Exploring modularity and hierarchy in the NK-landscape

Adam Campbell
University of Central Florida

Laura Feeney
Swedish Institute of Computer Science and Uppsala University

Orion Penner
University of Calgary

Meritxell Vinyals
Artificial Intelligence Research Institute(IIIA)
Spanish Scientific Research Council

Draft: 15 August, 2008
Please do not re-distribute.

Abstract

The NK-model is a well-known combinatorial optimization problem, which can be interpreted as the minimization the total energy of a network of coupled spins, where the energy contribution of each spin is determined by the states of its neighbors.

One study[4] of the NK-model on a regular lattice suggests that lower energy solutions can be obtained using a divide-and-conquer approach, in which the network is partitioned into patches that are optimized independently of each other. This result is somewhat surprising, because couplings between spins in different patches are not taken into account. We note, however, that improvement is very small, though statistically significant, and present only in certain parameterizations.

In this work, we reproduce a subset of these results and take advantage of recent advances in the detection of modularity and community structures in real-world networks to study the effectiveness of the divide-and-conquer strategy in non-lattice networks. We look at two cases, the purely modular approach outlined above and a hierarchical approach, which alternates between modular optimization and optimizing over the network.

In general, our results are not dissimilar to those above: we see small differences, but find it difficult to argue that they are consistently statistically significant. This report presents intermediate results obtained during the authors' participation at the Santa Fe Institute Complex Systems Summer School.

1 Introduction

Simon[9] proposed the idea of *nearly decomposable systems* as a model for how complexity can be addressed in both natural and constructed systems. In such systems, the interactions between modules are weak and the subsystems behave nearly independently. This means that some interactions are relatively less important than others and can be ignored, with minimal effect on the functioning of the system. This powerful notion has been applied in many fields, including social sciences, psychology and artificial intelligence and engineering design.

The theory of nearly decomposable systems can also be applied to optimization problems. Common optimization procedures are intended to improve performance of the system as a whole. In the case of hard (usually NP-Complete) optimization problems, this approach is often infeasible due to the large size and complexity of the solution space. Partitioning or modularization is a way of exploiting the problem structure to obtain a near-optimal solution at lower computational cost.

Kauffman and co-authors have studied this approach in the context of optimization on the NK landscape. The NK landscape[3] is an optimization problem in which the goal is to minimize total system energy for a collection of spin states, where the contribution of each spin depends on the state of K other spins[1]. Kauffman and co-authors have also studied[4] the effect of partitioning on an NK landscape consisting of a regular lattice, where each spin is coupled with its K nearest neighbors on the lattice.

This lattice has a natural partitioning into square patches, each of which can be optimized "selfishly", minimizing the total energy of the patch, without regard to the whether the chosen configuration leads to a global minimum. The surprising result of this work is that, as K increases, there is an intermediate patch size which obtains a lower energy solution, relative to optimizing directly over the entire lattice. That is, partitioning the lattice – ignoring some subset of spin couplings while attempting to find an optimal configuration within each patch – can obtain a lower total minimum energy! Kauffman and co-authors further report that the patch size at which this occurs is associated with the transition to a frozen state, which occurs "at the edge of chaos" (we do not address this topic in our work). It should be noted, however, that the performance difference, though statistically significant, is very small and is present only in certain configurations.

The partitioning of a lattice into square patches defines a very simple kind of structure. In the last few years researchers have proposed various algorithms [8] [7] [1] [5] for finding the modular structures of complex real-world networks. To the best of our knowledge, these algorithms have never been applied to exploit the structure in optimization problems. In this work, we take advantage of recent advances in the study of modularity and community structure in networks to study the possibility of applying the divide-and-conquer approach described above to optimization in more complex networks.

[1]In this work, we interpret the model in terms of minimizing energy.

Specifically, we apply the community detection algorithm defined in [5] to partition a variety of non-lattice NK-landscapes and to study the effect on the optimization process. We also test a hierarchical approach, where patches are selfishly optimized, then combined into larger patches and re-optimized, using the result of the previous optimization as the initial state. We continue to use the NK-models presented above as a framework for or experiments.

This paper is structured as follows. The next four sections describe the NK model and combinatorial optimization problem, the Glauber dynamics-based solver, the partitioning algorithms, and some details about our implementation. Our experimental results follow and are divided into three main categories: replicating the results of [4]; modular and hierarchical optimization of NK-landscapes for the case of "highly modular" and random networks; and the complete solution of two small and computationally tractable NK models.

2 NK-landscapes

The NK-model proposed by Kauffman[3] defines a rugged energy/cost surface or fitness landscape, where the number of interacting elements elements is controlled by a parameter K. The model is very general because the elements can be interpreted in many ways: in the biological context, as adaptation in a fitness landscape; in a physical system, as the minimization of energy; or in an economic context as the minimization of cost or maximization of gains. In this work, we use the language associated with minimizing the energy associated with an network of coupled spins.

More specifically, an NK-landscape is composed of N elements, each of which can be in one of S states; typically they are modeled as binary states or spins. Each element takes on an energy value that depends on the state of that element and of K other elements. The energy function is a random function that maps each of $S^(K+1)$ combinations of states to a fixed value. The value of K, which can range from 0 to $N-1$, determines the level of cross-coupling in the system. For any given configuration, the energy of the system is defined as the average of the energy of its elements. The optimization problem is to determine a set of N states that minimizes the energy of the system.

