An Improved Ant Colony Optimisation Algorithm for the 2D HP Protein Folding Problem

Alena Shmygelska and Holger H. Hoos[*]

Department of Computer Science, University of British Columbia,
Vancouver, B.C., V6T 1Z4, Canada
{oshmygel,hoos}@cs.ubc.ca
http://www.cs.ubc.ca/labs/beta

Abstract. The prediction of a protein's structure from its amino-acid sequence is one of the most important problems in computational biology. In the current work, we focus on a widely studied abstraction of this problem, the 2-dimensional hydrophobic-polar (2D HP) protein folding problem. We present an improved version of our recently proposed Ant Colony Optimisation (ACO) algorithm for this $\mathcal{NP}$-hard combinatorial problem and demonstrate its ability to solve standard benchmark instances substantially better than the original algorithm; the performance of our new algorithm is comparable with state-of-the-art Evolutionary and Monte Carlo algorithms for this problem. The improvements over our previous ACO algorithm include long range moves that allows us to perform modification of the protein at high densities, the use of *improving* ants, and selective local search. Overall, the results presented here establish our new ACO algorithm for 2D HP protein folding as a state-of-the-art method for this highly relevant problem from bioinformatics.

1 Introduction

Ant Colony Optimisation (ACO) is a population-based approach for solving combinatorial optimisation problems that is inspired by the foraging behaviour of ants. The fundamental approach underlying ACO is an iterative process in which a population of simple agents ("ants") repeatedly construct candidate solutions; this construction process is probabilistically guided by heuristic information on the given problem instance as well as by a shared memory containing experience gathered by the ants in previous iterations ("pheromone trails"). Following the seminal work by Dorigo *et al.* [5], ACO algorithms have been successfully applied to a broad range of hard combinatorial problems (see, *e.g.*, [6, 7]).

In this paper, we present a substantially improved version of the ACO algorithm first proposed in [18] for solving an abstract variant of one of the most challenging problems in computational biology: the prediction of a protein's structure from its amino-acid sequence. Genomic and proteomic sequence information is now available for an increasing number of organisms, and genetic engineering methods for producing proteins are well developed. The biological function and properties of proteins, however,

[*] To whom correspondence should be addressed.

are crucially determined by their structure. Hence, the ability to reliably and efficiently predict protein structure from sequence information would greatly simplify the tasks of interpreting the sequence data collected, of designing drugs with specific therapeutic properties, and of developing biological polymers with specific material properties. Currently, protein structures are primarily determined by techniques such as NMRI (nuclear-magnetic resonance imaging) and X-ray crystallography, which are expensive in terms of equipment, computation and time. Additionally, they require isolation, purification and crystallization of the target protein. Computational approaches to protein structure prediction are therefore very attractive. In this work, we focus on one of the most studied simple protein models — the two dimensional Hydrophobic-Polar (2D HP) Model. Even in this simplified model, finding optimal folds is computationally hard ($\mathcal{NP}$-hard).

The remainder of this paper is structured as follows. In Section 2, we introduce the 2D HP model of protein structure, and give a formal definition of the 2D HP Protein Folding Problem as well as a brief overview of existing approaches for solving it. Our improved ACO algorithm for the 2D HP Protein Folding Problem is described in Section 3. (More information on our previous Ant Colony Optimisation algorithm can be found in [18]). An empirical study of our new algorithm's performance and the role of various algorithmic features is presented in Section 4. In Section 5 we draw some conclusions and point out several directions for future research.

2 The 2D HP Protein Folding Problem

Since the processes involved in the folding of proteins are very complex and only partially understood, simplified models like Dill's Hydrophobic-Polar (HP) model have become one of the major tools for studying proteins [12]. The HP model is based on the observation that hydrophobic interaction is the driving force for protein folding and the hydrophobicity of amino acids is the main force for development of a native conformation of small globular proteins [12, 15].

In the HP model, the primary amino-acid sequence of a protein (which can be represented as a string over a twenty-letter alphabet) is abstracted to a sequence of hydrophobic (H) and polar (P) residues, *i.e.*, amino-acid components. The protein conformations of this sequence are restricted to self-avoiding paths on a lattice; for the 2D HP model considered here, a 2-dimensional square lattice is used. An example for a protein conformation under the 2D HP model is shown in Figure 1.

One of the most common approaches to protein structure prediction is based on the thermodynamic hypothesis which states that the native state of the protein is the one with the lowest Gibbs free energy. In the HP model, based on the biological motivation given above, the energy of a conformation is defined as a number of topological contacts between hydrophobic amino-acids that are not neighbours in the given sequence. More specifically, a conformation c with exactly n such H-H contacts has free energy $E(c) = n \cdot (-1)$; *e.g.*, the conformation shown in Figure 1 has energy -9.

The 2D HP Protein Folding Problem can be formally defined as follows: Given an amino-acid sequence $s = s_1 s_2 \dots s_n$, find an energy-minimising conformation of s, *i.e.*, find $c^* \in C(s)$ such that $E^* = E(c^*) = \min\{E(c) \mid c \in C\}$, where $C(s)$ is the

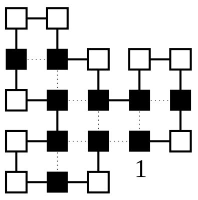

Fig. 1. A sample protein conformation in the 2D HP model. The underlying protein sequence (Sequence 1 from Table 1) is HPHPPHHPHPPHPHHPPHPH; black squares represent hydrophobic amino-acids while white squares symbolise polar amino-acids. The dotted lines represents the H-H contacts underlying the energy calculation. The energy of this conformation is -9, which is optimal for the given sequence.

set of all valid conformations for s. It was recently proved that this problem and several variations of it are $\mathcal{NP}$-hard [11].

