

Towards an analysis of local optimization algorithms

Tassos Dimitriou¹
Department of Computer Science
University of California
San Diego, CA 92093-0114
tassos@cs.ucsd.edu

Russell Impagliazzo¹
Department of Computer Science
University of California
San Diego, CA 92093-0114
russell@cs.ucsd.edu

Abstract

We introduce a variant of Aldous and Vazirani's "Go with the winners" algorithm that can be used for search graphs that are not trees. We analyze the algorithm in terms of the properties of a tree-decomposition of the search graph. We show a large class of distributions for search graphs so that "Go with the winners" works well with high probability for almost all graphs from the distribution. We also give a sufficient combinatorial property that ensures good performance.

1 Introduction

Many commonly used optimization methods are based on metaphors to either physical phenomena such as annealing, or biological processes such as "survival of the fittest". In addition to the appeal of using methods similar to those that make the "real world" work quite impressively, these search methods have the intuitive justification of combining a greedy approach, which attempts to find better solutions by making small changes, with randomization to prevent the algorithm from getting into a "mental fix".

These algorithms have been widely used and tested. Intuitively the combination of randomness and local optimization would seem to give them some advantage over either alone. However, their efficiency and effectiveness have never been analyzed in a satisfactory manner. In

¹Research Supported by NSF YI Award CCR-92-570979, Sloan Research Fellowship BR-3311, grant #93025 of the joint US-Czechoslovak Science and Technology Program, and USA-Israel BSF Grant 92-00043.

particular, very little is known about the types of problems for which they perform well, the expected trade-off between optimality of the solution found and time spent searching, or how to set the various parameters optimally.

"Go with the winners", introduced by Aldous and Vazirani, is loosely based on an asexual (no crossover) version of genetic algorithms. Many particles randomly search the solution space in parallel, trying to find better and better solutions. Successful particles "reproduce", but unsuccessful particles "die out". Their algorithm was described only for trees, and was intended to be used in combination with another search procedure such as simulated annealing.

We present a modification of their algorithm which can be directly used for general search graphs. It seems simpler than algorithms that are more "scientifically correct", and easier to analyze, but with many of the same benefits for optimization. We show that, for any one of a large class of distributions on search graphs, our version of "Go with the winners" finds an optimal valued node in polynomial time (in the logarithm of the number of nodes) with high probability for almost all search graphs drawn from the distribution. We also show that a certain combinatorial property, the existence of a tree decomposition with good expansion between layers, suffices for the algorithm to work well. Our results help give combinatorial insight into which search problems might be susceptible to a randomized search algorithm, and suggest a simplification of these algorithms, which, while less appealing in its connections to the physical world, might be equally effective as an optimization heuristic.

2 Go with the winners strategy

Let Π be a maximization problem. A search graph G_Π for an instance of Π has nodes which represent possible solutions to Π . Two nodes are connected by an edge if

and only if one results from a local change performed to the solution represented by the other. Furthermore, there is a payoff function f defined on G_Π , i.e. if u is a node of G_Π then $f(u)$ is the value of the solution represented by u .

From now on we will consider only problems whose search graphs are layered. That is if u and v are neighbors in G_Π , then the values of their solutions satisfy $|f(u) - f(v)| = 1$ (maximum matchings belong to this category). Later, in Section 7, we will see how to extend our results to include arbitrary search graphs.

Many heuristics for optimization (local optimization, metropolis, simulated annealing, genetic algorithms without crossing) can be viewed as moving particles in such a search graph. These heuristics are all (possibly probabilistic) strategies for moving particles to neighboring solutions or jumping particles to previously visited solutions. Moves are based on the values of explored solutions and the goal is to find a node of large value. However, very little is known about the time analysis of such heuristics.

Here we analyze a “Go with the winners” strategy. [AV] proposed “Go with the winners” as a modification for simulated annealing. They modelled the state space as a tree where a particle, representing some randomized algorithm, is moving down from the root to a leaf of the tree, choosing its path according to the probabilities the algorithm assigns to each node of the tree. Since the “cooling schedule” of finite simulated annealing can be seen as a rule for taking a walk in a similar tree, the tree model captures the effect of lowering the temperature.

By considering many particles instead of one and by introducing interactions between those particles, they were able to give an algorithm to find deep vertices on trees with running time polynomial in h and κ , where h is the height of the tree and κ is a measure of the imbalance of the tree. The type of interaction was simply to move each particle that is at a leaf to the position of a randomly chosen particle that is not at a leaf. More precisely,

“Go with the winners” algorithm for trees

- Stage 0: Start at the root of the tree with B particles. Move each particle to a child chosen at random according to the transition probabilities.
- Stage i : Each of the particles is at depth i . If all particles are at leaves, stop. Otherwise, move each leaf particle to the position of a randomly chosen non-leaf particle. Then move each particle to a child chosen at random and proceed with the next stage.

Given a search graph G_Π for some maximization problem Π we reduce the problem of finding a good solution for Π to that of finding a deep vertex in a tree \mathcal{T}_Π defined as follows. We let the nodes of \mathcal{T}_Π at depth d to be all connected components S of G_Π such that if u is a node in S then $f(u) \geq d$, where f is the payoff function associated with Π .

