GSTF Journal on Computing (JoC) Vol.3 No.1, March 2013

The Complexity of Dance Choreography Procedures

Nigel Gwee, *Southern University and A&M College*

Abstract—**We derive the conceptual link between dance choreography and other significant problems in computer science, mathematics, and operations research. Dance choreography problems, formulated as decision problems, are shown to be NP-Complete. The corresponding optimization problems are shown to be NP-Hard and NP-Easy, and hence NP-Equivalent. We discuss the rationale behind these classifications, and evaluate various algorithmic solutions and their implementations**.

Keywords-software; NP-Complete; NP-Hard; NP-Easy; NP-Equivalent; ballroom; dance; software engineering

I. Introduction

For several decades now, very many significant computational problems have been continually identified for which efficient solutions have never yet been found. In 1971, Stephen A. Cook theorized a formal framework within which to classify such problems when he formulated the NP-Complete concept, and identified the first NP-Complete problem, the Boolean Satisfiability problem (SAT) [1]. Since then, the NP-Complete equivalence class of problems has expanded steadily. Thanks to contributions from various researchers, this class now contains thousands of important and interesting problems. These NP-Complete problems share a common property: should an efficient solution for any one NP-Complete problem be found, an efficient solution for all such problems is automatically assured. In this paper, we formulate a number of NP-Complete problems in the domain of Dance Choreography, and link them conceptually to other computational problems in diverse realms, such as computer science itself, mathematics, and operations research.

In the following sections, we describe the complexity classes NP-Complete, as well as NP-Hard, and NP-Easy. We then define some Dance Choreography problems and classify them accordingly, and argue for the case for putting them on par with other significant problems. Two algorithmic approaches to solving such problems are then discussed, with some illustrative experimental data. Finally, we describe a software implementation of some of the problems classified, showcasing some of its features.

II. Complexity Theory

Cook's seminal paper sparked a debate that is to this day unresolved: the question whether or not the set of problems that can be solved in polynomial time with respect to problem size is equal to the set of problems whose solution can be verified in polynomial time if a non-deterministic solution is derived, i.e., the question whether $P = NP$. We now discuss some complexity categories currently researched, define dance choreography problems that belong in these categories, and show how they not only conceptually link to, but can also be used to solve other similarly complex problems.

A. NP, NP-Complete, NP-Hard, NP-Easy, NP-Equivalent Complexity Classes

It is important to distinguish between the notion of sets of problems and the notion of algorithmic solutions to these problems. All the terms we are describing here refer to the former, although when we state that a certain problem is solvable using a certain time metric, we are referring to the performance of the best algorithm that has been discovered for that problem. These problem categories are based on the difference between efficient and non-efficient algorithm solutions available for the respective problem. All algorithms that are capable of solving a problem within a polynomial time factor with respect to input size of the problem are regarded as efficient; all algorithms not so capable, and take exponential time in the worst case, are not regarded as efficient. Problems for which it can be shown that no efficient algorithm is available for their solution are known as intractable problems. The NP- complexity classes we shall now discuss belong in that gray area where so far no known efficient algorithms exist for the problems in these classes, but these problems are yet to be proven intractable. To this day, no one knows if such a proof is even possible.

The concept of non-determinism is used frequently in computational science. It is a convenient conceptual device for considering the solution potential of an algorithm by suspending certain requirements, such as a concrete solution process. In all the classes in which we are concerned, the N-prefix refers to this non-deterministic aspect of the solution process. Thus, one way to characterize NP problems is that solution verification in polynomial time is possible, given a non-deterministically derived solution.

Furthermore, some NP problems are also NP-Complete. They are called NP-Complete because they possess the additional property mentioned in the introductory section of this paper, viz., that an efficient (i.e., polynomial-time) algorithmic solution for any one of the NP-Complete problems ensures an efficient solution for all NP-Complete, as well as NP, problems. Informally, every NP-Complete problem is at least as "difficult" to solve as all other NP problems. The solution to NP-Complete problems plays a key role in the question of *P* and *NP* equivalence.

