Enhanced Forma Analysis of Permutation Problems

Tao Gong
Department of Computing
School of Informatics, City University of London
London EC1V 0HB, United Kingdom
t.gong@soi.city.ac.uk

Andrew L. Tuson
Department of Computing
School of Informatics, City University of London
London EC1V 0HB, United Kingdom
andrewt@soi.city.ac.uk

ABSTRACT

Forma analysis provides an approach to formally derive domain specific operators based on domain-independent operator templates by manipulating a set of equivalence relations (i.e., the basis), which is used to describe the search space. In the case of permutation problems, where the basis is highly constrained, the declarative nature of forma analysis encounters some difficulties which give rise to some additional issues, such as the interpretation of declarative constraints and the complexity of the application of operator. This paper aims to address these issues by introducing *Enhanced Forma Analysis* that explores a broader view of forma analysis by using ideas from constraint satisfaction.

Categories and Subject Descriptors

F.2.2 [**Theory of Computation**]: ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY—*Nonnumerical Algorithms and Problems*

General Terms

Theory

Keywords

Forma analysis, constraint satisfaction, permutation problems

1. INTRODUCTION

Forma analysis [7] has been introduced as a formalism to rigorously utilize domain knowledge in evolutionary operator design, in which case the domain knowledge can be incorporated in a declarative manner by using equivalence relations to describe the search space. This paper aims to present a principled design methodology that extends forma analysis to properly analyze and derive operators for permutation problems: *Enhanced Forma Analysis* (EFA).

Permutation problems (e.g., the TSP) are a set of highly constrained discrete problems, such that the basis is not orthogonal. In other words, not all combinations of equivalence classes (formae) are valid permutations. Although specialized operators for permutation problems have been designed under various purposes [2, 1], a principled design methodology can still be very beneficial in the sense that effective operators can be obtained in a more accurate and concrete manner.

Since permutation representations induce constraints on the basis to guarantee feasibility, we propose that operators can be understood as a process of constraint satisfaction. In other words, concrete operators are obtained based on the forma analysis operator specifications being operationalized from a Constraint Satisfaction Problem (CSP) [10] solving perspective[1].

As the forma analysis is purely declarative, issues such as time complexity are hard to analyze. The constraint satisfaction perspective provides us with an enhanced approach to analyze operators from a complementary point of view. Most usefully it allows us to properly define both declarative and procedural semantics to evolutionary operators.

In the following section 2 and 3, we briefly review the previous work in forma analysis and formal descriptions for permutations respectively. Section 4 discusses the difficulties with forma analysis on permutation problems and suggests a CSP approach to enhance our understanding of some other aspects of forma analysis. In section 5 and 6, "concrete" operators for permutation problems are derived based on some basic operator templates with our framework. Section 7 clarifies the consistency of our framework, before section 8 gives our conclusions and suggests some future work.

2. FORMA ANALYSIS

Forma analysis [7] is a formal but practical method that allows the problem representation and its operators to be specified in a formal manner by using *equivalence relations*. Each equivalence relation Ψ divides the search space into disjoint equivalence classes Ξ_ψ (depending which value the solutions match), with individual equivalence classes being denoted by ξ, which gathers solutions that are equivalent under a certain equivalence relation.

The initial aim of forma analysis [7] was to codify knowledge of the problem domain using a set of *equivalence classes* (or *formae*) which is assumed to be able to cluster solutions with related performance in order to guide the search process

[1]Of course, these constraints only exist if we are only interested in searching feasible regions, while search techniques making use of infeasible regions are out of the scope in this discussion.

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

more effectively, e.g. edges if we are considering TSP. Since equivalence relations/classes have the ability to capture the properties of solutions, the operators can thus be mathematically derived by manipulating these equivalence relations in a formal way. Some of the characteristics and operator templates related to forma analysis [7, 9, 11] are given below to facilitate our further discussion.

2.1 Describing the Search Space

The key concept is that of a basis: a set of equivalence relations that allows us to properly describe the search space.

Definition 1. A subset Ψ of a set of equivalence relations is termed as a basis for the set of equivalence relations, if Ψ spans the set and Ψ is independent.

An encoding can thus be derived by taking the image of the basis equivalence classes corresponding to a particular solution in the search space.

2.2 Domain Independent Operator Templates

Forma analysis can derive operators that explicitly manipulate the given equivalence relations. This is achieved by combining the basis with domain independent operators which specify operator behavior in terms of basis. Two of these templates are key to the work presented in this paper.

One such template corresponds to the k-change operator template [11], formally as:

$$O_k(x, k, \Psi) = \{y \in S \mid dis_\Psi(x, y) \leq k\}. \quad (1)$$

Furthermore, we can define a variant of the generalized k-change operator, where revisiting or canceling of the previous moves is not allowed, termed as the *strict* k-change operator, formally as:

$$O_{|k|}(x, k, \Psi) = \{y \in S \mid dis_\Psi(x, y) = k\}. \quad (2)$$

Given a basis Ψ for a set of equivalence relations over a search space S, the Random Transmitting Recombination (RTR) [7] operator template is defined to select a child solution z out of the *dynastic potential* of the parent solutions x and y. RTR(x, y, Ψ) can be formally defined as:

$$\{z \in S \mid \forall \psi \in \Psi : \psi(x, z) = 1 \vee \psi(y, z) = 1\}, \quad (3)$$

where the actual child solution z is chosen from the set above uniformly at random.

3. FORMA ANALYSIS: PERMUTATIONS

According to the literature [11], there are mainly three different representations for permutation that we can adopt to describe the permutation problems: the position-based representation which decides the absolute position of an item, the precedence-based representation which decides whether one task is performed before another, and the adjacency-based representation which decides whether two items are next to each other.

In the following sections, formal descriptions [11] for permutation with a set of n elements $N = \{e_1, \ldots, e_n\}$ will be reviewed in form of equivalence relations, followed by some further definitions of the induced constraints and distance measurements.

