A Hybrid Genetic Algorithm for the Travelling Salesman Problem

Xiao-Bing Hu and Ezequiel Di Paolo

Abstract Genetic Algorithms (GAs) for the Travelling Salesman Problem (TSP) are often based on permutation representations, which makes it difficult to design effective evolutionary operators without causing feasibility problems to chromosomes. This paper attempts to develop a binary representation based hybrid GA to solve the TSP. The basic idea is to design a pre-TSP problem (PTSPP), where the input is the coordinates of a point in the map of cities, and the output is a feasible route connecting all cities. An effective deterministic algorithm is developed for this PTSPP to search the local optimum starting from the coordinates of a given point. The new GA is then designed to randomly choose and evolve the coordinates of generations of points for the PTSPP, and also to find out the global optimum or sub-optima for the TSP. The preliminary experiments show the potential of the proposed hybrid GA to solve the TSP.

1 Introduction

The Travelling Salesman Problem (TSP) is often used as a benchmark NP-complete problem to test optimization algorithms and methods [1]. As large-scale parallel stochastic searching algorithms widely used in various applications [2, 3], Genetic Algorithms (GAs) also attract much attention to tackle the TSP [4–11].

However, designing GAs for the TSP is not a straightforward task. This is largely because, as is well known, GAs were originally developed for those problems whose solutions are based on value, but TSP routes are based on the order of visiting cities.

X.-B. Hu
Centre for Computational Neuroscience and Robotics, University of Sussex
xiaobing.hu@sussex.ac.uk

E.D. Paolo
Centre for Computational Neuroscience and Robotics, University of Sussex
ezequiel@sussex.ac.uk

X.-B. Hu and E.D. Paolo: *A Hybrid Genetic Algorithm for the Travelling Salesman Problem*, Studies in Computational Intelligence (SCI) **129**, 357–367 (2008)
www.springerlink.com

As a result, many permutation representations rather than the original binary representation have to be used to transfer a TSP route into a chromosome in GAs [4–11]. Even though some of them only contain binary values, e.g., the matrix representation in [7], they are actually based directly on the visiting orders or the connections between cities. The use of permutation representations always imposes special constraints that a feasible chromosome must satisfy. However, due to the stochastic nature of GAs, it is very likely to obtain unfeasible offspring from feasible parents if classic evolutionary operations are applied without taking any special measures to avoid this. To address this feasibility problem caused by permutation representation of TSP routes, many evolutionary operators totally different from the original ones have been proposed in the last few decades. For example, to apply the idea of crossover to TSP routes, many efforts have been made, including simple crossover [4], partially matched crossover [5, 7], greedy subtour crossover [6], edge assembly crossover [8], order crossover and cycle crossover [7]. Unlike the original crossover, which can be applied to a wide range of problems, these crossover operators are usually specifically designed for the TSP, and therefore have little use in other kinds of problems. Another issue is that these especially designed evolutionary operators can hardly make most of the original idea of evolution by natural selection, because to them the feasibility issue is often more important than the evolutionary principle.

The above difficulties exist in many GA applications to problems whose solutions are based on order or permutation of elements. In this paper, we attempt to test the idea of applying the original binary representation and all classic evolutionary techniques to such problems. To this end, a generic hybrid GA scheme is proposed. The TSP is especially chosen as a case study. In the hybrid GA scheme, the TSP is cast into a pre-problem whose input is the coordinates of a randomly given point in the map of cities. A deterministic procedure is proposed to map such an input to a TSP route. Then a very basic GA can be used to evolve the input to the pre-problem, in order to get a good TSP route.

2 The basic idea of a hybrid GA

As large-scale parallel stochastic search and optimization algorithms, GAs have a good potential to be applied to a wide range of optimization problems [2, 3]. The choice of representation of solutions and the design of evolutionary operators play crucial roles in a successful application of GAs. Permutation representations are often used when applying GAs to the TSP, but they make it difficult to design effective and efficient evolutionary operators. A major problem is that those evolutionary operators based on permutation representations could cause serious feasibility problems.

