Combinatorial Optimization Using Electro-Optical Vector by Matrix Multiplication Architecture

Dan E. Tamir[1], Natan T. Shaked[2], Wilhelmus J. Geerts[3], and Shlomi Dolev[4]

[1] Department of Computer Science, Texas State University, San Marcos, Texas 78666, USA
dt19@txstate.edu
[2] Department of Electrical and Computer Engineering, Ben-Gurion University of the Negev, P.O. Box 653, Beer-Sheva 84105, Israel
natis@ee.bgu.ac.il
[3] Department of Physics, Texas State University, San Marcos, Texas 78666, USA
wg06@txstate.edu
[4] Department of Computer Science, Ben-Gurion University of the Negev, P.O. Box 653, Beer-Sheva 84105, Israel
dolev@cs.bgu.ac.il

Abstract. A new state space representation of a class of combinatorial optimization problems is introduced. The representation enables efficient implementation of exhaustive search for an optimal solution in bounded NP complete problems such as the traveling salesman problem (TSP) with a relatively small number of cities. Furthermore, it facilitates effective heuristic search for sub optimal solutions for problems with large number of cities. This paper surveys structures for representing solutions to the TSP and the use of these structures in iterative hill climbing (ITHC) and genetic algorithms (GA). The mapping of these structures along with respective operators to a newly proposed electro-optical vector by matrix multiplication (VMM) architecture is detailed. In addition, time space tradeoffs related to using a record keeping mechanism for storing intermediate solutions are presented and the effect of record keeping on the performance of these heuristics in the new architecture is evaluated. Results of running these algorithms on sequential architecture as well as a simulation-based estimation of the speedup obtained are supplied. The results show that the VMM architecture can speedup various variants of the TSP algorithm by a factor of 30x to 50x.

Keywords: Optical Computing, Parallel Processing, Combinatorial Optimization, The Traveling Salesman Problem, Heuristic Search, Hill Climbing, Genetic Algorithms.

1 Introduction

This paper presents a new weight incidence representation of Hamiltonian cycles along with a set of operations on this representation including mutations and crossovers. The proposed representation facilitates efficient mapping of heuristic search techniques onto a new electro optical matrix by vector multiplication (VMM) architecture and enables fast search for a sub optimal solution to the TSP with a relatively large

S. Dolev and M. Oltean (Eds.): OSC 2009, LNCS 5882, pp. 130–143, 2009.

number of cities. In addition, the paper outlines efficient mapping of exhaustive search for solutions of TSP with a small number of cities onto the VMM.

The VMM consists of an electrical unit and an optical unit [1] and serves as a co-processor to a controller. Under the proposed implementation the controller is working in parallel with the optical unit in a pipeline mode. The controller prepares 256 vectors which constitute a specific representation of TSP solutions according to a given algorithm and sends them in real time to the VMM. The fact that the controller can prepare the vectors in real time is due to the novel representation of TSP solutions and novel representations of basic operations on TSP solutions. Through the multiplication of a matrix containing solutions and a vector of weights, the optical unit computes the cost of 256 Hamiltonian tours in parallel at a rate 125 million cost evaluations per second. The controller/VMM system can complete 125 million iterations of an algorithm such as hill climbing in one second thereby achieving a speedup of 30x to 50x in various variants of the algorithms. In the current architecture, the role of the optical unit is solely to compute the cost of each solution. In the future, we plan to explore porting additional parts of the search algorithm to the optical unit.

Many combinatorial optimization problems require an exponential amount of resources (time and / or space) with respect to a given parameter of the problem and necessitate NP complete or intractable procedures for finding an optimal solution [2]. A common approach for solving such problems is to define a search space that is an enumeration of potential solutions to the problem or steps toward the solution, and perform heuristic search in that space [3]. A multitude of heuristic search and combinatorial optimization methods is under extensive research. The list of these techniques includes the A*, Hill Climbing, iterative hill climbing (ITHC), genetic algorithms (GA), tabu search, swarm, and simulated annealing [3-10].

The traveling salesman problem is one of the most commonly addressed combinatorial optimization problems and is chosen as the platform of study for this research. In the context of this research, the TSP is stated in the following way: "Given a complete weighted undirected graph, find the minimal Hamiltonian cycle of the graph [11,12]." As a consequence of assuming undirected graph this paper is only concerned with the symmetric TSP, similar results are expected for the asymmetric problem.

Because of the complexity of the TSP, exhaustive solutions are limited to problems with a small number of cities and heuristic search might require a significant amount of computing resources before settling on an acceptable solution. As a consequence of the inherent complexity, multitude of algorithms, heuristics, architectures, and data structures for representing the problem as well as parallel processing procedures for the solution of the TSP have been explored [13-18].

Due to the potential for achieving high speed of computation, optical computing may be an interesting platform for exploring approaches for exhaustive solution for relatively small, bounded, problems and heuristic solutions of large problems. Shaked et al., have developed a new edge-weight based representation for the TSP and certain other NP complete problems [1,11,12]. This representation enables computing the cost of a TSP solution through a dot product operation. Shaked has mapped this representation to an optical vector by matrix multiplication system where TSP solutions are represented in a transparency matrix and the dot product operation is implemented using an optical vector by matrix multiplication [19]. In addition, Shaked has demonstrated an efficient method for electrical and optical construction of the matrix [20].