3.1 Position-based Description

For position-based description, each position i in the permutation is defined as a equivalence relation ψ_i (for $i = 1, \ldots, n$) to form the basis of *position equivalence relations* such that $\Psi_{pos} = \{\psi_i \mid i \in n\}$. The equivalence classes (formae) for each position i will be the set of n elements $\Xi_{\psi_i} = \{\xi_i^{e_1}, \ldots, \xi_i^{e_n}\}$. For example, the permutation $(1, 4, 3, 2)$ can be described by the set of equivalence classes $\{\xi_1^1, \xi_2^4, \xi_3^3, \xi_4^2\}$.

In addition, an induced feasibility constraint for this description C_{pos} needs to be added to ensure that different elements do not occupy the same position and no two different positions can take the same element, formally as follows.

Definition 2. For any two equivalence relations ψ_i and ψ_j in a permutation, the position-based feasibility constraint C_{pos} can be defined as:

$$\forall i, j \ (i \neq j) : \psi_i \neq \psi_j. \quad (4)$$

A direct implication of this constraint is that the 1-change neighborhood structure is an empty set as this would involve placing two elements in the same position.

The distance metric for this formal description is simply the number of positions in the permutation that have different elements (i.e., the *hamming distance*) according to the our definition of *forma distance* [5]. For example, the distance between $(1, 4, 3, 2)$ and $(1, 4, 2, 3)$ is 2, since they are different in two positions.

3.2 Precedence-based Description

For precedence-based description, a set of basis precedence equivalence relations Ψ_{prec} between any two different elements in the permutation will be considered, formally as:

$$\Psi_{prec} = \{\psi_{prec(e_i,e_j)} \mid e_i, e_j \in N \ \wedge \ e_i \neq e_j\}.$$

However, by considering the fact (or feasibility constraint) that $\psi_{prec(e_i,e_j)}$ and $\psi_{prec(e_j,e_i)}$ are reverse relations

$$\forall e_i, e_j \in N \ (e_i \neq e_j) : \psi_{prec(e_i,e_j)} \Leftrightarrow \neg\psi_{prec(e_j,e_i)},$$

we can remove unnecessary relations by enforcing a sequence (e.g., $e_1 < e_2 < \ldots < e_n$) in the definition of the relation, such that

$$\Psi_{prec} = \{\psi_{prec(e_i,e_j)} \mid e_i, e_j \in N \ \wedge \ e_i < e_j\}.$$

Obviously, the equivalence classes are simply true/false for whether element e_i precedes element e_j in the permutation, formally as:

$$\Xi_{\psi_{prec(e_i,e_j)}} = \{\xi^0_{prec(e_i,e_j)}, \xi^1_{prec(e_i,e_j)}\}.$$

In addition, the feasibility constraint C_{prec} needs to be added that a valid permutation exists if and only if the relationship among the precedences is consistent (in that the transitivity condition is preserved), as shown below.

Definition 3. Given any two equivalence relations in a permutation, say $\psi_{prec(e_i,e_j)}$ and $\psi_{prec(e_j,e_k)}$, the precedence-based feasibility constraint C_{prec} can be defined as:

$$\begin{aligned} &\forall e_i, e_j, e_k \in N \ (e_i \neq e_j \neq e_k) : \\ &(\psi_{prec(e_i,e_j)} \ \wedge \ \psi_{prec(e_j,e_k)} \Rightarrow \psi_{prec(e_i,e_k)}). \end{aligned} \quad (5)$$

The distance metric can thus be specified as the number of different precedence relations between two solutions. For example, the distance between permutation $(1, 2, 3, 4)$ and

$(1,4,2,3)$ can be obtained by comparing the following two sets (with the differences underlined):

$$\{\xi^1_{prec(1,2)}, \xi^1_{prec(1,3)}, \xi^1_{prec(1,4)}, \xi^1_{prec(2,3)}, \underline{\xi^1_{prec(2,4)}}, \underline{\xi^1_{prec(3,4)}}\},$$

$$\{\xi^1_{prec(1,2)}, \xi^1_{prec(1,3)}, \xi^1_{prec(1,4)}, \xi^1_{prec(2,3)}, \underline{\xi^0_{prec(2,4)}}, \underline{\xi^0_{prec(3,4)}}\}.$$

In practice, *bubble sort* will be sufficient to calculate the distance from one permutation towards another. This can be achieved by sorting one permutation towards another, when setting one as the initial permutation and the other as the goal permutation which represents the right "order". If looking at the previous example with two permutations, to calculate the distance from $(1,2,3,4)$ (i.e. the initial permutation) to $(1,4,2,3)$ (i.e. the goal permutation) the "priority" information can be obtained from $(1,4,2,3)$:

$$order(1) = 1, order(4) = 2, order(2) = 3, order(3) = 4.$$

Thus, the initial permutation can be represented (or more closely, implemented) in terms of $(e_i, order(e_i))$:

$$((1,1),(2,3),(3,4),(4,2)),$$

and it should be sorted according to the values of $order(e_i)$ in order to "transform" to the goal permutation.

