Parameterized Complexity Analysis in Robot Motion Planning*

Marco Cesati
Dipartimento di Scienze dell'Informazione
Università degli Studi di Roma "La Sapienza"
via Salaria 113, 00198 Roma, Italy
marco@dsi.uniroma1.it

H. Todd Wareham
Computer Science Department
University of Victoria
Victoria, British Columbia
Canada V8W 3P6
harold@csr.uvic.ca

1. Introduction

Given a robot in an environment composed of some set of obstacles and initial and final positions of the robot within this environment, the *motion planning problem* involves finding a sequence of motions that move the robot from the initial to the final position without intersecting any of the obstacles. Though polynomial-time algorithms are known for this problem for very limited kinds of robots, e.g., line segments, disks, rectangles (see [21] and references), the best known algorithm for arbitrarily complex robots requires $O(n^k(\log n \cdot d^{O(k)} + 2dk^{O(k^2)}))$ time [3], where k is the degrees of freedom of movement, n is the number of polynomials required to describe the surfaces of the robot and its environment, and d is the maximum degree of these polynomials [21]. The terms exponential in d and k in these running times are not daunting because values of d and k in practice are typically small, e.g., $d = 4$ for a polygonal robot in a planar polygonal environment and $k \leq 7$ for industrial robot arms. However, it is thought unlikely that algorithms such as that in [3] can eliminate the n^k term in their running time because such algorithms must compute all points in a special k-dimensional space called *FP space*, and the number of points in this space is $O(n^k)$ in the worst case [21, Theorem 3.1].

A series of $PSPACE$- and NP-hardness results (see papers in [22] and references) suggest that no general algorithm for robot motion planning can be polynomial in *all* of its input parameters, i.e., at least one parameter x must be exponential relative to a constant, e.g., 2^x, or another parameter of the problem, e.g., y^x. However, they have not answered the more relevant question posed by the FP space-based algorithms above – namely, whether there is a general algorithm that is polynomial in all input parameters *except* k, in which k may yet be exponential relative to a constant or itself.

In this paper, using the theory of parameterized computational complexity developed by Downey and Fellows [5], we establish that the answer to this question is probably "no". In Section 2, we give an overview of this theory. In Section 3, we derive our main result. Finally, in Section 4, we briefly discuss the implications for robotics of both these results and the parameterized complexity framework.

*Appeared in *Proceedings of the 25th IEEE International Conference on Systems, Man, and Cybernetics (Vancouver, BC: October 23–25, 1995)*: Volume 1. IEEE Press; Los Alamitos, CA. 880--885.

2. Parameterized Complexity Theory

In light of the intractability of many computational problems [12], many forms of approximation have been developed to solve such problems in practice, e.g., randomized algorithms, simulated annealing, bounded-cost approximation schemes. One such approach seeks not to solve all instances approximately but rather to solve all small instances exactly, where "small" instances are those having a small value for a chosen input parameter. Computational problems exhibit two algorithmic behaviors when an input parameter k is bounded in value:

1. Some problems are *fixed-parameter (f.p.) tractable*, i.e., they have $f(k)n^\alpha$ time algorithms where f is an arbitrary function, n is the largest value among the remaining input parameters, and α is a constant independent of k.

2. Some problems seem to be solvable only by "brute force" $O(n^k)$ time algorithms.

Both types of algorithms are polynomial-time for fixed values of k. However, f.p. tractable algorithms are preferable if typical instances of a problem have small values of k, e.g., when $k = 10$ and $n = 1000$, $2^k n^3 = n^4 << n^{10} = n^k$.

Though f.p. tractable algorithms have been derived by a variety of techniques for a number of problems [1, 7, 10], many problems have resisted all such attacks. The question, then, is how to determine whether or not a problem has an f.p. tractable algorithm. There does not seem to be any correlation between the general, e.g., $NP/PSPACE$-hard, complexity of a problem and whether or not it will be f.p.

tractable. This is nicely illustrated by the following two problems from VLSI design.[1] Each of them takes as input a graph $G = (V, E)$ and a positive integer k, and in each case the bounded parameter is k.