Existing 2D HP Protein Folding Algorithms

A number of well-known heuristic optimisation methods have been applied to the 2D HP Protein Folding Problem, including Evolutionary Algorithms (EAs) [10, 11, 20, 19] and Monte Carlo (MC) algorithms [1, 3, 9, 13, 14, **?**]. The latter have been found to be particular robust and effective for finding high-quality solutions to the 2D HP Protein Folding Problem [9].

An early application of EAs to protein structure prediction was presented by Unger and Moult [19, 20]. They presented a nonstandard EA incorporating characteristics of Monte Carlo methods, which was able to find high-quality conformations for a set of protein sequences of length up to 64 amino-acids (see Table 1). Unfortunately, it is not clear how long their algorithm ran to achieve these results.

Various Monte Carlo methods are among the best known algorithms for the 2D HP Protein Folding Problem, including the Pruned Enriched Rosenbluth Method (PERM) of Grassberger *et al.* [1, 9]. PERM is a biased chain growth algorithm. Using this method, the best known solution for Sequence 7 ($E^* = -36$), Sequence 9 ($E^* = -53$) and Sequence 11 ($E^* = -48$) from Table 1 were found; however, it took 30 hours on a 500 MHz DEC 21264 CPU to obtain the best-known conformation for Sequence 8 [9].

Other methods for this problem include the dynamic Monte Carlo algorithm by Ramakrishnan *et al.* [14], which found conformations with energies -46 and -44 for Sequence 10 and 11, respectively. Liang *et al.* [13] introduced the evolutionary Monte Carlo (EMC) algorithm which works with population of individuals that each performs Monte Carlo (MC) optimisation. They also implemented a variant of EMC which reinforces certain secondary structures (α-helices and β-sheets). EMC found the best-known conformation (with energy -42) for Sequence 8 and a conformation with energy -52 for Sequence 9 (with secondary structure constraints), but failed to find the best known conformation for Sequence 7. Chikenji *et al.* introduced the Multi-self-overlap ensemble (MSOE) Monte Carlo method [3], which considers overlapping chain configurations; it found a best-known configuration for Sequence 10 ($E^* = -50$) and a sub-optimal configuration for Sequence 11 ($E^* = -47$).

Seq. No.	*Length*	E^*	*Protein Sequence*
1	20	**-9**	$(HP)_2PH_2PHP_2HPH_2P_2HPH$
2	24	**-9**	$H_2P_2(HP_2)_6H_2$
3	25	**-8**	$P_2HP_2H_2P_4H_2P_4H_2P_4H_2$
4	36	-14	$P_3H_2P_2H_2P_5H_7P_2H_2P_4H_2P_2HP_2$
5	48	-23	$P_2HP_2H_2P_2H_2P_5H_{10}P_6H_2P_2H_2P_2HP_2H_5$
6	50	-21	$H_2(PH)_3PH_4PHP_3HP_3HP_4HP_3HP_3HPH_4(PH)_3PH_2$
7	60	-36	$P_2H_3PH_8P_3H_{10}PHP_3H_{12}P_4H_6PH_2PHP$
8	64	-42	$H_{12}(PH)_2(P_2H_2)_2P_2H(P_2H_2)_2P_2H(P_2H_2)_2P_2HPHPH_{12}$
9	85	-53	$H_4P_4H_{12}P_6(H_{12}P_3)_3HP_2(H_2P_2)_2HPH$
10	100	-50	$P_3H_2P_2H_4P_2H_3(PH_2)_3H_2P_8H_6P_2H_6P_9HPH_2PH_{11}P_2H_3PH_2$ $PHP_2HPH_3P_6H_3$
11	100	-48	$P_6HPH_2P_5H_3PH_5PH_2(P_2H_2)_2PH_5PH_{10}PH_2PH_7P_{11}H_7P_2H$ $PH_3P_6HPHP_2$

Table 1. Benchmark instances for the 2D HP Protein Folding Problem used in this study with optimal or best known energy values E^*. (E^* values printed in bold-face are provably optimal.) The first eight instances can also be found at http://www.cs.sandia.gov/tech_reports/compbio/tortilla-hp-benchmarks.html, Sequence 9 is taken from [10], and the last two instances are taken from [14]. (H_i, P_i, and $(\ldots)_i$ indicate i-fold repetitions of the respective symbol or subsequence.)

Finally, the Core-directed Chain Growth (CG) method of Beutler *et al.* approximates the hydrophobic core of the protein with a square (this is a very restrictive heuristic that finds only certain native states). CG was able to find optimal or best known conformations for Sequences 1 through 8, except for Sequence 7 [2].

Currently, none of these algorithm appears to completely dominate the others in terms of solution quality and run-time.

3 The Improved ACO Algorithm

The ants in our ACO algorithm construct candidate conformations for a given HP protein sequence, apply local search to achieve further improvements, and update the pheromone trails based on the quality of the solutions found, as seen in the outline Figure 2.

As in [11], candidate conformations are represented using local structure motifs (or relative folding directions) *straight* (S), *left* (L), and *right* (R) which for each amino-acid indicate its position on the 2D lattice relative to its direct predecessors in the given sequence (see Figure 3).

Since conformations are invariant with respect to rotations, the position of the first two amino-acids can be fixed without loss of generality. Hence, we represent candidate conformations for a protein sequence of length n by a sequence of local structure motifs of length $n-2$. For example, the conformation of Sequence 1 shown in Figure 1 corresponds to the motif sequence LSLLRRLRLLSLRRLLSL.

```
procedure ACO-for-static-optimisation
    input: problem instance
    output: candidate solution
    initialise pheromone trails
    while (termination condition not met) do
        construct candidate solutions
        perform local search
        update pheromone trails
    end while
    return(best found solution)
end procedure
```

Fig. 2. Algorithmic outline of a generic ACO algorithm for a static combinatorial optimisation problem.