3.3 Adjacency-based Description

For adjacency-based description, a set of basis adjacency equivalence relations Ψ_{adj} is considered for any two elements to decide whether they are adjacent, formally as:

$$\Psi_{adj} = \{\psi_{adj(e_i,e_j)} \mid e_i, e_j \in N \ \wedge \ e_i \neq e_j\}.$$

Due to the fact that $\psi_{adj(e_i,e_j)}$ and $\psi_{adj(e_j,e_i)}$ are equivalent relations for some *symmetric* problems[2] (as adjacency relation is more meaningful for symmetric permutation problems)

$$\forall e_i, e_j \in N(e_i \neq e_j) : \psi_{adj(e_i,e_j)} \Leftrightarrow \psi_{adj(e_j,e_i)},$$

we can remove redundant relations by enforcing a sequence (e.g., $e_1 < e_2 < \ldots < e_n$) in the definition of the relation:

$$\Psi_{adj} = \{\psi_{adj(e_i,e_j)} \mid e_i, e_j \in N \ \wedge \ e_i < e_j\}.$$

As undirected edges are considered for adjacency-based description, the equivalence classes are simply true/false for whether there exists an edge between element e_i and element e_j, formally as:

$$\Xi_{\psi_{adj(e_i,e_j)}} = \{\xi^0_{adj(e_i,e_j)}, \xi^1_{adj(e_i,e_j)}\}.$$

In this case, $\xi^1_{adj(e_i,e_j)}$ represents a *positive edge* so that edge (e_i, e_j) must exists in the solution (e.g., a tour), while $\xi^0_{adj(e_i,e_j)}$ stands for a *negative edge* so that edge (e_i, e_j) can not be included into the solution.

In addition, the feasibility constraint C_{adj} needs to be added so that each vertex of the undirected graph corresponding to the permutation can only participate in two edges and still be a valid permutation, as follows.

Definition 4. Given any equivalence relations $\psi_{adj(e_i,e_j)}$ in a permutation, the adjacency-based feasibility constraint C_{adj} can be defined as:

$$\forall e_i, e_j, e_k, e_l \in N : \psi_{adj(e_i,e_j)} \ \wedge \ \psi_{adj(e_i,e_k)} \Rightarrow \neg\psi_{adj(e_i,e_l)}. \quad (6)$$

[2]Surry presented an extensive study about *directed edge* representation for permutation in his work [9].

The distance between any two solutions in the search space under adjacency basis is thus calculated as the number of different edge relations they possess. For example, the distance between permutation $(1,2,3,4)$ and $(1,2,4,3)$ can be obtained by calculating the number of different adjacency relations between the following two sets:

$$\{\xi^1_{adj(1,2)}, \underline{\xi^0_{adj(1,3)}}, \underline{\xi^1_{adj(1,4)}}, \underline{\xi^1_{adj(2,3)}}, \underline{\xi^0_{adj(2,4)}}, \xi^1_{adj(3,4)}\},$$

$$\{\xi^1_{adj(1,2)}, \underline{\xi^1_{adj(1,3)}}, \underline{\xi^0_{adj(1,4)}}, \underline{\xi^0_{adj(2,3)}}, \underline{\xi^1_{adj(2,4)}}, \xi^1_{adj(3,4)}\}.$$

However, on the "phenotypical" level this *forma distance* reduces to the number of different *positive* edges (n minus common *positive* edges), which should be 2 in this case. This is mainly because negative edges do not directly affect the quality of solution, although they implicitly affect the selection of positive edges through the feasibility constraints.

4. ENHANCED FORMA ANALYSIS

As noted in the introduction, the formal descriptions (i.e., bases) for permutation all involve some feasibility constraints to guarantee a valid permutation, which is different from the fully-orthogonal case (e.g., the binary case). Since forma analysis advocates specifying operators in a purely declarative manner to manipulate domain knowledge (including feasibility constraints), it hardly reveals anything about the underlying implementation from the computational aspect, which prevents it from being used to analyze some additional issues about the operator design.

In fact, constraint handling plays a very important role in the application of constraint-preservative operators, which usually incurs additional computational costs. For example, Surry [9] introduced the "*patching*" method to handle the incompatible formae. However it failed to address the issue of feasibility constraints properly from the forma analysis viewpoint as it did not fully integrate constraints into the formalization, nor provided an efficient mechanism by which these operators may be operationalized.

Because the application of search operators for permutation inevitably requires satisfying the feasibility constraints, it would be appropriate that we regard the application of search operators as a process of constraints satisfaction. So, any declarative operator with a specified basis and feasibility constraints following the forma analysis formalism corresponds to a constraint satisfaction problem (CSP) [10], such that the computational aspects of forma analysis can be evaluated from the CSP perspective. Since this CSP approach enhances our understanding of some additional (but important) aspects of forma analysis formalism, we call it *Enhanced Forma Analysis* (EFA). In the next section, we will apply our enhanced forma analysis to permutation problems.

5. EFA OF PERMUTATION MUTATION

According to the k-change (mutation) operator template, a distance of k should be applied under a certain basis to generate a new permutation, formally as:

$$dis_{\Psi}(P_1, P_2) \leq k.$$

Now, we are in a position to analyze what operators can be obtained with regard to the aforementioned three formal descriptions of permutation, which should give rise to three

different k-change operators: position-based k-change operator (O_{pos_k}), precedence-based k-change operator (O_{prec_k}), and adjacency-based k-change operator (O_{adj_k}).

5.1 Position-based k-change Operator

For position-based description, since each position in the permutation is defined as an equivalence relation, the distance between two permutations $dis_{\Psi_{pos}}(P_1, P_2)$ is the number of positions where they are different (or the hamming distance under Ψ_{pos}). Thus, we only need to apply a distance of k based on P_1 to generate P_2 such that

$$dis_{\Psi_{pos}}(P_1, P_2) \leq k.$$

For example, given permutation $P_1 = (1, 2, 3, 4)$ such that

$$P_1 = \{\xi_1^1, \xi_2^2, \xi_3^3, \xi_4^4\},$$

what would be the possible solution P_2 so that they have a distance of k?

The most straightforward thought would be to randomly select k equivalence relations and change the equivalence classes they fall into. However, it must not be neglected that the feasibility constraint C_{pos} induced by position-based description should be satisfied to produce a valid permutation. Thus, O_{pos_k} will involve a *constraint satisfaction subproblem*, where C_{pos} must be satisfied to guarantee valid permutation. The CSP we consider corresponds to the operator template O_k with C_{pos} satisfied. Classical CSP techniques can be directly utilized to implement O_{pos_k}. Now, we will illustrate how CSP techniques can be effectively used in the design of O_{pos_k}.