In this paper, we aim to test the idea of using very basic binary GAs to solve those problems which usually require permutation representations. The binary GAs

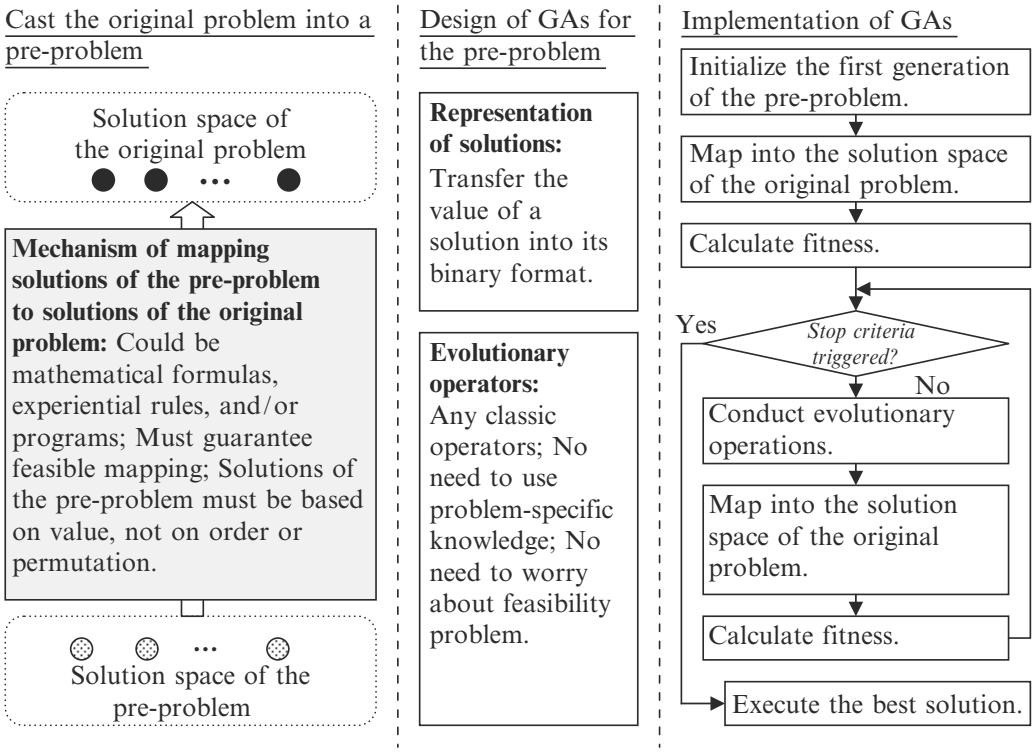

Fig. 1 Design of a hybrid GA

used must, free of feasibility problem, be compatible to all classic evolutionary operators. To this end, we propose a hybrid GA, the basic idea of which is illustrated in Fig. 1. Basic binary GAs are easy to design for those problems where the solutions are based on value, and to such problems all classic evolutionary operations, such as mutation, one-point crossover and uniform crossover, are usually applicable. However, if the solutions are based on order or permutation, such as a work schedule and a TSP route, it is very difficult, if not impossible, to design binary GAs, and even if it is possible to design binary GAs for such problems, crossover is often discarded because it is usually more distructive rather than effective. In our hybrid GA scheme, an original problem, whose solutions are based on order or permutation and therefore are unsuitable for binary representation, needs to be cast into a pre-problem whose solutions are based on value. Then we design a binary GA for this new pre-problem, and most classic GA techniques can apply straightforwardly. In the implementation of the hybrid GA, the only thing different from a classic GA is that, before the fitness of a chromosome is calculated, the represented solution to the pre-problem needs to be mapped into the associated solution to the original problem. Obviously the most important and also the most difficult step in the hybrid GA scheme is to design a proper pre-problem, which depends largely on each individual original problem. In this paper we will propose a pre-problem especially designed for the TSP, and then we report a hybrid GA which is very effective in finding quite good TSP routes on all problem scales.