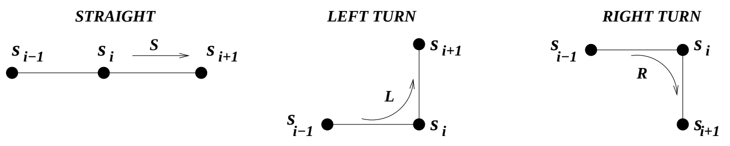

Fig. 3. The local structure motifs that form the solution components underlying the construction and local search phases of our ACO algorithm.

Construction Phase, Pheromone and Heuristic Values

In the construction phase of our ACO algorithm, each ant first randomly determines a starting point within the given protein sequence. From this starting point, the given protein sequence is folded in both directions, adding one amino-acid symbol at a time. The relative directions in which the conformation is extended in each construction step are determined probabilistically using heuristic values $\eta_{i,d}$, as well pheromone values $\tau_{i,d}$ (also called trail intensities), where where i is a sequence position and $d \in \{S, L, R\}$ is the direction of folding at position i. These relative directions correspond to local structure motifs between triples of consecutive sequence positions $s_{i-1}s_is_{i+1}$ that form the solution components used by our ACO algorithm; conceptually, these play the same role as the edges between cities in the Traveling Salesperson Problem, a classical application domain of ACO.

Likewise, pheromone values $\tau'_{i,d}$ and heuristic values $\eta'_{i,d}$ are used when extending a conformation from position i to the $i-1$. In our algorithm, we use $\tau'_{i,L} = \tau_{i,R}$, $\tau'_{i,R} = \tau_{i,L}$, and $\tau'_{i,S} = \tau_{i,S}$ and analogous equalities for the respective η' values. This reflects a fundamental symmetry underlying the folding process: Extending the fold from sequence position i to $i+1$ by placing s_{i+1} right of s_i (as seen from s_{i-1}) or extending it from position i to $i-1$ by placing s_{i-1} left of s_i (as seen from s_{i+1}) leads to the same local conformation of $s_{i-1}s_is_{i+1}$.

The heuristic values $\eta_{i,d}$ should guide the construction process towards high-quality candidate solutions, *i.e.*, towards conformations with a maximal number of H-H interactions. In our algorithm, this is achieved by defining $\eta_{i,d}$ based on $h_{i+1,d}$, the number of new H-H contacts achieved by placing s_{i+1} in direction d relative to s_i and s_{i-1} when folding forwards (backwards folding is handled analogously). Note that if $s_{i+1} = \mathrm{P}$, this amino-acid cannot contribute any new H-H contacts and hence $h_{i+1,S} = h_{i+1,L} = h_{i+1,R} = 0$. Furthermore, for $1 < i < n-1$, $h_{i+1,d} \leq 2$ and $h_{n,d} \leq 3$; the actual $h_{i+1,d}$ values can be easily determined by checking the seven neighbours of the possible positions of s_{i+1} on the 2D lattice (obviously, the position of s_i is occupied and hence not included in these checks). The heuristic values are then defined as $\eta_{i,d} = h_{i+1,d} + 1$; this ensures that $\eta_{i,d} > 0$ for all i and d which is important in order to not exclude *a priori* any placement of s_{i+1} in the construction process.

When extending a partial conformation $s_k \ldots s_i$ to s_{i+1} during the construction phase of our ACO algorithm, the relative direction d of s_{i+1} w.r.t. $s_{i-1}s_i$ is determined based on the heuristic and pheromone values according to the following probabilities:

$$p_{i,d} = \frac{[\tau_{i,d}]^\alpha [\eta_{i,d}]^\beta}{\sum_{e \in \{L,R,S\}} [\tau_{i,e}]^\alpha [\eta_{i,e}]^\beta} \tag{1}$$

The case of extending partial conformation $s_i \ldots s_m$ to s_{i-1} is handled analogously.

In the previous version of the ACO algorithm [18], when folding from a randomly determined starting point l (independently chosen for each ant), first, the partial conformation $s_l \ldots s_1$ is constructed, followed by the partial conformation $s_l \ldots s_n$. Here, we consider a new, physically more plausible mechanism, in which folds are probabilistically extended in both directions; more precisely, in each step an extension in each direction is performed with a probability equal to the number of residues left to fold at the respective end divided by the sum of the number of unfolded residues at both ends. As in previous work [18], we also studied variants of our algorithm in which all ants start their construction process at the same point (left end, middle, or right end of the protein sequence). Performance results for these alternative mechanisms are reported in Section 4. As in our first ACO algorithm for 2D HP Protein Folding [18], we use a backtracking mechanism in the construction process to recover from infeasible conformations, which are frequently encountered when folding long protein sequences.

Local Search

Similar to other ACO algorithms known from the literature, our algorithm for the 2D HP Protein Folding Problem incorporates a local search phase. In this work, we modified the local search mechanism from our previous ACO algorithm by using a new type of *long range move*, *selective local search*, and *improving ants* that perform *probabilistic iterative improvement* on the best conformations seen so far.

Long range moves in the local search phase allow for chain reconfigurations even when the protein conformation is very compact. Similar attempts were previously undertaken but different from our approach these involve disconnection of the chain [14]. Conventional short range moves, such as point mutations (*i.e.*, random modifications of the direction of a single residue, which correspond to a rotations) and macro-mutations (sequences of point mutations between two sequence positions) — both types of moves have been employed in EAs for HP Protein Folding — result in infeasible conformation when the protein is already dense. Likewise, standard Monte Carlo moves, such as the end-move, crankshaft move, and corner move [14], are not powerful enough to change a given configuration significantly.