Formally, in a NK-model each element $i \in N$ defines an energy function:

$$E_i(s_i; s_{i1} \dots s_{ik}) : S^{K+1} \rightarrow \Re \tag{1}$$

Where s_i is the state of the element i and $s_{i1} \dots s_{ik}$ are the states of K elements, neighbors of i. Values of the energy function are drawn from a uniform distribution on [0,1)..

The energy system for any given configuration of elements is defined as:

$$E(s_1, \dots, s_N) = \frac{1}{N} \sum_{i=1}^{N} E_i(s_i; s_{i1}..s_{ik}) \tag{2}$$

which is the average energy per element.

3 Optimization solvers

In this section we discuss the complexity of a solver that minimizes the energy of systems as NK-landscape defined in section above.

Notice that an optimal solver has to find the system configuration that minimizes the system equation. Formally:

$$s* = min_{s \in S_1 \times \ldots \times S_N} E(s) \tag{3}$$

Where $S_1 \times \ldots \times S_N$ are all possible system configurations. Notice that this solver has the problem that it is exponential to the number of elements of the system. This is because it has to explore S^N possible solutions in order to get the optimal one.

Therefore in order to avoid this combinatorial explosion different procedures have been proposed in order to get a good solution for the system reducing the complexity of the search.

One of this procedures is the *Glauber dynamics* procedure [6] that consists of picking an initial configuration and flipping at each iteration a single spin if and only if it minimizes the total energy of the system. Notice that this procedure reduces the combinatorial explosion of an optimal solver by considering the effect of flipping each element independently of other elements. However it may not give the optimal solution. Imagine a system composed of two elements e_1 and e_2. The optimal state for the system is $e_1 = 1, e_2 = 1$ and the initial state is $e_1 = 0, e_2 = 0$. Then if states $e_1 = 0, e_1 = 1$ and $e_1 = 1, e_0 = 0$ are both lower than the initial state this procedure will find the solution. For all these algorithms is necessary in order to find a solution that exists a sequence of states such that:

- The first step is the initial state and the final state is the optimal system state
- In all the sequence the state n+1 has a Hamming distance of 1 with respect to the state n.
- The system energy lowers through all the states

This is equivalent to find a path that lowers the system energy in the N-bit binary hypercube.

In what follows we introduce the solver used through all the experiments, a random version of the Glauber dynamics procedure.

3.1 Random solver

The random solver is an implementation of the Glauber dynamics procedure that chooses at each iteration one spin at random to update. Hence elements are updated independently in a random order and flipped if they satisfy the criterion. This solver updates the system, using update equations that depend on a single element:

$$s_i^{t+1} = argmax_{s_i \in S} E(s_i; s^t) \tag{4}$$

At each time only one element is updated.

4 Partitioning algorithms

Imagine we partition the system into P non-overlapping partitions, where each partition p is composed of a subset of elements, formally $p_1 \cup \ldots \cup p_{|P|} = N$. Then the energy of this system, formulated in Eq.2 , can be rewritten as the sum of the energy over partitions:

$$E(s_1, \ldots, s_N) = \frac{1}{N} \sum_{p \in P} \sum_{i \in P}^{N} E_i(s_i; s_{i1}..s_{ik}) \tag{5}$$

When the random solver, introduced in section above, is applied over a partitioned system it updates at each iteration one patch, choosing randomly one spin in such patch. There are different ways of partitioning a problem. In what follows we introduce the techniques used in our experiments: regular partitioning, fast community algorithm and random partitioning.

4.1 Regular partitioning

Regular partitioning divides the NK-landscape in regular patches, thus are patches of the same size. It is the only partitioning algorithm used in the experiments of Kauffman et al. in [4]. However this type of partitioning was defined only over a lattice NK-landscape, where square regular patches gives a result patches that maximize the number of intra-dependencies, thus are dependencies among elements in the same patch.

4.2 Modular partitioning

Using square patches is a very natural way to partition a network where spins are coupled spatially in a regular lattice. Inevitably, any partition will lead to a structure in which many spin couplings are ignored, due to all the couplings associated with spins on (or near) the edge of each patch.

To further explore the effect of modularity on the solution of NK-landscapes, we would also like to consider cases where the network has a more a more modular structure and it is possible to decompose the network into a structure where few couplings are broken by the partition, reflecting Simon's notion of nearly decomposable systems.

There has been considerable work in the area of finding community structures in networks, whether they are biological, communication or social networks. In this work, we use a community finding algorithm due to Newman[5]; which has been efficiently implemented in the freely available igraph library [2]. The idea behind this mechanism is to divide the network such that the modularity value, or difference between the number of inter and intra-partition edges, is greater than can be expected by chance. One advantage of this approach is that the size of partitions need not be set *a priori*, another advantage is its computational efficiency. We have however, noted that the algorithm sometimes struggles in k-regular regular graphs, producing results that do not correspond to intuitive partitioning.

4.3 Random partitioning

The random patch generator generates patches at random and it is used to compare if how we partition the problem matters or we can partition the problem randomly in different patches of certain size and we get the same performance.to the previously mentioned patching algorithms to determine if patching the problem using a more complex approach matters, or can patches be generated at random and get the same performance.

