Dealing with Complexity:
Integrated vs. Chunky Search Processes*

Oliver Baumann
Department of Marketing & Management
University of Southern Denmark
Campusvej 55
5230 Odense M, Denmark
Tel: +45 6550-4433
Fax: +45 6615-5129
oliv@sam.sdu.dk

Nicolaj Siggelkow
2211 Steinberg Hall-Dietrich Hall
Wharton School
Philadelphia, PA 19104
Tel: (215) 573-7137
Fax: (215) 898-0401
siggelkow@wharton.upenn.edu

* For helpful comments, we thank Richard Burton, Felipe Csaszar, Dan Levinthal, Christoph Loch, and two anonymous referees. We are grateful to the Mack Center for Technological Innovation and the German Academic Exchange Service for generous funding.

Dealing with Complexity:
Integrated vs. Chunky Search Processes

Abstract: Organizations are frequently faced with high levels of complexity. While the importance of search for dealing with complex systems is widely acknowledged, how organizations should structure their search processes remains rather unexplored. This paper starts to address a basic question: how much of the entire system, and thus complexity, should be taken into consideration at any given time of a search process? Should a problem solver pursue an integrated search and be concerned with the whole system right from the start, or should a problem solver incrementally expand the "search domain" – the subset of system elements and interdependencies that are included in the search efforts, and if yes, how "chunky" should these steps be? Our analysis of a simulation model yields four insights: (1) expanding the search domain in smaller steps can yield a distinct advantage in final system performance; (2) following a completely incremental expansion pattern is not necessary as long as larger chunks are added early on in the process; (3) the value of chunky search is particular high if highly influential system elements are considered first, while highly dependent elements are added later; (4) under time pressure, chunky search can lose its performance advantage over more integrated search processes. We discuss the implications of our findings for managing organizational search and complex systems, more broadly.

Keywords: Complex systems, organizational search, search structuring, organizational problem solving

1 Introduction

When faced with a complex problem composed of many interdependent decisions, should an organization try to tackle the problem as a whole, or should it break the problem apart? Organizations are frequently faced with complex problems in domains such as product development, organization design, or strategic management. Dealing with complexity, however, is far from straightforward since problem solvers cannot optimize each part individually. Consider, for instance, the challenge faced by a manager who needs to change a firm's set of activities after a large environmental shock. A change to activities at one stage of the value chain may have effects on other parts of the value chain and require further adjustments there as well. Thus, optimizing individual stages of the value chain, without taking interdependencies into account, is not likely to yield a high-performing new combination of activity choices. Similarly, consider a team designing a complex product with many interacting components. If the team wants to create a high-performing product, it cannot optimize components individually but needs to pay attention to the entire system.

As documented by a large body of research, problem solvers often do not deal with complex systems by calculating optimal solutions, but engage in an adaptive search for satisfactory ones (March and Simon 1958; Cyert and March 1963; Nelson and Winter 1982). Even though the role of search in approaching complex systems is widely acknowledged, how such search processes should be structured remains rather unexplored (Fleming 2001). In particular, problem solvers face the question how much of the entire system, and thus complexity, should be taken into consideration at any given time of a search process. Should a problem solver be concerned with the whole system right from the start and always focus on improving overall system performance? Or should a problem solver use initial search efforts to learn about a few elements and interdependencies, in the interim de-emphasizing other elements, and, only subsequently, grow the number of elements and interdependencies that are actively considered? For instance, should a manager attempt to find a good holistic set of new activities right away, or should she expand the range of activities that she considers in her search efforts in a more incremental manner?

Likewise, should a product design team always consider all components simultaneously, searching for designs that have high overall performance, or should it first experiment with a subset of components and expand this set gradually in the course of the design process?

The case of Team New Zealand's efforts in designing a yacht for the America's Cup (Iansiti and MacCormack 1996) provides an illustrating example for the latter approach. As any racing yacht, it was comprised of four essential elements: the hull, the keel, the mast, and the sails. Because these components are highly interdependent, many critical tradeoffs need to be made during the design process to come up with a high-performing yacht, a light, yet stable boat with a low drag factor. At the same time, small performance differences tend to separate the best boats from the worst. In order to meet these challenges, Team New Zealand applied a particular search pattern: Initially, the team focused exclusively on learning about the design of the hull, applying only very simple keel variations and ignoring the mast and the sails. Only when a robust design for the hull and the (simple) keel had evolved, did the team move from designing prototypes (that were limited to some aspects of the overall system) to the design of a real yacht, which then allowed the design team to tweak also the previously neglected design dimensions.

In this paper, we study the consequences that arise when the "search domain" – the subset of system elements and interdependencies that problem solvers include in their search efforts – is expanded in different ways. When is it beneficial to search in an integrated way, and when is it beneficial to increase the search domain in several steps? If the search domain is increased in several steps, how "chunky" should these steps be?[1] Should the steps be of equal size, e.g., first focusing on one half of a system and then also taking the second half into account? Or should early additions to the search domain be small and later additions be large; or conversely, should the early additions be large and the later ones be small? We use a simulation model to address these questions. Our model contains problem solvers who search for good solutions to complex problems. By endowing the problem solvers with different patterns of how

[1] With "chunking," we denote the expansion of a search domain by a number of new elements (a "chunk"). In cognitive psychology, in contrast, the term "chunking" refers to the process of (mentally) grouping together similar pieces of information and storing or processing them as a single object (Miller 1956).

they expand their search domains, we analyze the effect of different expansion patterns on the performance of the search process.

Our analysis yields four insights. First, our results point to the possible advantage of incremental expansion over fully integrated search: When problems are complex, expanding a search domain in steps rather than trying to improve a system in its entirety from the beginning can yield a distinct advantage in final system performance. However, to reap the value of the incremental expansion pattern, a problem solver must proceed with gradually decreasing parochialism, initially focusing on only a few system elements and their performance, and widening this focus over time until all elements and performance dimensions are considered. Second, we find that it is not necessary to follow a completely incremental expansion pattern. Instead, final system performance is often not materially affected if the expansion pattern contains incremental steps as well as larger chunks. However, this only holds if a problem solver starts with large chunks and makes small additions to the search domain subsequently; starting with a small search domain and adding larger chunks later leads to lower performance. Third, if a complex system contains elements that either affect, or are affected by, many other elements, the value of incremental expansion is particular high when the highly influential elements are added first and the highly dependent elements are added later in the expansion process. Adding the dependent elements early and the influential elements late, in contrast, creates a significant performance penalty. Finally, we reveal the tradeoffs that the incremental expansion of a search domain entails. To reap the performance advantage of this search pattern, a problem solver requires more time than needed for an integrated search. Allocating the available time appropriately across the different expansion steps can help smoothen this tradeoff to some extent.

The paper is organized as follows. The next section reviews prior research. Section 3 describes the model, while section 4 presents the simulation results. Section 5 discusses the findings and concludes.

2 Prior research

For complex systems, limitations of time, resources and cognitive abilities render an exhaustive search across all potential solutions generally infeasible, because the number of possible ways to configure a system, i.e., the combinatorial complexity of possible solutions for the problem, grows exponentially with the number of elements that need to be considered (Simon 1962). Consequently, a problem solver with bounded rationality (Simon 1955; 1956) must search selectively to find a solution, i.e., a "good" combination of the elements that comprise a system (Newell and Simon 1972; Simon 1996). Moreover, in these circumstances, problem solvers tend to search locally and find a local optimum (March and Simon 1958; Cyert and March 1963; Nelson and Winter 1982), as documented by work in fields like organization and strategy (Stuart and Podolny 1996; Katila and Ahuja 2002; Rosenkopf and Almeida 2003), technological evolution (Fleming 2001; Arthur 2009), or cognitive psychology (Newell and Simon 1972; Frensch and Funke 1995; Gigerenzer et al. 1999).

While various scholars of problem solving in complex systems agree that the structuring of the search process can have important effects on the performance of the resulting solution (e.g., Thomke et al. 1998; Fleming 2001; Katila and Ahuja 2002; Nickerson and Zenger 2004), our notion of expanding a search domain has not been researched systematically. Prior work on organizational search processes can, however, provide some information, as it has dealt with two topics that are closely related to choosing and expanding a search domain: (1) the value of taking an integrated, system-level perspective when searching for solutions in complex systems, and (2) the effects of (temporarily) constraining the search space by focusing attention on only a subset of the relevant elements and interdependencies.

As for the first topic, a large body of research on the design of complex product and organizational systems has studied the benefits of keeping interdependent system elements together and addressing them simultaneously, thus pursuing an integrated search process (e.g., Thompson 1967; Burton and Obel 2004; Ulrich and Eppinger 2007). Since modifications of one system variable may affect other variables, using a non-integrated approach can lead to "problem-solving oscillations," where problem solvers go back and forth between various local adaptations, creating a negative effect on a project's schedule, budget, or

overall performance (Terwiesch and Loch 1999; Mihm et al. 2003). Hence, only if a system is (nearly) decomposable, i.e., if interdependencies can be grouped into relatively independent modules, can search proceed in an (almost) independent manner, with each module being designed individually (Simon 1962; Baldwin and Clark 2000; Ethiraj and Levinthal 2004; MacCormack et al. 2006).

Our model extends this body of research by analyzing problem solvers who eventually follow an integrated search process, i.e., are concerned with all elements and interdependencies that make up a system, but that do not start with integrated search right away. Instead, they differ in the path they follow toward an integrated search, i.e., in their patterns of expanding their search domains.

The second related topic, (temporarily) constraining the search space, has been addressed from two major perspectives (Knudsen and Levinthal 2007): a) how new decision alternatives are generated, and b) how they are evaluated. For instance, Siggelkow and Levinthal (2003) have argued that a temporarily decentralized organizational structure – in which new decision alternatives are generated and evaluated independently in different departments of a firm – can lead to superior search outcomes than structures that are either permanently centralized or permanently decentralized. As they point out, the temporary constraints on the evaluation of new activity choices that ignores interdependencies between the decentralized units can help a firm avoid "getting stuck" on the first set of consistent choices that is found and prolong the search for better alternatives. However, they also stress the need for an eventual integration of the subsystems that ensures that all elements and interdependencies of the system will be evaluated as a whole again.