Therefore, we designed a new type of long range move that closely models the movement of real proteins. First, a sequence position at which the move is going to originate is chosen uniformly at random.[1] Then we randomly modify the direction of the chosen amino-acid and adjust the location of the remaining residues probabilistically as follows. For each subsequent residue, we first attempt to place it using its previous folding direction. If this is infeasible, we refold the residue probabilistically using the same heuristic values as during the initial folding (but ignoring the pheromone values). This initiates a chain reaction that continues until all the residues have found feasible directions. Intuitively, this mechanism mimics the fact that in real proteins, a moving residue will typically push its neighbours in the chain to different positions.

Since these long range moves are computationally quite expensive, our new algorithm applies local search selectively. More precisely, local search is only applied to the best conformations constructed in a given iteration of the algorithm; the fraction of ants that are allowed to perform local search is a parameter of our algorithm. This *selective local search mechanism* is based on the intuition that the improvement in solution quality that can be achieved by the local search procedure strongly depends on the energy of the given starting conformation; in particular, bad starting conformations can rarely be improved to high quality solutions.

In our original ACO algorithm for 2D HP Protein Folding [18], we only considered *forager* ants that perform heuristic conformation construction followed by iterative improvement ("greedy") local search. Here, we additionally introduce *improving* ants that take the global best solution found so far (or best solution in the current iteration) and apply *probabilistic iterative improvement* ("randomised greedy") local search to it. Iterative improvement accepts a new conformation generated via long range moves only when the energy of the new conformation c' improves over the energy of the current conformation, c. Our probabilistic iterative improvement mechanism accepts worsening steps depending on the respective deterioration of the evaluation function with probability $p = E(c')/E(c)$. Algorithm outlines for the iterative improvement and probabilistic iterative improvement local search procedures are given in Figures 4 and 5. The number of improving ants used in each iteration of our algorithm is specified as a fraction of the total number of ants; empirical results on the impact of this parameter on the performance of our algorithm are reported in Section 4.

Update of the Pheromone Values

After each construction and local search phase, selected ants update the pheromone values in a standard way:

$$\tau_{i,d} \leftarrow (1-\rho)\tau_{i,d} + \Delta_{i,d,c} \qquad (2)$$

where $0 < \rho \leq 1$ is the pheromone persistence, a parameter that determines how fast the information gathered in previous iterations is "forgotten", and $\Delta_{i,d,c}$ is the relative solution quality of the given ant's candidate conformation c if that conformation contains

[1] We also tested a probabilistic selection of the origin (based on the constrainedness of the residue position), but results were not significantly different from those for uniform random choice.

```
procedure II-LS
    input: conformation c
    output: conformation $c'$
    while (termination condition not met) do
        choose sequence index $i$ uniformly at random from $1..n$
        $c'$ = longRangeMove($i$)
        if $E(c') \leq E(c)$ then
            return $(c')$
        else
            return $(c)$
        end if
    end while
end procedure
```

Fig. 4. The iterative improvement local search performed by *forager* ants after the construction phase.

a local structure motif d at sequence position i and zero otherwise. We use the relative solution quality, $E(c)/E^*$, where E^* is the known minimal energy for the given protein sequence or an approximation based on the number of H residues in the sequence, in order to prevent premature search stagnation for sequences with large energy values.

4 Empirical Results

To assess its performance, we applied our improved ACO algorithm to the eleven standard benchmark instances for the 2D HP Protein Folding Problem shown in Table 1; these instances have been widely used in the literature [1–3, 11, 13, 18–20]. Experiments were conducted by performing a number of independent runs for each problem instance (500 runs for sequence length $n \leq 50$, 300 for $50 < n \leq 64$, and 100 runs for $n > 64$). Unless explicitly indicated otherwise, we used parameter settings $\alpha = 1$, $\beta = 2$, $\rho = 0.8$ for all experiments; furthermore, a population of 500 ants was used for small sequences ($n \leq 48$) while 1500 ants were used for larger sequences ($n > 48$); 1% of these were allowed to perform local search, and the number of improving ants was set to 0.5% of the total colony size. The local search procedure was terminated if no solution improvement had been obtained while scanning through the protein sequence once (for $n \leq 48$) or twice (for $n > 48$). Furthermore, we used elitist pheromone update in which only the best 1% of the total colony size were used for updating the pheromone values. Run-time was measured in terms of CPU time and all experiments were performed on PCs with 1GHz and Pentium III CPUs, 256KB cache and 1GB RAM.

In the following, we report results from several series of experiments that highlight the impact of various features of our new ACO algorithm on its performance. In these experiments we used primarily two test sequences: Sequence 4 (short sequence, length 36) and Sequence 5 (longer sequence, length 48); these sequences were chosen since the CPU time required to find the best known solutions was sufficiently small to perform a

```
procedure PII-LS
    input: conformation $c$
    output: conformation $c'$
    while (termination condition not met) do
        choose sequence index $i$ uniformly at random from $1..n$
        $c'$ = longRangeMove($i$)
        if $E(c') \leq E(c)$ then
            return $(c')$
        else
            with probability $p = E(c')/E(c)$ do
                return $(c')$
            otherwise
                return $(c)$
            end whith
        end if
    end while
end procedure
```

Fig. 5. The probabilistic iterative improvement local search procedure performed by the *improving* ants on the best configuration seen so far in the optimisation process.

large number of runs (300–500 per instance). We also tested other benchmark sequences from Table 1 and generally obtained results similar to the ones described below.