The papers by Shaked address exhaustive solution for bounded NP complete problems. One advantage of the approach presented in the paper is that the same matrix can be used for solving any problem with up to N cities and the only element that changes from one instance of a traveling salesman problem to another instance is the weight vector.

Only a few other papers address optical solution to the TSP. Haist et al. presents an optical TSP system that provides a polynomial-time solution for the problem yet requires an exponential amount of photons [21]. As a result, his system is also limited for an exhaustive solution of relatively small instances of the TSP. Collings has demonstrated the use of optical hardware to find local optimal solutions to the TSP using a Hopfield neural network based heuristic [22]. Additional papers on optical systems addressing the TSP and related problems are listed in [23-27].

Recently, Tamir et al., have proposed a new fast electro-optical architecture for vector by matrix multiplication [1]. In the proposed architecture, the VMM consists of a matrix of 256x256x8 bits which is multiplied by a vector of 256x8 bits. The proposed VMM is capable of performing 16 Tera integer operations per second and completes the multiplication of a 256x256x8 matrix by a 256x8 vector at a rate of 125 million vector-by-matrix multiplications per second. The paper shows how to map the solution of bounded NP complete problems developed by Shaked et al., into the new proposed VMM. Using the representation proposed by Shaked, the architecture can check 125 million solutions of up to 23 cities in 1 second. In this paper we adopt a new representation for Hamiltonian cycles. Using this representation the VMM, working along with a controller, can efficiently handle heuristic solutions to problems with thousands of cities.

2 Strategies for Solving the TSP

2.1 Exhaustive Search in the TSP Search Space

Given a representation of a solution, exhaustive search enumerates the entire set of solutions and evaluates the cost of each solution. The best cycle is retained. The complete, undirected, weighted graph with N vertices (C_N) contains $\frac{(N-1)!}{2}$ solutions. Hence, enumerating all of these solutions is of complexity of the order of $O(N^N)$. The complexity is generally related to time complexity. Nevertheless, certain TSP solution strategies trade time complexity by space complexity [6], number of constraints [28], or number of photons required to complete the computation on an optical computing device [21]. Some papers detail methods for efficient implementation of exhaustive search. Nevertheless, due to the high growth rate of the complexity function, exhaustive search is only practical for problems with a relatively small number of cities.

2.2 Heuristic Search Procedures

Despite the reduction in problem size, heuristic search algorithms are often very complex and require careful utilization of computer resources such as processing power and storage space. Many heuristic search algorithms are designed for optimal utilization of processing power in order to reduce processing time and do not fully exploit

the storage space or the trade-offs between time and space complexity. Several researchers have studied time/space trade-offs [6,29,30]. A related problem is the problem of anytime and any space algorithm [29,30]. In parallel to the research on mapping heuristic search to the VMM we are performing a research on efficient utilization of time and space resources for addressing NP-complete problems. For this end, we propose novel methods of record keeping using a cache for tracking solutions explored by the heuristic. This method enables an anytime / any space solution to the TSP. The current paper introduces the concept of record keeping along with the mapping of the problem to a parallel execution on the VMM thus enabling a significant speedup over existing approaches.

2.2.1 Hill Climbing and Iterative Hill Climbing

Hill climbing is a steepest descent greedy search algorithm [3-6,31]. There are two parts to the hill climbing algorithm. First, a valid solution, called an initial configuration, is generated. Next, the hill climbing algorithm attempts to improve the current solution by making local changes to the configuration and taking the best new solution that exists in the local search space. Each improvement is referred to as a step. This improvement process continues until the hill climbing algorithm can no longer find a better solution, at which point the procedure returns the last solution found. Iterative Hill Climbing (ITHC) is a variation of hill climbing that addresses the problem of getting trapped at a local minimum. In ITHC, once a local minimum is found, another initial configuration is generated from the global search space and the climbing process restarts. This process, of generating initial configurations and improving them, continues iteratively until the global search space is exhausted or a desired number of iterations are completed.

2.2.2 Genetic Algorithm Based Search

Genetic Algorithms are based on the principle of natural selection in which solutions are represented by chromosomes [7]. Each chromosome contains a series of characteristics which encode the solution at a particular point in a solution space, without ambiguity. For example, in combinatorial optimization problems, such as feature selection, solution chromosomes are represented by bit strings. A *1* bit represents a feature selected, and a *0* bit represents a feature which is not selected. New generations of solutions are produced by elimination of low quality solution chromosomes and addition of solutions through chromosome crossovers and mutations to produce children solutions. Crossovers usually consist of swapping of chromosome bits from two good quality chromosomes, while mutations can be random changes to the chromosomes of individual solutions. Over iterations these processes gives rise to solutions which, on average, have a higher fitness than the previous generation [7].

