timetabling. The first of these uses *cyclic exchange neighborhoods*, in which an ordered subset of exams in disjoint time slots are swapped cyclically such that each exam moves to the time slot of the exam following it in the order. The neighborhood of all such cyclic exchanges may be searched effectively for an improving set of moves, making this technique computationally reasonable in practice. We next describe the idea of *optimized crossover* in genetic algorithms, where the parent solutions used in the genetic algorithm perform an optimization routine to produce the 'most fit' of their children under the crossover operation. This technique can be viewed as a form of multivariate large-scale neighborhood search, and it has been applied successfully in several areas outside timetabling. The final topic we discuss is *functional annealing*, which gives a method of incorporating neighborhood search techniques into simulated annealing algorithms. Under this technique, the objective function is perturbed slightly to avoid stopping at local optima. We conclude by encouraging the timetabling community to further examine the promising potential of these techniques in practice.

1 Introduction

1.1 Timetabling Problems

The scheduling of classes and examinations is a key practical problem that is faced by nearly all schools and universities. Substantial effort has been devoted to developing effective timetabling procedures over the last thirty to forty years. The problems tackled by such procedures include *examination timetabling*, in which a set of exams is to be scheduled over a set of time periods, and *course timetabling*, where a set of courses must be scheduled over the length of an entire semester.

* This work was supported in part through NSF Grant DMI-0217123.

E. K. Burke, H. Rudová (Eds.): PATAT 2006, pp. 36–52. ISBN 80-210-3726-1.

Timetabling problems are often complicated by numerous constraints; for instance, in the examination timetabling problem, students should not be scheduled to take two exams at the same time. These constraints are typically divided into *hard constraints*, which must not be violated (in the course timetabling problem, a hard constraint might be that no teacher is scheduled to teach two classes at once), and *soft constraints*, which possess a penalty for being violated (in the examination timetabling problem, a soft constraint might be to minimize the number of students who take two exams back-to-back). Because of the number and variety of constraints, such timetabling problems typically constitute NP-hard problems that are quite difficult to solve manually. This in turn has led to an increased emphasis on finding effective *automated* timetabling algorithms.

Recent surveys on automated timetabling (see [21, 23, 24, 43]) illustrate the wide array of methods that have been applied to timetabling problems. Traditional techniques tested in timetabling include *direct heuristics* [34], which fill up the timetable one event at a time and resolve conflicts by swapping exams, and a reduction to the *graph coloring* problem [38], where events are associated with vertices of a graph and edges with potential conflicts. More modern heuristics include *memetic* [20] and *genetic algorithms* [19, 27, 30], which use techniques inspired by evolutionary biology; *simulated annealing* algorithms [18, 46], where nonimproving solutions are permitted with progressively decreasing probability; *tabu search* heuristics [26, 42], where a list of recently visited timetables are forbidden to be visited; and *constraint logic programming* approaches [25], which are based on applying declarative logic programming systems to constraint satisfaction problems.

In this paper, we address the application of very large-scale neighborhood search techniques (see Section 1.2) to timetable scheduling problems, including one approach based on genetic algorithms (Section 3) and one that resembles simulated annealing (Section 4). Neighborhood search has long been used in timetable scheduling, from the swap (2-opt) techniques used in the direct approaches to the variety of forms of neighborhood search used in genetic algorithms. However, the area of *very large-scale neighborhood search* has only recently been investigated with respect to timetable scheduling [1, 13, 33] (see Section 2). We believe there are many untapped possibilities for useful algorithms in this context.

1.2 Very Large-Scale Neighborhood Search

Neighborhood search algorithms (also known as *local search* algorithms) are a class of algorithms that start with a feasible solution and attempt to find an improving solution in the *neighborhood* of the current solution. The neighborhood structure may be defined in a variety of ways, typically so that all solutions in the neighborhood of the current solution satisfy a set of prescribed criteria. In *very large* neighborhoods, the size of the neighborhood under consideration is extremely large (typically, exponential) in the size of the problem data, making it impractical to search such neighborhoods explicitly.

A *very large-scale neighborhood search* (VLSN) algorithm is one that searches over a very large neighborhood, giving an improving solution in a *relatively efficient* amount of time. Such algorithms tend to search *implicitly* over the neighborhood rather than explicitly, since the quantity of solutions precludes performing an exhaustive search.

There are three main categories of very large-scale neighborhood search algorithms that are outlined in [5]. The first of these is *variable depth* methods, which partially search an exponentially large neighborhood by using heuristics. The second kind are *network flow-based* methods, which use network flow techniques to search over the neighborhood and identify improving neighbors. The third main category consists of neighborhoods based on *restrictions* of NP-hard problems that are solvable in polynomial time. Ahuja, Ergun, Orlin, and Punnen [5, 6] provide a thorough exposition of the algorithms in these categories in their surveys on the topic.

Very large-scale neighborhood search techniques have been applied to a wide range of problems in combinatorial optimization. These include the traveling salesman problem [28, 35, 39], the quadratic assignment problem [7], vehicle routing problems [2, 29], the capacitated minimum spanning tree problem [10], the generalized assignment problem [50, 51], and parallel machine scheduling problems [3]. In several of these problems, the VLSN search algorithms give the strongest known computational results, making the development of such algorithms desirable in practice.