ACO algorithms exploit heuristic information as well as information learned over multiple iterations (the latter is reflected in the pheromone matrix). In a first experiment, we investigated the impact of these two components and their relative importance for the performance of our algorithms. Following the methodology of Hoos and Stützle [8], we measured run-time distributions (RTDs) of our ACO algorithm, which represent the (empirical) probability of reaching (or exceeding) a given solution quality within a given run-time; the solution qualities used here and in the following experiments are provably optimal or best known energies for the respective sequences. All RTDs are based on 100–500 successful runs; we generally show semi-log plots to give a better view of the distribution over its entire range.

As can be seen from the results shown in Figure 6, both, the pheromone values and the heuristic information are important; when ignoring either of them ($\alpha = 0$ or $\beta = 0$, respectively), the algorithm performs substantially worse, especially for larger sequences. The optimal settings for α and β depends on the problem instance; as shown in Figure 6, the heuristic information seems to be more important than the pheromone information for small sequences. For longer sequences, the pheromone information appears to become more important than the heuristic information. These observations were confirmed for other benchmark instances.

Secondly, we tested how the colony size, *i.e.*, the number of ants that construct candidate solutions in each iteration affects the performance of our ACO algorithm. The proportion of ants that perform local search, the proportion of elitist ants, and the proportion of improving ants was chosen such that in all cases the number of local search

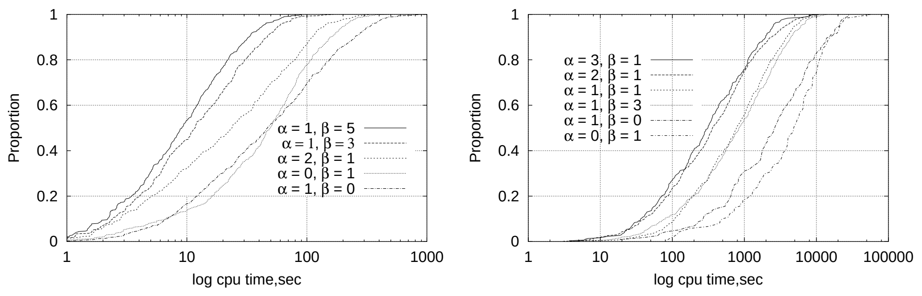

Fig. 6. Effect of the α and β weights for pheromone and heuristic information respectively on the average CPU time required for reaching optimal confirmations. Left side: Run-time distributions (RTDs) for Sequence 4 (length 36), right side: RTDs for Sequence 5 (length 48).

ants, elitist ants, and improving ants, remains the same for all colony sizes. (Colony sizes tested were between 15 and 2000 ants.)

As can be seen from the results shown in Figure 7, there appears to be a single optimal colony size for each problem instance; optimal performance for longer sequences is achieved using larger colonies ($1000-1500$ ants) than for shorter sequences ($500-1000$). It may be noted that using a single ant only (not shown here) was found to result in extremely poor performance. These results can be intuitively explained as follows. For very few ants, the probability of constructing high quality initial solutions is very small and local search requires substantial amounts of CPU time for finding conformations of the desired quality. Beyond a certain colony size, on the other hand, the computational expense incurred for constructing additional conformations cannot be amortised by reductions in local search cost. The longer the given sequence, the more conformations need to be constructed to obtain the coverage (or exploration) of the corresponding more extensive search spaces required to find good starting points for the subsequent local search phase.

Our next experiment was designed to analyse the effectiveness of the selective local search mechanism used in our new ACO algorithm, in which local search is only performed by a certain fraction of all ants that constructed high quality conformations. The results shown in Figure 8 indicate that there is a small optimal interval for the fraction of local search ants; this optimal fraction depends on colony size and on the given problem instance. Essentially, if the fraction of local search ants is too small, the search process has difficulties in finding high quality conformations (lack of search intensification). On the other hand, if too many ants perform local search, the benefit of the additional local search does not amortise the higher computational cost. Our results indicate that longer sequences require a lower fraction of local search ants than shorter sequences; however, given the the larger optimal colony size, the optimal number of local search ants increases with sequence size. This is consistent with the interpretation that larger sequences require a more diversified search process, as provided by locally optimising

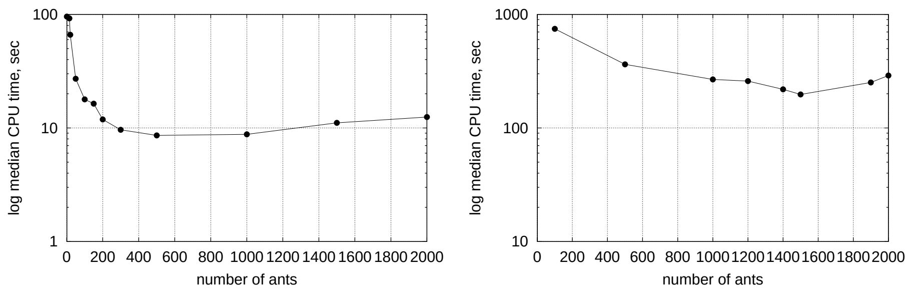

Fig. 7. Median CPU time required to obtain the best known solution quality as a function of the colony size (number of the ants on which construction is performed) for Sequence 4 (left side) and Sequence 5 (right side).

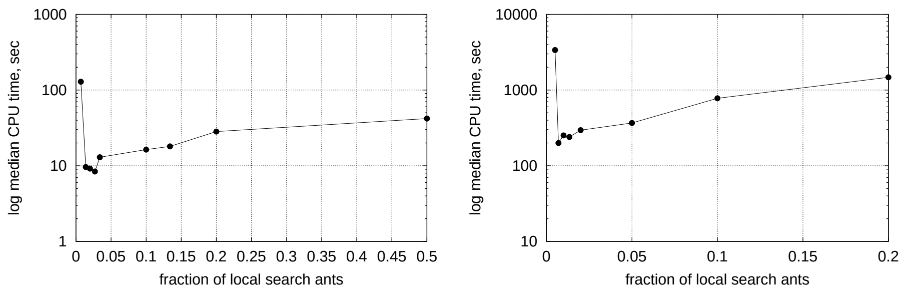

Fig. 8. Median CPU time required to obtain the best known solution quality as a function of the proportion of the ants on which local search is performed for Sequence 4 (left side, colony size 500) and Sequence 5 (right side, colony size 1500).

a larger number of candidate solutions in each iteration. It is worth noting that without a local search phase, the performance of our ACO method is abysmal.

