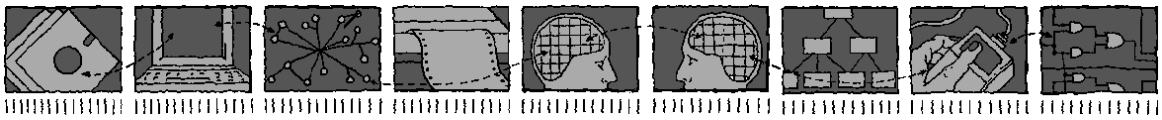
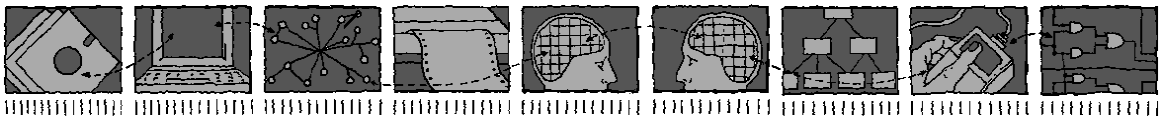


Department of Computing Science and Mathematics
University of Stirling



Augmenting Metaheuristics with Rewriting Systems

Jerry Swan[1], Martin Edjvet[2], Ender Özcan[2]

Technical Report CSM-197

ISSN 1460-9673

January 2014

Department of Computing Science and Mathematics
University of Stirling

Augmenting Metaheuristics with Rewriting Systems

Jerry Swan[1], Martin Edjvet[2], Ender Özcan[2]

Department of Computing Science and Mathematics
University of Stirling
Stirling FK9 4LA, Scotland

Telephone +44 1786 467 421, Facsimile +44 1786 464 551
Email jerry.swan@cs.stir.ac.uk,{martin.edjvet,ender.ozcan}@nottingham.ac.uk

Technical Report CSM-197

ISSN 1460-9673

January 2014

Abstract

We describe the use of a rewriting system to determine equivalence classes over the search-space of optimisation problems. These equivalence classes may be used to derive redundant subsequences in the search-space for incorporation into metaheuristics such as backtracking, genetic algorithms and tabu-search. We use this approach as the basis for a new tabu-search variant — 'Equational-TS' and apply it to the Quadratic Assignment Problem, yielding significant results in terms of the number of iterations to convergence.

1 Introduction

Search techniques involving backtracking can employ constraints on partial solution vectors to reduce the size of the search space. Similarly, the tabu-search metaheuristic ([24], [23]) introduced the mechanism of a 'tabu list' — a method for the injection of domain knowledge via prohibitions on the search trajectory. Following the conventions of the tabu search literature, we consider an *attribute* to be associated with a transition in the state-space graph via some domain-specific *attribute function* with signature $f : S \times O \times S \to A, f : (sourceState, op, targetState) \mapsto attr$ where S is solution state, O is operator and A is attribute. We describe a method for capturing all equivalences implied by the algebraic properties of a set of attributes via the theorem-proving technique known as the *Knuth-Bendix algorithm for Strings* [38]. The output of the Knuth-Bendix algorithm is a *rewriting system* — a set of rules for transforming a sequence of attributes into an equivalent normal form (i.e. a representative for an equivalence class of attribute sequences). These rules are expressed as a regular language [20] and this transformation can then be incorporated into backtracking, tabu-search or any metaheuristic for which there is interest in eliminating redundant attribute sequences. A concrete example takes the set of attributes to be the set of operators: if the algebraic structure of the operators can be expressed in an equational form, we may then determine all cycles in the state-space graph as an offline activity. The ability to reduce operator sequences to their minimal length equivalent is of interest for several reasons:

- To find shorter paths to a solution.
- To avoid the costly application of redundant operations.
- To meet feasibility constrains on operator sequence length.

In this article, we illustrate the application of rewriting systems to the algebraic structure representing the set of permutations of n elements. This structure is known as the *symmetric group on n elements* (denoted S_n) and is a ubiquitous combinatorial-search neighbourhood, being used, for example, to represent solutions to the Travelling Salesperson Problem [14].

Figure 1 gives concrete motication for the application of rewriting systems. It is well-known that any permutation can be expressed as a a sequence of transpositions and Figure 1 depicts the redundancy present in S_n by plotting the length reduction possible for randomly-generated transposition sequences of S_{30}, where redundancy is given by the ratio of reduced length to original length. Redundancy can be observed to be a monotone function of sequence length.

The remainder of this article is organized as follows: in Section 2 we describe the operation of Knuth-Bendix for Strings (KBS) and its application to S_n. In Section 3 we describe several ways in which the redundancy of Figure 1 can be exploited by incorporating rewriting into backtracking and genetic algorithms. In Section 3.4 we introduce a novel tabu-search variant — 'Equational Tabu Search' that uses a rewriting system created offline to derive tabu list entries. In Section 4, we give a specific case study in which Equational Tabu Search is applied to the Quadratic Assignment Problem (QAP) [39]. In Section 4.1 we show that the convergence results for application of Equational-TS to QAP are promising and discuss scalability issues in Section 4.2. In Section 5 we draw some conclusions and discuss the wider applicability of the method.

2 Background

The background for Knuth-Bendix for Strings is most conveniently described in terms of the theory of *finitely-presented monoids*. Informally, a monoid can be considered to be a set of strings (that includes

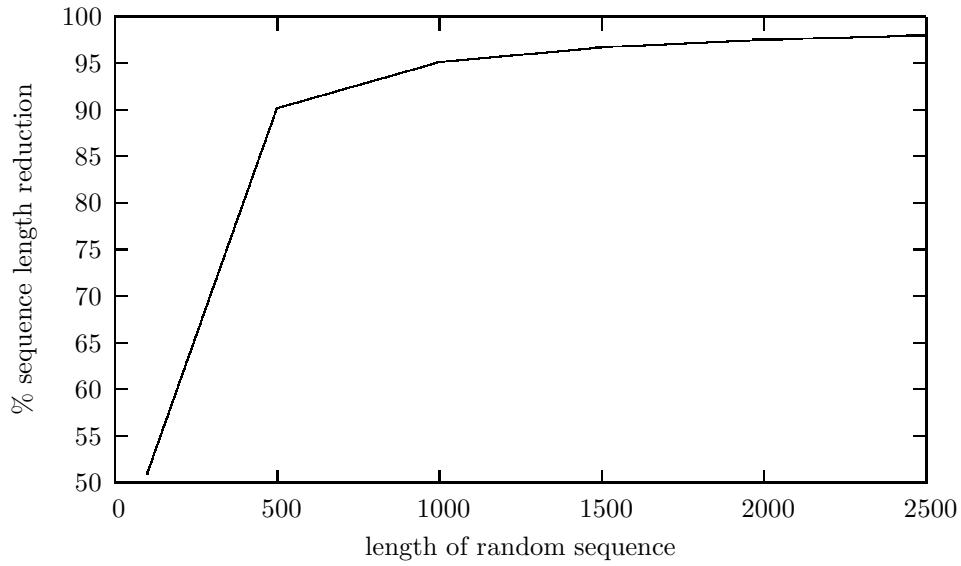


Figure 1: Percentage reduction by maximum sequence length for sets of 10,000 random transposition sequences in S_{30}