When the model is spatial we generate random partitions of the problem but consider the problem structure. We do this by using an spatial random partition solver that starts with all elements in one partition. At each iteration this algorithm creates a new partition by choosing at random one element of an existing partition and tracing a line in that point with a random gradient. Since the model is spatial, this line divides the existing patch in two new patches. Notice that with this method elements in the same patch have always an spatial relationship in the space. This algorithm also allows to specify the number of desired partitions.

5 Implementation

5.1 Implementation of the NK-landscape

In this section we explain some details of our implementation of the NK-landscape.

The NK-Landscapes used throughout some of these experiments have upwards of 14400 spins (N=120), with each spin having up to a maximum of 24 neighbors (k=24). Each of the 2^{24+1} possible neighborhood configurations for each spin is assigned a random value, and to store all of these values in a lookup table would require $14400 * 2^{25}$ entries. To remove the need for storing all of these values, the implementation used throughout this paper does not store a lookup table, rather it queries a pseudo-random number generator (RNG) for the energy value of a particular configuration for a spin.

To be more precise, each of the $144000 * 2^{25}$ entries can be obtained by mapping these values to an integer and then using that integer to seed a RNG. After the RNG has been seeded we simply query it for the next floating point number between zero and one.

To ensure that all of the entries are represented by a unique integer, a 64-bit bitmask is used as the RNG's seed. The $\lceil N \rceil$ least significant bits hold the spin's id (the id of a spin is a unique number in the range $[0, N-1]$), and the next K least significant bits are used to keep track of the states of that spin's neighbors. The remaining bits can be used to hold a run number so that different NK-Landscapes can be explored.

This method has two pitfalls. First of all, seeding the RNG each time we need to obtain the energy for a particular state is very slow. For efficiency, once the RNG has been seeded and the energy value has been obtained, it is entered into a hash table. The hash table is then queried with the bitmask before the RNG is seeded in order to reduce the number of RNG seedings. For large values of K, the hash table can get large, and so a limit must be put on its size. Another drawback of this method is that although most standard RNG classes (such as Java's) use a 64-bit seed, not all 64 bits are actually used. One must ensure that the bitmask they are using is working as intended by either using smaller values of N and K or by using a third-party RNG that uses all 64 bits of its seed.

6 Experimental results

We performed three major groups of experiments using this implementation of the NK-model. These results include:

- Validating our implementation by replicating the results reported in [4].
- Optimization in high-modularity NK models using two approaches: the purely modular approach of [4], in which each patch is optimized independently, over n iterations; and the hierarchical approach, which uses the modular approach for $n/2$ generations, followed by optimization over the whole network for $n/2$ partitions.
- Study of two small (15 and 16 element) instances of an NK model. For these models, it is possible to enumerate and compute all possible solutions.

6.1 Replicating previous results

To verify our implementation, we begin by reproducing the results reported in [4].

Specifically, the experiment compares the performance of regular partitioning when varying patch size on a toroidal spatial (lattice) NK-landscape with $N = 120 \times 120$ elements. The experiment is executed varying parameter K ($K = \{4, 8, 12, 24\}$), thus varying the level of coupling of the 120x120 lattice NK-landscape. The k neighbors of a particle i are chosen as the k closest particles in the lattice.

Figure 1 and Table 1 shows the final energy of the solution given by the random solver when varying the patch size. Results are the averages of 3 different instantiations of each 120x120 landscape for the same K. Error bars refer to the standard deviation over the 3 different runs. We run the random solver for 10 generations (a generation is an update of N*N spins).

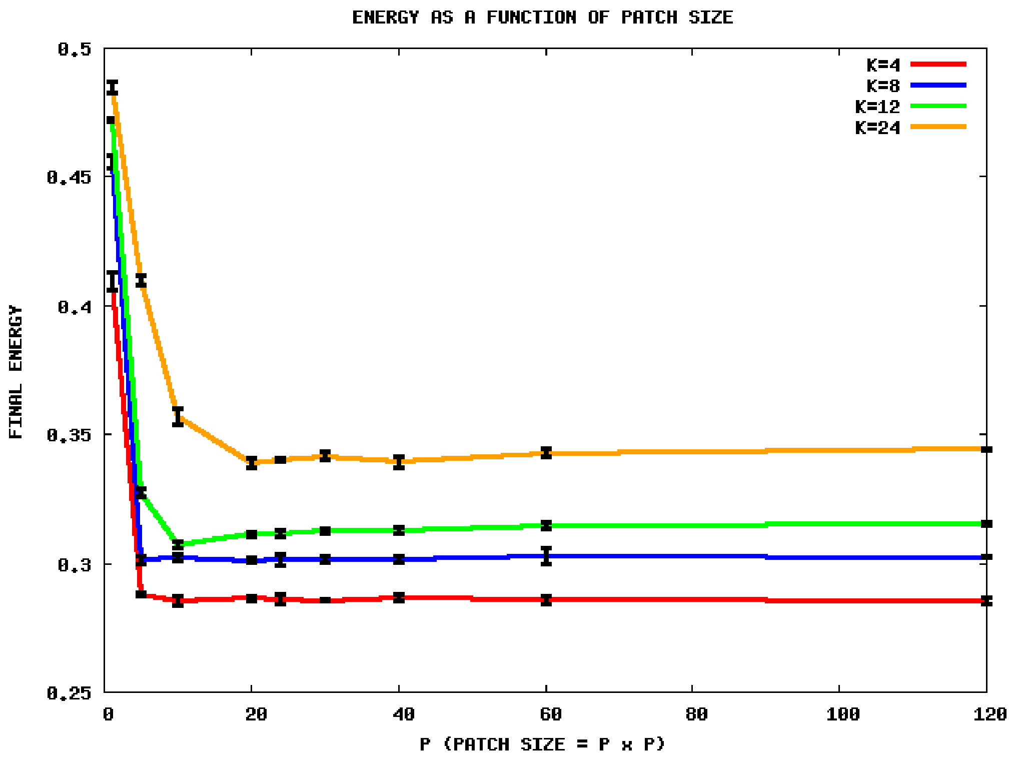


Figure 1: Reproducing the results reported in [4]. Energy vs patch size for different K.