3 Pre-TSP problem (PTSPP)

Consider the following pre-problem to the original TSP: in a map of cities, an input point (IP) is drawn; this point is not associated in principle with the location of any city. What we need to do is to find a closed-loop route which connects all cities nearest to the IP. Then we need to insert all the second nearest cities to the route, then the third nearest cities, and so on until all cities are included in the route once and only once. What we try to achieve is to make the final route as short as possible. Hereafter, this pre-problem is called as pre-TSP problem (PTSPP), and a deterministic method is proposed as the following to solve the PTSPP.

Suppose there are N_C cities the salesman needs to visit, and M_{Dis} is a matrix whose entry $M_{Dis}(i,j)$ records the distance between city i and city j. It is assumed that neither the cities nor the IP has the same coordinates of the origin. This assumption can be easily satisfied by moving the coordinates axes. Once an IP is given, we will determine a TSP route according to the following procedure.

Step 1: Calculate the distances between cities and the given IP. Let $V_{Dis}(i)$ be the distance between city i and the IP. Sort $V_{Dis}(i)$ in an ascending order. Let $S_{Dis}(j) = i$ mean city i has the jth shortest distance to the IP, in other words, $V_{Dis}(i)$ is the jth smallest distance to the IP.
Step 2: Calculate the angles of the vectors determined by cities and the given IP. Let $V_{Ang}(i)$ be the angle associated with city i. Sort $V_{Ang}(i)$ in an ascending order. Let $S_{Ang}(j) = i$ mean city i has the jth smallest angle, in other words, $V_{Ang}(i)$ is the jth smallest angle.
Step 3: Let R denote the route to be determined, and put city $S_{Dis}(1)$ and city $S_{Dis}(2)$ as the first two cities in the route, i.e., $R(1) = S_{Dis}(1)$ and $R(2) = S_{Dis}(2)$. Record how many cities the current R covers: $L_R = 2$. Set up a range to indicate to search how many cities with $V_{Ang}(i)$ nearest to that of the current city which needs to be inserted in the route R:

$$R_{S1} = \max(1, ceil(N_C/N_G)) \tag{1}$$

where "*ceil*" is the function that rounds a number to the nearest integer greater than that number, and N_G is an algorithm related parameter to define how many groups the N_C cities could roughly be classified into according to $V_{Ang}(i)$.
Step 4: While $L_R + 1 \leq N_C$, do

Step 4.1: Set $S_{Dis}(L_R + 1)$ as the current city that needs to be inserted to the route R. Set up a range to indicate to search how many cities which are already in the route R and have $V_{Ang}(i)$ nearest to that of the current city:

$$R_{S2} = \min(R_{S1}, ceil(L_R/2)) \tag{2}$$

Step 4.2: Find out the R_{S2} cities which are already in the route R and have $V_{Ang}(i)$ nearest to that of the current city $S_{Dis}(L_R + 1)$. For each of these cities, calculate to which side city $S_{Dis}(L_R + 1)$ should be inserted, such that the

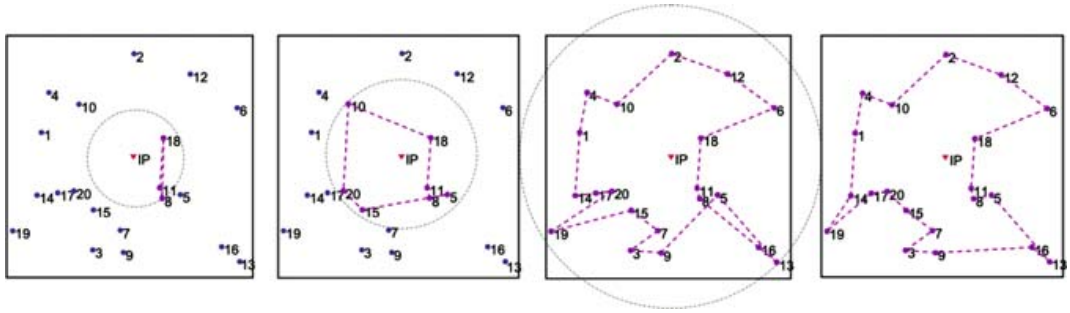

Fig. 2 An example of route searching based on the ripple-spreading process (IP=(0,0), d=789.8767)

increase of the total travelling distance of R is smaller. Compare all these R_{S2} cities to find out the final position where city $S_{Dis}(L_R+1)$ should be inserted.
Step 4.3: Insert the current city $S_{Dis}(L_R+1)$ at the final position in the route R. Let $L_R = L_R + 1$.