the empty string) under concatenation. For a general introduction to the theory of monoid and group presentations, the reader is referred to [51] and [35] respectively. Formally, a monoid is a set of *generators* (a.k.a. the *alphabet*) equipped with an associative binary operation with identity. Let Σ be a finite alphabet. A *word* in Σ is some finite sequence of elements of Σ. Σ^* then denotes the free monoid of words in Σ under concatenation, i.e. the set of all words over Σ. The identity element is the empty word ε. A *congruence* on a monoid M is an equivalence relation $\sim$ on M such that $x \sim y$ implies $wxz \sim wyz$ $\forall w, x, y, z \in M$.

A *finitely-presented monoid* is denoted by $M = Mon\langle\Sigma|R\rangle$, where Σ are the generators of M and R are the *defining relations* for the monoid, i.e. a finite set of equations between words in Σ^*. M is then defined as the quotient monoid $\Sigma^*/\sim_R$ where $\sim_R$ is the congruence on Σ^* of the defining relations R. The elements of M are thus equivalence classes of words.

Finitely-presented groups are defined in an analogous fashion, and are denoted $\langle X|R\rangle$ for *group generators* X, i.e. the existence of inverses is implied for all elements of X. The application of Knuth-Bendix to a finitely-presented group proceeds via the equivalent monoid presentation for the group, in which group inverses are explicitly introduced as generators and the two-sided equations between generators and their inverses are also added to the relations. For example, the free group of order 2 has group presentation $\langle a, b|\rangle$ and corresponding monoid presentation

$$Mon\langle a, A, b, B|a * A = \varepsilon, A * a = \varepsilon, b * B = \varepsilon, B * b = \varepsilon\rangle$$

where $A = a^{-1}, B = b^{-1}$.

2.1 The Knuth-Bendix Algorithm

In order to apply the Knuth-Bendix algorithm to the attribute trajectory, we express the algebraic relationships between the attributes $\Sigma = \{a_1, \ldots, a_k\}$ as a finitely-presented monoid. The domain knowledge we wish to exploit is represented by the finite set of relations $R \in \Sigma^* \times \Sigma^*$, being the axiomatic equations known to hold between the attributes. We also require a *reduction ordering* $\prec$ on Σ^* which is a well-ordering with the additional property of *translation invariance*, i.e. $a \prec b \Rightarrow xay \prec xby$ for all $a, b, x, y \in \Sigma^*$. The pair $\mathcal{R} = (R, \prec)$ is known as a *rewriting system*. The elements of $\mathcal{R}$ are known as *rewrite-rules* and we replace occurrences of l in a word by r. For $u, v \in \Sigma^*$, we write $u \to v$ if there exist words x, y and $(l, r) \in R$ such that $u = xly; v = xry$, i.e. if v is obtained from u by the single application of a rewrite rule. A word u is said to be $\mathcal{R}$-irreducible (or $\mathcal{R}$-reduced) if there is no word

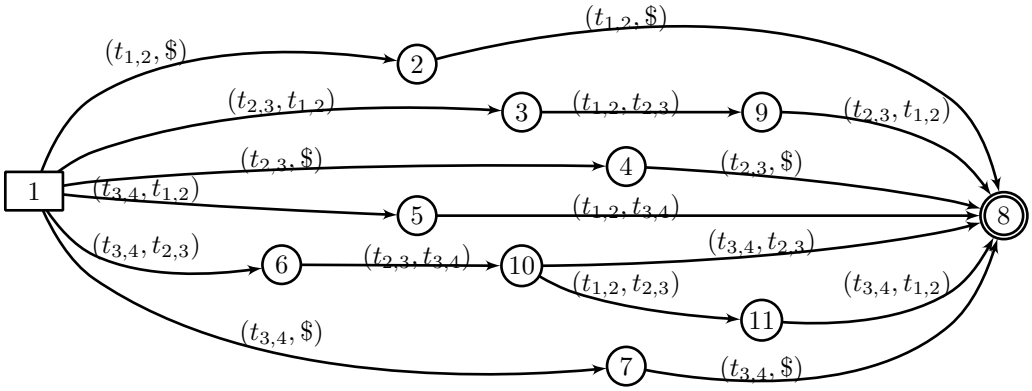


Figure 2: Automaton accepting minimal confluent rewrite rules of S_4

$v \in \Sigma^*$ with $u \to v$. A rewrite rule $(l, r) \in \mathcal{R}$ is said to be $\mathcal{R}$-irreducible if r and all proper substrings of l are $\mathcal{R}$-irreducible. The system $\mathcal{R}$ is said to be *Noetherian* or terminating if there are no infinite rewriting sequences. This implies that any word in Σ^* rewrites to an irreducible. If $\prec$ is a reduction ordering, it follows that $\mathcal{R}$ is Noetherian. $\mathcal{R}$ is said to be *confluent* if all applicable sequences of rewrite rules acting on an expression eventually yield some word that is common to all sequences. If $\mathcal{R}$ is Noetherian and locally confluent then $\mathcal{R}$ is confluent [46].

The Knuth-Bendix algorithm attempts to complete a Noetherian rewrite system via the addition of new rules. The essential element of the Knuth-Bendix algorithm is a constructive test for local confluence. If, during the application of this test, we encounter two words a and b that are equivalent under $\mathcal{R}$ but which rewrite to distinct irreducible words c and d, R is not locally confluent and c is in fact equal to d under the complete set of rewrite rules. The Knuth-Bendix algorithm operates by orienting any such c and d into a new rewrite rule via $\prec$, and repeating the test for local confluence over all pairs of rules. As described, this process may fail to terminate (we are effectively attempting to solve the *word problem for groups*, which is well-known to be formally undecidable [15]) and that success may also depend on the choice of reduction ordering [20]. The existence of a confluent rewriting system is nonetheless well-known for many specific cases, including all finite groups (and hence the application to QAP described below). If dealing with an *infinite* set of attributes for which confluence is not known *a priori*, one may consider obtaining the set of rewrite rules as an offline process taking place prior to the execution of any metaheuristic.