The use of improving ants that, instead of iteratively constructing conformations, use probabilistic iterative improvement local search on the best conformations seen so far is an important new feature of our new ACO algorithm. Figure 9 illustrates the results from our empirical analysis of the effectiveness of this feature in terms of the impact of the fraction of improving ants on the performance of our algorithm. Overall, the use of improving ants results in an performance increase of our algorithm for all sequences; this effect is especially pronounced for long sequences.

It is interesting to note that the optimal ratio between the number of (forager) ants that perform iterative improvement local search and the number of improving ants performing probabilistic iterative improvement appears to be $1 : 1$ for all sequences.

Finally, we studied the effect of the starting point for the construction of conformations on the performance of our improved ACO. It has been shown that real proteins

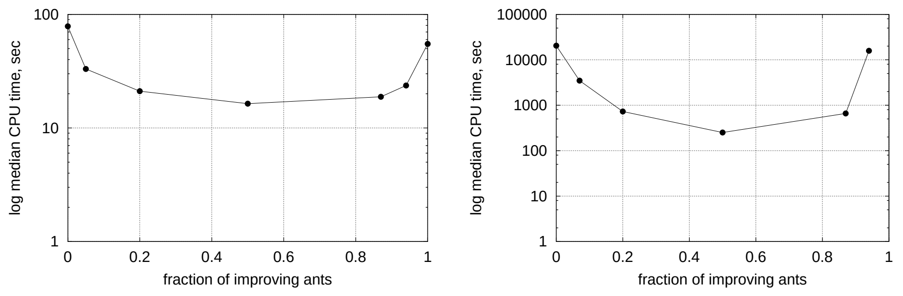

Fig. 9. Median CPU time required to obtain the optimum as a function of the proportion of the improving ants for Sequence 4 (left side) and Sequence 5 (right side).

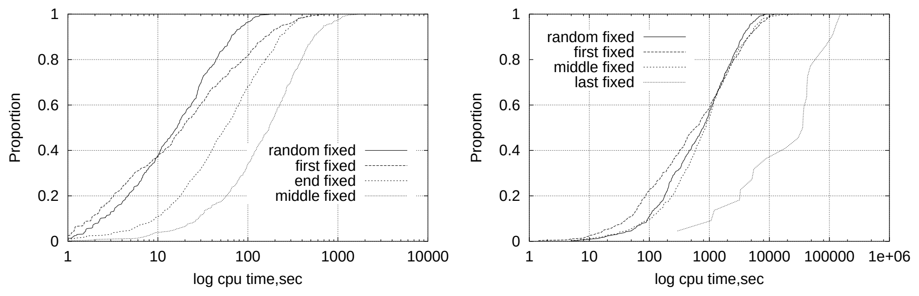

Fig. 10. Impact of various strategies for choosing the starting point for constructing candidate conformations. Left side: RTDs for Sequence 4, right side: RTDs for Sequence 5.

fold by hierarchical condensation starting from folding nuclei; the use of complex and diverse folding pathways helps to avoid the need to extensively search large regions of the conformation space [16]. This suggests that the starting point for the folding process can be an important factor in searching for optimal conformations. We tested four strategies for determining the starting point for the folding process performed in the construction phase of our algorithm: all ants fold forwards, starting at the N-terminus of the given sequence (position 1); all ants fold backwards, starting at the C-terminus (position n); all ants fold forwards and backwards, starting at the midpoint of the given sequence; and all ants fold forwards and backwards, starting at randomly determined sequence positions. As can be seen from Figure 10, the best choice of the starting folding point depends on the given sequence, and in general, most consistent performance on all sequences is obtained by allowing all ants start the folding process from randomly chosen sequence positions. This is particularly the case for longer sequences, which require to a larger extent the added search diversification afforded by multiple and diverse starting points.

Instances			*Original ACO from [18]*			*New ACO (this paper)*		
No	$Length$	E^*	$\hat{E}$	$sr[\%]$	$t_{avg}[CPUsec]$	$\hat{E}$	$sr[\%]$	$t_{avg}[CPUsec]$
1	20	-9	**-9**	100	23.90	**-9**	100	3.33
2	20	-9	**-9**	100	26.44	**-9**	100	2.52
3	25	-8	**-8**	100	35.32	**-8**	100	10.62
4	36	-14	**-14**	16.4	(4,746.12)	**-14**	100	11.81
5	48	-23	**-23**	0.6	(1,920.93)	**-23**	100	405.79
6	50	-21	**-21**	41.9	(3,000.28)	**-21**	100	4,952.92
7	60	-36	-34	0.8	(4,898.77)	**-36**	100	6,2471.24
8	64	-42	-32	4.5	(4,736.98)	**-42**	100	5,844.93

Table 2. Performance comparison of the original and the improved ACO algorithm, where $\hat{E}$ is the best solution quality over all runs, sr is the success rate of the algorithm reported due to the fact that some of the runs of older algorithm were unsuccessful, t_{avg} is the average run-time (in CPU sec on a 1GHz Intel Pentium III machine) required by the algorithm to reach energy $\hat{E}$. Measurements are based on $20 - 700$ runs for the original ACO algorithm and $300 - 500$ tries for the improved algorithm. Energies indicated in bold-face are currently best-known values.