NP-Complete problems encompass decision problems. Such problems are those that require a binary response (yes/no). We can further define the NP-Hard, NP-Easy, and

DOI: 10.5176/2251-3043_3.1.248

NP-Equivalent categories. These latter categories encompass more general problems that do not necessarily require a binary response. An NP-Hard problem is often an optimization problem related to an NP-Complete decision problem (see the next section for some examples). Informally, an NP-Hard problem is at least as "difficult" again as an NP-Complete problem. Quite often, an NP-Hard problem can be shown to be NP-Easy, meaning it is no more "difficult" to solve than any NP problem. This is the case for many optimization problems, such as the ones defined below. A problem that is shown to be both NP-Hard and NP-Easy is, by definition, NP-Equivalent. This implies that the decision problem and the related optimization problem that have been categorized as NP-Complete, and NP-Hard/Easy, respectively, are "equivalent" from the efficiency aspect. Many advanced texts are available for further discussion of this fascinating topic [3, 9].

B. Complexity of Dance Choreography Problems

We introduce several dance choreography problems and classify them into the above complexity classes. The aim is to link these problems, from a solution efficiency standpoint, to other problems in diverse domains of computing, such as computer science, mathematics, and operations research.

First we define a relatively specialized dance choreography decision problem (DC), and prove it is NP-Complete by restricting it to a known NP-Complete problem. Then we define a more general version of the dance choreography decision problem (DCk). This latter problem is in turn shown to be NP-Complete by restricting it to the previously proven NP-Complete problem DC. We then define optimization versions of these problems, and show that these are NP-Hard. We also show that one of these NP-Hard problems is also NP-Easy, and hence, NP-Equivalent.

DANCE CHOREOGRAPHY (DC)

INSTANCE:
A finite set *Figures*;
a set *Follow* of tuples $\langle f_i, f_j \rangle$, $f_i, f_j \in Figures$;
$n \in \mathbb{N}$;
$f_{start}, f_{end} \in Figures$;
a set $Compulsory \subseteq Figures$.

QUESTION:
Is there a sequence $Amalgamation = \langle f_1, f_2, \ldots, f_n \rangle$,
$f_i \in Figures$, $1 \le i \le n$, and $\langle f_j, f_{j+1} \rangle \in Follow$, $1 \le j < n$
such that
$f_1 = f_{start}$ and $f_n = f_{end}$
and
$\{f_i : 1 \le i \le n\} \supseteq Compulsory$?

This problem can be proven to be NP-Complete by showing (a) that it is in NP, and (b) that another, previously proven, NP-Complete problem can be transformed into it in polynomial time.

The first part of the proof, showing that it is in NP, i.e., that a non-deterministically obtained solution (if one is available) is verifiable in polynomial-time, is trivial. This process simply involves checking that the figures used in the amalgamation are valid with respect to *Follow*, that they are all contained in *Compulsory*, and that the first and last figures are as specified in the problem instance. Since similar processes are involved in the proofs of other Dance Choreography decision problems presented below, they are omitted in these other proofs.

In the second part of the proof, we select a known NP-Complete problem, called the HAMILTONIAN CIRCUIT (HC) [9], and show that it is many-one reducible to DC (HC $\propto$ DC). The proof technique used here is that of restriction [5].

HAMILTONIAN CIRCUIT (HC)

INSTANCE:
A graph *G*.

QUESTION:
Does *G* have a Hamiltonian Circuit (simple cycle containing all the vertices)?

THEOREM 1: DC is NP-Complete.

PROOF: Restrict to HC: Make $n = |Figures| + 1$, $Compulsory = Figures$, $f_{start} = f_{end}$. ♦

In order to define conveniently a related optimization problem, we modify DC slightly by introducing the notion of distinctness of figures used in an amalgamation, resulting in the problem we call Dance Choreography *k* (DCk). The modified problem is defined as follows, and shown also to be NP-Complete (DC $\propto$ DCk):

DANCE CHOREOGRAPHY *k* (DCk)

INSTANCE:
A finite set *Figures*;
a set *Follow* of tuples $\langle f_i, f_j \rangle$, $f_i, f_j \in Figures$;
$n \in \mathbb{N}$;
$f_{start}, f_{end} \in Figures$;
a set $Compulsory \subseteq Figures$;
$k \in \mathbb{N}_0$.

QUESTION:
Is there a sequence $Amalgamation = \langle f_1, f_2, \ldots, f_n \rangle$,
$f_i \in Figures$, $1 \le i \le n$, and $\langle f_j, f_{j+1} \rangle \in Follow$,
$1 \le j < n$
such that
$f_1 = f_{start}$ and $f_n = f_{end}$
and
$\{f_i : 1 \le i \le n\} \supseteq Compulsory$
and
$k \le |\{\text{distinct elements } f_1, .. f_n\}|$?