Given P_1 ($\{\xi_1^1, \xi_2^2, \xi_3^3, \xi_4^4\}$), we can first un-instantiate k (e.g., $k = 2$) equivalence relations to produce a partial permutation for a potential distance of 2. This gives us C_4^2 options from which we can uniformly select one to serve as the base (partial permutation) of P_2 (e.g., $\{\xi_1^-, \xi_2^2, \xi_3^-, \xi_4^4\}$).

Then, what we need to do is simply to re-instantiate these 2 equivalence relations to suitable classes such that C_{pos} is satisfied. Figure 1 illustrates briefly how to re-instantiate the partial permutation to a complete permutation.

In Figure 1, the first position ξ_1^- of the partial permutation $(-, 2, -, 4)$ is considered first. The domain of the first position is $\{1, 3\}$, while the domain of the third position is $\{1, 3\}$ as well. After an equivalence class (1 or 3) is chosen for the first position, the domain is reduced for the third position through constraint propagation. After instantiating all the possible solutions, P_2 can be randomly selected among the whole set of feasible solutions to the CSP[3].

However, it is obvious from Figure 1 that "revisit" is not forbidden for each of the un-instantiated positions, which effectively produces a distance no larger than k. To impose "strictness" on the operator where revisit is not allow, we can simply tighten the restriction on the domain available for each position by removing the value taken by P_1 for the corresponding position. Thus, the left branch in Figure 1 where the first position takes the element 1 must be cut off to enforce "strictness", because element 1 has already been removed from the domain of the first position for P_2 (as element 1 has been taken for the first position of P_1).

In addition, it is straightforward to realize that the CSP induced by O_{pos_k} can be solved in a *backtrack-free* manner,

[3]It is of course not necessary that all the feasible solutions to the CSP should be generated to get a potential candidate for P_2.

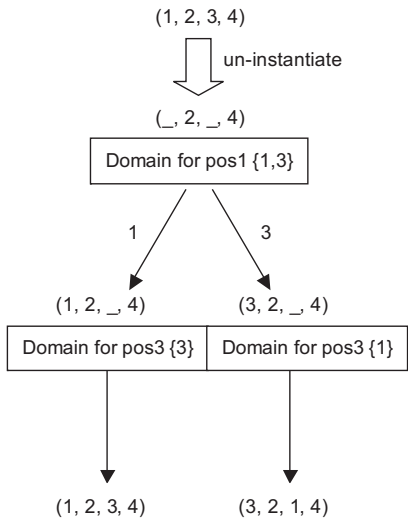

Figure 1: Illustration of the re-instantiation of a partial permutation based on position-based description.

since the domain of the un-instantiated relations will not be empty until the valid permutation is generated and the constraint C_{pos} can be satisfied automatically during the re-instantiation. This is very important if we want to consider some complexity issues of the operator through the CSP we solve.

In fact, the working mechanism of O_{pos_k} has a rather similar effect as the *scramble mutation* [2], which re-arranges a certain number (i.e., k) of positions. The only difference is that the selection of these positions should be purely random, other than inside a continuous block. Moreover, a "strict" 2-change in the positional basis reduces to the traditional *swap mutation* [2], which simply selects two elements and swaps their positions.

5.2 Precedence-based k-change Operator

For precedence-based description of permutation, precedence relations between the elements are considered. The distance between two permutations is the number of different precedence relations $dis_{\Psi_{prec}}(P_1, P_2)$. For example, given two permutation $(1, 2, 3, 4)$ and $(3, 2, 1, 4)$ with the equivalence classes:

$$\{\underline{\xi^1_{prec(1,2)}}, \underline{\xi^1_{prec(1,3)}}, \xi^1_{prec(1,4)}, \underline{\xi^1_{prec(2,3)}}, \xi^1_{prec(2,4)}, \xi^1_{prec(3,4)}\}$$

$$\{\underline{\xi^0_{prec(1,2)}}, \underline{\xi^0_{prec(1,3)}}, \xi^1_{prec(1,4)}, \underline{\xi^0_{prec(2,3)}}, \xi^1_{prec(2,4)}, \xi^1_{prec(3,4)}\}$$

the distance is 3, because they are different in 3 precedence relations (as underlined).

Assuming that a k-change (e.g., $k = 3$) is applied to a permutation $P_1 = (2, 3, 4, 1)$, the product P_2 can be obtained by solving a CSP as shown in Figure 2.

Given that P_1 can be represented as

$$\{\xi^0_{prec(1,2)}, \xi^0_{prec(1,3)}, \xi^0_{prec(1,4)}, \xi^1_{prec(2,3)}, \xi^1_{prec(2,4)}, \xi^1_{prec(3,4)}\},$$

applying a 3-change will be a re-instantiation of 3 equivalence relations. This gives us $C^3_{C^2_n}$ options from which we can uniformly select one as the partial permutation to generate