The above procedure can be roughly likened to throwing a stone into a pool where there stand N_C stakes randomly distributed. When the ripple spreads out from the point where the stone hits the pool, i.e., the IP, it reaches every stake sooner or later according to the distance from each stake to the hit point. Base on the order in which the ripple reaches each stake, plus referring to the angle from each stake to the hit point, we can work out a TSP route to connect the N_C stakes. Fig. 2 illustrates how the above ripple-spreading-likened procedure calculates a complete TSP route step by step. Different hit point, i.e., the IP, results in different order to reach each stake, and consequently leads to different TSP route.

Obviously this ripple-spreading procedure is deterministic rather than stochastic. It will be used to calculate fitness in our GA, and this is why we call the GA a hybrid GA.

4 Binary representation based GA for TSP

4.1 Representation of solutions

Most existing GAs for the TSP use no-binary representation of solutions. For example, a popular practice is using permutation representation, where the value of each gene in a chromosome is the serial number of a city, and the order the cities appear in the chromosome determines how the cities are connected to form a route, as illustrated in Fig. 3(d). Direct binary representation of solutions to the TSP is also possible. Sometimes the paths between cities rather than cities themselves are used to construct chromosomes. In this case, 0-1 matrix representation is used to record the connections between cities, as illustrated in Fig. 3(e), where a gene $C(i, j) = 1$ means city i and city j are connected. Figure 3(f) gives another direct binary representation