To apply “Go with the winners” to a problem we don’t need to construct \mathcal{T}_Π explicitly (search graphs G_Π are usually exponentially large and constructing \mathcal{T}_Π would require at least exponential time). Instead we assume it is possible to do the following in polynomial time: generate a random solution; given a solution x , compute its value $f(x)$; and given a solution x generate a list of all its neighbors. We also assume that solutions are known to have values between 1 and h and that a value i_0 with $Prob_x[f(x) = i_0] \geq 1/h$ is known. As before, we work in stages maintaining the following invariant: At the beginning of stage i every particle x is at level i , i.e. $f(x) = i$. The stages are very similar to the ones defined above:

“Go with the winners” algorithm for layered search graphs

- Stage $i_0 - 1$: Generate random solutions until B solutions of value i_0 are found. Place particles on each of these solutions. (This takes an expected Bh uniform samplings and evaluations.)
- Stage i : We proceed in two phases. In the *redistribution* phase, we first compute for each particle x the *down degree* d_x , which is the number of neighbors y at the next level, i.e. $f(y) = f(x) + 1 = i + 1$. If all particles have down degree zero, i.e. they are all in local maxima of f , then stop. Otherwise, create B new particles and place each one, independently, at a particle x from the previous stage with probability $d_x / \sum_y d_y$. In the *randomization* phase, each new particle x is moved to a neighbor of value $i + 1$ chosen at random with probability $1/d_x$.

Go to the next stage.

The intuition for the above algorithm is that in the i -th stage and beyond, we are implicitly deleting all nodes of value less than i from the graph. This divides the graph into components. We want to have all components receive particles in proportion to their size. The redistribution phase will make sure of two things. First, particles that are at leaves are distributed among non-leaf particles since their down degree is zero. Second, each component receives particles in proportion to the

number of nodes in all child components of value $i + 1$. Then the randomization phase will make sure that each component receives particles in proportion to its size.

The problem now is to derive a minimal set of conditions so that every problem Π whose search graph G_Π satisfies this set of conditions can be “easily solved” with high probability using the “Go with the winners” scheme. Our goal in the rest of the paper is to give some such set of conditions although probably not minimal. In the next section we define the tree corresponding to a search graph and give a set of assumptions that we make for the rest of the paper. In Section 4 we perform a probabilistic analysis of “Go with the winners” and prove that the algorithm works well for almost all search graphs with the above properties. In Section 5 we add an additional combinatorial property, namely that every bipartite graph between a component and its children has good expansion, which suffices for the algorithm to work. In Section 6 we present an example of a real problem that has the previous property and prove that “Go with the winners” works well. Finally, in Section 7 we extend our results to nonlayered search graphs.

3 Assumptions

We now formally describe the assumptions we make about the search graphs. The values of solutions give us as much information as the connections between them, so we will want to look at a representation of the graph that also gives us information about the structure of the values. Such a representation is a *tree decomposition* of the graph, where every solution of value i is assigned to a node of depth $i - i_0$ in a tree representing “neighborhoods” of related solutions, and all edges are between solutions at a node and its parent. Intuitively, one would get a tree-decomposition as follows:

We only look at those nodes of value i_0 or greater. This separates the graph into different components and we consider the subset of each component of value i_0 as the root of a tree in a forest. For each such component we consider how it breaks up after the nodes in the root are deleted. The subsets of these components of value $i_0 + 1$ are the children of the root. We proceed recursively which gives us a tree decomposition of each component where the nodes of value i are at depth $i - i_0$ of the tree.

However, this has the drawback of making each local minimum, a solution with no better neighbors, a separate node of the tree. For non-trivial problems this makes the tree exponentially large. We prefer to allow local minima to be put in with other “neighborhoods” arbitrarily. So we do not assume that the tree-

decomposition used to describe the graph was obtained following the intuitive approach, but instead allow any tree-decomposition with certain combinatorial properties. Since both the search graph and the values can be reconstructed from this tree decomposition we list the assumptions that we make about the search graphs in terms of this tree decomposition.

We let T be the underlying tree for a tree decomposition where each node C of T is labelled by the set of solutions from the search graph that belong to C . We call this set the *layer* associated with C . We also let \mathcal{T} be the entire tree decomposition. In other words \mathcal{T} labels each edge of T with the induced bipartite graphs between the layers of its endpoints.

We make the following assumptions about the component tree T . Denote by n_d the number of solutions of value d .

1. Let C be a layer at level d of T . Then $|C| > n_d/p(n)$ for some polynomial $p(n)$. (In particular this means that the number of layers is bounded by a polynomial and that two layers C, C' have sizes that are within a polynomial of each other.)
2. If a layer C is at depth d , then every solution in C has $e(d)$ neighbors in the parent layer, where $e(d) \geq 3$.
3. Let D be a child of layer C . Then $|C|/\sum_D |D| > 1$.
4. Consider the number of solutions at levels $d-1$ and d of T . Then their ratio satisfies: $\frac{n_{d-1}}{n_d} < e(d)$.

Condition 1 seems to be necessary since if it wasn't true we would miss some components entirely. We can probably weaken condition 2 considerably but that would make the arguments more complicated. We need condition 3 in the probabilistic analysis of the algorithm to get an upper bound on the down degree of each particle. In a real problem Π the degree of each solution is usually some polynomial on the size of the input, so this condition doesn't constitute a real restriction (for the approach we take in Section 7 this condition will always be true). If condition 4 is not at least approximately true, i.e. $n_{d-1} \gg e(d)n_d$, then almost all solutions would be local maxima which would be bad for any local search algorithm. Later, in Section 5 we will see that $e(d)$ can be replaced by any polynomial in the size of the input provided the graph has certain combinatorial properties.