As discussed above, the output of a successful application of Knuth-Bendix is a regular language representing the rewriting system. Fig 2 gives the automaton accepting the minimal confluent rewrite rules for S_4, the symmetric group on 4 generators, using the 'standard presentation' [12] with *shortlex* reduction order, i.e. ordered first by length, then lexicographically according to some given element ordering. The element ordering in this case is taken to be $t_{i,i+1} < t_{i+1,i+2}$:

$$\begin{aligned}\langle t_{1,2}t_{2,3}, \ldots, t_{n-1,n} | (t_{i,i+1}t_{i+1,i+2})^3, 1 \leq i \leq n-2,\\ (t_{i,i+1}t_{j,j+1})^2, 1 \leq i < n-2, i+1 < j \leq n-1,\\ (t_{i,i+1})^2, 1 \leq i \leq n-1\rangle\end{aligned}$$

In Fig. 2, the start state is denoted by a box and the accept state by a double circle. A slight technical complication arises from the possibility that a word may rewrite to another of unequal length. In order to accommodate this, the alphabet for this automaton is the set of pairs of *padded generators* over the alphabet $\Sigma \cup \$$, where $\$$ is the *padding symbol* corresponding to the empty word. In order to obtain a rewrite rule from such automata, the path from a start state to an accept state is 'unzipped', i.e. one appends the first element of each transition label to the left hand side of the rule and the second element to the right hand side of the rule. Hence, the topmost rule in Fig. 2 is given by $t_{1,2} * t_{1,2} \to \$ * \$$, i.e. $t_{1,2}^2 \to \varepsilon$. Fig. 3 gives the entire corresponding set of rewrite rules.

$$t_{1,2}^2 \rightarrow \varepsilon$$
$$t_{2,3}^2 \rightarrow \varepsilon$$
$$t_{3,4}^2 \rightarrow \varepsilon$$
$$t_{2,3}t_{1,2}t_{2,3} \rightarrow t_{1,2}t_{2,3}t_{1,2}$$
$$t_{3,4}t_{1,2} \rightarrow t_{1,2}t_{3,4}$$
$$t_{3,4}t_{2,3}t_{3,4} \rightarrow t_{2,3}t_{3,4}t_{2,3}$$
$$t_{3,4}t_{2,3}t_{1,2}t_{3,4} \rightarrow t_{2,3}t_{3,4}t_{2,3}t_{1,2}$$

Figure 3: Minimal confluent rewrite rules for S_4

```
void backtrack( List< Attribute > c ) {
  if( rewrite( c ) ≺ c or reject( c ) )
    return;

  if( yieldsSolution( c ) )
    output( 'Solution found:' + c );

  List< Attribute > s = c;
  if( s.length() < MAX_ATTRIBUTE_SEQUENCE_LENGTH )
    s.append( leastAttribute() );

  for( ; ; )
  {
    backtrack( s );
    Attribute a = nextAttribute( s.getLastElement() );
    if( a == null )
      break;
    s.setLastElement( a );
  }
}
```

Listing 1: Backtracking over attribute sequences using Knuth-Bendix

3 Application to Metaheuristics

We now describe the application of Knuth-Bendix for Strings to backtracking [32], genetic algorithms [29] and tabu search ([24],[23]).

3.1 Application to Backtracking

Knuth-Bendix for Strings may be used to constrain searches that backtrack over attribute sequences. In order for this to be applicable, the backtracking procedure must generate attribute sequences in the order determined by $\prec$. Each successive attribute sequence s is then reduced to its normal form n. If n precedes s under the reduction ordering $\prec$ we may eliminate s from further consideration since its equivalent normal form has already been encountered. Listing 1 outlines a recursive implementation of backtracking augmented in this fashion. The function *leastAttribute* returns the minimal attribute under $\prec$ and the function *nextAttribute(a)* returns the minimal attribute b such that $a \prec b$ (or *null* if no such attribute exists). Since both of these functions are static properties of the set of attributes, they have time-efficiency $\mathcal{O}(1)$. Any additional domain-specific constraints on attribute sequences may be expressed via the *reject* function.

3.2 Application to Genetic Algorithms

Problems for which the solution is the minimal depth path to a goal state are readily expressed as genetic algorithms in which the genome is a (fixed or variable-length) operator sequence. The application of Knuth-Bendix to rewrite potential solutions is thus a useful way of rejecting infeasible solutions: if we employ a length-preserving $\prec$ (i.e. rewrites have non-increasing word-length — such as shortlex, described above) then if the normal form is shorter, the potential solution contains redundant subsequences and is thus infeasible. For example, in [44] genetic algorithms over operator sequences are applied to a number of potential counterexamples to the longstanding Andrews-Curtis conjecture [2]. To the chagrin of group-theorists and metaheuristic researchers, depth-first search employing secondary storage represents the state-of-the-art in this area ([28], [7], [55]). Since it is well-known that the operators for Andrews-Curtis form a group with a rich algebraic structure, Knuth-Bendix might profitably be used to exploit this domain knowledge and increase the effectiveness of genetic algorithms or other metaheuristics over operator sequences.

3.3 Application to Tabu Search

The essential mechanism of tabu search (termed *recency-based memory*) maintains a restricted local neighbourhood by prohibiting the choice of a transition if (some attribute of) that transition has been encountered recently [24]. The simplest (or *strict-tabu*) implementation implements the recency structure as a sequence of the last k attributes encountered, where k is the *tabu-tenure*. In addition, it is possible to maintain an *absolute-tabu* attribute list that constrains the search-space via problem-specific knowledge. Tabu-search is generally equipped with an *aspiration criterion*, which allows the selection of a transition that would otherwise be prohibited (e.g. if the target state evaluates as superior to any state yet encountered).

3.4 Equational Tabu-Search

We now introduce the *Equational Tabu Search* (Eq-TS), a novel tabu-search mechanism. In Eq-TS, the elements of the tabu-list are normal forms of attribute sequences. As part of its current state, Eq-TS maintains the normal form of the operator sequence obtained from the metaheuristic trajectory. On each successive transition in the state-space graph, the operator associated with that transition is appended to the attribute sequence and the resulting sequence reduced to normal form. Eq-TS thus describes how to create successive elements of the tabu list and is independent (at least at a conceptual level) of the policies used for determining the tabu list size and diversification policy (i.e. whatever combination of static, robust, reactive etc features are employed). Based on the assumption that frequently-encountered attribute sequences are representative of basins of attraction in attribute space, we have elected in this article to employ the mechanisms of reactive tabu search [5, 4] for the other policy aspects of Eq-TS.

Given the relative complexity of TS as a local search strategy, it simplifies both exposition and implementation to use the techniques of *generative programming* [13] to decompose the algorithm into generic policy components. These components may then be combined to yield alternative search algorithms. This is the approach adopted for the HotFrame C++ local search framework [21], and we outline the implementation of Eq-TS with reference to its realization as a HotFrame policy. The invoking framework for the policy is then precisely as described in [21]. Listing 2 contains a minimal HotFrame tabu policy. The policy for Eq-TS is given in listing 3. For brevity, some 'helper' classes are referenced in this listing:

Word: a sequence of operators.

