How to Solve Problems with Crowds: A Computer-Based Simulation Model

Article *in* Creativity and Innovation Management · June 2014

DOI: 10.1111/caim.12059

2 authors:

Oana Vuculescu
Aarhus Business School
10 PUBLICATIONS **60** CITATIONS
SEE PROFILE

Carsten Bergenholtz
Aarhus University
27 PUBLICATIONS **219** CITATIONS
SEE PROFILE

How to solve problems with crowds: A computer-based simulation model

Oana Vuculescu and Carsten Bergenholtz

Innovation Management Group, Department of Business Administration

Business and Social Sciences, Aarhus University, Denmark

Abstract:

Organizations rely on crowds for a variety of reasons, e.g. in order to evaluate (Amazon), create content (Threadless) but also to solve given problems (InnoCentive and OpenIDEO). Several studies have examined how to organize problem solving activities. However, most papers have examined the crowdsourcing process using a partial perspective and a wide-ranging outlook has been missing. This study uses a computer-based simulation model and anecdotal case studies of InnoCentive and OpenIDEO, in order to simulate the dynamics of collective problem solving behaviour. Results show how the number of users, number of iterations and different selection mechanisms impact the ability to find an optimal solution to the given problem.

1. Introduction

Organizations increasingly rely on crowds to accomplish a series of tasks ranging from evaluation (Amazon, Netflix), content creation (Threadless) to problem solving (OpenIDEO and InnoCentive). Following the problem solving perspective (Nickerson and Zenger 2004; Felin and Zenger 2013) we focus on collective problem solving (henceforth CPS). There is growing empirical evidence as to the efficiency of such setups in innovation management studies (Poetz and Schreier, 2012) and InnoCentive (Bonabeau, 2009), Foldit (Khatib et al., 2011) and Dell's IdeaStorm (Bayus, 2013) are only a few examples. The agenda of the paper is to identify main CPS design elements and how different designs are likely to lead to different CPS performances. The body of literature concerning this topic is growing (Estelles-Arolas and Gonzalez-Ladron-de-Guevara, 2012) and preceding approaches highlight different angles such as the importance of diversity of perspectives (Jeppesen and Lakhani, 2010; Boudreau 2012), the importance of collaboration between solvers (Bullinger et al., 2010; Hutter et al., 2011) and problems pertaining to competing motivations (Terwiesch and Xu, 2008). However the difficulty with prior approaches is that they are primarily descriptive and often involve only a partial perspective on CPS (Adamczyk, Bullinger et al., 2012).

Previous work thus leaves unresolved issues concerning to what extent there should be any communication and networks between participants, the number of participants to involve and in particular how to govern and select the best solutions. While indeed empirically grounded studies argue that interaction between solvers is a vital element of successful CPS set-ups (Leimeister, Huber et al., 2009; Schweitzer, Buchinger et al., 2012) and experiments show that inefficient networks (corresponding to low communication) will decrease exploration (Mason and Watts, 2012), Lazer and Friedman (2007) using computer simulations find the counter-intuitive result that on the long run interaction between participants in CPS is always detrimental and only on the medium time frame is moderate communication desirable. Furthermore, the need for attracting a high (and therefore diverse) number of solvers is largely agreed upon in the literature (Chesbrough et al., 2006) but not always easy and, as we show in the following pages, a need that can be alleviated through design mechanisms. Finally, we address how ideas are to be selected in the different phases of the CPS process, contrasting the traditional linear approach (Cooper 2002) with an iterative perspective (Goldberg, 2002).

To investigate the above mentioned tensions in the literature, we explore the dynamics between the main features of CPS by simulating (Davis et al., 2007) CPS as a search optimization process. The chosen model is grounded both in literature and empirically and as such it can be inferred that it does capture essential aspects in human problem solving behavior. Furthermore we rely on two anecdotal case studies, chosen to represent two contrasting ways of designing CPS, InnoCentive and OpenIDEO, in order to illustrate and empirically ground the computer simulations. The repeated computer simulations allow then for inferences to be made on how to optimize the design of CPS - e.g. number of solvers, number of iterations and selection mechanisms, networks and communication between participants - to reach high quality solutions. There are two reputable approaches to modeling search processes: the NK model and the genetic algorithm (Mitchell, 2009). We use a model based on a specific type of genetic algorithm (SASEGASA) to simulate the relationship between the main features of CPS and their efficiency. As a number of organizational studies have applied the NK model (Fioretti, 2013), despite its drawbacks (Mckelvey et al., 2013) we also rely on it to validate the findings from our genetic algorithm model.

Main results show how in designing a CPS setup, a) the quality of the solutions and the speed of the CPS effort are affected by allowing parallel search processes, and b) having a low number of contributors can be mitigated by resorting to iterative problem solving phases as well as c) using advanced mechanisms to select solutions offered by the collective. Hereby we contribute to more effective management decisions regarding the design of CPS.

In the next section we introduce a brief literature review on crowdsourcing with a focus on relevant aspects for CPS. In section 3 the nature of the model is reviewed while section 4 presents the results from the simulation models, with illustrations from the case study findings. Section 5 discusses the theoretical implications of the simulation findings.

2. Conceptual background: Collective problem solving

2.1. Crowdsourcing

While innovation contests have a long history (Adamczy et al., 2012) the term crowdsourcing is very recent, being coined by Jeff Howe (2009). Crowdsourcing is meant to encompass a wide range of phenomena otherwise linked with concepts such as “collective intelligence”, “collaborative innovation” or “commons based peer production”. Estelles-Arolas and Gonzalez-Landon-de-Guevara (2012) aggregate a definition on crowdsourcing following an extensive overview of state of the art papers, centered around six characteristics: who forms the crowd, type of task, incentives for contributors, who is the initiator, what is the goal pursued by the initiator (e.g. marketing, problem solving) and characteristics of the process.

We take departure in the above mentioned crowdsourcing definition, but exclude two elements that are not relevant for the present perspective. First of all, we define collective problem solving as a process that involves large numbers of actors and large numbers of interactions that do not occur in a hierarchy: there is no distribution of tasks, no control over who contributes, what the contributions are and there is little if any control over the interaction between solvers. Consequently, CPS necessarily has a clearly stated goal, cf. to solve problems. Finally CPS is not intrinsically connected with an initiator, since the process can work on idea management platforms set-up by firms but also on independent communities such as forums or emailing groups. We thus in the following pages offer an overview structured around the four remaining features of the definition provided by Estelles-Arolas and Gonzales-Landon-de-Guevara (2012): the crowd, incentives for contributors, characteristics of the process and features of the task (i.e. problem).

2.2.1. The crowd: information diversity and motivation diversity

The majority of authors agree that a "large group of individuals" is important for a successful CPS (Benkler, 2002; Estelles-Arolas and Gonzalez-Ladron-de-Guevara, 2012). A large number of contributors imply that a more diverse pool of individuals has been involved and the main consequence of population size is thus information diversity. This finding is in line with the open innovation perspective (Chesbrough et al., 2006). Differently formulated, crowds follow Linus's famous law "*given enough eyeballs, all bugs are shallow*"(Raymond 1999, p8).

However, authors such as Fullerton and McAfee (1999) caution against the costs of having too many solvers, as too many solutions might imply underinvestment in solvers' effort, which Alexy et al.'s (2011) study of how organizations solicit ideas from externals illustrates. Terwiesch and Xu (2008) investigate this tension by changing the award structure and the overall incentives for the contributors. However, their approach does not address the problem of increased costs associated with large numbers of solvers, a cost which is mainly connected with the idea evaluation (Dahlander and Piezunka, 2013).