4 Probabilistic analysis

In this section we prove:

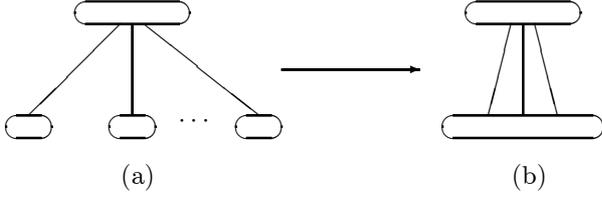


Figure 1: Lines correspond to sets of random edges between the layers. A collection of bipartite graphs in (a) is equivalent to a single bipartite graph in (b).

Theorem 1 *Let $e(d)$ be any given function with $e(d) \geq 3$, and let T be a tree with properties 1, 2 and 4. Let \mathcal{T} be a uniformly chosen labelling of T satisfying 3, i.e. for each solution in a given layer at depth d we pick $e(d)$ random neighbors in the parent layer. Then the modified “Go with the winners” algorithm, with $B = O(\hat{d}^2 h^8 p^2(n) \ln h)$ particles, where $\hat{d} = \max_d e(d) + h$ is the maximum down degree of each particle, succeeds in reaching depth h after $O(hB)$ steps, with high probability (over both the choice of \mathcal{T} and the random moves of the algorithm).*

One thing to notice is that we can view a layer of the tree together with its children as a single bipartite graph since it is the union of a collection of random bipartite graphs (Figure 1).

It will be helpful in the analysis to imagine the tree \mathcal{T} as being constructed one level at a time. Thus during the first i stages of the algorithm only the first i levels of the tree have been revealed while the rest remain hidden. It is only *after* the end of stage i that the children of a layer in level i will be revealed. All we see before this uncovering is a single bipartite graph between a component and the union of its children. This is fine because the algorithm works in stages and particles never exceed depth i during the i -th stage of the algorithm.

So, intuitively, the analysis argues that the algorithm maintains the following invariant: the B particles are independently and uniformly distributed among the solutions of value i . In particular B is large, so all components have many particles. However, the particles are not quite independent due to a process we call “clustering”, where many particles are placed in the same solution. If the particles, while advancing to deeper levels of \mathcal{T} , start forming clusters and the size of these clusters tends to become unbounded then the algorithm will be no better than a simple random walk. We will bound the size of clusters in Lemma 4. Let $\kappa_i(x)$ be the number of particles at the same solution as x at stage i of the algorithm.

4.1 Analysis

In the rest of this section we will prove the main lemma, which shows that particles remain distributed among the layers at level i almost in proportion to their sizes. Denote by N_C and M_C the number of particles in layer C before and after the redistribution phase respectively, and by $\text{Child}(C)$ the set of children of layer C . Let $r_i = e^{i/h^2}$, $\alpha = 1/(6h^2 + 1)$, $\kappa_i = ic^2h^2$, $\delta_i = 2ie^{-B/c\hat{d}^2h^8p^2(n)}$ for some small constant $c > 1$.

Lemma 1 *At the beginning of stage i , with probability at least $1 - \delta_i$, no cluster has size greater than κ_i and for any two layers C and C' at the i -th level of \mathcal{T} ,*

$$\frac{1}{r_i} \frac{|C|}{|C'|} \leq \frac{N_C}{N_{C'}} \leq r_i \frac{|C|}{|C'|} \quad (1)$$

Proof: The proof is by induction on i . Initially, all particles are uniformly distributed to solutions of level i_0 , so with high probability, all layers get roughly their share of particles, and all clusters are single particles. Assume the lemma is true for stage $i > i_0$. We first prove that after the redistribution phase, all layers have particles in proportion to the sum of the sizes of their children (Lemma 2). Then we show that in the randomization phase of stage i , the number of particles in any two layers D and D' at the $i + 1$ -th level of \mathcal{T} is proportional to their sizes (Lemma 5). Lemma 1 will then follow from combining the two and a simple calculation. \square

Corollary 1 *With probability at least $1 - \delta_i$*

$$\frac{B}{r_i} \frac{|C|}{\sum_{C'} |C'|} \leq N_C \leq r_i B \frac{|C|}{\sum_{C'} |C'|},$$

where the sum is taken over all layers C' at level i .

Proof: We just use Lemma 1 and note that $B = \sum_{C'} N_{C'}$. \square

We now give a series of lemmas that will be used to prove Lemma 1.

Lemma 2 *Assume at the beginning of stage i , each layer C at level i has N_C particles in it and no cluster has more than κ_i particles. At the end of the redistribution phase, with probability at least $1 - p(n)e^{-\alpha^2 B/c\hat{d}^2h^4p(n)}$, the number of particles in any two layers C and C' will satisfy*

$$\frac{1}{r_i} \left(\frac{1 - \alpha}{1 + \alpha} \right)^2 \frac{\sum_{D \in \text{Child}(C)} |D|}{\sum_{D' \in \text{Child}(C')} |D'|} \leq \frac{M_C}{M_{C'}} \leq r_i \left(\frac{1 + \alpha}{1 - \alpha} \right)^2 \frac{\sum_{D \in \text{Child}(C)} |D|}{\sum_{D' \in \text{Child}(C')} |D'|}$$

where $\hat{d} = e \max_i e(i) + h$ and c some constant greater than 1.

Proof: Consider a particle $x \in C$ before the redistribution phase of stage i . In order to get an estimate on M_C , the number of particles in C after the redistribution phase, we must first get a bound on $\sum_{x \in C} d_x$ since C will receive particles in proportion to this number.