The design of a successful VLSN search algorithm depends on the choice of an appropriate neighborhood function and the development of an effective heuristic method to search the neighborhood for improving solutions. VLSN search techniques may also be *combined* within the framework of other heuristic methods, such as tabu search [32, 33] and scatter search [41], to provide further computational improvements. See [5, 6] for a comprehensive discussion of techniques for developing strong VLSN search algorithms.

1.3 Contributions of this Paper

We describe three applications of very large-scale neighborhood search techniques to timetabling problems. For simplicity, we consider the *examination* timetabling problem in each of these instances, but our approaches can be modified to apply to classroom timetabling problems as well.

In Section 2, we describe the *cyclic exchange* neighborhood and how it may be applied to timetabling problems. In this neighborhood, an ordered subset of exams in disjoint time slots are swapped in a cyclic fashion such that each exam moves to the time slot of the exam following it the order. We consider recent applications of the cyclic exchange neighborhood in the timetabling literature, and relations to other neighborhood search techniques in timetabling.

We discuss the idea of *optimized crossover* in genetic algorithms in Section 3. In an optimized crossover, the parent solutions used in the genetic algorithm perform an optimization routine to produce the 'most fit' of their children under the crossover operation. This can be viewed as a form of very large-scale

neighborhood search, where the neighborhood is defined over *both* of the parent solutions. We discuss problems for which the optimized crossover has been applied, and how a heuristic for optimized crossover could be incorporated into genetic algorithms for timetabling problems.

In Section 4, we review a new metaheuristic algorithm known as *functional annealing* that combines neighborhood search techniques with a type of simulated annealing algorithm. This algorithm allows the application of very large-scale neighborhood search techniques within an annealing framework, which was not previously practical due to the random selection of solutions in simulated annealing. We discuss how this algorithm has the potential to be very useful in timetable scheduling problems, on which simulated annealing algorithms have performed well in the past.

2 Cyclic Exchange Neighborhood

2.1 Definition

The cyclic exchange neighborhood is defined for partitioning problems. We present the problem here in terms of scheduling a set of exams over a collection of time periods, where potential conflicts between the exams are implicitly encoded in the objective function. However, it should be noted that this neighborhood extends to any problem that can be expressed in terms of partitioning the members of one set, so long as the cost of a partition is the sum of the cost of its parts.

Let $E = \{e_1, e_2, \ldots, e_n\}$ be a set of n exams, and let $P = \{p_1, p_2, \ldots, p_m\}$ be a set of m time periods in which we wish to schedule the exams. Suppose that $S = \{S_1, S_2, \ldots S_m\}$ is a *partitioning* of the exams in E into m sets, such that each exam belongs to exactly one set in S, and each set S_i corresponds to the collection of exams scheduled in period p_i. Let $c(S)$ denote the cost of solution S. We assume that any conflicts between students and exams are implicitly encoded in the objective function $c(S)$, so that *any* valid partitioning of the exams represents a feasible solution to the problem. This is similar to the approach taken by Abdullah, Ahmadi, Burke, and Dror [1].

Consider a sequence $e_{i_1}, e_{i_2}, \ldots, e_{i_k}$ of exams in E such that exam e_{i_j} is contained in set S_j, for each j. Suppose we switch exam e_{i_j} from set S_j to set S_{j+1}, for all $j = 1, \ldots, k-1$, and we switch exam e_{i_k} into set S_1. We call such an operation a *cyclic exchange*. We can also think of the exams as forming a *cycle* $e_{i_1} - e_{i_2} - e_{i_3} - \ldots - e_{i_k} - e_{i_1}$, such that each exam switches to having the time slot of the exam following it in the cycle. An illustration of a cyclic exchange is given in Figure 1. In the figure, the sequence $e_1 - e_4 - e_{10} - e_{13}$ of exams forms a cycle; exam e_1 switches from S_1 to S_2, exam e_4 switches from S_2 to S_4, exam e_{10} switches from S_4 to S_5, and exam e_{13} switches from S_5 to S_1. The set S_3 is not included in the cyclic exchange, so its exams are not changed.

In the case where $k = 2$, this operation is equivalent to the *2-opt* operation, where a single pair of exams switch time slots. Neighborhoods defined over the

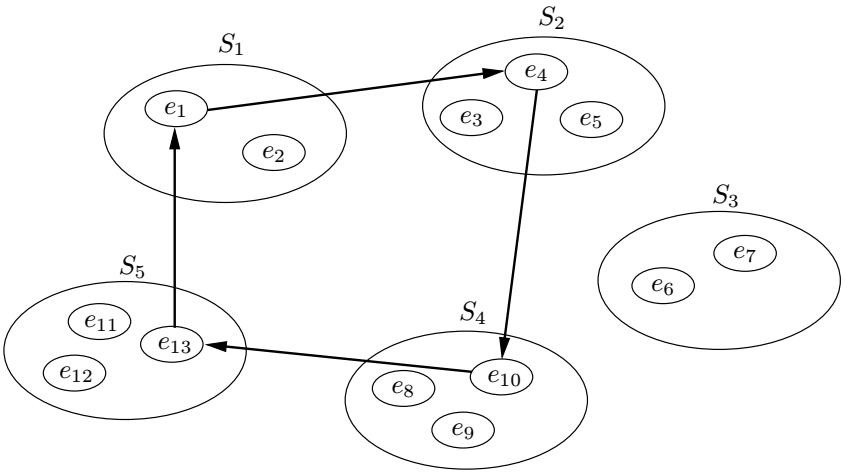

Fig. 1. The cyclic exchange neighborhood.