The second aspect pertains to diversity of motivation: the more contributors, the higher the diversity of motivations. Reviewed studies show that indeed there is no one single motivation that characterizes CPS solvers. As such, contributors are driven by their sense to facilitate their own work using what they create (Raymond, 1999; Hertel, Niedner et al., 2003), by the fact that they enjoy the work itself and they have a strong internal urge of "scratching an itch" (Raymond, 1999) or the need for enhanced reputation among peers (Raymond, 1999; Jeppesen and Frederiksen, 2006; Dellarocas, 2010). Therefore, Benkler (2002) argues that with a large enough pool of solvers, while there is no ideal incentive system it is also more likely that a platform will be able to attract enough highly motivated people.

2.2.2. The process

Estelles-Arolas and Gonzalez-Ladron-de-Guevara (2012) note great heterogeneity in the literature when it comes to defining crowdsourcing as a process (e.g. outsourcing (Howe, 2009) or the production model (Benkler, 2002)). Several papers focus on the processes within the crowd (interactions between solvers) in terms of cooperation or competition and studies suggest that both mechanisms are of value in CPS (Bullinger et al., 2010; Hutter et al., 2011; Afuah and Tucci, 2012; Schweitzer et al., 2012).

The question about process also refers to how the CPS task is to be governed. In the innovation management literature it has been discussed whether the governing mechanism should rely on a linear stage-gate approach (Cooper, 2005) or a non-linear, iterative approach (Levardy and Browning, 2009). In an attempt to integrate different discussions, Felin & Zenger (2013) argue that the choice of design mechanism should depend on the complexity of the innovation problem at hand. Problems of high complexity are suitable for communities, while problems of low complexity are suitable for innovation contests.

2.2.3. The problem: size and complexity

Innovation literature agrees upon the fact that a problem (i.e. task) should be reducible to smaller components, a characteristic which is known as “modularity” (Benkler, 2002; Baldwin and Hippel, 2011; Lakhani and Tushman, 2012). Baldwin and Von Hippel (2011) argue that it is the modular architecture of a problem that allows for what he calls “open collaborative innovation” to emerge and that this specific type of architecture is the basis for dividing the tasks among the contributors. Benkler (2002) expands on the need for problem modularization and advances an explanation: large tasks require a lot of resources, focus and most importantly motivation and thus will have a smaller chance of being undertaken by individual contributors.

The problem complexity is discussed by Rivera and Slawsby (2007) who argue for a reverse relationship between the complexity of the problem and the number of people that can contribute to the optimum solution, meaning that, in their view, extremely complex problems should not be solved by means of CPS. However, Hansen (1999) indirectly challenges this contention and suggests that weak ties are efficient when dealing with search in complex problem solving although inefficient when complex knowledge transfers need to occur.

Overall, this paper thus anchors a relatively new topic of research (i.e. crowdsourcing) in the wider literature of problem solving and resorts to a complexity science based research method, cf. computational simulations (as proposed by Afuah and Tucci, 2012) to investigate the defining characteristics of CPS set-ups. The aim is to computer simulate the relationship between system performance (i.e. reaching a high quality solution) and various possible configurations for CPS set-ups (i.e. the number of contributors, number of phases, parallel or non-parallel search, type of selection) and how that relationship is influenced by the type of problem that is to be solved.

3. Research methods

By running simulations of several model configurations, we aim to optimize configurations of the CPS main features on a twofold dependent variable: solution quality and the time required to reach it. Previous research as well as empirical grounding served as a starting point in calibrating the model which is then used in developing theories (Davis et al., 2007) about CPS. We calibrate two distinct classes of simulations – one for a complex problem and the other for a less complex problem. We thus construct a framework in which detailed assumptions can be made and tested about number and type of interactions between solvers, the size and type of problem.

Two anecdotal case studies serve as empirical grounding for the model and are used to illustrate general findings (Burns, 2000). Information on how InnoCentive and OpenIDEO design their CPS activities has been collected, from openly accessible archival data (their websites accessed May-September 2012). The cases were chosen to represent two particular ways of designing CPS set-ups as we oppose InnoCentive, the classical example, with OpenIDEO. Following recommendations proposed by Fioretti (2013) and Davis et al. (2007) we hereby confront the results from the simulation models with empirical findings and other previous studies, in order to validate the conclusions (Fioretti 2013).

3.1. Cases

Innocentive is a massive problem solving platform (initially spun off from Eli Lilly) with over 250.000 registered solvers more than 1400 challenges posted and an average success ratio of 50%. The process involves largely four steps: a company comes up with a problem, Innocentive dissects the problem into smaller challenges, solvers (scientists across the world) submit possible solutions, Innocentive analyses the proposals, chooses the winning solutions and integrates them back as the answer to the original problem. Solvers register on the Innocentive website, read the challenge and within a specific timeframe send back a solution. Interactions between solvers regarding a specific challenge are only possible if they formally decide to form a team and share the prize. Throughout the entire process solvers do not receive any feedback on their own solution, nor see each other's solutions.

One example of Innocentive Challenge: The Dual Use – Off Grid – Illumination Device Challenge. Solvers were asked to design a dual purpose, interior, self-contained, off-grid lighting device.

Unlike Innocentive, OpenIDEO is built to be an open CPS set-up. Each problem is divided into several themes by OpenIDEO and then published on the website, where it goes through four phases. In the first "Inspiration" phase, contributors are encouraged to publish any sort of relevant material. The actual construction of an answer to the problem takes place in the "Concepting" phase. Using inspirations as foundations for their designs, but also alternative sources, solvers

publish concepts that follow a pre-defined frame. At the end of the concept phase, contributors can applaud favored solutions and try to promote them into the final phases. "Refinement" is a phase in which a small number of chosen solutions are enhanced by their authors and the community. This shortlist then goes through the "Evaluation" phase which is again community endorsed, as users have a chance to "grade" solutions on a fixed-criteria basis.
One example of OpenIDEO challenge: "How can we improve sanitation and better manage human waste in low-income urban communities?"

[Insert figure 1 about here]

3.2. Modeling problem solving – the framework

Both models used in this paper rely on the "rugged landscape metaphor", as presented in the seminal work of Simon (1956) where he describes problem solving as a search through a landscape. According to Simon, when humans solve problems they operate within a problem space where they sample different positions trying to find the highest peak. Since the solver is bounded rational, a livelier image would be that this is a search through a rugged landscape, covered in thick fog – the solver doesn't know without checking which is the best solution (i.e. where is the highest peak). Problem solving is then the costly trial and error attempt of checking each position and evaluating it to see how suitable it is as a solution.
If individual problem solving is a search through a landscape, collective problem solving works as a search heuristic and from the relatively wide range of such computational models, the genetic algorithm works best at approximating the search behavior of the solvers (Goldberg, 2002). As such, as more agents enter the landscape at the same time, while none of them can see the maximum, they can 1) sample more positions more quickly and 2) inform each other as to their current position and consequently attain better performances, i.e. reach higher peaks. Solutions are then combinations (schemas) of attributes (Simon, 1956) and as solvers communicate they can create new combination of various attributes and change their position on the landscape.
For instance, in the InnoCentive challenge "Dual Use – Off Grid – Illumination Device" the following schema can be conceptualized in a simplified way: type of energy (kerosene, gas, petrol, electricity etc) and light type (incandescent, luminescent, LED). Various combinations of these attributes are then submitted by InnoCentive solvers. Should a solver be allowed to communicate he might start from "Petrol, incandescent" but through successive recombinations resulted from repeated encounters he could find "electricity, LED". The former solution is of lower value, but the

latter is the simplified version of the winning solution. As such, the genetic algorithm model that we propose allows for a close match between the model itself and the processes and interactions that occur on actual CPS platforms.

The two chosen case studies can be explained via Simon’s search metaphor in two different ways: with 250.000 registered solvers InnoCentive attempts within each challenge to cover as much of the landscape as possible, while OpenIDEO also profits from having multiple solvers scattered across the landscape (albeit not as many as InnoCentive), but additionally allows them to exchange information and construct solutions together. For example through the “Inspiration” phase where contributors are encouraged to publish descriptions of similar (previously solved) problems and their corresponding solutions, effectively generating a “pool” of potential attributes, from which anyone can ”borrow” to combine into potential solutions.