If there weren't any clusters this would be the sum of N_C independent random variables each with mean equal to the average down degree of layer C , $\mu_C = e(i+1) \sum_{D \in \text{Child}(C)} |D|/|C|$, since the i -th level of the tree has not been revealed yet. What is true however is that even though particles form clusters, these clusters will still be independently distributed random variables with values in $[1, \dots, \hat{d}\kappa_i]$. Using Chernoff bounds,

$$\begin{aligned} \Pr\left[\left|\sum_{x \in C} d_x - N_C \mu_C\right| > \alpha N_C \mu_C\right] &< e^{-(\alpha N_C \mu_C / \hat{d} \kappa_i)^2 / N_C} \\ &< e^{-\alpha^2 N_C / \hat{d}^2 \kappa_i^2} \end{aligned}$$

where \hat{d} is the maximum down degree of any particle in C . In order to get an estimate on this probability we must upper bound both \hat{d} and κ_i . We do this in Lemmas 3 and 4. Using Corollary 1 to lower bound N_C by $B/r_i p(n)$ and summing over at most $p(n)$ layers in level i , this probability is at most

$$p(n) e^{-\alpha^2 B / r_i i^2 e^4 h^4 \hat{d}^2 p(n)}.$$

After redistribution, the new number of particles in any layer C is given by

$$M_C = (1 \pm \alpha) B \frac{\sum_{x \in C} d_x}{\sum_y d_y} \quad (2)$$

with probability $1 - p(n) e^{-(\alpha B \sum_{x \in C} d_x / \sum_y d_y)^2 / B}$, where the $p(n)$ factor comes because we are considering all layers at level i . Lower bounding $\sum_{x \in C} d_x / \sum_y d_y$ by $1/12p(n)$, the probability is at least $1 - p(n) e^{-\alpha^2 B / 12p(n)}$. Using (2) the ratio of particles after redistribution of any two layers C and C' will satisfy:

$$\begin{aligned} \frac{M_C}{M_{C'}} &\leq \frac{1 + \alpha \sum_{x \in C} d_x}{1 - \alpha \sum_{x \in C'} d_x} \\ &\leq \left(\frac{1 + \alpha}{1 - \alpha}\right)^2 \frac{N_C \mu_C}{N_{C'} \mu_{C'}} \\ &\leq r_i \left(\frac{1 + \alpha}{1 - \alpha}\right)^2 \frac{\sum_{D \in \text{Child}(C)} |D|}{\sum_{D' \in \text{Child}(C')} |D'|} \end{aligned}$$

To obtain the lower bound on $M_C/M_{C'}$ we work similarly. The total success probability of this phase will be at least $1 - p(n) e^{-\alpha^2 B / c \hat{d}^2 h^4 p(n)}$ for some constant $c > 1$. \square

Lemma 3 *Let \hat{d} be the maximum down degree of any particle in C . With probability at least $1 - B e^{-h}$, $\hat{d} \leq e \max_i e(i) + h$.*

Proof: Since clusters are independently distributed, the down degree of each cluster in C will be a Poisson distributed random variable with mean equal to μ_C , the average down degree of C . Therefore, the probability that the down degree has value k is given by $\mu_C^k / k! e^{-\mu_C} < (e\mu_C/k)^k e^{-\mu_C}$. Choosing $k = e\mu_C + h$ and summing over all B particles this probability becomes exponentially small, at most $B e^{-h}$. Since $\mu_C < e(i+1)$ (Condition 3 of Section 3), the maximum down degree \hat{d} is at most $e \max_i e(i) + h$. (In a real combinatorial optimization problem Π the number of the neighbors of each solution is usually some polynomial on the size of the input. Then we can set \hat{d} to be that polynomial and Condition 3 is no longer necessary.) \square

Lemma 4 *With probability at least $1 - i B e^{-h/4e^3}$, $\kappa_i \leq i e^2 h^2$.*

Proof: Let d_x be the down degree of particle x . Recall that $\kappa_i(x)$ is the number of particles at the same solution as x . After redistribution, we expect $B d_x \kappa_i(x) / \sum_y d_y$ particles to be placed on x . By computing the ratio $B / \sum_y d_y$, we can estimate the number of particles each of the d_x neighbors of x will get. Using Corollary 1 we find that

$$\begin{aligned} \sum_y d_y &= \sum_C \sum_{x \in C} d_x \\ &\geq (1 - \alpha) \sum_C N_C \mu_C \\ &\geq (1 - \alpha) \sum_C \frac{B}{r_i} \frac{|C|}{\sum_{C'} |C'|} \frac{e(i+1) \sum_{D \in \text{Child}(C)} |D|}{|C|} \\ &\geq (1 - \alpha) \frac{B \sum_C e(i+1) \sum_D |D|}{r_i \sum_C |C|} \\ &\geq (1 - \alpha) \frac{B e(i+1) n_{i+1}}{r_i n_i} \\ &\geq B(1 - \alpha) / r_i \end{aligned}$$

where n_i, n_{i+1} are the number of solutions at levels $i, i+1$ respectively.