THEOREM 2: DCk is NP-Complete.

PROOF: Restrict to DC: Make $k = 0$. ♦

We are now able to define the optimization version of DCk, which we call Dance Choreography Optimization (DCO), a modification of the decision problem. This problem is shown to be NP-Hard.

DANCE CHOREOGRAPHY OPTIMIZATION (DCO)

INSTANCE:
A finite set *Figures*;
a set *Follow* of tuples $\langle f_i, f_j \rangle$, $f_i, f_j \in$ *Figures*;
$n \in \mathbb{N}$;
$f_{start}, f_{end} \in$ *Figures*;
a set *Compulsory* $\subseteq$ *Figures*.

QUESTION:
What is the optimal *Amalgamation* = $\langle f_1, f_2, \ldots, f_n \rangle$,
$f_i \in$ *Figures*, $1 \le i \le n$, and $\langle f_j, f_{j+1} \rangle \in$ *Follow*,
$1 \le j < n$,
in the sense that $|S|$ is maximal,
where S = {distinct elements $f_1, .. f_n$};
that satisfies
$f_1 = f_{start}$ and $f_n = f_{end}$
and
$\{f_i : 1 \le i \le n\} \supseteq$ *Compulsory*?

In this definition, the optimality measure is that of set cardinality: the more varied the figures used in the amalgamation, the better. Informally, the optimization problem is one of maximizing the variety of figures used. We can show that this problem is NP-Hard by describing a Turing reduction from an NP-Complete problem, such as DCk, i.e., DCk $\propto_T$ DCO.

THEOREM 3: DCO is NP-Hard.

PROOF: DCk is Turing-reducible to DCO in the following manner: Let *max* be the optimal solution for an instance of DCO, where *max* = $|\{f_1, .. f_n\}|$. Then, the answer for an instance of DCk is "yes" if $k \le max$, "no" otherwise. ♦

DCO is specialized in that the only optimality measure is cardinality of the set of figures used in the amalgamation. The problem can be generalized through the optimality measure. In other words, other optimality measures could be included and brought under a general optimality measure, defined by a fitness function F: $\{\langle f_1..f_n \rangle\} \rightarrow \mathbb{N}_0$. We call this version Dance Choreography Optimization 2 (DCO2), which is also NP-Hard because DCO $\propto_T$ DCO2. In the following sections, we discuss algorithmic implementations of DCO2. These implementations offer various specific optimality measures: "No preference" (meaning no optimality measure is in place, so all valid amalgamations are considered equally good), "Variety" (essentially the optimality measure used by DCO), "Simplicity," and "Repetition avoidance" (see fig. 6 below).

DANCE CHOREOGRAPHY OPTIMIZATION 2 (DCO2)

INSTANCE:
A finite set *Figures*;
a set *Follow* of tuples $\langle f_i, f_j \rangle$, $f_i, f_j \in$ *Figures*;
$n \in \mathbb{N}$;
$f_{start}, f_{end} \in$ *Figures*;
a set *Compulsory* $\subseteq$ *Figures;*
a function F: $\{\langle f_1, f_2..f_n \rangle\} \rightarrow \mathbb{N}_0$.

QUESTION:
What is the *Amalgamation* = $\langle f_1, f_2, \ldots, f_n \rangle$,
$f_i \in$ *Figures*, $1 \le i \le n$, and $\langle f_j, f_{j+1} \rangle \in$ *Follow*,
$1 \le j < n$,
that satisfies
$f_1 = f_{start}$ and $f_n = f_{end}$
and
$\{f_i : 1 \le i \le n\} \supseteq$ *Compulsory*
for which F ($\langle f_1, f_2, \ldots, f_n \rangle$) is maximal?

THEOREM 4: DCO2 is NP-Hard.