3.2.1. The genetic algorithm model

The simulations are run using an enhancement of a generic genetic algorithm: Self-Adaptive Segregative Genetic Algorithm including aspects of Simulated Annealing (SASEGASA). The model is run in Heuristics Lab, a freeware framework for heuristic and evolutionary algorithms (see http://heal.heuristiclab.com/ for a detailed description).

Starting from the above detailed assumption that CPS can be modeled with a genetic algorithm, by analyzing the performance of the SASEGASA runs on different problems (varying in size and complexity) in different experimental settings (e.g. combinations of design factors) we were essentially able to generate detailed results about how problem choice and the various design mechanism impact the quality of the solution and the speed of the search.

Genetic algorithms are a class of optimization techniques which share a set of common features and assumptions derived from population genetics: the Mendelian understanding of underlying organism structure (Brownlee, 2011) (e.g. genetic information such as chromosomes, genes, alleles) and evolutionary mechanisms (recombination and mutation). As such, all genetic algorithm variants build on an iterative process where individuals are part of “populations” and can be described by their “genetic material”, which gives a measure of their “fitness” to the environment. The fitness of a given individual is then used in determining whether or not he will have “offsprings” – i.e. whether he will be selected and given the chance to have its genetic material passed on to the next generation. First generations are created by random instantiation (i.e. individuals are randomly scattered across the environment) and sub-sequent generations are created (evolve) through a mix of selection followed by recombination and random mutation. There are various strategies (for a description of the selectors used, see Appendix 2) in selecting which individuals will become part of the pool of "parents", all building on the Darwinian idea that better fit individuals should be allowed better chances of reproduction – i.e. the genetic equivalent of an innovation funnel.

Recombination assumes that two given individuals exchange genetic information, while mutation is a genetic mechanism that allows the introduction of so called "copying errors" with a given probability, traditionally very small. There is a wide literature dedicated (Brownlee, 2011) to the various ways in which recombination (i.e. cross-over) can occur, specific to the type of encoding used. In the case of the model used in this paper which relies on real encoding (as opposed to for instance binary encoding) the cross-over operator involves simply the random selection of the alleles from one "parent" and the other to generate a new solution (e.g. a discrete crossover).

An example could be: from the parents X1 (412, 2) and X2(234, 13) an offspring could be (412, 13)

GAs have been used in a series of problems and domains (Mitchell, 1996) , but the strength of this optimization technique lays in its ability to search the solution space quicker and in a broader way than heuristics based on myopic search (Affenzeller and Wagner, 2003). However, for complex problems, given reasonable mutation ratios, there is simply not enough genetic information in the system and GAs get stuck in local optima. In order to understand how to optimally design CPS set-ups, this paper then builds on a particular type of genetic algorithms that addresses this problem. Affenzeller and Wagner show via experiments that by introducing the two mechanisms presented below it is possible to overcome premature convergence generate generating high quality solutions on a wide array of problems, problem sizes and complexities (Affenzeller and Wagner, 2003).

What SASEGASA does is that it divides the entire population into a certain number of sub-populations (villages or clusters) which evolve independently of each other, until premature convergence occurs. As such, offsprings are only created using selected individuals from these sub-populations, until the fitness of that sub-population stagnates. Subsequently, adjacent sub-populations are "reunified" and the process is repeated, until the stop condition is met, i.e. a satisfactory solution is found. The segregation and independent evolution of sub-populations insures that essential genetic material (i.e. building blocks that are part of the optimal solution) is not discarded in initial stages of the evolutionary process.

The second mechanism implemented by SASEGASA is the self-adaptive selection pressure that works as a steering mechanism: 1) It allows for a quicker search and 2) is used to detect premature convergence and schedules "reunification". The self-adaptive selection pressure is simply a control mechanism that only allows the survival of a certain percentage of offsprings that under-perform their parents to become part of the new generation (for a detailed description see Affenzeller and Wagner (2003)). In effect, traditionally, GAs do not put any restrictions on how solutions recombine, the only pressure mechanism being given by selectionism (i.e. the various selection strategies). The self-adaptive selection pressure fits well with our model as, while solvers are boundedly rational (i.e. they do not know a priori where the optimal solution is to be found) they are still neither always capable of choosing the better solutions, nor randomly choosing, but have a (tunable) propensity towards choosing combinations that lead to improved fitness (the Success Ratio parameter).

[Insert figure 2 about here]

[Insert figure 3 about here]

The algorithm solves a one objective minimization problem using two different fitness functions Schwefel's (Figure 2) and Griewank's function (Figure 3). Experiments were run for two and three dimensions problems. Each of the coordinates of the points represents a building block or an attribute. The unique combination of attributes constructs the solution. The chosen quality parameter is the distance between the best found solution (i.e. across the entire population of solutions, which is the best one) and the optimal solution of the function (in both cases 0) – "Best Quality" parameter. For each chosen configuration there were 50 runs to account for the stochastic character of the search and tests were conducted to compare the differences between means and their significance (t-tests or ANOVAs).

The two functions used in the GA simulations correspond to two different types of problems and are commonly used in GA benchmarking (Affenzeller and Wagner, 2003). They are both multi-modal functions (i.e. they do not describe one-peaked landscapes, which are easy to search). The Schwefel function is a separable function meaning that it can be re-written as a sum of simpler functions (highly modular), while the Griewank function is a non-separable function, of higher complexity with high interdependency between its components.

3.2.3. The NK approach

The NK-model as introduced by Kauffman (1993) is defined by N, the number of attributes an agent carries and K, which is the number of elements that affect the functioning of each attribute.

The NK model allows for a tunable description of the landscape where the K parameter can be assimilated to a complexity index. K=0 describes systems that are the least complex and a problem space that is “smooth”, with a single global optimum, i.e. one single valley. As K increases, the landscape becomes increasingly rugged (with many valleys and peaks), given the epistatic interactions. K=N-1 describes a maximally rugged landscape (Figure 1).

The NK model accounts for such incompatibilities (i.e. the epistatic effect), however, the search for a good solution (vector of attributes) to a problem can best be described as a “myopic hill climbing”, i.e. any solver agent copies only configurations of higher performance from neighboring agents until he can no longer find an agent with higher fitness.

The NK model is widely used, especially in the innovation management literature, but also implies limitations. In a recent paper Billinger et al. (2013) find that actual human search behavior is not as local as the traditional use of the NK model in organizational literature implies. Also Mckelvey at al. (2013) draw attention to the fact that the particular way in which fitness of actors is calculated can lead to “artifactual results” (for an elaboration, see Mckelvey et al. (2013)). We therefore mainly rely on genetic algorithms that do not use the same unrealistic assumptions but cross-validate our results with NK simulations, in order to corroborate our findings with existing literature relying on the NK model.

4. Results

The simulations are conducted as experiments are run in a lab: the influence of various factors is obtained by analyzing the variance in our two dependent variables after running simulations at different levels of the factors (e.g. 1 - control, 5, 10, 15 clusters), holding all other parameters constant.

4.1. Parallel computation: clusters

The first set of simulations attempts to illustrate the advantages posed by allowing for several clusters of solutions to be formed and evolve in parallel. This is exploited by OpenIDEO who allows solvers to group themselves around similar approaches to solving a problem (via tagging) and encourages them to work together in refining their solution.