2.3 Representing an Hamiltonian Cycle

The literature on GA for the TSP contains numerous ways for representing Hamiltonian cycle [32,33]. Laranga et al, describe several representation methods including path based, binary, adjacency, ordinal, and matrix, methods, other representations include vector based methods [32]. The path based method is the most natural way to represent a Hamiltonian cycle. It is a vertex based representation obtained by listing

the cities visited in the order of visitation. So the cycle $1-2-3-4-5-6-1$ is represented as the sequence $\{1,2,3,4,5,6\}$. The problem with this and most other representations is that the result of crossover is not necessarily a valid Hamiltonian cycle. Hence, these representations do not enable straight forward implementation of classical crossover.

The path can be represented in an adjacency vector structure where element i of the vector contains the value j if vertex j is located in place i of the cycle. Several matrix representations have been considered. The first representation is a vertex based representation where a $N \times N$ matrix is used for a N vertex graph. The element (i,j) of the matrix contains 1 if and only if the vertex i is traversed before the vertex j. Another matrix representation includes 1 in place (i,j) if, and only if, in the vertex j is explored immediately after vertex i [32].

We consider three additional vector representations. The first representation is due to Shaked [19]. This is a binary edge based vector where 1 in place k of the vector denotes that edge k is traversed by the Hamiltonian cycle and the edges are enumerated in a consistent way. For example for the graph C_5 the edges can be enumerated in the following order $[e_{1,2}, e_{1,3}, e_{1,4}, e_{1,5}, e_{2,3}, e_{2,4}, e_{2,5}, e_{3,4}, e_{3,5}, e_{4,5}]$ In this example and under this representation, the vector $B = [1,0,0,1,1,0,0,1,0,1]$ denotes that edges $e_{1,2}$, $e_{1,5}$, $e_{2,3}$, $e_{3,4}$, and $e_{4,5}$ are traversed. Hence, the vector B represents the cycle $1-2-3-4-5-1$. We refer to this vector as the "binary incident vector." Note that due to the symmetry the edge $e_{1,5}$ stands also for the edge $e_{5,1}$. Shaked uses another vector which is an edge-weight based vector. The compatible edge-weight vector for C_5 is:

$$W = [w_{1,2}, w_{1,3}, w_{1,4}, w_{1,5}, w_{2,3}, w_{2,4}, w_{2,5}, w_{3,4}, w_{3,4}, w_{4,5}].$$

The novelty of this representation is that it enables calculating the cost (or length) of a specific cycle represented by a binary incidence vector B through the dot uct $W.B$.

Tamir et al. presented an electro-optical unit capable of fast vector by matrix multiplication (VMM). For this VMM we are proposing a new vector representation where the weight vector contains the weights of the edges traversed in the order of traversal. So the cycle $1-2-3-4-5-1$ is represented by the vector $W = [w_{1,2}, w_{2,3}, w_{3,4}, w_{4,5}, w_{1,5}]$. We refer to this vector as the weight incidence vector of the cycle $1-2-3-4-5-1$. In this case a dot product between the vector W and the vector $\bar{1} = [1,1,1,1,1]$ yields the cost of a cycle. Note that the weight incidence vector described above can be inferred from the respective cycle. A cycle however is not always uniquely defined by the weight incidence vector. In some of our implementations, a pair of vectors is maintained, the adjacency vector (e.g., $[1,2,3,4,5]$) and the incidence weight vector ($[w_{1,2}, w_{2,3}, w_{3,4}, w_{4,5}, w_{1,5}]$).

2.4 Operations on Hamiltonian Cycle Representations

We divide the operations on Hamiltonian cycles into two types of operations: mutations and crossovers. A mutation on a representation of a cycle alters the representation in a given way. A crossover uses the representation of two cycles to generate a new cycle. In classical genetic algorithms a chromosome is a binary string that

represents a solution. Chromosome mutation is implemented by a permutation on the binary string. In the case of TSP, mutations are a bit more involved. First, some of the representations are not binary. Second, for a non-complete graph an arbitrary permutation of a Hamiltonian cycle representation does not necessary yield an Hamiltonian cycle. Hence, in general, a mutation in the TSP domain has two stages, permutation of the representation and if needed repair of the representation or regeneration of a valid Hamiltonian cycle.

The 2-Opt operation is commonly used in heuristic TSP solution and as a GA mutation. The 2-Opt algorithm removes 2 edges from a Hamiltonian cycle and replaces them with 2 edges whose sum is less than the sum of the edges that were removed, while maintaining the Hamiltonian property of the new cycle [11,12]. This operation is referred to as a "butterfly." Other mutations include: displacement mutation, exchange mutation, insertion mutation, scramble, and random permutation [32,33].

The classical genetic algorithm of crossover between chromosomes takes two chromosomes split each chromosome into two or more parts and then combines different parts to create one or more chromosomes. Again, in the case of TSP, an arbitrary crossover of the representation of two Hamiltonian cycles does not necessary yield Hamiltonian cycles. Thus, a general TSP crossover involves a stage that resembles classical crossover followed by repair and / or regeneration of a cycle. Some of the commonly used crossover operations include: partially mapped crossover (PMX), ordered crossover (OX), position based crossover, heuristic crossover, neighborhood relationship crossover, meta ordering crossover, and self crossover. A comprehensive list of these and additional mutations, crossovers, and permutations can be found in [32,33].