PROOF: An algorithmic solution with specialization of the evaluation function F of DCO2 to compute set cardinality provides the solution to DCO. ♦

We now show that the optimization problem DCO is NP-Easy, and hence, NP-Equivalent. To facilitate our task, we first define yet another decision problem that is a modification of DCk, Dance Choreography Extension (DCE):

DANCE CHOREOGRAPHY EXTENSION (DCE)

INSTANCE:
A finite set *Figures*;
a set *Follow* of tuples $\langle f_i, f_j \rangle$, $f_i, f_j \in$ *Figures*;
$n \in \mathbb{N}$;
$f_{start}, f_{end} \in$ *Figures*;
a set *Compulsory* $\subseteq$ *Figures*;
$k \in \mathbb{N}_0$;
a partial amalgamation $\langle f_1, f_2, \ldots, f_m \rangle$, $m \le n$,
$f_i \in$ *Figures*, $1 \le i \le m$, $\langle f_j, f_{j+1} \rangle \in$ *Follow*,
$1 \le j < m$.

QUESTION:
Can the partial amalgamation be extended to length n, i.e.,
$\langle f_1, f_2, \ldots, f_m, \ldots, f_n \rangle$,
$f_i \in$ *Figures*, $1 \le i \le n$, and $\langle f_j, f_{j+1} \rangle \in$ *Follow*,
$1 \le j < n$
such that
$f_1 = f_{start}$ and $f_n = f_{end}$,
$\{f_i : 1 \le i \le n\} \supseteq$ *Compulsory*, and
$k \le |\{$distinct elements $f_1, .. f_n\}|$?

The optimization problem DCO can now be shown to be NP-Easy by showing how it is polynomial-time Turing reducible to DCE.

THEOREM 5: DCO is NP-Easy.

PROOF: We show that DCO is polynomial-time Turing reducible to DCE (DCO $\propto_T$ DCE) in two stages, both of which

can be accomplished in polynomial time. Let *DCEAlg* be a hypothetical polynomial-time algorithm for DCE. (a) A binary search (a polynomial-time process) for the best score (k), bounded by cardinality of the set of figures used (n), can be performed using *DCEAlg*. (b) To determine the actual amalgamation, we call *DCEAlg* no more than $(n–1)(n–2)$ times if a solution exists. This is also is a polynomial-time process. Hence the reduction from DCO to DCE is a polynomial-time Turing reduction. ♦

Similar arguments can be advanced for the classification of the other optimization problems defined here. In our present case, since DCO is both NP-Hard (Theorem 4) and NP-Easy (Theorem 5), it is, by definition, NP-Equivalent. We have seen that finding an efficient solution for DCO automatically ensures an efficient solution for DCk. Now we find that the reverse is also the case, i.e., since DCE is in NP and DCk is NP-Complete, we can claim DCE ∝ DCk. Thus, we can use an efficient solution for DCk (if one were ever to be discovered) to solve DCO efficiently as well. In fact, the reductions we have shown during the various proofs form many cycles in a transformation graph, such as the one shown in fig. 1.

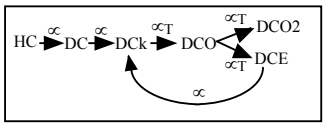

Figure 1. Transformation sequence of dance NP problems.

Among other things, this shows that we do not lose generality in dealing with only the decision versions of an optimization problem. For practical reasons, however, it is more useful to use the optimization versions of the problems, especially if we intend to implement algorithmic solutions to these problems.

In general, all the dance choreography problems here defined (including DCE, which, incidentally, can also be shown to be NP-Complete) are conceptually connected to all other NP-Complete problems, those known and those yet to be discovered. It is entirely possible, for example, that in the future these dance choreography problems may be used to prove that a newly identified problem is NP-Complete or NP-Hard.

C. A Solution Procedure For NP Problems

Besides accomplishing their theoretical purpose of establishing the complexity classes of computational problems, these proofs also furnish transformational descriptions that are useful in actually solving these problems. For example, any instance of the HAMILTONIAN CIRCUIT problem (HC) can be solved by transforming it into an instance of the Dance Choreography problem (DC), along the lines described in the NP-Completeness proof of the latter. Incidentally, another well-known computational problem, the so-called TRAVELING SALESMAN problem (TS), also used HC to prove that it was NP-Complete. Thus, HC can also be solved using a solution to TS. The key unanswered question is whether the solution of these problems can be accomplished efficiently.

Example 1. An instance of DC is shown in fig. 2. The graph on the left represents a simplified hypothetical syllabus of four figures (R[ight] F[oot] Closed Change, L[eft] F[oot] Closed Change, Reverse [Turn], and Natural [Turn]) with their permissible interactions. By imposing the constraints $n = |Figures| + 1$ and $Compulsory = Figures$, $f_{start} = f_{end}$, this instance becomes also one of HC. Thus, the same algorithmic solution to DC problems can be used to solve HC problems. The graph on the right represents a solution to the present instance (the dark arcs indicate a possible amalgamation).