Four configurations were chosen with “Number of villages” (clusters), which was set to be 1 for the “control runs” and 5, 10 and respectively 15 for the “parallel search”. The results favor undeniably the parallel search (Figure 4 and 5) with all solutions reached in any of the parallel

configurations being closer to the global low compared to the "control runs"– e.g. 4.1 for 10 clusters (Schwefel function, N=2). The "control runs", corresponding to a canonic genetic algorithm quickly get stuck in a local optimum – 140.45 (see Table 1). Simulations also indicate that, while the quality of the solution does increase with the number of clusters, having too many clusters relative to the population size is detrimental for the speed of the execution, and for separable functions, the increase in solution quality is not significant from 10 to 15 clusters, while at the same time the execution time increases significantly. However, for complex problems, a significant increase is to be noted even for higher number of clusters, indicating that more complex problems benefit more from performing a parallel search. For all problem sizes and types, simulations reveal a tradeoff to be made between quality and speed (Table 1).
We confirmed these results using the NK model (Table 2). Even though it seems that in the NK model the set-up with "no communication" outperforms the others, thus disproving previous "clustering" results, it should be noted that by design, in the NK model, all agents are able to communicate at least with their (N-1) neighbors. All other tested network structures (e.g. communication strategies) are either inferior (Table 2a) or differences are insignificant in low complexity problems. The NK runs thus endorse that there is to be noticed a gain in efficiency when the network is neither dense, nor sparse. We have run further simulations to investigate how allowing solvers to sample remote points (i.e. "long jumps") in the solution space in the attempt to find better configurations performance was increased (Table 2b), but the result that it is the sparse communication strategy, is upheld.

[Insert Table 1. about here]

[Insert Table 2. about here]

[Insert figure 4 about here]

[Insert figure 5 about here]

4.2. Iteration

A different experiment was set up to compare an iterative approach to CPS (such as OpenIDEO) with a non-iterative approach (such as InnoCentive). Even if InnoCentive's initial population was chosen to be significantly higher and it does lead to good results (the mean solution quality is 0.096), comparatively, the simulations provide evidence that population size does not make up for the creativity lost by not allowing repeated iterations – simulations using the iterative approach have an average solution quality of 5.39E-07 (rounded up to 0) (Table 3). Overall what the simulations indicate is that the adaptive (i.e. using several generations) approach leads consistently to good results with smaller populations for non-separable problems (Table 3) and it should be underlined that in order to obtain results that are similar to OpenIDEO, InnoCentive must attract considerably more contributors to work on a given challenge (the population size was 10.000, 100 times higher than OpenIDEO's).

[Insert Table 3. about here]

[Insert Table 4. about here]

However, for separable functions (Table 4), the property is reversed and simulations suggest that for simpler problems having a larger initial population leads to higher quality in the final solution corroborated with a significantly lower execution time. As such using an iterative problem solving design seems to be the preferable approach when solving complex problems. Similar results were obtained for N=3 problems. In the NK model, results were confirmed for moderate and high complex problems in comparing an iterative approach with long jumps to the "large population" approach, showing that indeed iteration can be used as a successful strategy in mitigating small numbers of contributors (Table 5), but were not confirmed when the iterative approach was restricted to myopic search as this type of search does not generate enough diversity to compensate for the lack of diversity in the initial population (i.e. without long jumps, large numbers of solvers are needed).

[Insert Table 5. about here]

4.3. Selection strategies and selection pressure

Traditionally, CPS set-ups use selection strategies best assimilated with "best selectors" either by simply eliminating non-performing solutions, or by encouraging by design the survival of the "best solutions" – e.g. by generating a public classification of solutions with the "highest scores", "most voted" etc. (Dellarocas, 2010). In subsequent phases, contributors are prone to borrow, tweak, modify these very good solutions, but on the long run simulations suggest that this is a losing strategy, as the "best" selector is significantly outperformed in all experiments as it drives diversity out of the system too quickly and as such essential information is lost. Separable problems of low dimension (N=2), show significant differences between selection strategies, indicating that it is only after a certain level of ruggedness that this property holds (Table 6).

Simulation results are inconclusive as to which of the remaining selection strategies is the most appropriate one. They show nevertheless that a more complex strategy should be employed then the simple elimination of the less performing solutions to insure the survival of less than optimal solutions for longer time and as such preserve enough diversity in the system.

[Insert Table 6. about here]

[Insert Table 7. about here]

5. Discussion

Following the problem-solving perspective (Nickerson and Zenger, 2004; Felin and Zenger, 2013; Lakhani et al., 2013) we start from the conjecture that different kinds of problems "demand" different collective problem solving set-ups as shaped by the different combinations of design mechanisms. Unlike previous studies our contribution thus focuses exclusively on CPS and investigates how design factors influence the efficiency of CPS set-ups given different problem sizes and degrees of complexity. In particular, our results shed light on three different mechanisms: clustering, iterative approach and selection mechanisms.

Our simulations, in line with prior studies (Benkler, 2002; Padget et al. 2009; Estelles-Arolas and Gonzalez-Ladron-de-Guevara, 2012), confirm that a large population size is essential for successful CPS. The larger the population, the higher are the chances that candidate solutions would populate as much as possible of the problem space. In contrast, if the population is too small, the risk of premature convergence is increased (Holland, 1992). Since an increased number of contributions involve increased costs (Alexy et al., 2011) a main challenge is how to optimize the number of contributors. Consequently, we propose three mechanisms that enhance the performance of CPS set-ups and at the same time help alleviate problems associated with having relatively reduced numbers of solvers: creating an iterative set-up, allowing solvers to work in clusters and resorting to more advanced selection mechanisms.

Simulations suggest that allowing for repeated interactions between solvers can mitigate having an initially reduced number of solvers. Allowing communities of solvers to work together for a large number of generations will, all things being equal contribute to a significant improvement of the solution, even with initial small populations as additional diversity can be created as solvers are allowed to see other positions in the landscape and inform their search accordingly. This result is in line with recent studies regarding the benefits of "free revealing" and "knowledge brokering" (Villarroel and Tucci, 2013) and recent studies in the innovation management literature arguing against linear processes and acknowledging the importance of trials and feedback loops (Levardy and Browning, 2009).

The mechanisms highlighted above are consistent with the design of OpenIDEO: Unlike InnoCentive who relies on finding as many candidate solutions as possible to see which is the one that closest to solving the problem, OpenIDEO's approach is to populate the same landscape with a smaller number of contributors and endorse initial solutions' improvement over repeated iterations, each iteration bringing it closer to the optimal solution. While the website has a fairly large number of users (over 22.000) each challenge has an average of only 500 users in its "Inspiration" phase, traditionally the one that attracts the largest number of contributors. By allowing repeated interactions that facilitates seeing other solvers' positions in the problem landscape, the quality of the final solution is significantly improved.

Parallel problem solving is another design tool that we propose as a highly efficient way of structuring the interactions between the solvers allowing high quality solutions to be reached without a significantly large initial pool of contributors. Allowing the initial population to form several smaller clusters which evolve in parallel greatly enhances both the speed of execution and the quality of the solution reached. The reasoning behind this result can be found in the efficiencies gained by resorting to parallel search. Letting parallel solutions evolve simultaneously, irrespective of their comparative fitness at a given time, allows for essential information to be kept within the system and better solutions to be reached. A system with initial diversity, but no mechanisms that allow for parallel evolution of solutions, will not exploit to its full potential the problem space and will eliminate prematurely diversity from the system. OpenIDEO implements parallel search by not limiting the number of concepts that are published after the first round and later in "Refinement" in which several concepts are allowed to enter.

This allows for valuable ideas to be saved, combined with other contributions and improved solutions to be created.

Parallel search is also a way of dealing with problem size. In this respect our simulations are in line with evolutionary computation literature (Brownlee, 2012) and show that problem size is negatively correlated with the quality of the solution reached and positively correlated with the execution time. Limiting the size of the problem by setting up parallel tracks of search is a technique used by both OpenIDEO and Innocentive. InnoCentive receives the problem from a seeker, decomposes it into several smaller problems and then publishes them as separate challenges while OpenIDEO does so within one challenge. OpenIDEO allows users to use tagging mechanisms (such as themes, titles etc.) so that they can further freely decompose the problem – i.e. they spontaneously identify sub-problems they want to pursue and several solvers group around one such sub-problem.