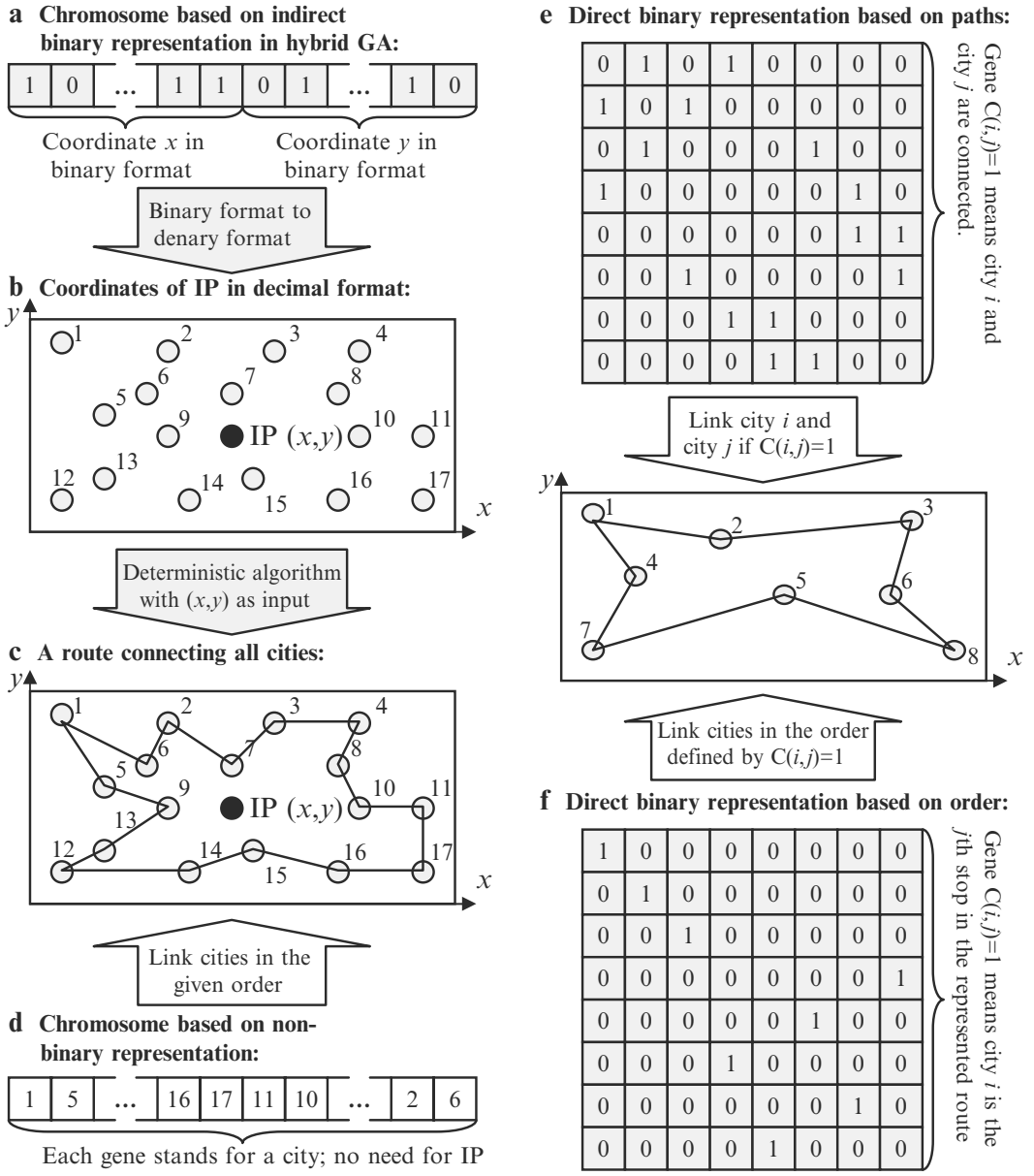

Fig. 3 Various representations of solutions to the TSP

of solutions to the TSP based on the order each city appears in a route, where the chromosome is also a 0-1 valued matrix, and a gene $C(i, j) = 1$ means city i is the jth stop in the represented route. Based on the PTSPP discussed in the previous section, we propose an indirect binary representation, as illustrated in Fig. 3(a). A chromosome of the new structure is actually the coordinates of an IP in binary format, which are input to the PTSPP. Suppose all cities are within a rectangle of size $X_R \times Y_R$, and the minimum searching steps along the two axes are X_S and Y_S. We can then calculate how many bits a binary coded coordinate x or y is composed of:

$$L_X = ceil(\log_2(X_R/X_S)) \tag{3}$$

Table 1 Features of different representations

	Meaning of a gene	Meaning of a chromosome	Size of a chromosome	Constraints for evolutionary operators
Permutation representation	$C(j) = i$ means city i is the jth stop in the route	A route with cities connected in the order given by genes	N_C numbers in denary format	$C(i) \in \{1,\ldots,N_C\}$ $C(i) \neq C(j)$ if $i \neq j$
Direct binary representation based on paths	$C(i,j) = 1$ means cities i and j are connected	Which paths between cities are chosen to form a route	N_C^2 bits	$\sum_{i=1}^{N_C} C(i,j) = 2$, $\sum_{j=1}^{N_C} C(i,j) = 2$
Direct binary representation based on order	$C(i,j) = 1$ means cities i is the jth stop in the route	Which city is given in which order in the route	N_C^2 bits	$\sum_{i=1}^{N_C} C(i,j) = 1$, $\sum_{j=1}^{N_C} C(i,j) = 1$
Indirect binary represenation	A digital bit in the binary number of coordinates of IP	The coordinates of IP in the map of cities	$(L_X + L_Y)$ bits	No constraints

$$L_Y = ceil(\log_2(Y_R/Y_S)) \tag{4}$$

where L_X and L_Y are the length of binary coded x and y, respectively.

Unlike the permutation representation and the two direct binary representations given in Fig. 3, this indirect binary representation has no direct links to the solutions to the TSP. As shown in Table 1, neither the meaning of a gene, nor the meaning of a chromosome, nor the size of a chromosome in the indirect binary representation has anything to do with the solutions to the TSP. The reason for the distinguishing features of the indirect binary representation is because, due to the introduction of the PTSPP, the space the GA needs to search is not the solution space of the TSP. As shown in Fig. 3(b), to transform a chromosome based on the indirect binary representation into a solution to the TSP, the deterministic algorithm proposed in Section 2 needs to be employed.