FrequencyElem: a pair of integers representing the iteration number and frequency count associated with an *Attribute*.

MapAttribToFrequencyElem: a mapping from *Attribute* to *FrequencyElem*, implementable (for example) as a red-black tree [11] or as a hashtable [32].

```
class StrictTabuByTrajectory
{
  CircularList< Attribute > recencyList;
public:
  bool isTabu( Attribute a ) const {
    return recencyList.contains( a );
  }

  void add( Attribute a, int currentIteration ) {
    recencyList.add( a );
  }
};
```

Listing 2: HotFrame strict tabu policy

4 Application of Equational Tabu Search to the Quadratic Assignment Problem

The QAP was first introduced in by Koopmans and Beckmann in 1957 and requires a minimal cost assignment of n resources to n locations. It remains one of the hardest optimization problems of common interest — it is known to be NP-hard [49] and exact results are not generally obtainable for instances of $n > 20$. Originally formulated as the *facility location problem* [39], the problem is described as follows. Given two $n \times n$ matrices $A = (a_{ij})$ and $B = (b_{kl})$, find a permutation π on n integers that minimizes the objective function

$$h(\pi) = \Sigma_{i=1}^{n} \Sigma_{j=1}^{n} a_{ij} * b_{\pi_i \pi_j}$$

In the original formulation, a_{ij} is the 'flow' between facilities i and j; b_{kl} represents the distance between locations k and l; π represents the assignment of facilities to locations and the objective function therefore gives the 'transportation-cost' between facilities.

The QAP can be formulated as an integer program with quadratic objective function [39]; as a concave quadratic minimization problem ammenable to cutting-plane approaches [6]; via a trace formulation (for symmetric QAPs) for obtaining lower bounds from eigenvalues ([17],[18]) and (under some additional constraints) reformulated as a linear assignment problem using Kronecker products in [25]. There are a number of alternative linearizations that reformulate the QAP as a mixed-integer linear program ([41], [37], [22],[1]), but this approach is primarily of use in computing lower bounds. Exact solution methods for the QAP include branch and bound and cutting plane methods and the gamut of metaheuristic search methods (including EO, GRASP, tabu and ant systems) have also been applied — the reader is referred to [3, 42, 16] for overviews and to [33, 53, 34, 45] for state-of-the-art results.

The search-space of the QAP of size n is the set of permutations on n elements. As discussed above, this is equivalently denoted by S_n, the symmetric group on n elements. One possible choice of operators is T_n, the set of all pairwise transpositions of elements. It is well-known that T_n can be described as a finitely-presented group that is isomorphic to S_n. The group generators of T_n are $t_{i,j}, 1 \leq i < j \leq n$ with each generator self-inverse. The relations that hold in T_n are as follows (with ε denoting the empty word):

$$[t_{i,j}, t_{k,l}] = \varepsilon \text{ if } |\{i,j\} \cap \{k,l\}| = 0, \text{ for } 1 \leq i < j \leq n, 1 \leq k < l \leq n \quad \text{(R1)}$$

$$(t_{ij} t_{kl})^3 = \varepsilon \text{ if } |\{i,j\} \cap \{k,l\}| = 1, \text{ for } 1 \leq i < j \leq n, 1 \leq k < l \leq n \quad \text{(R2)}$$

$$t_{i,j} t_{j,k} t_{i,j} t_{i,k} = \varepsilon \text{ for distinct } i,j,k, \text{ taking } t_{y,x} \mapsto t_{x,y} \text{ if } y > x \quad \text{(R3)}$$

where the *commutator* $[g,h]$ is defined as $g^{-1}h^{-1}gh$.

We illustrate the application of Knuth-Bendix to T_4, taking $\prec$ as the shortlex ordering previously described. Element order is given by $t_{1,2} < t_{1,3} < t_{1,4} < t_{2,3} < t_{2,4} < t_{3,4}$. We applied the Knuth-Bendix algorithm to this presentation using the software package MAF [58], which is capable of producing a finite-state automaton that accepts precisely the minimal confluent set of rewrite rules (Fig. 4). The isomporphic rewrite rules for T_4 are given in Fig. 5.

```
class EquationalTabu {
  int                          lastReaction;
  Word                         normalAttribSequence;
  List< Word >                 recencyList;
  MapAttribToFrequencyElem     frequencyList;

public:

  bool isTabu( Attribute a ) const
  {
    Word w  = normalAttribSequence.append( a );
    return recencyList.contains( rewrite( w ) );
  }

  void add( Attribute a, int currentIteration )
  {
    normalAttribSequence = rewrite(normalAttribSequence.append(a));

    if( frequencyList.contains( normalAttribSequence ) )
    {
      extendRecencyList();
      lastReaction = currentIteration;
      updateMovingAverage();

      // update iteration and frequency
      frequencyList.get( normalAttribSequence ).update( currentIteration );

      if( number of repeated solutions in frequencyList exceed a threshold
          parameter )
      {
        escape();
      }
    }
    else
    {
      frequencyList.put( normalAttribSequence, FrequencyElem( currentIteration, 1
          ) );
      if( currentIteration - lastReaction > movingAverage )
      {
        shrinkRecencyList();
        lastReaction = currentIteration;
      }
    }
  }
};
```