Finally, the choice of selection mechanism also has an impact on the effectiveness of the CPS process, since simulations suggest that implementing more advanced types of selectors is more efficient and leads to better solutions. In order to benefit from iterative interaction, one can select to display various solution proposals during the different phases of the CPS. Our findings show that one should not only display what currently appears to be the best solution, but display according to various criteria (an approach already used by OpenIDEO). E.g., one could rely on displaying "most recently added" solutions or "fast rising" solutions. In this way solutions that at a given phase seem less efficient are preserved for later phase, in order to avoid eliminating false negatives (cf. Chesbrough et al., 2006). Thus organizations face a strategic choice between eliminating unviable solutions as early as possible, to minimize time needed to reach a solution on the one hand and preserve enough diversity to reach the high quality solutions on the other.

As conjectured by the problem-solving perspective (Felin and Zenger, 2013) problem complexity influences the problem solving set-up. In the genetic algorithm simulations, the complexity of a problem is modeled as a property of the function onto which the search is applied (i.e. of being or not separable). Simulations indicate that modular functions can be solved with higher efficiency (in terms of time and size of initial population). These findings are in line with the "modularity" property a task should have (as identified by Baldwin and Hippel (2011) and Benkler (2002)). More significantly though, results suggest that CPS set-ups which resort to clustering and advanced selection mechanisms are able to deal with a certain degree of epistatic effect successfully and are capable of finding solutions that display interdependence between components, as the efficiency gains due to repeated interactions are greater in the case of highly complex function, as well as the positive effect of clustering and using advanced selection mechanisms. Overall, we contribute to the emerging literature of problem solving perspective of the firm with a particular focus on crowdsourcing.

6. Limitations and future research

As pointed out by Mason and Watts (2012), Billinger et al. (2013) and Mckelvey et al. (2013) the use of modeling and simulations in social sciences can have serious drawbacks as often times modelers rely on un-realistic assumptions of human behavior which in turn lead to un-realistic results. Our simulation model can replicate observations made by previous research in connected fields such as crowdsourcing (Afuah and Tucci 2012), cognitive psychology and artificial intelligence (Newell and Simon 1972), user-driven innovation (Baldwin and Hippel 2011), problem solving literature (Page, 2007, Lazer and Friedman, 2007, Mason and Watts, 2012) as well as with Felin & Zenger's (2013) recent attempt to conceptually identify an optimal design mechanism configuration contingent on the problem to be solve. Nevertheless, an agent-based modeling approach would be desirable as it would allow for more detailed assumptions about the agent (such as the influence of social status or motivation) and findings to be also validated within an experimental setting, in order to construct a more accurate picture of real life problem solving behavior and advance an understanding of how humans collectively solve problems.

One final limitation of this paper is that the way the genetic model is set-up it can only account for a twofold distinction – complex and non-complex problems (non-separable versus separable), when in practice, as the NK metaphor suggests, there is a continuous spectrum to be accounted for. These issues are to be tackled by future theoretical and empirical research.

Reference list

Adamczyk, S., A. C. Bullinger, et al. (2012) Innovation Contests: A Review, Classification and Outlook. *Creativity and Innovation Management* 21(4), 335-360.

Affenzeller, M., Winkler, S., Wagner, S. and Beham, A., (2009) *Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications*. Boca Raton: CRC

Affenzeller, M. and S. Wagner (2003) SASEGASA: An Evolutionary Algorithm for Retarding Premature Convergence by Self-Adaptive Selection Pressure Steering. *Computational Methods in Neural Modeling* Springer Berlin Heidelberg. 2686, 438-445.

Afuah, A. and C. L. Tucci (2012) Crowdsourcing as a Solution to Distant Search. *Academy of Management Review* 37(3), 355-375.

Alexy, Oliver and Criscuolo, Paola and Salter, Ammon, (2011) No Soliciting: Strategies for Managing Unsolicited Innovative Ideas. *California Management Review*, 54 (3), 116-139.

Baldwin, C., and Hippel, V.E (2011) Modeling a Paradigm Shift: From Producer Innovation to User and Open Collaborative Innovation. *Organization Science* (22.6), 1399-1417.

Bayus, B. L. (2013) Crowdsourcing New Product Ideas over Time: An Analysis of the Dell IdeaStorm Community. *Management Science* 59(1), 226-244.

Bonabeau, E. (2009) Decisions 2.0: The Power of Collective Intelligence. *Mit Sloan Management Review* 50(2), 45-+.

Boudreau, K. J. (2012) Let a Thousand Flowers Bloom? An Early Look at Large Numbers of Software App Developers and Patterns of Innovation. *Organization Science* 23(5), 1409-1427.

Benkler, Y. (2002) Coase's Penguin, Or, Linux and The Nature of the Firm, *The Yale Law Journal.*

Brownlee, J. (2011) Clever Algorithms: Nature-Inspired Programming Recipes.

Bullinger, A. C., A.-K. Neyer, et al. (2010) Community-Based Innovation Contests: Where Competition Meets Cooperation. *Creativity and Innovation Management* 19(3), 290-303.

Chesbrough, H. W., W. Vanhaverbeke, et al. (2006) *Open innovation: researching a new paradigm.* Oxford, Oxford University Press.

Cooper, R. G., S. J. Edgett, et al. (2002) Optimizing the stage-gate process: What best-practice companies do - I. *Research-Technology Management* 45(5), 21-27.

Davis, J. P., K. M. Eisenhardt, et al. (2007) Developing theory through simulation methods. *Academy of Management Review* 32(2), 480-499.

Dellarocas, C. (2010) Online Reputation Systems: How to Design One That Does What You Need. *Mit Sloan Management Review* 51(3), 33-+.

Estelles-Arolas, E. and F. Gonzalez-Ladron-de-Guevara (2012) Towards an integrated crowdsourcing definition. *Journal of Information Science* 38(2), 189-200.

Felin, T. and T. R. Zenger (2013) Open Innovation, Problem Solving, and the Theory of the (Innovative) Firm. *Research Policy* Forthcoming

Fioretti, G. (2013) Agent-Based Simulation Models in Organization Science. *Organizational Research Methods.*16(2), 227-242

Fullerton, R. L. and R. P. McAfee (1999) Auctioning entry into tournaments. *Journal of Political Economy* 107(3), 573-605.

Goldberg, D. E., (2002) Design of Innovation: Lessons From and For Competent Genetic Algorithms, Kluwer, Boston, MA.

Hansen, M. T. (1999) The Search-Transfer Problem: The Role of Weak Ties in Sharing Knowledge across Organization Subunits. *Administrative Science Quarterly* 44(1), 82-111.

Hertel, G., S. Niedner, et al. (2003) Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. *Research Policy* 32(7), 1159-1177.

Holland, J. H. (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. Cambridge, Mass., MIT Press.

Howe, J. (2009) Crowdsourcing: Why the power of the crowd is driving the future of business. London, Random House.

Hutter, K., J. Hautz, et al. (2011) Communitition: The Tension between Competition and Collaboration in Community-Based Design Contests. *Creativity and Innovation Management* 20(1), 3-21.

Jeppesen, L. B. and L. Frederiksen (2006) Why do users contribute to firm-hosted user communities? The case of computer-controlled music instruments. *Organization Science* 17(1), 45-63.

Jeppesen, L. B. and K. R. Lakhani (2010) Marginality and Problem-Solving Effectiveness in Broadcast Search. *Organization Science* 21(5), 1016-1033.

Kauffman, S. A. (1993) The origins of order: self-organization and selection in evolution. New York, Oxford University Press.

Khatib, F., S. Cooper, et al. (2011) Algorithm discovery by protein folding game players. Proceedings of the National Academy of Sciences. 108(47), 18949-18953.