CUTWIDTH: Is there a linear ordering of V, i.e., a 1:1 function $f : V \rightarrow \{1, 2, \ldots, |V|\}$, such that all i, $1 < i < |V|$, $|\{\{u, v\} \in E : f(u) \leq i < f(v)\}| \leq k$?

BANDWIDTH: Is there a linear ordering of V such that for all $\{u, v\} \in E$, $|f(u) - f(v)| \leq k$?

Though both problems are NP-complete [12, Problems GT44 and GT40], the first is solvable in linear time for fixed values of k while the best known algorithms for the second require $O(|V|^k)$ time [11].

One approach to showing that efficient algorithms do not exist for a problem is that used in computational complexity theory [12, 18]. Essentially, one defines a class F of efficiently-solvable problems, a class C such that $F \subset C$, and a means for isolating the hardest problems in C.[2] If a given problem X is at least as hard as the hardest problem in C, then X does not have an efficient algorithm modulo the strength of the assumption that $F \subset C$. The roles of F and C are played by classes P and NP in traditional computational complexity theory [12].

This class-based approach to proving intractability is at the heart of parameterized complexity theory [5]. Within this theory, class F above corresponds to the class FPT of parameterized problems that are f.p. tractable, and class C is one of the members of the W-hierarchy, a set of classes $\{W[1], W[2], \ldots, W[SAT], W[P], \ldots\}$ defined by successively more powerful solution-checking circuits. The interested reader is referred to [5, 8] for details. These classes form the following hierarchy.

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[SAT] \subseteq W[P]$$

It is conjectured that all inclusions in this hierarchy are proper [6]. Hence, no W[x]-complete problem is f.p. tractable unless all problems in W[x] are f.p. tractable. Problems from areas as diverse as VLSI design, molecular biology, formal logic, and computational linguistics have been classified within the W-hierarchy [2, 9, 11]. Known results include GATE MATRIX LAYOUT (in FPT), VAPNIK-CHERVONENKIS DIMENSION (W[1]-complete), WEIGHTED BINARY INTEGER PROGRAMMING (W[2]-complete), and MINIMUM AXIOM SET (W[P]-complete). Over one hundred results are listed on-line [13] (see the Parameterized Complexity Home Page at `http://www-csc.uvic.ca/home/mhallett/research.html`).

Essentially, a W-hardness result for a k-parameterized problem suggests that some (possibly sublinear) form of k cannot be removed as an exponent of other input parameters in the running time of any general algorithm for that problem without forcing some other input parameter to go exponential. Note that this consequence holds only when we are considering all possible values of k; W-hardness does not preclude an algorithm whose running time is $f(k)n^\alpha$ for $k \leq c$, where c is some constant.

3. The Parameterized Complexity of the Generalized Mover's Problem

A formal definition of our problem is as follows [19]. Note that the polyhedra in this definition are not necessarily convex; they are only required to be *rational*, i.e., specifiable as a finite union of convex polyhedra, each of which is specified by a finite set of linear inequalities with rational coefficients.

d-DIMENSIONAL EUCLIDEAN GENERALIZED MOVER'S PROBLEM (dD-GMP, $d \in \{2, 3\}$)
Instance: A set O of obstacle polyhedra, a set P of polyhedra which are freely linked together at a set of linkage vertices V such that P has k degrees of freedom of movement, and initial and final positions p_I and p_F of P in d-dimensional Euclidean space.
Question: Is there a legal movement of P from p_I to p_F, i.e., is there a continuous sequence of translation and rotations of the polyhedra in P such that at each point in time, no polyhedron in P intersects any polyhedron in O and the polyhedra in P intersect themselves only at the linkage vertices in V?