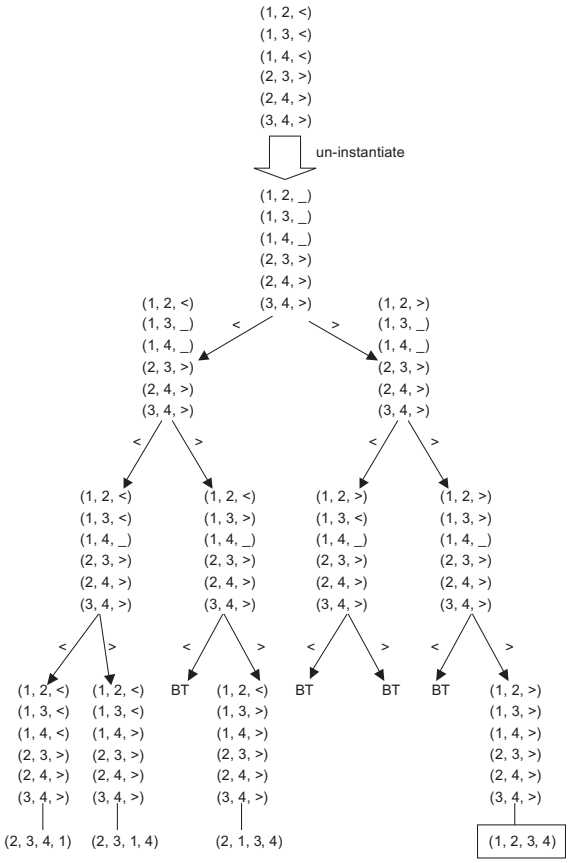

Figure 2: Illustration of the re-instantiation of a permutation with precedence-based description. Symbol "$(i, j, >)$" means element i is before element j, while "$(i, j, <)$" means otherwise. The framed permutation has a "strict" distance of 3 to the initial permutation.

P_2. For example, the partial permutation can be

$$\{\xi^{-}_{prec(1,2)}, \xi^{-}_{prec(1,3)}, \xi^{-}_{prec(1,4)}, \xi^{1}_{prec(2,3)}, \xi^{1}_{prec(2,4)}, \xi^{1}_{prec(3,4)}\}.$$

Re-instantiating these relations to suitable classes such that C_{prec} is satisfied can give us potential candidates for P_2.

As shown in Figure 2, the re-instantiation of precedence relations is carried out one after another. The domain of each relation is simply $\{0, 1\}$. Alternatively we use "$<$" and "$>$" to represent the precedence such that symbol "$(i, j, >)$" means element i is before element j while "$(i, j, <)$" means otherwise. P_2 can be randomly selected among the set of feasible solutions to the CSP for O_{prec_k}.

Again, "revisit" can be forbidden by imposing restriction on the domains of those un-instantiated relations. Due to the fact that precedence relation only takes binary values ($\{0, 1\}$) and the value taken for that relation of P_1 must be removed from the domain of the corresponding relation of P_2, the operator effectively reduces to a "flip" based on the binary values for each of those un-instantiated relations. For example, if 0 is taken by P_1 for $\psi_{prec(i,j)}$, the only possibility for P_2 for $\psi_{prec(i,j)}$ will be 1. Thus, there can be *at most one* P_2 such that the distance is "strict".

However, there are cases that imposing "strictness" on the re-instantiation of P_2 does not give us any feasible solution such that C_{prec} is satisfied. In these cases, back-tracking may have to return to the un-instantiation level to seek for other options to re-instantiate. This is because, *the k un-instantiated relations effectively define the structure of the CSP used to generate* P_2. Alternatively, *relaxation* of "strictness" may also be considered to complete a valid permutation to reduce the computational costs.

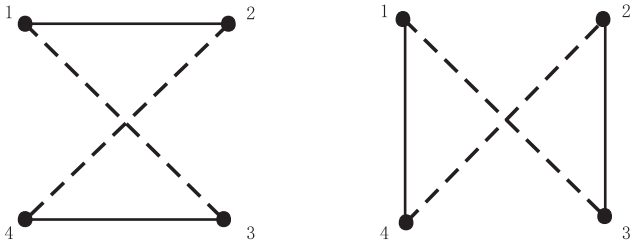

Figure 3: Illustration of edge-difference between two permutations (the dashed lines represent the common edges, while the solid lines represent the different edges).

By observing the effect of changing a single precedence equivalence class, it is not difficult to find that a 1-change (minimal mutation) reduces to the *adjacent swap* mutation [2], which swaps two contiguous elements without violating the feasibility constraint C_{prec}. Thus, O_{prec_k} can be approximately regarded as equivalent to a k-iterated adjacent swap mutation. In addition, the *shift mutation* [2], which removes an element from a position and re-inserts it in another position, can be considered as a constrained version of O_{prec_k} with linkage specialization, such that the k-iterated adjacent swaps are in a fixed sequence.

5.3 Adjacency-based k-change Operator

Since each potential edge is defined as an equivalence relation for adjacency-based description of permutation, the distance of two permutations $dis_{\Psi_{adj}}(P_1, P_2)$ corresponds to the number of different adjacency relations between them. For example, given permutations $(1, 2, 4, 3)$ and $(1, 3, 2, 4)$ (as shown in Figure 3), with the equivalence classes:

$$\{\underline{\xi^{1}_{adj(1,2)}}, \xi^{1}_{adj(1,3)}, \underline{\xi^{0}_{adj(1,4)}}, \underline{\xi^{0}_{adj(2,3)}}, \xi^{1}_{adj(2,4)}, \underline{\xi^{1}_{adj(3,4)}}\}$$

$$\{\underline{\xi^{0}_{adj(1,2)}}, \xi^{1}_{adj(1,3)}, \underline{\xi^{1}_{adj(1,4)}}, \underline{\xi^{1}_{adj(2,3)}}, \xi^{1}_{adj(2,4)}, \underline{\xi^{0}_{adj(3,4)}}\}$$

the distance (more specifically, the "edge-difference") is 4 because there are 4 different edge relations (as underlined) involved in these two permutations.