Lakhani, K. R., H. A. Lifshitz, et al. (2013) Open Innovation and Organizational Boundaries: Task Decomposition, Knowledge Distribution and the Locus of Innovation. *Handbook of Economic Organization: Integrating Economic and Organization Theory*. A. Grandori. Northampton, MA, Edward Elgar Publishing: 355-382.

Lakhani, K. R., K. J. Boudreau, et al. (2013) Prize-based contests can provide solutions to computational biology problems. *Nature Biotechnology* 31(2), 108-111.

Lazer, D. and A. Friedman (2007) The Network Structure of Exploration and Exploitation. *Administrative Science Quarterly* (54.2), 667-694

Leimeister, J. M., M. Huber, et al. (2009) Leveraging Crowdsourcing: Activation-Supporting Components for IT-Based Ideas Competition. *Journal of Management Information Systems* 26(1), 197-224.

Levardy, V.; Browning, T.R. (2009) An Adaptive Process Model to Support Product Development Project Management, *Engineering Management, IEEE Transactions on*, vol.56, no.4, pp.600,620

Mason, W. and D.J. Watts (2012) Collaborative learning in networks. *Proceedings of the National Academy of Sciences*, vol. 109, no. 3, pp. 764–769, National Acad Sciences, 2012

Mckelvey, B., L. Meng, et al. (2013) Re-thinking Kauffman's NK fitness landscape: From artifact & groupthink to weak-tie effects. *Human Systems Management* 32, 17-42.

Mitchell, M. (2009) Complexity : a guided tour. Oxford England ; New York, Oxford University Press.

Newell, A. and H.A. Simon (1972) Human problem solving. Englewood Cliffs, N.J., Prentice-Hall.

Nickerson, J. A. and T. R. Zenger (2004) A Knowledge-Based Theory of the Firm - The Problem-Solving Perspective. *Organization Science* 15(6), 617-632

Page, S. (2007) The Difference: How the Power of Diversity Creates Better Groups, Firms, Schools and Societies, Princeton University Press.

Piezunka, H. and L. Dahlander (2013) Open to Suggestions: How Organizations Elicit Suggestions Through Proactive and Reactive Attention. *Research Policy* Available online 26 July 2013

Poetz, M. K. and M. Schreier (2012) The Value of Crowdsourcing: Can Users Really Compete with Professionals in Generating New Product Ideas? *Journal of Product Innovation Management* 29(2), 245-256.

Raymond, E. S. (1999) The cathedral and the bazaar: musings on Linux and open source by an accidental revolutionary. Beijing; Cambridge, O'Reilly.

Schweitzer, F. M., W. Buchinger, et al. (2012) Crowdsourcing: Leveraging Innovation through Online Idea Competitions. *Research-Technology Management* 55(3), 32-38.

Simon, H. A. (1996) The sciences of the artificial. Cambridge, Mass., MIT Press.

Slawsby, A. and C. Rivera (2007) Collective innovation. Thesis (M.B.A.)—Massachusetts Institute of Technology, Sloan School of Management

Tate, D. M. and Smith, A. E. (1995) A genetic approach to the quadratic assignment problem. *Computers & Operations Research*, vol. 22, pp. 73-83.

Terwiesch, C. and Y. Xu (2008) Innovation contests, open innovation, and multiagent problem solving. *Management Science* 54(9), 1529-1543.

Villarroel, A. J., J. E. Taylor, et al. (2013) Innovation and learning performance implications of free revealing and knowledge brokering in competing communities: insights from the Netflix Prize challenge. *Comput Math Organ Theory* 19, 42-77.

Table 1. Variation of performance with number of clusters (Schwefel function, n=2)

Schwefel N=2			
No of.Clusters	N	Mean Execution time	Mean Best Quality
1	50	0:02:14	140.45
5	50	0:33:27	22.38
10	50	1:51:47	4.1
15	50	3:46:57	2.36
Total	200	1:33:36	42.32

	Clusters	Clusters	Significance values
Tamhane T2 test (inequality of variance)	1	5	0
		10	0
		15	0
	5	1	0
		10	0.004
		15	0.001
	10	1	0
		5	0.004
		15	0.853
	15	1	0
		5	0.001

Table 2. Communication strategies in NK runs

JUMP =1 N=14	No network	Fully conn	Linear	Random 10	SWN 10
k=1	0,7198	0,7170	0,7185	0,7187	0,7188
k=7	0,7877	0,7790**	0,7809**	0,7798**	0,7792**
k=11	0,7791	0,7775**	0,7749**	0,7744**	0,7753**
JUMP = no limit	No network	Fully conn	Linear	Random 10	SWN 10
k=1	0,7186	0,7173	0,7201	0,7180	0,7146
k=7	0,7892	0,7788**	0,7791**	0,7782**	0,7783**
k=11	0,7842	0,7794**	0,7777**	0,7789**	0,7775**

**sig .01 refers to the comparison between “no network” and the respective strategy.

Table 3. Non-iterative/iterative set-ups. (Griewank function, n=3)

Griewank N=3				
	N	Mean	Std. D.	Std.Error Mean
g=1/p=10000	50	0.096**	0.053	0.005
g=100/p=100	50	0.00**	0	0

**sig.01

Table 4. Non-iterative/iterative set-ups. (Schwefel function, n=2)

Schwefel N=2					
	N	Mean	St. Deviation	Std. Error	Mean execution time
p=10.000/g=1	50	2,463*	2,469	0,349	0,01
p=100/g=100	50	10,06*	22,176	3,136	0,08

*sig .05

Table 5. Non-iterative/iterative set-ups. (NK runs)

N=14	100 (no jump)	100 (w. Jump)	10.000
k=1	0,7172	0,7186	0,7185
k=7	0,7792	0,7892**	0,7810
k=11	0,7783	0,7842**	0,7804

***sig .01*

Table 6. Performance quality variation with selection strategy. (Griewank function, n=2)

Griewank N=2				
	N	Mean	Std. Deviation	Std. Error
Proportional	50	1.31806E-06	5.99185E-06	0.000
Tournament	50	7.3634E-09	4.93723E-08	0.000
Best	50	0.015	0.011	0.002
generalized	50	0.000	0.000	0.000
Total	200	0.004	0.008	0.001

	(I) Selectors	(J)selectors	Sig.
Tamhane	Proportional	Tournament	0.561
		Best	0**
		Generalized	0.555
	Tournament	Proportional	0.561
		Best	0**
		Generalized	0.879
	Best	Proportional	0**
		Tournament	0**
		Generalized	0**
	Generalized	Proportional	0.555
		Tournament	0.879
		Best	0**

*The mean difference is significant at the 0.01 level.

Table 7. Performance variation with selection strategy.