Our model builds upon this body of research by assuming that problem solvers evaluate novel candidate solutions by taking only those elements into account that their search domain includes. However, we investigate in more depth the consequences of different patterns of expanding the search domain. Moreover, our model also addresses the temporal aspects of these search patterns, i.e., how well these search patterns perform if a problem solver has only a limited amount of time available, and also considers the role of the underlying interaction pattern.

3 Model

The theoretical focus of our paper is on problem solving in complex systems by problem solvers who are boundedly rational and who need to search for good solutions. Specifically, our goal is to systematically explore the implications that arise when problem solvers expand their search domains in different ways. For our simulation model to correspond with our theoretical focus, we need to address three basic aspects: (1) complex systems, (2) how problem solvers search for performance-improving alternative solutions, and (3) a representation of the different patterns by which problem solvers expand their search domains. In modeling these aspects, we focus on the interplay of three key characteristics: the expansion of a search domain over a set of interdependent elements, the resulting (sub-) system performance, and the time available or necessary for the search and expansion process. To describe our model in concrete terms, we will use the product design context, as for instance the case of Team New Zealand, as our main setting. Accordingly, we call the problem solver in our model a "design team".[2] Given the abstract nature of our research question and model, our results, however, can also be applied to other instances of organizational search, as we will discuss in section 5.

3.1 Complex systems

To model complex systems, we create, in a stochastic manner, high-dimensional solution spaces with numerous local optima and little correlation between "similar" system combinations. To do so, we adapted Kauffman's *NK* model (Kauffman 1993; 1995), a model that was originally developed in evolutionary biology and that has become a canonical model in the management sciences to represent problems with tunable complexity (e.g., Levinthal 1997; Rivkin 2000; Ethiraj and Levinthal 2004; Siggelkow and Rivkin 2005; Lenox et al. 2006).[3]

Specifically, we conceptualize design teams to be facing a set of interdependent design elements that

[2] Our model does not, however, contain any "intra-team dynamics," such as political infighting or shirking. Thus, the term "team" could also be replaced by the term "designer" or more generally "problem solver."

[3] The value of the *NK* model does not stem primarily from how it serves to generate complex systems, as other stochastic techniques might be also applied for this purpose (see, e.g., Winter et al. 2007). Rather, it conveys the advantage that its general properties are already well-understood, which makes it easier to evaluate and understand the findings of models that build upon the basic structure.

lead to system performance. The complex systems that the teams are designing require them to make decisions of how to configure the various elements. Since the value of one element may depend on how other elements are configured, system performance may respond in nonlinear ways to changes of individual elements, as is typical for complex design problems. In our model, each team must make choices with respect to a set of N binary design elements a_i, i.e., $N = \{a_1, a_2, \ldots, a_N\}$. For instance, a_1 may represent difference choices for the hull design, a_2 different choices for the keel design, etc. Thus, the design space of a team consists of a total of 2^N possible combinations of choices, each of which can be represented by a binary vector $\boldsymbol{a} = (a_1, a_2, \ldots, a_N)$. In the model, each element a_i makes a contribution c_i to system performance $V(\boldsymbol{a})$. To represent a high degree of interdependence between the elements, we assume that each performance contribution c_i depends on both how element i is configured and on how all other $K = N$-1 elements are configured. Thus, each performance contribution c_i is a function of the entire combination of design choices $\boldsymbol{a}$, i.e., $c_i(\boldsymbol{a})$. (In section 4.2, we study situations of less pervasive interdependencies, i.e., lower values of K; in section 4.3, we consider the case of interdependencies that are not randomly distributed, but exhibit certain patterns.) For instance, in the case of two elements (N = 2, K = 1), the performance contribution of element 1, c_1, would have four potential values: $c_1(0, 0)$, $c_1(0, 1)$, $c_1(1, 0)$, and $c_1(1, 1)$, depending on how each of the two elements are configured (0 or 1). The same holds for the performance contribution c_2 of element 2. Particular values for all possible c_i's are determined by drawing randomly from a uniform distribution over the unit interval, i.e., $c_i(\boldsymbol{a}) \sim u[0;1]$. Finally, the overall performance of a given set of choices $\boldsymbol{a}$ is calculated as the average of its N performance contributions: $V(\boldsymbol{a}) = [c_1(\boldsymbol{a}) + c_2(\boldsymbol{a}) + \ldots + c_N(\boldsymbol{a})] / N$.

It has become common to interpret the payoffs to the different combinations of choices as a performance landscape (Wright 1932; Levinthal 1997). A performance landscape represents a high-dimensional space, in which each of the N elements that the team needs to configure represents one "horizontal" axis, each offering different options (two in our case). The "vertical" or performance axis then maps each possible combination of choices $\boldsymbol{a}$ to a performance value $V(\boldsymbol{a})$. Hence, while each "point" on the landscape represents one particular combination, its "height" denotes the corresponding

performance. The goal of problem-solving search is to reach a high point on the performance landscape, i.e., to find a combination of choices that together create high performance. Interdependencies among the elements, however, result in rugged landscapes with numerous local peaks, making the search for a high peak difficult (Kauffman 1993; Levinthal 1997).

3.2 Problem-solving search

Following prior modeling efforts (e.g., Levinthal 1997; Rivkin 2000), we embed local search in our model by restricting teams to consider only alternatives that differ from the current combination of choices in one element. (In section 4.2, we also discuss extensions in which we relax this assumption, providing teams with higher degrees of creativity and foresight.) For example, if $N = 8$, a team that considers the entire system and that is currently at choice combination 00000000 would have eight local alternatives that it might consider: e.g., 00000001, 00010000, or 10000000.

At the beginning of the simulation, each design team starts with a randomly chosen choice combination. (In section 4.2, we also cover the case in which teams might have better or worse prior knowledge that would allow them to start at more or less advantageous initial choice combinations.) In each subsequent period, the team randomly picks one new local alternative that it has not experimented with so far, evaluates its performance, and compares it with the performance of its current choice combination. If the alternative yields a higher performance than the current combination, the team implements it and in the next period continues to search from the new choice combination. If the performance of the alternative is inferior, it is discarded (yet memorized) and a new design alternative is generated and assessed in the next period. The team stops its search once it has implemented a combination that cannot be further improved through any of the local alternatives. Following Rivkin and Siggelkow (2003), we call choice combinations at which search terminates the “sticking points” of a team.

3.3 Search domains

3.3.1 Definition

We define a search domain as a subset of design elements $N_D = \{a_1, a_2, \ldots, a_D\}$ such that

a) new alternatives that a team considers involve only local changes within these D elements, and

b) new alternatives are evaluated only with respect to the contributions of these D elements: $V_D(\boldsymbol{a}) = [c_1(\boldsymbol{a}) + c_2(\boldsymbol{a}) + \ldots + c_D(\boldsymbol{a})] / D$. (In section 4.2, we also relax this assumption.)

Following from a), given a search domain of size D, the team's design space consists of 2^D possible combinations of choices $\boldsymbol{a}_D = (a_1, a_2, \ldots, a_D)$. The other $N - D$ elements, in contrast, remain fixed at their initial (randomly chosen) state. One could say that these elements are implemented without attention to detail. Only once an element is included in the search domain does the team consider making changes to it. Following from b), it is important to note that even though the team is only concerned with the contributions of the D elements within the search domain when evaluating alternatives, the contributions of these D elements are still affected by how the entire system is configured, i.e., each c_i is a function of the full vector $\boldsymbol{a}$. For instance, say $N = 8$ and the current choice combination is 00000000. Assume the search domain contains the first three design elements. In this case, the design team would pick from the alternatives 10000000, 01000000, and 00100000. If the team chooses the first one, it would evaluate this alternative as $V_D(10000000) = [c_1(10000000) + c_2(10000000) + c_3(10000000)]/3$

In sum, in constructing alternatives, the team focuses its attention on making adjustments to only a subset D of all design elements. When evaluating whether an alternative is attractive, only the performance of these D elements is taken into account. In this sense, the team behaves "parochially" with respect to its search domain: As long as it has not expanded to control all elements of the system, it may implement a set of choices that is performance-enhancing from the point of view of its current search domain, but performance-decreasing if all elements and performance contributions were taken into account. As a result, while a team may have stopped its search given its current search domain, once the team expands its search domain, it may resume its search again.

3.3.2 Different expansion patterns

Our goal is to study the relative performance of teams that eventually configure the same full system, comprised of all N design elements, but that follow different patterns of expanding their search domains along the way. In our model, each team starts with an initial search domain and then increases the size of the search domain D by adding design elements that have not been under the team's control until $D = N$. One way to characterize the different expansion patterns is to consider (1) the size of the initial search domain D_0, (2) the number of expansion steps, and (3) the number of elements D_s that are added at each step s, i.e., the chunking sequence.

We use the following notation: We label each expansion pattern with an expression such as {5+3} or {1+1+1+1+4} (for the case of $N = 8$). The {5+3} pattern would denote a team whose initial search domain contains the first five design elements and that subsequently, in a single expansion step, expands its search domain by the remaining three elements. For instance, assume that search started at 00000111. At this point, the team might consider alternatives such as 10000111 or 01000111, experimenting with local changes to the first five design elements but always holding the last three elements constant. Likewise, when evaluating, for instance, the first alternative, the team would compute $V_D(10000111) = [c_1(10000111) + c_2(10000111) + c_3(10000111) + c_4(10000111) + c_5(10000111)]/5$, thereby ignoring the contributions of the last three elements. Only after the expansion step would the team consider changes to the last three elements and take the contributions of all eight elements into account.[4] In contrast, with the {1+1+1+1+4} pattern, the team starts with a search domain that contains only the first element. Initially, the team considers only alternatives to the first design element and cares only about the contribution of this choice; it subsequently increases its search domain by the second design element, thus now considering alternatives (and caring about the contributions) that involve both the first and the second design elements. The search domain is subsequently expanded three more times, twice by adding one

[4] We assume that whenever a team increases its search domain, it requires the first period after the increase to learn about the current choice combination and the performance contributions of the new choices. Starting in the subsequent period, the team continues to search for better combinations in its expanded domain. This assumption has been made for plausibility reasons and does not influence the outcome of the model in a qualitatively significant way.

more element, and at the end by a chunk containing the remaining four design elements. Thus, in this case the design team engaged in four expansion steps.