4.2 Evolutionary operators

A feasible solution to the TSP is a route where each of all cities to be visited is included once and only once. In the permutation representation and the direct binary representations, the chromosomes are actually different ways to describe potential routes, feasible or not. Even though all chromosomes in the initial generation are feasible, it is not an easy task to design efficient evolutionary operators which can guarantee the offspring are also feasible to the TSP. Particularly, it is very difficult, if not impossible, to design an efficient crossover operator which can automatically ensure the feasibility of offspring without imposing extra constraints. For instance, in the permutation representation, randomly swap sections between two parent chromosomes often causes the offspring chromosomes to have some cities

twice. Although unfeasible chromosomes are allowed in some GA designs in order to maintain the diversity level of chromosomes, if the evolutionary process of mixed chromosomes, i.e., both feasible and unfeasible, is not reasonably controllable or predictable, it is better to take necessary measures to avoid unfeasible chromosomes in the first place, otherwise it is very likely to end up with a GA of extremely low cost-efficiency. In the design of GAs based on permutation representation or direct binary representation, the result of directly and randomly operating on potential solutions to the TSP is usually very difficult to predict. Therefore, extra constraints and/or special deterministic knowledge need to be imposed on and/or integrated into the evolutionary process. Extra constraints are used to check and eliminate unfeasible chromosomes, while special deterministic knowledge can guide the evolutionary process to avoid generating unfeasible chromosomes. Special deterministic knowledge is actually an implicit version of extra explicit constraints. Table 1 gives some typical constraints for the permutation representation and the two direct binary representations. These constraints ensure the feasibility of chromosomes, but they also make it difficult to design effective and efficient evolutionary operators.

Thanks to the indirect binary representation, we need no constraints to guarantee the feasibility of chromosomes in our new GA for the TSP. Even if we want to limit the coordinates of IP within a certain region, say, $x \in [0, X_R]$ and $y \in [0, Y_R]$ are a must, the predetermined size of chromosomes, i.e., $L_X + L_Y$, will automatically ensure all chromosomes are feasible, let alone that there is no such a mandatory boundary for (x, y) according to the modeling of the PTSPP. Without any constraints, we have the full freedom to design evolutionary operators: mutation and crossover. The mutation operator in the new GA is to randomly reverse a gene in the chromosome: if gene $C(i)$ is chosen to mutate at a given probability, then we have

$$C(i) \rightarrow 1 - C(i). \tag{5}$$

Crossover is used to exchange sections of chromosomes. Although there has been a long debate about whether crossover is really necessary to GAs, we assume in this paper that crossover is useful by observing the natural evolutionary process of biological species. Therefore, the question left to us is how to design an effective and efficient crossover operator. In the new GA, we choose the uniform crossover operator, which is very difficult, if not impossible, to apply to the permutation representation or direct binary representation based GAs for the TSP. Uniform crossover uses two parents to produce only one offspring, and the principle is simple: the *i*th gene in the offspring inherits the *i*th gene of either parent at a 50% probability [12].

5 Preliminary experiments

In this section, we only give the results of some preliminary experiments, Fig. 4, to demonstrate the proposed hybrid GA can work quite well on all problem scales. Scalability is an advantage of the proposed hybrid GA against those based on