Suppose we have a permutation $(1, 2, 3, 4)$

$$P_1 = \{\xi^{1}_{adj(1,2)}, \xi^{0}_{adj(1,3)}, \xi^{1}_{adj(1,4)}, \xi^{1}_{adj(2,3)}, \xi^{0}_{adj(2,4)}, \xi^{1}_{adj(3,4)}\},$$

to apply a k-change to P_1 such that $dis_{\Psi_{adj}}(P_1, P_2) \leq k$, we can first un-instantiate k (e.g. $k = 4$) relations (randomly chosen from $C^{4}_{C^{2}_{n}}$ options) to produce the partial permutation for P_2. One of such partial permutations can be:

$$P_2 = \{\xi^{-}_{adj(1,2)}, \xi^{-}_{adj(1,3)}, \xi^{1}_{adj(1,4)}, \xi^{1}_{adj(2,3)}, \xi^{-}_{adj(2,4)}, \xi^{-}_{adj(3,4)}\}.$$

By solving the CSP to generate P_2 such that C_{adj} is satisfied, we can get all the candidate solutions, including the solution

permutation $(1, 3, 2, 4)$ with a "strict" distance of 4 to P_1:

$$\{\xi^0_{adj(1,2)}, \xi^1_{adj(1,3)}, \xi^1_{adj(1,4)}, \xi^1_{adj(2,3)}, \xi^1_{adj(2,4)}, \xi^0_{adj(3,4)}\}.$$

However, observant readers should have realized that not all un-instantiation of adjacency relations can lead to a P_2 with a "strict k" distance to P_1, such as:

$$P_2 = \{\xi^-_{adj(1,2)}, \xi^-_{adj(1,3)}, \xi^-_{adj(1,4)}, \xi^-_{adj(2,3)}, \xi^0_{adj(2,4)}, \xi^1_{adj(3,4)}\}.$$

In this case, we have to sacrifice either *accuracy* or *complexity*. In another word, seeking for a feasible un-instantiation to achieve accuracy may increase the complexity of the algorithm, while relaxation of "strictness" which allows"revisit" (so that accuracy is sacrificed) can highly reduce the complexity.

It is also worthy of mentioning that some value for k (e.g. 1) may not be valid to be applied due to the nature of C_{adj} that adjacency relations are highly linked. This eventually makes the CSP not "*strictly*" solvable when revisit is forbidden, so that no feasible permutation can be produced. In this case, we may either consider that the k is too insignificant (e.g. $k = 1$) to make any difference to the raw permutation P_1 which results in $P_2 = P_1$, or seek for a nearest valid distance as an approximation.

The minimal mutation implied by the adjacency-based description is an edge 2-change mutation[4], because 1-change automatically violates the feasibility constraint C_{adj}. This operator is equivalent to the *edge-reverse mutation* [2] in literature, which reverses the positions of a segment in the permutation in such a manner that the feasibility constraint C_{adj} can be satisfied automatically. In a general sense, an *edge* k-change mutation can be approximated by a k'-iterated edge-reverse mutation with

$$k' = round(k/2),^5$$

as any edge-reverse mutation involves the change of 2 edges.

6. EFA OF PERMUTATION CROSSOVER

As *transmission* [7] is a desirable property that we want to achieve during crossover in order to exploit child solutions with parental material, we will use the RTR operator template to illustrate how enhance forma analysis can be utilized to obtain crossover operators for permutations with the aforementioned formal descriptions, which should give rise to three different RTR operators: position-based RTR operator (RTR_{pos}), precedence-based RTR operator (RTR_{prec}), and adjacency-based RTR operator (RTR_{adj}).

From the CSP perspective, RTR indicates that equivalence classes should be randomly chosen from the parental permutations (e.g. P_{p1} and P_{p2}) to construct (or instantiate) the child permutation P_c such that the the feasibility constraint(s) ($C_{pos}/C_{prec}/C_{adj}$) can be satisfied. In other words, the product of RTR is effectively a random valid permutation chosen from the "constrained" *dynastic potential* of the parents. Given a fully un-instantiated permutation as the partial permutation to generate the child permutation P_c, RTR effectively decides which equivalence class should be taken from which parental permutation to re-instantiate P_c for each relation.

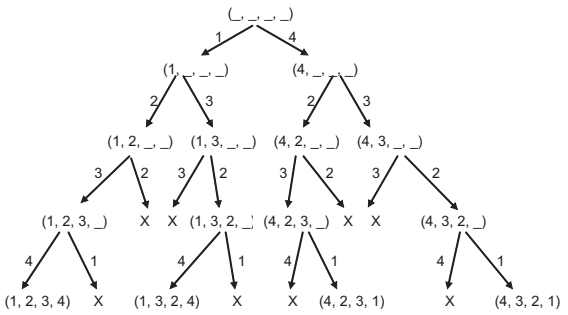

Figure 4: The re-instantiation of P_c for RTR_{pos}.

6.1 Position-based RTR

For position-based description, feasibility constraint C_{pos} has to be satisfied so that no two positions share the same element. This largely reduces the number of feasible solutions to the basic CSP where RTR is interpreted as recombination of parental equivalence classes, because not all combinations of them lead to valid permutations. By satisfying C_{pos} while instantiating P_c, we can obtained all potential candidates for RTR_{pos} as a result of solving the CSP.

For example, given $P_{p1} = (1, 2, 3, 4)$ and $P_{p2} = (4, 3, 2, 1)$, RTR_{pos} can produce potential candidate(s) for P_c by re-instantiating it (as shown in Figure 4) with corresponding equivalence classes, as shown in Equation 7.

$$\begin{array}{lcl} P_{p1} & = & \{\underline{\xi^1_1}, \xi^2_2, \xi^3_3, \underline{\xi^4_4}\} \\ P_{p2} & = & \{\underline{\xi^4_1}, \underline{\xi^3_2}, \underline{\xi^2_3}, \underline{\xi^1_4}\} \\ P_c & = & \{\xi^1_1, \xi^3_2, \xi^2_3, \xi^4_4\} \end{array} \quad (7)$$

To transmit position features from parents to children and interpret the feasibility constraint C_{pos} in a more natural way, we produce a fully-transmitting crossover for permutation, namely *Position Transmitting Crossover* (PTX), by identifying the constraint satisfaction process of operator as a CSP.