In the case of $N = 8$, two extreme patterns are worth noting: the {1+1+1+1+1+1+1+1} pattern and the {8} pattern. For compactness, we will use an ellipse (…) to denote any string of incremental "+1" expansion steps. Thus, the fully incremental pattern {1+…+ 1} has the maximum number of expansion steps (seven). The fully integrated pattern {8} does not have any expansion of the search domain at all. In this case, the team always considers the full system. One should note that all expansion patterns effectively end in a fully integrated {8} state, since after the last expansion step all teams are considering local alternatives that involve possible changes to any of the eight design elements, and are evaluating alternatives by taking into account the contributions of all eight elements. Thus, while expansion patterns have a phase in which parochial decisions with respect to subsystems are made, this parochialism gradually declines until the search domain comprises the full system.

4 Results

In the following, we report results for the case of $N = 8$, $K = 7$. In all simulations, we equipped each team with a specific expansion pattern and placed it on a random starting point on the performance landscape, letting it search for a specified number of periods. To make sure that the differences we find are inherent to our model and do not result from any stochastic influence, we repeat each experiment for 10,000 performance landscapes. We always measure system performance relative to the performance at the global peak (the global maximum) of a particular landscape, i.e., the performance is 1.0 if the team reaches the global peak.[5] Unless indicated otherwise, reported performance differences are significant at the 0.001 level. Furthermore, as we are only interested in how well different expansion patterns cope with the same system (comprising all $N = 8$ elements), we only report overall (i.e., system-level) performance values.

[5] We determine the global peak of each performance landscape by evaluating all points of each landscape.

4.1 Performance effects of different expansion patterns with no time constraints

4.1.1 Core result

We start our analysis with a comparison of the two extreme patterns, integrated search {8} and fully incremental expansion {1+…+1}, as well as a number of intermediate patterns. These patterns, for example {6+1+1} or {3+5}, differ in their number of expansion steps and in the sizes of the chunks by which the search domains are expanded, i.e., their chunking sequence. In the initial stage and after each subsequent expansion we provide each team with enough time to search for improvements until it has reached a sticking point from which it does not move. At that point, the team expands its search domain (by the amount stated by its expansion pattern) and resumes its search. (In section 4.4, we model different ways of when a team decides to engage in the expansion of the search domain.) Figure 1 illustrates the final performance of the different expansion patterns, while Table 1 shows, for a selection of different expansion patterns, which performance differences are statistically significant from each other.

< Insert Figure 1 and Table 1 about here >

The results of Figure 1 point to the relevance of the number of expansion steps and the chunking sequence: First, we can observe a general trend of decreasing performance with a decreasing number of expansion steps. For instance, the fully incremental pattern {1+…+1} with seven expansion steps achieves a performance of 0.876, while pattern {4+4} with one expansion step achieves a performance of 0.864, and lastly, pattern {8} with no expansion step achieves only a performance of 0.860. Second, the results indicate that holding the number of expansion steps constant, the chunking sequence can have a very significant effect on performance. For instance, pattern {5+1+1+1} with three expansion steps significantly outperforms pattern {1+1+1+5}, also with three expansion steps (0.875 vs. 0.864). In general, it is advantageous to start with a big search domain and then add smaller chunks rather than to start with a small search domain and then add bigger chunks. As a matter of fact, up to a certain size of the initial search domain, the system's final performance is not materially affected by the early chunking. In our case, the performance differences between the fully incremental pattern {1+…+1} and the patterns

that start with a larger search domain are not statistically significant with $p < 0.05$ until the initial search domain has size 5, and different with $p < 0.001$ until the initial search domain has size 6 (see Table 1).[6]

4.1.2 The role of gradually decreasing parochialism

What drives the result that teams that expand their search domains in an incremental manner outperform those that start right away searching across the entire target system? As both types of teams end up searching the landscape locally in an integrated manner, they will ultimately get stuck on local peaks on the landscape, i.e., choice combinations $\boldsymbol{a}$ such that $V(\boldsymbol{a}) > V(\boldsymbol{a'})$ for all $\boldsymbol{a'}$ that differ from $\boldsymbol{a}$ in only one element. However, as the first two rows of Table 2 indicate, teams that followed an incremental expansion pattern are more likely to reach the global peak of the landscape, and, even if they did not reach the global peak, they ended up on local peaks with higher average performance than teams that followed a less incremental search process. For instance, contrasting the extreme patterns, 6.96% of teams that followed the expansion pattern {1+…+1} reached the global peak, whereas only 5.35% of teams that followed pattern {8} reached the global peak. Likewise, the teams that followed {1+…+1} and did not reach the global peak, finished on local peaks with an average performance of 0.868, while teams that followed {8} finished on average on lower local peaks, yielding an average performance of only 0.852.

< Insert Table 2 about here >

The driver creating this difference between the two search patterns is the gradually decreasing parochialism of the incremental expansion pattern, which has two main effects: First, it creates a broader search over possible choice combinations than the integrated search pattern. In the early stages, a team that follows {1+…+1} has to worry only about whether a possible change affects the performance of the

[6] While Figure 1 reports the performance of all "extreme" expansion patterns (such as the {N+1+…+1} patterns that consist of an initial chunk that is followed by a sequence of fully incremental steps), it represents only a subset of all possible expansion patterns. To corroborate our finding that the efficient frontier (given different numbers of expansion steps) is established by the {N+1+…+1} patterns, we proceeded as follows: We first analyzed all possible expansion patterns that have one expansion step (seven possible patterns), two expansion steps (21 possible patterns), five expansion steps (21 possible patterns) and six expansion steps (seven possible patterns); furthermore, we analyzed, in an exhaustive manner, every expansion pattern for the case of $N = 4$, $K = 3$. Both sets of experiments showed that, for any given number of expansion steps, the highest performance is achieved by the {N+1+…+1} pattern and the lowest performance by the {1+…+N} pattern. Details are available from the authors.

sub-system it is currently working on (e.g., after three expansion steps, the first four design elements), and not whether the change would decrease overall system performance. This initial parochialism loosens search constraints and creates overall broader search, thereby increasing the likelihood of finding the global, or a high local peak. The third and fourth rows of Table 2 provide evidence for the broader search. Teams that followed {1+…+1} evaluated on average 50.7 different design alternatives before they reached their final design. In contrast, teams that pursued the fully integrated pattern {8} evaluated on average only 13.4 alternatives before they terminated their search.[7] As another metric for exploration, we consider the total number of contribution values (c_i's) that have been considered. (Recall, a contribution value answers the question, "How valuable is element *i*, given that the other elements are configured in their current way?") Overall, teams that followed expansion pattern {1+…+1} assessed 125.2 contribution values, while teams that pursued {8} evaluated only 106.9 contribution values. In sum, the incremental expansion paths, with initial parochial search, create broader exploration, which is useful given the very rugged landscapes that the teams are searching.

Broad search, however, is only one half of the story. (As we will show in section 4.2, if a parochial search pattern induces high levels of search but does not eventually provide coordination across the entire system, performance suffers significantly.) The second beneficial effect of gradually decreasing parochialism is that teams following {1+…+1} are able to avoid low-performing combinations of choices that are sticking points for teams that search in an integrated manner, and gravitate towards higher local peaks instead. A team that follows {8} will get stuck on the first local peak it encounters, including very low ones. In contrast, should a team that pursues {1+…+1} encounter this low performing combination prior to the last expansion step, it is quite possible that a parochial change to the subsystem that the team is currently considering will be performance-enhancing to the subsystem, letting the team move away from this choice combination. For instance, we find that the likelihood that a team pursuing {1+…+1}

[7] One should note that for a team that follows the integrated pattern, each "evaluation" involves the value of all eight design variables. The evaluations made by a team that follows the incremental pattern involve different numbers of design variables, depending on the team's stage in the expansion process. Our next metric takes this difference between the two patterns into account.

will get stuck on the lowest local peak on a landscape is less than half of the likelihood that a team pursuing {8} will get stuck on that peak.[8]

In addition, even though a team pursuing {1+…+1} will move off low local peaks, its search is still guided. Since the team is attracted to high "local peaks" in its current search domain, it is likely to reach choice combinations that have relatively high system performance and that serve as good starting points for subsequent integrated search since they provide a (relatively high) lower bound on performance.[9] As the results in Table 2 show, similar effects concerning search and the ability to avoid low sticking points help explain the performance differences among the expansion patterns that lie between the fully incremental expansion pattern and the fully integrated search. These results, thus, also clarify our findings that starting with a big search domain and adding smaller chunks subsequently is more advisable than the converse pattern, and that up to a certain chunk size, starting the search with a large chunk does not significantly affect final system performance.

4.2 Boundaries of the core result

To establish the boundaries of our core result, we explored five major elements of our model: 1) the degree of localness of the search process, 2) the quality of the initial knowledge held by the design teams, 3) the type of parochialism that the teams exhibit when evaluating alternatives, 4) the size of the design problem, and 5) the complexity of the design problem. In the respective panels of Figure 2, we report the performance difference relative to the {8} pattern, i.e., how our core result – the performance advantage of the fully incremental pattern over integrated search – is affected when each element of the model is modified individually.

< Insert Figure 2 about here >

[8] Moreover, one can show that the probability that a team that follows {1+…+1} moves off a local peak is a decreasing function of local peak height. Details are available from the authors.

[9] After each expansion, teams will only adopt changes if they are performance-enhancing for the current subsystem. In particular, after the last expansion step, they will only implement changes if overall system performance increases. Thus, the performance at that time, which is generated by a sticking point of the "subsystem" they have been searching up to that time, forms the lower bound of performance. As a result, the range of local peaks that this team will end up with (either directly or by tweaking further elements) is truncated at the lower end. Numerical details are available from the authors.