Listing 3: Equational-Reactive tabu criterion

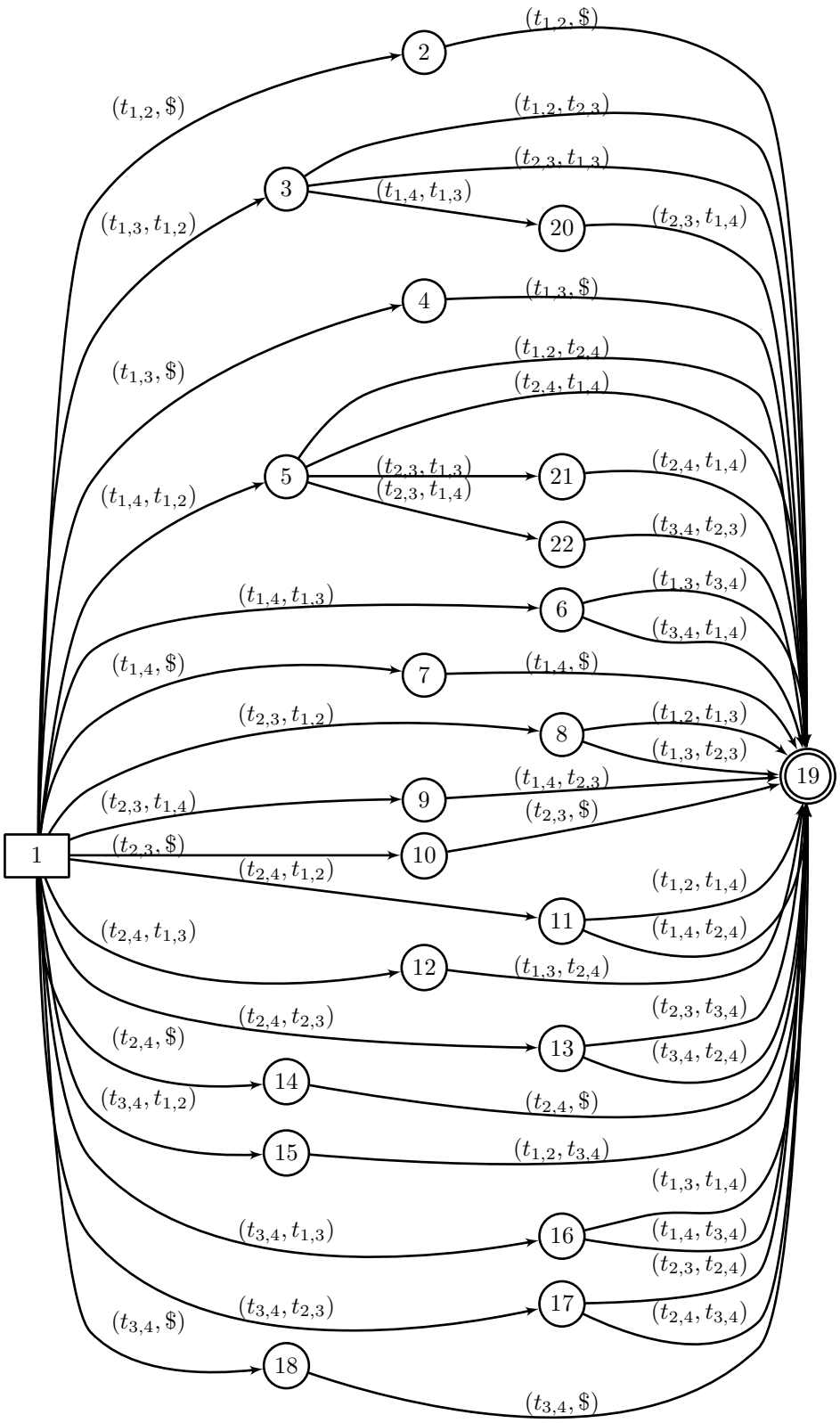


Figure 4: Automaton accepting minimal confluent rewrite rules of T_4

$$
\begin{aligned}
&t_{1,2}^2 \to \varepsilon\\
&t_{1,3}^2 \to \varepsilon\\
&t_{1,4}^2 \to \varepsilon\\
&t_{2,3}^2 \to \varepsilon\\
&t_{2,4}^2 \to \varepsilon\\
&t_{3,4}^2 \to \varepsilon\\
&t_{1,3}t_{1,2} \to t_{1,2}t_{2,3}\\
&t_{1,3}t_{1,4}t_{2,3} \to t_{1,2}t_{1,3}t_{1,4}\\
&t_{1,3}t_{2,3} \to t_{1,2}t_{1,3}\\
&t_{1,4}t_{1,2} \to t_{1,2}t_{2,4}\\
&t_{1,4}t_{2,3}t_{2,4} \to t_{1,2}t_{1,3}t_{1,4}\\
&t_{1,4}t_{2,3}t_{3,4} \to t_{1,2}t_{1,4}t_{2,3}\\
&t_{1,4}t_{2,4} \to t_{1,2}t_{1,4}\\
&t_{1,4}t_{1,3} \to t_{1,3}t_{3,4}\\
&t_{1,4}t_{3,4} \to t_{1,3}t_{1,4}\\
&t_{2,3}t_{1,2} \to t_{1,2}t_{1,3}\\
&t_{2,3}t_{1,3} \to t_{1,2}t_{2,3}\\
&t_{2,3}t_{1,4} \to t_{1,4}t_{2,3}\\
&t_{2,4}t_{1,2} \to t_{1,2}t_{1,4}\\
&t_{2,4}t_{1,4} \to t_{1,2}t_{2,4}\\
&t_{2,4}t_{1,3} \to t_{1,3}t_{2,4}\\
&t_{2,4}t_{2,3} \to t_{2,3}t_{3,4}\\
&t_{2,4}t_{3,4} \to t_{2,3}t_{2,4}\\
&t_{3,4}t_{1,2} \to t_{1,2}t_{3,4}\\
&t_{3,4}t_{1,3} \to t_{1,3}t_{1,4}\\
&t_{3,4}t_{1,4} \to t_{1,3}t_{3,4}\\
&t_{3,4}t_{2,3} \to t_{2,3}t_{2,4}\\
&t_{3,4}t_{2,4} \to t_{2,3}t_{3,4}
\end{aligned}
$$

Figure 5: Minimal confluent rewrite rules for T_4

n	time(s)	rewriting-fsa states	rewriting-fsa equations
10	< 1	103	538
20	16.56	498	5,378
30	398.42	1,193	19,518
40	3,899	2,188	47,958
50	23,498	3,483	95,698
60	99,430	5,078	167,738
70	338,646	6,973	269,078

Table 1: Computation time for $(T_n, \prec_R)$ using MAF utility AUTOMATA -NOWD

Shortlex ordering enjoys a number of useful theoretical and practical properties and is therefore generally to be preferred as a reduction ordering [19]. However, for groups such as T_n that have large numbers of generators, Knuth-Bendix using shortlex may quickly become intractable: using MAF, we were only able to obtain a confluent rewriting system for $n \leq 10$. Further results were therefore obtained using a *wreath-product* reduction ordering [51], denoted $\prec_R$. Table 1 gives the time in seconds required by MAF's AUTOMATA program to produce a confluent rewriting system for T_n $(n = 10 - 70)$ under a wreath-product ordering (all computations in this article were performed on a Pentium® 3 GHz PC). Once a rewriting system for T_n is obtained, it is of course applicable to any problem with an isomorphic set of operators (e.g. TSP [26], appropriate variants of Group-Shop scheduling [50] etc). Rewriting systems successfully obtained for T_n can therefore be made available as a resource of general utility.

4.1 Results

Problem instances were taken from the QAPlib repository [9]. As a benchmark for comparing the performance of Eq-TS, we elected to use the publicly-available implementation of *robust tabu search* (Ro-TS) for the QAP due to Taillard [56], since it features prominently as a method of obtaining the optimum\universally best-known solutions across many QAPlib instances.