Let κ_i be the maximum cluster size during stage i . From the discussion above, the expected number of particles at any solution at level $i + 1$ is at most $\kappa = \kappa_i r_i / (1 - \alpha)$. The actual distribution is Poisson with expectation κ . The probability that there exist k particles in some solution is given by $\kappa^k / k! e^{-\kappa} < (e\kappa/k)^k e^{-\kappa}$. Choosing $k = \kappa + s$ this probability is at most $e^s (\kappa / (\kappa + s))^{\kappa+s}$. Set $s = \beta\kappa$, for some $\beta \leq 1/2$. The above probability becomes

$$\begin{aligned} e^{\beta\kappa} \left(\frac{\kappa}{\kappa + \beta\kappa} \right)^{\kappa + \beta\kappa} &= \left[e^{\beta(1 + \beta)^{-(1 + \beta)}} \right]^\kappa \\ &= e^{(\beta - (1 + \beta) \ln(1 + \beta))\kappa}. \end{aligned} \quad (3)$$

Now use the Taylor series for $\ln(1 + \beta)$ to get $\ln(1 + \beta) > \beta - \beta^2/2$. Plugging this value in (3), the probability is at most

$$\begin{aligned} e^{(-\beta^2/2 + \beta^3/2)\kappa} &= e^{-\beta^2\kappa(1 - \beta)/2} \\ &< e^{-\beta^2\kappa/4} \\ &= e^{-s^2(1 - \alpha)/4r_i\kappa_i}. \end{aligned} \quad (4)$$

Choose now $s = h^2$ and $\kappa_{i+1} = \kappa_i r_i / (1 - \alpha) + s$. Assuming without loss of generality that $\kappa_0 = 0$, we find that $\kappa_i < ih^2 r_1 r_2 \cdots r_i / (1 - \alpha)^i$, or that $\kappa_i < ie^2 h^2$, where for the last result we used the values for α and r_i defined in Lemma 5. Substituting this value into (4) we see that $\Pr[\kappa_i > ie^3 h^2] < e^{-h^4/4e^3 ih^2} = e^{-h^2/4e^3 i} < e^{-h/4e^3}$. Summing over all particles B and the first i stages, the probability is at most $iBe^{-h/4e^3}$. \square

We now proceed to prove that Lemma 1 is true for stage $i + 1$.

Lemma 5 *In the end of stage i , with probability at least $1 - \delta_{i+1}$, the number of particles in any two layers D, D' at the $i + 1$ -th level of \mathcal{T} is proportional to the sum of the sizes of these layers, i.e.*

$$\frac{1}{r_{i+1}} \frac{|D|}{|D'|} \leq \frac{N_D}{N_{D'}} \leq r_{i+1} \frac{|D|}{|D'|}$$

where $r_{i+1} = r_i(1 + \alpha)^3 / (1 - \alpha)^3$ and $\delta_{i+1} = \delta_i + 2e^{-B/c\hat{d}^2 h^8 p^2(n)}$.

Proof: Let D be a child of C at level $i + 1$. Before the end of stage i all we see is a single bipartite graph between C and the union of its children. Throwing random edges between the two layers induces a uniform distribution on the nodes of C and therefore a particular *cluster* will be in D with probability $p_D = |D| / \sum_{D' \in \text{Child}(C)} |D'|$.

Denote by N_D the number of particles in D . These particles form clusters which, as in Lemma 2, are independently distributed. By applying Chernoff bounds we get that

$$|N_D - M_{CPD}| > \alpha M_{CPD} \quad (5)$$

with probability at most $e^{-(\alpha M_{CPD}/\kappa_i)^2/M_C} < e^{-\alpha^2 \hat{B}/64\kappa_i^2 p^2(n)}$. Using the value of κ_i found in Lemma 4 and summing over all layers at the same level, the probability that (5) is true for some layer is at most $p(n)e^{-\alpha^2 B/ch^6 p^2(n)}$, for some small constant $c > 1$. The number of particles in any two layers D, D' at level $i + 1$ will be

$$\begin{aligned} \frac{N_D}{N_{D'}} &\leq \frac{1 + \alpha}{1 - \alpha} \frac{M_{CPD}}{M_{C'D'}}, \quad D, D' \text{ children of } C, C' \\ &\leq r_i \left(\frac{1 + \alpha}{1 - \alpha} \right)^3 \frac{|D|}{|D'|} \end{aligned}$$

where the lower bound can be obtained similarly. Setting $r_{i+1} = r_i(1 + \alpha)^3 / (1 - \alpha)^3$, $r_0 = 1$ we find that $r_i = (1 + \alpha)^{3i} / (1 - \alpha)^{3i}$. Choosing $\alpha = (6h^2 + 1)^{-1}$, with h being the height of \mathcal{T} , $r_i < e^{i/h^2}$, for $i = 0, \dots, h$.

To define δ_i we work similarly. Substituting the value of α found above, the failure probability of both phases is at most $e^{-B/c\hat{d}^2 h^8 p^2(n)}$ which is bigger than the probability in Lemmas 3 and 4. Setting $\delta_{i+1} = \delta_i + 2e^{-B/c\hat{d}^2 h^8 p^2(n)}$, $\delta_i = 0$ we find that $\delta_i = 2ie^{-B/c\hat{d}^2 h^8 p^2(n)}$. What is the probability that the algorithm fails in reaching depth h ? By choosing $B = O(\hat{d}^2 h^8 p^2(n) \ln h)$, $\delta_h < 1/4$. Lemma 1 and Theorem 1 then follow from the discussion above. \square

5 Combinatorial properties

In the previous sections we gave an algorithm that works well for most graphs whose component tree is constructed according to certain criteria. One such criterion was that the neighbors of a node were chosen at random. We would then like to argue that in a real life problem, the bipartite graph between layers has to be “random enough” for a similar analysis to work. Redefining our goals we would like to replace this randomness assumption by a set of *combinatorial* properties so that every problem whose state space has these properties can be solved with high probability using the “Go with the winners” strategy.