4.2.1 Localness of search

In our main model, design teams only consider local changes to their existing design. We now assume that design teams are cognitively less severely bounded in the sense that they can contemplate design alternatives that differ more radically from their status quo combination of the design variables. In particular, we assume that design teams have a "search radius" of 2 or 3 and thus can identify and evaluate alternatives that simultaneously differ in up to two (or three) choices from any given status quo.[10] As one would expect, a higher search radius, which creates broader exploration, results in higher performance for all expansion patterns. At the same time, the performance advantage of the fully incremental expansion pattern over the integrated one prevails but is reduced (see Panel A of Figure 2). Thus, the less bound design teams are to local search, the smaller the performance benefit will be that can be reaped by expanding a search domain in an incremental manner.

4.2.2 Prior knowledge

In our main model, teams started with a randomly chosen combination of design choices. We next explore the notion that design teams may differ in their degree to which they have some prior understanding of the structure of the performance landscape, i.e., different "cognitive maps" as a result, for instance, of related experience or analogical reasoning (Gavetti and Levinthal 2000; Gavetti et al. 2005). Following Kauffman, Lobo, and Macready (2000) and Gavetti, Levinthal, and Rivkin (2005), we operationalize prior knowledge and cognition as the quality of the initial starting point for search: the better the prior knowledge, the better-performing is the initial combination of choices.

As one might expect, the quality of the starting point has an effect on the eventual performance. For instance, teams that start with a performance that is in the lowest 25% of all starting points and that pursue a fully integrated pattern {8}, achieve a final performance of 0.852. In contrast, teams that start in the highest 25% achieve a final performance of 0.877. Yet similar positive effects can be observed for

[10] For instance, a design team whose search radius is 2 and whose choice combination is 00000000 (given $N = 8$) might consider options as different from the status quo as 11000000 or 10000001 (or alternatives closer to the status quo, such as 00000001). If the search radius is 3, options such as 1101000 or 10001001 can be contemplated.

teams that use the fully incremental expansion pattern {1+…+1}. In Panel B of Figure 2, we report the implications that different starting point qualities have on the performance advantage of the fully incremental expansion pattern {1+…+1} over the fully integrated pattern {8}. We find that if teams start their search with a performance within the lowest 25th percentile of starting values, the performance advantage of the incremental pattern is, on average, the same as for a randomly chosen starting point (about 0.017). For starting point qualities that range between the 25th and the 50th percentile, the benefit of incremental expansion is slightly higher (0.018), and rises further for starting points between the 50th and 75th percentile (0.021). For starting point qualities that belong to the top 25%, the performance advantage of the incremental pattern is somewhat lower (0.013) than for an average starting point as reported for the main model (0.017). Thus, our core result – the performance advantage of the fully incremental expansion pattern over the fully integrated pattern – remains robust with respect to the quality of the initial starting point.

4.2.3 Type of parochialism

In our main model, teams following an incremental expansion pattern evaluated alternatives with respect to subsystem performance. Over time, this parochial evaluation decreased until the search domain comprised the full system (i.e., after the last expansion step) at which point the teams evaluated alternatives with respect to full system performance. As elaborated in section 4.1.2, this gradually decreasing parochialism drives the performance advantage of the incremental patterns over integrated search. (The baseline performance of our main model with "gradually decreasing" parochialism is shown in Panel C (left node).)

To further explore the interaction of parochial evaluation and incremental expansion, we model two additional search approaches that likewise exhibit elements of incrementalism, but at the same time differ in their type of parochialism from our teams in the baseline case. We dub the first approach a "non-parochial incremental" expansion pattern. A team following this pattern expands the space of alternatives in which it searches for better local alternatives in the same way as the {1+…+1} pattern. Thus, it starts

by contemplating local changes only to the first element; then to the first two elements, etc. When evaluating new alternatives, however, the team always considers overall system-level performance $V(\mathbf{a}) = [c_1(\boldsymbol{a}) + c_2(\boldsymbol{a}) + \ldots + c_N(\boldsymbol{a})] / N$, rather than just subsystem performance, thereby also taking into account the contributions of elements that are currently not considered to be changed. As shown in Panel C (middle node), if a team follows this pattern, the performance advantage over the integrated search {8} disappears. This result arises because even though the range of alternatives slowly grows over time, the team always chooses among alternatives as if following the integrated search {8}. As a result, the team never implements any alternative that – although performance-improving if only the elements of the team's current expansion stage were considered – is performance-decreasing from the point of the overall system. Put differently, non-parochial incremental patterns do not reduce the constraints for alternatives to be accepted. Consequently, they do not trigger design teams to search more broadly and do not help them avoid low-performing local peaks that are sticking points for the {8} pattern. Thus, to have any beneficial effects on search, parochialism is required in the evaluation of alternatives.

The second new search approach we study does not gradually decrease its parochialism but is "always parochial." Here, the team focuses always on only one element at a time, and always takes only the performance contribution of the particular element into account. For instance, if the choice combination is 11000011, the team would first consider changing the first element to 0 or not, looking only at the performance contribution of the first element ($c_1(\boldsymbol{a})$). It would then think about changing the second element to 0, again taking into account only the contribution of the second element ($c_2(\boldsymbol{a})$). This process continues with the first element once the eighth element has been considered. The search that is generated by a team following this pattern is in fact even broader than that of the {1+…+1} pattern (e.g., in terms of the number of evaluations that are made or design alternatives that are generated). As Panel C (right node) shows, however, the resulting performance of the "always parochial" pattern is considerably lower even than that of integrated search {8}, because the team suffers from never taking a system perspective. Given that elements interact, the team's "improvements" to subsystems never accumulate at

the system level but undermine performance.[11] Hence, while initial parochialism is necessary to create beneficial search, continued parochialism is destructive.

4.2.4 Size of the design problem

In Panel D of Figure 2, we show the results of systematically increasing the system size from $N = 4$ to $N = 16$, while keeping interdependence high ($K = N$-1). We find that, regardless of system size, the fully incremental expansion pattern always significantly outperforms the integrated pattern. Thus, our core effect is robust and increases in the size of the design problem.

4.2.5 Complexity of the design problem

Lastly, in Panel E, we vary the degree of interdependence (K) between the variables that make up the design problem, letting K range from 0 to N-1 while keeping N fixed at 8. When $K < N$-1, each element interacts with K randomly chosen other elements. We find that for all values of $K > 0$, the incremental expansion pattern significantly outperforms the integrated pattern. However, as design problems are getting less complex (decreasing K), the performance advantage of expanding the search domain in incremental steps decreases. High levels of interdependence (high K) create rugged performance landscapes, and consequently, the broader search that the incremental expansion pattern generates is more valuable. In contrast, for lower levels of complexity, performance landscapes are less rugged, and the additional search that is induced by the incremental expansion pattern becomes less relevant. For $K = 0$, all teams, regardless of expansion pattern, reach the global peak; as a result there is no performance difference.

In sum, this range of analyses reveals three conditions that need to be jointly present for our main result to hold. The incremental expansion pattern only yields a (long-run) performance advantage over the integrated expansion pattern, if a) design teams search mainly locally (some distance from the current

[11] In fact, the excessive search dynamics of the "always parochial" pattern prevents the teams from reaching a sticking point in most cases. Instead, the teams typically enter an oscillating state, in which a change to the first element affects the value of the second element, thus in the next period triggering a change to the second element, and so on. To account for this behavior, we measure the average performance of these teams in period 400, i.e., at a time when the average performance does not improve anymore.

alternatives is allowed); b) alternatives are initially evaluated parochially but over time parochialism is reduced; and c) interdependencies among design elements are present. If any of these three conditions fails to hold, the {1+…+1} pattern loses its performance advantage over {8}. At the same time, the core result is robust to (modest) changes in the search radius, to varying degrees of initial knowledge, and to different sizes and complexities of the design problem.

4.3 Effects of the underlying interaction pattern

In the above experiments, we assumed a fully interdependent design problem or, when varying the degree of interdependence in section 4.2.5, a random distribution of interdependencies between the variables that make up the problem. In many real-world settings, however, interdependencies are patterned rather than random, which has been shown to have important implications for the effectiveness of search (Rivkin and Siggelkow 2007). To probe into how the underlying interaction pattern may affect the advantage of chunky over integrated search, we model two stylized yet fundamental patterns: In the first pattern, some design elements are highly influential, affecting many other elements, whereas the remaining elements do not influence any other elements. Should a team consider the influential elements first, adding the less influential elements later, or vice versa? The second pattern assumes that some design elements are highly affected by many other elements, whereas the remaining elements are independent. Again, should a team consider the highly dependent elements first, adding the independent elements later, or vice versa? We represent both patterns, and an example of a random pattern, in Figure 3 in the form of interaction matrices. An "x" in row *i*, column *j* of a matrix denotes that design element *j* affects element *i*. For instance, in the "highly influential" matrix, we assume that the first three elements affect all eight elements (the first three columns are filled with x's). To allow for comparisons across interaction patterns, we keep the total number of interactions (i.e., the number of x's in the matrices) constant. In all cases, we have a total of 24 off-diagonal interactions, which corresponds to $K = 3$. (Obviously, structured patterns only arise for $K < N-1$, because otherwise the entire matrix is filled.)

< Insert Figure 3 about here >

As Figure 3 shows, considering the highly influential design elements first, and those that do not affect any other elements later, increases the benefits of the incremental pattern. The opposite approach that considers the highly influential elements at the end of the expansion process, in contrast, creates a performance penalty. The former approach initially exploits our main mechanism – gradually decreasing parochialism – to deal with the highly influential elements. Once those elements have been configured in a way that has avoided low sticking points, adding (and potentially changing) the remaining (uninfluential) elements will have no effect on them. In contrast, if the highly influential elements are added in the end, a change to any of these elements will completely alter the contributions of all the other elements.

In a similar vein, we find significant differences caused by the order in which elements are added for the case of highly dependent elements. If some elements are affected by many other elements, adding these elements first, leads to a performance that is much lower than that created by an integrated search. (Given that this interaction pattern results in a much higher number of local peaks (Rivkin and Siggelkow 2007), the performance of the {8} pattern suffers significantly, too.) Adding the highly dependent elements in the end, in contrast, dramatically increases the value of the incremental expansion pattern. The latter approach initially exploits the value of gradually decreasing parochialism; once the elements that are not affected by others have been configured, the highly dependent elements can be added and configured, knowing how the elements look like that affect them. In contrast, if the highly dependent elements are added at the outset, adding (and potentially changing) any of the remaining elements will subsequently completely alter the contributions of the highly dependent elements, thereby undermining the early search efforts.