Although we note that our current experimental results are on a more limited scale than [4], representing both fewer runs and fewer iterations, the results are very close to those reported in the original experiment.

6.2 Highly modular networks: pure modular

An obvious network in which to study the impact of a modular structure on the solution of an NK landscape is one that is "highly modular", that is, it can be partitioned such that the patches are nearly independent.

In order represent an NK landscape, such graphs must also be k-regular. The graph structure that we use is shown in figure 2: The values n and k are chosen such that n spins are evenly partitioned into patches of size $k + 1$. In each patch, every spin is coupled with to each of the k other spins in the patch. A small example of such a graph is shown in figure 2, for $n = 25$ and $k = 4$. Here, three of the five spins in each patch have no external spin couplings and two spins have three internal and one external. (Note that this pattern could also be used to generate networks with 2, 3, or more extra-module links, giving

Patch size K	**1**	**5**	**10**	**20**	**24**	**30**	**40**	**60**	**120**
4	0.4094(0.0034)	0.2878(0.0008)	0.2856(0.0019)	0.2863(0.0011)	0.2858(0.0018)	0.2856(0.0002)	0.2867(0.0012)	0.2858(0.0017)	0.2854(0.0013)
8	0.4560(0.0024)	0.3013(0.0016)	0.3021(0.0010)	0.3011(0.0009)	0.3013(0.0021)	0.3016(0.0012)	0.3016(0.0010)	0.3030(0.0031)	0.3023(0.0002)
12	0.4721(0.0005)	0.3273(0.0014)	0.3071(0.0013)	0.3115(0.0009)	0.3115(0.0011)	0.3125(0.0009)	0.3128(0.0010)	0.3146(0.0011)	0.3152(0.0008)
24	0.4848(0.0021)	0.4099(0.0018)	0.3570(0.0030)	0.3391(0.0018)	0.3403(0.0005)	0.3415(0.0014)	0.3392(0.0020)	0.3429(0.0013)	0.3444(0.0002)

Table 1: Reproducing the results reported in [4]. Energy vs patch size for different K (standard deviation in parenthesis).

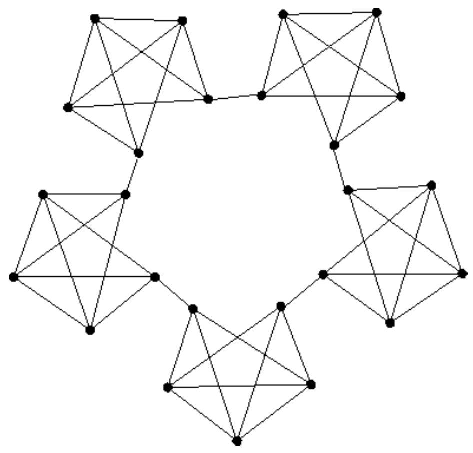


Figure 2: A "highly modular" NK-landscape: N=25, K=4.

us progressive increase in modularity.)

Note that as k increases, each patch has only two external couplings. In other words, the proportion of extra-modular spin couplings decreases with increasing k. For these graphs, the structure of the graph is very clear and we do not need to determine the partitioning algorithmically.

Results shown in Figures 3 (a)(b) and 4(a)(b) are the total energy that each partition get after running 50 generations for different initial configurations of one NK-landscape (a generation refers to an update of all elements of the system, thus a total of N updates). The high-modular NK-landscape used has a N=900 and a parameter K that we vary between $K = \{3, 8, 14, 24\}$.

As we observe with small K the single partition has always better results but as K increases this is not longer true. For K=24 for example we get better average results with the fast algorithm partitioning than with the non partitioning.

6.3 Highly modular networks: hierarchical

How to partition the problem is only the first approach in using partitions. We can also define a hierarchy of partitions, a dendogram that defines after a number of generations which partitions we recombine in order to get another one. In this section we introduce experiments to test this hierarchy with recombination at each level.

This experiment compares the performance of using the fast community algorithm partitions with a single level of hierarchy. Thereafter, we run one half of generations with the partitions defined by the fast community algorithm introduced in section 4 and the other half of iterations with no partitioning (these all partitions are recombined in the same level). Hence the best configuration obtained using fast community algorithm algorithm partitioning is the initial condition set when recombining them.

Results shown in Figures 5 (a)(b) and 6(a)(b) are the total energy that each partition get after running 50 generations for different initial configurations of one NK-landscape (a generation refers to an update of all elements of the system, thus a total of N updates). The high-modular NK-landscape used has a N=900 and a parameter K that we vary between $K = \{3, 8, 14, 24\}$.