	Schwefel		Griewank	
	N=2	N=3	N=2	N=3
Worst performing selector*	/	Best	Best	Best

*sig. 05

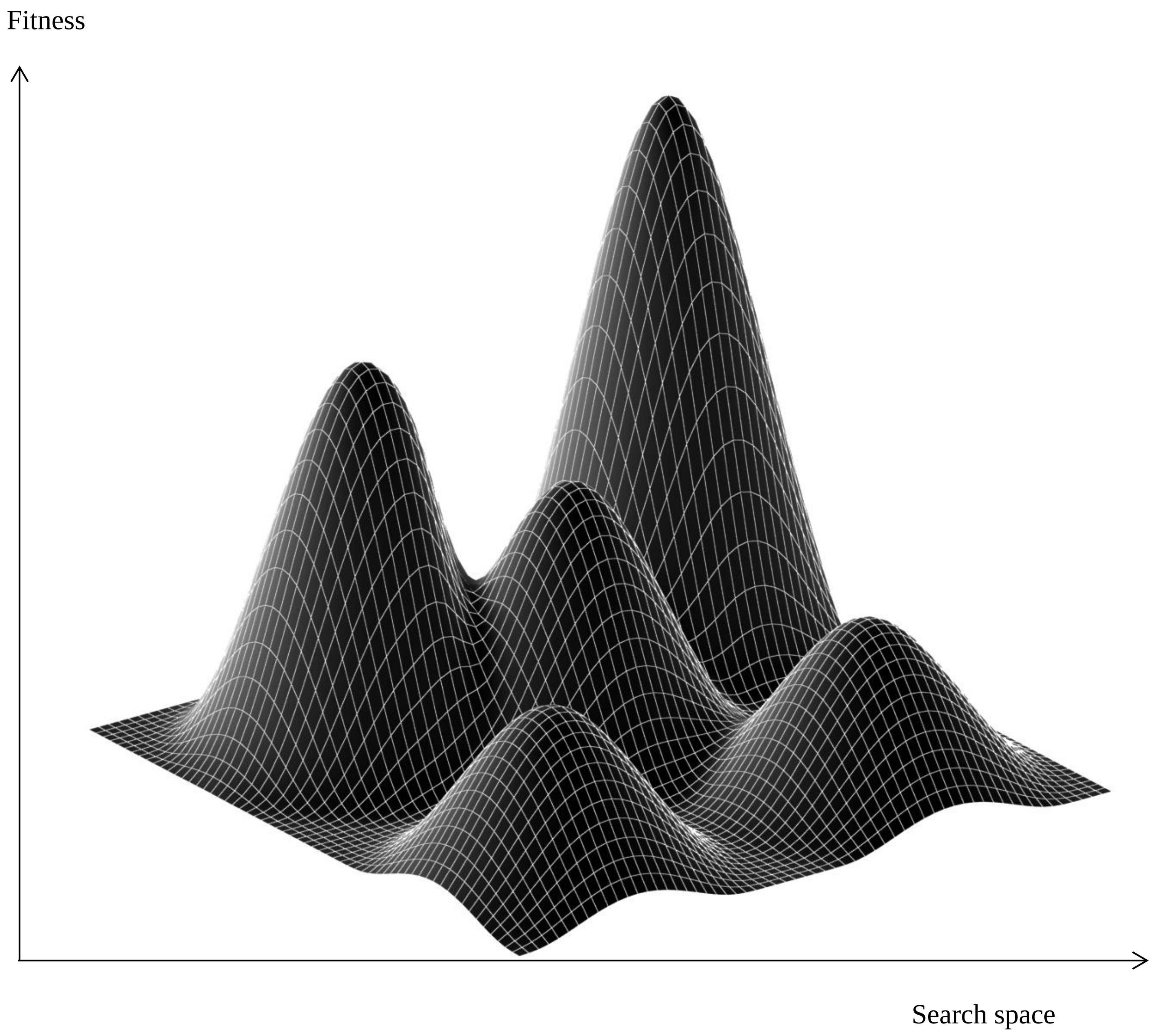

Figure 1: NK landscape

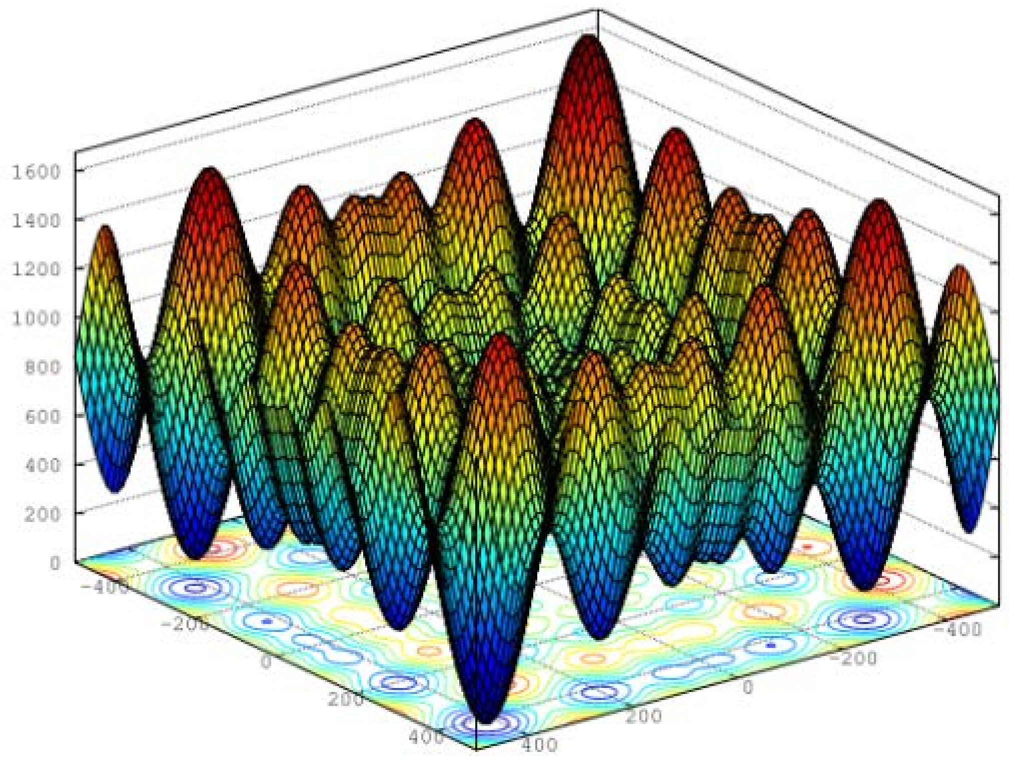

Figure 2: *3D illustration of multimodal separable function*- Schwefel function (Source: http://dev.heuristiclab.com)

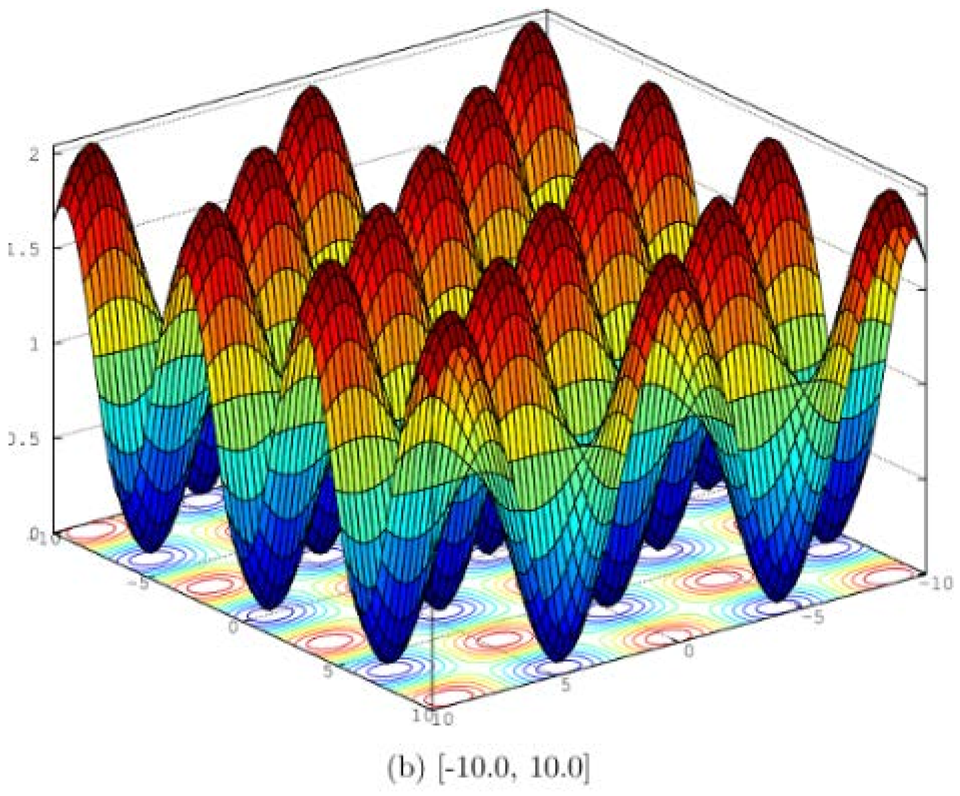

Figure 3: *3D illustration of multimodal non-separable function*- Griewank function (Source: http://dev.heuristiclab.com)