In sum, we find that if there a few highly influential elements, a team should start its expansion process with them; if there are a few elements that are affected by many others, the team should leave them to the end of the expansion process. Doing so (and avoiding the opposite approaches) will significantly increase the performance advantages of the incremental expansion pattern over integrated search.

4.4 Effects of time constraints and different time allocation patterns

So far, we have ignored the fact that teams may have only a short or limited time to come up with a good design, instead giving them enough time to exhaust their potential for further improvement at each expansion stage. We now relax this assumption to explore the temporal properties of expanding a search domain. First, we let our teams search as before – assessing one new local alternative in each period and expanding the search domain once a sticking point is reached – but limit the overall time that is available. Hence, subject to how much time is available and depending on the chosen expansion pattern, it is possible that a team has not fully expanded and has not reached a sticking point for the overall system once time is up. In Figure 4, we show the expansion pattern that has the highest performance given any overall time that is available. (In addition, we show the results for the {4+1+…+1} pattern for the entire range for comparison purposes.)

< Insert Figure 4 about here >

As Figure 4 shows, when a team has only little time, it is preferable to pursue the fully integrated search {8}. Incremental expansion patterns, in contrast, become valuable when the team has more time available. In particular, once more time is available, patterns that start with large initial search domains become increasingly attractive. The more time is available, the smaller the optimal initial search domain. For instance, once 18 periods are available, teams following a {7+1} expansion pattern outperform teams with a {8} pattern. Likewise, once 26 periods are available, {6+1+1} yields higher performance than {7+1}.

Limitations of the overall time that is available for search thus highlight the costs of incrementalism by revealing a tradeoff between the speed of improvement and the final possible performance of the different expansion patterns: When having only little time, it is not advisable to focus on the (long-run) superiority of the incremental patterns. For short time horizons, the fact that the parochial search can lead to changes that actually decrease overall system performance is having an adverse effect. If a team does not have enough time to subsequently find alternatives that exploit the initial broader search, an integrated search process, which avoids performance-deteriorating moves at all times, is preferable.

In the simulation above, teams expanded their search domain once they could not find a performance-increasing local alternative at the current expansion stage, which implied the possibility that time was up before a team could fully expand. However, teams might be aware of their time constraints, and will try to avoid the risk of running out of time prematurely. In Figure 5, we study the basic impact of three stylized ways of how teams might allocate the available time to different expansion stages. In doing so, we focus on the {4+1+…+1} pattern, which requires the least time to yield the same performance as the fully incremental pattern {1+…+1}, as shown in Figure 4 and Table 1. The first time-allocation pattern, dubbed front-loading, is identical to the modeled behavior that led to Figure 4: Teams start searching at the first stage and expand once they cannot find any performance-improving local alternative anymore. If time is left, it gets pushed forward to the next expansion stage. In this pattern, the risk arises that teams run out of time before they have considered the entire system. Teams that follow the second time-allocation pattern, fixed time, distribute the available time equally to each expansion stage before starting to search. Once the time at a particular stage is up, the teams expand the search domain. If they have reached a sticking point before that, they likewise expand, but shift the remaining time to the subsequent stage. With this time allocation, teams might have to expand their search domain before they have found a sticking point (and they may not have enough time to find a sticking point after their last expansion). Finally, the third pattern, back-loading, causes teams to spend most of the available time at the final expansion stage, i.e., when the search domain has been fully expanded. In this time-allocation pattern, we assume that teams allocate two periods to each expansion stage and the remaining time to the final stage.[12]

< Insert Figure 5 about here >

The results reported in Figure 5 convey three main findings. First, back-loading can make an expansion pattern (relatively) more attractive when little time is available. By spending most of the time at the final expansion stage, this approach makes sure that teams have enough time to configure the final

[12] Two periods at the stages prior to the last one are necessary due to our assumption that one period is spent to learn about the value of the expanded search domain, while the first alternative is evaluated in the second period.

system. Second, we find that the fixed time allocation always outperforms the front-loading pattern (or yields the same performance). In particular, when relatively little time is available, the fixed pattern, in contrast to the front-loading pattern, guarantees at least a few periods of search when the search domain is large. Third, once the available time is long enough, both the fixed time pattern and the front-loading pattern perform identically, as both leave teams with enough time to search exhaustively at each stage before expanding to the next one.

In sum, in the presence of time constraints, teams can employ two levers to structure their search process: the expansion pattern and the time allocation pattern. In Table 3, we report the optimal combination of expansion pattern and time allocation pattern for different total periods that are available for search. Consistent with our earlier results, we find that the more time is available, the larger the optimal number of expansion steps (always starting with a large initial search domain). In general, a fixed time allocation is advisable, especially if the initial search domain is large or if expansion occurs in many steps. In these cases, it is important that enough time is saved for the later stages in which the system as a whole (or almost as a whole) is being considered. Back-loading, in contrast, is never optimal. (If back-loading for a given expansion pattern becomes preferred over the other time allocation patterns, as seen in Figure 4 when few periods of time are available, a change to a more integrated expansion pattern is even more beneficial.)

< Insert Table 3 about here >

4.5 Limits and extensions

Despite the broad set of analyses reported above, our model has limitations that offer potential for future research. For instance, we assumed that teams can assess the value of new alternatives in an "offline" manner, i.e., by means of simulations or prototypes, and with full certainty. In reality, however, learning often requires multiple trials, it may leave some sort of legacy, or yield only delayed performance feedback (Levitt and March 1988; Gavetti and Levinthal 2000; Denrell et al. 2004). Introducing such aspects of "online" and imperfect search would be a fruitful extension of the model.

Also, we treated each design decision as a binary choice. By extending the range of options for each dimension, a different interpretation of local search could arise, with search being constrained not only across design dimensions but perhaps even within design dimensions. Finally, even though we called the agents in our model "teams," they were single decision makers; i.e., we abstracted from many factors that make search "organizational" such as the presence of a hierarchy (Mihm et al. 2010) or multiple searchers that need to coordinate (Lounamaa and March 1987).

5 Discussion and conclusion

The challenge raised by complex systems – how to deal with numerous interdependent elements that need to fit together to create high overall performance – is found in a broad range of problems that organizations face. This task is inherently perilous: Because complex systems create vast and multi-peaked spaces of potential alternatives, the risk of ending up with a set of choices that create low performance, i.e., a low local peak, is significant. Under these circumstances, how search is structured has important implications because it affects how problem solvers will traverse the search space. While extant work has largely focused on different cognitive, organizational, or environmental determinants of organizational search, how the process of search itself should be structured has seen little attention.

In this paper, we have studied a basic related question, namely: how much complexity should be taken into consideration at any given time? Should a problem solver pursue an integrated search and be concerned with the whole system right from the start? Or should a problem solver expand the search domain more incrementally, and if yes, how chunky should these steps be? Our results speak to four major issues: the performance of these two search processes in general, the performance implications of different chunking patterns, the role of the underlying interaction structure, and the effects of time constraints on the various expansion patterns. We can summarize our main findings as follows:

1. *Benefits from chunky search.* For complex systems comprised of many interdependent elements, expansion of a search domain in several chunks can outperform integrated search. Realizing the value of chunky search, however, requires gradually decreasing parochialism: Problem solvers need not only focus their *search* initially, but also their *evaluations*. Moreover, over time they need to widen their focus until overall system performance is eventually considered.

2. *Choosing a chunking pattern.* It is not necessary to follow a completely incremental expansion pattern. Final system performance is often not materially affected if a problem solver starts with large chunks and then makes small additions to the search domain subsequently. The converse, however, is not true: starting with small chunks and adding larger chunks later, does lead to significantly lower performance.

3. *Considering the underlying interaction structure*. If a complex system consists of elements that either affect, or are affected by, many other elements, a problem solver should add the highly influential elements first and the highly dependent elements later in the expansion process. Adding the highly dependent elements early and the highly influential elements late, in contrast, creates a significant performance penalty.

4. *Minding the available time.* Chunky search processes take more time than integrated search. As a consequence, the shorter the available time, the more beneficial it becomes to search in an integrated manner and with a time allocation pattern that allows for search at all expansion stages, whereas the more time is available, the more incremental the expansion process should become.[13]

Our findings suggest a number of avenues for future empirical research on how search processes are structured. For instance, it would be very worthwhile to investigate, in a qualitative or experimental manner, situations in which problem solvers are searching for solutions in complex systems. How do they expand their search domains, if at all, and how do they decide how to divide a large system into a sequence of chunks? Looking at the performance implications of the search processes, can we reveal the differences between the different chunking patterns as suggested by our model? Do problem solvers consider the interaction structure underlying the system? And how do problem solvers factor in time constraints? Specifically, under which conditions are problem solvers likely to end up with a mismatch, either choosing a search pattern that requires more time to pay off than the time constraints allow, or not choosing a pattern that given the available time would lead to even higher performance?

In addition to suggesting empirical work to validate our abstract findings, our study may also inform more applied context-specific work on how problem-solving search processes are structured. For example, one might study if a component supplier that wants to transform itself into a system supplier would be better off, if it temporarily focused on learning about a less complex subsystem and, over time,

[13] One should note again that these results depend on the presence of a complex system with many interdependencies. For instance, for modular systems, parallel search would have obvious speed advantages over integrated search. For each non-decomposable module, however, our results would apply again.

expanded its search domain in several steps. Similarly, consider an entrepreneur who needs to configure numerous interdependent activities along the value chain. One might hypothesize the new venture to fare better if the entrepreneur started off by focusing on a broad but incomplete set of activities (e.g., those related to operations, marketing, sales, and logistics), and subsequently made further incremental additions to the search domain (e.g., by adding also activities related to procurement or human resource management), rather than configuring only a few activities (e.g., all activities related to operations) and adding larger numbers of activities later on. Considering the context of complex product development, are successful design teams more likely to initially focus on those elements of the product that affect many other components, adding and adapting the remaining components that do not affect many other components, once these components have been designed? Likewise, are successful teams more likely to adopt any highly dependent product components late, rather than early, in the design process, as our findings might suggest? Finally, consider a manager of a publicly traded firm who has to re-configure a firm's system of interdependent activities after a large, fit-destroying environmental shock. If the capital markets do not grant the manager much time to re-establish firm performance, one might expect this manager to trade in final possible system performance for speed of improvement by preferring an integrated search or following an expansion pattern with few steps. In contrast, a manager of a firm with more "patient" capital might want to employ a more incremental search structure, allowing higher final performance.