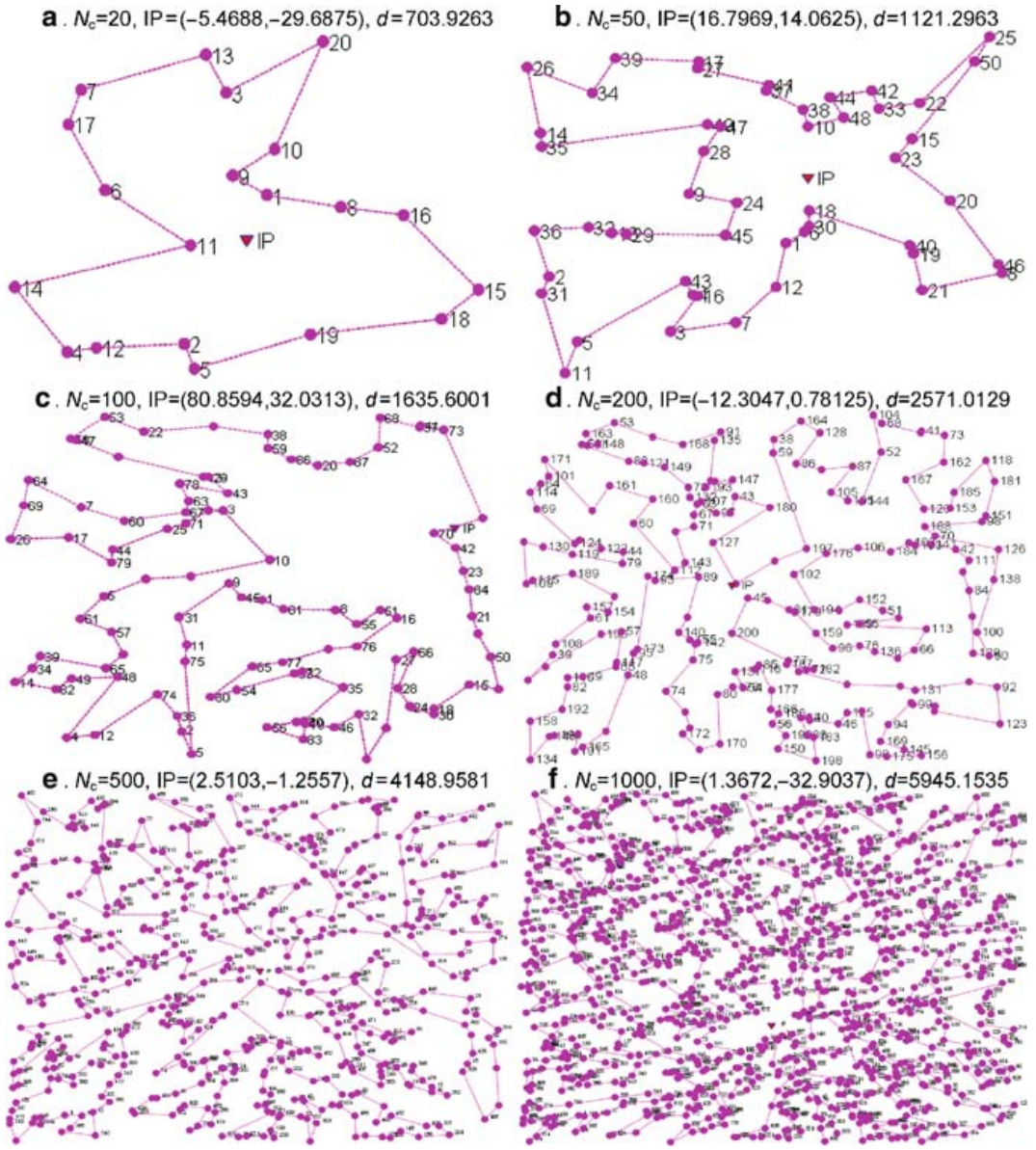

Fig. 4 Results of some preliminary experiments

permutation representations. For instance, for the case of $N_C = 1000$, a single chromosome with of the matrix representation of either Fig. 3(e) or Fig. 3(f) requires at least 1M bits memory, and then a generation of 1000 chromosomes (a larger N_C demands a huger population in order to maintain the level of solution quality) needs a memory capacity of at least 1G bits. Therefore, running GAs with such representations for the TSP with $N_C \geq 1000$ is almost a mission impossible on standard personal computers. In contrast, the proposed hybrid GA has no such memory-inefficiency problems. This is because a chromosome with the new binary representation simply records the coordinates of an IP, rather than the complex information of the associated TSP route, which is calculated online by the deterministic method with the IP as input. In other words, the proposed hybrid GA sacrifices its

computational efficiency in order to avoid memory-inefficiency problems. It should be noted that GAs with other representations also suffer from heavy computational burdens due to those constraints such as given in Table 1 for feasibility purposes.