3 The VMM Architecture

The VMM can serve as a co-processor attached to a DSP or a dedicated CPU, referred to as the controller. A high level schema of the VMM is included in Fig. 1. As the figure shows, the VMM is composed of two main components: the optical unit and the electrical driver.

3.1 The VMM Optical Unit

One of the configurations that can be used to implement the optical component of the VMM is based on the Stanford multiplier principle and illustrated in Fig. 2 [1]. As shown in this figure, the input vector of the VMM is represented by a set of light sources, the matrix of the VMM is represented by a slide mask or a real-time SLM and the output (multiplication-product) vector of the VMM is represented by a set of sensitive detectors. The light from each of the sources is spread vertically so that it illuminates a single column of the matrix. Next, each row in the matrix is summed onto a single detector in the detector array. This VMM configuration can be implemented through several optical techniques. One of these techniques uses two sets of lenses each of which contains a cylindrical lens and a spherical lens with focal lengths of f. A single set of lenses has an equivalent focal length of $f/2$ in the vertical/horizontal direction and f in the other direction. As shown in Fig. 2, the first set of

lenses is positioned between the input vector (represented by the light sources) and the matrix (represented by the SLM). This set of lenses is positioned so that the light coming from each of the sources illuminates only a single column in the matrix, which means collimating the light diverging vertically from each of the sources but imaging it in the horizontal direction. The other set of lenses is positioned between the matrix and the output vector (represented by the sensitive detectors) and is rotated by 90 degrees compared to the first set of lenses (see Fig. 2). Therefore, these lenses image a row in the matrix onto a vertical position of a single detector in the detector array and spread the light from a single column of the matrix. Other papers proposed additional versions and applications of the optical VMM. Comprehensive literature reviews in this subject can be found in references [1,34].

3.2 The VMM Electrical Driver

The electrical driver is comprised of *256* single electrical driver (SED) units and has two types of inputs: a *1×256×8* bit input vector *A*, which is the VMM input vector (that is, the VCSEL source array driving signal), and a set of *256* vector inputs B_0 to B_{255} (total of *256×256×8* bit). The output of the VMM is a *1×256×20* bit vector *C*. The VMM output vector C, is an aggregation of the set of scalar outputs (C_0 to C_{255}), where the output C_j emerging from the j^{th} SED unit is a single *20*-bit bus which is generated by the output detector array. Within each SED, the input vector B_j can be stored in an internal dual-port modular memory buffer/shifter before it is directed to the SLM. The output C_j is directed to the external output. The electro-optical configuration supports a dot-product operation between the input row vector A (being converted into light by the VCSELs) and one column of the entire SLM matrix. Each SED unit performs one vector dot-product operation per cycle of *8* ns (the reciprocal of *125* MHz). Combined together, the *256* units perform a vector (*1×256×8* bit) by matrix (*256×256×8* bit) multiplication operation at a rate of *125* MH.

The proposed system can perform multiplication of a *1×256×8* bit vector by a *256×256×8* bit matrix at a rate of *125* million vector-by-matrix multiplications per second. If the matrix and vector are smaller than *256×256* and *1×256*, respectively, then it may be possible to achieve the same rate with multiple small matrices/vectors. If the matrix and / or the vector have more than *256×256* (*256×1*) elements, then there is an overhead related to decomposing the matrix into sub matrices. In most of the cases, this overhead is negligible.

4 VMM Implementation of Hill Climbing and Genetic Algorithms in the TSP Search Space

The VMM is assumed to be a co-processor attached to a controller. Our current proposal and simulations assume that the controller prepares vectors according to the specific algorithm and sends them to the VMM where the fast matrix multiplication yields the cost of 256 vectors per cycle. In future implementations we will consider using an FPGA, ASIC, or other types of dedicated hardware to implement an electrical unit that is designed for the TSP algorithm. This unit will work directly with the SLM matrix and will supply a direct link for exchanging information between the

electrical unit and the optical unit. In addition, we consider implementing other parts of the algorithm such as identifying the K best solutions on the optical unit. We have implemented a simulation of the ITHC algorithm on the VMM. The simulation of the genetic algorithm implementation on the VMM is currently in advanced stages. We report on the actual ITHC simulations in this section.

4.1 VMM Implementation of Iterative Hill Climbing in the TSP Search Space

The ITHC in the TSP search space procedure is implemented through iterations on two execution phases: tour construction and tour improvement. Following extensive experimentation with tour construction algorithms such as the random restart and the greedy randomized adaptive search procedures (GRASP) [6,35], we adopted a deterministic approach which employs a greedy enumeration (GE) of spanning paths according to their weight and converts them into spanning cycles by adding an edge connecting the first and last vertex [6].