While providing a starting point for future theoretical and empirical inquiries into how search processes are structured, our paper makes a number of further contributions: First, a significant literature has dealt with the question of how to avoid problem-solving oscillations (e.g., Terwiesch and Loch 1999; Mihm et al. 2003) in which designers evaluate and reject or accept the same alternatives several times during the course of a design project. Without doubt, an integrated search can help avoid such oscillations. In contrast, if a team follows an incremental expansion pattern, oscillations can occur as designers may return to a previously discarded alternative once they expand their search domain. However, since the larger search domain involves considerations of an additional (set of) variable(s) and interdependencies, a new and

different light might be shed on the previous design. Thus, our findings suggest that problem-solving oscillations *between different* expansion stages could at times actually be beneficial, whereas it is inefficient if they occur at the *same* expansion stage.

Second, prior research on the role of managerial cognition in organizational search has pointed to the value of lower-dimensional "maps" of an organization's environment as well as to analogies derived from similar contexts in seeding and guiding a firm's search of the entire landscape (Gavetti and Levinthal 2000; Gavetti et al. 2005). Adding to this literature, our study suggests a different cognitive mechanism which, by reducing the complexity of the real world, may convey a search benefit and improve managerial decision making. Rather than trying to construct a lower-dimensional representation of the overall system, managers may select a subset of all relevant elements (e.g., those, whose interdependencies and ramifications they can assess more accurately), while needing only some rough assumptions about the remaining elements, and try to improve how these elements are configured before expanding their search domain and taking all elements into account.

Third, while our analysis was motivated by "deliberate" search processes – problem solvers such as product design teams that may choose how to expand their search domain – our findings also relate to more evolutionary search processes. For instance, a large body of research on technological and industrial change has studied the evolution of dominant designs, pointing out that the rate of major innovations in an industry tends to be high initially, only to decline sharply and give rise to more incremental innovations once a dominant design has emerged (Abernathy and Utterback 1978; Anderson and Tushman 1990; Suarez and Utterback 1995). Based on our model, we might offer a search-theoretical framing for this observation: As firms try to come up with solutions to a novel problem, they first experiment with a larger chunk of the problem, considering various, but not all, potential dimensions and thus generating rather different, competing solutions. Once a firm has found a solution that is particularly promising (or one that for other reasons becomes the industry standard), this design is picked up by other firms. At the same time, the search domain in the industry is expanded, as firms will now consider also the dimensions that were so far neglected in order to differentiate themselves and further tweak the

performance of the solution. Moreover, considering the role of the underlying interaction pattern (as discussed in section 4.3), our model also corresponds with the notion that dominant designs are characterized by the choice of core components, followed by a transition to experimenting with peripheral components that give rise to a broad variety of (slightly) different designs (Murmann and Frenken 2006).

Finally, our findings relate to the management of complex systems by boundedly rational decision makers, and to the question of how such systems can be productively embraced. Two approaches have featured prominently in this realm: the decomposition of larger systems into nearly independent subsystems, or, if decomposition is not an option, the management of a system in an integrated manner. In this paper, we add a third approach: gradually decreasing parochialism, i.e., the focus on only a few elements of a system and the deliberate suppression of the remaining system context and the complexities it entails, and the stepwise expansion of this focus. Growing a search domain in a chunky manner appears to be a powerful strategy to address the challenges of dealing with complex systems.

References

Abernathy, W. J., J. M. Utterback. 1978. Patterns of industrial innovation. *Technology Review* 80(7) 40-47.

Anderson, P., M. L. Tushman. 1990. Technological discontinuities and dominant designs: A cyclical model of technological change. *Administrative Science Quarterly* 35(4) 604-633

Arthur, W. B. 2009. *The Nature of Technology: What it Is and How it Evolves*, Free Press, New York.

Baldwin, C. Y., K. B. Clark. 2000. *Design Rules, Volume 1: The Power of Modularity*, MIT Press, Cambridge, MA.

Burton, R. M., B. Obel. 2004. *Strategic Organizational Diagnosis and Design: The Dynamics of Fit*, Kluwer, Boston.

Cyert, R. M., J. G. March. 1963. *A Behavioral Theory of the Firm*, Prentice Hall, Englewood Cliffs, NJ.

Denrell, J., C. Fang, D. A. Levinthal. 2004. From T-mazes to labyrinths: Learning from model-based feedback. *Management Science* 50(10) 1366-1378.

Ethiraj, S. K., D. A. Levinthal. 2004. Modularity and innovation in complex systems. *Management Science* 50(2) 159-173.

Fleming, L. 2001. Recombinant uncertainty in technological search. *Management Science* 47(1) 117-132.

Frensch, P. A., J. Funke. 1995. *Complex Problem Solving: The European Perspective*, Lawrence Erlbaum Associates, Hillsdale, NJ.

Gavetti, G., D. A. Levinthal. 2000. Looking forward and looking backward: Cognitive and experiential search. *Administrative Science Quarterly* 45(1) 113-137.

Gavetti, G., D. A. Levinthal, J. W. Rivkin. 2005. Strategy making in novel and complex worlds: The power of analogy. *Strategic Management Journal* 26(8) 691-712.

Gigerenzer, G., P. M. Todd, the ABC Research Group. 1999. *Simple Heuristics That Make Us Smart*, Oxford University Press, New York.

Iansiti, M., A. MacCormack. 1996. Team New Zealand (C), Harvard Business School Case 697-041.

Katila, R., G. Ahuja. 2002. Something old, something new: A longitudinal study of search behavior and new product introduction. *Academy of Management Journal* 45(6) 1183-1194.

Kauffman, S. A. 1993. *Origins of Order: Self-Organization and Selection in Evolution*, Oxford University Press, New York.

Kauffman, S. A. 1995. *At Home in the Universe: The Search for the Laws of Self-Organization and Complexity*, Oxford University Press, New York.

Kauffman, S. A., J. Lobo, W. G. Macready. 2000. Optimal search on a technology landscape. *Journal of Economic Behavior and Organization* 43(2) 141-166.

Knudsen, T., D. A. Levinthal. 2007. Two faces of search: Alternative generation and alternative evaluation. *Organization Science* 18(1) 39-54.

Lenox, M. J., S. F. Rockart, A. Y. Lewin. 2006. Interdependency, competition, and the distribution of firm and industry profits. *Management Science* 52(5) 757-772.

Levinthal, D. A. 1997. Adaptation on rugged landscapes. *Management Science* 43(7) 934-950.

Levitt, B., J. G. March. 1988. Organizational learning. *Annual Review of Sociology* 14 319-340.

Lounamaa, P. H., J. G. March. 1987. Adaptive coordination of a learning team. *Management Science* 33(1) 107-123.

MacCormack, A., J. Rusnak, C. Y. Baldwin. 2006. Exploring the structure of complex software designs: An empirical study of open source and proprietary code. *Management Science* 52(7) 1015-1030.

March, J. G., H. A. Simon. 1958. *Organizations*, Wiley, New York.

Mihm, J., C. H. Loch, A. Huchzermeier. 2003. Problem-solving oscillations in complex engineering projects. *Management Science* 49(6) 733-750.

Mihm, J., C. H. Loch, D. Wilkinson, B. A. Huberman. 2010. Hierarchical structure and search in complex organizations. *Management Science* 56(5) 831-848.

Miller, G. A. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review* 63(2) 81-97.

Murmann, J. P., K. Frenken. 2006. Toward a systematic framework for research on dominant designs, technological innovations, and industrial change. *Research Policy* 35(7) 925-952.

Nelson, R. R., S. G. Winter. 1982. *An Evolutionary Theory of Economic Change*, Harvard University Press, Cambridge, MA.

Newell, A., H. A. Simon. 1972. *Human Problem Solving*, Prentice Hall, Englewood Cliffs, NJ.

Nickerson, J. A., T. R. Zenger. 2004. A knowledge-based theory of the firm: The problem-solving perspective. *Organization Science* 15(6) 617-632.

Rivkin, J. W. 2000. Imitation of complex strategies. *Management Science* 46(6) 824-844.

Rivkin, J. W., N. Siggelkow. 2003. Balancing search and stability: Interdependencies among elements of organizational design. *Management Science* 49(3) 290-311.

Rivkin, J. W., N. Siggelkow. 2007. Patterned interaction in complex systems: Implications for exploration. *Management Science* 53(7) 1068-1085.

Rosenkopf, L., P. Almeida. 2003. Overcoming local search through alliances and mobility. *Management Science* 49(6) 751-766.

Siggelkow, N., D. A. Levinthal. 2003. Temporarily divide to conquer: Centralized, decentralized, and reintegrated organizational approaches to exploration and adaptation. *Organization Science* 14(6) 650-669.

Siggelkow, N., J. W. Rivkin. 2005. Speed and search: Designing organizations for turbulence and complexity. *Organization Science* 16(2) 101-122.

Simon, H. A. 1955. A behavioral model of rational choice. *Quarterly Journal of Economics* 69(1) 99-118.

Simon, H. A. 1956. Rational choice and the structure of the environment. *Psychological Review* 63(2) 129-138.

Simon, H. A. 1962. The architecture of complexity. *Proceedings of the American Philosophical Society* 106(6) 467-482.

Simon, H. A. 1996. *The Sciences of the Artificial*, MIT Press, Cambridge, MA.

Stuart, T. E., J. M. Podolny. 1996. Local search and the evolution of technological capabilities. *Strategic Management Journal* 17(7) 21-38.

Suarez, F. F., J. M. Utterback. 1995. Dominant designs and the survival of firms. *Strategic Management Journal* 16(6) 415-430.

Terwiesch, C., C. H. Loch. 1999. Managing the process of engineering change orders: The case of the climate control system in automobile development. *Journal of Product Innovation Management* 16(2) 160-172.