Let k-dD-GMP denote the parameterized version of this problem in which k is the bounded parameter. Reif [19] showed that 3 D-GMP is $PSPACE$-hard by a reduction from a variant of the bounded-space symmetric Turing Machine (TM) computation problem defined in [17]. As Reif's reduction is also a parameterized reduction (see Theorem 4 below), in order to show that k-3D-GMP is W[SAT]-hard, it will suffice to show that the appropriately parameterized version of symmetric TM computation is W[SAT]-hard.

Let $M = (Q, \Sigma, \delta, \Box, q_0, q_F)$ be a standard single one-way-infinite tape deterministic Turing machine (DTM), where Q is the set of states, Σ is the tape alphabet, $\Box$ is the tape blank symbol, $\delta : Q \times \Sigma \rightarrow Q \times \Sigma \times \{-1, 0, +1\}$ is the transition function, and

[1] Note that the outputs of these problems are not actual solutions but rather the answer to a yes/no question about these solutions. All problems examined in this paper will be phrased as the latter *decision problems* rather than the former *search problems*. This is done because (1) decision problems are easier to analyze, and (2) the complexity of an appropriately-defined decision problem is a lower bound on the complexity of the associated search problem, i.e., finding a solution can be no easier than answering a question about that solution.

[2] This mechanism is typically a reducibility between pairs of problems. Given a pair of problems X and Y, a reducibility α efficiently transforms instances of X into Y, i.e., $X \; \alpha \; Y$ (read "X reduces to Y"), such that an efficient algorithm for Y can be used to efficiently solve instances of X. Note that $X \; \alpha \; Y$ also implies that if X is not efficiently solvable, neither is Y; hence, a reducibility orders problems by computational difficulty. A reducibility can be used to isolate the hardest problems in a class C via the notions of hardness and completeness. A problem X is *C-hard* if for all problems $Y \in C$, $Y \alpha X$; if X is also in C, then X is *C-complete*.

$q_0, q_F \in Q$ are the initial and final states, respectively (see a standard text such as [15] for details). A *configuration* $C = (q, h, t)$ is a description of the current computation status of M, where $q \in Q$ is the current state, $h \in \mathcal{Z}^+$ is the current position of the read/write head on the tape, and t is the current tape contents of all tape squares visited up to that point in the computation. A computation of M on x is a sequence of configurations $C_0, C_1, \ldots, C_l$ such that $C_0 = (q_0, 1, x)$ and for each i, $1 \leq i \leq l$, there is a transition in δ that transforms C_{i-1} into C_i. Consider the following problem involving such DTM.

Compact deterministic Turing machine computation (CDTMC)
Instance: A DTM $M = (Q, \Sigma, \delta, \Box, q_0, q_F)$, a string $x \in \Sigma^*$, and an integer k.
Parameter: k
Question: Is there an accepting computation of M on input x that visits at most k squares?

Let I_0-CDTMC denote the version of this problem which has the empty string as input.

Theorem 1 [4, Theorem 8] I_0-CDTMC is W[SAT]-hard.

Reif's Turing machines differ in three respects from the standard model used by Cesati:[3]

1. Each TM has an explicit space bound $s = s(|x|)$, and the tape consists of s squares bounded at the left and right by marker symbols ($).
2. Transitions are of the form $(q, L, R, D) \rightarrow (q', L', R', D')$, where $q, q' \in Q$, $L, R, L', R' \in \Sigma$, and $D, D' \in \{1, 0, -1\}$ such that $D = -D'$ and $R = R'$ if $D = 0$.
3. The machine can only enter the final state q_F if all tape squares are blanks and the head is on the leftmost blank on the tape.

The transitions are interpreted as follows. Let t_i (t'_i) be the symbol in position i on the tape before (after) the transition, and let h (h') be the position of the read/write head on the tape before (after) the transition. Note that $h' = h + D$.

- If $D = 1$ or $D = 0$ then $t_h = L$, $t_{h+1} = R$, $t'_h = L'$, and $t'_{h+1} = R'$.
- If $D = -1$ then $t_{h-1} = L$, $t_h = R$, $t'_{h-1} = L'$, and $t'_h = R'$.