2-opt operation have been studied previously in the timetabling community by Alvarez-Valdez, Martin, and Tamarit [12], Colorni, Dorigo, and Maniezzo [26], and Schaerf [42], among others. If instead we do not require exam e_{i_k} to move into set S_1, then we call the operation a *path exchange*, which can be described by the path of exams $e_{i_1} - e_{i_2} - e_{i_3} - \ldots - e_{i_k}$. We can show mathematically that path exchanges may be modeled as a special case of cyclic exchanges, by adding dummy nodes as appropriate [10].

We define the *cyclic exchange neighborhood* of solution S as all partitions $T = \{T_1, T_2, \ldots, T_m\}$ that can be obtained from the sets $\{S_1, S_2, \ldots S_m\}$ via a cyclic exchange operation. The size of this neighborhood is exponential in m, the number of periods; for a fixed value of m, the total number of cyclic neighbors of a given solution is $O(n^m)$. Since the size of this neighborhood is enormously large, the neighborhood structure will only be useful in practice if we have an effective search method for finding improving solutions. Fortunately, Thompson and Psaraftis [49] and Ahuja, Orlin, and Sharma [9, 10] have identified several methods of finding such solutions.

2.2 Searching the Cyclic Exchange Neighborhood

We use the concept of an *improvement graph*, introduced in Thompson and Orlin [48] and further examined by Thompson and Psaraftis [49]. Rather than explicitly searching over each possible solution in the neighborhood, the improvement graph allows us to *implicitly* search the neighborhood for improving solutions. This helps dramatically reduce the amount of required computations.

For a feasible partition $S = \{S_1, S_2, \ldots S_m\}$ of the exams, the improvement graph $G(S)$ is a directed graph with n nodes, each corresponding to one of the exams in $e_1, e_2, \ldots, e_n$. The arc (e_i, e_j) represents the transferring of exam e_i

from the subset $S[i] \in S$ that contains it to the subset $S[j] \in S$ containing exam e_j, with exam e_j becoming unassigned. More formally, if we let $S[i]$ denote the subset in S containing exam e_i, we can define the edge set as $\{(e_i, e_j) \mid S[i] \neq S[j]\}$, with the interpretation of each arc as previously described. The cost of arc (e_i, e_j) is set to $c_{ij} = c(\{e_i\} \cup S[j] \backslash \{e_j\}) - c(S[j])$. This is exactly equal to the cost of adding exam e_i to set $S[j]$ and unassigning exam e_j from $S[j]$.

We say a cycle W in $G(S)$ is *subset-disjoint* if the exams in E that correspond to the nodes in W are all scheduled in different time slots in S. (In other words, for every pair of nodes e_i and e_j in W, we have $S[i] \neq S[j]$.) Thompson and Orlin [48] showed that there exists a one-to-one correspondence between cyclic exchanges in S and subset-disjoint directed cycles in $G(S)$; most importantly, they both have the *same cost*.

This result suggests that to effectively search the cyclic exchange neighborhood, we need only to identify *negative cost subset-disjoint cycles* in the improvement graph. Unfortunately, although the problem of finding a general negative cost cycle is solvable in polynomial time [8], the problem of finding a negative cost subset-disjoint cycle is NP-hard [47, 48]. However, Thompson and Psaraftis [49] and Ahuja, Orlin, and Sharma [10] have identified effective heuristic algorithms that produce negative cost subject-disjoint cycles quickly in practice. Thompson and Psaraftis's heuristic begins by initially searching for only small negative cost subset-disjoint cycles (i.e., 2-cycles or 3-cycles), and uses a variable depth approach to increase cycle length and cost improvement. Although their algorithm generates and searches only a portion of the graph $G(S)$, it was found to be effective in practice. Ahuja, Orlin, and Sharma's heuristic is a modification of the label-correcting algorithm for the shortest path problem, which restricts every path found by the label-correcting algorithm to being a subset-disjoint path. They found that on test instances, the time to identify a negative cost cycle was less than the time needed to construct the improvement graph.

Hence, the idea of an improvement graph can be efficiently exploited to allow searching of the cyclic exchange neighborhood. Using the algorithms of Thompson and Psaraftis and Ahuja, Orlin, and Sharma, improving solutions in the neighborhood can be found successfully in practice. This suggests that the cyclic exchange neighborhood is a valuable network structure to consider in solving timetabling problems.

2.3 Cyclic Exchange in the Timetabling Literature

Cyclic exchange neighborhoods have been investigated only recently in the timetabling literature. For this reason, we believe this is a potentially fruitful area for research in timetabling. We now outline a couple of the studies in which the cyclic exchange neighborhood has been incorporated.

Abdullah, Ahmadi, Burke, and Dror [1] initiated the first study of the cyclic exchange neighborhood in examination timetabling problems. To identify negative cost subset-disjoint cycles, they used the heuristic of Ahuja, Orlin, and Sharma [10]. They additionally introduced an exponential Monte Carlo acceptance criterion (see [14]) for accepting nonimproving moves. In this way, their

algorithm is less likely to become stuck at a local optimum. Tests of the algorithm against other timetabling algorithms on common benchmarks showed that the performance of their algorithm is comparable to that of the best currently known timetabling algorithms.

Jha [33] has also recently studied the usefulness of cyclic exchange neighborhoods in timetabling problems. His algorithm uses a dynamic programming aproach to identify negative cost subset-disjoint cycles. He also combines the cyclic exchange heuristics with a tabu search framework, to avoid the problem of halting at local optima. In terms of implementation, he found that the VLSN-tabu search combination produced robust solutions in a reasonable amount of time. Compared to approaches using integer programming or neighborhood search alone, he found that the VLSN-tabu search algorithm performed better on larger test instances.