After studying the influence of various parameters on our algorithm we conducted a performance comparison with existing algorithms for the 2D HP Protein Folding Problem. As can be seen from the results reported in Table 2, our new ACO algorithm found optimal or best known solutions for benchmark sequences 1–8, while our previous ACO algorithm [18] had failed to find optimal solutions for the longer sequences 7 and 8. Moreover, the new algorithm finds best-known solutions in every run, and in cases where both algorithms have a success rate of 100%, it requires substantially less time for finding optimal solutions.

For most GA and MC methods found in literature, including [11, 13, 19, 20], only the number of valid conformations scanned during the search is reported. This makes a performance comparison difficult, since run-time spent for backtracking and the checking of partial or infeasible conformations may vary substantially between different algorithms. Table 3 illustrates the solution quality reached by various algorithms on the test instances. These results indicate that our new ACO algorithm is competitive with GA and MC methods described in literature; it works very well on sequences of sizes up to 64 amino acids and produces high quality suboptimal configurations for the longest sequences (85 and 100 amino acids) considered here.

We also compared our improved implementation to the best performing algorithm from the literature for which performance data in terms of CPU time is available — PERM [9] (we used the most recent implementation, which was kindly provided by P. Grassberger). PERM is an iterated heuristic construction algorithm; it evaluates partially folded conformations, and creates copies of those partial configurations that have high statistical weight (enrichment, based on energy achieved and folded length), and it eliminates partial conformations with low weight (pruning). After completing the folding of the current configuration, PERM performs backjumping to the next partial configuration that was put on the stack during enrichment (or starts a new chain). It should be noted that tries in PERM are not entirely independent, since some statistical

No	$Length$	$Energy$	GA	EMC	$MSOE$	New ACO
1	20	-9	**-9** $(30,492)$	**-9** $(9,374)$		**-9**
2	24	-9	**-9** $(30,491)$	**-9** $(6,929)$		**-9**
3	25	-8	**-8** $(20,400)$	**-8** $(7,202)$		**-8**
4	36	-14	**-14** $(301,339)$	**-14** $(12,447)$		**-14**
5	48	-23	**-23** $(126,547)$	**-23** $(165,791)$		**-23**
6	50	-21	**-21** $(592,887)$	**-21** $(74,613)$		**-21**
7	60	-36	-34 $(208,781)$	-35 $(203,729)$		**-36**
8	64	-42	-37 $(187,393)$	-39 $(564,809)$	**-42** (30 hrs)	**-42** (2 hrs)
9	85	-53		-52 $(44,029)$		-51 (6 hrs)
10	100	-50			-50 (50 hrs)	-47 (9 hrs)
11	100	-48			-47	-47 (3 hrs)

Table 3. Comparison of the solution quality obtained by the genetic algorithm (GA), the evolutionary Monte Carlo (EMC), Multi-Self-Overlap Ensemble (MSOE) and new ACO. For GA and EMC, the reported energy values are the lowest among five independent runs, and the values shown in parentheses are the numbers of valid conformations scanned before the lowest energy values were found. Gaps in the table indicate that particular method has not been tested on the respective instance. The CPU times on a 500 MHz CPU are reported in parenthesis for MSOE (where available) and CPU times on a 1 GHz CPU are reported in parenthesis for the new ACO. Energies indicated in bold are currently best known values.

information, including upper and lower thresholds on weights and statistical averages, are kept between tries. Although the PERM algorithm is randomised, as seen from empirical observations the time it takes to find the first optimal conformation has very low variation. The only fair comparison of the new ACO and PERM is by considering time it takes to reach first optimum. As can be seen from Table 4, our improved ACO algorithm requires less CPU time on average for finding best known conformations for Sequences 5 (slightly) and 8 (significantly); but PERM performs better for Sequences 6 and Sequence 7 (significantly). Sequence 8 has a very symmetrical optimal conformation that, as argued in [9], would be difficult to find for any chain growth algorithm; our ACO algorithm is able to handle it quite well, since a number of ants folding from different starting points can produce good folding motives with respect to various starting folding points. For the longest sequences (85 and 100 amino acids), our algorithm finds high quality configurations, but does not reach the solution quality obtained by PERM.

5 Conclusions and Future Work

In this paper we introduced an improved ACO algorithm for the 2D HP Protein Folding Problem that has shown promising performance for an extremely simplified but widely studied and computationally hard protein structure prediction problem. An empirical study of our algorithm demonstrated the effectiveness of the improved ACO approach for solving this problem and highlighted the impact of its new features, including long range moves, improving ants, and selective local search. Long range moves in combination with the non-greedy local search performed by improving ants allows for the re-

Instances			*New ACO*		*PERM*		
No	$Length$	E^*	$\hat{E}$	$t_{avg}[CPUsec]$	$\hat{E}$	$t_{first}[CPUsec]$	$t_{avg}[CPUsec]$
1	20	-9	**-9**	3.33	**-9**	0.01	0.01
2	20	-9	**-9**	2.52	**-9**	0.02	0.02
3	25	-8	**-8**	10.62	**-8**	80.01	16.29
4	36	-14	**-14**	11.81	**-14**	0.05	0.21
5	48	-23	**-23**	405.79	**-23**	1,762.69	881.67
6	50	-21	**-21**	4,952.92	**-21**	0.48	1.61
7	60	-36	**-36**	62,471.24	**-36**	0.52	4.43
8	64	-42	**-42**	5,844.93	-38*	(6.07)	(7.90)
9	85	-53	-51	(21,901.34)	**-53**	31.95	11.74
10	100	-50	-47	(29,707.22)	**-50**	19,962	14,432.10
11	100	-48	-47	(10,835.51)	**-48**	152.71	144.41

Table 4. Comparison of the improved ACO algorithm and PERM, where $\hat{E}$ is the best solution quality over all runs, t_{avg} is the average run-time required by the algorithm to reach energy $\hat{E}$ on 1GHz CPU machine (both for ACO and PERM), t_{first} is the time required to reach energy $\hat{E}$ in the first run (reported for PERM since only the first run is independent of the others and can be compared with ACO). Measurements are based on $100 - 500$ tries for ACO algorithm, PERM reports $20 - 200$ tries. Energies indicated in bold are currently best-known values.
* After running PERM for 2 days wall clock time, -38 was the best energy reached.

laxation of compact conformations, which helps the search to escape from local optima encountered by greedy local search. Selective local search reduces the time complexity of the local search phase by performing this time critical operation only on promising, low energy conformations (which provide the best starting points for the optimisation *via* local search).