This formulation of transitions allows symmetric TM computation to be defined easily – namely, TM M is *symmetric*, i.e., M is a STM, if the transition-set δ of M is such that $(q, L, R, D) \rightarrow (q', L', R', D') \in \delta$ if and only if $(q', L', R', D') \rightarrow (q, L, R, D) \in \delta$. Define the deterministic variant of these TM in the standard manner, i.e., TM M is *deterministic* if at most one transition can be applied to each configuration $C = (q, h, t)$ of M, where $q \in Q$, $h \in \{1, \ldots, s\}$, and $t \in \$(\Sigma \cup \{\Box\})^s\$$.

We now use the following two reductions to show that I_0-CDTMC for Reif-style symmetric TM is W[SAT]-hard. Let I_0-CDTMC(R) and I_0-CSTMC(R) be the versions of I_0-CDTMC on Reif-style deterministic and symmetric TM, respectively.

Theorem 2 I_0-CDTMC(R) is W[SAT]-hard.

Proof: The reduction is from an instance (M, x, k) of CDTMC to an instance (M', x) of CDTMC(R) such that $s(|x|) = k$. Given a DTM M in an instance of CDTMC, construct a Reif-style DTM M' as follows: Let M' have the same alphabet and space-bound s as M, set $Q' = Q \cup \{q_F\} \cup Q''$ (see (4) below), create a new final state ${q'}_F$, and construct the transition set δ' from δ as follows. Let $\Gamma = \Sigma \cup \{\Box\}$.

1. For each $(q, a) \rightarrow (q', b, 1) \in \delta$, add $\{(q, a, x, 1) \rightarrow (q', b, x, -1) \mid x \in \Gamma\}$ to δ'.
2. For each $(q, a) \rightarrow (q', b, 0) \in \delta$, add $\{(q, a, x, 0) \rightarrow (q', b, x, 0) \mid x \in \Gamma \cup \{\$\}\}$ to δ'.
3. For each $(q, a) \rightarrow (q', b, -1) \in \delta$, add $\{(q, x, a, -1) \rightarrow (q', x, b, 1) \mid x \in \Gamma\}$ to δ'.
4. Add a set of transitions to δ' which, on entry into the original final state q_F of M, use a special group of states Q'' to go to the rightmost tape-marker symbol, return to the leftmost tape-marker symbol blanking out all symbols as it goes, and enter state ${q'}_F$.

Note that the only transitions that need to deal with the leftmost and rightmost tape-marker symbols $ are those in groups (2) and (4). No other transitions involving the leftmost tape-marker symbol are required by M' as no accepting computations of M moved the head off the leftmost end of the tape; similarly, no transitions involving the rightmost tape-marker are required because no accepting computation of M used more than s tape squares.

We now show that the constructed TM M' is equivalent to M. Suppose M' is not deterministic, i.e., there is a configuration C' of M' such that at least two transitions of M' can be applied to C' (we can assume that none of these transitions is from group (4) above). Select any two of these transitions and call them δ'_1 and δ'_2. As any transition of M' in groups (1) – (3) above corresponds naturally to a transition in M, the transitions $\delta_1, \delta_2 \in \delta$ corresponding to δ'_1 and δ'_2 both apply to the configuration C of M corresponding to C'. This, however, implies that M is not deterministic, which is a contradiction. Hence M' is deterministic. Let $x \in \Sigma^*$ be an arbitrary string. The correspondence noted above between transitions in M and M' implies that if M' accepts x then M accepts x; moreover, given any sequence of configurations by which M accepts x, one

[3] Note that, unlike [19], the Reif-style TM in this paper allow transitions that do not move the tape head. However, this does not cause significant changes to any of the results from [19] that are used here.

can easily derive a corresponding sequence of configurations for M' by the construction given above. Hence, the languages accepted by M and M' are the same. To complete the proof, note that this construction can be performed in time polynomial in $|M|$, and that the resulting machine M' has the same space bound as M. □

The second reduction is essentially a simplified version of the machinery developed in [17, Section 3], as adapted for Reif-style TM.