Together, these two studies suggest that the *combination* of cyclic exchange techniques with other suitable timetabling heuristics can make for especially strong algorithms. Whether the methods used are Monte Carlo acceptance techniques or tabu search, the combination of the VLSN methodology with the existing algorithms can be used to produce a more effective algorithm overall.

2.4 Relation to Other Techniques in the Literature

As mentioned in Section 2.1, the *2-opt* operation is a special case of the cyclic exchange operation, where each cycle has length equal to 2. This is occasionally referred to as the *swap* operation, since it consists of swapping the time slots of a pair of exams. The *2-opt neighborhood* is defined as the set of all possible solutions that can be reached from a given solution by performing a single 2-opt move.

Many papers in the timetabling literature have used neighborhood search over the 2-opt neighborhood to refine timetabling solutions, though not necessarily using that name and most often in conjunction with other techniques. Alvarez-Valdes, Martin, and Tamarit [12] used 2-opt moves combined with tabu search in finding solutions for timetabling problems in the Spanish school system. Schaerf [42] combined tabu search and the randomized nonascendent method with 2-opt neighborhood search techniques in solving high school timetabling problems. Colorni, Dorigo, and Maniezzo [26] used 2-opt techniques along with simulated annealing, tabu search, and genetic algorithms for problems from Italian high schools; they found the combination of genetic algorithms with tabu search to be especially powerful. Carter [22] addresses the scheduling of classes at the University of Waterloo by decomposing the problem into several subproblems, which are then solved using a greedy procedure including 2-opt moves.

It should be noted that while 2-opt moves can be done efficiently in the improvement graph (since there are only $O(n^2)$ possible such moves), they are inherently a lot weaker than cyclic exchange moves. For this reason, it would be interesting to apply the cyclic exchange neighborhood to the same classes of problems. This presents a fruitful, and largely unexamined, avenue for new research.

3 Optimized Crossover in Genetic Algorithms

3.1 Overview of Genetic Algorithms

Genetic algorithms are an optimization technique based on the mechanisms of evolution and natural selection [37]. In applying genetic algorithms to timetabling problems, we assume (as in Section 2) that any valid partitioning of exams into a timetable T represents a feasible solution, and that potential conflicts between the exams are implicitly encoded in the objective function. (It should be noted that it is also possible to extend the following definitions to the constrained version of the problem.) There are a wide range of ways to implement genetic algorithms. We describe a classic approach.

In each iteration of a genetic algorithm, a *population* of solutions is maintained, which represent the current set of candidate solutions. At time $t = 0$, a population of timetables $\{T_1^0, T_2^0, \ldots, T_K^0\}$ is generated randomly from the set of all possible solutions. In further iterations, the population $\{T_1^{t+1}, T_2^{t+1}, \ldots, T_K^{t+1}\}$ is generated from the population at time t according to the *fitness* of each of the candidate solutions T_i^t, along with *crossover* and *mutation* operations.

The *fitness* function is a problem-specific measure of how good a timetable is. One obvious candidate for the fitness of a solution is its objective function value. (However, in problems for which calculating the objective is time-consuming, alternative methods of fitness can be formulated.) In selecting a set of candidate solutions at time t to produce the next generation at time $t + 1$, the algorithm begins by assessing the fitness of all timetables at time t. Next, K individuals of the population are randomly selected, based on a weighted randomization scheme; the 'fitter' a solution is, the more likely it is to be selected.

The *crossover* operation functions by taking two of the selected timetables T_i and T_j and combining them to form a new timetable. The selected timetables are referred to as the *parent* timetables, and the new timetable is called the *child* timetable. In what follows, we assume that the parent timetables are represented in the form $(p_1^k, p_2^k, \ldots, p_n^k)$, where p_ℓ^k represents the time period in which exam e_ℓ is scheduled in timetable T_k.

The crossover operation can take several forms, of which the *fixed point* crossover is very common. In this situation, a given position $\ell \in \{1, \ldots, n-1\}$ is selected; the child solution is created by concatenating the first ℓ periods in the timetable of the first parent with the last $n - \ell$ periods in the timetable of the second parent. Hence, if T_i and T_j are the first and second parents, their child solution will have the form $(p_1^i, \ldots, p_\ell^i, p_{\ell+1}^j, \ldots, p_n^j)$.

Another frequently used crossover scheme is the *two-point crossover*, where two random positions ℓ_1 and ℓ_2 ($\ell_1 < \ell_2$) are selected; in this case, the child is formed by taking the periods of the first parent in the intervals $(1, \ell_1)$ and $(\ell_2 + 1, n)$ and the periods of the second parent in the interval $(\ell_1 + 1, \ell_2)$, giving a solution of the form $(p_1^i, \ldots, p_{\ell_1}^i, p_{\ell_1+1}^j, \ldots, p_{\ell_2}^j, p_{\ell_2+1}^i, \ldots, p_n^i)$. Similarly, we can define *multi-point crossovers* by first generating a random number N, arbitrarily determining N crossover positions, and then creating the child by taking each odd interval from the first parent and each even interval from the second parent.