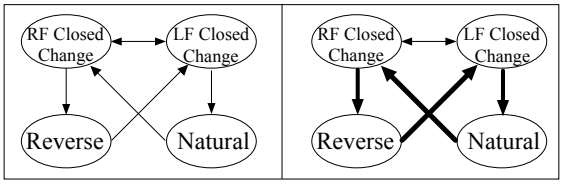

Figure 2. Solution of a DC and HC problem instance.

In a paper published soon after Cook's, Karp introduced twenty-one new NP-Complete problems. Among them was VERTEX COVER (VC), which Karp proved was NP-Complete by showing how an instance of it could be transformed to one of HC, thus showing that a solution for the latter could be used to solve the former [9]. Both VC and HC are graph theory problems, but these can be used to solve propositional logic problems as well, such as the 3-SATISFIABILITY problem (3SAT). This problem in turn was proven NP-Complete by transformation of the SATISFIABILITY problem (SAT). As noted above, SAT was in fact the first ever proven NP-Complete problem: Cook proved it to be so by showing that an instance of any NP problem was transformable to an instance of it [1].

Therefore, if direct transformations are not readily available, an instance of any NP problem can be solved by first transforming it into an instance of SAT using the procedure outlined by Cook, which in turn can be transformed eventually into an instance of any NP-Complete problem for which an algorithmic solution is available. The practical application of the conceptual link between dance and other significant computational problems should now be apparent (fig. 3).

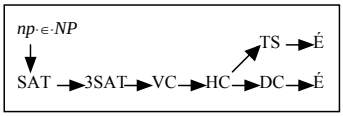

Figure 3. General transformation sequence of NP problems.

D. Exhaustive and Heuristic Algorithms for Dance Choreography

Procedures designed to solve problems such as the dance choreography problem are potentially very time consuming, unless some concessions are made. If we insist on perfect solutions, the procedure involved could become very impractical once the problem reaches a certain size. In the case of the dance choreography problem, once the number of figures

required, called the amalgamation length, exceeds about ten, things very quickly go haywire.

The Exhaustive algorithm is one that tries to investigate every eventuality, by using a backtracking process in the search. In our implementation, we pruned the search tree as soon as a figure was reached that could not be followed. Even so, very many possibilities arise exponentially as the amalgamation length increases. The NP-Complete theory outlined above warns us that it is futile, at the present, to investigate more efficient procedures while requiring perfect results.

Example 2. As an indication of the exponential increase in computing time, we show what can happen as we attempt to create amalgamations of increasing length. We ran the Exhaustive algorithm to search for amalgamations of increasing length from one to twelve figures. The figures were chosen from our proprietary Quickstep syllabus. To ensure a uniform start, the opening figure in each trial was the figure called "Quarter Turn to the Right." No ending figure was specified for any of the amalgamations. The optimality measure was as defined in DCO, which is the degree of variety of figures: within the specified length of the amalgamation, preference was given to those amalgamations comprising as many different figures as possible; the optimality values are normalized to the interval [0..10]. The "optimal" column refers to the number of optimal amalgamations found under this measure for that particular amalgamation length; the "suboptimal" column refers to all other otherwise valid amalgamations found. The experiment was performed on a 2.3 GHz Intel Core i5 processor. Processing time was measured in clock ticks (60 clock ticks equal one second). As Table I shows, processing time increased exponentially with increasing amalgamation length. ♦

TABLE I. PERFORMANCE OF THE EXHAUSTIVE ALGORITHM (QUICKSTEP)

length	time	optimal	suboptimal	total
1	<1	1	0	1
2	<1	5	0	5
3	<1	22	1	23
4	1	64	12	76
5	4	233	94	327
6	21	792	501	1293
7	95	2612	2732	5344
8	424	8198	13576	21774
9	1880	24431	64879	89310
10	8323	68431	296896	365327
11	36586	178856	1317186	1496042
12	161291	436003	5687687	6123690

Fine-tuning tactics in an Exhaustive search, such as pruning of the search tree mentioned above, will still result in exponential running time in the worst case. Fortunately however, problems such as these do benefit from less idealistic procedures. These procedures make use of various problem-specific heuristics, and are often the most practical solutions for non-trivial problem sizes.