We can observe that with recombination the fast community algorithm get better results also with small K ($K = 4$).

6.4 Experiments with small examples

When N is small, all of the possible spin configurations can be enumerated. From each configuration we can do a minimization. The following experiments do this for a four-by-four toroidal grid and a modular network containing fifteen spins. For each of the two topologies, ten landscapes will be used, and with each of the landscapes, thirty minimizations will be conducted. The experiments will be used to investigate how minimizing with small patches and then combining those patches and minimizing again affects the overall minimization of the system.

Four-by-four example

With a four-by-four toroidal landscape, there are $2^{16} = 65536$ possible state configurations. Each spin only looks at its immediate neighbors when calculating its current energy.

The patches can be two-by-two squares, or two-by-four rectangles, or the full four-by-four grid. The experiments show six different ways to do patching on a four-by-four landscape: one four-by-four patch, four two-by-two patches, two two-by-four patches, two-by-two then one patch, two-by-four then one patch, two-by-two then two-by-four then one patch. For the last three experiments, the landscape is minimized using Kauffman's technique with the smaller patches. After that minimization is complete, the patch sizes are increased, and the minimization occurs again.

Table 2 shows the average amount of minimization that was done for each landscape and patching scheme. (The standard deviations of those averages

	4x4	2x2	2x4	2x2,4x4	2x4,4x4	2x2,2x4,4x4
1	0.191594	0.144709	0.168021	0.199145	0.201846	0.203826
2	0.192644	0.130890	0.165889	0.198108	0.200267	0.203162
3	0.185488	0.142352	0.173000	0.201660	0.200295	0.210427
4	0.207870	0.138629	0.184448	0.216184	0.222096	0.224965
5	0.192797	0.132923	0.171504	0.200215	0.204660	0.208792
6	0.188941	0.139611	0.170073	0.201766	0.200341	0.208864
7	0.190952	0.133361	0.170914	0.203251	0.208663	0.213565
8	0.195765	0.141773	0.176415	0.205496	0.211436	0.215508
9	0.176311	0.116055	0.157238	0.179297	0.187249	0.186483
10	0.187584	0.124285	0.167068	0.194806	0.198512	0.202516
avg.	0.190995	0.134459	0.170457	0.199993	0.203537	0.207811

Table 2: Ten different landscapes on a four-by-four toroidal grid. The top row shows the types of patches used. The values indicate the average difference between the initial configuration and configuration obtained after minimization. These averages are obtained over all 65536 initial configurations and 30 different minimizations. The bottom row gives the average of each column.

are shown in Table 3.) To obtain a value in this table, all 65536 spin-state configurations of the grid are used as initial starting points for the minimization. The difference between the initial configuration and the configuration after the minimization are obtained for all 65536 starting points. The minimization for each initial configuration was done thirty times. These values are then averaged to obtain a single data point in Table 2. The values on the bottom row show the average for each column. Note, larger values indicate better performance as this implies the average minimization was larger.

The results indicate that the patching does increase the amount of minimization. Also, a larger decrease in energy is noticed when the patching is done multiple times.

Modular graph with 15 spins

A modular 15 spin network is used to examine the effects of patching on highly modular networks. This network consists of three, five spin graphs. Each of these three networks is fully connected, except for one link. The two spins that are not connected are each connected to one of the other two graphs. Each spin has an equal number of neighbors. The three patches consist of each of the three clusters in the network. Table 4 shows the data obtained using this network topology. The data is aggregated in the same way the data from Tables 2 and 3 were. When only three three-by-five patches are used, the minimization does not get as low as when no patches are used. When the three-by-five patches are used, and then minimization is done on the full network, the total energy is always less than when no patches are used. Recall that the data is showing the amount of minimization that occurred, so larger numbers are better.

	4x4	2x2	2x4	2x2,4x4	2x4,4x4	2x2,2x4,4x4
1	0.000133	0.000211	0.000170	0.000127	0.000103	0.000099
2	0.000123	0.000221	0.000197	0.000136	0.000123	0.000131
3	0.000191	0.000298	0.000188	0.000116	0.000135	0.000128
4	0.000134	0.000203	0.000233	0.000147	0.000153	0.000145
5	0.000119	0.000165	0.000166	0.000162	0.000154	0.000124
6	0.000141	0.000231	0.000176	0.000146	0.000157	0.000140
7	0.000136	0.000315	0.000184	0.000161	0.000169	0.000203
8	0.000173	0.000332	0.000260	0.000205	0.000187	0.000185
9	0.000159	0.000220	0.000181	0.000141	0.000123	0.000123
10	0.000137	0.000266	0.000117	0.000117	0.000095	0.000105
avg.	0.007998	0.008965	0.007088	0.009241	0.009184	0.010088

Table 3: The standard deviations of the values given in Table 2.