Theorem 3 I_0-CSTMC(R) is W[SAT]-hard.

Proof: This reduction has two parts. First, given a Reif-style DTM M in the instance of CDTMC(R), construct a Reif-style DTM M' by adding a new final state that can be entered only from the old final state. Second, given M', construct a Reif-style STM M'' by adding the appropriate transitions to δ' to make it symmetric.

We now establish that the constructed Reif-style STM M'' is equivalent to M. It is obvious that M' accepts the same language and has the same space bound as M. As M' is deterministic and takes as input only the empty string, there is a unique sequence of configurations computed by M', which either leads to the final state or rejects, i.e., either halts when there is no applicable transition or runs endlessly; call this sequence S. It is obvious that M'' can replicate S by using the original transitions of M'. However, we need to be sure that M'' cannot use its new transitions to accept the empty string in the case that M' rejects it. We can show this by contradiction. Suppose that M'' accepts but M' does not; this implies that there is a sequence of configurations S' for M'' that leads from the initial to the final state. There must then be a configuration C in S where the computations of M' and M'' diverge such that M' and M'' apply different transitions; as M' is deterministic, M'' must apply to C a transition that is not available to M', i.e. a transition from the set $\Delta = \delta'' - \delta'$, where each member of Δ is just the inverse of some transition in δ'. None of the transitions following C in S' can be in δ', as such transitions can only force the computation back towards C (assume that this was not so, and that at some point C' reached by a transition $\delta_1 \in \Delta$, one could apply a transition $\delta_2 \in \delta'$ to C such that δ_2 did not "undo" δ_1; however, this would imply that the inverse $\delta_1' \in \delta'$ of δ_1 and δ_2 are both valid transitions from C' in M', which contradicts the determinism of M'); hence each transition in S' after C must be from Δ. However, by the construction of M', there can be no transition in Δ from any state in Q to the final state, which contradicts the assumption that S' leads M'' to accept. Hence, though the symmetric computation of M'' can reach configurations that are not accessible to its deterministic counterpart M', M'' cannot compute a result different from that computed by M'. Thus, M'' accepts the same language and has the same space bounds as M' and M. To complete the proof, note the concatenation of the transformations from M to M' and M' to M'' runs in time polynomial in $|M|$. □

We can now state our main result.

Theorem 4 k-3D-GMP is W[SAT]-hard.

Proof: Given an instance of CSTMC(R) consisting of a Reif-style STM M with space-bound s, the reduction given in [19] constructs a robot that has $s+2$ degrees of freedom in space logarithmic and hence (by [15, Theorem 12.10(b)]) time polynomial in $|M|$. □

We can obtain results relative to additional parameters via previous reductions in the robotics literature [14, 16, 19]. Let a maximally-linked group of polyhedra in P be a *component*, and each polyhedron in P be a *part*. Define the following parameters of 3D-GMP:

- The number of *components* of P (c);
- The maximum number of parts in any component in P (p^c) and the total number of parts in P (p^t);
- The maximum number of algebraic inequalities, i.e., planar surfaces, lines, curves, needed to define any part (s^p) or component (s^c) of P, or the total number of such inequalities needed to define P (s^t);
- The maximum number of linkage vertices on any part (v^p) or component (v^c) of P, or the total number of linkage vertices in P (v^t); and
- The maximum number of degrees of freedom of any part (k^p) or component (k^c) of P, or the total number of degrees of freedom of P ($k^t = k$).

The best known hardness results are given in Table 1. The classical and parameterized complexity results in this table are not incompatible; using the definitions in [5, 8], it is easy to show that if a problem X is C-hard for some classical complexity class C when a parameter k is fixed at a constant value, then the parameterized version k-X cannot be in (and hence must be harder than) W[P] unless $P = C$. Hence, as $P = PSPACE$ is thought to be unlikely, the parameterized problems in Table 1 that are $PSPACE$-hard are at least W[P]-hard and indeed probably much harder.