At this stage of our study, many questions still remain unsolved and many improvements are needed. For example, does the problem casting process guarantee all optima, or at least one, of the original TSP be mapped into the solution space of the PTSPP? What is the proportion of good solutions to the original TSP being covered by the solution space of the PTSPP? How can we improve the computational efficiency of the deterministic method? Will it be better to introduce some stochastic features into the current deterministic ripple-spreading procedure? Can such stochastic features be parameterized and integrated with the IP, so that they can also evolve by the GA? Therefore, more improvements, statistical analyses and comparative experiments are being carried out.

6 Conclusions

This paper reports a binary representation based hybrid Genetic Algorithm (GA) to solve the Travelling Salesman Problem (TSP). In order to use highly efficient evolutionary operators such as uniform crossover without causing feasibility problem to chromosomes, and also to make most of problem-specific knowledge, the TSP is cast into a pre-TSP problem (PTSPP), where the input is the coordinates of a point in the map of cities, and the output is a feasible route calculated by an effective deterministic algorithm which mimic the ripple-spreading-process. Then a binary GA is used to randomly choose and evolve the coordinates of generations of points, and the optimal or sub-optimal routes are then found out during the evolutionary process. Further research includes theoretical analysis on the problem casting process and extensive comparative experiments.

Acknowledgments This work is supported by EPSRC Grant EP/C51632X/1.

References

1. Reinelt, G. (2004) TSPLIB, Travelling Salesman Problem, Universität Heidelberg, found on http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/
2. Holland, J.H. (1975) Adaptation in Natural and Artificial Systems, Ann Arbor, MI: University of Michigan Press
3. Mitchell, M., (1998) An Introduction to Genetic Algorithms, Cambridge, MA: MIT Press
4. Ingo, W. (1999) Evolutionäre Algorithmen, found on http://www.informaticadidactica.de/HyFISCH/Spitzenforschung/Wegener.htm
5. Thomas, A. (2001) Solving the Travelling Sales Man Problem using a Genetic Algorithm, found on http://www.generation5.org/content/2001/ga_tsp.asp

6. Sengoku, H. and Yoshihara, I. (1998) A Fast TSP Solver Using GA on JAVA, Hitachi Ltd & Tohoku University, found on http://www-cse.uta.edu/ %7Ecook/ai1/lectures/applets/gatsp/TSP.html
7. Jürgen, H. University of Mannheim (2002) Lecture slides for Evolutionary Algorithm, found on http://webrum.uni-mannheim.de/math/scovis/Vorlesung/EA/WS0304/EAScript3.pdf
8. Warson, J., Ross, C., Eisele, V., Denton, J., Bins, J., Guerra, C., Whitley, D., and Howe, A. (2001). The Travelling Salesman Problem, Edge Assembly Crossover, and 2-opt, Colorado University, Fort Collins
9. Jog, P., and Suh, J.Y, and Van Gucht, D. (1989). The Effects of Poplation Size, Heuristic Crossover and Local Improvement on a genetic Algorithm for the Travelling Salesman Problem, Proceedings of the 3rd International Conference on Genetic Algorithms, Indiana University, USA.
10. Julstrom, B. A. (1999). Coding TSP Tours as permutations via an insertion heuristic, Proceedings of the 1999 ACM symposium on Applied computing, St. Cloud State University, St. Cloud.
11. Whitley, D., and Strakweather, T., and Fuquay, DA. (1989). Scheduling Problems and Travelling Salesman: The Genetic Edge Recombination Operator, Proceedings of the 3rd International Conference on Genetic Algorithms, Indiana University, USA.
12. Sywerda, G. (1989) Uniform crossover in genetic algorithms, Proceedings of the 3rd International Conference on Genetic Algorithms, Indiana University, USA.