In general, we expect that the improvements introduced in this work for an ACO algorithm for the 2D HP Protein Folding Problem can be utilised for solving more traditional artificial intelligence problems (such as constraint satisfaction problems [17]). For example, the use of improving ants provides a general means of intensifying the search around high quality solutions, while long range moves in local search can be useful for escaping from local optima by considering higher order neighbourhoods relevant to the particular problem.

There are many directions for future research on ACO algorithms for protein folding problems. It might be fruitful to consider ACO approaches based on more complex solution components than the simple local structure motifs used here. Furthermore, separate pheromone matrices could be used for independently reinforcing secondary and tertiary interactions. Finally, it would be interesting to develop and study ACO algorithms for other types of protein folding problems, such as the 3-dimensional HP model [4].

Overall, we strongly believe that ACO algorithms offer considerable potential for solving protein structure prediction problems robustly and efficiently and that further work in this area should be undertaken.

Acknowledgements. This work has been supported by an NSERC Postgraduate Scholarship (PGS-A) to AS and by HH's NSERC Individual Research Grant #238788. We thank Peter Grassberger for kindly providing us with his PERM implementation.

References

1. Bastolla U., H. Frauenkron, E. Gestner, P. Grassberger, and W. Nadler. *Testing a new Monte Carlo algorithm for the protein folding problem.* Proteins: Structure, Function & Genetics 32 (1): 52-66, 1998.
2. Beutler T., and K. Dill. *A fast conformational search strategy for finding low energy structures of model proteins.* Protein Science 5: 2037–2043, 1996.
3. Chikenji G., M. Kikuchi, and Y. Iba. *Multi-Self-Overlap Ensemble for protein folding: ground state search and thermodynamics.* Phys. Rev. Let. 83(9): 1886–1889, 1999.
4. Dill, K. A., S. Bornberg, K. Yue, K. Fiebig, D. Yee, P. Thomas, and H. Chan. *Principles of protein folding - a perspective from simple exact models.* Protein Science, 4: 561-602, 1995.
5. Dorigo, M., V. Maniezzo, and A. Colorni. Positive feedback as a search strategy. Technical Report 91–016, Dip. Elettronica, Politecnico di Milano, Italy, 1991.
6. Dorigo, M. and G. Di Caro. The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, eds., *New Ideas in Optimization,* pp. 11–32. McGraw-Hill, 1999.
7. Dorigo, M., G. Di Caro and L.M. Gambardella. *Ant Algorithms for Discrete Optimization.* Artificial Life 5(2): 137–172, 1999.
8. Hoos, H.H., and T. Stützle. *On the empirical evaluation of Las Vegas algorithms.* Proc. of UAI-98, Morgan Kaufmann Publishers, 1998.
9. Hsu, H.P., V. Mehra, W. Nadler, and P. Grassberger. *Growth Algorithm for Lattice Heteropolymers at Low Temperatures.* e-print cond-mat/0208042, 2002.
10. Konig, R., and T. Dandekar. *Improving genetic algorithms for protein folding simulations by systematic crossover.* BioSystems 50: 17–25, 1999.
11. Krasnogor, N., W.E. Hart. J. Smith, and D.A. Pelta. *Protein structure prediction with evolutionary algorithms.* Proceedings of the Genetic & Evolut. Comput. Conf., 1999.
12. Lau, K.F., and K.A. Dill. *A lattice statistical mechanics model of the conformation and sequence space of proteins.* Macromolecules 22: 3986–3997, 1989.
13. Liang F., and W.H. Wong. *Evolutionary Monte Carlo for protein folding simulations.* J. Chem. Phys. 115 (7): 3374–3380, 2001.
14. Ramakrishnan R., B. Ramachandran, and J.F. Pekny. *A dynamic Monte Carlo algorithm for exploration of dense conformational spaces in heteropolymers.* J. Chem. Phys. 106 (6): 2418–2424,1997.
15. Richards, F. M. *Areas, volumes, packing, and protein structures.* Annu. Rev. Biophys. Bioeng. 6: 151–176, 1977.
16. Rose, G. D. *Hierarchic organization of domains in globular proteins.* J. Mol. Biol. 134: 447–470, 1979.
17. Solnon, C. *Ants Can Solve Constraint Satisfaction Problems.* IEEE Transactions on Evolut. Comput., 6 (4): 347-357, August 2002.
18. Shmygelska, A., R. Hernandez and H.H. Hoos. *An Ant Colony Algorithm for the 2D HP Protein Folding Problem.* In Proc. of ANTS 2002, LNCS, Vol. 2463,p. 40.
19. Unger, R., and J. Moult. *A genetic algorithm for three dimensional protein folding simulations.* In Proc. 5^{th} Intl. Conf. on Genetic Algorithms, pp. 581–588, 1993.
20. Unger, R., and J. Moult. *Genetic algorithms for protein folding simulations.* J. of Mol. Biol. 231 (1): 75–81, 1993.