The actual implementation of the ITHC using the VMM is done in the following way: First, the controller is generating the initial starting configuration using the modified Kruskal's algorithm [11]. This is called the initial reference cycle. Reference cycles are represented using the adjacency vector representation as well as the weight incidence vector. All the other cycles are represented only by their weight incidence vector. Next, the controller is generating all the 2-Opt butterflies that evolve from the reference cycle. Each butterfly is a permutation which changes the location of two vertices in the reference cycle or a swap of two pairs of edges in the weight incidence vector. Hence, a butterfly implies removing two weights from the reference weight incidence vector and adding two weights to the vector. As the controller prepares these

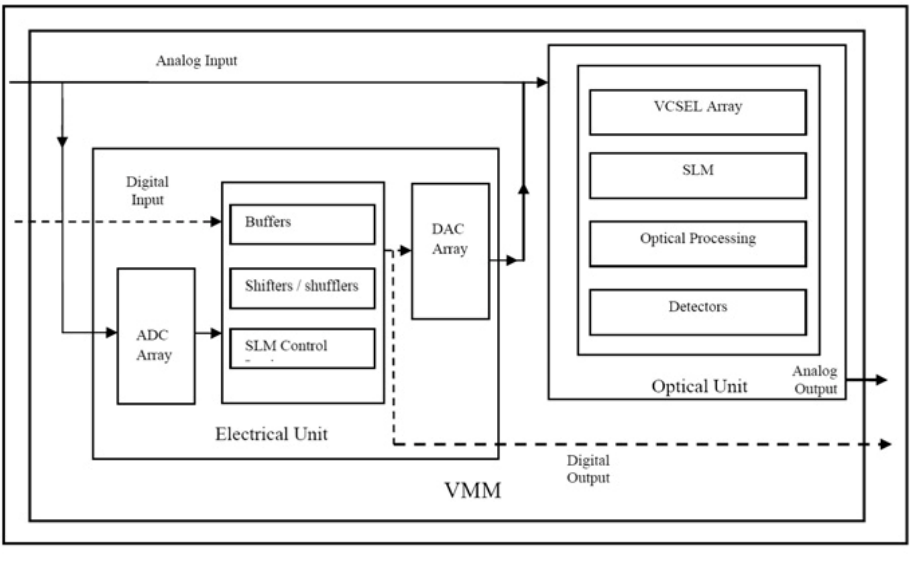

Fig. 1. The electro optical computing system

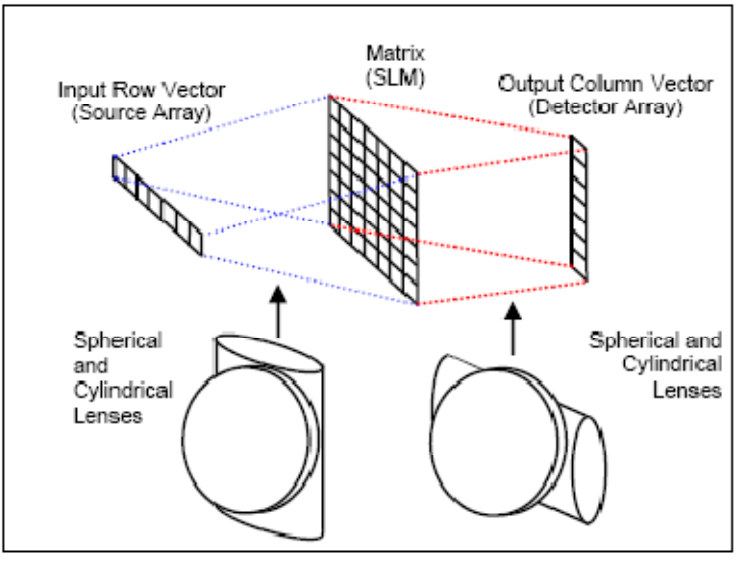

Fig. 2. The VMM optical unit

vectors, they are sent to the VMM. Since the VMM cycle is 8 ns the controller can prepare 256 vectors, and send them to the VMM, in real time. The VMM computes the cost of 256 vectors at a time and returns the results to the controller which retains the best result. After generating all the possible butterflies the best cycle is selected and becomes the reference cycle. This completes the first step of hill climbing. The controller verifies that the current step is not in the cache, places the current step in the cache, prepares the next set of weight incidence vectors, and sends them in real time to the VMM. Again, the best cycle is retained. It becomes the reference cycle and a new step is created. This process is repeated until a restart condition is met. That is, either a local optimum has been achieved or a hit on the cache, denoting that this step has already been explored, occurs. At this point the controller implements the greedy enumeration procedure to generate a new initial reference cycles and a new climber is spawned to climb from this point [6]. The process of generating new climbers repeats until a time limit or a limit on the number of climbers is reached.