Ro-TS and Eq-TS were applied to a number of instances for up to 100,000 iterations (terminating prematurely if the optimum value is obtained) and the performance averaged over 50 runs. The results are summarised in Table 2, the columns of which are given by:

%-converged Percentage of samples that converge to the best-known value.

mean$(d_{\mathbf{opt}})$ Mean of the distance to the optimal (or universally best-known) value.

sd$(d_{\mathbf{opt}})$ Standard deviation of the distance to the optimal (or universally best-known) value.

mean$(i_{\mathbf{best}})$ Mean of the number of iterations to the best value obtained.

sd$(i_{\mathbf{best}})$ Standard deviation of the number of iterations to the best value obtained.

Some other instances analysed for which convergence was almost immediate for both algorithms (e.g. some members of the esc32 family and the entire esc16 family) are omitted from these results.

Eq-TS can be seen from Table 2 to exhibit a higher percentage of convergence in 8 of the 15 cases, an equal percentage in 6 cases and a lower percentage in 1 case. All the cases in which the convergence percentage is equal have a convergence of 100%.

To further illustrate the nature of the differences in convergence, we compared the underlying distributions for d_{opt} and i_{best} according to the confidence intervals of the Mann-Whitney nonparametric test [30]. We tabulate entries defined as follows: $=$ denotes equal distributions according to this test, $+$ or $-$ respectively indicating which of Eq-TS or Ro-TS is to be preferred. Since some entries lie in the 90-95% confidence range, the results are split into Tables 3 and 4, which are derived from 95 and 90 percent confidence intervals respectively.

From Tables , 3 and 4 we therefore conclude that Eq-TS is to be preferred in terms of the solution quality and\or number of iterations to convergence in 11 of the 15 instances studied.

Instance	Alg	%-converged	$mean(d_{opt})$	$sd(d_{opt})$	$mean(i_{best})$	$sd(i_{best})$
bur26a	Ro-TS	98	22.12	154.84	21283.7	21114.6
bur26a	Eq-TS	100	0	0	7968.96	7389.09
bur26b	Ro-TS	98	129.26	904.82	21774.6	23095
bur26b	Eq-TS	100	0	0	6640.4	6026.57
bur26c	Ro-TS	92	484.76	2306.46	19627.7	22242.9
bur26c	Eq-TS	100	0	0	6141.88	5977.97
bur26d	Ro-TS	94	123.14	846.347	14791.5	13466.4
bur26d	Eq-TS	100	0	0	3608.3	3793.95
bur26e	Ro-TS	96	233.48	1583.73	14574.7	20907
bur26e	Eq-TS	100	0	0	2870.58	2621.33
bur26f	Ro-TS	90	539.36	2515	13703.7	17895.6
bur26f	Eq-TS	100	0	0	1898.86	1842.45
bur26g	Ro-TS	100	0	0	14377	12913.4
bur26g	Eq-TS	100	0	0	5467.22	5378.24
bur26h	Ro-TS	98	14.48	101.36	14366.5	18498.8
bur26h	Eq-TS	100	0	0	1681.2	1798.68
chr12a	Ro-TS	100	0	0	1443.96	1056.32
chr12a	Eq-TS	100	0	0	1079.14	757.658
esc32b	Ro-TS	100	0	0	4899.04	4243.36
esc32b	Eq-TS	100	0	0	1076.44	867.815
esc32d	Ro-TS	100	0	0	3381.84	1814.14
esc32d	Eq-TS	100	0	0	207.32	183.05
kra30a	Ro-TS	84	190.4	436.261	20664.3	17045.2
kra30a	Eq-TS	86	108	302.549	38921.2	25149.1
lipa30a	Ro-TS	100	0	0	4659.18	4645.88
lipa30a	Eq-TS	96	3.76	18.6629	20083	17152.2
lipa30b	Ro-TS	100	0	0	247.2	526.157
lipa30b	Eq-TS	100	0	0	266.42	314.133
nug25	Ro-TS	100	0	0	3182.04	3216.24
nug25	Eq-TS	100	0	0	11230.8	10288.2

Table 2: Convergence metrics for QAPlib problem instances using Tabu Search variants

Instance	d_{opt}	i_{best}
bur26a	=	+
bur26b	=	+
bur26c	+	+
bur26d	+	+
bur26f	+	+
bur26g	=	+
bur26h	+	+
chr12a	=	+
esc32b	=	+
esc32d	=	+

Table 3: Results of Mann-Whitney test (95% confidence) on distributions of Ro-TS versus Eq-TS for QAP problem instances

Instance	d_{opt}	i_{best}
bur26e	+	+
kra30a	=	-
lipa30a	-	-
lipa30b	=	-
nug25	=	-

Table 4: Results of Mann-Whitney test (90% confidence) on distributions of Ro-TS versus Eq-TS for QAP problem instances

4.2 Scalability

As discussed above, the term-rewriting aspect consists of two phases: a) the offline application of the KBMAG procedure to construct the automata representing the rewriting system b) the online application of rewriting (which is also employed for the confluence test within the offline application KBMAG) to reduce a word to normal form.

With respect to the above presentation for S_n, the current implementation of MAF can yield a rewriting system for n up to 70, before failing due to resource limitations (addressable memory is currently a maximum of 4 GB on all supported platforms). Local implementation concerns such as memory issues notwithstanding, it is clearly desirable to be able to derive a RWS for much larger values of S_n, thereby opening up the possibility of application to problems such as the Travelling Salesperson [14].

We therefore proceed to describe the efficiency-critical aspects of the KBMAG procedure and outline possible alternative implementations.

4.3 Efficiency of Rewriting

The function *rewrite(c)* returns the normal form for c with respect to $\prec$. In the most general case, *rewrite(c)* has time-efficiency $\mathcal{O}(|c|^2)$, but for finite monoids, rewriting can be achieved by coset-table lookup in $\mathcal{O}(|c|)$ [57].

When rewriting, we want the longest suffix of w (if any) that is the left side of a rule. For finding overlaps, we want the longest suffix of w that is a prefix of a rule. To achieve this efficiently requires the use of an *index structure* for the rewriting system. A variety of term indexing strategies exist, each exhibiting a particular space and time tradeoff [51, 52]. An *index automaton* recognises the set of all irreducible words and represents an extremely time-efficient search strategy for rewriting (it may also be augmented for for finding overlaps). However index automata are also very memory-intensive and are therefore less suitable for very large sets of rewrite rules.

An alternative indexing strategy employing *generalized suffix trees* is given in [54]. Having constructed a generalized suffix tree then, for all pairs of left sides (l_1, l_2), finding the longest prefix of l_1 that is a suffix of l_2 can be achieved in (sequential) linear time. Hariharan gives an optimal parallel construction algorithm for suffix trees that exhibits logarithmic speedup [27].