The *mutation* operation is used to ensure diversity of the timetables generated. In this operation, a given position ℓ in timetable T_k is selected with some (small) probability P_m, and exam e_ℓ is reassigned from period p_ℓ^k in which it is currently scheduled to another randomly selected time period. This has the effect of 'mutating' the ℓth exam period from its original value. In this way, time periods that are not a part of the set of parent timetables can be present in the successive generation, which occasionally leads to better solutions.

3.2 Optimized Crossover

In the previous section we discussed the crossover operation in genetic algorithms. One striking feature of this method is that the crossover points are determined *randomly*, and the resulting child is created *without regard to the objective function*. Hence occasionally the fitness of a child can deviate quite widely from the fitness of its parents. Aggarwal, Orlin, and Tai [4] suggested instead choosing the *best* child from all possible children, building on an idea of Balas and Niehaus [15] in the area of graph theory.

The set of all possible children T_{ij} from two timetables T_i and T_j can be written as $\{T_{ij} \mid p_{ij}^\ell = p_i^\ell \text{ or } p_{ij}^\ell = p_j^\ell, \text{ for all } \ell = 1, \ldots, n\}$. Thus, the period in which any exam is scheduled in T_{ij} will either be the same as the period in which it is scheduled in T_i, or else the same as the period in which it is scheduled in T_j. The problem of finding the *best* child is then the problem of choosing from among the $O(2^n)$ possible children the one with the best objective function.

We can think of solving the optimized crossover problem as a type of very large-scale neighborhood search. In this case, the neighborhood is defined over a *pair* of parent solutions, instead of a single solution. This is a somewhat unusual use of the term 'neighborhood,' but we claim the concept is plausible since the neighborhood is well-defined. For each pair of solutions T_i and T_j, the crossover neighborhood is defined as the set of all possible children T_{ij} that can be produced from T_i and T_j. The problem of finding the *best* child can be viewed as that of finding the child with the best objective value in the crossover neighborhood.

The idea of optimized crossover has not been previously used in genetic algorithms for timetabling problems, and we believe it is an excellent candidate for study. In the next two sections, we detail a few of the areas in which optimized crossover has proven to be useful, followed by comments on the feasibility of the method on timetabling problems in particular.

3.3 Previous Applications of Optimized Crossover

Aggarwal, Orlin, and Tai [4] were the first to apply the concept of optimized crossover to genetic algorithms. They studied the independent set problem, for which they gave an effective method of combining two independent sets to obtain the largest independent set in their union. This was based on a related technique of Balas and Niehaus [15]. Their resulting genetic algorithm incorporated this

optimized crossover scheme, and was shown to be superior to other genetic algorithms for the independent set problem. This approach was further verified by Balas and Niehaus [16].

Ahuja, Orlin, and Tiwari [11] later extended the idea of optimized crossover to genetic algorithms for the quadratic assignment problem. They presented a matching-based optimized crossover heuristic that finds an optimized child quickly in practice. This technique can also be applied to other assignment-type problems, as it relies on the structure of the problem rather than the objective function.

Most recently, Ribeiro and Vianna [40] have applied the idea of optimized crossover to genetic algorithms for building phylogenetic trees, which are trees showing evolutionary relationships among species with a common ancestor. Their algorithm outperforms the best algorithms currently available. Lourenço, Paixão, and Portugal [36] have also used a type of optimized crossover heuristic in their study of bus driver scheduling. They solve a set-covering subproblem to determine the best child solution; their algorithm outperforms other algorithms tested, albeit at a higher computational cost.

3.4 Optimized Crossover in Timetabling Problems

As mentioned in Section 3.2, for an optimized crossover to be effective in practice, it requires a method of quickly obtaining a best (or very good) child solution from two parents. The problem of finding the optimized crossover *explicitly* in timetable scheduling problems is unfortunately NP-hard, via a transformation from the Minimum Set Cover problem (see [31]). Hence, the best we can hope for is to find a strong heuristic for obtaining a good crossover. We now describe how this can be accomplished in timetabling problems.

The algorithm we consider here is a *greedy algorithm*, which starts with the two parent solutions T_i and T_j. First it randomly selects an order to consider the exams in. The algorithm proceeds through the exams in order, where for each exam e_k it places the exam in either slot T_i^k or T_j^k according to which one gives the smallest increase in the objective function. The result will be a scheduling of exams that (hopefully) gives a low objective value. (Many other variations in the greedy algorithm are possible.)

This algorithm will perform quickly in practice, as once the ordering is decided upon there are only two choices for each of the exams. The quality of the solutions produced by the algorithm may vary depending on the quality of the ordering.

Thus we have given a heuristic for solving the optimized crossover problem in genetic algorithms for timetabling problems. Though this method has not been tested in a timetabling context, we believe the strong results obtained for the crossover method in other problems (see [4, 11]) make it an attractive avenue to pursue in the area of timetabling.

4 Functional Annealing

4.1 The Functional Annealing Algorithm

The functional annealing method is a relatively new metaheuristic for combinatorial optimization problems. Proposed by Sharma and Sukhapesna [44, 45], it combines the attractive components of both a neighborhood search method and a simulated annealing algorithm. As simulated annealing algorithms have been extensively examined in the timetabling literature (see, for instance, [18] and [46]), we believe this method should be greatly appealing to the timetabling community. In this subsection and the next two, we outline the functional annealing algorithm and its properties, followed by a discussion of applying functional annealing techniques to timetabling problems in particular.

The main idea of the functional annealing method is to introduce a stochastic element into the objective function, while employing an efficient neighborhood search strategy. The stochastic element is given in terms of an *annealing function*, which tends to the original objective as the number of iterations increases. The perturbed objective allows the algorithm to escape efficiently from local optima, while the neighborhood search heuristic provides for a more effective search of the feasible space.