Thomke, S., E. von Hippel, R. Franke. 1998. Modes of experimentation: An innovation process - and competitive - variable. *Research Policy* 27(3) 315.

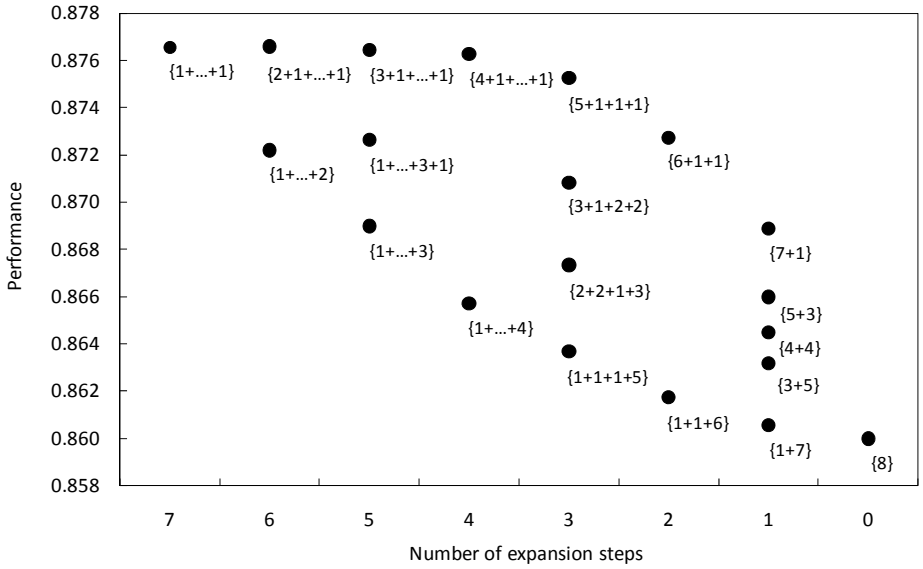
Thompson, J. D. 1967. *Organizations in Action*, McGraw-Hill, New York.

Ulrich, K. T., S. D. Eppinger. 2007. *Product Design and Development*, McGraw-Hill, Boston, MA.

Winter, S. G., G. Cattani, A. Dorsch. 2007. The value of moderate obsession: Insight from a new model of organizational search. *Organization Science* 18(3) 403-419.

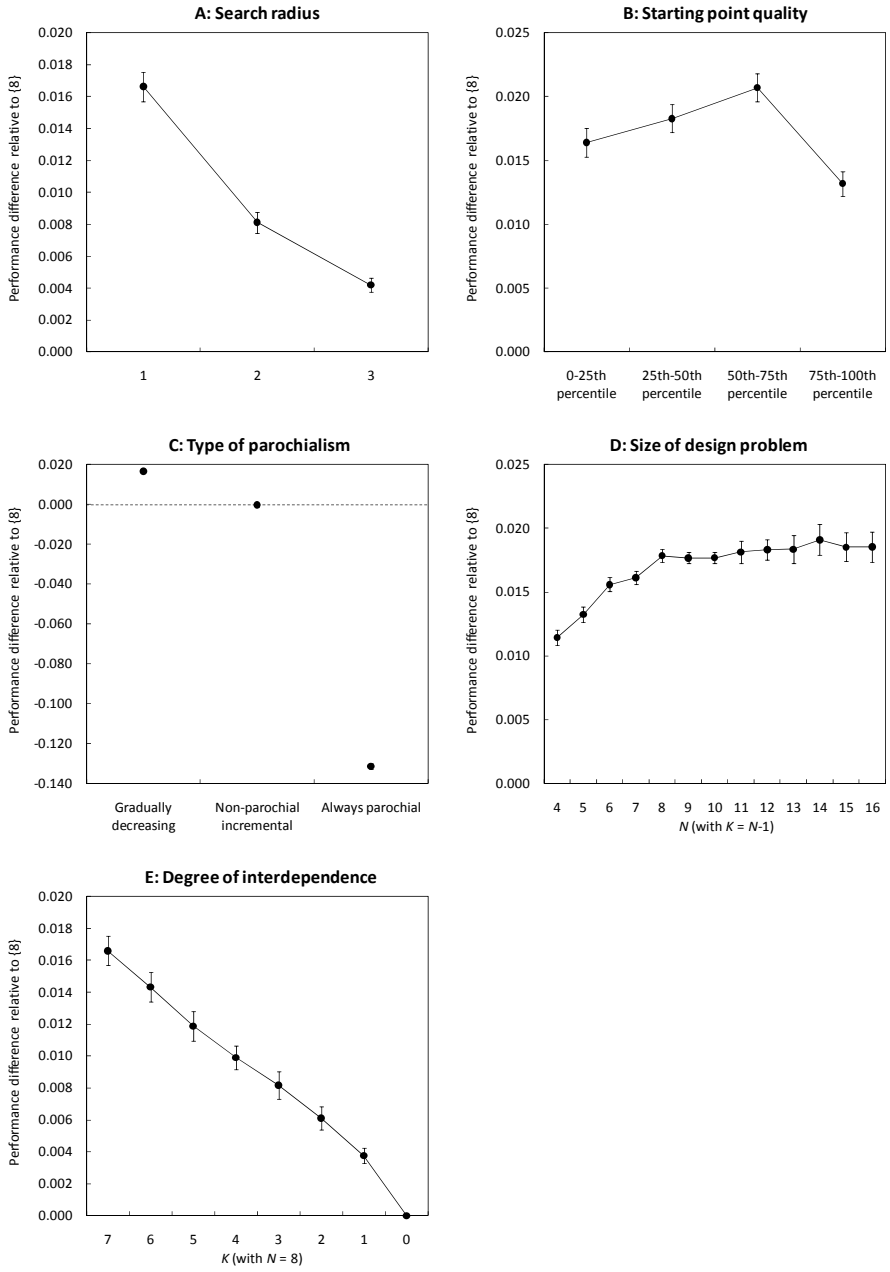
Wright, S. 1932. The roles of mutation, inbreeding, crossbreeding, and selection in evolution. *Proceedings of the Sixth International Congress on Genetics* 355-366.

Figure 1: Performance implications of different expansion patterns



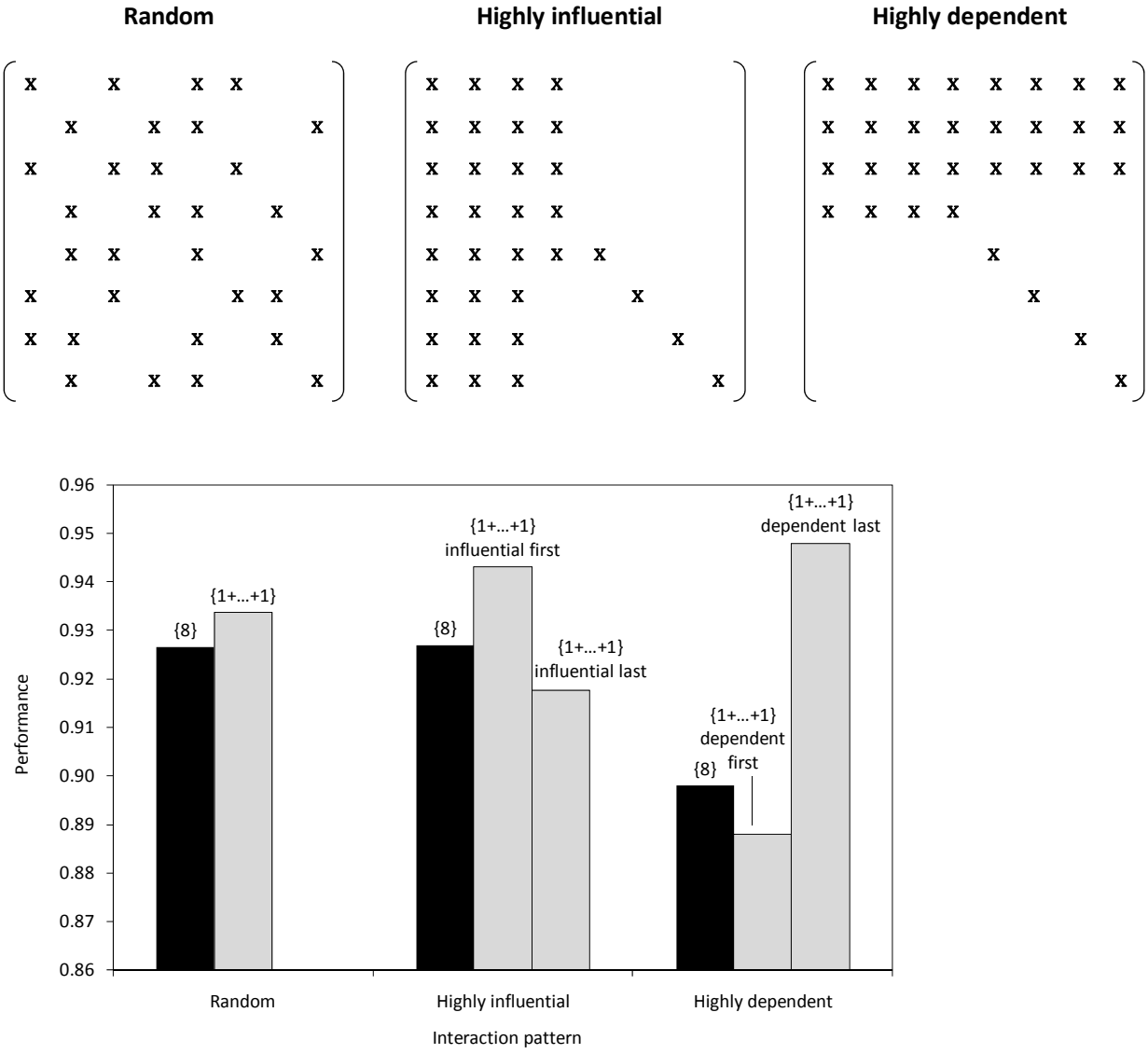
This figure reports the average final system performance over 10,000 landscapes with $N = 8$, $K = 7$. Performance values are normalized with respect to the global peak in each landscape. Project teams differ in their expansion patterns. An ellipsis in the label of a pattern denotes a series of steps with increment 1. At each expansion stage, teams search until no performance-improving local alternative can be identified anymore.