In general, it is well-known that Knuth-Bendix is highly ammenable to parallelism [48], with nearly linear speedup reported by [59]. In particular, more than one left side may be a subword of w. Bündgen et al [8] give a parallel implementation of rewriting with speedup that is linear in the number of matching left sides.

4.4 Efficiency of the Automata Construction

The most expensive aspect of the automata construction involves the *composite* operation on two DFAs M_x, M_y, an operation that is derived from the Cartesian product of the automata [31]. This requires the determinization of an NFA, and is therefore asymptotically $O(2^{|S|})$. Composite can be seen to be associative, and we define M_w for $w \in \Sigma^*$ by repeated application. The verification process requires that we form M_r for all relators r. It is possible to achieve a logarithmic reduction in the number of composite operations by factoring out common subexpressions in the relators [54].

5 Conclusion

We described the use of a completion theorem prover (the Knuth-Bendix algorithm as employed within the KBMAG procedure) to derive all equations that hold in the monoid structure of a set of attributes associated with the metaheuristic trajectory.

We introduced 'Equational-TS' — a tabu search variant that uses rewriting to derive a tabu list of equivalent attribute sequences and applied it to the Quadratic Assignment Problem. The results show that this technique leads to convergence with fewer evaluations of the objective function as compared to 'Robust-TS' (an approach known to be effective with QAP).

Alternative implementation strategies for improved efficiency were also discussed. In addition to theoretical results regarding the Knuth-Bendix algorithm [36] that imply efficiency gains, the overall performance of KBMAG is strongly driven by the choice of reduction heuristics [19]. Beyond the application of the known efficiency improvements we describe, further investigation of reduction heuristics together with combination of these approaches could make much larger problem instances accessible.

In many application domains, it is well-known that the evaluation of the objective function is often the dominant computational cost in state-space search, these results encourage investigation into the further incorporation of algebraic domain knowledge. In addition, problems in which the attribute algebra has a richer set of equations than that of the QAP would exploit the potential of the rewriting mechanism more fully.

If the algebraic structure of the attributes is no more descriptive than a monoid, we may use this method to obtain an efficiency improvement, e.g. in an application of genetic algorithms in which the genome represents an operator sequence, we can employ our confluent set of rewrite rules to transform it into its equivalent length-minimal sequence. If the algebraic structure of our attributes is indeed a group (rather than merely a monoid), we can turn these relations into relators (i.e. ensure that the right-hand side of each rewrite-rule is the empty word by mulitplying both sides by the inverse of the right-hand side). By this means, we can determine all redundant attribute sequences and may use these sequences to prune the search-space in backtracking, via tabu lists or other representations where the elimination of redundancy is required for purposes of efficiency or feasibility. Further generalizing the nature of the algebraic structure leads to wider applications, for example in genetic programming [40] and hyper-heuristics [10]: normal forms may be used to combat bloat or minimize evaluation cost in genetic programming [43] or grammatical evolution [47].

References

[1] W. P. Adams and T. A. Johnson. Improved linear programming-based lower bounds for the Quadratic Assignment Problem. In P. M. Pardalos and H. Wolkowicz, editors, *Quadratic Assignment and Related Problems*, volume 16 of *Series on Discrete Mathematics and Theoretical Computer Science*, pages 43–75. DIMACS, Providence, RI, 1994.

[2] J. J. Andrews and M. L. Curtis. Free groups and handlebodies. *Proceedings of the American Mathematical Society*, 16(2):192–195, 1965.

[3] Kurt M. Anstreicher. Recent advances in the solution of quadratic assignment problems. *Math. Program.*, 97(1-2):27–42, 2003.

[4] Roberto Battiti. Reactive search: Toward self–tuning heuristics. In V. J. Rayward-Smith, I. H. Osman, C. R. Reeves, and G. D. Smith, editors, *Modern Heuristic Search Methods*, pages 61–83. John Wiley & Sons Ltd., Chichester, 1996.

[5] Roberto Battiti and Giampietro Tecchiolli. The reactive tabu search. *ORSA Journal on Computing*, 6(2):126–140, Spring 1994.

[6] Mokhtar S. Bazaraa and Hanif D. Sherali. On the use of exact and heuristic cutting plane methods for the Quadratic Assignment Problem. *J Oper Res Soc*, 33(11):991–1003, 1982.

[7] R. Sean Bowman and Stephen B. McCaul. Fast searching for Andrews-Curtis trivializations. *Experimental Mathematics*, 15(3), 2006.

[8] Reinhard Bündgen, Manfred Göbel, and Wolfgang Küchlin. A fine-grained parallel completion procedure. In *Proceedings of the international symposium on Symbolic and algebraic computation*, ISSAC '94, pages 269–277, New York, NY, USA, 1994. ACM.

[9] R. E. Burkard, S. Karisch, and F. Rendl. QAPLIB - a Quadratic Assignment Problem library. *European Journal of Operational Research*, 55(1):115–119, November 1991.

[10] Edmund K. Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and John R. Woodward. A classification of hyper-heuristic approaches. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research and Management Science*, pages 449–468. Springer US, 2010.

[11] Thomas H. Cormen, E. Leiserson, Charles, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[12] H. S. M. Coxeter and W. O. J. Moser. *Generators and relations for discrete groups*, volume 14 of *Ergebnisse der Mathematik und ihrer Grenzgebiete [Results in Mathematics and Related Areas]*. Springer-Verlag, Berlin, fourth edition, 1980.

[13] Krzysztof Czarnecki and Ulrich Eisenecker. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley Professional, 2000.

[14] G Dantzig, R Fulkerson, and S Johnson. Solution of a large-scale traveling-salesman problem. *Operations Research*, 2:393–410, 1954.

[15] M. Dehn. Über unendliche diskontinuierliche Gruppen. *Mathematische Annalen*, 71:116–144, 1911. 10.1007/BF01456932.

[16] Zvi Drezner, Peter M. Hahn, and Éric D. Taillard. Recent advances for the quadratic assignment problem with special emphasis on instances that are difficult for meta-heuristic methods. *Annals OR*, 139(1):65–94, 2005.

[17] C.S. Edwards. The derivation of a greedy approximator for the Koopmans-Beckmann quadratic assignment problem. *Proc. CP77 Combinatorial Prog. Conf.*, pages 55–86, 1977.

[18] C.S. Edwards. A branch and bound algorithm for the Koopmans-Beckman quadratic assignment problem. *Mathematical Programming Study*, 13:35–52, 1980.

[19] D. B. A. Epstein, D.F. Holt, and S.E. Rees. The Use of Knuth-Bendix Methods to Solve the Word Problem in Automatic Groups. *Journal of Symbolic Computation*, 12:397–414, 1991.