We now describe the algorithm more formally, following the structure of Sukhapesna [45]. Suppose we are given a 0-1 discrete optimization problem (such as a timetabling problem), with a cost function $c(x)$ and a neighborhood $N(x)$ for each element x in the set $F \subseteq \{0,1\}^n$ of feasible solutions. We let $c(x,w) = c(x) + w'x$ be our annealing function, where w is a random vector in $\mathbb{R}^n$ with independent and identically distributed elements. The *volatility* of w is determined by a control paramater U, such that w approaches zero as U approaches zero. We assume we are given a sequence $\{U_k\}$ of such control parameters, such that $U_k > 0$ for all $k \geq 0$ and $\lim_{k\to\infty} U_k = 0$. Thus, the longer the algorithm runs, the less stochasticity there is in the objective function. The functional annealing algorithm is described in Figure 2.

As can be seen from the algorithm, the random vector w_k is always chosen so that the perturbation attempts to make the current solution worse than its neighbors, which has the effect of forcing the algorithm to move away from its current solution. Moreover, the magnitude of the perturbation vector w_k is such that the greater the number of iterations, the smaller the influence of the perturbation. Hence for small values of k, the algorithm behaves similiarly to a search for a random neighbor, and for large enough values of k, the algorithm behaves more like a deterministic neighborhood search algorithm.

One of the appealing features of using a neighborhood search strategy in tandem with the functional annealing approach is that the algorithm will not spend multiple iterations at a solution that is not a local optimum, in contrast to the standard simulated annealing algorithm. Another item of note is that in the case of a linear objective, the algorithm is equivalent to a problem where the data is perturbed to avoid lingering at local optimal solutions (see [17]).

algorithm *functional annealing*
begin
choose an initial solution x_0 in F;
set $k = 0$;
while stopping criteria are not met, **do**
generate a vector w_k such that $w_k(i) = e_k(i)$ if $x_k(i) = 1$ and $w_k(i) = -e_k(i)$ if $x_k(i) = 0$, where e_k is distributed exponentially with mean U_k;
using a neighborhood search algorithm, find a neighboring solution $y \in N(x) \cup \{x\}$, such that $c(y, w_k) \leq c(x, w_k)$;
set $x_{k+1} = y$;
set $k = k + 1$;
end;
end;

Fig. 2. The functional annealing algorithm.

4.2 Properties of the Algorithm

A natural question one might have about the functional annealing algorithm is whether it is guaranteed to reach the set of optimal solutions. Indeed, Sharma and Sukhapesna [44, 45] have shown that the algorithm is guaranteed to attain the set of optimal solutions with probability 1, provided that the neighborhood search algorithm is such that at any given step each improving solution is chosen with positive probability. Moreover, the expected number of iterations needed to reach an optimal solution is finite.

With respect to the choice of improving neighbors, the authors consider a *randomized first improvement* strategy, in which improving solutions in the neighborhood are selected with equal probability. If no improving neighbor is found, then the current solution is kept for the next iteration. They show that the chance of exiting from the current solution under such a strategy is *not worse* than that of simulated annealing, and for large numbers of iterations the exiting probability is about $|N(x)|$ times *greater* than that of simulated annealing. Thus the functional annealing algorithm is better in theory than simulated annealing in terms of becoming stuck at local optima. They also show that a *best improvement* strategy is also guaranteed to reach the set of optimal solutions with probability one, though the time to find a solution takes longer than with the first improvement strategy.

Sharma and Sukhapesna [44, 45] give a thorough computational study of functional annealing algorithms applied to the quadratic assignment problem. They show that the functional annealing algorithm performs *significantly better* than both simulated annealing and neighborhood search algorithms on instances of the problem, confirming the earlier theoretical results. This improvement holds regardless of the size of the instance being considered. They also show that the best improvement strategy tends to outperform the randomized first improvement strategy on small instances, while on larger instances the difference is less pronounced. They conclude by showing that incorporating a *statistical learn-*

ing technique along with the functional annealing algorithm gives the strongest computational results overall.

4.3 Functional Annealing and VLSN Search

Functional annealing provides a way to integrate *very large-scale neighborhood search* techniques within the framework of annealing methods. Since the only condition on the neighborhood search algorithm is that it should be able to produce an improving solution in the neighborhood in a reasonable amount of time, we can easily apply existing VLSN techniques to the functional annealing algorithm.

For instance, the cyclic exchange neighborhood (see Section 2) can be incorporated into the functional annealing algorithm. This neighborhood is too large to be of practical interest with the *pure* simulated annealing algorithm, since the simulated annealing algorithm functions by comparing the performance of random solutions in the neighborhood. The cyclic exchange neighborhood is so large that there is no reason to believe that a random solution will perform well. This problem is alleviated in the functional annealing approach, because it does not rely on the generation of purely random solutions in the neighborhood.

Sharma and Sukhapesna [44, 45] incorporated the cyclic exchange neighborhood in their analysis of functional annealing algorithms for the quadratic assignment problem. They found that in small problem instances, algorithms using the cyclic exchange neighborhood consistently outperformed algorithms based on a 2-opt structure (see Section 2.4). The results for large problem instances were less dramatic.