Figure 2: Boundaries of the core result



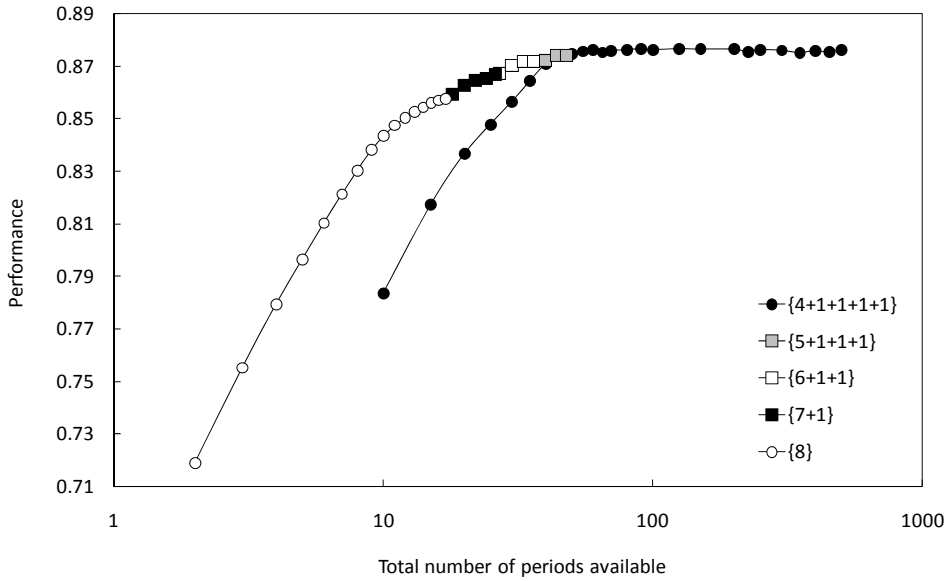
All panels report performance differences relative to the {8} pattern. Performance values are averages over 10,000 landscapes with $N = 8$ and are normalized with respect to the global peak in each landscape. In each panel (except B and D), the very left node represents the {1+…+1} pattern under the conditions of Figure 1, i.e., local search, gradually decreasing parochialism, and a complex problem ($K = 7$). (For panel B, the effect for a randomly chosen starting point is 0.017. In panel D, the main model is in the middle of the figure, N = 8.) The error bars in each panel represent +/- one standard error. In panel A, teams differ in their search radii. In panel B, teams differ in the quality of their starting point. In panel C, teams following the "non-parochial incremental" pattern gradually expand their search domain, but always take overall system performance into account when evaluating alternatives. Teams following the "always parochial" pattern look at one design element (and its performance contribution) at a time on a continuous basis. (The error bars are not visible in this panel due to the scale of the y-axis.) In panel D, the problem size (N) is varied, while K is set to N-1. In panel E, the degree of interdependence (K) is varied, holding N fixed at 8.

Figure 3: Role of the underlying interaction pattern



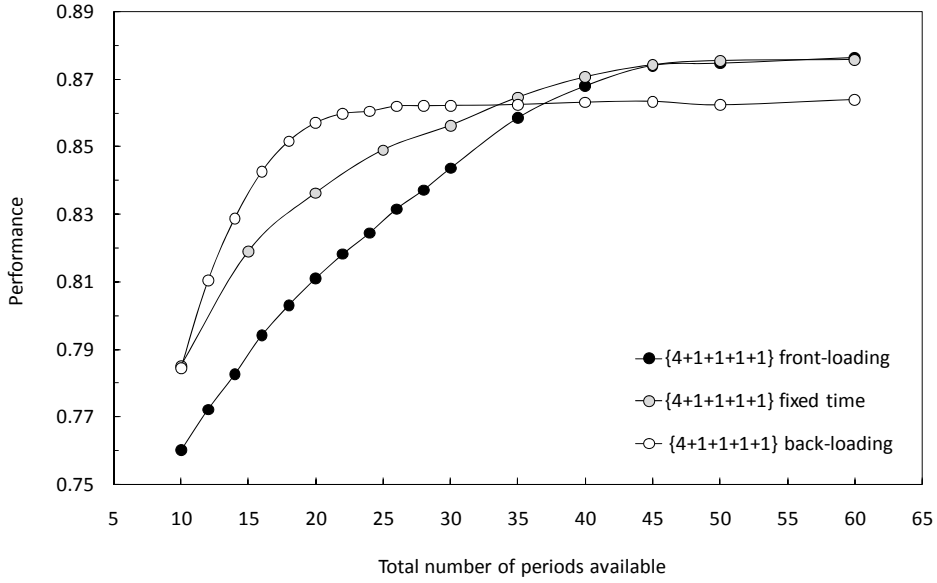
The upper half of the figure illustrates three stylized interactions patterns. In each matrix, an entry in row *i*, column *j* denotes that design element *i* affects element *j*. The patterns are: (a) a "random" pattern, in which each element interacts with three other elements (given $K = 3$) that are randomly assigned for each landscape; (b) a pattern, in which a number of "highly influential" elements exist (elements 1-4) that affect many other elements, whereas the remaining elements 5-8 do not affect any other elements; and (c) a pattern in which a number of "highly dependent" elements exist (elements 1-4) that are affected by many other elements, whereas the remaining elements 5-8 are independent. The lower half of the figure reports the performance of the {8} search pattern (black bars) and the {1+…+1} pattern (grey bars) subject to the different interaction patterns of the underlying problems. Performance values are averages over 10,000 landscapes with $N = 8$, $K = 3$, and are normalized with respect to the global peak in each landscape. The two bars on the very left represent the conditions of Figure 1, i.e., a random interaction pattern, but with K = 3 (see also Figure 2, Panel E, the fifth node from the left). For the highly influential interaction pattern, we let the {1+…+1} add the highly influential elements either early ("influential first") or late ("influential last") in the expansion process. For the highly dependent interaction pattern, we let the {1+…+1} add the highly dependent elements either early ("dependent first") or late ("dependent last") in the expansion process.

Figure 4: An efficient frontier of expansion patterns given overall time constraints



This figure reports the average final system performance over 10,000 landscapes with N = 8, K = 7, given that the time for search is limited. Performance values are normalized with respect to the global peak in each landscape. In addition to the {4+1+…+1} pattern, only the highest-performing expansion pattern for a particular amount of time is depicted. At each expansion stage, project teams search until no performance-improving local alternative can be identified anymore. Once the available time is exhausted, the search ends. The x-axis is scaled logarithmically.

Figure 5: Performance implications of different patterns of allocating the overall search time



This figure reports the average final system performance over 10,000 landscapes with N = 8, K = 7, given that the time for search is limited. Performance values are normalized with respect to the global peak in each landscape. All project teams follow the same expansion pattern, {4+1+1+1+1}, but differ in their patterns of allocating time across the different expansion stages.

Table 1: Performance differences between different expansion patterns

Expansion pattern	{1+…+1}	{1+1+6}	{1+…+5}	{1+1+…+4}	{4+1+…+1}	{5+1+1+1}	{6+1+1}	{4+4}	{8}
{1+…+1}	-	***	***	***	n.s.	*	***	***	***
{1+1+6}		-	*	***	***	***	***	***	*
{1+…+5}			-	*	***	***	***	n.s.	***
{1+…+4}				-	***	***	***	n.s.	***
{4+1+…+1}					-	*	***	***	***
{5+1+1+1}						-	**	***	***
{6+1+1}							-	***	***
{4+4}								-	***

This table reports the statistical significance of the performance differences among a selection of the expansion patterns depicted in Figure 1. Significance levels are labeled as *** ($p < 0.001$), ** ($p < 0.01$), * ($p < 0.05$).

Table 2: Performance and exploration metrics for different expansion patterns

Expansion pattern	{1+…+1}	{1+1+6}	{1+1+1+5}	{1+1+…+4}	{4+1+…+1}	{5+1+1+1}	{6+1+1}	{4+4}	{8}
Percentage of teams at global peak	6.96%	5.35%	5.74%	5.93%	6.84%	6.65%	6.36%	5.92%	5.35%
Performance if not at global peak	0.868	0.854	0.855	0.858	0.867	0.866	0.864	0.856	0.852
Number of evaluations made	50.7	18.2	22.3	27.7	41.7	36.4	30.0	18.7	13.4
Number of evaluated contributions	125.2	105.4	106.7	109.2	125.1	124.6	122.6	109.0	107.2

This table reports performance and exploration metrics for the expansion patterns listed in Table 1.

Table 3: Optimal combinations of expansion and time allocation patterns given time constraints

Total periods available	Highest performance	Reached by expansion pattern(s)	Time allocation(s)	Interpretation
10	0.845	{8}	n.a.	Integrated search yields the fastest performance improvements.
20	0.862	{7+1} {6+2} {5+3} {4+4}	Fixed time Fixed time Fixed time/front-loading Fixed time/front-loading	Expand with one step and put the large chunk first; make sure that enough time is left at the final stage. This is particularly relevant the larger the initial chunk.
30	0.869	{6+1+1} {6+2} {5+2+1}	Fixed time Fixed time/front-loading Fixed time/front-loading	Expand with one or two steps and put the large chunk first; make sure that enough time is left at the final stage. This is particularly relevant the larger the initial chunk.
40	0.874	{6+1+1} {5+1+1+1} {4+2+1+1} {3+3+1+1} {3+2+2+1}	Fixed time/front-loading Fixed time Fixed time/front-loading Fixed time/front-loading Fixed time/front-loading	Expand with two or three steps and put the large chunk first; make sure that enough time is left at the final stage. This is particularly relevant the larger the initial chunk.
50 or more	0.877	{4+1+1+1+1}	Fixed time/front-loading	Expand with four steps and put the large chunk first.

This table reports the average system performance over 10,000 landscapes with $N = 8$, $K = 7$ for teams that follow different expansion patterns and different patterns for allocating the available time. Performance values are normalized with respect to the global peak in each landscape. Only the highest-performing (set of) expansion and time allocation pattern(s) for a particular amount of total available time is listed.