4.2 VMM Implementation of Genetic Algorithm Based Search

As in the case of iterative hill climbing the role of the controller is to prepare vectors according to the algorithm and send them for cost evaluation to the VMM. We use the incidence edge-weight vector as the primary representation for chromosomes. Nevertheless, a crossover might require maintaining the adjacency vector. A mutation is a 2-Opt operation as described in the previous section. We are experimenting with several versions of crossovers including PMX and OX [32], in addition, we have developed a new and novel crossover operation which is efficient in computation time and contribution to solution quality. Moreover it has a good fit with the electro-optical architecture, thus it is efficiently implemented on the VMM. The new crossover

operator is an extension of an operator proposed in [13]. It involves the following operations: First, the cycle is split into two distinct paths at an arbitrary point. Each of the paths is going through two different mutations and then the complementary paths are recombined to generate new cycles. The advantage of this method is that it does not require a repair operation after the recombination since the parts of the path that are recombined are mutually exclusive. To further explain, let C_1 be a chromosome. The path represented by C_1 is divided into two mutually exclusive paths. That is, the two paths span the graph vertices and have no common vertices. The two path are represented by the chromosomes C_{11} and C_{12}. Next C_{11} and C_{12} are going through two mutations generating C'_{11}, C''_{11}, C'_{12}, and C''_{12}. Finally, the partial paths are recombined to generate two new cycles. The two new cycles are selected arbitrarily from the four valid recombination options. For example, the two new paths can be the cycles which correspond to the recombinations $C'_1 = C'_{11}||C''_{12}$ and $C'_2 = C''_{11}||C'_{12}$ (the operator $||$ stands for concatenation of one chromosome in the internal point of another). Since the two parts of the chromosomes are mutually exclusive with respect to vertices there is no need for a repair operation.

5 Simulation Results

At this stage only the ITHC simulation has been completed. The GA based search is in advanced stages of development. With respect to ITHC we have performed experiments with randomly created graphs as well as graphs taken from the TSPLIB set of benchmarks [36]. In this section we show the results of the search as well the computation of the speedup that can be obtained by the VMM.

Figure 3 shows the result of running the ITHC with randomly created graphs and demonstrates a speedup of more than 4x, obtained with GE and caching in a problem with 40 vertices (cities). So far, we have addressed problems with randomly generated graphs of up to 100 cities and obtained a speedup of about 4x. Note that this speedup is due to the greedy enumeration and caching and is not related to the VMM speedup detailed below.

Table 1 shows the results of running the ITHC with TSPLIB benchmarks. The experiments include greedy enumeration without cache. On the other hand an "infinite" memory model is assumed. That is, all the steps of all the climbers are stored in memory. Since the number of climbers is limited (1 million in these experiments) the available memory is sufficient to store the entire set of steps.

The quality of the cycles is good. In several cases the cost of a cycle is the same as the best cost reported in TSPLIB in other cases a degradation of up to 1.5% is observed. This is tolerable given the simplicity and practicality of ITHC. In addition, real applications might include many more climbers. The dedicated memory demonstrates the any-space features of the GE algorithm [29,30]. It also demonstrates the redundancy (%Dup) or the upper limit for improvement with cache (90% in some cases).

5.1 The VMM Contribution to the Speed-Up of the ITHC Based TSP Solution

We start with an upper bound on the speedup that can be obtained using the VMM with a problem of 256 cities (which requires a vector of 256 weights), then we show

the results of simulation with 100 cities. For the speedup simulation we are making the following assumptions: 1) the controller is working at 2 GHz or more. 2) The controller bus to memory is operating at 2 GHz, and 3) The VMM is operating at a rate of 125MHz. Note that assumptions 1 and 2 are conservative. The limiting factor is the access rate to the bus and 2 GHz is above the current state of the art. Using a bus with less bandwidth will increase the VMM speedup. Using a processor with higher frequency will have a minor effect on speedup.

In the VMM system a butterfly requires 4 addition or subtraction operations and 4 accesses to memory. Hence, generating 256 butterflies takes about 3 ns. In addition

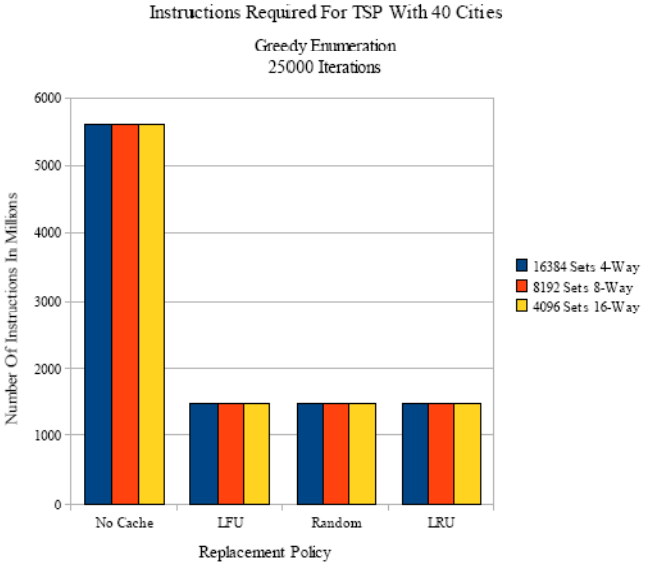

Fig. 3. Number of instructions for a 40-city problem (GE)