A well-known heuristic approach is the so-called Greedy algorithm. There is no moral attachment to the name of this algorithm: it merely suggests that a search guided by the concept of the "best" choice is attempted, with no intent to backtrack, as happens in the Exhaustive algorithm. In our implementation of the Greedy algorithm, the amalgamation is built figure-by-figure, and the next figure chosen in the sequence is the one that appears the most promising with respect to the fitness function, as defined in DCO. Once a figure is chosen, there is no turning back, i.e., there is no backtracking. As a result, the Greedy algorithm runs much faster than does the Exhaustive algorithm, at the risk of producing non-optimal results (Example 3).

Example 3. The results of the application of the Greedy algorithm to the same parameters as in Example 2 are shown in Table II. The same fitness function, as defined in DCO, was used. The "time" column shows the clock ticks; the "fitness" column shows the optimality measure, with values in the range [0..10]. Note that the speed trade-off is in the quality of amalgamations: not all the amalgamations were optimal compared to the results obtained by the Exhaustive algorithm. Nevertheless, an overall acceptable quality was obtained in this and several other trials not shown here. The running time was drastically shortened, however. ♦

TABLE II. PERFORMANCE OF THE GREEDY ALGORITHM (QUICKSTEP)

length	time	fitness
1	<1	10
2	<1	10
3	<1	10
4	<1	10
5	1	10
6	1	10
7	1	10
8	1	10
9	1	9.4
10	1	9.0
11	1	8.6
12	1	8.3

Besides the Greedy algorithm, other heuristics-based algorithms could be used. One promising method is the Genetic Algorithm. We have used this approach in the NP-Hard music composition problem of Species Counterpoint [6]. Smart modifications to the Exhaustive algorithm are also possible, for example, the branch-and-bound technique to implement best-first searching.

III. MULTIMEDIA DANCE SOFTWARE

Terpsichore© is a multimedia-based interactive computer software for dance training and education, copyrighted on July 1, 2011 (Registration Number TXu 1-762-479) [4]. It generates ballroom dance amalgamations based on figures formalized by various International Standard dance associations: DVIDA (Dance Vision International Dance Association), IDTA (International Dance Teachers' Association), and ISTD (Imperial Society of Teachers of Dancing) [2, 7, 8]. Based on structural and pedagogical considerations, we have compiled our own syllabus by grouping these dance figures into three categories:

"Foundational," "Intermediate," and "Advanced." We now describe some features of Terpsichore© related to implementations of the problems defined above.

A. *Simplified Implementation of the Optimization Problem Using the Exhaustive and Greedy Algorithms*

Two simplified implementations in our software using the Exhaustive algorithm omit the *Compulsory* set specification. One enables specification of both f_{start} and $f_{end,}$ while the other omits the specification of f_{end}. In fig. 4, they are labeled "Thorough Search (with specified starting and ending figures)" and "Thorough Search (with specified starting figure)," respectively. The latter implementation was used to generate the output shown in Table I.

Similarly, two simplified implementations using the Greedy algorithm are labeled "Quick Search (with specified starting and ending figures)" and "Quick Search (with specified starting figure)," respectively (fig. 4). The latter implementation was used to generate the output shown in Table II.

B. *Full Implementation of the Optimization problem Using the Exhaustive Algorithm*

The various components of the generalized optimization problem described above (DCO2) are realized as follows. The Exhaustive algorithm that implements this problem is called "Thorough Search (with specified required figures)" (fig. 4). The user selects a finite set *Figures*, specifies f_{start} and f_{end} from our proprietary syllabus (fig. 5), specifies the amalgamation length, n, determines the evaluation function F (fig. 6, showing the "Variety" criterion), and chooses a set of *Compulsory* figures (fig. 7).