4.4 Functional Annealing and Timetabling Problems

Functional annealing techniques can be applied to timetabling problems in *much the same way* that simulated annealing algorithms are currently used. (See [18] and [46] for details on the implementation of simulated annealing algorithms in timetabling problems.) Typically, the only restriction on the format of the solutions is that they are represented in such a way that the neighborhood search subroutine can be performed adequately. In the case of the cyclic exchange neighborhood, for instance, we could use the problem structure previously outlined in Section 2.

A main advantage of the functional annealing algorithm is that it allows us to use very large-scale neighborhood search techniques along with annealing algorithms, which have already been used successfully in timetabling problems (see [43] for a survey). For this reason, we believe that this algorithm has a potential to be very valuable to the timetabling community.

5 Concluding Remarks

In this paper, we have discussed one application and two potential applications of very large-scale neighborhood search techniques in examination time-

tabling problems. The applications range from one that has been used before in timetabling problems (the *cyclic exchange neighborhood*), to one that has been widely used in contexts other than timetabling (*optimized crossover* in genetic algorithms), to a relatively new concept that we believe has a great potential for timetabling problems (*functional annealing* algorithms).

Although these applications are presented in the context of examination timetabling, the techniques are general enough to apply to a wide range of timetabling problems. It is our hope that the timetabling community will make use of these techniques and incorporate them into further studies in the timetabling literature. Based on the existing work, we believe that very large-scale neighborhood search techniques may be very useful in the design of new timetabling algorithms.

References

1. S. Abdullah, S. Ahmadi, E. Burke, and M. Dror. Applying Ahuja-Orlin's large neighborhood for constructing examination timetabling solution. In *Proceedings of the Fifth International Conference on the Practice and Theory of Automated Timetabling*, number 3616 in Lecture Notes in Computer Science, pages 413–420, Pittsburgh, PA, 2004. Springer.
2. R. Agarwal, R. Ahuja, G. Laporte, and Z. Shen. A composite very large-scale neighborhood search algorithm for the vehicle routing problem. In *Handbook of Scheduling: Algorithms, Models and Performance Analysis*, chapter 49. Chapman & Hall/CRC, Boca Raton, FL, 2003.
3. R. Agarwal, Ö. Ergun, J. Orlin, and C. Potts. Solving parallel machine scheduling problems with variable depth local search. Working Paper, Operations Research Center, MIT, Cambridge, MA, 2004.
4. C. Aggarwal, J. Orlin, and R. Tai. Optimized crossover for the independent set problem. *Operations Research*, 45:226–234, 1997.
5. R. Ahuja, Ö. Ergun, J. Orlin, and A. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123:75–102, 2002.
6. R. Ahuja, Ö. Ergun, J. Orlin, and A. Punnen. Very large-scale neighborhood search: theory, algorithms, and applications. Working Paper, Operations Research Center, MIT, Cambridge, MA, 2006.
7. R. Ahuja, K. Jha, J. Orlin, and D. Sharma. Very large-scale neighborhood search for the quadratic assignment problem. Working Paper, Operations Research Center, MIT, Cambridge, MA, 2002.
8. R. Ahuja, J. Orlin, and T. Magnanti. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Upper Saddle River, NJ, 1993.
9. R. Ahuja, J. Orlin, and D. Sharma. Very large-scale neighborhood search. *International Transactions in Operational Research*, 7:301–317, 2000.
10. R. Ahuja, J. Orlin, and D. Sharma. Multi-exchange neighborhood structures for the capacitated minimum spanning tree problem. *Mathematical Programming*, 91:71–97, 2001.
11. R. Ahuja, J. Orlin, and A. Tiwari. A greedy genetic algorithm for the quadratic assignment problem. *Computers & Operations Research*, 27:917–934, 2000.

12. R. Alvarez-Valdes, G. Martin, and J. Tamarit. Constructing good solutions for the Spanish school timetabling problem. *Journal of the Operational Research Society*, 47:1203–1215, 1996.
13. P. Avella, B. D'Auria, S. Salerno, and I. Vasil'ev. Computational experience with very large-scale neighborhood search for high-school timetabling. Working Paper, Research Center on Software Technology, Università del Sannio, Italy, 2006.
14. M. Ayob and G. Kendall. A Monte Carlo hyper heuristic to optimise component placement sequencing for multi-head placement machine. In *Proceedings of the Fourth International Conference on Intelligent Technologies*, pages 132–141, Chiang Mai, Thailand, 2003. Institute for Science and Technology Research and Development, Chiang Mai University.
15. E. Balas and W. Niehaus. Finding large cliques in arbitrary graphs by bipartite matching. In *Clique, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, pages 29–53. AMS, 1996.
16. E. Balas and W. Niehaus. Optimized crossover-based genetic algorithms for the maximum cardinality and maximum weight clique problems. *Journal of Heuristics*, 4:107–122, 1998.
17. D. Bertsimas and J. Tsitsiklis. *Introduction to linear optimization*. Athena Scientific, Belmont, MA, 1997.
18. B. Bullnheimer. An examination scheduling model to maximize students' study time. In *Proceedings of the Second International Conference on the Practice and Theory of Automated Timetabling*, number 1408 in Lecture Notes in Computer Science, pages 78–91, Toronto, Canada, 1998. Springer.
19. E. Burke, D. Elliman, and R. Weare. A hybrid genetic algorithm for highly constrained timetabling problems. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 605–610, Pittsburgh, PA, 1995. Morgan Kaufmann.
20. E. Burke, J. Newall, and R. Weare. A memetic algorithm for university exam timetabling. In *Proceedings of the First International Conference on the Practice and Theory of Automated Timetabling*, number 1153 in Lecture Notes in Computer Science, pages 241–250, Edinburgh, United Kingdom, 1996. Springer.
21. E. Burke and S. Petrovic. Recent research directions in automated timetabling. *European Journal of Operational Research*, 140:266–280, 2002.
22. M. Carter. A comprehensive course timetabling and student scheduling system at the University of Waterloo. In *Proceedings of the Third International Conference on the Practice and Theory of Automated Timetabling*, number 2079 in Lecture Notes in Computer Science, pages 64–82, Konstanz, Germany, 2000. Springer.
23. M. Carter and G. Laporte. Recent developments in practical examination timetabling. In *Proceedings of the First International Conference on the Practice and Theory of Automated Timetabling*, number 1153 in Lecture Notes in Computer Science, pages 3–21, Edinburgh, United Kingdom, 1996. Springer.
24. M. Carter and G. Laporte. Recent developments in practical course timetabling. In *Proceedings of the Second International Conference on the Practice and Theory of Automated Timetabling*, number 1408 in Lecture Notes in Computer Science, pages 3–19, Toronto, Canada, 1998. Springer.
25. C. Cheng, L. Kang, N. Leung, and G. White. Investigations of a constraint logic programming approach to university timetabling. In *Proceedings of the First International Conference on the Practice and Theory of Automated Timetabling*, number 1153 in Lecture Notes in Computer Science, pages 112–129, Edinburgh, United Kingdom, 1996. Springer.
26. A. Colorni, M. Dorigo, and V. Maniezzo. Metaheuristics for high-school timetabling. *Computational Optimization and Applications*, 9:275–298, 1998.