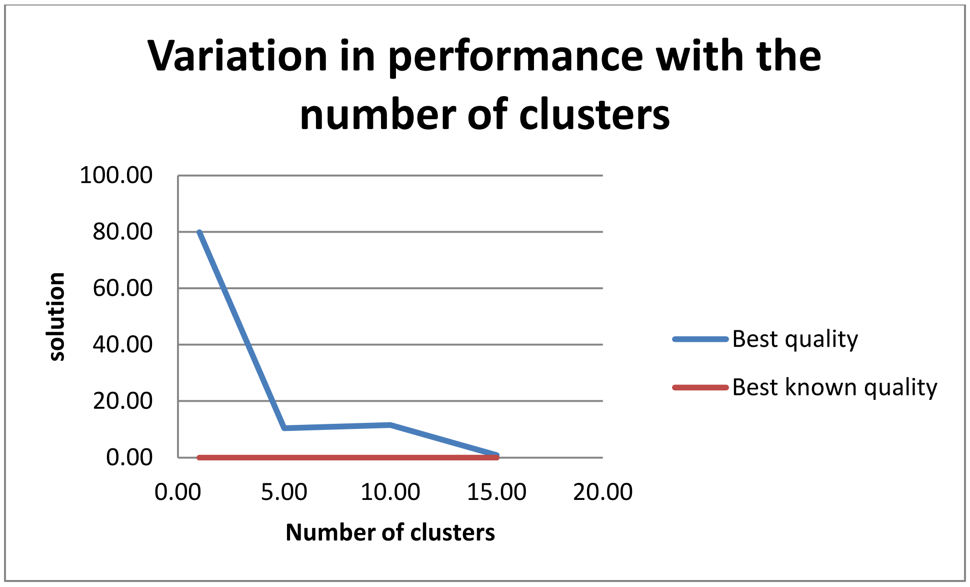

Figure 4: Variation of performance with number of clusters (Schwefel function, n=2)

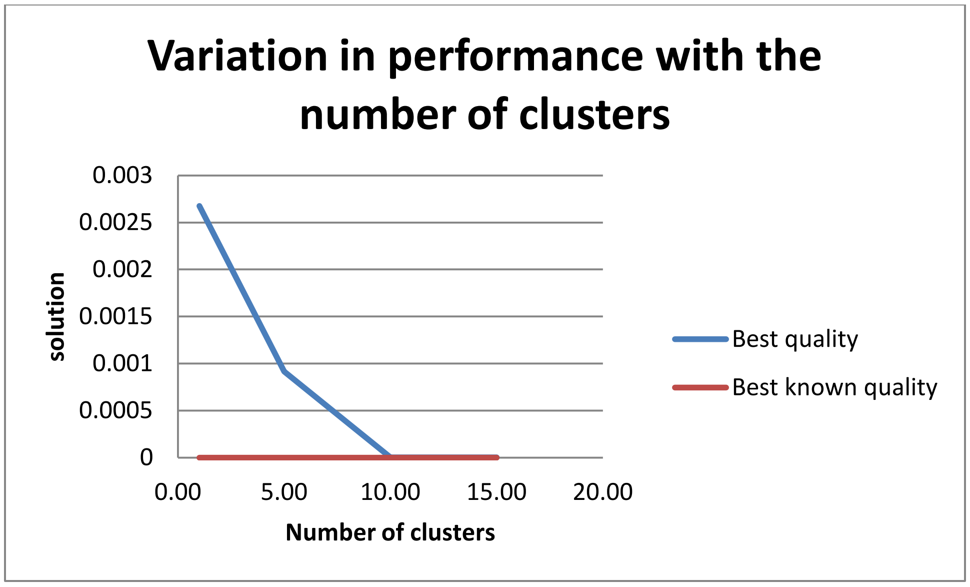

Figure 5. Variation of performance with number of clusters (Griewank function, n=2)

Appendix 1: NK model (available here: https://wiki.bath.ac.uk/display/sendero/NK)

NK model parameters

The NK simulations are run on Sendero – a freely downloadable java applet developed at Bath University, UK. The model uses a straightforward implementation of the NK model in which the landscape represents a performance map of all possible solutions to a given problem. Any solution is seen as a vector of attributes, which can have multiple states. In line with NK literature, we select a binary design (e.g. an attribute can only be 1 or 0), also reducing time needed to carry out simulations. In the NK model, the contribution to the fitness of each individual attribute is given by a random draw from a uniform distribution and the fitness of a solution is simply the average of all its composing attributes. Agents search the landscape trying to identify better positions, mostly by local search (by randomly sampling the fitness of their neighbors). Long jumps (the equivalent of distant search) can be absent, limited or un-limited. In the Sendero model, an agent can only engage in distant search if he can no longer find better alternatives in his neighborhood and he has a limited or un-limited number of long jumps that he can sample. Agents do not have "memory" and as they sample the solution space they may encounter and adopt solutions that they previously discarded.

1. Number of agents: 100 respectively 10.000
2. Number of generations: 100 respectively 1 (i.e. the number of times an agent can adjust his position based on information from the environment – either his neighbors or via long jumps)
3. K=0, n-1 landscape ruggedness – the number of interdependencies between each attribute. We have chosen K=1, K=7 and K=11 corresponding to three different degrees of landscape ruggedness (i.e. from flat to rugged). Although innovation problems do have a high degree of interdependence between components, we believe that the moderate level of complexity is the more accurate description of most innovation problems (also see Lazer and Friedman, 2007), therefore we have not run simulations for K=13.
4. N = problem dimension – the number of attributes (alleles loci) that each solution has. N was chosen to be 14, although several runs were made with different (smaller) problem sizes and results were consistent with the findings presented here
5. D = number of neighbors – (A-1) N – one mutant neighbors – in our case A (number a states an attribute can have) was set to 2 hence the number of neighbors was N (i.e. 14)
6. Social network structure

- Small worlds network (with a radius) – i.e. in our tables the notation SWN 10 refers to a small world network with a 10 radius
- Random (with a connectivity probability percentage) – i.e. in our tables the notation Random 10 refers to the density of the random network
- Linear network
- Fully connected
- No network (each agent can only see its neighbors)

7. Type of search used in the reported runs: “Random one mutant”, i.e. each agent randomly chooses a neighboring configuration and moves there if and only if the new configuration has a higher fitness.

Appendix 2: SASEGASA model (available here: http://www.heal.heuristiclab.com/)

SASEGASA parameters

Problem: single objective

Fitness functions:

a) Schwefel *f(x) = 418.9829*n + sum(-x(i)*sin(sqrt(abs(x(i)))) with -500 <= x(i) <= 500*

and

b) Griewank *f(x) = 1/4000*sum(xi-100)^2 - prod((xi-100)/sqrt(i)) + 1 with -600 <= x(i) <= 600*

Problem size: 2 respectively 3

Number of agents: 100 respectively 10.000

Number of generations: 1 respectively 100

Number of villages (clusters): 1, 5, 10, 15

Success Ratio 0.5

Type of selector:

i) Best solutions are evaluated and a sub-population of the best solutions is formed. Parents are stochastically chosen from this pool of solutions.

ii) Proportional – solutions are evaluated and ranked according to fitness. Solutions with higher fitness are given higher chances to pass their genetic information into the next generation (also known as Roulette Wheel Selector)

iii) Tournament – pairs or groups of 3 or higher number of solutions are evaluated and winners enter the pool of potential parents.

iv) Generalized rank - The generalized rank selection (Tate and Smith, 1995) is a type of selection that is inclined towards choosing the better solutions from the current population, while still allowing for random cross-over to occur.