	1x15	3x5	3x5,1x15	1x15	3x5	3x5,1x15
1	0.170101	0.160546	0.174780	0.000178	0.000160	0.000125
2	0.178729	0.163231	0.182775	0.000164	0.000128	0.000160
3	0.182321	0.171179	0.187398	0.000263	0.000218	0.000210
4	0.209254	0.187871	0.212562	0.000284	0.000382	0.000303
5	0.168167	0.163774	0.179939	0.000193	0.000249	0.000144
6	0.191955	0.183375	0.199777	0.000229	0.000234	0.000199
7	0.158233	0.152263	0.165279	0.000118	0.000182	0.000170
8	0.203789	0.180371	0.207699	0.000245	0.000270	0.000216
9	0.184925	0.185193	0.200719	0.000238	0.000288	0.000242
10	0.184904	0.168550	0.189469	0.000189	0.000186	0.000197
avg.	0.183238	0.171635	0.190040	0.015730	0.012028	0.015041

Table 4: Ten different landscapes on a modular, 15 spin, network. The top row shows the types of patches used. The values indicate the average difference between the initial configuration and configuration obtained after minimization. These averages are obtained over all initial configurations and 30 different minimizations. The left columns show the averages, and the three right-most columns show the standard deviations of those averages. The bottom row gives the average of each column (or the standard deviation of that average in the three right-most columns).

7 Conclusions

We have studied the effectiveness of modularization and hierarchy in solving the combinatorial problem of minimizing energy on an NK landscape, extending the results of [4] to the case of non-lattice networks. Our results are similar to previous results: while modularization and hierarchy can find lower energy solutions, the differences are small and are not necessarily statistically significant.

Acknowledgments

This work was partially supported by the Santa Fe Institute whose research and education programs are supported by core funding from the National Science Foundation and by gifts and grants from individuals, corporations, other foundations, and members of the Institute's Business Network for Complex Systems Research.

References

[1] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70:066111, 2004.

[2] G. Csardi and T. Nepusz. The igraph library for complex network research, 2008. version 0.5.

[3] S. Kauffman and S. Levin. Towards a general theory of adaptive walks on rugged landscapes. *J Theor Biol*, 128:11–15, 1987.

[4] S. Kauffman, W. Macready, and E. Dickinson. Divide to coordinate: Coevolutionary problem solving. Working Paper No. 94-06-031, The Santa Fe Institute, 1994.

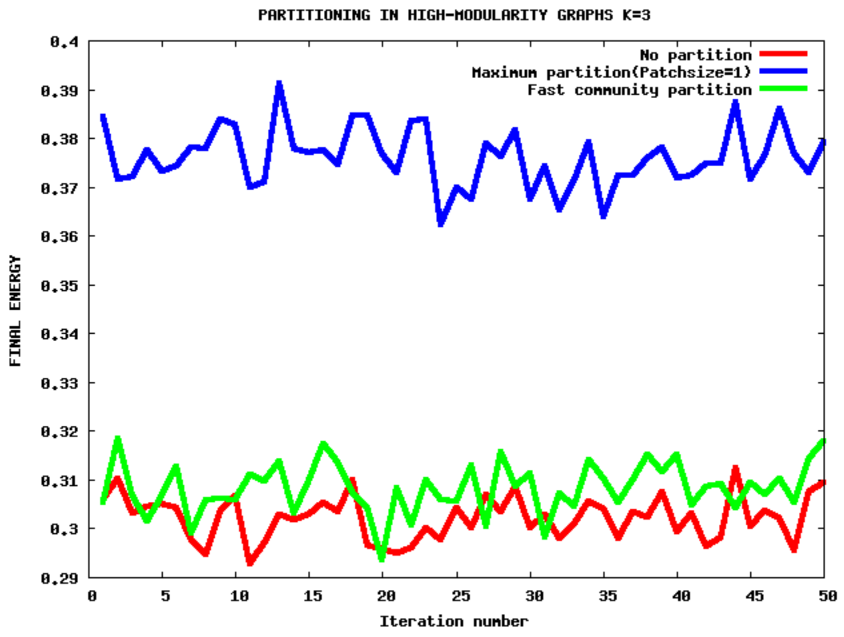
[5] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 74(3), 2006.

[6] R.Glauber, 1963.

[7] M. Rosvall and C. T. Bergstrom. An information-theoretic framework for resolving community structure in complex networks. *PNAS*, 104(18):7327–7331, May 2007.

[8] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, pages 0706851105+, January 2008.

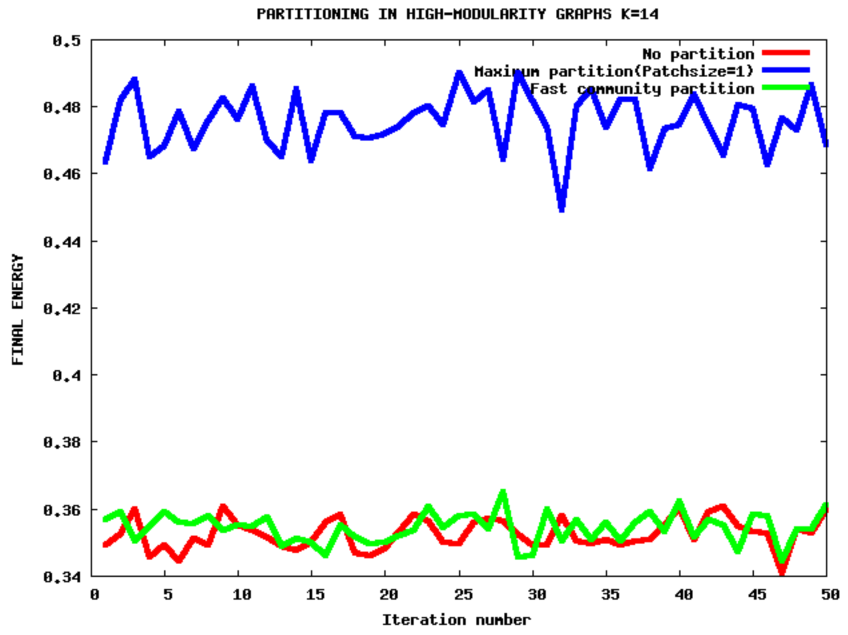
[9] H. A. Simon. *The Sciences of the Artificial - 3rd Edition*. The MIT Press, October 1996.