One combinatorial property that guarantees certain pseudorandomness is expansion. Graphs with good expansion have the nice property, among other things, that a random walk on the nodes of the graph is rapidly

mixing, i.e. it gets very close to the stationary distribution after a number of steps that is only polylogarithmic on the size of the graph [Alon86, Broder86, Aldous87, JS88].

Definition 1 *The expansion v of a graph $G(V, E)$ with n vertices is:*

$$v = \min_{|S| \leq \frac{|V|}{2}} \left\{ \frac{|N(S)|}{|S|} \right\} \quad (6)$$

where $N(S)$ is the set of vertices in \bar{S} which are adjacent to some vertex in S .

Let P be the transition matrix of an ergodic Markov Chain X with states $V = \{1, 2, \dots, n\}$. Then $P_{i,j} = \Pr[X_{t+1} = j | X_t = i]$. Let $\pi_t = \pi_0 P^t$ be the probability distribution of X_t at time t and π be the unique stationary distribution the chain converges independently of the initial distribution. Denote by $\Delta(t)$ the distance between π_t and π at time t : $\Delta(t) = \max_i \frac{|\pi_{i,t} - \pi_i|}{\pi_i}$.

The lemma that relates the expansion of a graph to the speed of a convergence to the stationary distribution is:

Lemma 6 ([Alon86, Broder86, Aldous87, JS88]) *Let G be a graph on n vertices and P the transition matrix of a random walk on G . If G has expansion v then*

$$\Delta(t) < \frac{(1 - \frac{v^2}{d})^t}{\min_i \pi_i}$$

where d is the maximum degree of G .

Corollary 2 *For $t = \frac{d}{v^2}(\ln \epsilon^{-1} + \ln \pi_{\min}^{-1})$, $\Delta(t) < \epsilon$.*

At this point we will replace the assumption of the existence of a random bipartite graph between a component and the union of its children by that of the bipartite graph being a good expander (in the sense of Corollary 2). However, since the bipartite graphs may be highly unbalanced, in that one layer may have many more nodes than the other (Condition 4 of Section 3), we would like the bipartite graphs to be good expanders in the following way: Suppose the bipartite graph consists of two layers of nodes A and B , with A being the top layer. Let $S \subseteq B$. Then we would like the following to be true

$$|N(N(S)) - S| \geq v|S|, \quad |S| < |B|/2.$$

In other words, let $x \in A$ and $y, z \in B$ be two of the neighbors of x . Then for every x, y, z as above, connect y, z by an edge and call the resulting graph on B the induced graph G_B . Then we want G_B to be a good expander.

Having eliminated the random edges between layers it is no longer clear whether each layer will receive particles in proportion to its size. Fortunately enough, the only change we need to make to the algorithm is at the randomization phase. Let t be as in Corollary 2. The modified stage i will be as follows:

- Stage i : We proceed again in two phases:

1. The *redistribution* phase remains the same.
2. In the *randomization* phase, for each new particle perform a random walk for $2t + 1$ steps, restricting the moves between levels i and $i+1$. Thus, in odd steps, we are at a node x with $f(x) = i$, and we go to a random neighbor y with $f(y) = i + 1$, and, in even steps, we are at a node x with $f(x) = i + 1$, and we go to a random neighbor y with $f(y) = i$.

Go to the next stage.

The randomization phase will make sure that at the end of the walk each particle is at a random position in the component it started in, but at level $i + 1$. Since the expansion of the bipartite graph between successive layers is large we should be able to do this by moving each particle for a number of steps that is polylogarithmic in the size of the state space, or polynomial in the size of the instance of problem II.

5.1 Analysis

To prove that the algorithm works correctly, we need to argue that particles are distributed evenly among the children of a layer. With some small changes we can follow the outline of Section 4.1.

One difference in the arguments is that cluster size is irrelevant if there is good expansion. No matter how big clusters become during the redistribution phase, at the end of the randomization phase, with high probability, all clusters will be of size one since particles take independent random walks and all such walks end in almost the stationary distribution.

This also means that we can weaken the assumption that relates the number of solutions of value $d - 1$ to the number of solutions of value d (Condition 4 of Section 3). In particular, it is no longer necessary for this ratio to be bounded by $e(d)$, since this assumption was needed only to prove that clusters never become too big. So, at this point we will replace the $e(d)$ bound on the ratio of two levels by any polynomial on the size of the input, since expansion guarantees that clusters will stay small.

Due to space restrictions we only give a sketch of the proof. Lemma 2 will still hold since the only thing we

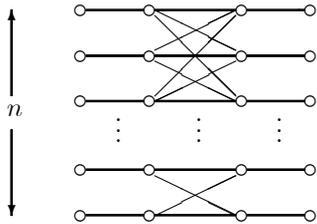


Figure 2: Graph G of a matchings problem where “Go with the winners” works well.

need to show is that particles are independently distributed among the nodes of C . This will be true however, because at the end of the randomization phase of stage $i - 1$, particles will be distributed among the nodes of C according to the stationary distribution. Similarly, for Lemma 5 we need to show that if D is a child of C then a particle will be in D with probability $p_D = |D| / \sum_{D' \in \text{Child}(C)} |D'|$. In the case of an expander bipartite graph where each node has $e(i)$ neighbors in the i -th level, particles will be distributed according to the stationary distribution which is uniform on the nodes of the children components. Even if the bipartite graph is not regular but the stationary probability π_D , of being in a particular child D , is proportional to the size of D then the result will still be true.