4. Discussion

In this paper, we have shown that the 3-DIMENSIONAL EUCLIDEAN GENERALIZED MOVER'S PROBLEM is W[SAT]-hard when the number of degrees of freedom k is a bounded parameter. This suggests that there is no general algorithm for robot motion planning whose running time can avoid having some function of k as an exponent of some input parameters of the problem without forcing some other input parameter to go exponential. Note, however,

Parameter		3D-GMP	2D-GMP
Components	c	$PSPACE$-hard [16, 19] (1)	$PSPACE$-hard [16] (1)
Parts	p^c	$PSPACE$-hard [14] (1)	$PSPACE$-hard [14] (1)
	p^t	W[SAT]-hard	???
Surfaces	s^p	$PSPACE$-hard [16] (1)	$PSPACE$-hard [16] (1)
	s^c	$PSPACE$-hard [14] (4)	$PSPACE$-hard [14] (4)
	s^t	W[SAT]-hard	???
Degrees of Freedom	k^p	$PSPACE$-hard [14, 16, 19] (2)	$PSPACE$-hard [14, 16] (2)
	k^c	$PSPACE$-hard [14] (2)	$PSPACE$-hard [14] (2)
	k^t	W[SAT]-hard	???

Table 1: The Parameterized Complexity of the Generalized Mover's Problem. A number c in parentheses after a reference, e.g., "(4)", indicates that the complexity result holds when the parameter in question is fixed at value c.

that this function could be sublinear, e.g., $O(n^{\log^3 k})$, which does not preclude algorithms more efficient than the best known today.

Our main result answers the question posed in the introduction. Indeed, this result in conjunction with previous reductions in the literature shows that robot motion planning is not f.p. tractable even when many of the parameters defined above are bounded *simultaneously* (in some cases, to constants). This suggests two directions for future research:

1. *Examine new parameters:* Though additional parameters may be defined by restricting the form of the robot further, a more fertile area for inquiry is perhaps restrictions of the robot's environment, e.g., number / surface-complexity of obstacles.

2. *Examine the effect of bounding other groups of parameters simultaneously to constants*: Problems derived by severely constraining a whole aspect of the problem, e.g., a robot composed of a constant number of disks or rectangles, may yet be tractable. For example, when the robot is a single rigid polyhedron, 3D-GMP can be solved in polynomial time [19].

Such restrictions may seem nonsensical. However, such restricted instances may still be useful for deriving approximation algorithms for the original problem. For example, one might approximate a 3-dimensional movement by a series of 2-dimensional movements in different planes. Moreover note that the problems examined in this paper are only concerned with whether or not the robot can *reach* the final position from its initial position. It must first be established if any variants of this most basic case are f.p. tractable if there is to be any hope of establishing the f.p. tractability of more realistic versions of GMP which incorporate moving obstacles, optimality constraints on motion plans, uncertainty of robot motion, and unknown environments (see [19, 20, 21] and references).

In any case, the results given in this paper suggest that parameterized complexity theory is a more appropriate tool than classical computational complexity theory for analyzing motion planning problems in robotics. By isolating the contribution of each parameter to a problem's overall complexity, parameterized analyses are an ideal tool not only for showing that certain problems cannot be solved efficiently by restricting the values of particular input parameters, but also for suggesting other parameterizations of these problems that may yet be tractable in practice.

Acknowledgements

The second author would like to thank R. Murphree, J. Reif, M. Sharir, and A. van der Stappen for various helpful e-mail conversations, M. Fellows for comments on several earlier drafts, and P. Evans, A. Idler, and B. Shea for assistance at crucial times during the preparation of this paper.

References

[1] Abrahamson, K.A. and Fellows, M.R. (1993) "Finite Automata, Bounded Treewidth, and Well-Quasiordering." *Contemporary Mathematics*, **147**, 539–564.

[2] Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hallett, M.T., and Wareham, H.T. (1995) "Parameterized Complexity Analysis in Computational Biology." *Computer Applications in the Biosciences*, **11(1)**, 49–57.