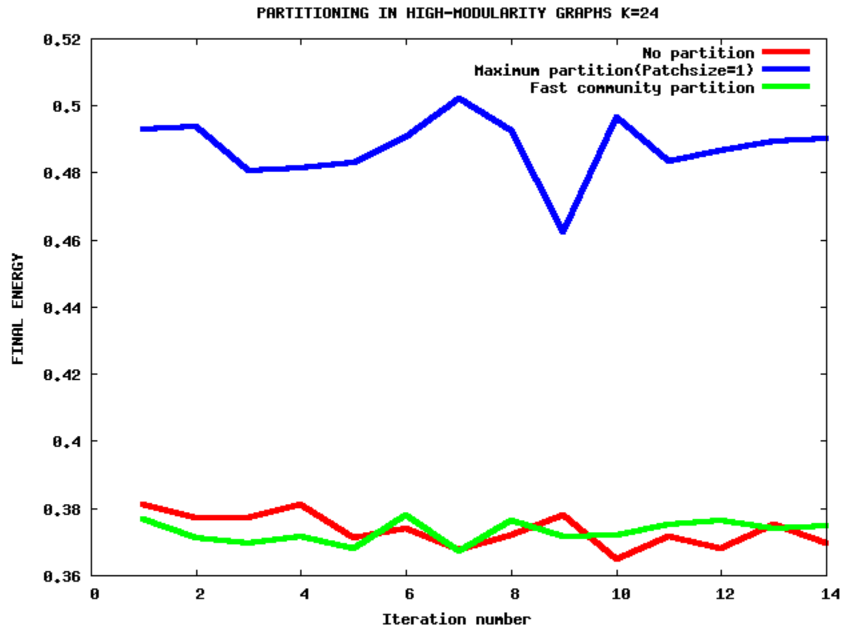
(a) Comparison of performance using different partitions in high-modularity graphs with $N = 900$ and $K = 3$

(b) Comparison of performance using different partitions in high-modularity graphs with $N = 900$ and $K = 8$

Figure 3: Partitioning experiment in high-modular graph with K=3 and K=8

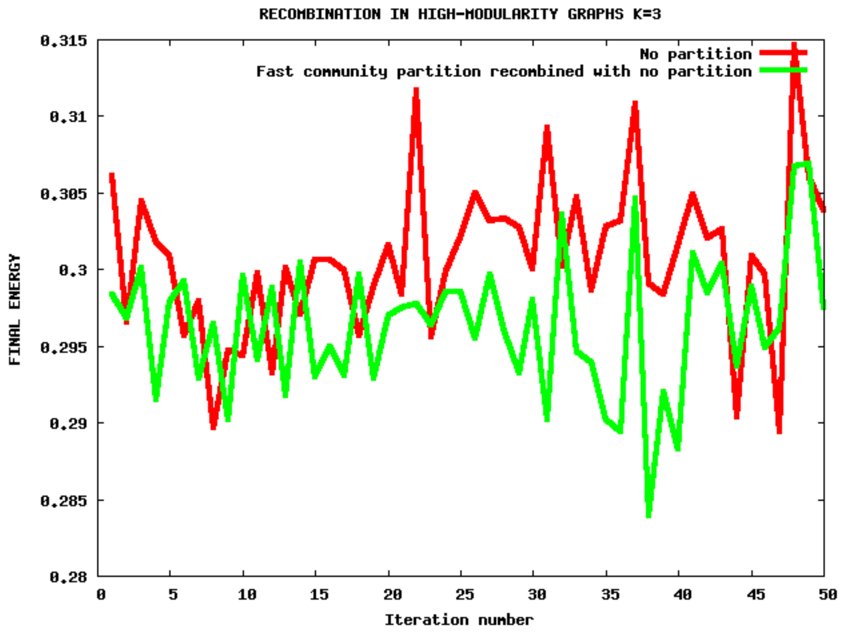


(a) Comparison of performance using different partitions in high-modularity graphs with $N = 900$ and $K = 14$