27. D. Corne, P. Ross, and H. Fang. Fast practical evolutionary timetabling. In *Proceedings of the Artificial Intelligence and Simulation of Behavior Workshop on Evolutionary Computing*, number 865 in Lecture Notes in Computer Science, pages 251–263, Leeds, United Kingdom, 1994. Springer.
28. V. Deĭneko and G. Woeginger. A study of exponential neighborhoods for the travelling salesman problem and for the quadratic assignment problem. *Mathematical Programming*, 87:255–279, 2000.
29. Ö. Ergun. *New neighborhood search algorithms based on exponentially large neighborhoods*. PhD dissertation, Massachusetts Institute of Technology, June 2001.
30. H. Fang. *Genetic Algorithms in Timetabling and Scheduling*. PhD dissertation, University of Edinburgh, September 1994.
31. M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, NY, 1979.
32. F. Glover. Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics*, 65:223–253, 1996.
33. K. Jha. *Very large-scale neighborhood search heuristics for combinatorial optimization problems*. PhD dissertation, University of Florida, June 2004.
34. W. Junginger. Timetabling in Germany - a survey. *Interfaces*, 16:66–74, 1986.
35. S. Lin and B. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.
36. H. Lourenço, J. Paixão, and R. Portugal. Multiobjective metaheuristics for the bus driver scheduling problem. *Transportation Science*, 35:331–343, 2001.
37. M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.
38. G. Neufeld and J. Tartar. Graph coloring conditions for the existence of solutions to the timetable problem. *Communications of the ACM*, 17:450–453, 1974.
39. A. Punnen and S. Kabadi. Domination analysis of some heuristics for the traveling salesman problem. *Discrete Applied Mathematics*, 119:117–128, 2002.
40. C. Ribeiro and D. Vianna. A genetic algorithm for the phylogeny problem using an optimized crossover strategy based on path relinking. In *Proceedings of the Second Brazilian Workshop on Bioinformatics*, pages 97–102, Macaé, Brazil, 2003. SBC.
41. S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Working Paper, Department of Computer Science, University of Copenhagen, Denmark, 2005.
42. A. Schaerf. Local search techniques for large high school timetabling problems. *IEEE Transactions on Systems, Man, and Cybernetics– Part A: Systems and Humans*, 29:368–377, 1999.
43. A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13:87–127, 1999.
44. D. Sharma and S. Sukhapesna. Functional Annealing technique for very large-scale neighborhood search. In Preparation, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, 2006.
45. S. Sukhapesna. *Generalized annealing algorithms for discrete optimization problems*. PhD dissertation, University of Michigan, September 2005.
46. J. Thompson and K. Dowsland. A robust simulated annealing based examination timetabling system. *Computers and Operations Research*, 25:637–648, 1998.
47. P. Thompson. *Local search algorithms for vehicle routing and other combinatorial problems*. PhD dissertation, Massachusetts Institute of Technology, May 1988.
48. P. Thompson and J. Orlin. The theory of cyclic transfers. Working Paper OR 200-89, Operations Research Center, MIT, Cambridge, MA, 1989.

49. P. Thompson and H. Psaraftis. Cyclic transfer algorithms for multivehicle routing and scheduling problems. *Operations Research*, 41:935–946, 1993.
50. M. Yagiura and T. Ibaraki. Recent metaheuristic algorithms for the generalized assignment problem. In *Proceedings of the Twelfth International Conference on Informatics Research for Development of Knowledge Society Infrastructure*, pages 229–237, Kyoto, Japan, 2004. IEEE Computer Society.
51. M. Yagiura, S. Iwasaki, T. Ibaraki, and F. Glover. A very large-scale neighborhood search algorithm for the multi-resource generalized assignment problem. *Discrete Optimization*, 1:87–98, 2004.