[3] Canny, J.F. (1987) *The Complexity of Robot Motion Planning*. The MIT Press; Cambridge, MA.

[4] Cesati, M. (1995) "On the Parameterized Complexity of Turing Machine Computations." Manuscript.

[5] Downey, R.G. and Fellows, M.R. (1992) "Fixed-parameter intractability (extended abstract)." In *Proceedings of the Seventh Annual Conference on Structure in Complexity Theory*. IEEE Computer Society Press; Los Alamitos, CA. 36–49.

[6] Downey, R.G. and Fellows, M.R. (1993) "Fixed-parameter tractability and completeness III: some structural aspects of the W-hierarchy." In Ambos-Spies, K., Homer, S. and Schoning, U. (eds). *Complexity Theory*. Cambridge University Press. 166–191.

[7] Downey, R.G. and Fellows, M.R. (1995) "Parameterized Computational Feasibility." In P. Clote and J.B. Remmel (eds.) *Feasible Mathematics II*. Birkhauser; Boston, MA. 219–244.

[8] Downey, R.G. and Fellows, M.R. (1995) "Fixed-parameter tractability and completeness II. On completeness for $W[1]$." *Theoretical Computer Science*, **141**, 109–131.

[9] Downey, R.G., Fellows, M.R., Kapron, B.M., Hallett, M.T., and Wareham, H.T. (1994) "Parameterized Complexity of Some Problems in Logic and Linguistics (Extended Abstract)." In A. Nerode and Y.V. Matiyasevich (eds.) *Logical Foundations of Computer Science*. Lecture Notes in Computer Science no. 813. Springer-Verlag; Berlin. 89–101.

[10] Fellows, M.R. (1989) "The Robertson-Seymour theorems: a survey of applications." *Contemporary Mathematics*, **89**, 1–18.

[11] Fellows, M.R. and Langston, M.A. (1992) "On well-partial-ordering theory and its applications to combinatorial problems in VLSI design." *SIAM Journal of Discrete Mathematics*, **5**, 117–126.

[12] Garey, M.R. and Johnson, D.S. (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company; San Francisco.

[13] Hallett, M.T. and Wareham, H.T. (1994) "A compendium of parameterized complexity results." *SIGACT News*, **25(3)**, 122–123.

[14] Hopcroft, J., Schwartz, J.T. and Sharir, M. (1984) "On the Complexity of Motion Planning for Multiple Independent Objects: $PSPACE$-Hardness of the 'Warehouseman's Problem'." *The International Journal of Robotics Research*, **3(4)**, 76–88.

[15] Hopcroft, J. and Ullman, J.D. (1979) *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley; Reading, MA.

[16] Joseph, D.A. and Plantinga, W.H. (1985) "On the Complexity of Reachability and Motion Planning Problems." In *Proceedings of the First ACM Symposium on Computational Geometry*. ACM Press; New York. 62–66.

[17] Lewis, H.R. and Papadimitriou, C.H. (1982) "Symmetric Space-Bounded Computation." *Theoretical Computer Science*, **19**, 161–187.

[18] Papadimitriou, C.H. (1994) *Computational Complexity*. Addison-Wesley; Reading, MA.

[19] Reif, J.H. (1987) "Complexity of the Generalized Mover's Problem." In Schwartz *et al.* (eds.) (1987). 267–281.

[20] Reif, J.H. and Sharir, M. (1994) "Motion Planning in the Presence of Moving Obstacles." *Journal of the ACM*, **41(4)**, 764–790.

[21] Schwartz, J.T. and Sharir, M. (1990) "Algorithmic Motion Planning in Robotics." In J. van Leeuwen (ed.) *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*. The MIT Press; Cambridge, MA. 391–430.

[22] Schwartz, J.T., Sharir, M. and Hopcroft, J. (eds.) (1987) *Planning, Geometry, and Complexity of Robot Motion*. Ablex Publishing Corporation; Norwood, NJ.