Table 1. TSPLIB Benchmarks

	Best	**% Dup**
ulysses16	100	85.5
ulysses22	100	87.4
gr24	100	86.4
fri26	100	86.9
bays29	100	91.1
att48	100.2	93.6
gr48 1M	101.5	90.1
eil51	100.7	93.2

the controller may need to retain a copy of the best cycle out of 256 cycles; hence 256 comparisons (subtractions) and additional 256 accesses to memory. This will take less than 2 ns. Hence, real time operation is maintained. This means that the controller can generate the vectors for the VMM in real time at a rate of operation of 125 million butterflies per second. Without the VMM, the controller has to calculate the cost of each cycle and maintain the cycle with the best cost. In the worst case, each butterfly requires copying the entire vector. Hence a butterfly involves 5 addition / subtraction operations (one more addition to adjust the final cost) and 260 accesses to memory. Additional overhead would be due to the need to maintain and update pointers to the memory. Under the current assumptions this will take about 160 ns. Thus, a speedup of 20x is obtained with the VMM. In the cases we ran with up to 100 cities, the simulation yields a speedup of about 8x which complies with the calculation for vectors of 256 elements.

5.2 The VMM Contribution to the Speed-Up of the GA Based TSP Solution

Each generation (of life of chromosomes) includes a percentage of mutations and a percentage of crossovers thus creating the next generation. Generally, the percentage of mutations is low; sometimes as low as 1%. Additionally, the size of the population within a generation is generally in the order of one hundred hence the fact that the VMM can hold 256 vectors at a time fits well with the population number. In one of our models a population size of 256 is used along with the PMX and OX crossovers. In the second model, the VMM holds up to 128 vectors representing 128 cycles each of which is divided into two mutually exclusive parts. This supports our new crossover operation. Under this model, we can efficiently handle TSP instances with up to 512 cities.

Mutations are implemented as 2-Opt and are identical to the 2-Opt operations in the ITHC. Hence the speedup of the VMM for mutations is expected to be the same. In the worst case a classical crossover operation requires recalculating the cost of a cycle hence the classical crossover may require 256 additions as well as 256 memory writes. We assume that addition operation takes half of the time of memory access operation. Hence, the speedup due to the VMM is expected to be about 30X. The new crossover requires 512 additions and memory writes. Thus, problems with a large number of cities may be subject to a speed-up of 50x or more due to the VMM.

6 Conclusions and Proposals for Further Research

We presented a new state space representation which enables efficient implementation of TSP solution algorithms including hill climbing and genetic algorithms with a large number of cities as well as exhaustive search with a relatively small number of cities. The representation uses a weight incidence vector, a data structure that holds the weights of edges that are traversed by a given Hamiltonian cycle. In addition, we have presented the mapping of exhaustive search, ITHC, and GA TSP solution algorithms to a newly proposed electro-optical VMM architecture.

We have performed a set of experiments with a sequential architecture and a simulation of the effect of a proposed parallel implementation where a controller prepares the vectors according to a given algorithm and sends them in real time to the VMM. In addition, the experiments demonstrate the effect of record keeping on the performance of these heuristics [6,37]. The simulation based estimation of the contribution of VMM to improved performance shows that the parallel electro-optical implementation has a

potential speedup of 30x to 50x over sequential implementations in various variants of the TSP algorithms.

In the future, we plan to explore porting additional parts of the search algorithm to the optical unit. In addition, we will consider using an FPGA, ASIC, or other types of dedicated hardware to implement an electrical unit that is designed for the TSP algorithm. This unit will work directly with the SLM matrix and will supply a direct link for exchanging information between the controller and the VMM. In addition, we consider implementing other parts of the algorithm such as identifying the K best solutions on the optical unit.

Acknowledgments. The last author is partially supported by the Rita Altura Trust in Computer Science.

References

1. Tamir, D.E., Shaked, N.T., Wilson, P.J., Dolev, S.: High-speed and low-power electro-optical DSP coprocessor. J. Opt. Soc. Am. A26, A11–A20 (2009)
2. Garey, M.R., Johnson, D.S.: Computers and intractability: A guide to the theory of NP-completeness. W.H. Freeman, New York (1979)
3. Pearl, J.: Heuristics: Intelligent search strategies for computer problem solving. Addison-Wesley, Reading (1984)
4. Russell, S.J., Norvig, P.: Artificial intelligence: A modern approach. Prentice-Hall, Englewood Cliffs (1995)
5. Xi, B., Liu, Z., Raghavachari, M., Xia, C., Zhang, L.: A smart hill-climbing algorithm for application server configuration. In: Proceedings of the 13th international conference on world wide web, pp. 287–296 (2004)
6. Karhi, D., Tamir, D.E.: Caching in the TSP Search Space. In: Next Generation Applied Intelligence, Tainan, Taiwan, pp. 221–230 (2009)
7. Vose, M.D.: The simple genetic algorithm: Foundations and theory. MIT Press, Cambridge (1999)
8. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Press, Dordrecht (1997)
9. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Academic Press, London (2001)
10. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220, 671–680 (1983)
11. Johnson, D.S., McGeoch, L.A.: The traveling salesman problem: A case study in local optimization. Local search in combinatorial optimization, 215–310 (1997)
12. Applegate, D.L., Bixby, R.E., Vasek, C., Cook, W.J.: The traveling salesman problem: A computational study. Princeton University Press, Princeton (2007)
13. Wang, L., Maciejewski, A.A., Siegel, H.J., Roychowdhury, V.P.: A comparative study of five parallel genetic algorithms using the traveling salesman problem. In: Proceedings of the 12th. International Parallel Processing Symposium, pp. 345–349 (1998)
14. Borovska, P.: Solving the travelling salesman problem in parallel by genetic algorithm on multicomputer cluster. In: Int. Conf. on Computer Systems and Technologies, pp. 1–6 (2006)
15. Gang, P., Iimura, I., Nakatsuru, T., Nakayama, S.: Efficiency of local genetic algorithm in parallel processing. In: Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies, pp. 620–623 (2005)
16. Langdon, W.B., Banzhaf, W.: A SIMD interpreter for genetic programming on GPU graphics cards. In: O'Neill, M., Vanneschi, L., Gustafson, S., Esparcia Alcázar, A.I., De Falco, I., Della Cioppa, A., Tarantino, E. (eds.) EuroGP 2008. LNCS, vol. 4971, pp. 73–85. Springer, Heidelberg (2008)