What about the running time of the algorithm? In light of Corollary 2 the running time increases only by a polynomial factor which corresponds to the time each particle spends in a random walk between successive layers.

Theorem 2 *Let \mathcal{T} be a component tree constructed according to the new criteria. The modified “Go with the winners” algorithm, with $B = O(\hat{d}^2 h^2 q^2(n) \ln h)$ particles, succeeds with high probability in reaching depth h after $O(hBt)$ steps.*

6 “Go with the winners” can beat simulated annealing

In this section we present an instance of a matchings problem (Figure 2) where “Go with the winners” works well. This is an one layer instance of a multilayered version appearing in [SH88]. We were able to prove, using an argument similar to that in [SH88], that simulated annealing takes exponential time to find the perfect matching no matter what the cooling schedule is.

In light of the results of the previous section, to argue that “Go with the winners” succeeds in finding the perfect matching we only had to show that the bipartite graph between layers had good expansion. We were able to do so using a canonical paths argument as in [JS88]. In this case however, the problem was to define the paths in such a way so that they never use any nodes (matchings of a given size) outside the two successive layers. Fortunately enough we were able to prove that such paths exist. We now state the two results that helped us prove that “Go with the winners” succeeds in finding the perfect matching.

Lemma 7 *The tree-decomposition for the given matchings problem is a path from the set of matchings of size 1 to the perfect matching of size $2n$, i.e., any matching of size i is transformable to any other matching of the same size by a sequence of alternating deletions and additions of edges of G .*

We were able to prove the above lemma by showing the existence of a path between any two matchings of size i that passed only through matchings of values $i - 1, i$. Since all matchings of a given size belong to the same layer, the resulting decomposition degenerates to a single path.

Lemma 8 *The bipartite graph between matchings of size $i - 1$ and i has expansion $1/\text{poly}$.*

To prove this result we used the techniques developed in [JS88] for a weighted version of expansion, the conductance of a graph. By defining canonical paths between matchings one can prove that a graph has good conductance and therefore good expansion (a canonical path between matchings u and v is a way of transforming u to v). Here however, the paths had to be defined in such a way so that they stayed between the two layers. Fortunately, for this particular example this was possible. Using the fact that the ratio of matchings that are local minima over matchings that are not was bounded by $n/2$, we were able to show that the ratio of any two layers is bounded by $n^2/2$ thus proving that Condition 4 of Section 3 is met. Due to space limitations however we defer the proof of the above results for the journal version of the paper.

Theorem 3 *The “Go with the winners” algorithm succeeds with high probability in finding the perfect matching in this instance of a matchings problem.*

7 Non-layered search graphs

In this section we describe an algorithm to be used with problems Π whose search graphs G_Π are not layered, i.e.

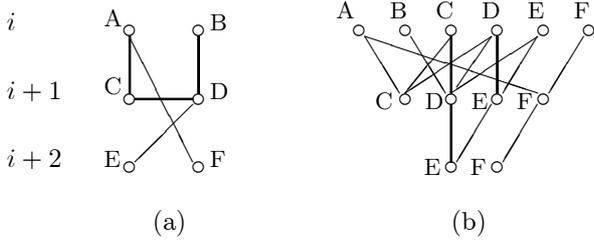


Figure 3: A non-layered graph G_{Π} in (a) is equivalent to the layered graph G'_{Π} in (b).

values of neighboring solutions in G_{Π} do not necessarily differ by one. The approach we take is, first, transform the graph G_{Π} into an equivalent *layered* graph G'_{Π} that preserves the structure of solutions and second, modify the “Go with winners” algorithm to find a good solution in G'_{Π} . Since the first step is only a conceptual one we can apply the modified “Go with the winners” algorithm directly to the original search graph G'_{Π} .

Intuitively, we obtain the graph G'_{Π} as follows: We let the i -th level of G'_{Π} consist of all the solutions of G_{Π} of value i or greater. Two solutions u, v in levels $i, i + 1$ of G'_{Π} are connected by an edge if and only if $u = v$ and the value of u is greater than i , or u, v are neighbors in G_{Π} (Figure 3). Observe now that a layer C at depth i of the tree decomposition \mathcal{T}' of G'_{Π} corresponds exactly to a component C of G_{Π} that contains solutions of value i or greater.

One drawback of this approach however is that we cannot directly apply “Go with the winners” to G'_{Π} . Consider what happens during the i -th stage of the algorithm. In the redistribution phase particles are redistributed according to their down degrees. Then in the randomization phase, each particle takes a random walk with moves restricted between the two successive layers. At the end of the walk, particles will be distributed among the solutions in level $i + 1$ of \mathcal{T}' (solutions of value at least $i + 1$ in a subcomponent of C in G_{Π}) according to the stationary distribution, which is proportional to the *up* degrees of solutions in level $i + 1$ of \mathcal{T}' .

Since the invariant we want to maintain is that particles are *uniformly* distributed among solutions in the $i + 1$ -th level of \mathcal{T}' , we can achieve this by adding another redistribution phase (this time according to the up degrees) immediately after the randomization phase of stage i . The analysis of the algorithm is similar to the analysis in Sections 4 and 5, but details are omitted from this version of the paper. When we combine these steps, the algorithm for general search graphs becomes:

“Go with the winners” algorithm for general search graphs

- Stage 0: Generate B random solutions. Place particles on each of these solutions.
- Stage i : Proceed in three phases. In the *preredistribution* phase, compute for each particle x its *up degree* d_x^u , which is the number of neighbors y of value at least $i - 1$. Then redistribute the B particles according to their up degrees. In the *postredistribution* phase redistribute the particles according to their down degree d_x^d , which is the number of neighbors y of value i or greater. Finally, in the *randomization* phase, for each new particle perform a random walk, restricting the moves as follows: in odd steps, go to a random neighbor y with value $f(y) \geq i$, and, in even steps, go to a random neighbor y with value $f(y) \geq i + i$.