Example 4. With the settings as shown in figs. 2, 3, 4, and *Compulsory* = ∅, 30 optimal and 33 suboptimal Quickstep amalgamations were found (fig. 8 shows one of these optimal amalgamations). Note that fixing the ending figure, f_{end}, results in far fewer amalgamations, compared to the number found in the corresponding setting shown in Table I. ♦

Example 5. With the settings as shown in figs. 2, 3, 4, and 5 (two figures chosen for the *Compulsory* set: "Progressive Chassé" and "Natural Spin Turn"), only 9 optimal and 0 suboptimal amalgamations were found (fig. 9). Not surprisingly, with the added restriction imposed by the compulsory figures, the overall number of optimal amalgamations is substantially less than that in Example 4. ♦

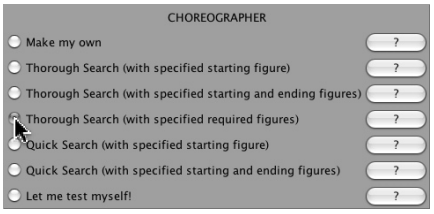

Figure 4. Implementations of dance choreography problems.

Figure 5. *Figures*, f_{start}, f_{end} selections.

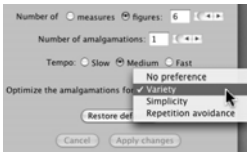

Figure 6. *n*, *F* selections.

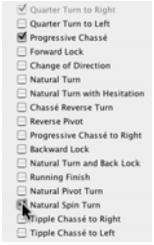

Figure 7. *Compulsory* figures selection.

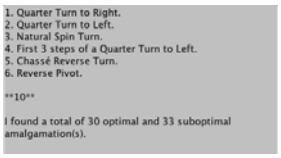

Figure 8. Output for *Compulsory* = ∅.

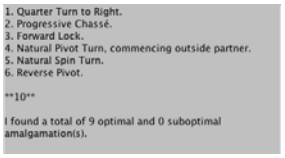

Figure 9. Output for |*Compulsory*| = 2.

IV. Conclusion

We have formulated several Dance Choreography problems and shown them to be variously NP-Complete, NP-Hard, and NP-Easy. The significance of these classifications lies in their relatedness to problems in computer science, mathematics, and operations research. We evaluated various algorithmic solutions to these problems, as implemented in our software, Terpsichore©.

Terpsichore© was designed to be extensible and portable. Therefore, we intend to pursue future software development along those lines. Theoretically, more such problems can be formulated, and we intend to make further relations apparent, not only between dance problems, but also to problems in related performing arts, such as music composition.

Acknowledgment

The author wishes to thank the Editors of the Journal of Computing for their kind invitation to publish this paper, and their attention to its preparation. Thanks are also extended to the reviewers of this paper for their invaluable suggestions. Mr. William ("Bill") Davies, Dancesport Adjudicator and Examiner, and former North American and United States Ballroom Dance Champion gave artistic advice during the development of Terpsichore©, for which the author is very grateful. The author is also very grateful to Attorney Frances Ball for legal advice in the copyrighting of the software.

REFERENCES

[1] S. A. Cook, "The complexity of theorem-proving procedures," in Proc. 3rd Annual ACM Symposium on Theory of Computing, Shaker Heights, Ohio: Association for Computing Machinery, 1971, pp. 151–158.

[2] Dance Vision International Dance Association, Bronze (Silver/Gold) Level International Style Standard Manual, Las Vegas, Nevada: Dance Vision, 2008.

[3] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, 20th printing. New York: Freeman, 1999.

[4] N. Gwee, "Software copyright: A programmer's perspective," Technology and Innovation, vol. 14, no. 3/4, 2012, pp. 237–247.

[5] N. Gwee, "Terpsichore©: From NP-Complete problem to multimedia software," Proc. 3rd Annual International Software Engineering & Applications Conference, Singapore, pp. 189–193, 2012.

[6] N. Gwee, "Effective heuristics for algorithmic music composition," Proc. International Conference on Artificial intelligence, Las Vegas, Nevada, vol. III, pp. 1027–1033, 2002.

[7] G. Howard, Technique of Ballroom Dancing, Brighton, England: KenadS Design & Print, 2007.

[8] Imperial Society of Teachers of Dancing, The Ballroom Technique, London, England: Lithoflow Ltd, 1994.

[9] R. M. Karp, "Reducibility among combinatorial problems," In Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, Eds. New York: Plenum, pp. 85–103, 1972.

Nigel Gwee is an associate professor of computer science at Southern University, Baton Rouge, Louisiana. He received his Ph.D. in musicology and his Ph.D. in computer science from Louisiana State University, Baton Rouge. He has written papers on software engineering and complexity analysis, and computer programs on music and dance. Recently, he won the Best Research Paper Award at the Third International Conference on Software Engineering & Applications, 2012, Singapore.