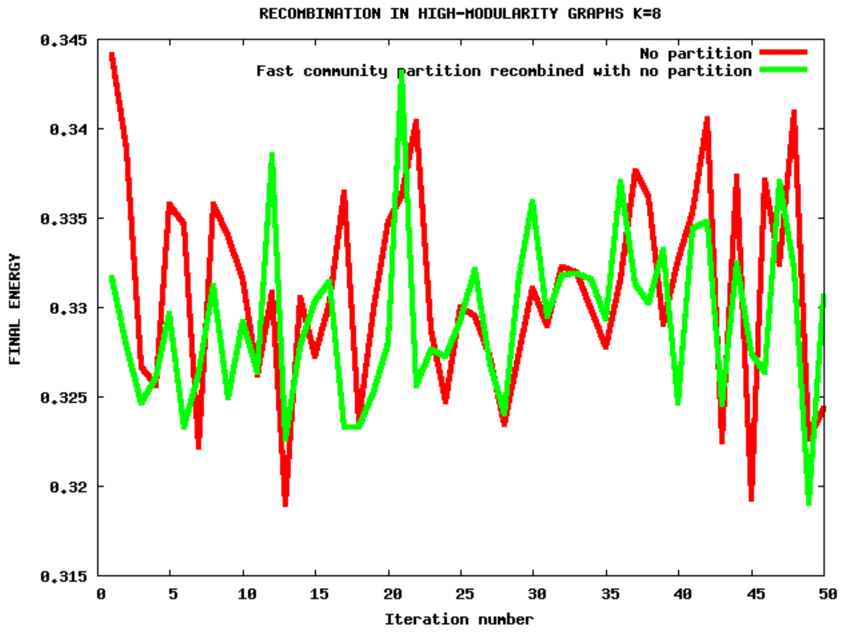


(b) Comparison of performance using different partitions in high-modularity graphs with $N = 900$ and $K = 24$

Figure 4: Partitioning experiment in high-modular graph with K=14 and K=24

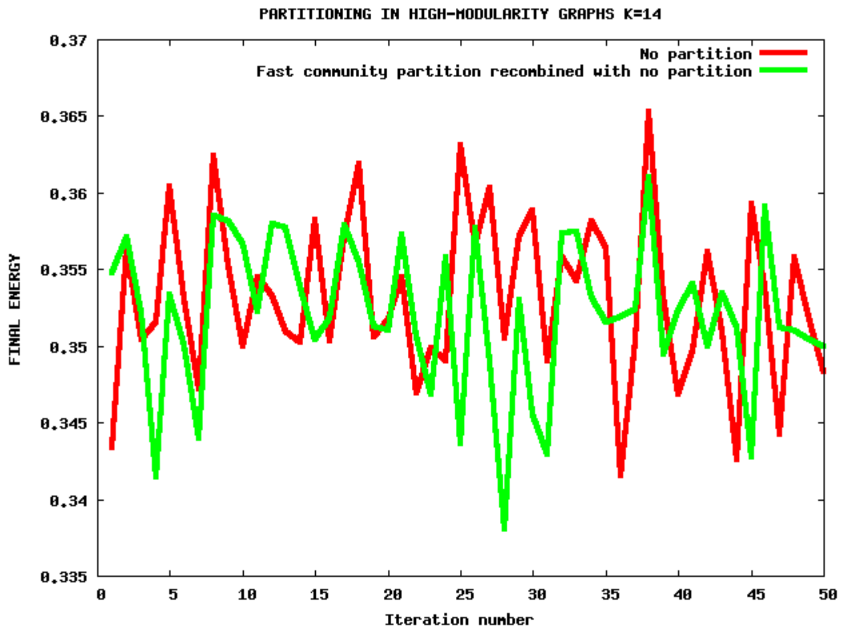


(a) Comparison of performance using fast community algorithm with recombination in high-modularity graphs with $N = 900$ and $K = 3$

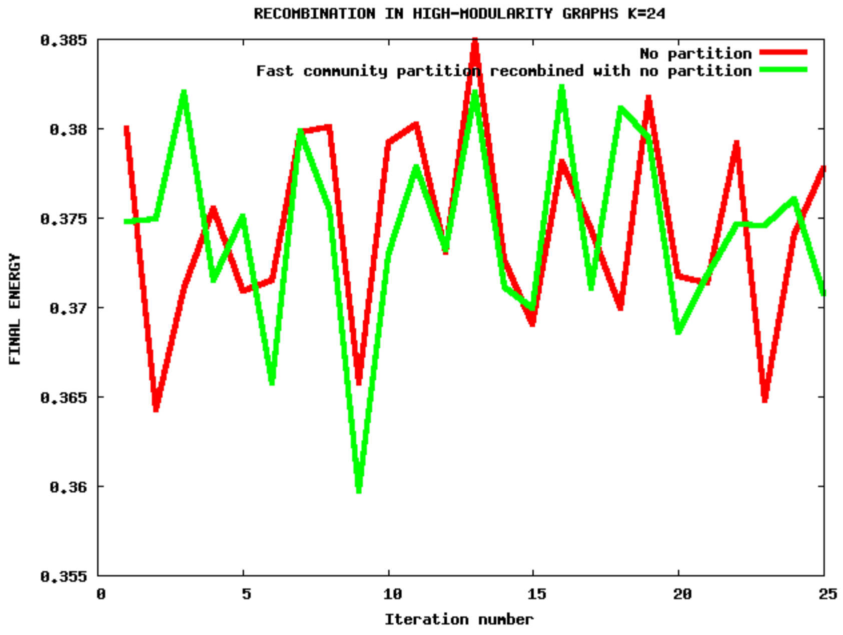


(b) Comparison of performance using fast community algorithm with recombination in high-modularity graphs with $N = 900$ and $K = 8$

Figure 5: Hierarchical experiment in high-modular graph with K=3 and K=8



(a) Comparison of performance using fast community algorithm with recombination in high-modularity graphs with $N = 900$ and $K = 14$



(b) Comparison of performance using fast community algorithm with recombination in high-modularity graphs with $N = 900$ and $K = 24$

Figure 6: Hierarchical experiment in high-modular graph with K=14 and K=24