Go to the next stage.

8 Conclusions and Future Research

In this work we presented a new theoretical framework for combinatorial optimization. While many heuristics (local optimization, simulating annealing, metropolis, genetic algorithms) can be viewed as moving particles in a solution space, the tradeoff between time and optimality of such algorithms is difficult to analyze rigorously.

In Section 2 we described a “Go with the winners” strategy. [AV] applied such a scheme in the concrete setting of searching for a leaf in a tree. Taking this idea one step further we reduced the problem of finding a good solution for an optimization problem Π to that of finding a deep vertex in a tree-decomposition of the search graph of Π .

In Section 3 we made a first attempt to capture the necessary properties the solution spaces should have so that our search algorithm can be applied successfully. In Section 4 we performed a probabilistic analysis of “Go with the winners” and we showed that when the state space has the above properties a good solution can be found quickly with high probability. In Section 5 we showed that a combinatorial property, namely expansion, was sufficient for the algorithm to work and in Section 6 we presented a matchings problem whose solution space had this property. Finally, in Section 7 we presented an algorithm for arbitrary search graphs. But, while expansion is a nice property, one would ex-

pect that things in real life tend to be more complicated. The following lemma from [DI95] makes this more clear.

Lemma 9 *In a random graph chosen with edge probability $p = \lambda(n)/n$, where $\lambda(n) \geq 3$, all but a fraction $O(e^{-\lambda(n)})$ of the nodes of G are contained in a large component whose expansion factor is $1/(\lambda(n) \log n)$. The rest of the nodes are either isolated or belong to tree components of size at most polylogarithmic in n .*

In the case of a bipartite graph, these nodes correspond to a part in the graph that has a *tree-like* structure, while the rest of the nodes correspond to a part between the two layers that has good expansion properties. Distinguishing between the two parts is important since particles belonging to the tree parts will form clusters as they forward to the next layer, thus making the algorithm less efficient.

We would then like to combine the ideas of Sections 4 and 5 to find a version of expansion that is both true of random bipartite graphs and suffices for “Go with the winners” to work well. One candidate would be to consider bipartite graphs having a part with good expansion and many smaller components of size polylogarithmic in the size of the large component, but probably additional assumptions need to be made for the algorithm to work.

Our “Go with the winners” algorithm seems to have many of the same advantages as using Aldous and Vazirani’s method supplementing simulated annealing. Since it is also simpler, this might give better overall performance. It would be interesting to compare the three approaches (simulated annealing, “Go with the winners”, or a combination as Aldous and Vazirani suggested) experimentally.

Of course, to use our results, one would have to understand the structure of search spaces for real combinatorial optimization problems. Some such problems have been analyzed for other search algorithms. For example, [SH88] give an analysis of the Metropolis algorithm for the maximal matching problem, and [JS093] analyze the Metropolis algorithm for the bisection problem for random graphs with planted bisections. We would like to find similar results for “Go with the winners”.

Some of the conditions on the tree-decompositions of graphs seem necessary for “Go with the winners” to work well. Are they also necessary for other local optimization algorithms? One approach to answering this positively is to prove a lower bound for optimization in a restricted model that captures the notion of “local search”, such as a modified JAG model [CR80].

9 Acknowledgements

We would like to thank T. C. Hu for raising the general issue of analyzing search algorithms, and Christos Papadimitriou and Umesh Vazirani for helpful discussions.

References

- [Aldous87] D. Aldous. “On the markov chain simulation method for uniform combinatorial distributions and simulated annealing”. *Probability in the Engineering and Informational Sciences*, 1987, 1, pp. 33–46.
- [AV] D. Aldous and U. Vazirani. “Go with the winners” Algorithms. In *Proc. 35th FOCS*, pages 492–501, 1994.
- [Alon86] N. Alon. “Eigenvalues and Expanders”. *Combinatorica*, 1986, 6, pp. 83–96.
- [Broder86] A. Z. Broder. “How hard is it to marry at random? (On the approximation of the permanent)”. In *Proc. 18th Ann. ACM Symp. on Theory of Computing*, 1986, pp. 50–58.
- [CR80] S. A. Cook and C. W. Rackoff. “Space Lower Bounds for Maze Threadability on Restricted Machines”. In *SIAM Journal on Computing*, 1980, 9, 4, pp.635–652.
- [DI95] T. Dimitriou and R. Impagliazzo. “On the expansion of a random graph”. *Manuscript*, 1995.
- [JS88] M. R. Jerrum and A. Sinclair. “Conductance and the rapid mixing property of Markov chains: The approximation of the permanent resolved”. In *Proc. 20th STOC*, pages 235–244, 1988.
- [JS093] M. R. Jerrum and G. Sorkin. “Simulating annealing for graph bisection”. In *Proc. 34th FOCS*, pages 94–103, 1993.
- [SH88] G. H. Sasaki and B. Hajek. “The time complexity of maximum matching by simulated annealing”. In *Journal ACM*, 1988, 35, 2, pp. 387–403.