In PTX, both C_{pos} and transmitting[6] C_t has to be satisfied as an interpretation of its CSP. In this sense, constrained positions are clustered to function separately. For example, given two permutations

$$(3, 6, \underline{5, 4, 2}, \underline{7, 8}, 1),$$

$$(3, 6, \underline{2, 5, 4}, \underline{8, 7}, 1),$$

we are able to construct the constraint graphs implied by both C_{pos} and transmitting C_t, as shown in Figure 5 with constrained positions linked together.

The construction of the constraint graphs is straightforward to understand–a value that has been taken for one position must be forbidden (constrained) for another position. For example, for position 3 (P_3 in Figure 5) either 5 or 2 should be chosen to achieve transmitting. However, choosing either of them will forbid another position (e.g. P_4 or P_5

[4]A standard edge 2-change mutation involves the "replacement" of 2 edges which in turn results in the change of 4 adjacency equivalence relations, since there are also changes involving negative edges. However, from the implementation perspective only phenotypical (or positive) edges are considered.

[5]In this equation, any function that rounds $k/2$ to its nearest integer is sufficient.

[6]Transmitting can be regarded as an additional constraint imposed by operator.

in Figure 5) from taking the same value, which effectively reduces the domain for another position.

The only possibility that the value taken by one position does not constrain the value taken by another is that the parents both take the same value for that position (e.g. P_1 in Figure 5), where taking (the only) one value automatically satisfies the constraint.

Thus, as long as the constrained positions are transmitted all-together to the child, PTX always satisfies both C_{pos} and C_t and the child solution is always a valid permutation. Following the above example, the child produced by PTX could be:

$$(3, 6, \underline{5, 4, 2}, \underline{8, 7}, 1).$$

It should also be mentioned that, in the extreme case when all the positions are constrained in the same constraint graph (e.g. $(\underline{1, 2, 3, 4})$ and $(\underline{4, 1, 2, 3})$), the child after PTX will have to take all the equivalence classes for either of its parents. The constrained graph in this case is shown in Figure 6.

In fact, PTX works in a very similar manner as the cycle crossover [6] in literature, which preserves absolute positions in parents. However, it should be mentioned that in literature [2] there are also different kinds of crossovers that does not strictly transmitting absolute positional information to perform well in some cases. In those cases, we argue that different "requirements" or characteristics, other than strict transmission, may be expected from the crossover operator for different problem instances. Additionally, there may also be some positive "side effects" by incorporating a mixture of different characteristics apart from pure transmission to enhance the exploration power in search.

6.2 Precedence-based RTR

To achieve transmitting in precedence-based crossover, both transmitting C_t and C_{prec} should be satisfied from the CSP viewpoint. It is easy to verify that this CSP is *solvable*, since (in the worst case) taking all the equivalence classes for either parent to produce child always gives one possible solution such that constraints (C_t and C_{prec}) are satisfied.

Furthermore, precedence relation is special in that its equivalence class is either 1 or 0. This means in the case when two parents are different for a certain equivalence relation $\psi_{prec(e_i,e_j)}$, the domain of $\psi_{prec(e_i,e_j)}$ for the child is always $\{0, 1\}$, which implicitly means $\psi_{prec(e_i,e_j)}$ can take any value for the child. In the case when two parents are the same for $\psi_{prec(e_i,e_j)}$, the corresponding equivalence class is fixed for the child to achieve transmitting.

For example, given permutations $(3, 1, 2, 4)$ and $(4, 2, 3, 1)$

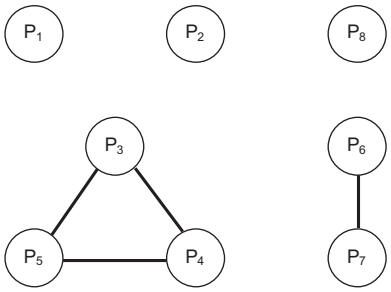

Figure 5: Illustration of the constraint graph of PTX.

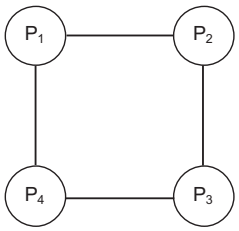

Figure 6: Constraint graph of PTX–an extreme case.

such that

$$\{\xi^1_{prec(1,2)}, \underline{\xi^0_{prec(1,3)}}, \xi^1_{prec(1,4)}, \xi^0_{prec(2,3)}, \xi^1_{prec(2,4)}, \xi^1_{prec(3,4)}\}$$

$$\{\xi^0_{prec(1,2)}, \underline{\xi^0_{prec(1,3)}}, \xi^0_{prec(1,4)}, \xi^1_{prec(2,3)}, \xi^0_{prec(2,4)}, \xi^0_{prec(3,4)}\}$$

the partial permutation as a child to achieve transmitting can be:

$$\{\xi^-_{prec(1,2)}, \xi^0_{prec(1,3)}, \xi^-_{prec(1,4)}, \xi^-_{prec(2,3)}, \xi^-_{prec(2,4)}, \xi^-_{prec(3,4)}\},$$

while the un-instantiated relations can be re-instantiated randomly to produce the child permutation, by solving a CSP such that simply C_{prec} should be satisfied.

In fact, strictly transmitting crossover is also possible for precedence-based description. Due to the fact that the re-instantiation process of precedence relations is equivalent to the *Topological Sorting Problems* [3], where a *partial order*[7] needs to be completed to a linear order (in a *directed acyclic graph* (DAG) based on precedence) with a complexity of O(edges+vertices), we argue that the re-instantiation of precedence relations in the considered CSP can be solved in a deterministic polynomial-time in terms of Topological Sorting Problems.

In literature, *Precedence Preservative Crossover* (PPX) [2] was found to be strictly transmitting. The underlining principle in PPX is that the precedence equivalence classes of parents are passed to the child in such an order (from left to right or more specifically from the node with no incoming edges in the precedence graph) that both C_t and C_{prec} are satisfied automatically.