17. Inoue, T., Sano, M., Takahashi, Y.: Design of a processing element of a SIMD computer for genetic algorithms. In: Proceedings of the High-Performance Computing on the Information Superhighway, pp. 688–691 (1997)
18. Vega-Rodriguez, M.A., Gutierrez-Gil, R., Avila-Roman, J.M., Sanchez-Perez, J.M., Gomez-Pulido, J.A.: Genetic algorithms using parallelism and FPGAs: The TSP as case study. In: International Conference Workshops on Parallel Processing, pp. 573–579 (2005)
19. Shaked, N.T., Messika, S., Dolev, S., Rosen, J.: Optical solution for bounded NP-complete problems. Applied Optics 46(5), 711–724 (2007)
20. Shaked, N.T., Tabib, T., Simon, G., Messika, S., Dolev, S., Rosen, J.: Optical binary-matrix synthesis for solving bounded NP-complete combinatorial problems. Optical Engineering 46(10), 108201, 1–11 (2007)
21. Haist, T., Osten, W.: An Optical solution for the traveling salesman problem. Opt. Express 15, 10473–10482 (2007)
22. Collings, N., Sumi, R., Weible, K.J., Acklin, B., Xue, W.: The use of optical hardware to find good solutions of the travelling salesman problem (TSP). In: Proc. SPIE 1806, pp. 637–641 (1993)
23. Dolev, S., Fitoussi, H.: The traveling beams: Optical solutions for bounded NP-complete problems. In: Crescenzi, P., Prencipe, G., Pucci, G. (eds.) FUN 2007. LNCS, vol. 4475, pp. 120–134. Springer, Heidelberg (2007)
24. Dolev, S., Nir, Y.: Optical implementation of bounded non-deterministic Turing machines. US Patent 7, 130, 093 B2 (January 2005)
25. Anter, A., Dolev, S.: Optical solution for hard in average NP-complete instances (using exponential space for solving instances of the permanent). In: Proc. of the 12th IEEE Meeting on Optical Engineering and Science in Israel (2nd OASIS), Israel (2009)
26. Oltean, M.: Solving the Hamiltonian path problem with a light-based computer. Natural Computing 7(1), 57–70 (2008)
27. Oltean, M., Muntean, O.: Solving NP-complete problems with delayed signals: An overview of current research directions. In: Proceedings of 1st international Workshop on Optical Super Computing, pp. 115–128 (2008)
28. Gutfreund, D., Shaltiel, R., Ta-Shma, A.: If NP languages are hard on the worst-case, then it is easy to find their hard instances. Comput. Complex. 16(4), 412–441 (2007)
29. Zilberstein, S.: Using anytime algorithms in intelligent systems. AI Magazine 17(3), 73–83 (1996)
30. Ramos, T., Cozman, G.: Anytime anyspace probabilistic inference. International Journal of Approximate Reasoning 38(1), 53–80 (2005)
31. Aarts, E., Lenstra, J.: Local Search in Combinatorial Optimization. Princeton University, Princeton (2003)
32. Larrinaga, P., Kuijpers, C.M.H., Murga: Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artificial Intelligence Review, 129–170 (1999)
33. Ursache, L.: Representation models for solving TSP with genetic algorithms. In: Proceedings of CNMI, Bacău, pp. 291–298 (2007)
34. Feitelson, D.G.: Optical computing: A survey for computer scientists. MIT Press, Cambridge (1988)
35. Pitsoulis, L., Resende, M.: Greedy randomized adaptive search procedures. Journal of Global Optimization 6(2), 109–133 (1995)
36. Reinelt, G.: TSPLIB -A traveling salesman problem library. ORSA Journal on Computing 3(4), 376–384 (1991), http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/
37. Lowell, D., El Lababedi, B., Novoa, C., Tamir, D.E.: The locality of reference of genetic algorithms and probabilistic reasoning. In: The International Conference on Artificial Intelligence and Pattern Recognition, Florida, pp. 221–228 (2009)