[20] David B. A. Epstein, M. S. Paterson, G. W. Camon, D. F. Holt, S. V. Levy, and W. P. Thurston. *Word Processing in Groups*. A. K. Peters, Ltd., Natick, MA, USA, 1992.

[21] Andreas Fink and Stefan Voß. Hotframe: A heuristic optimization framework. In S. Voßand D. Woodruff, editors, *Optimization Software Class Libraries*, OR/CS Interfaces Series, pages 81–154, Boston, 2002. Kluwer Academic Publishers.

[22] A.M. Frieze and J. Yadegar. On the quadratic assignment problem. *Discrete Applied Mathematics*, 5:89–98, 1983.

[23] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997.

[24] Fred Glover. Tabu Search - Part I. *INFORMS Journal on Computing*, 1(3):190–206, 1989.

[25] Alexander Graham. *Kronecker Products and Matrix Calculus With Applications*. Ellis Horwood Ltd, 1981.

[26] Gregory Gutin and Abraham Punnen. The traveling salesman problem. *Discrete Optimization*, 3(1), 2006.

[27] Ramesh Hariharan. Optimal parallel suffix tree construction. In *Proceedings of the Twenty-sixth Annual ACM Symposium on Theory of Computing*, STOC '94, pages 290–299, New York, NY, USA, 1994. ACM.

[28] George Havas and Colin Ramsay. Breadth-first search and the Andrews-Curtis conjecture. *IJAC*, 13(1):61–68, 2003.

[29] John H. Holland. *Adaptation in natural and artificial systems.* MIT Press, Cambridge, MA, USA, 1992.

[30] Myles Hollander and Douglas A. Wolfe. *Nonparametric Statistical Methods, 2nd Edition.* Wiley-Interscience, 2 edition, January 1999.

[31] Derek F. Holt, Bettina Eick, and Eamonn A. O'Brien. *Handbook of computational group theory.* Chapman and Hall/CRC, London, 2005.

[32] Ellis Horowitz and Sartaj Sahni. *Fundamentals of Computer Algorithms.* Computer Science Press, 1978.

[33] Mohamed Saifullah Hussin and Thomas Stützle. Hierarchical iterated local search for the quadratic assignment problem. In Maria J. Blesa, Christian Blum, Luca Di Gaspero, Andrea Roli, Michael Sampels, and Andrea Schaerf, editors, *Hybrid Metaheuristics*, volume 5818 of *Lecture Notes in Computer Science*, pages 115–129. Springer, 2009.

[34] Tabitha L. James, Cesar Rego, and Fred Glover. A cooperative parallel tabu search algorithm for the quadratic assignment problem. *European Journal of Operational Research*, 195(3):810–826, 2009.

[35] D.L. Johnson. *Presentations of Groups*, volume 15 of *London Math. Soc. Stud. Texts.* Cambridge University Press, Cambridge, 1990.

[36] Deepak Kapur, David R. Musser, and Paliath Narendran. Only prime superpositions need be considered in the knuth-bendix completion procedure. *J. Symb. Comput.*, 6(1):19–36, August 1988.

[37] L. Kaufman and F. Broeckx. An algorithm for the quadratic assignment problem using benders' decomposition. *European Journal of Operational Research*, 2:204–211, 1978.

[38] D.E. Knuth and P.B. Bendix. Simple word problems in universal algebras. In *Computational Problems in Abstract Algebra*, pages 263–297, 1970.

[39] Tjalling C. Koopmans and Martin Beckmann. Assignment Problems and the Location of Economic Activities. *Econometrica*, 25(1):53–76, 1957.

[40] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems).* The MIT Press, 1 edition, 1992.

[41] E.L. Lawler. The quadratic assignment problem. *Management Science*, 9:586–599, 1963.

[42] Eliane Maria Loiola, Nair Maria Maia de Abreu, Paulo Oswaldo Boaventura-Netto, Peter Hahn, and Tania Querido. A survey for the Quadratic Assignment Problem. *European Journal of Operational Research*, 176:657–690, 2007.

[43] Sean Luke and Liviu Panait. A comparison of bloat control methods for genetic programming. *Evol. Comput.*, 14:309–344, 2006.

[44] Alexei D. Miasnikov. Genetic algorithms and the Andrews-Curtis conjecture. *IJAC*, 9:671–686, 1999.

[45] Alfonsas Misevicius, Antanas Lenkevicius, and Dalius Rubliauskas. Iterated tabu search: an improvement to standard tabu search. *Information Technology and Control*, 35:187–197, 2006.

[46] M.H.A. Newman. On theories with a combinatorial definition of equivalence. *Annals of Mathematics*, 43(2):223–243, 1942.

[47] Michael O'Neill and Conor Ryan. Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):349–358, 2001.

[48] A. Pelin and W. Kraynek. The knuth-bendix algorithm in parallel architecture. In *Southeastcon '89. Proceedings. Energy and Information Technologies in the Southeast., IEEE*, pages 645 –648 vol.2, apr 1989.

[49] Sartaj Sahni and Teofilo Gonzalez. P-Complete Approximation Problems. *J. ACM*, 23(3):555–565, 1976.

[50] Michael Sampels, Christian Blum, Monaldo Mastrolilli, and Olivia Rossi-Doria. Metaheuristics for group shop scheduling. In *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature*, PPSN VII, pages 631–640, London, UK, UK, 2002. Springer-Verlag.

[51] Charles C. Sims. *Computation with finitely presented groups*. Cambridge University Press, 1994.

[52] Charles C. Sims. Users manual for the rutgers knuth-bendix package version 1.30, 1997.

[53] Thomas Stutzle. Iterated local search for the quadratic assignment problem. *European Journal of Operational Research*, 174(3):1519–1539, November 2006.

[54] Jerry Swan. Efficiency issues in the KBMAG procedure. *Journal of Logic and Algebraic Programming*, 80(8):444 – 452, 2011.

[55] Jerry Swan, Gabriela Ochoa, Graham Kendall, and Martin Edjvet. Fitness Landscapes and the Andrews-Curtis Conjecture. *International Journal of Algebra and Computation*, 22(02):1250009, 2012.

[56] Éric D. Taillard. Robust taboo search for the Quadratic Assignment Problem. *Parallel Computing*, 17(4-5):443–455, 1991.

[57] J. A. Todd and H. S. M. Coxeter. A practical method for enumerating cosets of a finite abstract group. *Procedings of the Edinburgh Mathematical Society (2)*, 5:25–34, 1936.

[58] A. Williams. Monoid Automata Factory, version 2.0.3. software package available from http://sourceforge.net/projects/maffsa/, 2010.

[59] Katherine A. Yelick and Stephen J. Garland. A parallel completion procedure for term rewriting systems. In *Conference on Automated Deduction*, pages 109–123. Springer Verlag, 1992.