Many readers may find that this is rather similar to the most popular algorithm used for topological sorting where the order can be completed by starting from the node(s) with no incoming edges. Switching between two parents simply aims to recombine the precedence equivalence classes of the two parents.

It is also easy to find that the set of all possible solutions produced by PPX is in fact a subset of the set of solutions produced by the above CSP approach. In other words, for each of the solution produced by PPX, there is always a corresponding re-instantiation of the partial child permutation.

6.3 Adjacency-based RTR

Regarding the adjacency-based description of permutation which has been proved to be non *g-separable* [7], literature [11, 9] pointed out that transmission can not be achieved without sacrificing assortment. From the CSP viewpoint, this implies that C_{adj} and C_t all together may make

[7]The partial order is defined by the equivalence relations where the parents are equivalent–the order that must be enforced for the child.

the corresponding CSP not *strictly* solvable (if the original parental permutations are not allowed to be repeated as a child permutation). Through investigation, we can also find that it is the case when two parents are the same for some equivalence relations that makes the CSP NP-complete in "strict" sense. For those adjacency relations that two parents are different, no restriction will be applied to the child solution for that relation (as both $\{1, 0\}$ are allowed).

Furthermore, for those edges which are absent in both parents ("negative edges" $\xi^0_{adj(i,j)}$), transmitting automatically forbids them from being included in the children. In this sense, the induced CSP is equivalent to the *Hamiltonian Cycle Problem* [4] in an *incomplete graph*, which is NP-complete[8]. Those edges which are common for both parents ("positive edges" $\xi^1_{adj(i,j)}$) effectively enforce that some edges must be included in the Hamiltonian cycles (solutions). This is actually a constrained version of the original Hamiltonian Cycle Problem induced by "negative edges", which is also NP-complete.

Thus, approximation through relaxation of C_t may be required to produce a valid *new* child permutation. In literature, *Enhanced Edge Recombination* devised by [8] has been noticed as an effective "edge-aware" recombination operator which has a high rate (98%) of adjacency transmission.

7. EXTENSION TO OTHER DOMAINS

In principle, our enhanced forma analysis framework is consistent across different problem domains (e.g., binary or real-parameter optimization), given appropriate description of problem in terms of equivalence relations, and any required feasibility constraints.

For example, the binary case automatically fits in our framework, as there is no feasibility constraint due to the orthogonality of binary description. For real-parameter optimization, with the formal descriptions introduced by Surry (*dedekind* cut and *isodedekind* cut) in [9] and the addressing of the feasibility constraints they induce [5], our proposed enhanced forma analysis framework is simply ready-to-adopt. However, we will not discuss this in detail here as it lies outside of the scope of this paper.

8. CONCLUSIONS

In this paper, we proposed the enhanced forma analysis framework where feasibility constraints implied by basis are handled from a CSP perspective. This facilitated our analysis of operators for those non-orthogonal descriptions (in this case, the permutation descriptions) where feasibility constraints exist, and allows an efficient default operationalization by using CSP solution algorithms.

In addition, the CSP viewpoint explores some further issues which were impossible to analyze with classical forma analysis, e.g. complexity issues. This is very beneficial as classical forma analysis is purely declarative which tells little about computational aspects of operator design. This framework bridges the gap between the formal formalism and our further demands for computational analysis.

[8]It should be pointed out that this Hamiltonian cycle problem is NP-complete only if the parental permutations are forbidden for the child, since the parents automatically gives 2 possible solutions. However, finding the third solution is still NP-complete.

Search operators for permutation problem have been studied following k-change and RTR operator templates to illustrate the basic approach of our enhance forma analysis, which gives a broader view of operator design for permutation with some further insights.

Applying our enhanced forma analysis to some other problem domain to obtain suitable operators would be a recommended future work. Further utilization of CSP techniques such as backjumping [10] to facilitate efficient operator implementation is also a good direction for research.

9. REFERENCES

[1] T. Bäck. *Evolutionary Computation 2: Advanced Algorithms and Operators*. Institute of Physics (IOP) Publishing, Bristol, UK, 2000.

[2] T. Bäck, D. Fogel, and Z. Michalewicz. *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of Physics (IOP) Publishing, Bristol, UK, 2000.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (Second Edition)*. MIT Press, 2001.

[4] M. DeLeon. A study of sufficient conditions for hamiltonian cycles. Technical report, Department of Mathematics and Computer Science, Seton Hall University, NJ, USA, 2000.

[5] T. Gong and A. L. Tuson. Formal descriptions of real parameter optimisation. In *Proceedings of IEEE Congress on Evolutionary Computation 2006, Vancouver*, pages 2119–2126. IEEE Press, 2006.

[6] I. M. Oliver, D. J. Smith, and J. R. C. Holland. A study of permutation crossover operators on the traveling salesman problem. In J. J. Grefenstette, editor, *Proceedings of the 2nd International Conference on Genetic Algorithms*, pages 224–230. Lawrence Erlbaum Associates, 1987.

[7] N. Radcliffe. The algebra of genetic algorithms. *Annals of Maths and Artificial Intelligence*, 10:339–384, 1994.

[8] T. Starkweather, S. McDaniel, K. Mathias, C. Whitley, and D. Whitley. A comparison of genetic sequencing operators. In R. Belew and L. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 69–76. Morgan Kaufmann, 1991.

[9] P. D. Surry. *A Prescriptive Formalism for Constructing Domain-specific Evolutionary Algorithm*. PhD thesis, University of Edinburgh, Edinburgh, Scotland, UK, 1998.

[10] E. P. K. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, London, UK, 1993.

[11] A. L. Tuson. *No optimization without representation: a knowledge based systems view of evolutionary/neighbourhood search optimization*. PhD thesis, University of Edinburgh, Edinburgh, 1999.